Fall 2005

Multi-agent opportunism

James H. Lawton
University of New Hampshire, Durham

Multi-Agent Opportunism

BY

James H. Lawton

B.S., Utica College of Syracuse University (1988)
M.S., Syracuse University (1993)

DISSERTATION

Submitted to the University of New Hampshire
in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

in

Computer Science

September 2005

This dissertation has been examined and approved.

[illegible]

Dissertation director, Roy M. Turner

Associate Professor of Computer Science

[illegible]

Elise H. Turner

Associate Professor of Computer Science

[illegible]

Ted M. Sparr

Professor of Computer Science

[illegible]

Philip J. Hatcher

Professor of Computer Science

Carla P. Gomes

Carla P. Gomes

Associate Professor of Computer Science,

Cornell University

7/29/2005

Date

Dedication

To Stephanie, for her unending love and support.

Acknowledgments

I would like to thank Carmel Domshlak for his important contributions to this research. Carmel was instrumental in the development of the planning and execution mechanism used in this work, as well as a patient sounding board for my many rants on agents and opportunism. Carmel's work on this project was supported by the Intelligent Information System's Institute (IISI) at Cornell University.

I would also like to thank the members of my dissertation committee: Roy Turner, Elise Turner, Ted Sparr, Phil Hatcher and Carla Gomes. You all have been very supportive. I have to give extra thanks to Carla, director of the IISI, for encouraging my collaboration with Carmel. I also owe a special debt of gratitude to Roy and Elise, for guiding me through this long journey while suffering their own life-altering challenges.

I would like to acknowledge the support of my employer, the Air Force Research Laboratory, for making this all possible. I would especially like to thank our Chief Scientist, Northrup Fowler, for never giving up on me.

Finally, I would have never made it this far without the continued support of my family. My thanks go out to my parents, Jim and Mary Ellen, my brother Perry and sister Lisa, and of course my darling wife, Stephanie.

Table of Contents

Dedication iv

Acknowledgments v

List of Tables ix

List of Figures x

Abstract xii

1 Introduction **1**

2 Example Scenarios **8**

2.1 Example Scenario 1 – Planetary Rovers 8

2.2 Example Scenario 2 – Autonomous Oceanographic Sampling Networks 12

3 Opportunism in Planning Agents **17**

3.1 Definitions 17

3.2 Single-Agent Opportunism 20

3.2.1 Active Approaches 21

3.2.2 Passive Approaches 23

3.2.3 Hybrid Approaches 27

3.3 Multi-Agent Opportunism 31

3.4 Summary 31

4 Multi-Agent Opportunism **32**

4.1 Multi-Agent Opportunities 33

4.2 Framework for Multi-Agent Opportunism 37

4.2.1 Multi-Agent Opportunity Recognition 38

4.2.2 Multi-Agent Opportunity Response 47

4.3 Summary 51

5 Planning and Execution **52**

5.1 Assumptions and Motivations 53

5.2 Abstract Model of Multi-Agent Planning and Execution 56

5.3 Basic Model for Planning and Execution 59

5.3.1 Planning 59

5.3.2 Execution 65

5.4 Discussion 80

6 Plan-Based Opportunism **82**

6.1 Model of Opportunistic Execution 82

6.2 Making POPGs More Efficient 91

6.2.1 Planning with Shortcuts 92

6.2.2 Predictive Plan Repair 96

6.2.3 Reactive Plan Repair 98

6.3 Addressing the Critical Issues 99

7 Empirical Evaluation **103**

7.1 Experimental Setup 103

7.2 Evaluation 109

7.3 Summary 122

8 Other Multi-Agent Systems **123**

8.1 Autonomous Oceanographic Sampling Networks 123

8.1.1 AOSN Planning 125

8.1.2 AOSN Coordination 129

8.2 Aircraft Maintenance Information System 132

8.3 Summary 141

9 Conclusion **143**

9.1 Framework Summary 143

9.2 Related Work 147

9.3 Future Work 155

9.3.1 POPG Complexity 155

9.3.2 Active Goals 157

9.3.3 Cue Sharing 159

9.3.4 Other Coordination Mechanisms 161

9.4 Final Remarks 165

Bibliography **168**

List of Tables

7.1 Statistical Results for Rovers experiments 115

List of Figures

1-1 Decision Framework for Single-Agent Opportunism 3

2-1 Workspace partitioning for `Rovers` example MAS. 9

2-2 AOSN-Layout (Adapted from [Turner et al., 1997]). 13

2-3 AOSN Organization. 15

4-1 Workspace partitioning for `Rovers` MAS. (Repeated from Fig. 2-1) . 34

4-2 Decision Framework for Multi-Agent Opportunism 37

5-1 Operators for `Rovers` example. 61

5-2 Partial order plan for the `Rovers` example. 62

5-3 Solved Planning Graph for the `Rovers` example. 63

5-4 Partial order plan graph (POPG) for the `Rovers` example. 64

5-5 Procedure for updating POPG $\mathcal{P}$ after performing action A. 68

5-6 $\mathcal{P}_1$: POPG for the `Rovers` example (repeated from Fig. 5-4). 69

5-7 $\mathcal{P}_2$: Result of calling Refine($\mathcal{P}_1$, `SampleRock(L1)`, s_1). 70

5-8 $\mathcal{P}_3$: Result of calling Refine($\mathcal{P}_2$, `TakePicture(L1)`, s_2). 71

5-9 $\mathcal{P}_4$: Result of calling Refine($\mathcal{P}_2$, `Navigate(L1, L2)`, s_2). 72

5-10 $\mathcal{P}_5$: Result of calling Refine($\mathcal{P}_1$, `Navigate(L1, L2)`, s_1). 74

5-11 $\mathcal{P}_2$: Result of calling Refine($\mathcal{P}_1$, `SampleRock(L1)`, s_1). 76

5-12 $\mathcal{P}_3$: Result of calling Refine($\mathcal{P}_2$, `TakePicture(L1)`, s_2). 77

5-13 $\mathcal{P}_6$: Result of calling Refine($\mathcal{P}_3$, `Navigate(L1, L2)`, s_4). 78

5-14 $\mathcal{P}_7$: Result of calling Refine($\mathcal{P}_2$, `Navigate(L1, L2)`, s_5). 79

5-15 $\mathcal{P}_5$: Result of calling Refine($\mathcal{P}_1$, `Navigate(L1, L2)`, s_1). 80

6-1 Basic planning and execution cycle of a single agent. 83
6-2 Extending the "seed" POPG: (a) The initially constructed POPG $\mathcal{P}$; (b) EPOPG $\mathcal{P}_e$, resulting from the extension of $\mathcal{P}$ by the addition of an alternative course of action `Navigate(L1,L3)`; (c-d) The result of Refine on $\mathcal{P}_e$ and the actions `Navigate(L1,L2)` and `Navigate(L2,L3)`, respectively. 84
6-3 POPG for the `Rovers` example (from Fig. 5-4). 86
6-4 Example POPGs: (a) Base plan; (b) Extended plan; (c) Extended plan with shortcuts; (d) Base plan with repairs. 93

7-1 Workspace partitioning for `Rovers` MAS. (Repeated from Fig. 2-1) . 104
7-2 Modified `TO2PO` algorithm (adapted from [Ambite and Knoblock, 2001]). 106
7-3 Normalized experimental results for capability-based selection. 112
7-4 Normalized experimental results for goal-based selection. 113

8-1 AOSN task network (partial) for sampling a volume of water. 127
8-2 AOSN POPG (partial) for sampling a volume of water. 128
8-3 Example of a hierarchical organization for the AOSN MAS. 131
8-4 Aircraft Maintenance MAS 133
8-5 Example Aircraft Repair Query Plan (POPG) 137

9-1 Partial POPG with costs and values. 156

ABSTRACT

Multi-Agent Opportunism

by

James H. Lawton
University of New Hampshire, September, 2005

The real world is a complex place, rife with uncertainty, and prone to rapid change. Agents operating in a real-world domain need to be capable of dealing with the unexpected events that will occur as they carry out their tasks. While unexpected events are often related to failures in an agent's plan, or inaccurate knowledge in an agent's memory, they can also be *opportunities* for the agent. For example, an unexpected event may present the opportunity to achieve a goal that was previously unattainable. Similarly, real-world multi-agent systems (MASs) can benefit from the ability to exploit opportunities. These benefits include the ability for the MAS itself to better adapt to its changing environment, the ability to ensure agents obtain critical information in a timely fashion, and improvements in the overall performance of the system.

In this dissertation we present a framework for *multi-agent opportunism* that is applicable to open systems of heterogeneous planning agents. The contributions of our research are both theoretical and practical. On the theoretical side, we provide an analysis of the critical issues that must be addressed in order to successfully exploit opportunities in a multi-agent system. This analysis can provide MAS designers and developers important guidance to incorporate multi-agent opportunism into their own systems. It also provides the fundamental underpinnings of our own specific approach to multi-agent opportunism.

On the practical side, we have developed, implemented, and evaluated a specific approach to multi-agent opportunism for a particular class of multi-agent system. Our evaluation demonstrates that multi-agent opportunism can indeed be effective in systems of heterogeneous agents even when the amount of knowledge the agents share is severely limited. Our evaluation also demonstrates that agents that are capable of exploiting opportunities for their own goals are also able, using the same mechanisms, to recognize and respond to potential opportunities for the goals of other agents. Further, and perhaps more interesting, we show that under some circumstances, multi-agent opportunism can be effective even when the agents are not themselves capable of single-agent opportunism.

Chapter 1

Introduction

We have developed a framework for *multi-agent opportunism* for agents operating in open, real-world *multi-agent systems* (MASs). Single-agent opportunism is the ability of an agent to alter its pre-planned course of action to pursue a goal that better achieves the agent's or its designers' objectives for its performance, based upon a change in the environment or in the agent's internal state—an opportunity [Francis, 1997; Lawton, 1999, 2003]. Multi-agent opportunism refers to the ability of agents operating in an MAS to assist one another by *recognizing* and *responding* to potential opportunities for each other's goals.

Our interest in multi-agent opportunism stems from a desire to improve the system-level performance and coordination of multi-agent systems operating in real-world environments. An MAS is a collection of individual agents operating in a common environment coordinating their activities to enable the performance of the group to exceed the capabilities of any individual member [Nwana and Ndumu, 1999]. There are many real-world problem domains for which multi-agent systems can provide superior solutions, including information systems [Shehory et al., 1999], oceanographic sampling [Turner and Turner, 2001; Curtin et al., 1993], distributed sensor networks, electricity distribution, and personal travel agents [Nwana and Ndumu, 1999]. Most real-world MASs are necessarily *open systems* [Hewitt, 1991], in which the structure of the system itself may change dynamically. The component agents of an open MAS may not be known in advance, can change over time, and are often heterogeneous

[Jennings and Wooldridge, 1998].

The real world is an extremely complex place, rife with uncertainty, and prone to rapid change. Agents operating in a real-world domain need to be able to cope with its lack of predictability. Real-world agents need to be capable of dealing with the unexpected events that will occur as they carry out their tasks. While unexpected events are often related to failures in an agent's plan, or inaccurate knowledge in an agent's memory, they can also be opportunities for the agent. For example, an unexpected event may present the opportunity to achieve a goal that was previously unattainable. Similarly, real-world MASs can benefit from the ability to exploit opportunities. These benefits include the ability for the MAS itself to better adapt to its changing environment, the ability to ensure agents obtain critical information in a timely fashion, and improvements in the overall performance of the system.

Taking advantage of an opportunity is a difficult task even for an individual agent. As summarized in Figure 1-1, the opportunity must first be recognized, the action it facilitates must be determined, and the agent must decide whether or not it is appropriate to pursue the action at the current time [Francis, 1997]. For example, suppose you happen across a stream while walking in the woods. You would first have to decide if this event presents an opportunity by determining if you had some use for the stream outside the scope of your current activity. This decision is often based on the presence of *opportunity cues*, which are easily identifiable changes or conditions in the environment that indicate a potential opportunity. If you were to recognize the presence of a stream as an opportunity cue, you would then have to determine what action the opportunity is related to, such as to quenching your thirst or washing your hands. Finally, you would have to decide if it is currently appropriate to take advantage of this opportunity. For example, it would probably not be appropriate to stop for a drink while being chased by a bear.

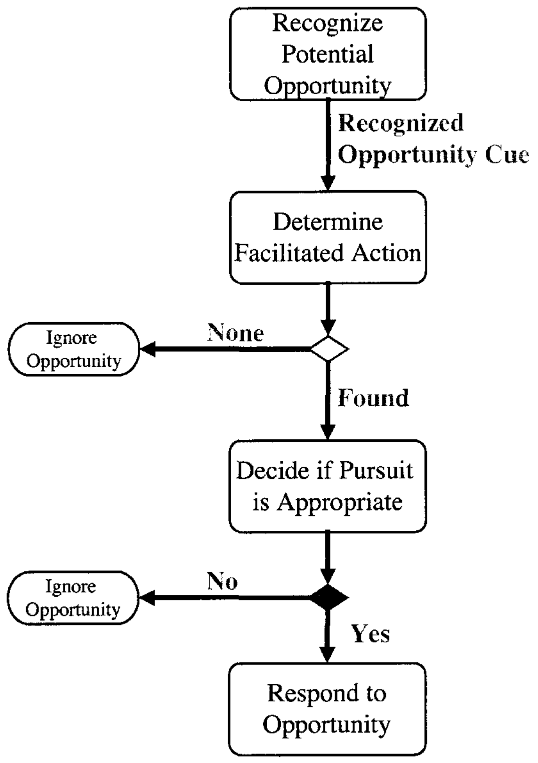
Recognize Potential Opportunity
Recognized Opportunity Cue
Determine Facilitated Action
None
Ignore Opportunity
Found
Decide if Pursuit is Appropriate
No
Ignore Opportunity
Yes
Respond to Opportunity

Figure 1-1: Decision Framework for Single-Agent Opportunism

The exploitation of opportunities by agents operating in an MAS is an even more complicated process. In addition to exploiting opportunities for themselves, agents in an MAS must also be capable of recognizing if a given event or situation may be an opportunity for a goal of *another* agent in the system, or even for a shared group goal. If an agent believes it has recognized an opportunity for another member of the MAS, it must first decide whether it should notify the other agent, or handle the opportunity itself by taking some action on behalf of the other agent. For example, if an agent happens across an object it knows another agent is looking for, it can notify the other agent of the object's location, or it might be able to acquire the object and deliver it to the other agent. To avoid conflicts, though, an agent would probably need to coordinate with the other agent before taking action on its behalf.

To further complicate matters, it is possible that more than one agent will simultaneously recognize a potential opportunity. If the agents respond to the opportunity by simply notifying the interested agent, then the agent receiving this notification must be capable of handling multiple notifications of the same event or situation. Otherwise, the response would almost certainly require coordination of actions among the agents involved. Continuing the above example, suppose two agents each find an object a third agent is looking for. As long as they each coordinate with the third agent, that agent can decide which should actually acquire and deliver the object. This approach also works even if the agents find different (but equivalent) objects.

Each of the decisions described above could require agents in an MAS to share a significant amount of information about each other. Because of this, our research has concentrated on addressing the critical knowledge-sharing and decision-making processes that are necessary for multi-agent opportunism. In particular, we have focused on the question: *Can multi-agent opportunism be effective in systems of heterogeneous agents with little or no shared knowledge?* That is, can open, real-world multi-agent systems benefit from being able to recognize and respond to opportunities for each other's goals? Or will the lack of shared information be too restrictive to allow cost-effective opportunistic assistance? We have developed a framework for multi-agent opportunism that assumes a minimal amount of shared knowledge. Based on experiments using an implementation of this framework, we show that multi-agent opportunism can indeed be effective even when the agents have only limited information about each other's capabilities. Further, through the empirical analysis described in Chapter 7, we quantify the effectiveness of multi-agent opportunism, as well as the limitation on the shared information, in the context of a specific class of MAS.

A second, related question that we have explored is: *Are agents that are capable* *of exploiting opportunities for themselves also capable, using the same mechanisms, of recognizing and responding to opportunities for other agents, given that they have adequate knowledge of each other?* Our framework for multi-agent opportunism described in this dissertation supports this conjecture. It provides agents with these capabilities, and, based on it, we are able define "adequate knowledge." Further, we have determined that under some circumstances, we can obtain multi-agent opportunism even when the agents are not capable of single-agent opportunism.

The contributions of this research are both theoretical and practical. On the theoretical side, our framework for multi-agent opportunism provides the basis for an analysis of the critical issues that must be addressed in order to successfully exploit opportunities in a multi-agent system. These critical issues involve the type and degree of knowledge the agents must share, as well as the decision-making capabilities they must possess. This analysis can provide MAS designers and developers important guidance to incorporate multi-agent opportunism into their own systems. It also provides the fundamental underpinnings of our own specific approach to multi-agent opportunism.

On the practical side, we have developed, implemented and evaluated a specific approach to multi-agent opportunism for a particular class of multi-agent system. Our approach enables systems of planning agents coordinated through a middle agent (e.g., a matchmaker or a broker) [Klusch and Sycara, 2001] to opportunistically assist one another achieve their goals, thus improving the overall system performance. We have achieved this by combining a particular form of single-agent opportunism known as *predictive encoding* [Patalano et al., 1991] with an approximate decision-theoretic planning mechanism that allows the agents to make critical information-sharing decisions. The planning mechanism uses a new abstract plan representation, *Partial Order Plan Graphs* (POPGs), that we developed specifically to support multi-agent opportunism. While the POPG representation was designed for this research project, it is still general enough to represent the features of most (if not all) techniques used in the area of classical (i.e., STRIPS-based [Fikes and Nilsson, 1971]) AI domain-independent planning [Smith, 2003].

We believe the contributions made by this approach are important for two key reasons. First, it should be applicable to many existing MASs. This is because we have leveraged well-understood technologies from the fields of AI planning (e.g., [Weld, 1999]) and MAS coordination (especially through middle agents) (e.g., [Wong and Sycara, 2000]), as well as the area of single-agent opportunism (discussed in Section 3.2). Second, our approach demonstrates that MASs can benefit from multi-agent opportunism even when the agents share little or no common knowledge of such things as plans, goals, and capabilities. This is because we have started from an assumption that the agents will *not* be homogeneous, in that they will possess different capabilities and that it is likely they will have been created by different software developers. As such, we can make no guarantees on the amount and type of shared knowledge the agents possess, so we assume that they would possess little or no shared knowledge at all. Our results thus provide a baseline, demonstrating that performance improvements can still be achieved even with these restrictive assumptions.

The remainder of this dissertation is organized as follows. Chapter 2 provides motivating examples of how opportunities could be exploited in MASs. These examples are used throughout the rest of this dissertation to illustrate various points. In Chapter 3 we review existing work in opportunism, including well-known approaches to single-agent opportunism. In the next four chapters, we describe our framework for multi-agent opportunism. In Chapter 4 we present an analysis of the critical information-sharing and decision-making issues that must be addressed by any approach to exploiting opportunities in an MAS. In Chapters 5 and 6 we describe our specific approach to multi-agent opportunism for systems of planning agents. In Chapter 7 we present an empirical analysis of the impact that our approach to multi-agent opportunism has on the performance of a particular class of MAS. In Chapter 8, we discuss how our framework could be applied to other classes of MASs. Finally, in Chapter 9 we summarize the results of this research and compare our approach to other research areas that are related to multi-agent opportunism. We conclude by providing directions for future research for our framework.

Chapter 2

Example Scenarios

In this chapter we present two example scenarios in which the agents are able to opportunistically assist one another satisfy their goals. These examples are used to illustrate our points throughout the rest of this dissertation. Note that in this chapter we only present the domains along with some specific examples of opportunistic behavior within them. The mechanisms used by the agents to achieve this behavior will be explained in detail in the remainder of this dissertation.

2.1 Example Scenario 1 – Planetary Rovers

Our first example domain is based on the standard planning benchmark domain `Rovers`, inspired by the planning problems for NASA's Mars Rovers, and used in International Planning Competition (IPC-2002) [Fox and Long, 2002]. The domain consists of a collection of agents (rovers) that can navigate the surface of a planet, perform scientific tasks, and communicate information back to a stationary lander.

For this example, consider a scenario in which a group of rovers is performing various tasks in a working area consisting of 25 waypoints, arranged at the cells of a 5×5 grid (see Figure 2-1). Each rover can travel among the waypoints and perform scientific tasks such as soil sampling, rock sampling, and taking pictures of objects of interest if they are visible from the rover's current location. We will assume that rovers can only travel to the 13 waypoints shown in white in Figure 2-1 (i.e., the

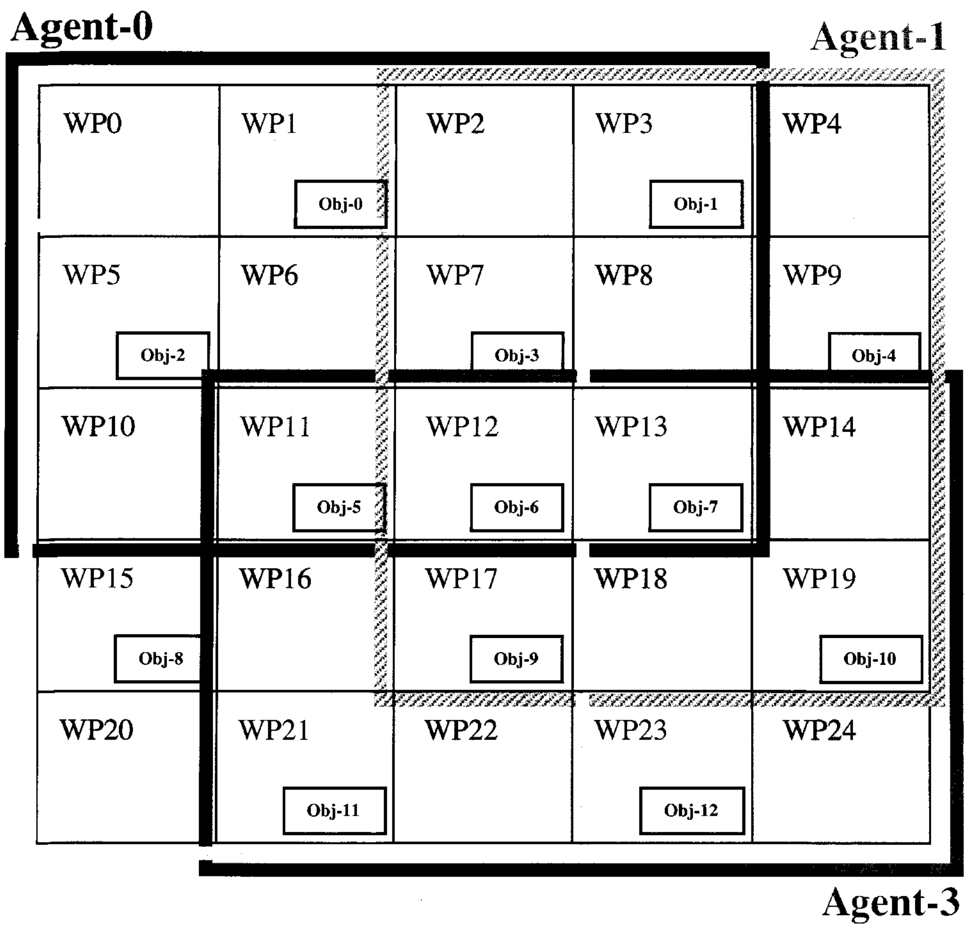

Figure 2-1: Workspace partitioning for `Rovers` example MAS.

12 grey waypoints are inaccessible). There are thus a total of 65 different goals in this scenario that can be assigned to the rovers, namely 13 rock samplings (at the accessible locations), 13 soil samplings, and 13 objects to be photographed, where each picture can be taken at three different levels of quality.

In this scenario we will also assume there are four rover agents, $\mathbb{A}_1, \ldots, \mathbb{A}_4$, with partially overlapping capabilities. The working area is divided into 4 partially overlapping regions (see Figure 2-1), one for each agent. Each rover agent can only travel within its designated 3×4 area, and thus can only be assigned goals for waypoints in this region. The goals that can be assigned to a rover $\mathbb{A}_i$ constitute its capabilities C_i, and these are restricted to the sub-area in which this rover can travel. Each agent is thus capable of performing some 35 (of the 65) goals. At the beginning of each planning/execution cycle, each agent is assigned a set of goals G_i from its capabilities (i.e., $G_i \subseteq C_i$) such that no goals are assigned to more than one agent (i.e., $\bigcap_i G_i = \emptyset$). The agents each plan for their individual sets of goals and then execute that plan.

For this example we will assume that the agents' activities are coordinated through the use of a *middle agent*, or more specifically a *task broker* [Klusch and Sycara, 2001], $\mathbb{B}$. As each agent comes on line, it registers its capabilities with the broker $\mathbb{B}$. As noted above, in this example an agent's capabilities are descriptions of the goals it can satisfy. This is used to simplify the decision-making process of the broker. In this research, we are not concerned with the broker's actual decision process or the details of its implementation. In general, however, a capability description language such as LARKS [Sycara et al., 1999], CDL [Wickler, 1998], or DAML-S [Ankolekar et al., 2001], would likely be needed.

As a specific example of how agents could opportunistically assist one another in this scenario, suppose that rover $\mathbb{A}_0$ is assigned the goal $g_0 =$ `have-rock-sample(WP7)`, but is unable to accomplish this goal (perhaps because it has insufficient resources). If $\mathbb{A}_0$ believes that there are other rover agents in the area that are also capable of taking this sample, it can ask them to try to satisfy this goal opportunistically. One way to accomplish this would work as follows: when $\mathbb{A}_0$ suspends g_0, it would contact the broker, $\mathbb{B}$, and request a list of agents, L, whose capability sets include `have-rock-sample(WP7)`[1]. If we assume all of the agents are capable of taking rock samples, then from Figure 2-1 we can see that $L = \{A_0, A_1, A_3\}$. $\mathbb{A}_0$ would notify each of these agents (excluding itself, of course) that it has suspended its assigned

[1]While capabilities like these would normally be represented with variables (e.g., `have-rock-sample(X)`), we have propositionalized them here for simplicity.

goal g_0. Such a notification could be in the form of a message similar to the FIPA `request-if` directive [FIPA, 2002], which in this context would be interpreted by the receiving agents as a request to incorporate the suspended goal into their current plans if possible, and satisfy it if an opportunity to do so arises.

Alternatively, if the broker $\mathbb{B}$ has been designed to support multi-agent opportunism, $\mathbb{A}_0$ could simply notify $\mathbb{B}$ that it is suspending g_0. $\mathbb{B}$ would determine which other agents are capable of satisfying g_0, and itself notify these other agents of the suspended goal as described above. This approach would reduce the number of messages that would be needed to disseminate the request for opportunistic assistance.

Now suppose that the goal g_1 = `have-rock-sample(WP12)` has been assigned to rover $\mathbb{A}_1$. Since $\mathbb{A}_1$ will be close to `WP7` when it is satisfying g_1, if it has sufficient resources it could adjust its intentions and satisfy g_0. It might do this by augmenting its current plan to take a detour to `WP7` and take a rock sample there. If and when $\mathbb{A}_1$ does satisfy g_0, it would notify $\mathbb{A}_0$ that it has taken action its behalf. $\mathbb{A}_0$ would then notify the other agents in $\mathcal{A}$ that the suspended goal had been satisfied. Alternatively, if the broker is being used for indirect notification, $\mathbb{B}$ could be informed of g_0's satisfaction, requiring it to notify the other agents.

Note that in this example the agents respond to opportunities to accomplish one another's suspended goals by simply satisfying them and notifying the original that it has done so. In this domain this is a reasonable response, since redundant satisfaction of the goals does not cause conflicts. In other domains further coordination may be required. Other response options will be discussed in Section 4.2.2.

2.2 Example Scenario 2 – Autonomous Oceanographic Sampling Networks

The second scenario, adapted from examples of Turner and Turner [2001, 1999, 1998] and Chappell et al. [1997], is taken from the domain of autonomous oceanographic sampling networks (AOSNs) [Curtin et al., 1993]. AOSNs are multi-robot systems designed to collect data from the ocean over long periods of time. They are composed of a variety of Vehicle and Instrument Platforms (VIPs), which are mobile and non-mobile platforms that support various data-gathering instruments. While we use examples from this scenario throughout this dissertation, we have not directly implemented our framework in this domain. In Chapter 8, however, we do discuss the key differences between this domain and the `Rovers` example, as well as how our framework for multi-agent opportunism could be applied.

One way of controlling an AOSN is to treat it as an MAS [Turner and Turner, 1998]. We should stress that our main interest in this example is the characteristics of the MAS, which itself is only one particular example of how an MAS could be used to support an AOSN. We have selected this particular example because it exemplifies the use of a particular class of MAS in a real-world domain.

For this scenario, assume there is a collection of agents assigned to accomplish the following tasks: (1) to characterize the temperature and salinity of a volume of water, which we will call the `background-survey-task`, (2) to make a detailed survey of *convective overturns* (i.e., CONVEX events [Bub et al., 1997]) when they occur, called the `CONVEX-task`, (3) to characterize the magnetic fields along the bottom (`magnetic-survey-task`), and (4) to collect rocks from areas that are unusually magnetic (`rock-collection-task`).

We will further assume that the agents are realized as the hardware and control software of several VIPs. These VIPs include two highly maneuverable but slow

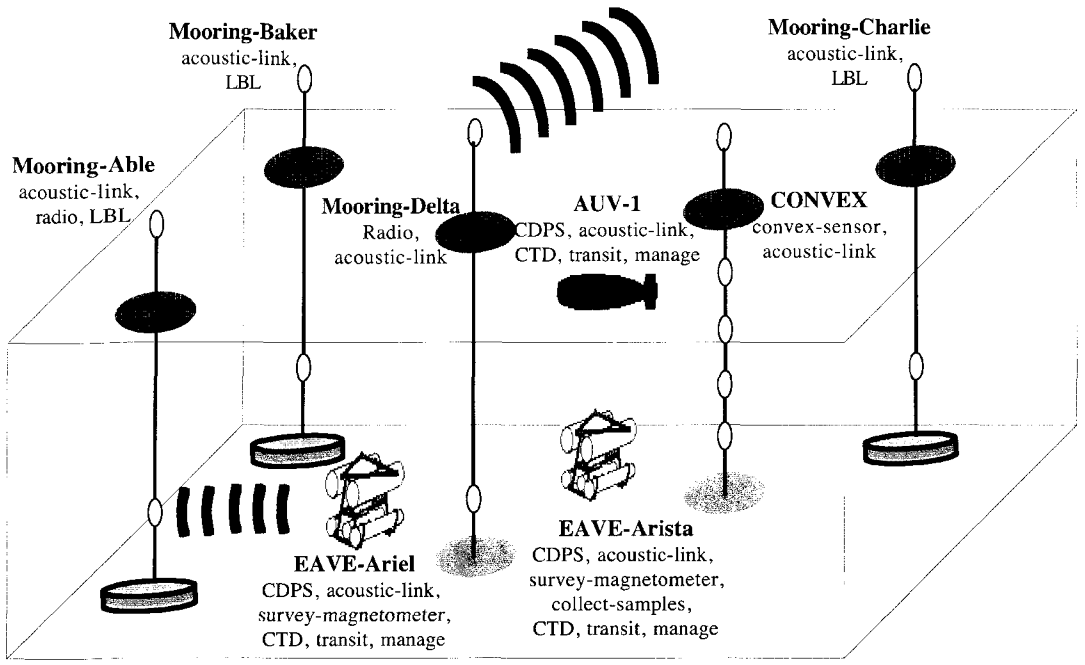

Figure 2-2: AOSN-Layout (Adapted from [Turner et al., 1997]).

Experimental Autonomous Vehicles (EAVEs) [Blidberg and Chappell, 1986], ARIEL and ARISTA, one long-range autonomous underwater vehicle (AUV), AUV-1, three navigational moorings, ABLE, BAKER, and CHARLIE, one communications mooring, DELTA, and one CONVEX-MOORING [Bub et al., 1997]. See Figure 2-2 for more information.

The `CONVEX-task` can be further broken down into a `CONVEX-detection-task` and a `CONVEX-survey-task`. The `CONVEX-survey-task` can only begin after a CONVEX event has been detected, and will involve several agents capable of performing a CTD-survey. The CTD-survey capability means that the agent is able to take conductivity (from which salinity can be determined), temperature and depth (CTD) measurements in a volume of water. In the `background-survey-task`, we will assume that each agent will survey a given volume of water, returning to a communications mooring regularly to download the collected data. We will also assume that each agent will perform some simple anomaly detection on the data before downloading it. Finally, the `magnetic-survey-task` will require agents capable of taking magnetic readings with a magnetometer in a volume of water (survey-magnetometer capability), while the `rock-collection-task` will require agents with both survey-magnetometer and collect-samples capabilities.

Before the agents actually begin operation, one is selected to generate a plan for accomplishing the mission (`background-survey-task`, `CONVEX-task`, `magnetic-survey-task` and `rock-collection-task`). The planner agent decomposes the assigned tasks into subtasks, and uses this information to determine the group's *task-level organization* (TLO) [Turner and Turner, 2001, 1999, 1998]. The TLO is an organization of roles with assigned tasks, with specific agents assigned to those roles. The TLO is used by the agents to coordinate their operational activities. In this example, the TLO is structured as a simple hierarchy, although other organizational structures are possible. The roles in the TLO are either labor roles, which are assigned tasks that actually do something in the world, or management roles, which are assigned tasks to manage labor roles or other management roles. In the hierarchy, labor roles occur at the leaves, while the management roles are the interior nodes (or the root). Figure 2-3 shows a possible TLO structure for our example scenario.

After the planning process, assume that the planner has assigned the following tasks to agents: The `CONVEX-detection-task` to the CONVEX-MOORING (leaving the `CONVEX-survey-tasks` unassigned); the `background-survey-tasks` to EAVE-ARIEL, EAVE-ARISTA, and AUV-1; the `magnetic-survey-task` is assigned to the EAVEs, and the `rock-collection-task` to EAVE-ARIEL. We will assume that the EAVEs both have survey-magnetometer capabilities, but only EAVE-ARIEL has collect-samples capabilities. Also, we will assume the planner has assigned volumes of water to the background-survey agents such that AUV-1 is surveying the upper 50%

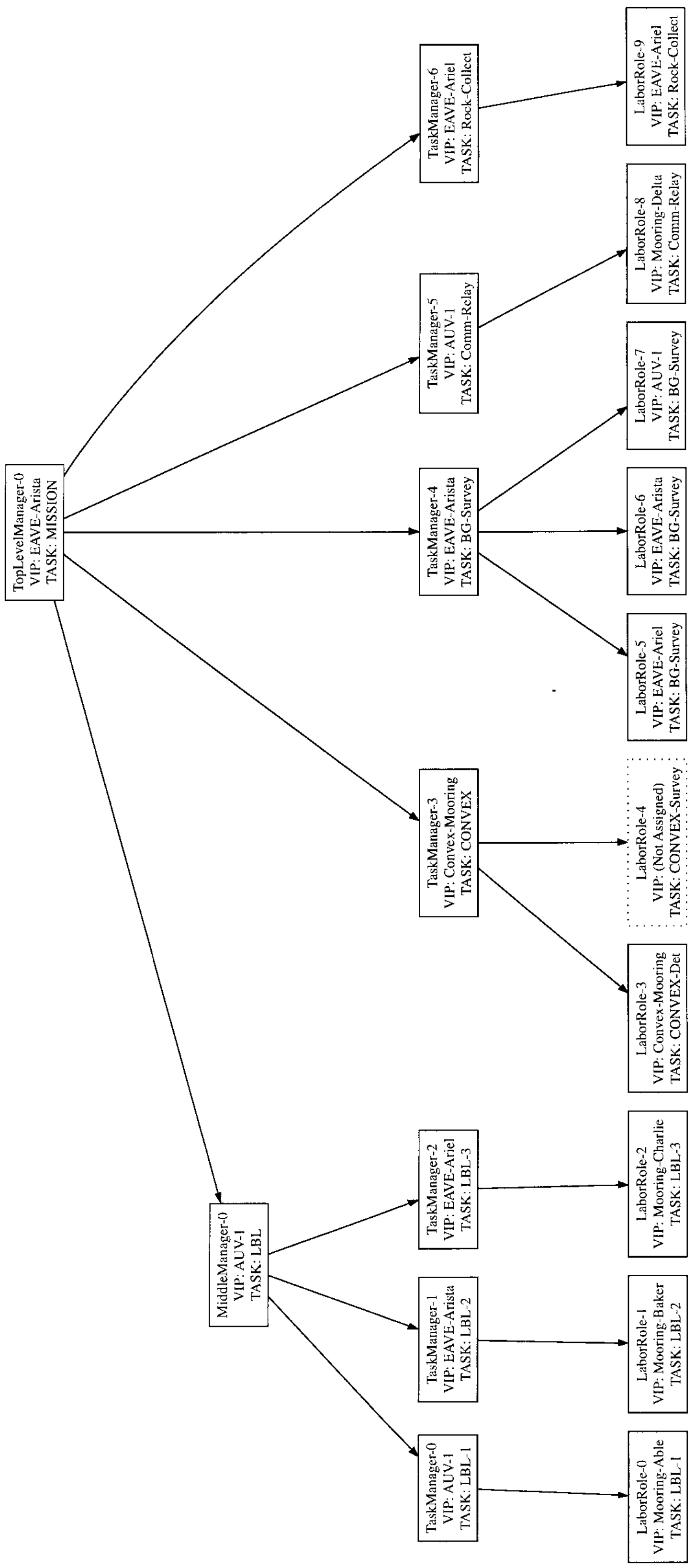

Figure 2-3: AOSN Organization.

of the volume, and the EAVEs are each surveying half of the remaining "bottom" volume (25% of the total volume each). The planner has also determined (or is told) that two other mission-related tasks are needed: a `long-baseline-navigational` (LBL) task, which it assigns to MOORING-ABLE, MOORING-BAKER, and MOORING-CHARLIE, and a `communications-relay-task`, which it assigns to MOORING-DELTA.

As a specific example of multi-agent opportunism in this scenario, suppose that EAVE-ARIEL has suspended its `rock-collection-task` while it is performing its `background-survey-task`, and that EAVE-ARISTA has begun its `magnetic-survey-task`. If EAVE-ARISTA knows about EAVE-ARIEL'S suspended `rock-collection-task`, or more directly that EAVE-ARIEL has a suspended task that is missing knowledge about the locations of unusual magnetic readings, EAVE-ARISTA could notify EAVE-ARIEL if it discovers any such readings. EAVE-ARIEL would then have to determine if it should suspend its current `background-survey-task` to take advantage of this opportunity, or to cache this knowledge for later. Note that in this case EAVE-ARISTA is unable to collect rock samples on EAVE-ARIEL'S behalf, since it does not possess the collect-samples capability.

Chapter 3

Opportunism in Planning Agents

In this chapter we present background information on opportunism. We first present various definitions of opportunities and opportunism that are generally accepted in the literature. We then briefly review existing approaches to exploiting opportunism in agents that use plans to achieve their goals.

3.1 Definitions

Just what constitutes an opportunity is not always clear. Very few authors have even attempted to define opportunities or opportunism, leaving it tacit. From the way it is used in the literature, we have identified two prominent views of what an opportunity is. We thus claim that an opportunity can be defined by one of the following:

1. A situation or condition favorable for the attainment of a goal, or
2. An unexpected change in the world that requires a response outside the scope of the current activity [Francis, 1997].

A subtle but key difference between these definitions is that the second focuses on a specific event which changes the current state of the environment, while the first is only concerned with the state, not how or when it got there. In addition, as the term is commonly used, opportunities are expected to be positive in nature, and are often unexpected, leading to a sense of serendipity.

We thus define opportunism as: *The ability to exploit an opportunity by altering a pre-planned course of action to pursue a different goal, based on some change in the environment or in the agent's internal state* [Francis, 1997; Lawton, 1999, 2003].

Considering these definitions, there are a number of different views of what kinds of situations or events are actually considered opportunities. From the way they have been used in the literature, we have identified the following types of opportunities that an agent might encounter. We simply list these types here. We will explain how each has been exploited in Section 3.2.

- Satisfaction of suspended goals: An event or situation that might allow a suspended goal to be satisfied [Birnbaum and Collins, 1984; Francis, 1997; Hammond et al., 1993; Patalano et al., 1991; Simina and Kolodner, 1995; Pryor, 1996; Pryor and Collins, 1992; Simina et al., 1997, 1998]. A suspended goal is one that cannot be fit into an agent's current agenda, and is thus postponed rather than abandoned [Schank and Abelson, 1977; Patalano et al., 1991]. It may also be a goal that the agent intends to achieve later in its plan, but can be satisfied early if an appropriate situation is encountered.

  Example: While performing a critical oceanographic sampling mission, an autonomous underwater vehicle's (AUV) power level drops below a certain threshold that causes a goal for recharging to be activated. The AUV decides that, because the recharging source is too far away, it cannot pursue the new goal until it finishes the current mission, and so suspends it. During the mission, though, a support vessel enters the area where the AUV is working, presenting an opportunity to recharge before completing the current mission.

- Execution of standing orders (e.g., repeating goals): An event or situation that allows for the satisfaction of goals that can be accomplished repeatedly as conditions arise [Fasciano, 1996; Hayes-Roth and Hayes-Roth, 1979; Hammond et al., 1996; Turner and Turner, 1998].

Example: An AUV is told to always take a soil sample when entering a new port. The AUV enters Boston Harbor, where it has never been before, presenting an opportunity to take a soil sample in the new port.

- Learning from failure: A failure in a plan or action can be an opportunity to learn [Fasciano, 1996; Hammond et al., 1996].

Example: An AUV is given a mission to collect samples from a volume of water where there is a strong current. It plans a sampling pattern that follows the direction of flow of the current. However, the current carries the AUV away from the sampling region before it completes its mission. It recognizes the plan failure, re-plans the mission to collect the samples while traveling against the current, and takes advantage of this opportunity to learn by storing the failed and successful plan in its memory.

- Learning new/better method for future goal: An event or situation may point out a new/better way to achieve a goal. This type of opportunity applies to future goals, which may never come up [Simina and Kolodner, 1995; Simina et al., 1997, 1998].

Example: An AUV successfully completes a mission to take samples in a volume of water where there is a strong current, using a sampling pattern that goes against the current. Later, the AUV observes another AUV that performs a similar sampling mission in less time by using a sampling pattern that goes across the current. It takes advantage of this opportunity to learn by storing the new plan in its memory.

- Searched-for situation: A situation or condition that could lead to the attainment of some goal that occurs while it is specifically being searched for. During the search, however, it is not known if the situation or condition actually exists, or how it will manifest itself [Birnbaum and Collins, 1984; Fasciano, 1996; Francis, 1997; Hammond et al., 1996; Hayes-Roth and Hayes-Roth, 1979; Simina and Kolodner, 1995; Simina et al., 1997, 1998].

  Example: An AUV is sampling a volume of water that is divided by a reef. It begins by taking samples on one side of the reef, suspending the sampling on the other side until it finds a way across. While it is taking samples along the reef, it finds a tunnel, allowing it to pass through the reef and complete its sampling task.

From these descriptions one can see that there are a number of different ways for an agent to take advantage of certain changes or conditions in its environment. In the next section we present several methods for an agent to exploit the various types of opportunities described above for its own goals.

3.2 Single-Agent Opportunism

The ability to exploit opportunities can be extremely beneficial to, and possibly even necessary for, planning agents for their correct functioning [Hammond et al., 1993; Francis, 1997]. There are a number of different approaches to enable an agent to recognize and exploit opportunities for its own goals (i.e., single-agent opportunism). They can, however, be broken into three general classes: *active approaches*, *passive approaches*, and *hybrid active/passive approaches*. In this section we discuss these general approaches, using specific example systems taken from the literature.

3.2.1 Active Approaches

When using an active approach to opportunism, some sort of process actively watches for a situation in which it can achieve its assigned goal or task. Systems employing an active approach tend to follow the first definition of opportunism given in Section 3.1, focusing primarily on the opportunity recognition problem. Notable examples of systems using an active approach to opportunism include the blackboard-based opportunistic planning system of Hayes-Roth and Hayes-Roth [1979], the *active goals* model proposed by Birnbaum and Collins [1984], and the opportunistic learning system included in MAYOR [Fasciano, 1996; Hammond et al., 1996], a real-time player of the SimCity simulation game.

The most significant advantage of an active approach to opportunism is that it is capable of recognizing any of the opportunity types described above in Section 3.1, although implementations have focused on the searched-for situation opportunity type. That is, instead of watching for changes in the environment that may indicate an opportunity, the active processes examine the current state of the environment, watching for conditions favorable to the satisfaction of pending goals or tasks. However, this opportunity type can subsume the other types: the active processes may look for any state or condition desired, be it for the satisfaction of a task or goal (current or suspended), the execution of standing orders, or the chance to learn.

Further, agents using an active approach to opportunism are capable of exploiting *execution-time opportunities* [Hammond et al., 1993; Pryor, 1996; Pryor and Collins, 1992; Simina and Kolodner, 1995], as opposed to opportunities that occur at *planning-time*. One way plans are used for controlling an agent's activity is to separate the development of a complete plan from the execution of that plan. The first phase is referred to as planning-time, while the second is known as execution-time. The ability to exploit execution-time opportunities is especially important in real-world environments, since that is when many opportunities will occur. The approaches developed by [Hayes-Roth and Hayes-Roth, 1979] and [Birnbaum and Collins, 1984] did not include an execution component, and thus only considered planning-time opportunities. However, systems such as MAYOR (above) and IMPROVISOR (discussed in Section 3.2.3) do use an active approach for noticing and exploiting execution-time opportunities.

A final important capability of systems using an active approach is that they may be able to identify *novel opportunities*—those that may provide nonstandard solutions but that were not specifically predicted [Wills and Kolodner, 1994; Simina and Kolodner, 1995]—because they need not anticipate every situation for the satisfaction of pending tasks or goals. To do this, the features being watched for by the active processes must be described abstractly enough to match any condition that might present an opportunity.

The core idea behind the active approaches—that active processes continuously watch the environment for opportunities—has been criticized as being an unlikely explanation of human opportunity recognition [Hammond et al., 1993; Patalano et al., 1991]. There are two key arguments behind this criticism. First, the idea is considered unlikely because of the computational demands: it is simply not computationally feasible to constantly watch every aspect of the environment looking for opportunities. This problem is exacerbated when considering novel opportunities, since the computational load would be increased even more if a reasoner would have to make deep inferences about every observed environmental feature. Second, the active approach is not considered cognitively plausible. This is because of the cognitive resources that would be required, as described above, as well as because it does not take into account the fact that people regularly fail to recognize potential opportunities. A final, more subtle, problem with the active approach is that it lacks the intuitive sense of serendipity generally associated with opportunity recognition in humans.

However, none of these arguments are compelling for artificial systems operating in real-world domains. It is certainly likely that using active processes to continuously watch the environment for opportunities would have high computational demands in artificial systems, just as it does in people. It is not clear, though, that these demands would be beyond the capabilities of the computer. In fact the existence of complex systems such as MAYOR demonstrate that an active approach to opportunism is computationally feasible under some circumstances. Similarly, cognitive plausibility and a lack of serendipity are not important for artificial systems. Cognitive plausibility can be important for dealing with poorly understood activities, such as opportunity recognition, that humans are often very good at performing. Under such circumstances, emulating the way people deal with these activities may lead to an efficient approach for artificial systems. This does not mean, however, that a cognitively plausible approach is necessarily the best approach for a computational system. Further, a lack of serendipity may be important for explaining models of opportunism in people (and may not be much more than an introspective artifact for that, either), but is of little importance to computational systems.

3.2.2 Passive Approaches

Agents that utilize a passive approach to opportunism detect potential opportunities during the normal course of their processing. They spend few or no computational resources looking for opportunities. This does not mean that no computation to detect opportunities is performed. Rather, systems using a passive approach predetermine the environmental cues that might indicate an opportunity for some goal or task, and then use an efficient mechanism to detect these cues during the normal course of processing.

Systems that implement a passive approach to opportunism tend to subscribe to the second definition of opportunism given in Section 3.1, which is the ability to alter a pre-planned course of action to pursue a new goal, based on some change in the environment. As such, they require three particular capabilities for opportunistic behavior: the ability to recognize an opportunity, the ability to suspend or modify current goals or tasks to pursue an opportunity, and the ability to decide whether or not to pursue an opportunity in the current context [Francis, 1997]. The primary focus of most research, however, has been placed on opportunity recognition, often treating the latter two aspects incidentally.

Planning agents using a passive approach to opportunism are generally *active planners* [Hammond et al., 1993], which are a class of planners that both produce a plan and then actively pursue and alter that plan in the face of a changing environment. That is, they are planners that interleave planning and acting. Because active planners are confronted with new goals during execution as well as during planning, complete re-planning is often impossible, or at least undesirable. Instead the new goals must be addressed by the planning system as they arise. Those goals that cannot be fit into the current, on-going agenda of plans are *suspended*—postponed, but not abandoned—by the planner [Patalano et al., 1991].

A prominent method of passive opportunism involves the *predictive encoding* [Patalano et al., 1991] of suspended goals. With predictive encoding, suspended goals are associated at the time they are postponed with features of the environment in which goal achievement would likely be possible, and attached to the plan components that may be associated with those features. It is considered predictive because the features that indicate the relevance of a plan are anticipated and used to index the goal [Patalano et al., 1991]. When this happens, the planner must be able to anticipate the conditions that will potentially lead to satisfaction of the suspended goals. Thus, at planning time, it must have a clear idea of what an execution-time opportunity will look like [Hammond et al., 1993].

Some notable examples of systems employing passive opportunism include the UPS-like pickup/delivery planner TRUCKER [Hammond et al., 1993], its follow-on errand-running planner RUNNER [Hammond et al., 1996], and the autonomous underwater vehicle (AUV) control agent Orca [Turner and Turner, 1998; Lawton et al., 1999].

The most significant advantage of the passive approach is its computational efficiency. Since opportunities are recognized through the reasoner's normal reasoning processes (e.g., plan execution), little additional run-time processing is required. Further, features such as plan preconditions and resource requirements can be used as opportunity cues. As these features are explicitly represented in planning systems, they are likely to be well defined, or at least inferable, in the reasoner's domain, thus keeping the extra computation tractable.

Unlike the active approach, we do believe the passive approach is cognitively plausible. Predictive encoding has been shown to be a reasonable model for human opportunity recognition [Patalano et al., 1991]. Also, unlike the active approach, the passive approach accounts for missing opportunities, since predicting every feature that could indicate a potential opportunity is a nearly impossible task in any real-world domain. Again, though, cognitive plausibility is an interesting, but not critical quality in planning agents.

Finally, as with the active approaches, systems using a passive approach can take advantage of execution-time opportunities. In fact, since they interleave planning and acting, these systems will necessarily exploit runtime opportunities. As we noted earlier, real-world environments tend to be highly dynamic and unpredictable, and are likely to be filled with such execution-time opportunities. As such, the ability to exploit these opportunities is very important for agents operating in this type of domain.

The biggest drawback to the passive approach to opportunism is that the number of opportunity types it can recognize is limited. In fact, the only opportunity type that has been directly addressed by research into this form of opportunism is for satisfying suspended goals. However, agents using predictive encoding could be augmented to recognize the satisfaction of the standing orders opportunity type, by permanently suspending a repeating goal. By its nature, though, the passive approach essentially precludes the recognition of the searched-for situation type opportunities.

A related and similarly important problem with this approach is that it requires the reasoner to know in advance what the execution-time opportunity cues might be. This could require arbitrarily deep inferencing, depending upon the number and type of possible cues the reasoner wishes to come up with. For example, suppose a `Rover` agent has a goal of transmitting data back to the lander, which is suspended on the unmet precondition of being in line-of-sight with the lander. It might naturally select locations where it knows it can see the lander as potential opportunity cues to satisfy this goal. However, if it can reason that it could transmit the data to another `Rover` agent, which could in turn relay the data to the lander, it could then include encountering another agent as an opportunity cue. This reasoning could be very computationally complex, depending on how long the chain of inferences is that the agent attempts to make. Further, unless the cues are very abstractly specified (which would make recognition difficult), this all but precludes the possibility of recognizing novel opportunities, since by definition they are those events or conditions that could not be pre-determined.

Finally, in spite of the fact that the passive approach is significantly more computationally efficient than the active approach, it can, however, still be computationally complex. As mentioned above, pre-determining opportunity cues can be difficult. Further, although much of the research has been focused on opportunity recognition, this is only part of the problem. Once a reasoner determines that an event or condition may present an opportunity, it must then determine exactly what the event or condition may be an opportunity for, and whether it actually is an opportunity. If the event or condition is determined to be an actual opportunity, the reasoner must then determine if that opportunity should be pursued. Each of these decisions, which are included naturally in the active approach to opportunism, may themselves require significant inferencing.

3.2.3 Hybrid Approaches

Systems using the active approach can recognize any type of opportunity, including novel opportunities, but their computational demands are not practical. Those using a passive approach are computationally efficient, but can only recognize opportunities that have been previously considered. There have been a number of attempts at finding a middle ground between the extremes of these approaches, combining the best aspects of each along with other unique ideas. In this section we discuss these hybrid approaches.

IMPROVISER [Simina and Kolodner, 1995; Linda Wills and Janet Kolodner, 1994] is a reasoning system capable of recognizing opportunities in the domain of creative design. ALEC [Simina et al., 1997, 1998], essentially an extension of IMPROVISER, examines long-term problem solving and creative design as modeled after a case study of Bell's invention of the telephone. Both systems combine predictive encoding with a limited number of active goals in order to recognize both anticipated and novel opportunities for suspended goals. By limiting the number of active goals that are considered, these systems can deeply analyze new events with respect to just these goals, making the problem tractable.

The domain in which these systems are being applied does, however, simplify the recognition problem. As such, while the approach may be sound, it might not be generalizable to other, more complex environments. This is not to say design problems are not complex. Rather, the simplicity here is in the use of physical objects as the focus of recognition. Objects in general have well-defined, often easily recognizable features and characteristics. Abstract situations found in many planning domains are likely to have much less clearly identifiable features, requiring deeper analogical reasoning to identify novel opportunities.

PARETO [Pryor, 1996; Pryor and Collins, 1992] is a planner for a pickup/delivery agent that utilizes a filtering mechanism for opportunity recognition for its active goals. PARETO's filtering mechanism is based on Pryor's *critical factor hypothesis*, which states that the presence of a single factor is often crucial for the existence of an opportunity. The filtering process works by attaching reference features—labels for general functional properties of objects, such as "sharp," "absorbent" and "sturdy"—to the representations of situation elements (objects found in the environment), and to pending goals. Pryor claims that reference features tend to be stable across situations, as well as being highly predictive and cheap to recognize. As objects are encountered in the world, their reference features are matched with those of pending goals. But, even if the reference features of an object matches those of a pending goal, it does not guarantee that encountering that object is an opportunity to satisfy that goal. Thus, more detailed inferences must be made to determine if the presence of the new object does actually constitute an opportunity [Pryor, 1996].

The use of a filtering mechanism appears to be a reasonable approach to dealing with the computational complexity of opportunity recognition, especially for novel opportunities, by reducing the number of potential opportunities the agent must consider. It is not without its problems, though. The critical factor hypothesis has yet to be supported by any substantial study—its validity has been taken so far from anecdotal evidence. Even assuming it is a valid hypothesis, it requires the *a priori* identification of reference features. This is the same limitation that Pryor claims predictive encoding has—the need for anticipated opportunity cues. If, however, a generic set of reference features could be identified, it should be more computationally efficient than determining opportunity cues for each goal individually. Further, Like IMPROVISER and ALEC, PARETO's problem domain uses the features of physical objects as its focus of opportunity recognition. The effects of this simplification are perhaps even more pronounced in PARETO, considering that reference features have been only defined with respect to physical objects. Identifying reference features for abstract situations has not been examined, and could be considerably more difficult.

MOORE is the opportunistic memory subsystem of the memory-based reasoning system Nicole [Francis, 1997], which is claimed to implement a complete theory of agency. According to Francis, such a complete system must include a memory component capable of anytime, asynchronous retrieval based on the current context, a problem solver that can integrate new information at any time, and a meta-controller that can select which task to pursue based upon some measure of that task's utility. Further, in a system that implements a complete theory of agency, all reasoning processes must either be opportunistic themselves, or must be able to participate in opportunistic activity on the part of the agent's overall reasoning system [Francis, 1997].

MOORE was designed to be a generic, cognitively plausible memory system. Context focusing is accomplished through the use of a unified blackboard system for all reasoning, working memory, perception and action processes. Processing any opportunity begins with a change to the blackboard, and any change in the blackboard, from any source, can represent a potential opportunity [Francis, 1997].

Through MOORE and Nicole, Francis has certainly addressed a number of key issues of potential importance to passive opportunistic systems, including anytime, asynchronous retrieval, context sensitivity, and the need for both internal and external retrieval cues. Like Pryor, Francis points out the need for a reasoner to be capable of determining when a potential opportunity should be pursued. One should note, however, that although these issues have not been specifically addressed by systems like RUNNER and TRUCKER, it would be possible for passive opportunistic systems to include these capabilities. For example, the reasoners discussed in Section 3.2.2 use a working memory as a repository for both recording external environmental changes as well as the results of internal computations. As such, changes to values in working memory could be used as opportunity cues, in a manner similar to that used in MOOREs central blackboard.

It should also be noted that, while not specifically addressed, these shortcomings are already handled implicitly by systems using an active approach to opportunism. Recall that in such systems the active processes are constantly watching the environment for situations or events that present opportunities for the achievement of their given tasks or goals. Any given process can react asynchronously to other processes in the system. Further, the active processes can watch for any feature internal or external. For example, the items posted to the central data structures in both the Hayes-Roth and Hayes-Roth model (i.e., the blackboard) and in MAYOR (i.e., the agenda) represent external environmental features as well the results of internal inferences. These structures also already represent the operating environment, making the systems context sensitive as well. Finally, the decision as to when to pursue an opportunity is also necessarily handled by the active processes: whenever the situation is deemed appropriate by an active process, as determined by whatever criteria the process has programmed into it, the corresponding task or goal is pursued.

3.3 Multi-Agent Opportunism

To the best of our knowledge, there are no other research efforts that have studied or are currently studying multi-agent opportunism. There are a few projects that are examining some opportunistic methods in the context of multi-agents systems. There are also a number of projects that have studied related problems, as well as those that have inherent opportunistic capabilities, including coalition formation in multi-agent systems, agent coordination through plan merging, and team formation in systems of agents. We review this related work later in Section 9.2, where we are able to discuss it in the context of our own approach.

3.4 Summary

In this chapter we have defined the ideas of opportunities and opportunism that are generally accepted in the literature, and we have briefly reviewed existing approaches to single-agent opportunism in planning agents. In the next chapter, we present our general model for multi-agent opportunism, including a description of the types of opportunities that might arise in an MAS and a discussion of the difficult problems that must be addressed to achieve multi-agent opportunism.

Chapter 4

Multi-Agent Opportunism

In this chapter we present our general framework for multi-agent opportunism. In particular we discuss the various types of opportunities that an agent might encounter while operating in an MAS that would not otherwise be encountered if the agent were working alone. We also discuss the critical issues at the crux of the problem: the information-sharing and decision-making issues that the agents must address to be able to provide opportunistic assistance to one another.

We begin our discussion with a set of assumptions about the agents and the multi-agent systems for our framework for multi-agent opportunism. These assumptions are:

1. We are working with an open multi-agent system, made up of a collection of heterogeneous agents, and operating in a real-world domain, that is capable of accomplishing tasks in the given domain. This assumption is essentially a baseline for the problem, and covers such things as: a task-allocation mechanism exists, the capabilities of the various individual agents and the resources available are sufficient for the given tasks, etc. While we are assuming the MAS is capable of accomplishing the assigned tasks, we are not assuming that all such tasks do indeed get accomplished. That is, there may be situations where tasks are suspended and never completed.

2. The agents are able to communicate effectively with one another. This implies

that the agents will understand the tasks, goals, and opportunity cues, etc. that are communicated to them.

3. The agents in the MAS are cooperative. We are assuming that the agents will cooperate with one another whenever possible, and that they will not intentionally interfere with one another. However, they are still free to refuse task requests.

4. The agents may have little or no knowledge in common about each other's capabilities, goals, plans, etc. This assumption makes explicit the lack of shared knowledge implied by assuming an open system of heterogeneous agents (above). This also represents a significant difference between this model and more formal teamwork models (e.g., [Tambe, 1997; Cohen and Levesque, 1990; Jennings, 1995; Grosz and Sidner, 1990; Grosz and Kraus, 1996]), which assume that the agents have a great deal of knowledge in common.

5. A subset of the agents in the MAS are capable of single-agent opportunism. Specifically, we are assuming that agents can exploit opportunities for suspended goals. This implies that at least some of the agents can suspend their own tasks/goals, recognize opportunities for the accomplishment of these suspended tasks/goals, and respond to those opportunities by re-examining and possibly resuming the suspended tasks/goals. We are not assuming any particular approach to single-agent opportunism, just that some of the agents have the capability.

4.1 Multi-Agent Opportunities

An opportunistic agent working alone needs only to be capable of exploiting opportunities for its own goals. The traditional approaches to opportunism, such as

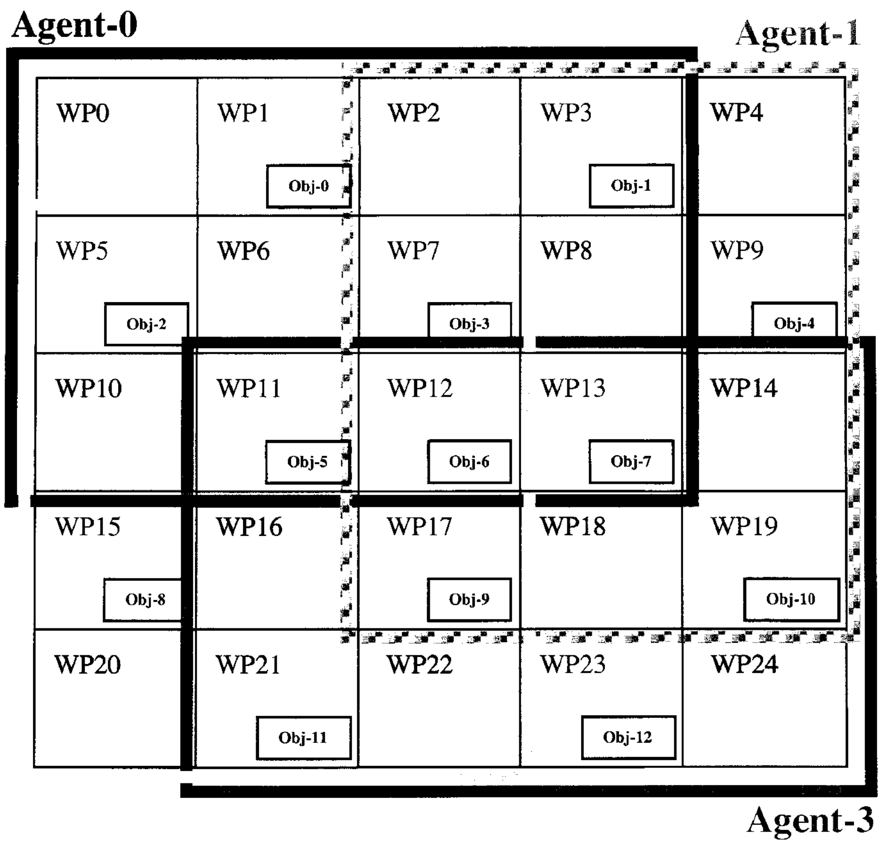

Figure 4-1: Workspace partitioning for Rovers MAS. (Repeated from Fig. 2-1)

opportunistic memory [Hammond et al., 1993] or active goals [Birnbaum and Collins, 1984], specifically address exploiting this type of opportunity. These approaches are surveyed in Section 3.2 and summarized in Figure 1-1.

Agents operating in multi-agent systems, however, should also be capable of recognizing and responding to opportunities related to the goals of *other* agents in the MAS. In order for an agent to recognize this type of opportunity, it has to know something about what the other agents are doing. This presumably means knowing about the other agents' current tasks and goals—both suspended and on-going—or at least about the cues that might identify an opportunity for the other agents' tasks and goals.

For example, consider the planetary rovers scenario from Section 2.1. Suppose rover $\mathbb{A}_0$ is assigned the goal $g_0 =$ `have-rock-sample(WP11)` (i.e., obtain a rock sample from waypoint 11), but is unable to accomplish this goal (perhaps because it has insufficient resources). Further suppose that rover $\mathbb{A}_1$ has been assigned the goal $g_1 =$ `have-rock-sample(WP12)`. Since $\mathbb{A}_1$ will be close to `WP11` when it is satisfying g_1 (see Figure 4-1), *if* it knows that $\mathbb{A}_0$ has suspended g_0, and if it has sufficient resources, it could adjust its intentions and satisfy g_0.

Similarly in the AOSN scenario, suppose that EAVE-ARIEL is given the goals of performing the temperature and salinity background survey as well as taking rock samples in areas with unusual magnetic readings. If EAVE-ARIEL decides to work on the `survey goal`, the `rock sampling goal` will get suspended. If, however, EAVE-ARISTA knows about EAVE-ARIEL'S suspended rock sampling goal, then EAVE-ARISTA can also be looking for unusual magnetic readings. If it detects any, it can notify EAVE-ARIEL of the location, and that agent can decide whether or not to pursue the opportunity.

There may be, however, suspended goals that an agent does not need to tell any other agent about. For example, consider the following situation: after collecting data for some time, one of the EAVEs enters a reduced-power state, causing a low-priority goal, g_r, to recharge its batteries to be generated. Because this is not a critical condition, g_r is suspended and the EAVE continues with its survey. Suppose, though, that sometime later the EAVE is ordered to return to MOORING-DELTA, perhaps because the survey mission has been canceled. It can take advantage of this opportunity—unexpectedly being at the mooring—to recharge its batteries (satisfying g_r), returning it to a normal power state. There is no point in the EAVE telling the other agents about its low-priority goal to recharge its batteries, since in this example the only help they could provide is the location of the mooring (which we can assume the EAVE already knows). This implies that each agent would not just want to arbitrarily broadcast knowledge about all of its goals to every other agents—a selection process would have to occur. Making this decision could be very difficult for an agent, however, since it may not have sufficient knowledge of what help the other agents could provide (e.g., how would the EAVE *know* that no other agent can assist it with g_r?). Designing a domain-independent selection process could thus be difficult.

In addition, for the agents to be able to recognize all possible opportunities for each other, it may not be sufficient for the agents to only know about each other's suspended goals. An agent might also need to know about another agent's active goals. An active goal is one an agent is currently making an effort to satisfy. There are situations where other agents can provide opportunistic assistance with an active goal, such as when an agent is searching for some object or environmental condition. For example, we know a CONVEX MOORING has as its primary (if not only) goal the detection of convective overturns. But suppose an agent doing the background survey detects a CONVEX event. That agent could notify the CONVEX MOORING'S task manager about the event, which could make the normal response of redirecting some agents to make a detailed survey of that region. As with suspended goals, a selection process is needed to determine which active goals might benefit from opportunistic assistance from other agents.

Finally, an agent operating in an MAS may also encounter a third type of opportunity. This is an opportunity that relates to the satisfaction of a shared goal—one that a subset of the agents are working together cooperatively to satisfy. For example, suppose a group of agents has been tasked with fixing up a playground, which specifically involves picking up trash and painting a fence. Suppose further that the paint for the fence has not yet arrived. The agents will all begin picking up trash,

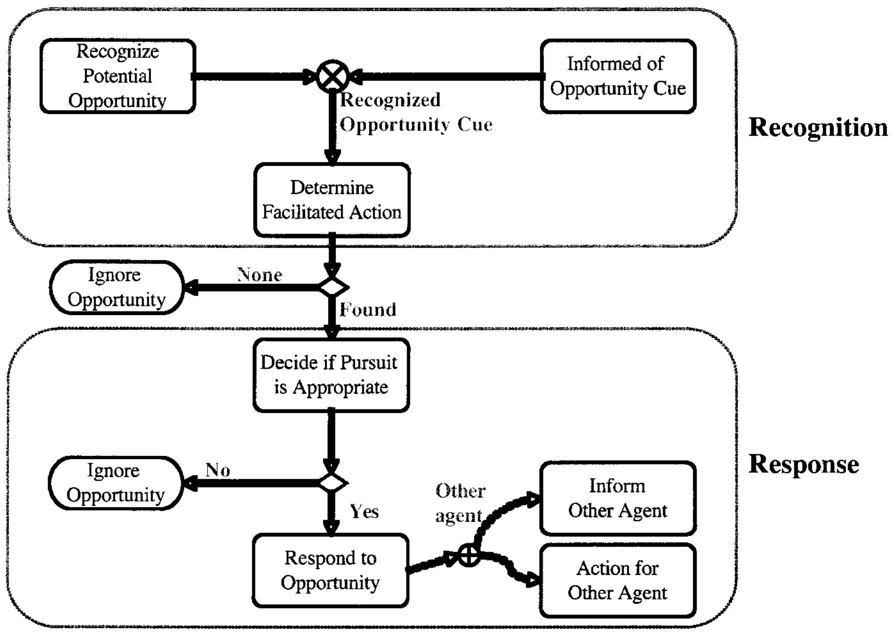
Recognize Potential Opportunity
Informed of Opportunity Cue
Recognized Opportunity Cue
Recognition
Determine Facilitated Action
Ignore Opportunity
None
Found
Decide if Pursuit is Appropriate
Ignore Opportunity
No
Yes
Other agent
Inform Other Agent
Respond to Opportunity
Action for Other Agent
Response

Figure 4-2: Decision Framework for Multi-Agent Opportunism

suspending the fence painting until the paint is delivered. Picking up trash and painting the fence are goals shared by the group. Suppose, however, that while picking up trash around a storage shed one of the agents discovers some paint left over from a previous job. This opportunity—the unexpected availability of paint—may allow the suspended fence painting shared goal to be resumed earlier than expected.

We believe that agents in a MAS can address opportunities for shared goals just as they would address opportunities for some other agent's goals, as long as at least one of the agents has an explicit representation of the shared goal. We do not, however, directly address opportunities for shared goals in this research.

4.2 Framework for Multi-Agent Opportunism

Our general framework for multi-agent opportunism is summarized in Figure 4-2. It presents a natural extension of the framework for single-agent opportunism summarized in Figure 1-1, described in Chapter 1. We will presume that we are working with a system of agents that meets the assumptions described at the beginning of this chapter. Thus, to enable these agents to assist one another in the opportunistic achievement of their goals and tasks, our framework requires the addition of the following two key capabilities:

1. The ability for one agent to *recognize* an opportunity for another agent's tasks/ goals. Just as with single-agent opportunism, the first step in exploiting an opportunity is recognizing it. The key to providing this ability in a multi-agent environment lies in providing a method for the agents to obtain sufficient information about each other's goals, or at least what is impeding those goals.

2. The ability for an agent to *respond* to a recognized opportunity for another agent's goal. The key to providing this capability lies in endowing the agents with appropriate decision-making processes so that they can determine when and how to respond to recognized opportunities. The responses that we consider are to simply notify one or more agents of the potential opportunity, or to take some action on behalf of another agent in response to the opportunity.

Examining ways to provide these capabilities in multi-agent environments has been central to this research. In the remainder of this chapter, we provide further details on these capabilities, as well as describe the specific issues that would have to be addressed by any approach to providing them. In Chapter 6 we present one such approach, along with a description of how we addressed these key issues.

4.2.1 Multi-Agent Opportunity Recognition

To enable agents to recognize potential opportunities for one another, we must provide a mechanism that will to allow them to recognize *opportunity cues* for other agents.

Recall that opportunity cues are easily identifiable changes or conditions in the environment that indicate a potential opportunity. The agents could infer information about the goals and tasks (or conceivably even the opportunity cues themselves) of other agents, and determine opportunity cues based on this inferred information. Alternatively, the agents could explicitly share opportunity-related information with one another. We will discuss each of these possibilities in turn.

One way the agents could infer opportunity-related information would be if they possess a great deal of shared knowledge about each other (e.g., their capabilities, goals, plans, etc.). As discussed in Section 3.3, the agents might possess this degree of shared knowledge if they are organized using a model of teamwork [Tambe, 1997; Jennings, 1995; Grosz and Kraus, 1996]. Since each agent would know what every other agent is doing, they could watch for opportunities for other agents. Similarly, agents could infer missing information about other agents by assuming they are homogeneous and using information about their own capabilities, goals, plans, etc. We do not, however, believe that either of these types of inference is realistic in an open, real-world MAS because the agents are likely to be heterogeneous and would probably not have sufficient knowledge in common about each other.

The agents could also infer information about each other's goals and tasks using a plan recognition mechanism. Generally speaking, agents using such a mechanism could monitor each other's activities and infer their plans and goals. Using this inferred information, one agent could then predict another agent's needs for supporting the goals it is pursuing, and assist in meeting those needs when possible. Further, every agent in the MAS does not need to do the plan and goal inference. There may be a small number of agents, such as middle agents, which naturally have access to information about the activities of other agents in the MAS that can be exploited. This information could be used to infer the plans and goals of the various agents, with the inferred information being distributed to other agents as appropriate. This type of plan recognition is, of course, itself not a simple problem.

The alternative to inferring opportunity-related information would be for the agents to explicitly share it with one another. We believe that this represents a more general and practical approach for open, real-world MASs. As such, in our study of multi-agent opportunity recognition, we have focused on the critical issues involved in the explicit sharing of opportunity-related information. These critical issues are:

- When (for which tasks or goals) should an agent request opportunistic assistance from other agents?
- Exactly what opportunity-related information should the agents share?
- How should an agent determine which other agents to share its opportunity-related information with?

Which tasks or goals?

The first issue to be considered involves when an agent should request opportunistic assistance from other agents. Since we do not want agents simply broadcasting all information about their current active and suspended goals, a selection process should be used to limit requests to only those tasks and goals that other agents are likely to be able to provide help with. In this research, we have focused on requesting opportunistic assistance for suspended goals, using the reasons the agent is suspending the goal (e.g., unmet preconditions or missing resources) to select the opportunity cues. We believe the same mechanisms discussed here should also be applicable to providing opportunistic assistance with active goals.

The reasons for suspending goals may also be used to further limit when to request opportunistic assistance. That is, we would prefer to have the agents only request opportunistic assistance for goals suspended on reasons it believe other agents in the MAS can help with. For example, when a `Rover` agent suspends a soil sampling task because it no longer has sufficient energy to go to the assigned location, it can notify other `Rovers` in the vicinity in the hope one of those agents can obtain the sample. If, however, the `Rover` suspends the task of transmitting data back to the `Lander` because the `Lander` is currently out of range, it should not request assistance from the other `Rovers`. Exactly how the agents can differentiate when other agents can provide help is is likely to require domain-specific knowledge, making a general approach difficult.

The agent must also decide when it is likely to be cost effective to request opportunistic assistance from other agents. This decision could be made using a utility function that takes into account the costs and benefits of receiving help in opportunistically accomplishing a task. If the agent only considers its own costs and benefits, then it may come to the conclusion that obtaining opportunistic assistance for *every* goal is the rational thing to do. That is, if the agent believes that it can get another agent to satisfy one of its assigned goal without expending its own resources, then it is reasonable to expect that agent to request opportunistic assistance for that goal. While this may be better for the individual agent, it could be catastrophic for the MAS as a whole, especially if no agent satisfies its assigned goals.

Since we are assuming a collection of cooperative (or at least benevolent) agents, we would prefer that the agents consider the impact of multi-agent opportunism on the overall system. To do this, they should limit their requests for opportunistic assistance to just those goals that are likely to be satisfied by some other agent with as little impact as possible on the system's (aggregate) resources. As such, an agent's decision should take into account the costs and benefits of receiving opportunistic help for itself as well as for other agents. The costs may include processing the request for assistance, watching for opportunity cues, and communication overhead. Benefits may include simply accomplishing the task early, accomplishing it at all, or meeting the preconditions of future tasks of this or other agents. Of course, while computing its own potential costs and benefits may not be difficult for an agent, determining the potential costs and benefits to other agents and to the system as a whole may be much more complicated. This is because in our heterogeneous environment, agents may not have sufficient information about one another to compute accurate values, making these decisions difficult.

What opportunity-related information to share?

Once an agent decides it can use opportunistic assistance from other agents, it must next determine exactly what information should be shared. The ultimate objective is to allow one agent to recognize opportunity cues for another agent. In general the more one agent knows about what the other agents are doing, the more help it can provide. However, since we are assuming the agents are heterogeneous, and that they may not even know about all of the other agents in the MAS, the real question is just how *little* knowledge the agents can share to still be helpful to each other.

There are three possibilities for explicitly sharing opportunity-related information: *cue sharing*, for which the task-owning agent determines and distributes the opportunity cues to the other agents; *goal sharing*, where the task-owning agent notifies the other agents of the tasks and goals for which it could use assistance; or some combination of these two, which we will call *mixed sharing*. We will examine each of these in turn. Recall that we are assuming the agents are able to communicate effectively with one another, which implies that the agents will understand the tasks, goals, and opportunity cues that are sent to them.

When using cue sharing, each agent determines the cues that might indicate an opportunity for one of its goals or tasks, and distributes these cues to the other agents. Often these cues are the unmet preconditions or missing resources of the method the agent is using to accomplish the goal or task. We use preconditions and resources as cues because they are often easily identifiable in a given domain and, as discussed in Chapter 3, they are often explicitly represented in most planning systems, making it possible to reason about them. The main advantage of cue sharing is that the agent selecting the cues is able to use contextual information about how it is planning to satisfy that goal. For example, the top-level manager of a group of AUVs might select contingency organizations to fit anticipated situations (e.g., a CONVEX event is observed). This may be the only agent that can take advantage of the processing done in the organization selection process to determine opportunity cues.

Using this approach, however, may limit the novelty of the opportunities that could be exploited. This is primarily because it is difficult for the task-owning agent to determine the appropriate level of abstraction to use for opportunity cues. A remote agent may have different abilities and experiences that would allow it to recognize ways in which a suspended task could be satisfied that the task-owning agent would not have considered. For example, if one agent has a task of cutting a rope and asks other agents to help it find a knife (because that is what its plan says to use), another agent may not know that the pair of scissors it possesses might do the job, since it does not know why the knife is needed.

When using goal sharing, knowledge of the goals for which agents could use opportunistic assistance is distributed to other agents, leaving each agent to determine cues on its own. If the remote agent computes the opportunity cues, the original contextual information is likely to be lost, but it will be able to use its own local knowledge of the world to select cues. Since we are assuming heterogeneous agents, it is possible that the remote agent could come up with and recognize a cue that the owning agent could not have. The above scissors-for-rope-cutting example demonstrates one such scenario. The task-owning agent would have to verify the novel opportunities (i.e., that it can actually use the scissors to cut the rope). But, verifying potential opportunities is a normal part of the second step in exploiting opportunities, determining the facilitated action (see Figure 4-2).

However, the use of goal sharing assumes that the remote agents are able to determine opportunity cues on their own, which may not be realistic in all MASs. We have assumed that, for multi-agent opportunism to be possible, at least some of the agents in the MAS are capable of single-agent opportunism. This implies that at least some of the agents are capable of both determining and detecting opportunity cues. However, it does not imply that they *all* can. It is possible for an MAS to contain agents that are capable of recognizing opportunity cues, but are not capable of coming up with the cues themselves (e.g., they may have the appropriate sensors, but no planning capabilities). Using strict goal sharing could thus leave out some agents that otherwise might be able to recognize potential opportunities.

Also, goal sharing may place a significant burden on the remote agents, since they must now compute a plan for satisfying the goal, determine the opportunity cues for themselves, and then watch for them. This is in contrast to the computational burden that cue sharing would impose, where the sending agent does the processing for the cue determination. In fact, even when using goal sharing, the originating agent must also compute opportunity cues for itself. This computation is not utilized by any other agent, and thus may result in wasteful redundancy.

It is possible that some combination of cue sharing and goal sharing may allow us to minimize the computational burden placed on the remote agents while maximizing the recognition of potential opportunities. One possible mixed sharing approach would require the task-owning agent to distribute both its opportunity cues and the goal to which they apply. This would allow the remote agents to examine the cues in the context of the related goal, which could enable them to recognize novel opportunities. It could also allow agents who cannot determine opportunity cues on their own to just use the ones sent to them, further increasing the possibility of noticing potential opportunities.

Which other agents?

Once an agent determines what opportunity-related information it will share, it must select the other agents in the MAS to share it with. Here there are three possibilities: the agent could tell every other agent in the MAS, it could tell only the specific agents that it believes are best suited to assist it in recognizing opportunities, or it could defer the selection process to another agent, such as a manager or a broker. Note that the opportunity-related information may also be tailored for the selected agents, based upon their capabilities.

Telling every other agent would certainly maximize the potential for recognizing an opportunity. It may also be a cost effective approach when the number of agents in the MAS is small, or when the normal communication mechanism broadcasts messages to all agents anyway. However, when the communication mechanism performs a broadcast by sending many messages (e.g., one message to every agent), the communication cost would likely be prohibitive for all but small groups of agents. Further, every agent would receive the request, including those that could not recognize or respond to potential opportunities for the request. Each of these agents would have to examine the request and determine if it could provide opportunistic assistance, increasing the overall processing cost. This processing would be wasteful if only a few of the agents were capable of providing opportunistic assistance for the request, as well as distracting the agents away from tasks they could be working on.

A potentially more cost-effective approach would be to have an agent select only those other agents that it believes would be able to recognize and respond to opportunities based on the opportunity-related information it provides. Since this information contains descriptions of goals and tasks, or specific opportunity cues derived from those goals and tasks, the selection process would use knowledge of the capabilities and available resources of the agents in order to determine which to request opportunistic assistance from. For example, a `Rover` agent with a suspended soil-sampling goal would only notify agents that can also take soil samples.

The difficult part of this selection process involves how an agent knows about the capabilities and resources of the other agents in the MAS. But this is the same information the agents need for delegating tasks to other agents in the MAS that they cannot do themselves. As such we can exploit the task allocation support mechanisms that are available for the particular coordination mechanism being used. Naturally, the details of how an agent would obtain the required information would depend upon the specific coordination mechanism in use (e.g., a matchmaker would be queried when using middle agents, a request for bids from agents with the needed capabilities would be issued when contract nets [Smith, 1980] are being used, etc.). It is also likely that obtaining this information would be more difficult using some coordination mechanisms than with others. But, by using the existing coordination mechanism, we can take advantage of the existing infrastructure and the processing that is already being done with a little additional burden on the agents.

Similarly, an agent could tell one other agent, or perhaps a small number of other agents, about its opportunity-related information. The chosen agent or agents would in turn select the other agents to distribute the this information to and forward this information. This might be desirable if a few agents in the MAS already possess knowledge of the capabilities and resources of the other agents in the MAS, reducing the need for the individual agents to conduct the selection process. Such agents would include matchmakers and brokers in MASs coordinated using middle agents, or manager agents in hierarchical organizations. Examples of both of these approaches to using a middle agent to find agents with particular capabilities are given in Section 2.1. Further, this type of deferred notification would be particularly suitable in open MASs, where the structure of the system is not known by every agent, and the organizational knowledge is likely to be maintained by only a few agents. These agents will be in the best position to get the opportunity-related information to the agents with the right capabilities, even if the structure of the MAS changes.

4.2.2 Multi-Agent Opportunity Response

After the opportunity-related information has been distributed to the appropriate agents, those agents can watch for potential opportunities to satisfy the opportunity requests. As we have previously noted, we are assuming that at least some of the agents are capable of recognizing opportunities for suspended goals, and thus we intend to exploit this ability. For those agents that can potentially provide opportunistic assistance, there are two critical issues that must be considered:

- How can an agent decide if it should provide opportunistic assistance?
- How should an agent respond to a recognized opportunity?

Should the agent respond?

Even if an agent is capable of providing opportunistic assistance, it may not be appropriate to do so. The agents must have a way to decide if supporting the goals of other agents would interfere with the achievement of their own goals, and if so, if the other goals are somehow more important.

The agents could conceivably employ a negotiation mechanism to attempt to find a globally optimal (or near optimal) re-allocation of the goals requiring opportunistic assistance, such as coalition restructuring [Shehory and Kraus, 1998] and multi-agent Markov decision processes (MMDP) [Boutilier, 1999; Goldman and Zilberstein, 2003]. While these methods will produce an allocation of goals among that agents that should lead to improved system performance, in general they require either a centralized mechanism to analyze the planned actions of the agents involved or an exchange of large amounts of plan-related information. Further, these methods are often very computationally complex, requiring an analysis of many thousands of actions and states even for simple coordination problems [Boutilier, 1999]. As such this approach is unlikely to be practical given our assumption of heterogeneous MASs with limited shared information.

The decision process an agent uses should, instead, utilize local knowledge almost exclusively, and be as computationally efficient as possible. That is, the decision itself should be opportunistic, in that the agent should analyze its current *intentions* (i.e., the actions it is currently planning to execute to satisfy its own goals) to determine if providing opportunistic assistance is feasible. As with the optimization approaches mentioned above, such a decision process should have to take into account the costs and benefits of helping satisfy some other agent's goal, possibly at the expense of not satisfying its own goals, but based on the agent's local knowledge. Because of its decision theoretic nature, even with the local restrictions a general approach to this issue could easily become intractable, so care must be taken when considering these decision processes. Limiting the decision to an analysis of current intentions, and avoiding re-planning, should help keep this tractable.

This decision process would likely be similar to the one used when an agent decides if it would be cost effective to request opportunistic assistance (Section 4.2.1). In fact these decision processes are more related than our presentation would indicate, in that knowing how an agent would decide if it should provide opportunistic assistance is likely to influence the decision of when opportunistic assistance should be requested. That is, if an agent knows what costs and benefits are considered by agents when they receive requests for opportunistic assistance, it can better decide when (for which goals) to make those requests, as well as to which other agents.

How to respond?

After an agent decides that it should provide opportunistic assistance for a particular goal, it can begin "watching for" potential opportunities using its existing opportunity recognition capabilities (recall that we are assuming that an agent that decides it can provide assistance is capable of single-agent opportunism). Once a potential opportunity is noticed, the agent must then decide how to respond. There are three general responses an agent can take: it can ignore the potential opportunity, it can simply notify the interested agent or agents of the opportunity, or it can take some action on behalf of the interested agent in response to the recognized opportunity.

When the response is to notify the interested agent or agents, the recognizing agent must determine which agents might be interested in the recognized opportunity. The obvious choice is to notify the agent that made the request for opportunistic assistance. But there may be other agents also interested in knowing about occurrence of the opportunity. Just as when an agent decides which other agents to share its opportunity-related information with (Section 4.2.1), the recognizing agent could notify every other agent in the MAS, or notify just a select set based upon their known capabilities. Similarly, a middle agent could be used both to find agent with particular capabilities, or as a means of indirect notification.

Under some circumstances, however, the agent that recognizes the opportunity may also provide further help by taking some action on behalf of the requesting agent. Continuing our rope cutting example from above, if one agent knows that another agent is looking for a knife so that it may cut a rope, and the agent has a pair of scissors (and knows how to use them), it may be able to cut the rope for the other agent and notify it that its rope-cutting task has been accomplished.

To avoid undesired side-effects, the goal-owning agent may need to include restricting conditions as part of the opportunity-related information that it shares (see Section 4.2.1). For instance, the goal-owning agent may want to specify that the end of the rope should not be damaged, preventing another agent from considering using a more drastic cutting tool, like a chain saw.

As mentioned in Chapter 1, the most straightforward way for an agent to determine if it should take some action on behalf of another agent in response to a potential opportunity is to coordinate with that other agent. This coordination could be carried out through a simple "standard operating procedure" (e.g., always take an action if the cost is less than X), or through an agreed upon communication protocol. For example, the recognizing agent could notify the goal-owning agent of the potential opportunity. The notification message could also include a description of the recognizing agent's capabilities as registered with the middle agent (thus avoiding the need to query the middle agent). If the recognizing agent has the appropriate capabilities, the goal-owning agent could then request the recognizing agent to carry out some action through the normal task request mechanism. As an added benefit, in the case where more than one agent simultaneously recognizes a potential opportunity for an agent's suspended goal, this approach would allow the goal owning agent to decide which, if any, of the agents should take some action.

There may be situations, however, where it is not possible for the recognizing agent to coordinate with the goal-owning agent. Unfortunately there is no simple domain-independent way for an agent to decide on its own when it should satisfy another agent's task. It would have to consider such things as the costs and benefits of performing the task, to both itself and the other agents. It would also have to consider if it would be appropriate to accomplish the task, as there may be side effects that it does not know about. Further, without coordination, multiple agents may end up taking action in response to a recognized opportunity, potentially causing conflict among the agents. Thus, when coordination is not possible, the best option is likely to be notification.

4.3 Summary

In this chapter we have presented our general framework for multi-agent opportunism, focusing on the the critical information-sharing and decision-making issues that must be addressed by any approach to providing this capability to an MAS. These key issues can be divided according to how they are used for supporting how agents can *recognize* and *respond* to potential opportunities. To enable agents to recognize opportunities for one another's goals, the following questions must be answered: *When should an agent request opportunistic assistance from other agents? Exactly what opportunity-related information should the agents share?* and *How should an agent determine which other agents to share its opportunity-related information with?* Similarly, for an agent to be able to respond to a potential opportunity for another agent, the following questions must be addressed: *How can an agent decide if it should provide opportunistic assistance?* and *How should an agent respond to a recognized opportunity?* In the next two chapters we describe a specific approach to incorporating multi-agent opportunism into a system of planning agents, paying particular attention to how we address these critical issues.

Chapter 5

Planning and Execution

The previous chapter describes a general framework for multi-agent opportunism, focusing on the critical knowledge-sharing and decision-making issues that would have to be addressed by any specific approach. While this general framework does provide important guidance, it is too abstract to be of practical use in a real-world MAS. In the next two chapters we describe a specific plan-based approach to multi-agent opportunism that should be applicable in most systems of agents that are capable of performing non-trivial planning tasks. The planning and execution scheme, described in this chapter, was developed especially to support opportunistic behavior of agents in various settings of shared knowledge. To preserve generality, this scheme does not assume that the agents are using any particular, or even the same, planning methodology. This is achieved by using a generic form of plan representation, with which we can represent characteristics of most (if not all) techniques used in the area of classical (i.e., STRIPS-based) AI domain-independent planning [Smith, 2003]. Further, the selected planning and execution scheme readily supports a computationally efficient form of opportunism based on predictive encoding [Patalano et al., 1993], in which potential opportunities are pre-computed and associated with existing plan elements. We present this model of multi-agent opportunism in Chapter 6.

Portions of the content of this chapter have been published in [Lawton and Domshlak, 2003, 2004a; Domshlak and Lawton, 2003, 2004].

5.1 Assumptions and Motivations

Our primary motivation for developing our plan-based approach to multi-agent opportunism was to find a way to make it easy for a collection of heterogeneous agents to share information with one another about potential opportunities for their goals. We also wanted to facilitate the decision processes agents use related to pursuing potential opportunities. In order to understand how these motivations led to the approach described in this and the following chapter, however, we must first review the assumptions we are making about the agents and the world they operate in. We start with the assumptions described in the beginning of Chapter 4:

1. We are working with an open multi-agent system, made up of a collection of heterogeneous agents, and operating in a real-world domain, that is capable of accomplishing tasks in the given domain.

2. The agents are able to communicate effectively with one another.

3. The agents in the MAS are cooperative.

4. The agents may have little or no knowledge in common about each other's capabilities, goals, plans, etc.

5. A subset of the agents in the MAS are capable of single-agent opportunism.

We will further assume that we are working with planning agents, and that the agents will have insufficient resources to satisfy all of their goals. The resource limitation assumption has been included to produce situations where the agents must decide which goals they should attempt to satisfy and which they should ignore. This naturally leads to situations where an agent can attempt to get other agents to help it satisfy the goals it is not pursuing. This is not an unrealistic assumption. Real-world domains, such as the Rovers domain from Section 2.1, often contain agents with severe resource restrictions.

Given these assumptions, we decided to exploit the information contained in the agents' plans to support multi-agent opportunism. We did *not*, however, want to require that the agents use any particular—or even the same—planning mechanism. Consequently, we needed to limit our assumptions about the plans these agents would use. To support this requirement, we determined that we needed to develop an abstract plan representation, as well as a mechanism for executing the represented plans.

The abstract plan representation had to serve two purposes. First, it would have to be able to represent plans generated by actual planners. We decided to focus on classical AI domain-independent planners, both because of their simple, yet powerful, capabilities, and because of the wide availability of existing planning systems [Smith, 2003]. This decision required that our plan abstraction be able to represent partially-ordered plan actions. Second, the plan representation would have to explicitly represent the information needed for reasoning about potential opportunities. As discussed in Section 4.2, for our model this meant that the plan abstraction would have to explicitly represent the preconditions, effects, and resource requirements of a plan's actions.

We also needed to develop an execution mechanism for our plan representation that would allow an agent to select a course of action that best uses its limited resources, and allows it to reason about potential opportunities at runtime. As mentioned above, an agent may not have sufficient resources to satisfy all of its goals. However, when the agent generates a plan to accomplish these goals, it ignores this fact. Further, since we are assuming the agent is operating in a dynamic environment, the resources used by the individual plan actions may differ from what was expected.

This means that there may be fewer (or more) resources for the remainder of the plan. Similarly, as opportunities arise to provide assistance to other agents with their suspended goals, the value of some of the goals in an agent's plan may change. An agent must be able select a course of action to take during the execution of its plan that satisfies the most valuable goals with its limited—and changing—resources. In a partially-ordered plan, this course of action is a particular total order among the various partial orders of the plan's actions. So as to impose as few restrictions on the agents as possible, we did not assume that the agents would be able to re-plan during execution. Thus the agents must be able to decide on the best course of action within their current plans without changing them. To make this possible, we needed to develop an execution mechanism for our plan representation that would enable an agent to select the next action at execution time that would most likely lead to the accomplishment of the most important goals.

The execution mechanism that we developed allows an agent to select a course of action in an approximate decision-theoretic way, using the expected action costs (described in terms of resource needs) and the externally assigned values of each goal. If during plan execution the agent realizes that it does not have the resources to accomplish all of its goals, our mechanism allows it to select the actions that will lead to the satisfaction of the most valuable goals. The mechanism is "approximate" in that that it does not consider *all* possible courses of action within a given plan. Rather, the analysis is limited to the average-cost, successful courses of action. As we discuss in the next section, without this restriction, the analysis would be intractable even for the simplest plans

To support multi-agent opportunism, our plan abstraction allows an agent to predictively encode *extra* goals that have not been assigned to it, but that may be suspended by other agents. This is a form of predictive encoding because the agent is pre-computing the opportunity cues and incorporating them into its plan, allowing the normal runtime execution mechanism to recognize and respond to potential opportunities. Initially, these extra goals would have no value in the agent's plan, thus our plan execution mechanism will allow the agent to ignore them. If, however, at runtime another agent requests opportunistic assistance on one of these extra goals, an agent that has the goal already encoded in its current plan need only increase the goal's value. The execution mechanism will automatically allow the agent to determine the best course of action. This provides an agent a consistent (and transparent) way to decide whether it should apply its resources toward one of its own goals or one of another agent that has been predictively encoded in the plan.

In the remainder of this chapter we present the details of our plan representation and execution mechanism. In the next chapter (Chapter 6), we will describe how our plan-based approach can be used to support both single-agent and multi-agent opportunism, including an explanation of how the approach addresses the key issues presented in Section 4.2. We begin our discussion with a detailed description of the assumptions we are making about the agents and the environment they operate in.

5.2 Abstract Model of Multi-Agent Planning and Execution

To represent an abstract multi-agent system, we have defined a model that both is based on models generally accepted in the MAS community and satisfies the assumptions described in the previous section. The model we are using is very similar to those used by Shehory and Kraus [1996] and Ogston and Vassiliadis [2001], but has been extended to express opportunistic behavior.

We model a multi-agent system as a collection of benevolent agents $\{\mathbb{A}_1, \cdots, \mathbb{A}_n\}$, where each agent $\mathbb{A}_i$ is associated with a set of capabilities $C_i = \{c_{i_1}, \cdots, c_{i_l}\}$, and a set of resources $R_i = \{r_{i_1}, \cdots, r_{i_m}\}$. Each r_{i_j} indicates the amount of the resource class $\mathbf{r}_j$ (e.g., time, energy, etc.) that agent $\mathbb{A}_i$ currently has. We currently only model consumable resources in this way, using plan conditions for reusable resources. For example, `on-board(`$camera_1$`, `$\mathbb{A}_2$`)` would indicate that $\mathbb{A}_2$ currently possesses the reusable resource $camera_1$.

In the related models mentioned above, the difference between the capabilities and the resources is not very clear. To clarify this point, in our model *the capabilities of an agent $\mathbb{A}_i$ correspond to the goals that can be assigned to $\mathbb{A}_i$*. For example, consider a team of three planetary rovers $\mathbb{A}_1$, $\mathbb{A}_2$ and $\mathbb{A}_3$, where both $\mathbb{A}_1$ and $\mathbb{A}_2$ are equipped with cameras, while $\mathbb{A}_3$ is not. In this group, the goal "have picture of location `L1`" is in both capabilities sets C_1 and C_2, but not in C_3. For any given domain, we will let C be the (possibly infinite) set of all goals that can be accomplished in the domain—i.e., the set of all possible capabilities. Thus for each agent $\mathbb{A}_i$, $C_i \subseteq C$.

The definition of capabilities in terms of a possibly infinite set of goals may at first appear to make our model intractable. To address this issue we assume that we are working with discrete, finite domains. For example, in the `Rovers` domain, we identify a finite, discrete set of locations for the various sampling and imaging goals. Most real-world domains can be made finite in this way. More important, in Chapter 7 we show that this assumption is only necessary for one case of our approach to multi-agent opportunism, where the shared knowledge among the agents is limited to just capabilities.

In addition to the acting agents $\{\mathbb{A}_1, \cdots, \mathbb{A}_n\}$, we assume there is an agent $\mathbb{B}$ acting as a *task broker* [Klusch and Sycara, 2001]. We use $\mathbb{B}$ to simplify the description of the information flow in the system: The primary job of $\mathbb{B}$ is simply to dispatch the goals of the system to the various agents. $\mathbb{B}$'s decision process and implementation are not important for our work. The only thing we assume about $\mathbb{B}$ is that it will assign goal g to agent $\mathbb{A}_i$ only if $g \in C_i$.

Given a set of goals $G_i = \{g_{i_1}, \cdots, g_{i_k}\} \subseteq C_i$, agent $\mathbb{A}_i$ plans for this set of goals, and begins the execution of the generated plan $\mathcal{P}$. To better support more realistic domains, we assume that each goal g is annotated with its value V_g (which may be parameterized by various features such as deadlines, energy remaining after completion, etc.), and that the planning process takes these value functions into account. Goal value functions will be explained in more detail in later Section 5.3.1. Further, each plan action is assumed to have its resource consumption associated with it. In most realistic domains, resource consumption will not be certain and thus will likely be represented using a probability distribution.

During the execution of a plan $\mathcal{P}$ by some agent $\mathbb{A}_i$, several aspects of the world could change, impacting the relative attractiveness of $\mathcal{P}$. For instance, any of the following may occur:

- $\mathbb{A}_i$ is assigned an additional goal $g_{i_{k+1}}$ by $\mathbb{B}$.
- The value V_g for some $g \in G_i$ changes (positively or negatively).
- Some of the goals in G_i becomes unreachable with respect to $\mathcal{P}$.
- The current resource levels are not as expected after executing some part of $\mathcal{P}$. This may be due to such things as the resource consumption by the part of $\mathcal{P}$ executed so far has been significantly different (positively or negatively) from what was expected during planning, the unexpected addition of new resources, etc.

In such cases, $\mathbb{A}_i$ should revisit its current course of action, possibly updating its set of active goals, and suspending goals it determines are no longer feasible. Normally, these suspended goals would be returned to the broker for redistribution to other agents in the multi-agent system, or possibly abandoned completely. In our model, though, $\mathbb{A}_i$ may attempt to satisfy these goals opportunistically by fitting them into some other part of its current plan, or into the current plan of another agent in the multi-agent system, without re-planning. Our approach to this is discussed in the following sections.

5.3 Basic Model for Planning and Execution

As we introduced in Section 5.1, our aim is to develop an abstract plan representation and corresponding execution mechanism that supports flexible, opportunistic behavior without execution-time re-planning. Instead of re-planning, we would like the agents to adapt to potential opportunities by reasoning about their current plans. Our approach, described in this section, was significantly inspired by the work on contingency planning for planetary rovers [Dearden et al., 2002]. Our main motivation for adopting this formalism was to stay as close to real-world domains as possible.

5.3.1 Planning

Following Dearden et al. [2002], we assume that there are two parts of a planning problem, qualitative and quantitative. The qualitative part of the problem is described using the propositional STRIPS formalism in which both positive and negative preconditions are allowed[1] [Bylander, 1994]. By using a propositional formalism, we can be sure that our example planning domains will have a discrete, finite set of goals, meeting our assumption discussed in the previous section. Each agent is associated with a description of its state (represented as a conjunct of valid propositions), a set of goal propositions to be achieved, and set of possible actions, each of which is characterized by its preconditions and effects. In what follows, we denote the preconditions and effects of action A by $\mathsf{prec}(A)$ and $\mathsf{effects}(A)$, respectively.

The quantitative part of the problem is described by the resource consumptions

[1]This is also exactly the formalism used for the first level of planning competition [Fox and Long, 2002].

of the actions and the values associated with each goal. Resource consumption is modeled using consumption probability distributions associated with each action. Goal values represent the contribution to the *system* of achieving the goal (i.e., its "global" value), and not just to the individual agent. Again following Dearden et al. [2002], we use a very general representation for goal values. They are modeled as functions of the resources available after achieving these goals. For example, suppose $\mathbf{r}$ is the only resource used by an agent. It may represent such things as the amount of energy the agent has, or the amount time remaining before some deadline. If the value function $V_g(\mathbf{r})$ of goal g is:

$$V_g(\mathbf{r}) = \begin{cases} 0, & \mathbf{r} \leq 0 \\ 10, & \mathbf{r} > 0 \end{cases}$$

then the value of g is 10 if we can achieve g with some amount of $\mathbf{r}$ remaining (i.e., the agent hasn't expended all of its energy, a key deadline hasn't passed, etc), and 0, otherwise. Of course, this representation may be unnecessary in many domains, thus goal values may also simply be constant functions (e.g., $V_g(\mathbf{r}) = 10$) that are independent of $\mathbf{r}$.

As with Dearden et al. [2002], in the planning stage we ignore the quantitative part of the problem, solving the STRIPS-based problem as if there is no resource consumption or difference in the importance of the goals. However, the difference between our model of planning and that used by Dearden et al. [2002] is that their model corresponds to the first stage of the Graphplan algorithm [Blum and Furst, 1997], resulting in a *planning graph*, while we are interested in a structure that has properties of both a planning graph and a *partial order plan* [McAllester and Rosenblitt, 1991]), without tying us to a specific formalism. In order to introduce the plan representation structure used in our framework, we will first briefly describe partial order plans and planning graphs. Note that we are not interested here in the planning

```
SampleRock(p)
    PRECONDITIONS: location(p) ∧ At(p)
    EFFECTS: hs(p)

TakePicture(p)
    PRECONDITIONS: location(p) ∧ At(p)
    EFFECTS: hp(p)

Navigate(p, q)
    PRECONDITIONS: location(p) ∧ location(q) ∧ At(p)
    EFFECTS: ¬At(p) ∧ At(q)
```

Figure 5-1: Operators for Rovers example.

process itself, but rather the representation of the plan created as a result of planning.

A partial order plan is a tuple $\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle$, where $\mathcal{A}$ is a set of actions, $\mathcal{O}$ is a set of *ordering constraints* over $\mathcal{A}$, and $\mathcal{L}$ is a set of *causal links*. For example, if $\mathcal{A} = \{A_1, A_2, A_3\}$ then $\mathcal{O}$ might be the set $\{A_1 < A_3, A_2 < A_3\}$. These constraints specify a plan in which A_3 is necessarily the last operator, but do not commit to a particular order on A_1 and A_2. Naturally, the set of ordering constraints must be consistent, i.e., there must exist some total order satisfying them. A causal link has the form $A_i \xrightarrow{q} A_j$, where A_i and A_j are actions and q is a proposition. Such a causal link denotes the fact that A_i produces (i.e., has the effect) q which is required by A_j (i.e., used to satisfy a precondition of A_j). Ordering constraints are imposed among the actions to ensure that other actions do not threaten the causal links.

For example, consider a simple problem that is based on the Rovers domain used in the recent planning competition [Fox and Long, 2002]. Three of the operators

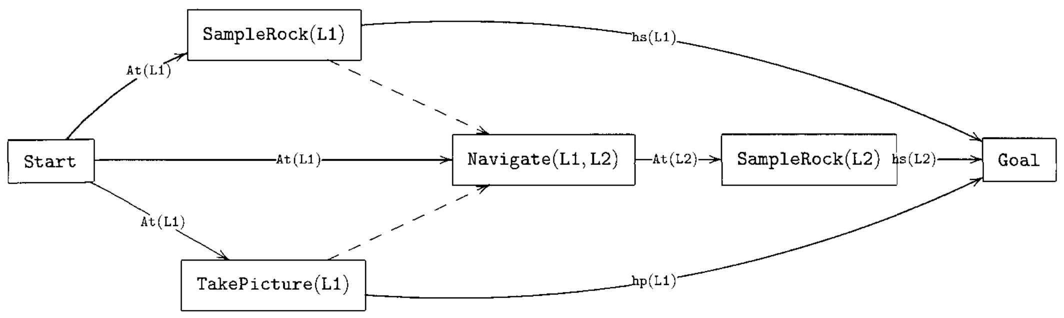

Figure 5-2: Partial order plan for the Rovers example.

available to an agent (simplified for ease of presentation) are shown in Figure 5-1, where the propositions hs(p) and hp(p) stand for "have rock sample" and "have picture" from location p, respectively. The relevant part of the initial state of the agent is $\mathtt{At(L1)} \wedge \neg\mathtt{hs(L1)} \wedge \neg\mathtt{hp(L1)} \wedge \neg\mathtt{hs(L2)}$, while the goals are hs(L1), hp(L1), and hs(L2). Figure 5-2 presents the relevant part of a possible partial order plan for this problem, where the solid edges represent the causal links (labeled with the corresponding propositions), and the dashed edges represent the ordering constraints that are not trivially entailed by the causal links[2]. It is easy to see that there are two totally ordered plans consistent with this partial order plan, and the only difference between them is the relative positions of SampleRock(L1) and TakePicture(L1).

In contrast, a planning graph is a directed graph arranged into alternating levels of proposition nodes and action nodes. To be precise, in what follows we describe a *solved* planning graph, which is the representation of a plan resulting from the GraphPlan algorithm. As we are only interested in this end product, we will omit some details (e.g., mutual exclusion relations) that are only relevant during plan generation.

Level 0 of a planning graph contains all of the relevant proposition nodes that are

[2]Start and Goals nodes are dummy actions acting as a producer of the initially valid propositions and the consumer of the goal propositions, respectively.

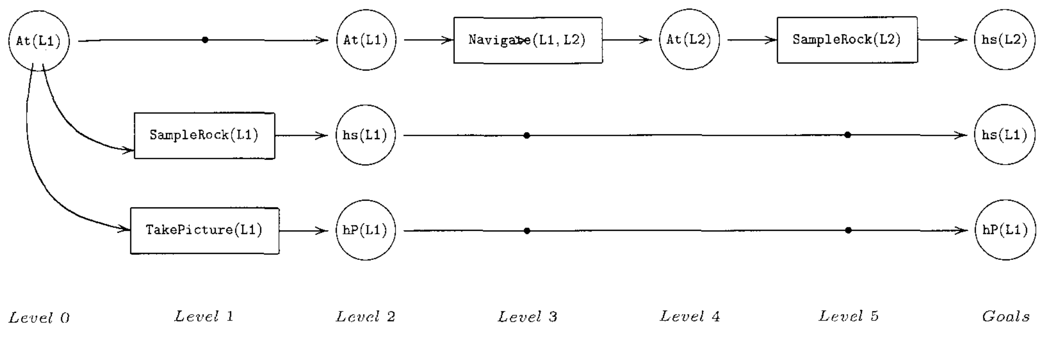

Figure 5-3: Solved Planning Graph for the Rovers example.

valid in the initial state. Nodes at level 1 are actions whose preconditions are met in the initial state. An edge in a planning graph that connects a proposition node q in level i to an action node A in level $i+1$ represents a precondition link. Similarly, edges from an action node A in level j to proposition nodes in level $j+1$ represent the effects of A. A path from action A_i at level k to action A_j at level $k+2$ through proposition q at level $k+1$ is equivalent to a causal link $A_i \xrightarrow{q} A_j$ in a partial order plan.

The levels of a planning graph are used to impose ordering constraints between the actions. An action A at level i cannot be executed until all actions at level $i-2$ have been executed. Actions at the same level may be executed in any order. NO-OP actions are used to carry unchanged proposition nodes forward in the graph.

Continuing with the Rovers example described above, Figure 5-3 presents the relevant part of the solved planning graph for this problem. Bullets (e.g "•") are used to represent NO-OP actions. Notice that even though planning graphs make explicit which actions may be executed "in parallel" at a given time, like partial order plans they only describe a partial ordering of these actions. Executing a planning graph still involves finding a valid total ordering of the actions.

To achieve a higher degree of flexibility for supporting plan-based multi-agent opportunism, we have merged characteristics of both partial order plans and planning

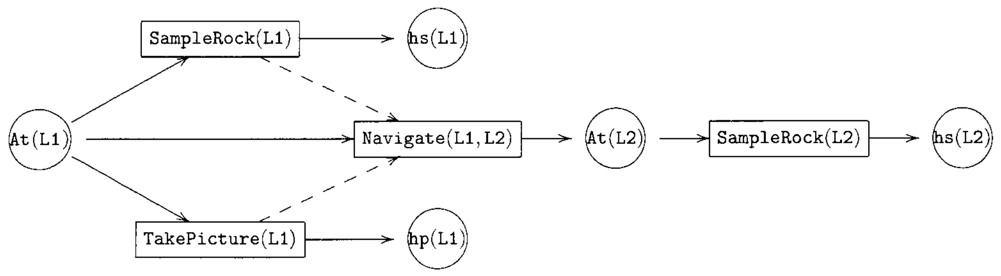

Figure 5-4: Partial order plan graph (POPG) for the `Rovers` example.

graphs into a structure that we refer to as a *partial order plan graph* (POPG). A POPG may be viewed as a partial order plan where the preconditions and effects are explicitly represented as propositions nodes. The action nodes are exactly the nodes of the original partial order plan (excluding the dummy start and goal nodes), while each proposition node is created by contracting the causal links corresponding to effectively the same proposition. Similarly, a POPG may be viewed as a solved planning graph where the levels have been collapsed. Like a planning graph, a POPG contains two types of nodes: proposition nodes and action nodes. However, in contrast to planning graphs, a POPG is not a leveled graph, and the alternative schedules of the plan are captured by explicit ordering constraints. The POPG for the running example is depicted in Figure 5-4.

Now that we have described our abstract plan representation, we can return to the discussion on our model of planning. We assume that the qualitative part of the actual plan for a given problem instance is generated by an external planner and converted into a POPG, creating to a "skeleton" plan. This plan structure is then augmented with the quantitative aspects of the problem: The actions are annotated with their resource consumption distributions and the goal nodes are annotated with their value functions. The resulting structure is ready to be used in the execution stage.

5.3.2 Execution

Given an initial state, a set of resources, and a POPG representation of the plan enriched by the quantitative information about resource consumption and values of the different goals, the agent can begin to execute its plan. At each intermediate state s during the execution, the agent must make a decision about the next action to perform. As the agent is provided with a partially-ordered plan, there may be more than one action applicable in the state s, given the current set of resources. For instance, in the initial state of the running example, both actions `SampleRock(L1)` and `TakePicture(L1)` are consistent with the plan represented by the POPG in Figure 5-4. In addition, however, observe that if we ignore the ordering links for the moment, the action `Navigate(L1,L2)` can be performed in the initial state as well. Clearly, if the agent has sufficient resources to accomplish all of its goals, performing this action will be irrational since the agent will lose its ability to achieve the goals `hs(L1)` and `hp(L1)`. On the other hand, if resources are limited and the goal `hs(L2)` is very important, it might be the case that the right thing to do is to forget about `hs(L1)` and `hp(L1)`, and to perform `Navigate(L1,L2)`, trying to achieve `hs(L2)` with as little risk as possible.

In general, deciding upon which course of action to take when there are several alternatives requires an estimate of how much value could be gained by each. Computing these values exactly is intractable, as it requires taking into account not only the uncertain resource consumption of each action to be executed in the future, but also capturing in the model all possible results of potential future failures. However, adopting the method in which the resource consumption distributions are abstracted by Dearden et al. [2002], we outline an approximation method for such a value estimation.

An agent executes its plan by repeatedly selecting and executing one of its plan actions, A, that can be executed in the current state, s, with the current resources, ρ. Executing A updates state s, creating the new (intermediate) state $\sigma(s, A)$, and reduces the resources ρ appropriately. For ease of presentation, in what follows we assume that there is only one resource, and ρ is the amount of this resource available when the agent is selecting its next action, A. When estimating the value of performing a given action, instead of taking into account the precise resource consumption distributions, each action A is annotated with (i) its expected consumption $\mu(A)$ and (ii) the minimal resource level $\min(A)$ required to allow A to be executed. This way, if $\rho \geq \min(A)$, then A can be executed, its execution is assumed to be successful, and its resource consumption is expected to be $\mu(A)$. Otherwise, if $\rho < \min(A)$, then the action is not executable, due to the risk associated with its failure. Note that, while $\mu(A)$ is defined purely by the resource consumption distribution of A, $\min(A)$ must be specified explicitly as part of the problem specification.

Let $\mathsf{actions}(\mathcal{P}, s, \rho)$ (specified by Eq. 5.1) be the set of actions in the plan represented by POPG $\mathcal{P}$ that are executable in state s with ρ amount of resource available.

$$\mathsf{actions}(\mathcal{P}, s, \rho) = \{A \in \mathcal{P} \mid \mathsf{prec}(A) \in s \ \wedge \ \rho \geq \min(A)\} \tag{5.1}$$

The value $U(\mathcal{P}, s, \rho)$ represents our estimate of how much value could be gained by executing plan $\mathcal{P}$ with ρ amount of resource, starting at the state s. As is specified by Eq. 5.2, the value $U(\mathcal{P}, s, \rho)$ is computed by finding the maximum expected value that could be obtained by selecting each action $A \in \mathsf{actions}(\mathcal{P}, s, \rho)$ as the next action to be executed. That is, for any action $A \in \mathsf{actions}(\mathcal{P}, s, \rho)$, executing A in s with ρ will produce several valid total orders of the partially-ordered actions in $\mathcal{P}$. Each total order will result in some value being obtained through the satisfaction of some or all of the goals in $\mathcal{P}$. The maximum expected value that could be obtained by executing A is thus the maximum of the values obtained by the various total orders starting with A. Similarly, the maximum of the expected values of executing each action $A \in \mathsf{actions}(\mathcal{P}, s, \rho)$ is the expected value of the plan, $U(\mathcal{P}, s, \rho)$.

$$\begin{aligned} U(\emptyset, s, \rho) &= 0 \\ U(\mathcal{P}, s, \rho) &= \max_{A \in \mathsf{actions}(\mathcal{P}, s, \rho)} \left[\alpha(\mathcal{P}, A, s, \rho) + \beta(A, \rho)\right] \end{aligned} \tag{5.2}$$

$$\begin{aligned} \alpha(\mathcal{P}, A, s, \rho) &= U\left(\mathsf{Refine}(\mathcal{P}, A, s), \sigma(s, A), \rho - \mu(A)\right) \\ \beta(A, \rho) &= \sum_{g \in \mathsf{effects}(A)} V_g(\rho - \mu(A)) \end{aligned} \tag{5.3}$$

As specified in Eq. 5.2, the expected value of each action $A \in \mathsf{actions}(\mathcal{P}, s, \rho)$ is computed as the sum of two components. The β-component is simply the value of the goals achieved directly by executing A. The α-component is the value of the remaining plan that the agent will have after performing A. That is, it is the maximum of the expected values obtained by the various total orders of $\mathcal{P}$ starting with, but not including, A.

The α- and β-components are specified in Eq. 5.3 by $\alpha(\mathcal{P}, A, s, \rho)$ and $\beta(A, \rho)$, respectively. The β-component is computed by $\beta(A, \rho)$ in the obvious way: by computing the sum of the values of each goal g, $V_g(\rho)$, that are satisfied as a direct result of performing action A (i.e., $g \in \mathsf{effects}(A)$) with resource ρ. The α-component is computed by $\alpha(\mathcal{P}, A, s, \rho)$ by recursively finding the value of the part of the plan $\mathcal{P}$ (i.e., the subgraph of POPG $\mathcal{P}$) remaining after performing action A in state s. This sub-plan is constructed by the procedure $\mathsf{Refine}(\mathcal{P}, A, s)$, which appears in Figure 5-5. Refine essentially simulates performing action A in state s producing state $\sigma(s, A)$, and removes the parts of $\mathcal{P}$ that would no longer be reachable from $\sigma(s, A)$.

Execution Example 1

To illustrate the process, we will first present an example where there are adequate resources to accomplish all of a plan's goals. Consider the POPG of the running

```
Refine($\mathcal{P}, A, s$)
1. Compute $\sigma(s, A)$.
2. Remove $A$ from $\mathcal{P}$, together with all its outgoing edges.
3. Until no more nodes can be removed, iteratively remove:
    • All the proposition nodes $p$ (together with their outgoing edges), such that $p \notin \sigma(s, A)$, and the node $p$ has no incoming edges, and
    • All the action nodes $A'$, such that for at least one of the preconditions $q \in \mathsf{prec}(A')$ there is no proposition node associated with $q$ and having an outgoing edge to $A'$.
4. Return the updated plan $\mathcal{P}$.
```

Figure 5-5: Procedure for updating POPG $\mathcal{P}$ after performing action A.

example (Figure 5-6), which we will call $\mathcal{P}_1$. Let the the value functions of the goals be the constant functions: $V_{\mathtt{hs(L1)}} = 2$, $V_{\mathtt{hp(L1)}} = 2$, and $V_{\mathtt{hs(L2)}} = 10$, and the resource consumptions of the actions be abstracted as follows (with the expected resource use, $\mu(A)$, equal to the minimum resource level, $\min(A)$, for ease of presentation):

A	$\mu(A) = \min(A)$
`SampleRock(L1)`	3
`TakePicture(L1)`	2
`Navigate(L1,L2)`	10
`SampleRock(L2)`	5

For this example, suppose that the initial resource level $\rho_1 = 20$, and the initial state is simply $s_1 = \{\mathtt{At(L1))}\}$, as shown in Figure 5-6. To begin executing this plan, the agent must select the first action to be performed. It would accomplish this by first computing the estimated value of $\mathcal{P}_1$ using $U(\mathcal{P}_1, s_1, \rho_1)$ as specified in Eq. 5.2, and then selecting the action $A \in \mathsf{actions}(\mathcal{P}_1, s_1, \rho_1)$ that achieves that estimated value. The computation of $U(\mathcal{P}_1, s_1, \rho_1)$ begins by determining the available actions,

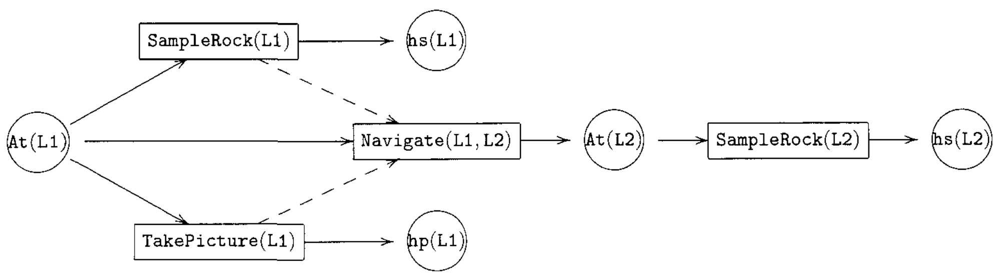

Figure 5-6: $\mathcal{P}_1$: POPG for the Rovers example (repeated from Fig. 5-4).

which in this example are:

$$\text{actions}(\mathcal{P}_1, s_1, \rho_1) = \{\texttt{SampleRock(L1)}, \texttt{TakePicture(L1)}, \texttt{Navigate(L1,L2)}\}$$

Thus, the initial call to $U(\mathcal{P}_1, s_1, \rho_1)$ will be computed as:

$$U(\mathcal{P}_1, s_1, \rho_1) = \max \begin{cases} \alpha(\mathcal{P}_1, \texttt{SampleRock(L1)}, s_1, \rho_1) + \beta(\texttt{SampleRock(L1)}, \rho_1) \\ \alpha(\mathcal{P}_1, \texttt{TakePicture(L1)}, s_1, \rho_1) + \beta(\texttt{TakePicture(L1)}, \rho_1) \\ \alpha(\mathcal{P}_1, \texttt{Navigate(L1,L2)}, s_1, \rho_1) + \beta(\texttt{Navigate(L1,L2)}, \rho_1) \end{cases} \quad (5.4)$$

The algorithm will consider these actions one at a time, beginning with `SampleRock(L1)`. Since `SampleRock(L1)` satisfies the goal `hs(L1)`, $\beta(\texttt{SampleRock(L1)}, \rho_1) = V_{\texttt{hs(L1)}} = 2$. For the computation of $\alpha(\mathcal{P}_1, \texttt{SampleRock(L1)}, s_1, \rho_1)$, we will call:

$$s_2 = \sigma(s_1, \texttt{SampleRock(L1)}) = \{\texttt{At(L1)}, \texttt{hs(L1)}\}$$

and

$$\rho_2 = \rho_1 - \mu(\texttt{SampleRock(L1)}) = 20 - 3 = 17$$

Thus, the computation of $\alpha(\mathcal{P}_1, \texttt{SampleRock(L1)}, s_1, \rho_1)$ is determined according to Eq. 5.3 as:

$$U_1 = U(\text{Refine}(\mathcal{P}_1, \texttt{SampleRock(L1)}, s_1), s_2, \rho_2)$$

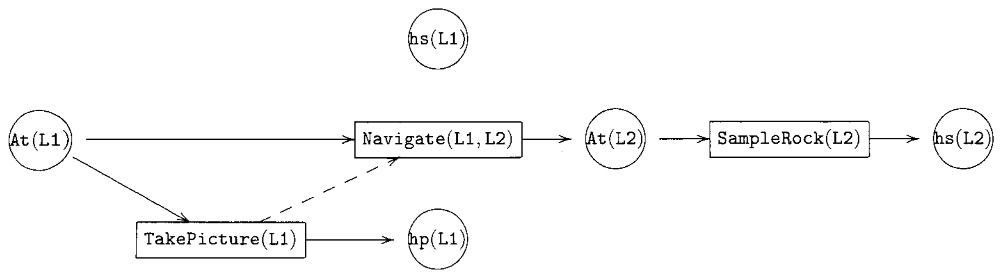

Figure 5-7: $\mathcal{P}_2$: Result of calling Refine($\mathcal{P}_1$, SampleRock(L1), s_1).

The call to Refine($\mathcal{P}_1$, SampleRock(L1), s_1) will simply remove SampleRock(L1) from $\mathcal{P}_1$, along with its links, producing the POPG $\mathcal{P}_2$ shown in Figure 5-7. No proposition nodes are removed since they are all either in s_2 or have incoming edges. No other action nodes are removed since none of their precondition nodes have been removed, and thus they are all still reachable in the graph.

The computation of $U_1 = U(\mathcal{P}_2, s_2, \rho_2)$ proceeds as before by first determining the available action, actions($\mathcal{P}_2, s_2, \rho_2$), which is the set { TakePicture(L1), Navigate(L1, L2) }. Thus, U_1 will be computed as:

$$U(\mathcal{P}_2, s_2, \rho_2) = \max \begin{cases} \alpha(\mathcal{P}_2, \texttt{TakePicture(L1)}, s_2, \rho_2) + \beta(\texttt{TakePicture(L1)}, \rho_2) \\ \alpha(\mathcal{P}_2, \texttt{Navigate(L1,L2)}, s_2, \rho_2) + \beta(\texttt{Navigate(L1,L2)}, \rho_2) \end{cases} \tag{5.5}$$

As before, these actions are considered one at a time, starting with TakePicture(L1). Since TakePicture(L1) satisfies the goal hp(L1), $\beta(\texttt{TakePicture(L1)}, \rho_2) = V_{\texttt{hp(L1)}} = 2$. For the computation of $\alpha(\mathcal{P}_2, \texttt{TakePicture(L1)}, s_2, \rho_2)$, we will call:

$$s_3 = \sigma(s_2, \texttt{TakePicture(L1)}) = \{\texttt{At(L1)}, \texttt{hs(L1)}, \texttt{hp(L1)}\}$$

and

$$\rho_3 = \rho_2 - \mu(\texttt{TakePicture(L1)}) = 17 - 2 = 15$$

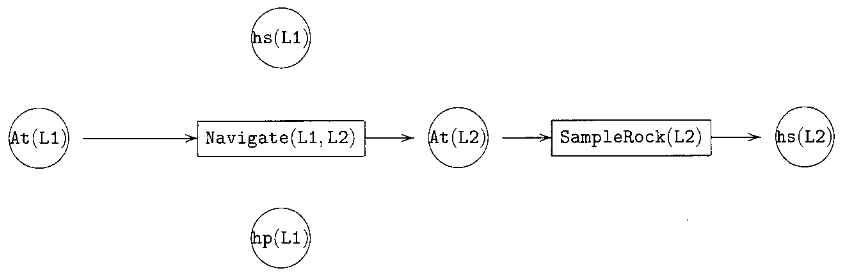

Figure 5-8: $\mathcal{P}_3$: Result of calling Refine($\mathcal{P}_2$, `TakePicture(L1)`, s_2).

Thus, the computation of $\alpha(\mathcal{P}_2, \texttt{TakePicture(L1)}, s_2, \rho_2)$ is determined according to Eq. 5.3 as:

$$U_2 = U(\text{Refine}(\mathcal{P}_2, \texttt{TakePicture(L1)}, s_2), s_3, \rho_3)$$

The call to Refine($\mathcal{P}_2$, `TakePicture(L1)`, s_2) will simply remove `TakePicture(L1)` from $\mathcal{P}_2$, along with its links, producing the POPG $\mathcal{P}_3$ shown in Figure 5-8. No proposition nodes are removed since they are all either in s_3 or have incoming edges. No other action nodes are removed since none of their precondition nodes have been removed, and thus they are all still reachable in the graph.

From Figure 5-8 we can readily see that there are no further choices in $\mathcal{P}_3$ about what actions to take. Executing all of the remaining actions in $\mathcal{P}_3$, `Navigate(L1,L2)` and `SampleRock(L2)` will require a resource level of $\mu(\texttt{Navigate(L1,L2)}) + \mu(\texttt{SampleRock(L2)}) = 10 + 5 = 15$. Since $\rho_3 = 15$, it is expected that there are sufficient resources to complete this plan. Without going through the details, it is easy to see that the the value of U_2 is simply the value of the remaining goal, `hs(L2)`, which will be computed as $\beta(\texttt{SampleRock(L2)}, \rho_i) = V_{\texttt{hs(L2)}} = 10$, where $\rho_i = \rho_3 - \mu(\texttt{Navigate(L1,L2)}) - \mu(\texttt{SampleRock(L2)})$.

The algorithm would next backtrack to Eq. 5.5 and consider `Navigate(L1,L2)` as the next action to take in plan $\mathcal{P}_2$ (i.e., the second step of the initial plan). It would

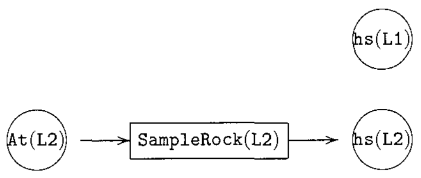

Figure 5-9: $\mathcal{P}_4$: Result of calling $\mathsf{Refine}(\mathcal{P}_2, \texttt{Navigate(L1,L2)}, s_2)$.

thus compute:

$$\alpha(\mathcal{P}_2, \texttt{Navigate(L1,L2)}, s_2, \rho_2) + \beta(\texttt{Navigate(L1,L2)}, \rho_2)$$

Since `Navigate(L1,L2)` does not directly satisfy any goals, $\beta(\texttt{Navigate(L1,L2)}, \rho_2) = 0$. For the computation of $\alpha(\mathcal{P}_2, \texttt{Navigate(L1,L2)}, s_2, \rho_2)$, we will call:

$$s_4 = \sigma(s_2, \texttt{Navigate(L1,L2)}) = \{\texttt{At(L2)}, \texttt{hs(L1)}\}$$

and

$$\rho_4 = \rho_2 - \mu(\texttt{Navigate(L1,L2)}) = 17 - 10 = 7$$

Thus, the computation of $\alpha(\mathcal{P}_2, \texttt{Navigate(L1,L2)}, s_2, \rho_2)$ is determined according to Eq. 5.3 as:

$$U_4 = U(\mathsf{Refine}(\mathcal{P}_2, \texttt{Navigate(L1,L2)}, s_2), s_4, \rho_4)$$

The call to $\mathsf{Refine}(\mathcal{P}_1, \texttt{Navigate(L1,L2)}, s_1)$ will first remove `Navigate(L1,L2)` from $\mathcal{P}_1$, along with its links. The proposition node `At(L1)` is next removed from $\mathcal{P}_1$, since it has no incoming links and is not in s_4. Because `At(L1)` is a precondition of `TakePicture(L1)`, that action, along with its links, is also removed from $\mathcal{P}_2$. Finally, since there are now no incoming links to the proposition nodes `hp(L1)`, this node is removed as well. The resulting plan, $\mathcal{P}_4$, is shown in Figure 5-9.

Again it is not difficult to see from Figure 5-9 that the estimated value of this refined plan, U_4, is simply $V_{\texttt{hs(L2)}} = 10$. Since both parts of Eq. 5.5 have been computed, the algorithm would determine that the estimated value of $\mathcal{P}_2$, $U(\mathcal{P}_2, s_2, \rho_2) =$

$\max((U_2 + \beta(\texttt{TakePicture(L1)}, \rho_2)), (U_4 + \beta(\texttt{Navigate(L1,L2)}, \rho_2))) = \max((10 + 2), (10 + 0)) = 12$.

The algorithm would next backtrack to complete the computation of the first choice in Eq. 5.4 (i.e., if `SampleRock(L1)` were selected to be the first action). This is computed as:

$$\begin{aligned}
U(\mathcal{P}_1, s_1, \rho_1) &\geq U_1 + \beta(\texttt{SampleRock(L1)}, \rho_1) \\
&\geq U_2 + \beta(\texttt{TakePicture(L1)}, \rho_2) + \beta(\texttt{SampleRock(L1)}, \rho_1) \\
&\geq \beta(\texttt{SampleRock(L2)}, \rho_i) + \beta(\texttt{TakePicture(L1)}, \rho_2) + \beta(\texttt{SampleRock(L1)}, \rho_1) \\
&\geq V_{\texttt{hs(L2)}} + V_{\texttt{hp(L2)}} + V_{\texttt{hs(L1)}} \\
&\geq 10 + 2 + 2 \\
&\geq 14
\end{aligned}$$

That is, at this point in the evaluation of $\mathcal{P}_1$, the algorithm has determined that this plan can obtain at least a value of 14 with the given initial resources $\rho_1 = 20$. Note that since this is the sum of the values of all of the goals in $\mathcal{P}_1$, we know that this is also the maximum value that could be obtained by this plan. Thus the algorithm could stop its analysis at this point, even without considering the other possible courses of action through $\mathcal{P}_1$, since `SampleRock(L1)` could be safely selected as the first plan action to be executed. We do not, however, currently use shortcuts like this (although it is planned for future work), and so we will continue describing the plan analysis.

It is not difficult to see that selecting `TakePicture(L1)` as the first action would also lead to the value estimate $U(\mathcal{P}_1, s_1, \rho_1) = 14$, since the actions `TakePicture(L1)` and `SampleRock(L1)` do not interfere with one another's preconditions. As such, we will not describe the details of the computation of the second part of Eq. 5.4. However, let us briefly look at the selection of `Navigate(L1,L2)` as the first action to execute from $\mathcal{P}_1$, (i.e., the third part of Eq. 5.4), which would be computed as:

$$\alpha(\mathcal{P}_1, \texttt{Navigate(L1,L2)}, s_1, \rho_1) + \beta(\texttt{Navigate(L1,L2)}, \rho_1)$$

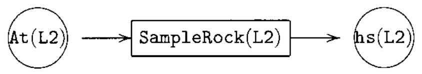

Figure 5-10: $\mathcal{P}_5$: Result of calling $\textsf{Refine}(\mathcal{P}_1, \texttt{Navigate(L1,L2)}, s_1)$.

Since Navigate(L1,L2) does not directly satisfy any goals, $\beta(\texttt{Navigate(L1,L2)}, \rho_1) = 0$. For the computation of $\alpha(\mathcal{P}_1, \texttt{Navigate(L1,L2)}, s_1, \rho_1)$, we will call:

$$s_5 = \sigma(s_1, \texttt{Navigate(L1,L2)}) = \{\texttt{At(L2)}\}$$

and

$$\rho_5 = \rho_1 - \mu(\texttt{Navigate(L1,L2)}) = 20 - 10 = 10$$

Thus, the computation of $\alpha(\mathcal{P}_1, \texttt{Navigate(L1,L2)}, s_1, \rho_1)$ is determined according to Eq. 5.3 as:

$$U_5 = U(\textsf{Refine}(\mathcal{P}_1, \texttt{Navigate(L1,L2)}, s_1), s_5, \rho_5)$$

The call to $\textsf{Refine}(\mathcal{P}_1, \texttt{Navigate(L1,L2)}, s_1)$ will first remove Navigate(L1,L2) from $\mathcal{P}_1$, along with its links. The proposition node At(L1) is next removed from $\mathcal{P}_1$, since it has no incoming links and is not in s_5. Because At(L1) is a precondition of both SampleRock(L1) and TakePicture(L1), these actions, along with their links, are also removed from $\mathcal{P}_1$. Finally, since there are no incoming links to the proposition nodes hs(L1) and hp(L1), these nodes are removed as well. The resulting plan, $\mathcal{P}_5$, is shown in Figure 5-10.

From Figure 5-10 we can easily see that the best value this plan could possibly obtain is $V_{\texttt{hs(L2)}} = 10$, since the other two goals have been removed from the plan. Thus, the algorithm will complete the computation of Eq. 5.4 as:

$$U(\mathcal{P}_1, s_1, \rho_1) = \max \begin{cases} 14 & if A = \texttt{SampleRock(L1)} \\ 14 & if A = \texttt{TakePicture(L1)} \\ 10 & if A = \texttt{Navigate(L1, L2)} \end{cases}$$

The agent could then select either `SampleRock(L1)` or `TakePicture(L1)` as the first action to perform, as both produce the maximum value for the plan. In this case, selecting `Navigate(L1,L2)` as first action for $\mathcal{P}_1$ would be irrational, since it precludes the accomplishment of some of the plan's goals, and thus leads to a lower estimated value.

Execution Example 2

For our second example, we will again consider the problem from Example 1, except now we will assume that the available resources are *not* sufficient to satisfy all of the plan's goals. Instead, for this example we will let $\rho_1 = 16$.

The algorithm would begin as in Example 1 by computing the estimated value of the initial plan, $\mathcal{P}_1$. Since we are using the same initial plan, the estimate $U(\mathcal{P}_1, s_1, \rho_1)$, is again described by (repeated from by Eq. 5.4):

$$U(\mathcal{P}_1, s_1, \rho_1) = \max \begin{cases} \alpha(\mathcal{P}_1, \texttt{SampleRock(L1)}, s_1, \rho_1) + \beta(\texttt{SampleRock(L1)}, \rho_1) \\ \alpha(\mathcal{P}_1, \texttt{TakePicture(L1)}, s_1, \rho_1) + \beta(\texttt{TakePicture(L1)}, \rho_1) \\ \alpha(\mathcal{P}_1, \texttt{Navigate(L1, L2)}, s_1, \rho_1) + \beta(\texttt{Navigate(L1, L2)}, \rho_1) \end{cases} \quad (5.6)$$

When considering `SampleRock(L1)` as the first action, $\alpha(\mathcal{P}_1, \texttt{SampleRock(L1)}, s_1, \rho_1)$ will again be computed as:

$$U_1 = U(\textsf{Refine}(\mathcal{P}_1, \texttt{SampleRock(L1)}, s_1), s_2, \rho_2)$$

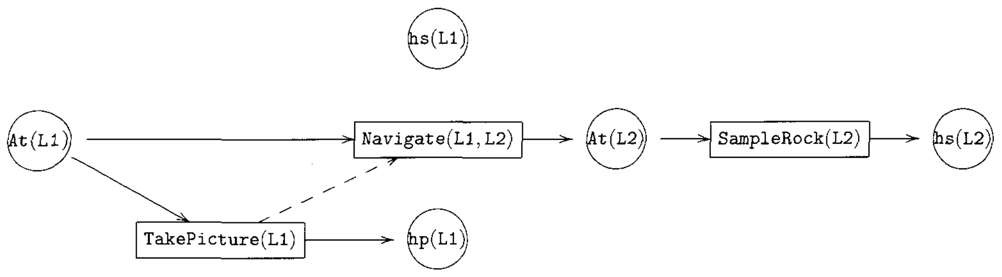

Figure 5-11: $\mathcal{P}_2$: Result of calling Refine($\mathcal{P}_1$, $\texttt{SampleRock(L1)}$, s_1).

with s_2 again being defined as:

$$s_2 = \sigma(s_1, \texttt{SampleRock(L1)}) = \{\texttt{At(L1)}, \texttt{hs(L1)}\}$$

but now ρ_2 is:

$$\rho_2 = \rho_1 - \mu(\texttt{SampleRock(L1)}) = 16 - 3 = 13$$

The call to Refine($\mathcal{P}_1$, $\texttt{SampleRock(L1)}$, s_1) will once again produce $\mathcal{P}_2$, shown in Figure 5-11 (repeated from Figure 5-7). As before, this leads to the computation of $U(\mathcal{P}_2, s_2, \rho_2)$ as (repeated from Eq. 5.5):

$$U(\mathcal{P}_2, s_2, \rho_2) = \max \begin{cases} \alpha(\mathcal{P}_2, \texttt{TakePicture(L1)}, s_2, \rho_2) + \beta(\texttt{TakePicture(L1)}, \rho_2) \\ \alpha(\mathcal{P}_2, \texttt{Navigate(L1,L2)}, s_2, \rho_2) + \beta(\texttt{Navigate(L1,L2)}, \rho_2) \end{cases} \tag{5.7}$$

Again following Example 1, when `TakePicture(L1)` is considered as the next action, $\alpha(\mathcal{P}_2, \texttt{TakePicture(L1)}, s_2, \rho_2)$ will still be computed as:

$$U_2 = U(\text{Refine}(\mathcal{P}_2, \texttt{TakePicture(L1)}, s_2), s_3, \rho_3)$$

where:

$$s_3 = \sigma(s_2, \texttt{TakePicture(L1)}) = \{\texttt{At(L1)}, \texttt{hs(L1)}, \texttt{hp(L1)}\}$$

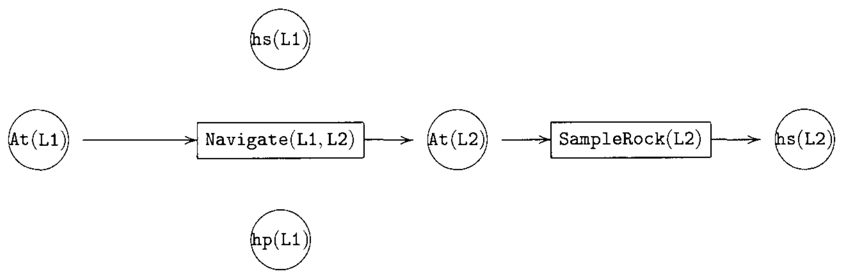

Figure 5-12: $\mathcal{P}_3$: Result of calling Refine($\mathcal{P}_2$, $\texttt{TakePicture(L1)}$, s_2).

but now:

$$\rho_3 = \rho_2 - \mu(\texttt{TakePicture(L1)}) = 13 - 2 = 11$$

The call to Refine($\mathcal{P}_2$, $\texttt{TakePicture(L1)}$, s_2) will once again produce the POPG $\mathcal{P}_3$ shown in Figure 5-12 (repeated from Figure 5-8).

Going a step further than we did in the first example, the algorithm would next select `Navigate(L1,L2)` as the next action (since there is no choice at this point), and compute $U(\mathcal{P}_3, s_3, \rho_3)$ as:

$$\begin{aligned} U(\mathcal{P}_3, s_3, \rho_3) &= \alpha(\mathcal{P}_3, \texttt{Navigate(L1,L2)}, s_3, \rho_3) + \beta(\texttt{Navigate(L1,L2)}, \rho_3) \\ &= U(\mathsf{Refine}(\mathcal{P}_3, \texttt{Navigate(L1,L2)}, s_3), s_4, \rho_4) + 0 \end{aligned}$$

where:

$$s_4 = \sigma(s_3, \texttt{Navigate(L1,L2)}) = \{\texttt{At(L2)}, \texttt{hs(L1)}, \texttt{hp(L1)}\}$$

and

$$\rho_4 = \rho_3 - \mu(\texttt{Navigate(L1,L2)}) = 11 - 10 = 1$$

The call to Refine($\mathcal{P}_3$, $\texttt{Navigate(L1,L2)}$, s_3) will produce the POPG $\mathcal{P}_6$ shown in Figure 5-13. The algorithm would try to continue, but since $\rho_4 = 1$ and min($\texttt{SampleRock(L2)}$) $= 5$, actions($\mathcal{P}_6, s_4, \rho_4$) $= \emptyset$, so $U(\mathcal{P}_3, s_3, \rho_3) = 0$, making $U_2 = 0$ as well.

The algorithm would next backtrack to Eq. 5.7 and consider `Navigate(L1,L2)` as the next action to take in plan $\mathcal{P}_2$. As in Example 1, it would again compute

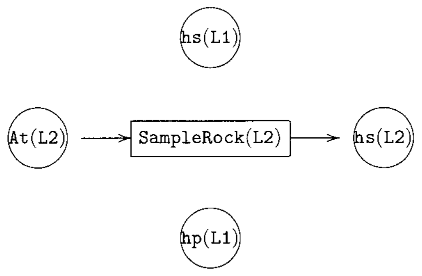

Figure 5-13: $\mathcal{P}_6$: Result of calling $\mathsf{Refine}(\mathcal{P}_3, \mathtt{Navigate(L1,L2)}, s_4)$.

$\alpha(\mathcal{P}_2, \mathtt{Navigate(L1,L2)}, s_2, \rho_2)$ as:

$$U_4 = U(\mathsf{Refine}(\mathcal{P}_2, \mathtt{Navigate(L1,L2)}, s_2), s_5, \rho_5)$$

where:

$$s_5 = \sigma(s_2, \mathtt{Navigate(L1,L2)}) = \{\mathtt{At(L2)}, \mathtt{hs(L1)}\}$$

but:

$$\rho_5 = \rho_2 - \mu(\mathtt{Navigate(L1,L2)}) = 13 - 10 = 3$$

The call to $\mathsf{Refine}(\mathcal{P}_2, \mathtt{Navigate(L1,L2)}, s_2)$ will produce the POPG $\mathcal{P}_7$ shown in Figure 5-14. The algorithm would try to continue, but since $\rho_5 = 3$ and $\min(\mathtt{SampleRock(L2)}) = 5$, $\mathsf{actions}(\mathcal{P}_7, s_5, \rho_5) = \emptyset$, so $U_4 = 0$.

The algorithm would then determine that:

$$U(\mathcal{P}_2, s_2, \rho_2) = \max \begin{cases} U_2 + \beta(\mathtt{TakePicture(L1)}, \rho_2) = 0 + 2 = 2 \\ U_4 + \beta(\mathtt{Navigate(L1,L2)}, \rho_2) = 0 + 0 = 0 \end{cases}$$

Backtracking to Eq. 5.6, since $U(\mathcal{P}_2, s_2, \rho_2) = 2$, at this point the algorithm would know that when $\rho_1 = 16$, the best value this plan could obtain with `SampleRock(L1)` as the first action is:

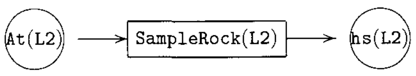

Figure 5-14: $\mathcal{P}_7$: Result of calling $\mathsf{Refine}(\mathcal{P}_2, \texttt{Navigate(L1,L2)}, s_5)$.

$$\begin{aligned}
U(\mathcal{P}_1, s_1, \rho_1) &\geq U_1 + \beta(\texttt{SampleRock(L1)}, \rho_1) \\
&\geq U_2 + \beta(\texttt{TakePicture(L1)}, \rho_2) + \beta(\texttt{SampleRock(L1)}, \rho_1) \\
&\geq U_2 + V_{\texttt{hp(L2)}} + V_{\texttt{hs(L1)}} \\
&\geq 0 + 2 + 2 \\
&\geq 4
\end{aligned}$$

As in Example 1, it is not difficult to see that selecting `TakePicture(L1)` as the first action would produce the same value for $U(\mathcal{P}_1, s_1, \rho_1)$, 4. However, we will examine the selection of `Navigate(L1,L2)` as the first action to execute from $\mathcal{P}_1$, (i.e., the third part of Eq. 5.6), which would again be computed as:

$$\alpha(\mathcal{P}_1, \texttt{Navigate(L1,L2)}, s_1, \rho_1) + \beta(\texttt{Navigate(L1,L2)}, \rho_1)$$

The computation of $\alpha(\mathcal{P}_1, \texttt{Navigate(L1,L2)}, s_1, \rho_1)$ is computed as:

$$U_5 = U(\mathsf{Refine}(\mathcal{P}_1, \texttt{Navigate(L1,L2)}, s_1), s_6, \rho_6)$$

where

$$s_6 = \sigma(s_1, \texttt{Navigate(L1,L2)}) = \{\texttt{At(L2)}\}$$

and

$$\rho_6 = \rho_1 - \mu(\texttt{Navigate(L1,L2)}) = 16 - 10 = 6$$

The call to $\mathsf{Refine}(\mathcal{P}_1, \texttt{Navigate(L1,L2)}, s_1)$ will produce POPG $\mathcal{P}_5$ as shown in Figure 5-15 (repeated from Figure 5-10). Without going through the details, it is not difficult to see from Figure 5-15 that $U(\mathcal{P}_5, s_6, \rho_6) = V_{\texttt{hs(L2)}} = 10$, since $\rho_6 >$

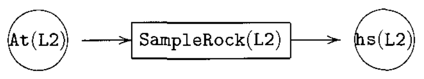

Figure 5-15: $\mathcal{P}_5$: Result of calling Refine($\mathcal{P}_1$, Navigate(L1, L2), s_1).

min(SampleRock(L2)). Thus, the algorithm can now complete the computation of Eq. 5.6 as:

$$U(\mathcal{P}_1, s_1, \rho_1) = \max \begin{cases} 4 & if A = \texttt{SampleRock(L1)} \\ 4 & if A = \texttt{TakePicture(L1)} \\ 10 & if A = \texttt{Navigate(L1,L2)} \end{cases}$$

Thus, unlike our first example, in this case, selecting Navigate(L1,L2) as first action for $\mathcal{P}_1$ is the rational choice, since it leads to highest value for the given resources.

5.4 Discussion

We conclude this chapter by examining the flexibility of our planning/execution model with respect to the various possible changes in the environment that we listed in Section 5.2. First, sudden unreachability of goals, as well as uncertainty in resource consumption by the agent's actions, is captured by the model implicitly. Second, if the values of some of the (still reachable) goals that the agent had planned to achieve have changed, the only thing that the agent should do is update the value functions associated with the corresponding goals in its plan. All the subsequent decisions will implicitly take this change into account. In particular, if one of the agent's goals, g, becomes completely irrelevant (i.e., $V_g = 0$), the agent could easily update its POPG by removing the node g, along with exactly those action nodes that are used to "produce" this node g and are not used to produce any other goal. However, in the next chapter we show that the latter is not necessarily the best way to handle such situations.

The only part of the environmental dynamics that still seems to be problematic is assigning a new goal to an agent (i.e., a goal that is not captured by the current plan $\mathcal{P}$). Such a goal can be either completely new to the multi-agent group, or one of the goals that has been suspended by some other agent in the group. Clearly, a complete re-planning for the extended set of goals will solve the problem, and in many domains such a painful solution might be unavoidable. However, in the next chapter we argue that, at least for some practical domains, we can extend the above model of planning and execution in a way that re-planning can often be avoided.

Chapter 6

Plan-Based Opportunism

The model for plan creation and execution described in the previous chapter provides an agent with enormous flexibility in selecting its course of action. This flexibility would in turn allow agents operating in real-world domains to better adapt to dynamic environments, especially in terms of the opportunistic satisfaction of suspended goals. In this chapter we discuss the way our model supports opportunistic behavior of the agents.

6.1 Model of Opportunistic Execution

Figure 6.1 summarizes the POPG-based model of an agent's planning and execution cycle as described in Chapter 5. Focusing on step 4e, we note that by taking a particular action A from the plan $\mathcal{P}$, the agent may suspend one or more of its goals, namely those that are not achievable along all courses of action beginning with action A. The suspended goals could be re-planned for in the next planning-execution cycle, returned to the task broker agent $\mathbb{B}$ for re-allocation to another agent, or abandoned completely.

As discussed in Section 5.2, however, conditions may change during execution

Portions of the content of this chapter have been published in [Lawton and Domshlak, 2003, 2004a,b; Domshlak and Lawton, 2003, 2004].

```
Loop forever:
1. Check for and process new goal assignments from broker $\mathbb{B}$, $G_i \subseteq C_i$, where $C_i$ is the set
   of the agent $\mathbb{A}_i$'s capabilities.
2. Generate a POPG $\mathcal{P}$ that achieves $G_i$ from the current state, $s$.
3. For each goal $g \in G_i$, annotate the corresponding nodes in $\mathcal{P}$ with $V_g$.
4. Loop until (all $g \in G$ are satisfied) $\vee$ ($\mathsf{actions}(\mathcal{P}, s, \rho) = \emptyset$)
   (a) Estimate the value $U(\mathcal{P}, s, \rho)$ of the current plan $\mathcal{P}$ using the value iteration process
       in Eq. 5.2.
   (b) Choose an action $A \in \mathsf{actions}(\mathcal{P}, s, \rho)$ that actually provides $U(\mathcal{P}, s, \rho)$.
   (c) Perform $A$, resulting in the new state $\sigma(s, A)$ and some remaining amount of resource
       $\rho' \leq \rho$.
   (d) Set $\mathcal{P} = \mathsf{Refine}(\mathcal{P}, A, s)$, $\rho = \rho'$, $s = \sigma(s, A)$.
   (e) Suspend all the goals $g$ that are no longer reachable in $\mathcal{P}$.
5. Notify broker $\mathbb{B}$ of success/failure of achieving each goal $g \in G_i$.
```

Figure 6-1: Basic planning and execution cycle of a single agent.

such that a suspended goal may indeed become achievable. In this context, an agent would exhibit single-agent opportunism if it can detect and respond to events and situations that may allow one of its suspended goals to be satisfied. In our model, single-agent opportunism uses a form of *predictive encoding* [Patalano et al., 1993], in which the agent examines its remaining current plan to find other places where a suspended goal may be achieved.

To support this behavior, an agent needs to be able to modify the structure of its current plan. Observe that nothing prevents us from enriching an existing POPG $\mathcal{P}$ with additional actions that might be inconsistent (i.e., cannot be merged together) with each and all possible complete executions of $\mathcal{P}$. For example, consider the (partial) POPG $\mathcal{P}$ depicted in Figure 6-2(a). In Figure 6-2(b) this valid "seed" plan $\mathcal{P}$ is extended to a new structure $\mathcal{P}_e \supset \mathcal{P}$ by adding the action `Navigate(L1,L3)`.

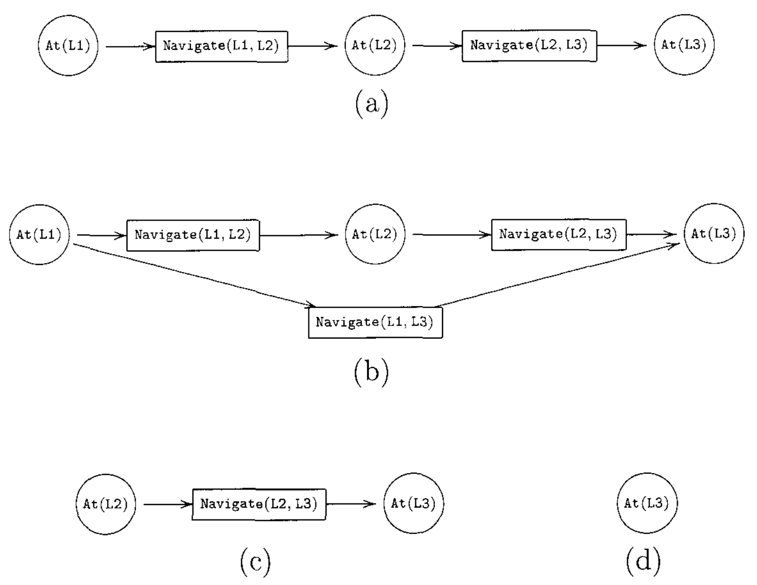

Figure 6-2: Extending the "seed" POPG: (a) The initially constructed POPG $\mathcal{P}$; (b) EPOPG $\mathcal{P}_e$, resulting from the extension of $\mathcal{P}$ by the addition of an alternative course of action `Navigate(L1,L3)`; (c-d) The result of Refine on $\mathcal{P}_e$ and the actions `Navigate(L1,L2)` and `Navigate(L2,L3)`, respectively.

In general, such an extension $\mathcal{E}$ can be any valid POPG such that the root nodes (i.e., the nodes with no incoming edges) of $\mathcal{E}$ are a subset of the proposition nodes of $\mathcal{P}$ (i.e., $\mathcal{E}$ is grounded in $\mathcal{P}$). Further, this plan extension process can be performed iteratively, allowing the plan to be extended as needed.

Clearly, the constructed plan $\mathcal{P}_e$, which we will call an *extended partial order plan graph* (EPOPG), may not be a valid partial order plan, and this is actually the case with the EPOPG depicted in Figure 6-2(b). However:

1. It does contain at least one non-trivial, valid partial order plan, and

2. At every execution state s, the agent will still choose an action providing it with the maximal expected reward.

The first statement appears to be obvious, since any single goal-achieving action by itself is a valid partial order plan. However, the fact that (i) we start with a valid "seed" plan for a given set of goals, and (ii) nothing from this plan is ever removed during the extension process, further justifies this statement. Similarly, recall that every extension $\mathcal{E}$ is itself a valid POPG. If the execution mechanism follows the course of action defined by an extension, even if some of the actions from the initial seed plan are removed, the plan will still contain at least this one valid plan.

The second statement is both less straightforward and very important, as maximizing the expected reward is the core part of our execution model. The soundness of this statement follows from the fact that, using the Refine procedure, the process of calculating $U(\mathcal{P}_e, s, \rho)$ via Eq. 5.2 exploits only valid total order sequences of actions from $\mathcal{P}_e$, even if $\mathcal{P}_e$ is not a valid partial order plan. As discussed in Section 5.3.2, this is because the procedure for estimating a plan's value defined by Eq. 5.2 considers all valid total orders, while Refine prunes unreachable actions out of the plan. The completeness follows from the fact that the first parameter of $U(\mathcal{P}, s, \rho)$ in Eq. 5.2 decreases monotonically with the nesting depth. We observe, therefore, that completeness is preserved even if the constructed EPOPG $\mathcal{P}_e$ contains cyclic dependencies.

In general, adapting a given plan to pursue additional goals can be computationally hard [Yang et al., 1992]. However, an agent can examine and extend an existing (E)POPG $\mathcal{P}_e$, since that structure is not required to represent a single valid plan. Thus, predictively encoding suspended goals for possible opportunistic execution in $\mathcal{P}_e$ can be performed much more efficiently than by updating a plan that must remain consistent. More directly, suppose that after performing an action A, the procedure Refine removes from the current EPOPG $\mathcal{P}_e$ an action A' because one (or more) of A's precondition nodes have also been removed from $\mathcal{P}_e$. Let us denote by $\mathcal{P}'_e$ the EPOPG resulting from the above refinement of $\mathcal{P}_e$. If A was necessary for achieving

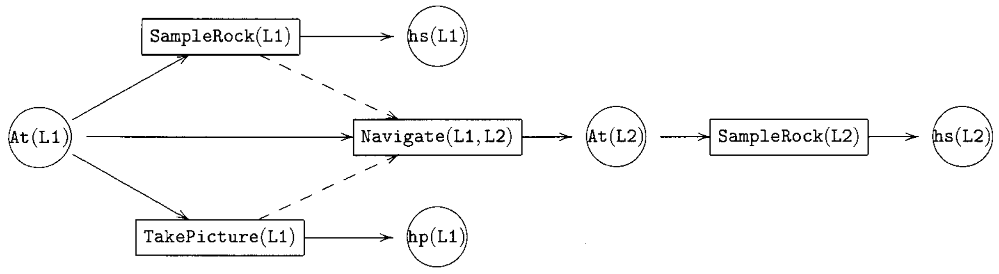

Figure 6-3: POPG for the Rovers example (from Fig. 5-4).

some assigned goal g, then g will have to be suspended. However, if there exists a POPG $\mathcal{P}' \subset \mathcal{P}_e$ that achieves g, the agent could try to predictively re-encode such a sub-plan $\mathcal{P}'$ into the refined EPOPG $\mathcal{P}'_e$ using the EPOPG extension approach described above, treating $\mathcal{P}'$ as the chosen extension $\mathcal{E}$.

For example, consider the EPOPG presented in Figure 6-3 (repeated from Figure 5-4). Suppose that the agent performs the action Navigate(L1,L2) before executing TakePicture(L1). The goal hp(L1) would be suspended, because the sub-plan $\mathcal{P}' = \{$TakePicture(L1)$\}$ has been pruned by the Refine procedure. The remaining plan could be extended by re-grounding $\mathcal{P}'$ at any other appearance of a node At(L1) in the EPOPG. The plan execution procedure would automatically reconsider opportunistically achieving hp(L1) if and when it encounters TakePicture(L1) again in the future. Unfortunately, in this particular example, there are no such places in our EPOPG, thus predictive encoding will be infeasible. As such, we need to seek alternative ways to achieve hp(L1), such as multi-agent opportunism.

Recall from Section 5.1 that our general approach to multi-agent opportunism is to have the agents include *extra* goals in their plans, on the chance that some other agent in the MAS might suspend one of them. If at runtime an agent does in fact suspend a goal that has been predictively encoded as an extra goal in some other agent's plan, the other agent would be in a position to opportunistically satisfy the suspended goal. But how does an agent determine which extra goals should be included in it plan?

Consider, for instance, an MAS $\{\mathbb{A}_1, \cdots, \mathbb{A}_n\}$ executing plans $\mathcal{P}_1, \cdots, \mathcal{P}_n$ for the goal sets $G_1, \cdots, G_n$, respectively. Suppose that, at some point agent $\mathbb{A}_i$ suspends a goal $g \in G_i$, and notifies some of the other agents that it can no longer satisfy g. Since we are assuming the agents in the MAS are cooperative (or at least benevolent), we would like the other agents $\mathbb{A}_j$ (such that $g \in C_j$) to at least consider whether they can achieve g themselves and whether the corresponding changes in their course of action would be feasible.

For some agent $\mathbb{A}_j$, if g is not already reachable in $\mathcal{P}_j$, *and* if $\mathbb{A}_j$ is capable of minimal re-planning, then $\mathbb{A}_j$ could create a new sub-plan $\mathcal{P}'$ just for achieving g starting from the current state. The (E)POPG $\mathcal{P}_j$ could then be extended into the EPOPG $\mathcal{P}_e$ to include $\mathcal{P}'$. While considering actions in the future, the POPG execution module of $\mathbb{A}_j$ (Section 5.3.2) will implicitly adjust its course of action with respect to this update. The suspended goal g will be opportunistically satisfied if $\mathbb{A}_j$'s conditions and resources permit. Although we do examine this approach in Section 6.2.3, recall that we have assumed that in general the agents are not capable of re-planning, and thus must consider other options.

In the more general case, when the agents are not capable of re-planning, then the plans $\mathcal{P}_1$, $\cdots$, $\mathcal{P}_{i-1}$, $\mathcal{P}_{i+1}$, $\cdots$, $\mathcal{P}_n$ cannot be changed, and any opportunistic assistance to $\mathbb{A}_i$ (if possible at all) must be based on them as they are. Notice, though, that even without re-planning, it might be the case that g is serendipitously present, and thus potentially achievable, in one of the POPG-represented plans $\mathcal{P}_j$ (e.g., as a side-effect of $\mathbb{A}_j$'s primary activities). To opportunistically adopt g as a new goal, $\mathbb{A}_j$ needs only to properly increase the value of g, updating the value function V_g in $\mathcal{P}_j$. Again the execution module of $\mathbb{A}_j$ will implicitly adjust its intention with respect to this update when considering future actions. Note that even if $\mathbb{A}_j$ has not abandoned g, but has suspended it and predictively re-encoded it for possible opportunistic achievement in the future, the node g will still be reachable in $\mathcal{P}_e$.

Let us consider some properties of this extension to our planning and execution scheme to support multi-agent opportunism. First, because we are using predictive encoding, the runtime computational complexity of some agent $\mathbb{A}_j$ providing opportunistic assistance to another agent $\mathbb{A}_i$ is, as desired, kept low. In our model, updating a value function V_g is all that is required to determine a potential opportunity for a suspended goal g. This process is linear in the size of POPG $\mathcal{P}_j$ in the worst case. Note that here we are only referring to the additional computation needed for opportunism. It is true that the worst case computational complexity of our action selection mechanism is exponential in the size of the plan. However, the size of a POPG is, in practice, often quickly reduced by the execution mechanism as unusable courses of action are removed. Also, in Section 9.3.1 we discuss various ways the efficiency of the action selection mechanism can be improved.

Second, the value of g is automatically compared to the values of other goals in G_j. As the choice of action in our scheme of execution is based on maximizing expected value, and since the value a goal g, V_g, represents its contribution to the system (and not just to the individual agent), achieving g will not come at the expense of other goals in G_j unless g is justifiably considered to contribute more to the MAS. Finally, even if agent $\mathbb{A}$ is not itself capable of single-agent opportunism, it may still be able to provide opportunistic support for other agents, as long as the goals suspended by these other agents are still reachable in the POPG of $\mathbb{A}$. This is interesting because we had initially considered multi-agent opportunism strictly as an extension of single-agent opportunism.

Returning to the discussion on our approach to opportunistic execution, one may rightfully say that if g were assigned as a goal to $\mathbb{A}_i$, it is not very likely that g would also appear in the plan of another agent. (In our Mars rovers example, indeed, why would a rover plan to sample rocks at a certain location if it was not assigned to so?) It again appears as if multi-agent opportunism without dynamic re-planning is not very promising. However, this is not necessarily the case.

Observe that nothing in our scheme for planning and execution prevents agents from planning for goals that were *not* assigned to them, i.e., goals having *zero* value from the local perspective of these agents. Since the decision mechanism behind the execution takes into account not only the value of the goals to be achieved, but also the risk behind the various courses of action (encountered via cumulative resource consumption), achieving a goal with a zero value will *automatically* be postponed. Similarly, if one of the goals that the agent has planned for becomes irrelevant, instead of removing this goal from the plan, the agent could simply zero its value function.

Now, recall that in our model each agent is characterized by a set of capabilities representing all the goals that can possibly be assigned to the agent. In general, instead of planning for just the set of goals that have been actually assigned to the agent, one can consider also planning for extra capabilities (goals) that have not been assigned to the agent, or even *planning for the whole set of capabilities*, and reasoning about the best course of action during execution, when the value of different capabilities is known better than during the off-line planning. This is, in fact, the key idea behind our approach to multi-agent opportunism, as it allows the agents to opportunistically satisfy goals that are suspended by other agents, since they are already part of their current plan.

Clearly, one may rightfully say that the whole set of capabilities may be huge, and even its explicit description may be intractable. Although we agree that in general nothing prevents the set of capabilities from being orders of magnitude larger than an average set of goals the agent is actually assigned, at least in some domains this does not seem to be the case. For instance, consider a group of planetary rovers that are constructed to fulfill some tasks on Mars [Estlin and Gaines, 2002]. At least at this stage of planetary rovers development, the superset of goals that each rover can be assigned is not very large, yet this domain poses many challenging research and development issues. We do, however, believe that in general it will be necessary for an agent to select and plan for a subset of its capabilities. In Section 7.1 we consider two domain-independent methods for selecting a specific subset of the opportunistically "most promising" capabilities to plan for. These methods assume that the agents will have very little shared knowledge about one another, and thus should provide a baseline of what is possible.

If planning for capabilities is considered to be as feasible as planning for the actual goals, technically nothing should be changed in the scheme of planning and execution described in Section 5.3. However, a plan generated to include extra capabilities, especially for the whole set of an agent's capabilities, can be far from efficient with respect to just the actual goals that have been assigned to the agent. For instance, suppose that a rover located at location `L1` has been assigned a single goal of sampling the rocks at location `L1000`. If this particular rover is capable of sampling rocks at any of the thousand locations $\mathtt{L1}, \mathtt{L2}, \ldots, \mathtt{L1000}$, the constructed plan may take the rover from `L1` to `L1000` through all the locations in between, as if preparing this rover to perform the other rock samplings as well. At first view, this observation seems to point to a serious drawback of planning for extra capabilities instead of just for assigned goals. However, in Section 6.2 we present several domain-independent methods to improve the efficiency of plans by avoiding unnecessary actions.

Finally, consider the value estimation process that an agent performs at every decision point. We have already discussed that computing a precise value estimation is intractable for most, if not all, practically interesting domains. Therefore, we began our discussion with an approximate estimation provided by Eq. 5.2, as this approximation procedure dramatically reduces the branching factor of the value iteration process. However, the complexity of calculating $U(\mathcal{P}_e, s, \rho)$ is still on the order of the number of alternative valid sequences of actions consistent with $\mathcal{P}_e$, s, and ρ, which in worst case is obviously exponential in the number of actions in $\mathcal{P}_e$. Therefore, to obtain truly practical execution schemes one would have to examine various techniques to limit the depth of value iteration with as little loss of decision accuracy as possible.

We believe that there are several ways to provide a good estimate of $U(\mathcal{P}_e, s, \rho)$, which in turn could be used instead of Eq. 5.2 starting from a certain depth of the value iteration process. For instance, the method introduced by Dearden et al. [2002] for off-line backpropagation of goal values to the internal nodes of a Graphplan-based planning graph would serve as a good starting point. In particular, it should be possible to apply this method to the non-leveled graphical structures such as POPG and EPOPG. Generally speaking, using this method, each action node A is associated with a value function $V_A(\rho, s)$ that provides an approximation of the combination of $\alpha(\mathcal{P}, A, s, \rho)$ and $\beta(A, \rho)$ from Eq. 5.3. While we have not fully explored this sort of efficiency improvement, it is an important part of our planned future work (Section 9.3.1).

6.2 Making POPGs More Efficient

In the previous section we noted that planning for extra goals can lead to inefficient plans. This is because such plans may lead an agent to execute unneeded actions on the chance that an extra goal would be suspended by another agent. To make the plans more efficient, we need to find a way to create a plan that can opportunistically satisfy extra goals as they arise (i.e., when they are suspended by other agents), yet efficiently satisfy the assigned goals when no opportunities are present.

In this section we address this issue by considering three domain-independent approaches for creating plan extensions. Two of these approaches, *planning with shortcuts* and *predictive plan repair*, operate during the off-line planning stage, while the third, *reactive plan repair*, augments the current plan at runtime when goals are suspended. In Chapter 7 we present the results of an empirical evaluation of these approaches.

6.2.1 Planning with Shortcuts

For planning with shortcuts, each agent first generates a plan for *all* of its assigned and extra goals, and then augments the structure of that plan by adding "shortcuts" to the assigned goals. Shortcuts are actions (or short sequences of actions) that bypass the segments of the plan devoted strictly to support predictively encoded extra goals. For example, given the plan in Figure 6-2(a), the action `Navigate(L2,L3)` added in Figure 6-2(b) can be seen as such a shortcut to the goal `At(L3)`.

We will begin by assuming that each agent $\mathbb{A}$ is assigned k goals by the broker $\mathbb{B}$, and that $\mathbb{A}$ selects k' additional goals to plan for on the expectation that they may lead to opportunistic execution. As in the basic approach (Section 6.1), $\mathbb{A}$ creates a plan $\mathcal{P}$ for all of these $k + k'$ goals. Since the external planner cannot differentiate between the assigned and extra goals, it will produce a plan that satisfies all of the goals in a manner that it considers "efficient"—often based on minimizing the number of actions. We would, however, like the agents to be able to dynamically skip those actions that do not contribute to achieving any of the goals that currently have a positive value.

Consider, for example, the small POPGs shown in Figure 6-4, where the circular and rectangular nodes stand for proposition and action nodes, respectively, and doubly-circled propositions stand for goals. Figure 6-4(a) shows a base plan for $k = 3$

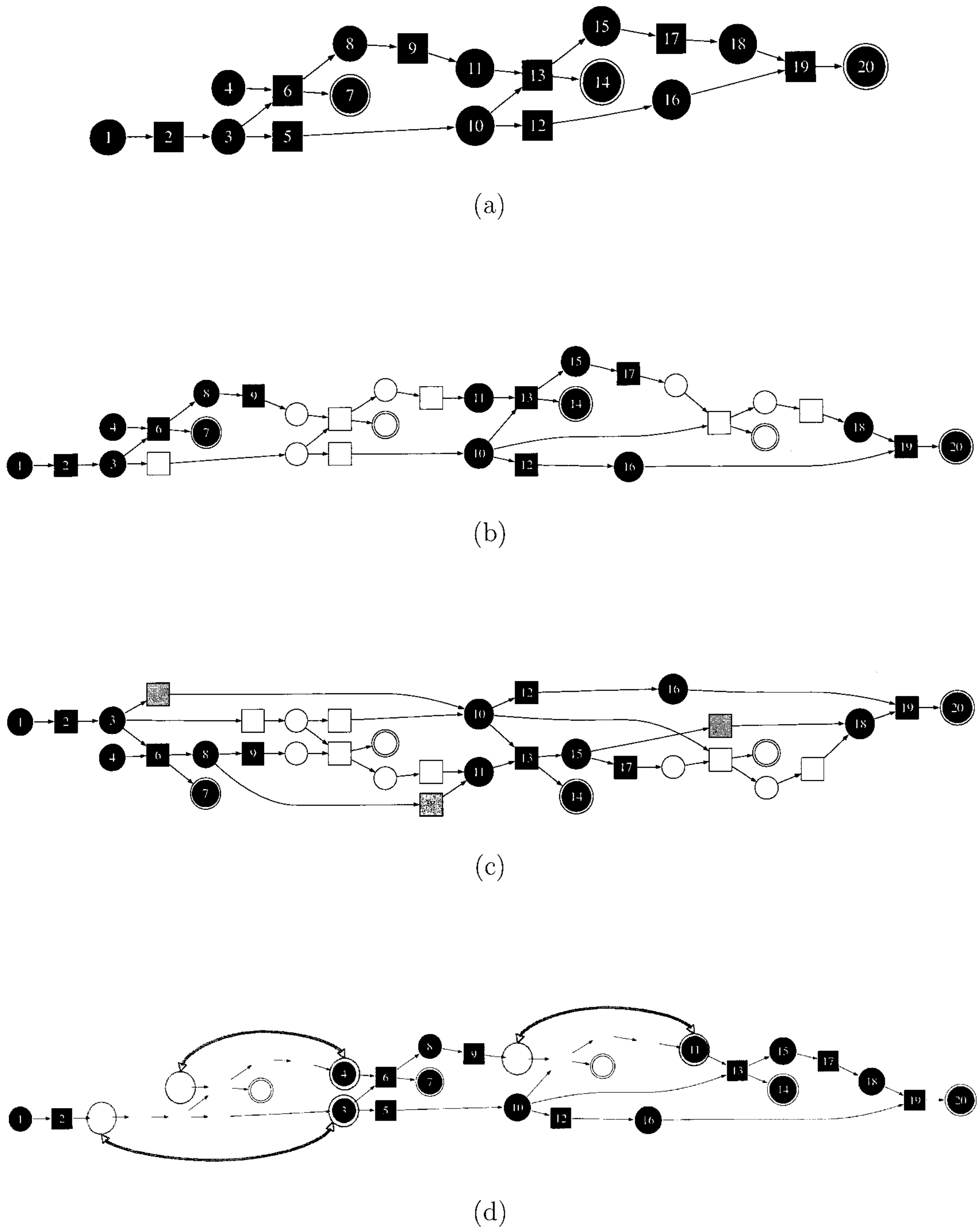

Figure 6-4: Example POPGs: (a) Base plan; (b) Extended plan; (c) Extended plan with shortcuts; (d) Base plan with repairs.

assigned goals, while Figure 6-4(b) shows a plan $\mathcal{P}$ for these k and some extra k' goals. The white nodes in Figure 6-4 represent the additional actions and conditions needed to accomplish these $k' = 2$ extra goals, while the numbered black nodes correspond to the nodes in the base plan. Figure 6-4(c) shows the plan $\mathcal{P}_s$, constructed from $\mathcal{P}$ by automatically adding shortcuts (shown as dark gray nodes) devoted to bypass the actions only needed for achieving the extra goals.

Adding shortcut nodes to $\mathcal{P}$ in this way makes $\mathcal{P}_s$ an EPOPG as discussed in 6.1. The real question is, however, how do we determine where to place the shortcut actions? Our approach is to trace through a *total order* of the plan $\mathcal{P}$, finding the start and end nodes of "skippable" sections. For each pair of start and end nodes, we can generate a plan fragment that bypasses the corresponding section of the core plan.

To formally specify the notion of skippable sections of a plan, assume that an agent $\mathbb{A}$ is assigned k goals $G_a = \{g_1, g_2, \ldots, g_k\}$ (with $V(g_i) > 0$ for $1 \leq i \leq k$), and that it also selects an additional k' goals $G_e = \{g_{k+1}, g_{k+2}, \ldots, g_{k+k'}\}$ (with $V(g_i) = 0$ for $k+1 \leq i \leq k+k'$). Further, let $\mathcal{P}$ be the plan generated for all $k+k'$ goals, and $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ be a total order of its n actions consistent with the ordering induced by the POPG. For each pair of actions $a_i, a_j \in \mathcal{A}$, we say that $a_i < a_j$ if $i < j$. This sequence of actions is implicitly associated with a sequence of $n+1$ states $\{s_0, s_1, \ldots, s_n\}$, such that s_0 is the initial state, and for $1 \leq i \leq n$, a_i moves the agent from state s_{i-1} to state s_i. Let $\alpha(g)$ denote the action that actually satisfies some goal $g \in G_a \cup G_e$, i.e., $\alpha(g) = a_m$ is the action such that $g \in s_m$, but for $0 \leq i < m$, $g \notin s_i$. Consider the assigned goals $\{g_1, g_2, \ldots, g_k\}$ numbered according to their achievement along the total order $\mathcal{A}$. That is, for $1 \leq i < j \leq k$, if $\alpha(g_i) \neq \alpha(g_j)$, we know $\alpha(g_i) < \alpha(g_j)$.

Suppose that for some pair of assigned goals $g_i, g_{i+1} \in G_a$, we have an extra goal

$g \in G_e$, such that $\alpha(g_i) < \alpha(g) < \alpha(g_{i+1})$. Let $\alpha(g_i) = a_j$ and $\alpha(g_{i+1}) = a_l$. To allow bypassing the actions needed *only* for achieving g, we can first create a plan fragment $\mathcal{P}'$ with $s'_0 = s_j$ (i.e., the initial state of $\mathcal{P}'$ is the state produced by action a_j) and the goal conjunct $G_{\mathcal{P}'} = s_{l-1}$ (i.e., the state immediately preceding action a_l), and then attach $\mathcal{P}'$ to $\mathcal{P}$, properly grounded at the appropriate proposition nodes in s_j and supporting the proposition nodes of s_{l-1}. This will create an alternate path around the skippable section, as illustrated in Figure 6-4(c) by the segments with dark gray nodes. Adding such shortcuts preserves the completeness of the plan with respect to the assigned goals G_a since the structure resulting from replacing the actions $a_{j+1}, \ldots, a_{l-1}$ in $\mathcal{P}$ with the plan fragment $\mathcal{P}'$ is a valid partial order plan that, given an unbounded amount of resources, achieves all the assigned goals G_a.

Given a POPG that has been extended with shortcuts, the agent would execute it in the usual way as described in Section 5.3.2. If during execution the agent is notified by some other agent that one of the extra goals, $g \in G_e$, predictively encoded into its plan has been suspended, g's value would be adjusted appropriately and the execution mechanism would automatically decide whether to execute the actions leading to satisfying g. Alternatively, if at runtime the value of one of these extra goals g remains zero, the execution mechanism would automatically select the shortcut path to follow, avoiding the actions used for satisfying g, since this would be a lower cost path. In the unlikely event that the costs of the paths are equal, a shortest-path heuristic could be used. This ensures the the extended plan performs at least as well as the base plan.

Observe further that the total set of proposition nodes in $G_{\mathcal{P}'}$ can be large, as it corresponds to the entire state s_{l-1}. However, in general there is no need to plan for all the propositions of s_{l-1}, since many of them may have no effect on the applicability of the remaining part $\{a_l, \ldots, a_n\}$ of $\mathcal{P}$. Therefore, without loss of either soundness or completeness, we can assign $G_{\mathcal{P}'}$ to contain only the propositions of s_{l-1} that act as pre-conditions of some actions in $\{a_l, \ldots, a_n\}$. That is:

$$G_{\mathcal{P}'} = \bigcup_{a \geq a_l} s_{l-1} \cap \mathsf{prec}(a) \tag{6.1}$$

6.2.2 Predictive Plan Repair

For predictive plan repair, we again assume that each agent $\mathbb{A}$ is assigned k non-zero-valued goals $G_a = \{g_1, g_2, \ldots, g_k\}$, and that it also selects an additional k' (zero-valued) goals $G_e = \{g_{k+1}, g_{k+2}, \ldots, g_{k+k'}\}$. Unlike in planning with shortcuts, however, the agent initially generates a plan $\mathcal{P}$ only for its k assigned goals G_a, and then *expands* $\mathcal{P}$ to include actions that accomplish the goals in G_e. For illustration, such a plan $\mathcal{P}$ for achieving $k = 3$ assigned goals is depicted in Figure 6-4(a). Figure 6-4(d) shows an expansion $\mathcal{P}_r$ of $\mathcal{P}$, achieved by augmenting $\mathcal{P}$ to include extra actions (shown in borderless gray) to satisfy the additional $k' = 2$ goals. The proposition nodes connected by double-ended arrows in this figure represent identical conditions. Thus we can see that the repair returns the plan to the same state.

As with the formalism for planning with shortcuts, let $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ be a total order of the actions of the current plan $\mathcal{P}$. To expand $\mathcal{P}$ with respect to an extra goal $g \in G_e$, the agent selects from its action set an action a that provides g and has the best support in $\mathcal{P}$ among all such actions. That is:

$$a = \underset{a' \text{ s.t. } g \in \mathsf{effects}(a')}{\text{argmax}} \left\{ \max_{1 \leq i \leq n} \{|s_i \cap \mathsf{prec}(a')|\} \right\} \tag{6.2}$$

where $\underset{domain}{\texttt{argmax}}\{\mathcal{F}\}$ returns the element $e \in$ *domain* for which $\mathcal{F}(e)$ is maximal. Similarly, s_i is the corresponding state supporting a, i.e.,:

$$s_i = \underset{s_j \in s_0, \ldots, s_n}{\text{argmax}} \{|s_j \cap \mathsf{prec}(a)|\} \tag{6.3}$$

To preserve the opportunistic nature, the agent should avoid predictive encoding of extra goals having insufficient correlation with the current plan. That is, the nature of opportunism is to take advantage of situations that already exist (or are likely to exist) in an agent's intentions. Adding a goal to an existing plan that would require the agent to deviate from that plan significantly would not be opportunistic. For instance, in our evaluation discussed in the next chapter, s_i is required to meet some fraction of a's preconditions (half, in our case, which was selected without extensive empirical testing), otherwise g is not considered to be a potential opportunity. Other measures, such as considering the difficulty of achieving certain preconditions for the goal, could also be used. We have not, however, considered any in this research.

Given the core plan $\mathcal{P}$, and an action/state pair a and s_i as above, the agent starts by creating a plan fragment $\mathcal{P}'$ for $s'_o = s_i$ and $G_{\mathcal{P}'} = \mathsf{prec}(a)$. Let s' be the state resulting from applying $\mathcal{P}'$ in s_i, and s'' be the state resulting from applying a in s'. Next, the agent creates another plan fragment $\mathcal{P}''$ for $s''_0 = s''$ and $G_{\mathcal{P}''} = s_i$ (i.e., $\mathcal{P}''$ returns to the initial state for $\mathcal{P}'$), and concatenates $\mathcal{P}'$, a, and $\mathcal{P}''$. Again, as with planning with shortcuts, the resulting "side-loop" $\mathcal{P}_r$ is attached to $\mathcal{P}$ by linking it to the appropriate proposition nodes in s_i (see Figure 6-4(d)). As with planning with shortcuts, the resulting plan would be executed as usual by the agent. If during execution the agent is notified by some other agent that one of the extra goals, $g \in G_e$, predictively encoded into its plan has been suspended, g's value would be adjusted appropriately and the execution mechanism would decide whether or not to follow the sub-plan $\mathcal{P}_r$. Alternatively, if at runtime the value of one of these extra goals g remains zero, the "side-loop" $\mathcal{P}_r$ added for achieving g will automatically be pruned. Thus, if during the execution of $\mathcal{P}_r$ all the extra goals G_e remain zero valued, the execution of $\mathcal{P}_r$ will be equivalent to executing the core plan $\mathcal{P}$. Since the execution mechanism always picks the most cost-effective course of action among those achieving maximal expected value, this ensures that the extended plan performs at least as well as the base plan.

Finally, as with planning with shortcuts, $G_{\mathcal{P}''} = s_i$ can be replaced with the specification as in Eq. 6.1. However, as the extra goals G_e are considered one by one, the whole process of plan repair is incremental with respect to G_e. Therefore, $G_{\mathcal{P}''}$ as described in Eq. 6.1 should be based on the *current* plan (i.e., the core plan with all the previously considered repairs), and not on the core plan for the k assigned goals only. This is necessary to ensure new plan repairs do not prevent previously added repairs from satisfying their extra goals if the corresponding opportunities arise.

6.2.3 Reactive Plan Repair

Planning with shortcuts and predictive plan repair are two forms of predictive encoding that can be adopted by agents which are not capable and/or not allowed to adjust their plans at execution time. If, however, some degree of online plan adjustment is possible, the agents can adjust their intentions as they learn about the suspended goals of other agents, without having to "guess" and plan for any extra goals in advance.

In this case, one may consider a reactive variant of predictive plan repair. Following the predictive plan repair approach, here again each agent generates a plan $\mathcal{P}$ for its k assigned goals. Unlike in the purely offline approaches to predictive encoding, however, the agents do *not* select any additional goals. Rather, when an agent is notified about some other agent's suspended goal, it uses the plan repair mechanism described in Section 6.2.2 to fit the goal (if possible) in to the remaining portion of its current plan.

A significant advantage of this approach is that the agents only plan for goals that actually get suspended. They do not waste any effort preparing for goals on the chance they might lead to opportunistic execution. Rather, they can focus their resources on considering opportunities for goals that they know cannot otherwise be satisfied. The reason we consider reactive plan repair in our analysis is two-fold. First, this approach corresponds to what we expect is the minimal form of online re-planning, preserving the qualitative core of the plan generated off-line. Second, as such, this approach provides us with yet another reference point for evaluating the attractiveness of purely offline forms of predictive encoding, the main interest of our work here.

6.3 Addressing the Critical Issues

We conclude this chapter with a discussion of how we have addressed the critical knowledge-sharing and decision-making issues from Chapter 4 in our plan-based approach to multi-agent opportunism. It is not difficult to see that these issues are highly intertwined, in that a choice of how one of these issues should be solved will influence how one or more of the other issues are addressed. For our approach, the first issue we tackled was question 4, how the agents should decide when it is feasible to provide opportunistic assistance.

We arrived at the decision process embodied in the POPG execution mechanism by assuming heterogeneous planning agents. Since we could not guarantee any particular planning mechanism, we needed to limit our assumptions about the plans these agents would use. This led us to develop the POPG representation, which focuses on the information needed to make decisions about potential opportunities: partially ordered actions with explicitly represented preconditions, effects, and resource requirements.

In defining the execution method, our interest was to provide a mechanism that would enable an agent to select the next action at execution time that would most likely lead to accomplishment of the most valuable goals, with an understanding that the agent is situated in a dynamic environment. This provides an agent a consistent (and transparent) way to decide whether it should apply its resources toward one of its own goals or one of another agent that has been predictively encoded in the plan.

The execution mechanism also provides the agent with a natural way to determine when it must suspend a goal. As such, limiting when to request opportunistic assistance to just suspended goals was a straightforward decision. Interestingly, though, the action selection procedure described in Section 5.3.2 could also be used to determine which active goals in the current plan might benefit from requesting opportunistic assistance. That is, the plan value estimation procedure defined by equations 5.2 and 5.3 could be used to identify "high-risk" goals, such as those with an expected cost that is just below some threshold of the currently available resources. We have not, however, pursued this idea, leaving it for future work.

Our POPG-based approach also quite naturally leads to satisfaction of suspended goals as the response to opportunity recognition. In this case, opportunity recognition is embodied in the action selection mechanism, which determines when conditions are favorable to pursue actions supporting a predictively encoded goal of some other agent. Similarly, the choice of action-taking as the response to opportunity recognition would seem to necessitate the use of goal-sharing as the method of determining what opportunity related information an agent should share. As we show in the next chapter, this is indeed an effective pairing given our MAS model.

The plan-based approach described here, however, is not limited to just satisfying suspended goals for other agents in response to opportunities. The POPG representation would allow any condition that can be described as an effect of a plan action to be viewed as a "goal" to be predictively encoded. For example, suppose that the condition $c_1 =$ `have-rock-sample(WP7)` is a precondition for some other set of actions in the plan, $\mathcal{P}_i$, for agent $\mathbb{A}_i$, and is not itself an assigned goal. If $\mathbb{A}_i$ could identify that c_1 is necessary for the achievement of a suspended assigned goal, g_1, then requesting opportunistic assistance for c_1 would actually be cue sharing. Determine what preconditions to use for opportunity cues in a domain-independent manner is not a simple matter, however, and is thus also left for future work.

Finally, the use of a middle agent to determine which other agents to request opportunistic assistance from is a direct consequence of our MAS model. That is, in the approach described here the agents defer to the task broker, $\mathbb{B}$, the decision of which agent should receive requests. The broker, in turn, uses its local knowledge of the capabilities of the agents in the MAS to select just those agents that can satisfy a given suspended goal. Since we are assuming an open MAS, we must also consider agents entering and leaving the system. When a new agent joins the system, it registers with the broker, and is included in the next cycle of planning and execution. If the new agent joins during an ongoing cycle, then some potential opportunities might be missed (i.e., the new agent may posses some information another agent could use), but missing opportunities is acceptable in our framework. When an agent leaves the MAS gracefully, it tells the broker about the change, which in turn informs the other agents in the MAS if they need to cancel any goals. Of course, if an agent just "dies," the broker never knows, and there is nothing that can be done. Any extra goals that other agents may have been attempting to opportunistically satisfy will not get canceled. On the other hand, if the recently deceased agent was unable to satisfy those goals, then having other agents try to satisfy them is a good idea.

In summary, our POPG-based approach to multi-agent opportunism addresses the critical issues from Chapter 4 as:

- When to request opportunistic assistance? Currently just for suspended goals, but "high-risk" active goals could also be used.
- What opportunity-related information should be shared? Currently goal-sharing, but cue-sharing could also be supported.
- Which other agents should be sent requests? All agents with appropriate capa-

bilities, deferred through the task broker.

- When to provide opportunistic assistance? Determined by the POPG plan execution scheme, primarily by the approximate decision theoretic action selection mechanism.

- How to respond to potential opportunities? The suspended goal is satisfied. Satisfying unmet preconditions could be supported as well.

Chapter 7

Empirical Evaluation

The previous chapters describe a plan-based model of multi-agent opportunism that is applicable to open systems of heterogeneous agents. In this chapter we describe our implementation of that model, along with the results of an empirical evaluation of the model in the simulated multi-agent environment.

7.1 Experimental Setup

Suppose that, as described in Section 6.1, an agent $\mathbb{A}_i$ considers planning not only for its assigned goals G_i, but also for a limited set of its other, opportunity-wise "most promising," capabilities. Can we determine in a domain-independent way how to select which extra capabilities to include? Can we generate enhanced plans containing extra capabilities that allow the agents to take advantage of opportunities, but without significant overhead if those opportunities never occur?

In an attempt to address these questions, we have implemented an evaluation testbed for MASs, in the form of a discrete-event simulation, using our planning and execution scheme. The benchmark problems we have used in the evaluation are based on the `Rovers` example described in Section 2.1. The simulator is written in

Portions of the content of this chapter have been published in [Lawton and Domshlak, 2004a,b; Domshlak and Lawton, 2004].

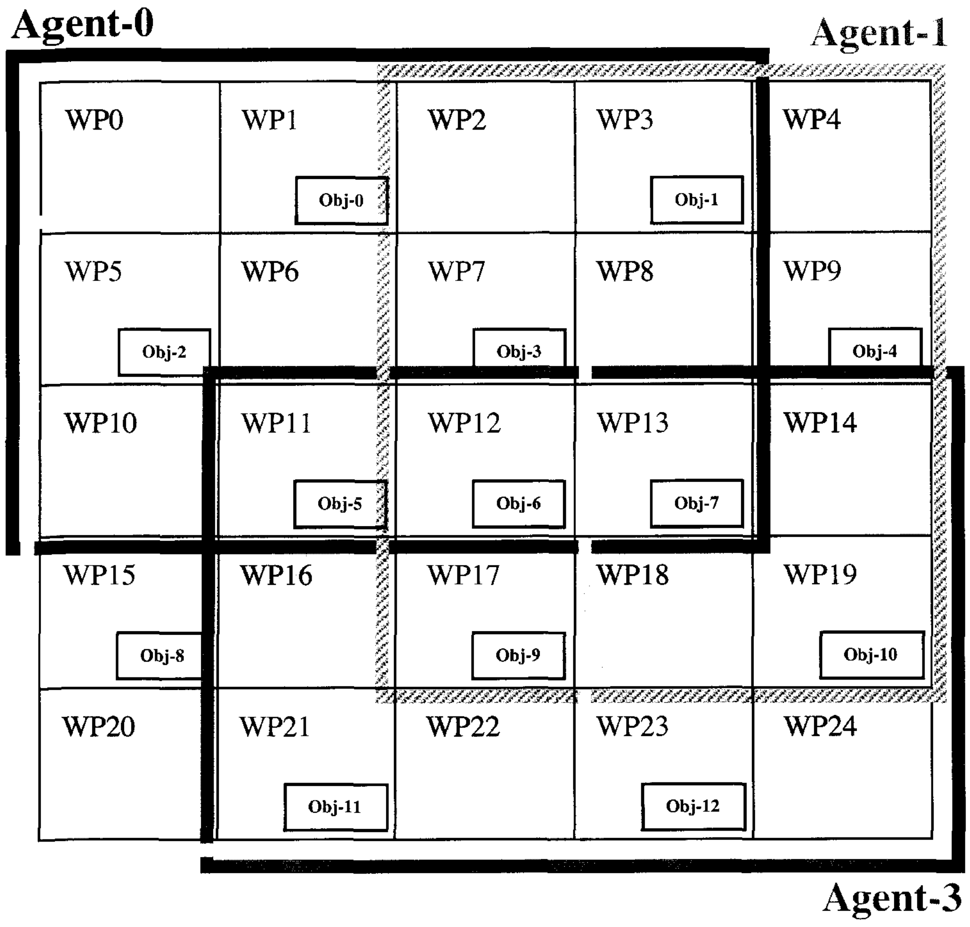

Figure 7-1: Workspace partitioning for Rovers MAS. (Repeated from Fig. 2-1)

Common Lisp; the experiments were run on several Sun workstations (UltraSparc III processors, running at either 900Mhz or 650Mhz).

As described in Section 2.1, the working area of the rovers consists of 25 waypoints, arranged in a 5×5 grid (see Figure 7-1). Each rover can only operate within a given 3×4 waypoint region of the grid. Within its assigned region, an agent can perform various scientific tasks such as soil sampling, rock sampling, and taking pictures of objects of interest if they are visible from the rover's current location. To reduce the problem's complexity that would be caused by a large number of potential goals, we have limited the number of locations where scientific tasks can be performed to 13 (those with a white background in Figure 7-1). Thus, there are total of 65 different goals that can be assigned to the rovers by the broker, namely 13 rock samplings, 13 soil samplings, and 13 objects to be photographed, where each picture can be taken at three different levels of quality.

The evaluation was performed on problem instances involving teams of 4 agents, $\{\mathbb{A}_1–\mathbb{A}_4\}$, with partially overlapping capabilities: The working area of the team was divided into 4 partially overlapping regions (see Figure 7-1), and each rover could operate only within its designated area. The scientific tasks that a rover $\mathbb{A}_i$ can perform constitute its set of capabilities C_i. These are restricted to tasks at the waypoints within the agent's region of operation. As mentioned above, we have restricted which waypoints will actually have goals associated with them, thus each agent will be assigned goals at only 7 of the 12 waypoints in its designated region. This way, each agent is capable of performing 35 of the 65 possible goals.

A 5×5 grid was selected because it is the smallest grid that would allow symmetric patterns of overlapping regions (and thus capabilities) such that there are waypoints accessible by exactly 1, 2, 3 and all 4 agents. For example, `WP1` is accessible by only agent $\mathbb{A}_0$, `WP5` can only be reached by agents $\mathbb{A}_0$ and $\mathbb{A}_2$, `WP11` is accessible just by agents $\mathbb{A}_0$, $\mathbb{A}_2$ and $\mathbb{A}_3$, and `WP12` is the only waypoint that all agents can get to. The symmetry in the region assignments is important because the goal assignments are generated at random, and we did not want to skew the load of any the agents.

At the beginning of each planning/execution cycle, each agent $\mathbb{A}_i$ is assigned a set of goals G_i from its capabilities (i.e., $G_i \subseteq C_i$) such that no goals are assigned to more than one agent (i.e., $\bigcap_i G_i = \emptyset$). Although this is a reasonable assumption for the `Rovers` domain, it is possible in other domains the goal assignments may indeed overlap. However assuming the goals are uniquely assigned allows us to focus on the effectiveness of planning for extra capabilities. The agents begin by planning for their

```
Procedure: TO2PO
Input: a valid total-order plan $(a_1, \ldots, a_n)$
Output: an equivalent partial-order plan $\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle$

for $i = n$ to 1 do
  Add $a_i$ to $\mathcal{A}$
  for $p \in Preconditions(a_i)$ do
    Choose $k < i$ such that:
       $p \in PositiveEffects(a_k) \wedge$
       $\nexists\, l$ such that $(k < l < i) \wedge (p \in NegativeEffects(a_l))$
    Add $a_k \xrightarrow{p} a_i$ to $\mathcal{L}$
  end for
  for $p \in NegativeEffects(a_i)$ do
    for $j = (i-1)$ to 1 do
      if $p \in Preconditions(a_j)$ then
        Add $a_j < a_i$ to $\mathcal{O}$
      end if
    end for
  end for
end for
Return $\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle$
```

Figure 7-2: Modified TO2PO algorithm (adapted from [Ambite and Knoblock, 2001]).

individual sets of goals using a domain-independent planning methodology. In our experiments the agents used the FF planner [Hoffmann and Nebel, 2001], but any "off-the-shelf" planner capable of producing plans for the IPC-2002 domains should be applicable. Since the IPC-2002 required the planners to produce a totally ordered plan, we used a version of Ambite and Knoblock's [2001] TO2PO algorithm (Figure 7-2), modified to include proposition nodes as well as action nodes, to produce the POPGs.

For the results reported here, an evaluation was performed on 100 randomly generated problem instances where each agent was assigned $k = 4$ goals. Given a problem instance for which the agents have generated individual plans $\mathcal{P}_1, \ldots, \mathcal{P}_4$, let $EC(\mathcal{P}_i)$ be the expected amount of energy required for $\mathbb{A}_i$ to fulfill its plan $\mathcal{P}_i$ completely. Each agent was allocated a fraction δ_i of $EC(\mathcal{P}_i)$, where δ_i is randomly chosen from a uniform distribution within $[0.5, 1.5]$. In addition to the k assigned goals, each agent was allowed to choose and plan for another $k' = 3$ capabilities, basing its choice on the knowledge available about the other agents. Recall that the idea is to select goals that might get suspended by other agents at runtime, thus predictively encoding potential opportunities. Since the focus of the evaluation has been on potential contributions of exploiting severely limited shared knowledge, in our experiments we have considered two different levels of such knowledge:

1. Individual Capabilities (CK): The agents have complete knowledge about each other's capabilities. Therefore, each agent $\mathbb{A}_i$ chooses for itself k' goals from:

$$\mathcal{C}_i = \left(\bigcup_j (C_i \cap C_j) \right) - G_i$$

i.e., from the capabilities that $\mathbb{A}_i$ shares with other agents that are not already in its set of assigned goals G_i.

2. Individual Goals (GK): The agents have complete knowledge about the goals, including their values, that have been assigned to every agent by the broker. Thus, each agent $\mathbb{A}_i$ chooses for itself k' goals from:

$$\mathcal{G}_i = \left(\bigcup_j (C_i \cap G_j) \right) - G_i,$$

i.e., from the assigned goals that happen to be in the capabilities of $\mathbb{A}_i$, but are not already in its set of assigned goals G_i.

It could be argued that the situation where the agents have *no knowledge* about each other would provide the true baseline for this evaluation. If, as discussed in Section 6.1, it was feasible for the agents to plan for their whole set of capabilities, even this extreme case could allow the agents to assist one another opportunistically. But as this is impractical due to the computational demands, when the agents share no knowledge about each other they would have to select a limited number of extra goals to include in their plans by making (possibly domain-specific) assumptions about one another's capabilities. Alternatively, the agents could choose extra goals from their capabilities entirely at random. While we certainly cannot dispute that this would represent an extreme condition of "severely limited shared knowledge," we believe that in most situations in the cooperative MASs that we have assumed, it is reasonable to expect that the agents would be able to determine each other's capabilities. Thus, we have used the CK condition as our baseline.

Given a particular level of shared knowledge (i.e., CK or GK), the agents need a way to select which of the candidate extra goals to predictively encode into their plans that best exploits that knowledge. In Section 6.1 we indicated that we were interested in domain-independent methods for selecting these extra goals. To that end, for these experiments we have developed two separate *choice functions*, one for selecting extra goals from each of the two agent capability subsets described above, $\mathcal{C}_i$ and $\mathcal{G}_i$. More accurately, the two functions, which we will refer to as CAPS_x and GOAL_y, represent classes of functions for selecting extra goals in different ways. We will explain each in turn.

First, let us consider the CK case, where the agents only share knowledge of each other's capabilities. Let $\bigcup_{j=1}^{n-1} \mathcal{C}_i^j$ (n is the number of agents) be a disjoint partition of $\mathcal{C}_i$, such that $\mathcal{C}_i^j$ consists of the capabilities of $\mathbb{A}_i$ that are also part of the capability sets of exactly j other agents in the system. The first $n-1$ (in our experiments, three) choice functions $\text{CAPS}_1, \ldots \text{CAPS}_{n-1}$ correspond to randomly choosing k' extra goals from $\mathcal{C}_i^1, \ldots, \mathcal{C}_i^{n-1}$, respectively. An additional choice function for the case of CK, denoted as $\text{CAPS}_{\text{norm}}$, picks k' extra goals from $\mathcal{C}_i$ at random, where the random choice is not uniform, but normalized with respect to the above partition of $\mathcal{C}_i$. The motivation behind this selection process is to attempt to normalize the chance that any given capability will be selected as an extra goal by only one agent. Thus, greater weight is given to capabilities shared be fewer agents, while capabilities shared by many agents have a lesser chance of being selected. Specifically, let $\gamma = \sum_{j=1}^{n-1} |\mathcal{C}_i^j|/j$. The probability of choosing $g \in \mathcal{C}_i^j$ is given by $1/\gamma j$.

Next we consider the GK case, where every agent knows about the goals, along with their values, that have been assigned to one another. This level of knowledge is better than in the CK case, since the agents can limit their selection to goals that they know actually have a chance of being suspended by some other agent. We would like the GOAL_y choice function to take advantage of this additional knowledge. One obvious choice function, which we will call GOAL_{max}, would have agent $\mathbb{A}_i$ select the k' capabilities with the largest values in $\mathcal{G}_i$. These would be the goals that are expected to make the greatest contribution to MAS. To compute GOAL_{max}, each agent $\mathbb{A}_i$ could sort $\mathcal{G}_i$ in descending order by value (i.e., V_g), and then select the first k' goals from the sorted list. Two additional choice functions that we will consider, GOAL_{min} and GOAL_{med}, correspond to choosing capabilities with the lowest and median values in $\mathcal{G}_i$, respectively. GOAL_{min} would be computed by sorting $\mathcal{G}_i$ in ascending order by value, and again selecting the first k' goals from the list. For GOAL_{med}, $\mathcal{G}_i$ could be sorted in either order, selecting the middle (median) k' goals. Median is used instead of mean for this choice function because it guarantees at least one of the k' goals will have the actual middle value.

7.2 Evaluation

For the purpose of this evaluation, a set of experiments with five separate cases was conducted. The results are presented in Figures 7-4 and 7-3. We will first explain how each experimental case was performed. We will then discuss the results presented in the figures. Each case was performed on the same 100 problem instances involving teams of 4 agents with partially overlapping capabilities as shown in Figure 7-1. In each problem instance the agents were assigned $k = 4$ goals. Each of these goals was assigned a constant value, V_g, randomly drawn from from a uniform distribution within [1,100]. The performance measure was the total goal value obtained by the MAS, computed as the sum of the values of each goal satisfied. For the cases that involved including extra goals in the plans, the agents selected an additional $k' = 3$ goals.

For the first case (the control case), the agents created and executed plans for the k assigned goals in each problem instance on their own, without using multi-agent opportunism. We will refer to this as the "No MAOpp" case. The results are shown as the "No MAOpp" lines in Figs. 7-3 and 7-4. This case shows how well the MAS performs under normal conditions, and thus serves as our baseline.

In the second experimental case, which we call the "Basic" case, the agents created and executed plans for the $k+k'$ assigned and extra goals using the basic methodology described in Section 6.1. They did not use any of the plan enhancement techniques described in Section 6.2. A total of seven experimental runs were performed for this case. For the first four Basic runs ("Basic" in Fig. 7-3), the agents only shared knowledge of one another's capabilities (i.e., the CK case). In these runs, the agents selected the k' extra goals using the four CAPS_x choice functions. For the remaining three Basic runs ("Basic" in Fig. 7-4), the agents had knowledge of one another's assigned goals (i.e., the GK case). In these runs, the agents selected the k' extra goals using the three GOAL_y choice functions.

In the third experimental case, the agents again created and executed plans for the same $k + k'$ assigned and extra goals used in the "Basic" case. They then enhanced those plans using the planning with shortcuts methodology described in Section 6.2. We refer to this as the "Shortcut" case. As with the Basic case, seven experimental runs were performed for this case: four using the CAPS_x choice functions ("Shortcut" in Fig. 7-3) and three using the GOAL_y choice functions ("Shortcut" in Fig. 7-4) to select the extra goals.

For the fourth case, which we will refer to as the "PPR" case, the agents used the predictive plan repair methodology to enhance their plans. That is, the agents created initial plans for the same k assigned goals as used in the "Basic" and "Shortcut" cases, and then enhanced those plans to include the same k' extra goals using the repair procedure described in Section 6.2. Again, seven separate experimental runs were performed in this set using the CAPS_x and GOAL_y choice functions to select the k' extra goals ("PPR" in Figs. 7-3 and 7-4).

For the fifth experimental case, which we will call the "RPR" case, the agents used the reactive plan repair mechanism from Section 6.2. That is, the agents initially created plans for just the k assigned goals (again, the same k assigned goals as used in the previous cases). They repaired these plans at runtime (if possible) to include goals that had been actually suspended by other agents. Because this methodology does not encode extra goals in the plans, the CAPS_x and GOAL_y choice functions were not used. The results for this case are shown as the "RPR" lines in Figs. 7-3 and 7-4.

The results of these experiments shown in Figs. 7-3 and 7-4 have been normalized against the baseline case, No MAOpp. That is, the performance measure used for these experiments was the total value obtained for satisfying the assigned goals over all 100 problem instances. Since the values assigned to these goals have no units of measure, it would not be meaningful to present the results in terms of these values. The results presented are thus the total value obtained by a given experimental run divided by the total value obtained by the baseline case, No MAOpp.

In each of Figs. 7-3 and 7-4, the lower, light-colored horizontal line depicts the results for the No MAOpp control case. Since this is our baseline case, its normalized value is, by definition, 1.00. It is presented as a line instead of a set of bar graphs in these figures since it really represents a single value (i.e., none of the choice functions

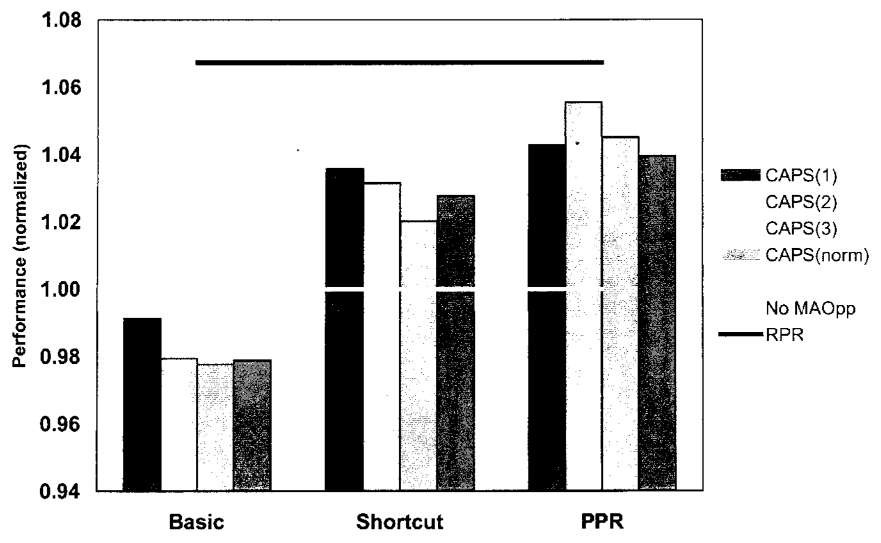

Figure 7-3: Normalized experimental results for capability-based selection.

were used in this case). Also, presenting it as a line spanning the entire graph makes it easier to visually compare the performance of the other cases to the baseline.

The results for the Basic, Shortcut, and PPR cases are presented as bar graphs in Figs. 7-3 and 7-4. In Fig. 7-3, each of these cases is shown as a group of four bars: one for each of the CAPS$_x$ choice functions. Similarly, in Fig. 7-4 each case is shown as a group of three bars: one for each of the GOAL$_y$ choice functions. The value of each of these experimental runs, represented by the height of the corresponding bar (where taller is better), is relative to the No MAOpp case. Thus a value of 0.98 would indicate the performance of that run was 2% worse than the performance of the No MAOpp case. Similarly, a value of 1.02 would indicate a 2% improvement over the baseline case. Finally, the upper, dark horizontal line in both graphs depicts the results of the reactive plan repair (RPR) case. As with the No MAOpp case, since no choice functions are used this result is really just a single value. It is thus easier to present this as a line instead of a bar graph.

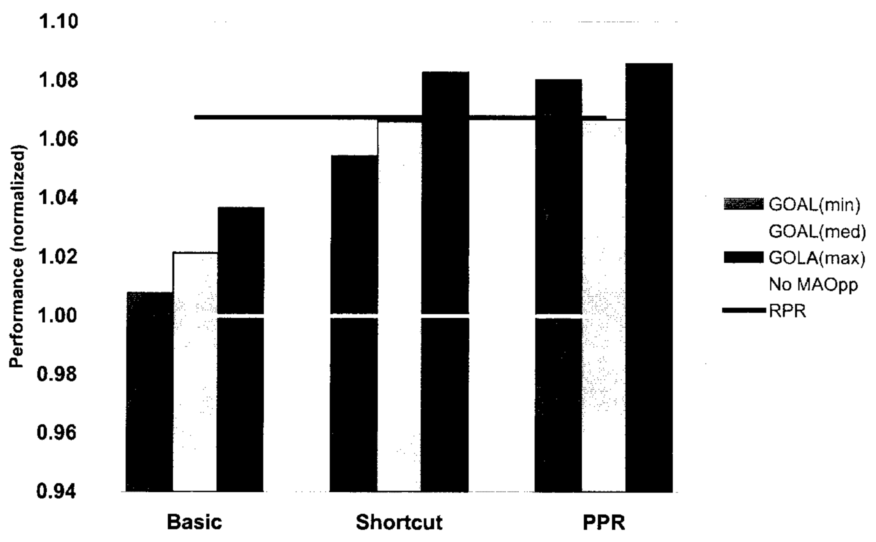

Figure 7-4: Normalized experimental results for goal-based selection.

The results of our experiments are also presented statistically in Table 7.1. Specifically, Table 7.1 presents three key statistics for each plan repair method combined with each choice function. The column labeled "Difference ± CI" presents the average improvement and confidence interval (at 95% confidence) of the given experimental run compared to not using opportunism (i.e., No MAOpp). This statistic is computed as the arithmetic mean of the per-problem-instance differences. Since we are using exactly the same problem instances, as well as the same random number seeds for each run, this is a better calculation of improvement than computing the difference of the average goal value of each run.

The column of Table 7.1 labeled "t-Test" presents the results of the paired two sample t-Test for means. Given the null hypothesis, H_0, that the two approaches (i.e. not using opportunism and using multi-agent opportunism with the given technique) are equivalent in that they will produce the same (average) total value, this statistic represents the probability the results could have been obtained by chance if H_0 were true. Thus, when p is less than our confidence threshold (0.05 for these experiments, representing 95% confidence in the results), we can safely reject H_0. Finally, the column of Table 7.1 labeled "% Difference" presents the average difference in the total value obtained in the given experimental run as compared to not using opportunism.

Let us first consider the results of the Basic experimental case. From Figs. 7-3 and 7-4 we can see that adopting multi-agent opportunism was only moderately effective when the agents knew about each other's assigned goals (i.e., the GK case), while it was actually *harmful* when the agents only knew about each other's individual capabilities (the CK case). It should be noted, though, that we cannot claim statistical significance for any of these results except the GOAL_{max} run (see Table 7.1). Because of the high variance in these results, we performed a second set of Basic runs on 400 different problem instances. Because the problem instances were different, we cannot do a direct comparison. However, these runs confirmed the same general results with better statistical certainty: performance was generally slightly better in the GK case (~3% improvement for GOAL_{max}, $p < 0.000004$; ~2% improvement for GOAL_{med}, $p < 0.006$; GOAL_{low} was statistically equivalent to the baseline), and generally worse in the CK case (~2% degradation for CAPS_2, CAPS_3, and CAPS_{norm}, $p < 0.004$; CAPS_1 was statistically equivalent to the baseline).

At first glance, given the decision-theoretic nature of the execution module, this is a somewhat unexpected result: Since each agent is attempting to maximize its (and thus the global) expected payoff, having more potentially valuable goals in a plan should only increase its flexibility, guaranteeing an improvement in the expected performance. Thus planning for extra goals, using either choice function, should lead to performance at least as good as when planning for only the assigned goals. The pitfall here is that this claim is sound only under an "all else being equal" assumption, i.e., only if we compare two qualitatively identical plans. In the case of plan-based

Method	Choice	Difference	± CI	t-Test ($p <$)	% Difference
Basic	GOAL$_{min}$	4.91	± 20.20	0.634899357	0.76%
	GOAL$_{med}$	13.76	± 20.20	0.184883010	2.13%
	GOAL$_{max}$	23.59	± 22.15	0.039436245	3.66%
	CAPS$_1$	-5.54	± 19.44	0.577778057	-0.86%
	CAPS$_2$	-13.27	± 19.33	0.181609496	-2.06%
	CAPS$_3$	-14.35	± 17.64	0.113943304	-2.23%
	CAPS$_{norm}$	-13.61	± 18.01	0.141788970	-2.11%
Shortcut	GOAL$_{min}$	34.97	± 19.02	0.000493012	5.42%
	GOAL$_{med}$	42.60	± 17.13	0.000004151	6.61%
	GOAL$_{max}$	53.34	± 17.98	0.000000075	8.27%
	CAPS$_1$	23.06	± 17.82	0.012783911	3.58%
	CAPS$_2$	20.33	± 18.40	0.012783911	3.15%
	CAPS$_3$	13.06	± 17.94	0.156734775	2.03%
	CAPS$_{norm}$	17.94	± 19.12	0.068872466	2.78%
PPR	GOAL$_{min}$	51.76	± 15.63	0.000000003	8.03%
	GOAL$_{med}$	43.02	± 17.63	0.000006048	6.67%
	GOAL$_{max}$	55.28	± 16.87	0.000000005	8.57%
	CAPS$_1$	27.55	± 17.52	0.002662352	4.27%
	CAPS$_2$	35.76	± 14.87	0.000007932	5.55%
	CAPS$_3$	29.05	± 17.12	0.001239015	4.50%
	CAPS$_{norm}$	25.46	± 16.37	0.002947308	3.95%
RPR		43.42	± 15.45	0.000000286	6.73%

Table 7.1: Statistical Results for `Rovers` experiments

predictive encoding, however, achieving k assigned goals using a plan created for these and some other k' goals can be far more complicated (and thus more risky and resource consuming) than achieving the k goals using a plan created only for them. These initial results are, in fact, the reason we examined the various plan efficiency improvements described in Section 6.2.

We next consider the cases using the various plan enhancement techniques: Shortcut, PPR, and RPR. From Figures 7-3 and 7-4, as well as from Table 7.1, it is easy to see that all three lead to an improvement in performance both over the baseline plan execution mechanism with no opportunism (No MAOpp) and over our original (Basic) approach to offline predictive encoding. Focusing first just on the results of the GK runs shown in Figure 7-4, we can see that none of the three advanced techniques seem to dominate the other two, despite the significant differences in the way that they enhance the agents' plans. One difference, however, between these techniques is in the time complexity of the corresponding planning and execution. In our experiments, the average execution time each agent spent per problem instance was 7.57 minutes with planning with shortcuts, 22.87 minutes with predictive plan repair, and only 0.43 minutes with reactive plan repair[1]. The corresponding execution time values for the runs without plan enhancements were an average of 2.34 minutes per problem instance when using the basic approach, and 0.02 minutes for the baseline NO MAOpp case. The dramatically shorter execution time for the reactive plan repair approach is not surprising, since the plans are modified only for goals that are known to be suspended. This leads to smaller plans to execute, with fewer contingency branches to consider, which in turn allows for faster execution. However, recall that reactive

[1]Since these experiments were run on comparable, but not identical computers, the precise relation between these numbers may slightly vary. Further, since identical computers were not available, and it required approximately 3 months to perform these experiments as presented here, it was impractical to attempt to use just one computer.

plan repair is feasible only if some degree of online re-planning is allowed. Otherwise, since we cannot show a statistically significant difference in performance between the other two methods, our results favor the use of planning with shortcuts.

In some ways, the results obtained with plan enhancement techniques while exploiting the less informative CK knowledge (see Figure 7-3) may be considered even more impressive than the GK results. Recall that our Basic approach to multi-agent opportunism actually produced a reduction in system performance as compared to not adopting opportunism at all (see the Basic group of bars in Figure 7-3). The results for CK with plan enhancements, however, show that we can significantly reduce the overhead involved in opportunistic planning, making adopting multi-agent opportunism attractive even in cases of limited shared knowledge.

It should be noted, though, that the limited number of possible goals in our experimental domain may make results for the CK runs appear more attractive than they are. Recall that in this scenario each agent's capability set contains only 35 goals. It shares 30 of these capabilities with other agents. In scenarios where the number of shared goals is greater, we would expect that the performance improvement due to multi-agent opportunism based on knowledge of shared capabilities to be reduced. But, our results also demonstrate that using multi-agent opportunism in conjunction with one of our plan enhancement techniques produces performance at least as good as, and often better than, not using opportunism at all. Thus, even as the number of shared goals increases, exploiting this knowledge can still lead to an improvement in performance with only limited additional overhead.

As with the GK results, neither planning with shortcuts or predictive plan repair was dominant when exploiting CK knowledge. Both produced moderate performance improvements compared to the baseline of not exploiting opportunities (an average of ~3% for planning with shortcuts, and ~4% for predictive plan repair). But again, since we cannot show a statistically significant difference in performance between these two methods, the difference in their execution time complexities (an average of 12.77 minutes per problem instance for planning with shortcuts, 26.65 minutes for predictive plan repair, and 2.35 minutes for the corresponding unenhanced basic approach) suggests a preference for planning with shortcuts. Of course, if the agents are able to re-plan at execution time, reactive plan repair would again be the preferred choice.

It might be argued that, even with our plan enhancement techniques, the improvements obtained by using multi-agent opportunism may not be worth the additional computational burden incurred. Indeed, Figure 7-4 shows that predictive encoding results in ~7% average improvement over the baseline, non-opportunistic planning and execution, while in Figure 7-3 the improvement is only ~4% on average. However, notice that each problem instance has a total of 16 goals assigned to the system of 4 agents. The value of each goal is selected at random from the uniform distribution between 1 and 100. Thus, the expected value of a goal is 50, and the expected total value of the 16 goals (if all were satisfied) is 800. Hence, a 7% improvement means an average increase of 56, or the equivalent of 1 goal that would have otherwise not been accomplished. Even a 4% improvement would provide an average increase of 32, which would likely indicate the accomplishment of an additional, lesser-valued goal that otherwise would have been unsatisfied. Our statistical analysis (Table 7.1) verifies that such qualitative improvements do in fact take place, and that they are statistically significant.

Finally, we must examine the impact of the various choice functions, GOAL_x and CAPS_y, on the performance of our experimental system. That is, given a particular level of knowledge (i.e., GK or CK), which method of exploiting that knowledge results in the best system performance? We will examine the two different levels of shared knowledge in decreasing order of how informative they are.

First, consider the results achieved when the agents share knowledge of one another's goals (i.e., GK), using the choice functions $\mathrm{GOAL}_{\mathrm{max}}$, $\mathrm{GOAL}_{\mathrm{min}}$, and $\mathrm{GOAL}_{\mathrm{med}}$. One may expect that predictive encoding of most the valuable potential opportunities (i.e., $\mathrm{GOAL}_{\mathrm{max}}$) will make multi-agent opportunistic execution more effective. In fact the results depicted in Figure 7-4 appears to support this expectation.

However, if we examine the results presented in Table 7.1, we cannot make this claim with statistical certainty for these experimental runs. This is because the confidence intervals of the "Difference" value actually overlap (due to the large variance in the values), effectively making them statistically equivalent. We should note, though, that we can claim with statistical certainty that all of the "Difference" values obtained using the various GOAL_x choice functions are indeed improvements, since they are all positive and the confidence interval is smaller than the difference value for each of them. This claim is further supported by the corresponding t-Test results.

For the experiments presented here, the evidence does appear to support the use of $\mathrm{GOAL}_{\mathrm{max}}$ as the appropriate choice function for GK. We believe, however, that in the general case of multi-agent opportunism, the attractiveness of some potential opportunity g for an agent $\mathbb{A}_i$ is captured by a more complex relationship. In fact, given a lack of any domain-specific knowledge that would suggest favoring the most highly valued goals, we would actually recommend selecting the more average valued goals, which for our example domain is the $\mathrm{GOAL}_{\mathrm{med}}$ choice function.

Suppose that an agent is asked to estimate some quantity that depends on a random variable X, but the agent has only limited knowledge (or no knowledge at all) about the properties of X. Adopting a Bayesian approach will lead the agent to base its estimation on the "most unbiased" assumption about X (e.g., assuming uniform distribution in the face of zero knowledge about X). Now, consider the case of GK, where the agents know about one another's assigned goals along with their values, V_g. When we consider the approximate decision-theoretic behavior of the agents when selecting their courses of action, the likelihood that an agent will suspend a goal g with a large value V_g is expected to be relatively small. This is because the agent assigned to achieve g will implicitly prioritize this goal. As such, including g as an extra goal in some other agent's plan would be expected to be wasted effort. For similar reasons, the likelihood that an agent will suspend one of its least valuable goals is expected to be relatively high. However, even if some agent does predictively encode such a goal, achieving it will be least beneficial to the overall success of the MAS. Therefore, if the agents share no knowledge except the assigned goals and their values, a Bayesian approach will lead them to select goals that represent a balance between the their values and the likelihood that they will be suspended. In our system, the GOAL_{med} choice function provides this balance because the goal values are uniformly distributed. Other value distributions may require the use of a different measure of central tendency other than median. Notice that in our experimental scenario, as the allocation of capabilities is symmetric among the agents, and the process of generating problem instances is completely unbiased, following a Bayesian approach in this MAS should be the correct thing to do. While our empirical results do not demonstrate a clear dominance of GOAL_{med}, they are not inconsistent with that selection, in that the GOAL_{med} choice function led to what we could characterize as an average improvement in system performance.

Now let us consider the results achieved when the agents only share knowledge of one another's capabilities, using the choice functions CAPS_1, CAPS_2, CAPS_3, and CAPS_{norm}. As with the case of GK, the results for CK shown in Figure 7-3 do not indicate that one choice function dominates. Also as with GK, the statistical results for CK in Table 7.1 show that the choice functions are again essentially equivalent, and that we can claim that the "Difference" values are indeed improvements for all but two case (CAPS_3 and CAPS_{norm} when planning with shortcuts).

We can again follow a middle-ground approach to recommending which choice function to use in the absence of domain-specific information. Consider the partition of agent $\mathbb{A}_i$'s capabilities $\{\mathcal{C}_i^j\}$ defined in Section 7.1. Since $\mathbb{A}_i$ has no knowledge about the value of the goals that have been assigned to the other agents, assuming an identical value distribution over $\mathcal{C}_i^1, \ldots, \mathcal{C}_i^{n-1}$ is at least as good as any other assumption (and thus is preferred in the Bayesian approach due to its minimal bias). Similarly, we can assume that for each agent $\mathbb{A}_i$ all the elements in its capability set C_i are equally likely to be assigned to $\mathbb{A}_i$ as its actual goals. However, this assumption implies that, for $1 \leq l < m \leq n-1$, the elements of $\mathcal{C}_i^m$ are more likely to be assigned to some agent (and thus more likely to be suspended!) than the elements of $\mathcal{C}_i^l$. On the other hand, the actually *assigned* elements of $\mathcal{C}_i^m$ are also more likely to be opportunistically accomplished by agents other than $\mathbb{A}_i$, because the likelihood of a "collision" on $\mathcal{C}_i^l$ is lower than it is on $\mathcal{C}_i^m$. In other words, while it is more likely that a capability that is shared by many agents will be suspended by some agent, it is also more likely that some other agent will be able to opportunistically satisfy that goal. Similarly, since the capabilities shared by only a few agents are less likely to be suspended, it is also less likely that some other will be able to opportunistically satisfy those goals. Thus the choice function $\text{CAPS}_{n-1/2}$ can be arguably expected to lead to a better balance between the likelihood of a goal being suspended and it being predictively encoded by some other agent. While our empirical results again do not demonstrate that CAPS_2 is the dominant choice function, they are not inconsistent with that selection. The results produced when using CAPS_2 were superior (albeit weakly) for predictive plan repair, and essentially average when planning with shortcuts.

7.3 Summary

In this chapter we have described an empirical analysis of the approach to multi-agent opportunism presented in Chapters 5 and 6. In particular, in this study we have focused on examining how to improve the performance of an MAS through multi-agent opportunism when the knowledge shared by the agents is severely limited. Our results demonstrate that with careful enhancement to an agent's core plans, multi-agent opportunism does indeed produce results as least as good as, and often better than, not using multi-agent opportunism. Further, we have argued that, in the general case, the best way to exploit limited shared knowledge for selecting extra goals is through a "middle-ground" approach that balances the chances a goal will be suspended and the chances that another agent will be able to opportunistically satisfy it. In the next chapter, we examine how our approach to multi-agent opportunism may be used in systems of agents that differ from the MAS model that we have used in this study.

Chapter 8

Other Multi-Agent Systems

In this chapter we discuss how our POPG-based approach could be applied to other types of multi-agent systems. Specifically, we will examine multi-agent opportunism in an autonomous oceanographic sampling network MAS (as presented in Chapter 2), and an aircraft maintenance information system.

8.1 Autonomous Oceanographic Sampling Networks

In Section 2.2, we introduced an example scenario for a specific type of autonomous oceanographic sampling network (AOSN) that uses a multi-agent system for collecting scientific data in the ocean. We again point out that this is only one particular example of how an MAS could be used to control an AOSN.

We base our example MAS on work that has been done in the CoDA project [Turner and Turner, 1998]. CoDA focuses on how to self-organize, control, and re-organize large, heterogeneous systems of vehicles and instrument platforms to allow them to be effective AOSNs. The project takes a cooperative distributed problem solving (CDPS) approach to the problem in which a loosely-organized *meta-level organization* (MLO) first self-organizes to discover the capabilities present in the AOSN. It then analyzes the situation and AOSN and designs a *task-level organization* (TLO) to actually carry out the mission. This organization could be a hierarchy, a team, a consensus-based organization, or other organizations as the situation warrants. If later there is too much change in the situation for the TLO to continue effectively, a new MLO forms, which creates a new TLO to fit the changed situation.

Here, we consider one type of organization that can be produced by CoDA's MLO, a hierarchy. To make the example more concrete, we will assume that the agents in this hypothetical AOSN are controlled by the schema-based planner Orca [Turner, 1994]. This type of MAS differs from the `Rovers` example used throughout this dissertation in two important ways. First, the agents are not using a "classical" (i.e., STRIPS-based) planning system to derive there intentions. Rather, the planner uses task decomposition to create a type of *task network* [Sacerdoti, 1977]. Second, the example MAS is coordinated as a hierarchy, which is a collection of cooperative agents organized into a network of manager and labor agents, instead of through a middle agent. We will address each of these differences in turn. One should note, however, that there is nothing about the AOSN domain that would prevent the agents from using classical planning or from using a middle agent for coordination. Further, even within the context of this specific MAS, other approaches to planning and coordination could be used. Rather, we provide this discussion to demonstrate how the characteristics of the MAS exemplified in this case would affect the use of our framework.

For the purposes of this discussion, we will assume that the AOSN agents will operate in a manner similar to the agents in the MAS model described in Chapter 6. Specifically, we will assume that they will request opportunistic assistance for suspended goals, will use goal-sharing as the opportunity cue selection method, and will respond to potential opportunities by satisfying the suspended goals when possible, notifying the other agents when done. Thus the following discussion will address how the agents can determine when they should provide opportunistic assistance, and how requests for opportunistic assistance can be directed to the most appropriate agents.

8.1.1 AOSN Planning

Since the approach to planning used in the AOSN example is not based on classical techniques, we cannot directly use our POPG-based approach to multi-agent opportunism with these agents. We are left with two options: we can either adapt our POPG evaluation mechanism to work with plans expressed as task networks, or we can translate the task networks into POPGs. Since there are existing techniques for translating task networks into partial order plans, we believe the latter option would be easier to accomplish (although still not trivial). As such, we will limit our discussion to this topic.

Task networks represent plans as a hierarchical network of nodes. The root node is a task node that represents the initial set of goals assigned to the agent. Task nodes are decomposed into subtasks, which may be primitive action nodes or other task nodes, possibly with constraints (e.g., ordering) relating the subtasks to one another. Primitive actions occur only as leaf nodes in the network. Task networks are generally considered to be more expressive than the STRIPS-based operators used by POPGs, in that they can represent recursive actions and complex constraints among the tasks [Erol et al., 1994; Lotem and Nau, 2000]. Thus, it is not a simple matter to translate these plans into POPGs. Below we offer several options for how to address this issue.

If a task network expresses a complete plan, one in which all of the leaf nodes are primitive actions with well-defined preconditions, effects and resource requirements, there are two ways we can translate it into a POPG. We should note that it is not common for the schema-based plans used in our example AOSN to be complete. We will, however, address this case first for the sake of other task decomposition planners that may produce complete plans. Later we discuss what can be done with partially decomposed task networks.

For the first method for translating a task network into a POPG, if we assume the plan is valid, then we know there is at least one total order of the actions. If we can extract this total order of actions, then just as we did in our experimental MAS described in Chapter 7, we can use our modified `TO2PO` algorithm [Ambite and Knoblock, 2001] to create a POPG. Alternatively, when we do not know or do not want to compute a total order of the actions, we could use a version of the algorithm presented by Lotem and Nau [2000] for translating task networks into STRIPS-based partial order plans, modified to explicitly include precondition and effect propositions as nodes, and derive the qualitative part of the POPGs from those plans. The quantitative part of the plan would be specified with the values of the assigned goals and the costs (resource requirements) of the primitive actions. Dummy actions, which are used to represent internal (non-primitive) nodes from the task network, would have no cost, as the cost of "performing" them is really just the resource needs of the primitive actions needed to meet their preconditions.

A limitation of the translation algorithm described above is that it only works on non-recursive task networks. It can, however, be used on partially decomposed task networks, allowing us to relax the above complete plan assumption. Task networks may be only partially decomposed when using active planners (see Section 3.2.2)—those that interleave planning and acting. For instance, the planning component of Orca uses *delayed commitment* to decompose subtasks only when needed, in the runtime context in which they will be addressed [Turner, 1994]. Thus at any given time its current plan may be represented by a partial task network.

As an example, suppose that one of the EAVEs (e.g., EAVE-ARIEL) has been assigned to sample a given volume of water (i.e., the `background-survey task`), and has decomposed that task into the (partial) task network shown in Figure 8-1. In this network, the plain boxes represent tasks or subtasks that must be decomposed, while the double-walled boxes represent primitive actions. Solid edges represent decom-

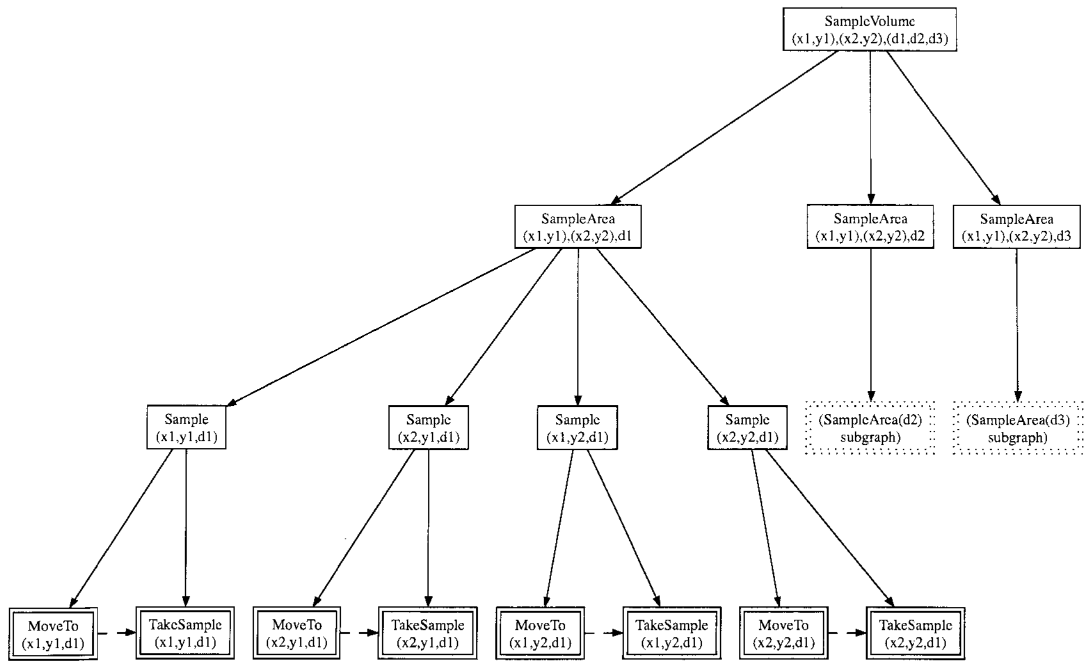

Figure 8-1: AOSN task network (partial) for sampling a volume of water.

position links, while dashed edges show ordering constraints. Following the method described by Lotem and Nau [2000], we can translate this task network into the (partial) POPG shown in Figure 8-2 [1]. As in Figure 8-1, in Figure 8-2 the double-walled boxes represent primitive actions. The plain boxes in this figure correspond to the decomposed tasks in the task network, and are thus in effect dummy actions. The elliptical nodes in Figure 8-2 represent the precondition and effect propositions of the action nodes. Artificial "task-done" nodes are used to represent the effects and preconditions of the dummy action nodes.

Once an agent translates its task network into a POPG, it could then use this structure to decide when it may provide opportunistic assistance. That is, when some agent, $\mathbb{A}_i$, notifies the MAS that it would like help in satisfying one of its sus-

[1]This POPG has actually been simplified, eliminating what would be redundant `Sample(X,Y,Z)` nodes.

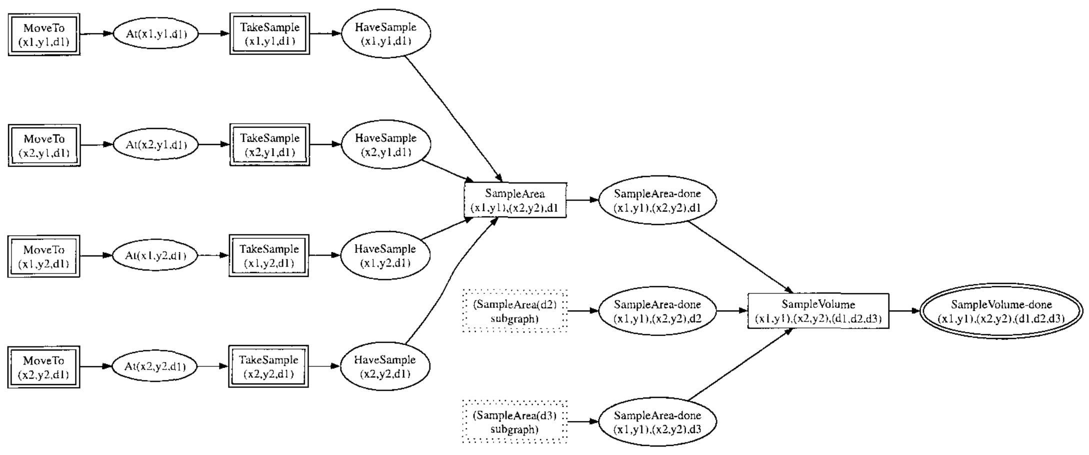

Figure 8-2: AOSN POPG (partial) for sampling a volume of water.

pended goals, g_i, some other agent, $\mathbb{A}_j$, could examine its intentions—as represented by its POPG $\mathcal{P}_j$—to determine if $g_i \in \mathcal{P}_j$. For example, `EAVE-Arista` may request opportunistic assistance in obtaining a sample from location $(x1, y1, d1)$, which, as shown in 8-2, `EAVE-Ariel` intends to obtain.

Unless it produces a complete (total order) plan, it is unlikely that an agent using task decomposition to derive its plans will be able to make direct use of the POPG execution mechanism described in Section 5.3. It should, however, be able to incorporate the plan evaluation mechanism (Equations 5.3 and 5.2) into its task selection mechanism. For example, Orca uses a context-sensitive *agenda manager* to determine which task should receive the focus of its attention. A mechanism like this could be augmented to consider input from the POPG evaluation when determining which task should have priority. As the task network is expanded, though, the corresponding POPG would have to be updated accordingly in order to ensure the task selection mechanism is working with current information.

A final consideration for applying our multi-agent opportunism approach to systems using task decomposition is how to incorporate planning for and predictively encoding extra goals into the task networks. The first question we must address is how will the agents select which extra goals to include? As long as the agents have access to information about each other's capabilities, or better still about their assigned goals, they should be able to select additional goals to plan for using the methods described in Chapter 7. In the next section we discuss how this information could be obtained by the agents in this AOSN example. Of course, as we indicated in Section 6.1, an agent's capability set may be huge, and even its explicit representation may be intractable. However, as discussed in Section 7.2, as long as we are careful in how we enhance the agents' plans, we should still be able to exploit this knowledge for multi-agent opportunism and produce results as good as, and hopefully better than, not using it.

Once an agent selects a set of extra goals, they would have to be predictively encoded into the agent's plan. One way to do this would be to include them in the root node along with the assigned goals, which would then be decomposed into a task network as usual. The task network could be translated into a POPG and used as described above. An agent like Orca that interleaves planning and acting, however, may never actually decompose the extra goals. This should not be a problem, though, since the planner would begin to decompose an extra goal into sub-tasks when it actually gets suspended by some other agent (i.e., when its value becomes > 0). The existing task selection mechanism (augmented with the POPG evaluation mechanism as mentioned above) would then determine if and when taking action to satisfy the suspended goal is feasible.

8.1.2 AOSN Coordination

As we have seen from the `Rovers` example, the MAS coordination mechanism can be used to support multi-agent opportunism by allowing the agents to obtain information about one another (e.g., their capabilities and assigned goals), and help direct requests for opportunistic assistance to the appropriate agents. In the example AOSN MAS, we can exploit the hierarchical organization for both of these purposes.

As discussed in the previous section, the agents need information about each other's capabilities and assigned goals in order to determine which extra tasks should be planned for and predictively encoded. In a hierarchical organization, agents can obtain information of this type from their task managers. While the managers may not have complete knowledge of the capabilities of the agents under their control (e.g., in our example MAS much of that information is lost when the MLO is dissolved), they should at least know about the capabilities related to the goals currently assigned to those agents. For example, consider the example AOSN organization from Section 2.2 shown in Figure 8-3 (repeated from Figure 2-3). Note that, as mentioned in Section 2.2, while we have used a hierarchy for the management structure in this example, it is not the only possible structure for an AOSN MAS. As long as the agents can communicate information about their capabilities, goals and opportunity cues through the network of managers, any organizational structure should suffice.

In the AOSN example, when considering its plans for the `background-survey-task`, EAVE-ARIAL can ask the manager of this task what other agents are involved, as well as what their capabilities are. The task manager (EAVE-ARISTA) would inform EAVE-ARIAL about the capabilities of itself and AUV-1, and it could even include information about other assigned tasks (e.g., the `rock-collection-task` of EAVE-ARISTA). EAVE-ARIAL might then decide to use this information to include other tasks (e.g., looking for unusual magnetic readings) to prepare for potential multi-agent opportunities. Interestingly, since some agents are their own task managers (e.g., EAVE-ARISTA on the `background-survey-task`), they could exploit information they already possess about other agents, such as the capabilities of the agents

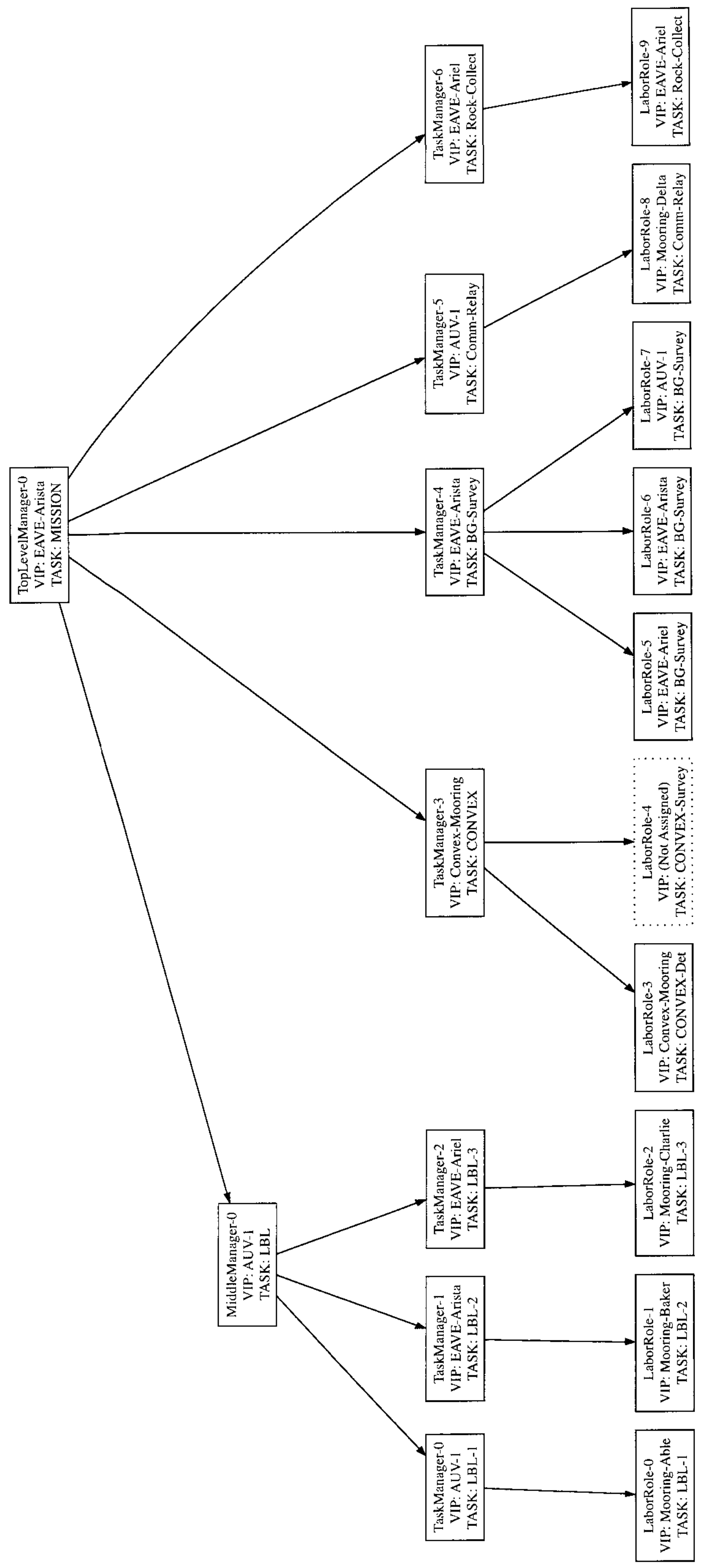

Figure 8-3: Example of a hierarchical organization for the AOSN MAS.

involved in the task they are managing, without communication.

Further, this scheme of querying task managers could also be used recursively, where task managers query higher-level managers for information about larger sub-coalitions. This capability would be important when agents with overlapping capabilities have been assigned to different subtasks. In such a case it would be necessary to forward the query to other task managers to ensure the agents with the appropriate capabilities are informed of the need for opportunistic assistance. Alternatively, in situations where a given task requires agents with similar or overlapping capabilities (e.g., the `background-survey-task`, where the agents all collect the same type of information from different volumes of water), the best information would be expected to come from the immediate managers of groups of labor agents.

We can also exploit the organizational structure to distribute the opportunity cues. Since we have assumed that we are using goal-sharing, the opportunity cues in this example would be the suspended goals. When an agent suspends a particular goal, it would be sent up the management chain as an opportunity cue. As cues come up the chain, managers would determine which subordinate agents (or sub-groups) possess the capabilities to address the cue (i.e., the suspended goal) and would send it to the appropriate subordinates. Managers could determine which subordinates possess the required capabilities and resources by examining the same knowledge they use when allocating tasks to these agents. The agents receiving the cues would incorporate them into their own opportunistic processing, if possible, as described in the previous section.

8.2 Aircraft Maintenance Information System

Another domain we will consider is a multi-agent information system, in which software agents collaborate to assist in aircraft maintenance. This example has been

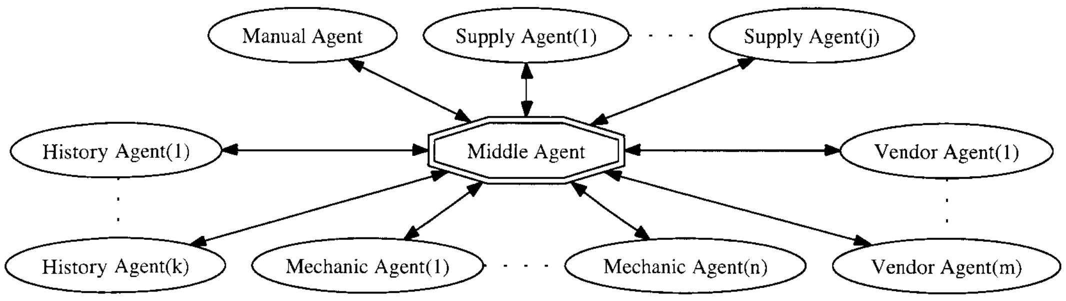

Figure 8-4: Aircraft Maintenance MAS

loosely adapted from the multi-agent system described by Shehory et al. [1999], which was designed to support aircraft maintenance at Warner Robins Air Force Base. Briefly, our example repair process starts with a mechanic inspecting an aircraft for problems. When a problem is found, the mechanic inputs a description to a user interface agent, which we will call the MechanicAgent. This agent finds and contacts information agents that maintain databases on previous repairs (HistoryAgents) and repair manuals (ManualAgents).

The information agents return documents describing previously performed repairs for similar problems and manual pages related to the possible repairs. The MechanicAgent (probably through interaction with the mechanic) would decide on the repair to be made and contact a SupplyAgent to obtain the needed parts. If the parts are in stock, the SupplyAgent arranges to have the parts delivered to the mechanic. Otherwise, the SupplyAgent would contact appropriate part VendorAgents, acquire the needed parts, and then deliver them to the mechanic. When the repair has been completed, the mechanic documents the details of the actual repair made with the MechanicAgent, who files that document with the local HistoryAgent for future reference.

As in the `Rovers` example, the agents in this MAS, shown in Figure 8-4, find one another through the use of middle agents, such as matchmakers or brokers. This MAS is a good example of an open multi-agent system, since the member agents are heterogeneous and may change over time. In fact in this example the agents are likely to also be geographically distributed over several repair bases. The agents in this MAS do not have *a priori* knowledge of which other agents will service their requests, and may never know about all of the other agents in the MAS. For example, a MechanicAgent might find out which HistoryAgents had information relating to a given repair from a matchmaker, and then contact them directly with information requests. Similarly, the MechanicAgents may never know of the existence of VendorAgents within the MAS, since they never use them directly.

As in the previous section, for the purposes of this discussion we will assume that the Aircraft Repair agents will operate in a manner similar to the agents in the MAS model described in Chapter 6. Specifically, we will assume that they will request opportunistic assistance for suspended goals, will use goal-sharing as the opportunity cue selection method, and will respond to potential opportunities by satisfying the suspended goals when possible, notifying the other agents when done. Further, since this MAS also uses middle agents, the questions of how agents find out about one another's capabilities and goals, as well as which agents should receive requests for opportunistic assistance, could also be addressed as in our model MAS. Thus in the following discussion we will address how the agents can determine when they should provide opportunistic assistance. Specifically we will examine two key issues: how can an agent exploit its current plans to determine if providing opportunistic assistance is feasible, and how can it prepare its plans for supporting multi-agent opportunism.

An agent in this system (e.g., a MechanicAgent) would likely use *query planning* [Ambite and Knoblock, 2000] in order to assemble the needed information from the various agents. Query planning involves determining a (partially ordered) sequence of data retrieval and data manipulation actions to satisfy a user's (or in this case, agent's) information request. This type of planning allows for information from different sources to be integrated in order to satisfy complex queries.

It has been shown [Ambite and Knoblock, 2000] that query planning can be done using classical operators, where the plan actions are often database-like operations (e.g., retrieve, join, union, etc.) with information items acting as preconditions and effects. As such, it should be possible to represent an information agent's plans using POPGs, as long as we assume the information-based preconditions and effects can be represented propositionally. Knoblock [1996] has used *functional predicates*—predicates defined by functions that compute some variables given the values of other variables—to represent query plan preconditions in classical plans. Functional predicates can represent plan conditions with enough detail for planners to reason over them, while still being abstract enough to represent information being gathered from different sources. Thus it is likely that a query-planning agent could make direct use of the approach discussed in Chapters 6 and 7 to exploit its plans for supporting multi-agent opportunism, with the possible exception of one important issue: resource usage.

The resource needs of plan actions, along with goal values, are used in our approach to enable an agent to determine its "best course of action." That is, these values are used to allow an agent to both decide when it should suspend one of its own goals, as well as to determine when it should attempt to opportunistically satisfy another agent's suspended goal. In an information system, the "cost" of an action may be expressed simply as the expected amount of time needed to execute it, or possibly in monetary terms (i.e., a charge for accessing some information servers). In some domains, the cost of assembling needed information might be considered negligible, such as when the system's users own all the resources. In other domains, gathering information may have the potential to be extremely expensive, as when an aircraft must go into a hostile region to gather intelligence, or when a doctor must perform a biopsy to gather information for a diagnosis.

When there are resource needs associated with the actions of an information query plan, it should be possible to make direct use of the POPG execution mechanism described in Section 5.3. It should also be possible to use the method for determining when to suspend goals, as well as the mechanisms for determining when providing opportunistic assistance are feasible, which are discussed in Section 6.1.

For example, consider the (partial) query plan shown in Figure 8-5, which might be used by a MechanicAgent to obtain a plan for repairing a particular problem instance, `F16-Rudder-123`. The MechanicAgent must first obtain a recommendation for making the repair, which it can get from either a HistoryAgent or a human planner. The agent may know (perhaps from past experience) that it takes an average of 5 minutes to obtain a response from a HistoryAgent, and 24 hours to obtain a response from a human planner. If time is considered a valuable resource, then the rational choice for the agent is to query the HistoryAgent first to obtain the repair recommendation.

Consider the situation, however, when the HistoryAgent is queried but cannot return a recommendation, and the MechanicAgent has only been allocated a limited amount of time (e.g., 2 hours) to come up with a repair plan. In this case, using our POPG execution scheme would allow the agent to recognize that this plan is likely to fail (because is has insufficient resources). While it might be best for the agent to handled this failure by reconsidering its intentions, if, as in the `Rovers` example, we assume that the agents cannot re-plan at runtime, this failure would cause the agent to suspend the associated goal. This in turn would lead the agent to request opportunistic assistance for the suspended goal. Similarly, an agent receiving notifi-

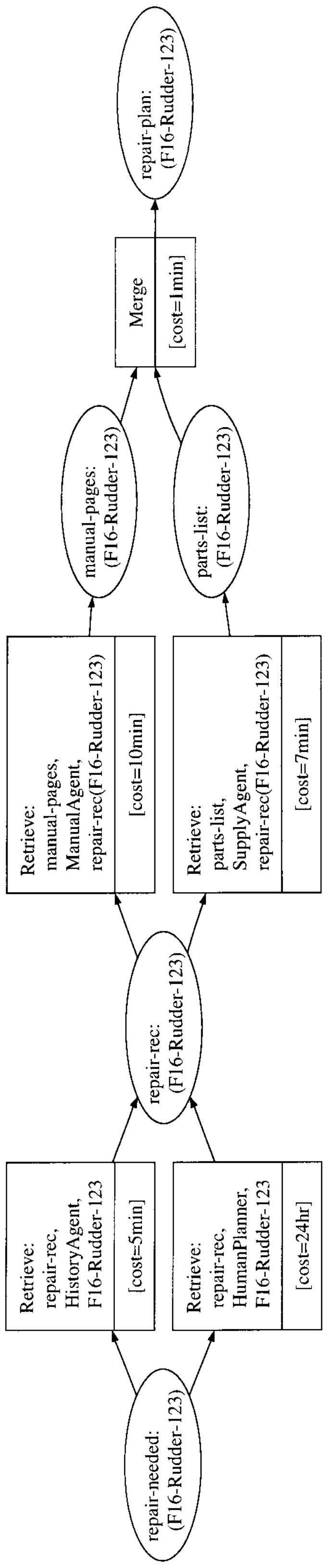
repair-needed: (F16-Rudder-123)
Retrieve: repair-rec, HistoryAgent, F16-Rudder-123
[cost=5min]
Retrieve: repair-rec, HumanPlanner, F16-Rudder-123
[cost=24hr]
repair-rec: (F16-Rudder-123)
Retrieve: manual-pages, ManualAgent, repair-rec(F16-Rudder-123)
[cost=10min]
Retrieve: parts-list, SupplyAgent, repair-rec(F16-Rudder-123)
[cost=7min]
manual-pages: (F16-Rudder-123)
parts-list: (F16-Rudder-123)
Merge
[cost=1min]
repair-plan: (F16-Rudder-123)

Figure 8-5: Example Aircraft Repair Query Plan (POPG)

cation of this suspended goal could examine its current plans to determine if it might be able to satisfy the goal opportunistically, and use our course of action selection mechanism to determine if it has the resources to help. Since we are assuming the agents in our MAS are cooperative, we would expect them to provide this assistance when possible.

There may be situations, however, where the cost of an action in a query plan is considered negligible. For example, a MechanicAgent executing the query plan in Figure 8-5 may not know (or care) how long information retrieval may take. It is still possible, though, for an agent to use a POPG's unmet preconditions to determine when it should suspend a plan and request opportunistic assistance. While we have generally assumed throughout this dissertation that plan actions always succeed, action failure is actually quite likely to occur in real-world systems. The result of action failure is that one or more conditions will not be achieved. When these conditions are preconditions of some future actions leading to a goal, these future actions cannot be performed. Either alternative actions will have to be performed to provide the needed preconditions, or the goal will have to be suspended.

Continuing with the example POPG shown in Figure 8-5, suppose that neither the HistoryAgent nor the human planner returns a repair recommendation (perhaps because of a communication failure). In this case, the `repair-rec(F16-Rudder-123)` condition will not be met, so the goal `repair-plan(F16-Rudder-123)` will have to be suspended, and the agent may request opportunistic assistance with achieving this goal. The agent might use the goal itself as the opportunity cue (i.e., goal sharing), as we had assumed above. The agent may, however, be able to use cue sharing or mixed sharing as discussed in Section 4.2.1. That is, since the agent knows the specific condition that is impeding its plan's progress (i.e., `repair-rec(F16-Rudder-123)`), it may request opportunistic assistance in satisfying just this condition (cue sharing), or it may include both this condition *and* the suspended goal in its request (mixed sharing). Since the conditions in information query plans can themselves be viewed as *information* or *knowledge goals* [Ambite and Knoblock, 2000; Ram and Hunter, 1992], agents using such plans are more likely to be able to utilize cue sharing or mixed sharing. This is because information held by one agent is, in general, more easily communicated to and utilized by another agent than "physical" preconditions (e.g., being at some location or possessing some object).

Although it may be straightforward to use unmet precondition to determine when to suspend a goal (and thus when to request opportunistic assistance), a lack of explicit resource needs in a query plan would make it difficult for an agent to use our plan evaluation mechanism (Section 5.3) to determine its best course of action. That is, if there is some risk of having insufficient resources (e.g., time) to satisfy all goals in a query plan, but those resources are not explicitly represented or known, it would not be possible for an agent to select a course of action that minimizes this risk. On the other hand, if the cost of executing a query plan is considered negligible, then any valid total order of the partially-ordered actions in a POPG would be equivalent. In this case, an agent would always choose to opportunistically satisfy another agent's suspended goal (or intermediate precondition), since it can only increase the overall benefit to the MAS.

This brings us to our final consideration for multi-agent opportunism in query plans: When would an opportunity for one agent to help satisfy another agent's goals even occur? That is, when would an agent's plan already have a condition included that may be a goal or other precondition for another agent? It is true that some opportunities might occur naturally when two agents generate plans for similar activities (e.g., two agents may request the same pages from the ManualAgent). In this case, the approach described in this section would indeed be able to exploit these opportunities. But, in general such natural opportunities cannot be relied on, and so our efforts must be focused on artificially increasing the number of potential opportunities.

In the approach described in Chapters 6 and 7, we used the concept of planning for capabilities to predictively encode goals at planning time in one agent's plan that might be suspended by another agent. If the agents had clear knowledge of each other's information needs, such as the specific repairs they are working on, they could potentially make effective use of planning for capabilities. A situation where this would be beneficial is when the information servers charge for queries, and the rates are better for volume queries (e.g., the CARFAX service [CARFAX], which provides histories of used cars, charges one price for a single query, but only a few dollars more for unlimited queries). In this situation, the overall cost to the system could be reduced by the opportunistic aggregation of queries to certain information servers.

In the type of information system exemplified by our Aircraft Maintenance MAS, it appears unlikely, however, that a domain-independent approach to planning for capabilities would help increase the number of potential opportunities, for two key reasons. First, without domain-specific guidance, the sheer number of different possible queries would likely prevent an agent from correctly guessing which extra conditions would be helpful to some other agent—after all, an aircraft has a lot of parts to repair. Second, in the given MAS structure (Figure 8-4), the agents all have access to the same information providers. Thus if one agent cannot satisfy its information needs with the information agents in the system, it is unlikely that another agent will be able to obtain this information even if it did have the foresight to predictively encode the appropriate information goal. This would not be the case, however, if the agents making information queries had access to different information providing agents, or if they used different methods for generating information query plans. Similarly, in the case of an open MAS, if a new information-providing agent joins the system after some agent suspends a goal, the change in the makeup of the MAS may also be an opportunity to provide the needed information.

So if planning-time predictive encoding will not benefit our information agents, and natural opportunities are unlikely to occur, it is likely that the agents will have to make the best use of runtime opportunities. That is, if the agents are capable of re-planning at runtime, then a form of reactive plan repair (Section 6.2.3) could be used to modify an agent's plans when it learns about another agent's suspended goals. It is not unreasonable to expect that query-planning agents could perform runtime re-planning, since responding to information queries is essentially an interactive process. For example, if the MechanicAgent executing the plan shown in Figure 8-5 finds out some other agent is missing some manual pages for the repair plan it is trying to obtain, the MechanicAgent may be able to augment its query to the ManualAgent to include the additional pages (on the hope that they are now available). A final possibility would be for the agents to store their "recent" query plans on the expectation that information they have already obtained might be useful to some other agent in the system. This would be useful when an information providing agent becomes unavailable. Note that unlike our previous approach, this would not exploit opportunities based on an agent's future intentions, but rather on its current knowledge.

8.3 Summary

In this chapter we have shown how our POPG-based approach to multi-agent opportunism could be applied to other types of MASs, demonstrating that with some straightforward extensions this approach can be used in environments other than the one for which it was designed. Specifically, in the context of an oceanographic sampling network, we have discussed the changes that could be employed to utilize our approach when faced with a non-classical planning system, and when the agent's actions are coordinated with something other than middle agents (e.g., a management hierarchy). Similarly, in the context of an aircraft maintenance information system, we have described how query planners could exploit a POPG-based approach even when the resource needs of the plan actions are not available, and when planning for capabilities may not be feasible. These discussions show that the general applicability of our approach is indeed promising.

Chapter 9

Conclusion

In this chapter we briefly summarize our framework for multi-agent opportunism and discuss several key areas for future research. We conclude by arguing that our framework answers the thesis questions presented in Chapter 1.

9.1 Framework Summary

As we discussed in the introduction, multi-agent opportunism is the ability for agents to opportunistically assist one another to accomplish their goals. That is, agents must be able to *recognize* and *respond* to potential opportunities for the goals of other agents in the MAS.

In order to recognize a potential opportunity for another agent, an agent must have some knowledge about the other agents in the MAS. There are two ways agents can obtain this necessary information about each other: they can infer it, or the agents can explicitly share it. As inferring this information is likely to be too difficult in an open system of heterogeneous agents, we have focused our study on explicit information sharing. More specifically, we have determined that an agent designer must address these three key issues:

- For which task or goals should an agent request opportunistic assistance?

  The agents should limit this to just those tasks or goals (suspended or active) which other agents are likely to be able to help satisfy. Determining this requires the agent to undertake a cost/benefit decision process.

- Exactly what opportunity-related information should the agents explicitly share? The agents should share the information that will best enable the other agents to recognize environmental cues that would indicate a potential opportunity. This may include the goals themselves, cues the requesting agents has determined for itself, or a mixture of both.

- With which other agents should an agent share this opportunity-related information? An agent should share this information with the other agents that are most likely to have the capabilities to help satisfy the goal. Determining this will, in general, make use of the system's existing coordination mechanism. We have discussed how this could be done when the agents coordinate their activities using middle agents or through a simple hierarchy. Other coordination mechanisms are considered in Section 9.3.

For an agent to respond to a potential opportunity, it must be able to determine what an appropriate response might be, and when that response is even feasible. To provide agents with this ability, an agent designer must address these two issues:

- How can an agent decide if it should provide opportunistic assistance? An agent should consider its current intentions—the actions it is already planning to take—to determine if they may provide a potential opportunity. The agent should also use its intentions to consider the potential impact that providing opportunistic assistance may have on its own goals. As we are working with planning agents, this decision process will often be a form of plan analysis.

- How should an agent respond to a potential opportunity for another agent? The response may be to simply notify the other agent of the potential opportunity, or to take some action on the other agent's behalf. An agent may decide to take some action using only local information (e.g., following a standard operating procedure), or it may need to coordinate its actions with the other agent. The costs and benefits of this response should be included when deciding when to provide opportunistic assistance (described above).

Addressing the five key issues described above is critical to exploiting opportunities in a multi-agent system. However, just understanding these issues is not sufficient for knowing the feasibility or potential benefits of multi-agent opportunism in a real-world system. As such, we have also examined a specific approach to multi-agent opportunism for a particular class of MAS. This particular class includes systems of heterogeneous agents that use classical planning to determine their intentions, and that are coordinated using middle agents. By including this specific approach, we are able to demonstrate how our key issues can be addressed in a concrete setting. We are also able to provide an empirical analysis of the potential impact that exploiting multi-agent opportunism may have on a system's performance.

For our specific approach, we have introduced a simple yet flexible model for planning and execution based on the abstract plan representation *Partial Order Plan Graphs* (POPGs). POPGs provide explicit representations of plan preconditions, effects, and resource requirements, all of which are needed when reasoning about potential opportunities. This model separates the qualitative and quantitative parts of the planning problem, and is well suited for systems where online re-planning cannot or should not considered. The plan execution scheme embodied in this model transparently allows an agent to determine when it should pursue potential opportunities for the suspended goals of other agents in the system. It also allows the agents to determine when they should suspend goals they are unlikely to be able to satisfy, and thus when they should request opportunistic assistance.

In order for systems of planning agents to exploit potential opportunities for one another's goals, the agents' plans must contain actions that may lead to conditions favorable for producing potential opportunities. While these conditions may occur naturally, it would be more beneficial if these plans could be augmented to improve the chances of providing opportunistically favorable conditions. It is critical, however, that the plan enhancements do not interfere with an agent's course of action when pursuing opportunities for other agents is not needed.

To improve the chances of providing potential multi-agent opportunities, our specific approach uses the notion of *planning for capabilities* to augment each agent's plans. That is, the agents in our example system select additional goal conditions from their own capability sets to include in their plans, in the expectation that these goals might be suspended by some other agent in the system. The selection process used depends upon the information about the agents that is currently available. It may be simply the overlapping capabilities of the various agents, or it may be the actual goals that the have been assigned to the agents. Planning for capabilities is a type of *predictive encoding*, a passive approach to single-agent opportunism. Using such an approach allows the agent to pre-compute potential opportunity cues, relying on the normal runtime plan execution mechanism to recognize their presence and determine if pursuing them is appropriate in the given context.

Augmenting an agent's plan to include additional goals, however, may lead to inefficiencies with respect to the agent's assigned goals. As mentioned above, we need to avoid interfering with the agent's regular course of action when it does not need to pursue potential opportunities. As such, we have examined three techniques for plan enhancement that allow the agents to avoid performing unneeded actions. In particular, we have considered two post-planning methods of enriching the core structure of a plan: one that adds "shortcuts" bypassing the segments of the plan devoted strictly to support predictively encoded extra goals, and one that predictively repairs the core plan to include subplans achieving the extra goals. We have also examined an online approach that assumes the agents possess limited runtime plan repair capabilities. Using this approach, the agent attempts to enhance its core plan only at the time it learns of a goal suspended by another agent.

In order to quantify the potential impact of our specific approach to multi-agent opportunism, we have conducted an empirical evaluation of a simulated MAS that incorporates our ideas. Through this study we have shown that with limited shared information, and even with no re-planning or plan repair capabilities, systems of heterogeneous agents can indeed assist one another opportunistically in accomplishing their goals. When the agents cannot do online re-planning, and the degree of shared information is limited to just the overlapping capabilities of each agents, the system performance improvement ranged from ~2% to ~5%. Further, when the agents know what goals have been assigned to the other agents, the system performance improvement was even better, ranging from ~5% to ~8%. Thus we can conclude that multi-agent opportunism is feasible, and can indeed improve the overall performance of this class of multi-agent system.

9.2 Related Work

As we indicated in Section 3.3, to the best of our knowledge, there are no other research efforts that have studied or are currently studying multi-agent opportunism. In this section we review several research areas that are related to multi-agent opportunism in various way, and compare them to our approach.

Oliveira and Garrido [1995] have posited the idea of using *cognitive cooperation facilitators* (CCFs) to detect *cooperation opportunities* among agents with similar but complementary interests (i.e., goals). CCFs are similar to matchmakers [Klusch and Sycara, 2001], in that agents register their interests with them and they establish connections. Unlike matchmakers, CCFs are proactive in establishing connections between agents with similar interests. They also require the agents to register their interests (in the form of goals) instead of their capabilities.

While CCFs could be used to support multi-agent opportunism, the idea was never sufficiently developed to provide a complete model. That is, the primary role of CCFs was to identify potential opportunities among pairs (or perhaps small groups) of agents to cooperate on similar goals and notify those agents. The agents were then expected to negotiate among themselves how they could potentially help one another satisfy their goals. The authors did not specify how these negotiations could be accomplished, or even how the agents could determine if cooperation was, in fact, beneficial. That is, the CCF approach only addressed how to recognize potential opportunities, not how to respond to them.

Beyond the incompleteness of the model, there are two other important difficulties with this approach. First, the requirements for agents to register their goals, which may change frequently, would likely lead to significant communication overhead keeping the CCFs updated. Second, the determination of similar but complementary goals is not well specified, and is likely to be highly domain specific, requiring tailored CCFs.

However, while the use of specialized CCF agents may not be directly usable in a model of multi-agent opportunism, it provides some insights. For example, if an interest-matching mechanism could be specified, then a matchmaker or broker agent could be modified to infer interests from match requests and notify interested parties of potential collaboration opportunities. This capability could potentially enhance the opportunity recognition capabilities of the model described in Chapter 6.

Berennji and Vengerof [1999] use a *coordination mechanism in which the agents* exchange task information to achieve better system coherence. The experimental domain used is a modified Tileworld [Pollack and Ringuette, 1990] environment. The tasks in this system were referred to as opportunities because they would appear as "goal tiles" at random locations and disappear after a random amount of time. At each time step, the agents would negotiate among themselves to determine how each of the tasks should be allocated, and then each of the agents would move toward their task's location.

Although this approach is not multi-agent opportunism as we have defined it, it may be able to achieve some of the same high-level goals. For instance, such an approach would be able to improve the overall system performance in a dynamic environment by taking advantage of the opportunistic appearance of goals. Similarly, this approach specifically attempts to improve the coordination of the agents, going so far as to attempt to optimize their actions.

This approach does not, however, meet our key assumptions. Specifically, the agents are assumed to be homogeneous in that they are all capable of performing any task. In fact it goes further in assuming that all of the tasks are of the same type, differing only in their value. Similarly, this approach requires that all of the agents share all of their task-related information with one another. As we have discussed throughout this dissertation, clearly homogeneous agents with complete (or even nearly complete) shared knowledge can opportunistically assist one another with their goals. Our model shows that even with limited shared knowledge, systems of agents with heterogeneous capabilities can also do this.

In Section 4.2.1 we discussed how an agent could conceivably infer another agent's goals, plans, or even opportunity cues, to allow it to recognize potential opportunities for the other agent. This is as opposed to the explicit sharing of opportunity-related information that is used in our framework. Busetta et al. [2001] have developed an approach in which agents assist one another by "overhearing" the communication messages of other agents, and offering suggestions to improve an agent's plan, in the form of new information or additional actions. It is opportunistic in that the *service providing agents* are given information or commands from *suggester agents* that are not already available to them. The service providing agents are free to incorporate or ignore the suggestions.

To make this approach possible, the suggester agents must have a model of the service providing agents, as well as a representation of their mental attitude [Busetta et al., 2001]. The authors accomplish this having the service providing agents publicize their "public behavior model" which includes their current beliefs, goals, and plans. As we have discussed, it is unrealistic to expect that this degree of shared knowledge would be available in an open system of heterogeneous agents. As such, this approach is not a general model of multi-agent opportunism.

The area of *multi-agent plan merging*, which has been used by several researchers to improve coordination in an MAS, is also related to multi-agent opportunism. For instance, von Martial [1992] has created a taxonomy of multi-agent plan relations that identify both negative and positive interactions between two agent plans, along with a basic model of how to merge the plans to achieve coordinated actions. The negative relations identify actions within the plans that might lead to conflicts, such as an overlapping use of resources or the negation by one agent of the effects of another agent's actions. Negative relations were generally addressed by adjusting the ordering or timing of the affected actions. The positive relations identify redundant and subsuming actions in the plans, where one agent is already performing actions that produce effects needed in the other agent's plan. In cases like these, only one of the agents needs to perform the overlapping action(s).

An additional positive relation identified by von Martial is the *favor* relation, where one agent can satisfy the goal of another agent if that goal can easily be incorporated into its current plan. This relation is closest to our idea of multi-agent opportunism, as it can be used to identify situations where agents may be able to help one another in a more cost-effective way. In fact the reactive plan repair mechanism described in Section 6.2.3 essentially exploits a favor relation. In von Martial's model, however, the identification of favor relations in plans, as well as the means to incorporate additional goals into the current plan, requires domain-specific information. Further, the model does not address using this relation to help satisfy suspended goals of another agent, although it should be possible to do so.

Similarly, Durfee and Lesser [1987] developed the Partial Global Planning (PGP) approach to enable agents to plan coordinated actions. Decker and Lesser [1998] later extended PGP into the Generalized Partial Global Planning (GPGP) approach. Where PGP primarily addressed conflicts between plans, GPGP also included von Martial's positive plan relations. Both approaches provide a mechanism to allow agents to exchange their individual plans, identify parts of the plans that might require coordination, and propose plan modifications to achieve that coordination. Each agent would use the exchanged plan information to create a *partial global plan* to represent its own view of the global plan that encompassed the actions of other agents as well as its own.

PGP and GPGP were designed to support cooperative distributed problem solving (CDPS), where the agents collectively try to satisfy some common goals through their individual actions. This is in contrast to the MAS model that we have assumed, described in Section 5.2, that characterizes the system as a loosely-coupled collection of agents acting independently to satisfy their assigned goals. We have adopted this model because it makes few assumptions about the amount and type of knowledge the agents share. This allows us to make claims about the feasibility of multi-agent opportunism that are more general than if we assumed a CDPS model.

More recently, the work of Cox and Durfee [2003] has focused on finding *synergies* among multi-agent plans. Synergies are essentially von Martial's positive relations, with a particular emphasis on subsuming and overlapping effects (instead of just actions) that would lead to plan merges. Cox and Durfee's approach exploits hierarchical plans, where the plan is represented as a hierarchy of steps. Leaf nodes in the hierarchy are primitive actions, while interior nodes are abstract steps. The use of hierarchical plans allows this approach to find synergies at higher levels of abstraction than other plan merging mechanisms that focus on individual actions.

Finally, M. de Weerdt [2000] has developed a resource logic for multi-agent plan merging that allows agents to coordinate their plans by exchanging excess resources. In de Weerdt's formalism, resources are an abstraction that encompass both the traditional notion of resources as well as the classical planning notion of preconditions and effects. De Weerdt has also proposed an auction-based mechanism that would enable agents to find needed resources for the coordination of their plans. It is not clear how the agents identify what resources they are lacking, though.

Each of these approaches does, in its own way, improve the overall coordination of the MAS by eliminating conflicts or finding synergies among the agents' plans. However, none of them is adequate for providing a general model of multi-agent opportunism, for two key reasons. First, they each require the agents in the MAS to *commit* to changes in their plans. The very nature of opportunism implies that agents should assist one another as situations arise, not by formally committing to applying its resources to the satisfaction of another agent's goals. While our own plan-based approach (Chapter 6) provides multi-agent opportunism by incorporating extra goals into the agents' plans, the agents do not commit to these goals. Rather, the extra goals are pursued only they are determined to be needed and advantageous to the MAS.

Second, the plan merging approaches require the agents to exchange their plans, either with one another or with a central coordinator agent, so that they can be compared and merged. This requires the agents to share a great deal of information with one another, which we are explicitly trying to avoid. Further, for plan merging to take advantage of runtime opportunities, the exchange of plan-related information would have to occur any time an agent changes its intentions (e.g., when a goal is suspended). This would be a communication-intensive process. It would also be computationally expensive, as plan merging is exponential in the general case [Cox and Durfee, 2003], and thus doing it repeatedly would likely be intractable.

The area of *coalition formation* in MASs also has some bearing on multi-agent opportunism. Much of the research in this area has focused on finding optimal coalitions among self-interested agents, often using game theoretic approaches [Kraus, 1997]. In such systems, agents form coalitions only when it will increase their own payoff. The MAS must then be designed to provide incentives for agents to cooperate, leading to better overall system performance [Kraus, 1997; Shehory and Kraus, 1998].

While opportunism has generally been ignored in this area of research, it is not unrealistic to expect that given the appropriate incentive (e.g., expectation of a reward), self-interested agents would also be able to support multi-agent opportunism, using the mechanisms from our model. However, while it is less likely in a cooperative MAS, there is an increased risk that a self-interested agent would opportunistically take advantage of knowledge it obtains from another agent seeking assistance, a tactic the business world calls *poaching* [Clemons and Hitt, 2004].

Some of the work in this area, though, has looked into coalition formation in open systems of cooperative agents. For example, Shehory and Kraus [1998] have examined the selection of optimal coalitions in MASs coordinated using a matchmaker, where tasks arrive dynamically. This work is interesting since the approach is able to opportunistically reallocate tasks when new tasks arrive, by restructuring the coalitions when necessary. They do not, however, address any other mechanism for agents to help one another with their assigned tasks, although it is conceivable that a suspended task could be treated as a new task, leading to opportunistic reallocation.

Finally we consider agents organized using models of teamwork, which refer to the formal coordination models that assume a small group of agents are coordinating their activities in support of a set of shared goals. We discuss these models as a contrast to our approach. That is, it is very likely that a team of agents could support multi-agent opportunism, essentially because of the amount of knowledge in common they are generally presumed to possess. However, this assumption is not realistic for the open real-world MAS we are considering.

One of the earliest formalisms for team coordination is *joint intentions theory* [Cohen and Levesque, 1990; Jennings, 1995], in which agents agree to cooperate on a joint goal and on how that goal will be jointly satisfied (i.e., what role each agent will play in a common plan). In a similar method called *shared plans* [Grosz and Sidner, 1990], and later *collaborative plans* [Grosz and Kraus, 1996], the agents are not required to commit to explicit joint commitments, but do operate from an understanding of the common plan to be executed and their roles within it.

Tambe [1997] has taken these approaches a step further by combining and extending them into the *STEAM* formalism. This formal model of teamwork requires the agents to use both joint intentions and shared plans. The model also includes a protocol for establishing the intentions, selecting the plans, and establishing roles of each agent within the plans.

Given the amount of common knowledge the agents using one of these models must have, it should be straightforward to extend them to support multi-agent opportunism (although, to the best of our knowledge, this has not been tried). For instance, when using STEAM, one agent knowing the role that another agent is filling in the team is sufficient for the first agent to know the other agent's assigned tasks. Thus, opportunities can readily be recognized for other agents in the team, as well for shared team goals, and the appropriate agent notified. However, as mentioned above, teamwork is not a general model for multi-agent opportunism for the types of MAS we have assumed. This is because the agent teams must posses a great deal of shared information to operate, which cannot be relied upon in real-world systems of heterogeneous agents.

9.3 Future Work

In Chapter 8, we discussed in depth several areas of future research in the context of applying our POPG-based approach to multi-agent opportunism to other types of multi-agent systems. In this section we will address some other areas identified in this dissertation as targets for our future research efforts.

9.3.1 POPG Complexity

One drawback of our POPG-based plan execution scheme is the computational complexity of the action selection mechanism. Since this mechanism must consider essentially all possible total orderings of the remaining actions in the POPG to determine

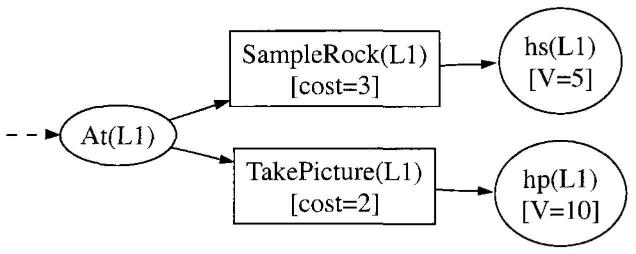

Figure 9-1: Partial POPG with costs and values.

the best course of action, its computational complexity is exponential in worst case. In our example Rovers MAS the plans tended to be somewhat "linear," in that there was usually only one or two possible actions to choose from at any given decision point. This fact tended to attenuate the complexity problem somewhat, making our experiments tractable on plans with 25-30 actions for up to 8 goals. Requiring several seconds to minutes to select the next action may be acceptable in domains like our Rovers example because the plans are at such a high level of abstraction. But in other domains stricter real-time constraints may make the use of our execution mechanism impractical. As such, we are particularly interested in reducing the computational complexity of our execution mechanism.

Interestingly, much of the computation done for action selection is redundant. Consider, for example, the partial POPG shown in Figure 9-1, which is a simplified version of the example shown in Figure 5-4. Assume for a moment that this is a subgraph of some larger POGP, and that it appears at the tail of the plan. As long as our resource $\rho \geq 5$, the evaluation of this section of this plan will determine that a value of 15 can be obtained. It will, however, have to consider both orderings of the actions `SampleRock(L1)` and `TakePicture(L1)` to determine this. Even worse, this subplan may have to be evaluated many times during the evaluation of its subsuming POPG, and in general as long as $\rho \geq 5$ the value will always be 15.

We would like to use a caching scheme, similar to the back-propagation mechanism used in [Dearden et al., 2002], that would reduce the redundancy in the computation. Specifically, we would like to associate with each node a caching function $\mathcal{F} : (\sigma, \rho) \rightarrow V$. That is, at each node, for a given level of resources ρ and in a given state σ, there will be a cached value V. In our above example, if we ignore the state for the moment (i.e., let $\sigma = any$), then the function $\mathcal{F}$ associated with the node `At(L1)` would be:

$$\mathcal{F}(\rho) = \begin{cases} 0, & \rho \leq 2 \\ 10, & 2 \leq \rho < 5 \\ 15, & \rho \geq 5 \end{cases}$$

It should be possible to build this cache function dynamically. That is, if during the normal action selection evaluation of a given node n, the current state and resource level is not represented in $\mathcal{F}$, then the evaluation continues as if caching was not being used. When a value is determine, $\mathcal{F}$ would be updated.

Although we did spend some time trying to implement a caching scheme in our simulated example MAS, the details of capturing and matching states and resources turned out to be a bit trickier than one would expect. Clearly one would not want to associate complete state descriptions, as they can be very large (leading to a trade-off between space and time). But we were unable to find a reasonable representation in the short time we spent on the problem. As reducing the time complexity of action selection could be a significant improvement to our approach, it is high on our list of future work.

9.3.2 Active Goals

Throughout this research project we have focused on the use of suspended goals as a way to determine when to request opportunistic assistance. This is not particularly surprising, considering suspended goals have often been a focus in research into opportunism (e.g., [Birnbaum and Collins, 1984; Francis, 1997; Hammond et al., 1993; Patalano et al., 1991; Simina and Kolodner, 1995; Pryor, 1996]). Further, they offer an easy indication of when an agent needs help—when it has a goal it *knows* it may not be able to satisfy itself.

As we mentioned in Section 4.1, an agent may also benefit from opportunistic assistance with its currently active goals. In that section we discussed an example from the AOSN domain in which the CONVEX-MOORING requests help from the other agents in detecting convective overturns. Such examples are likely to occur in many MASs. However, it is likely that identifying them will require domain-specific knowledge.

Our POPG-based approach to multi-agent opportunism provides two possibilities for a domain-independent method of identifying which active goals could benefit from opportunistic assistance. First, as was mentioned in Section 6.3, an agent could identify "high-risk" goals as those with an expected cost that is just below some threshold of the currently available resources. The expected cost of a particular goal can be computed as the sum of the expected costs of all of the actions in the current POPG that must be executed to satisfy the goal. Determining what a good threshold would be, or even if requesting opportunistic assistance for active goals would be beneficial, are issues that could readily be explored in our current simulated environment.

Similarly, an agent could request opportunistic assistance for an active goal based upon its value relative to the other assigned goals. For example, an agent may benefit from opportunistic assistance with unusually highly-valued goals, such as those with a value greater than the sum of all of the other assigned goals. While identifying such goals may be a simple matter, determining the types of relationships among the goals values that would benefit from requesting opportunistic assistance would clearly require further analysis. Our existing experimental environment would allow us to study this idea.

9.3.3 Cue Sharing

In the approach described in this dissertation, we have focused on using suspended goals for the opportunity-related information that the agents share with one another when requesting opportunistic assistance. This was a natural choice in our case, since the POPG-based planning and execution mechanism can readily be extended through planning for capabilities to prepare the agents for potential opportunities.

As mentioned in Sections 6.3 and 8.2, however, our POPG-based approach could also be used to support cue sharing as the means for determining opportunity-related information. As discussed in Section 6.3, the most direct way to use cue sharing would be if an unmet precondition of some agent's suspended goal is in the capability sets of other agents in the MAS. Borrowing the example from that section, suppose that the condition $c_1 =$ `have-rock-sample(WP7)` is a precondition for some other set of actions in the plan, $\mathcal{P}_i$, for agent $\mathbb{A}_i$, and is not itself an assigned goal. If $\mathbb{A}_i$ could identify that c_1 is necessary for the achievement of a suspended assigned goal, g_1, then requesting opportunistic assistance for c_1 would be cue sharing.

Note that we can still use our resource-based decision process to determine when to suspend a goal with this type of cue sharing. That is, when we suspend some goal g because of insufficient resources, we are also suspending the sequence of actions A from the agent's plan $\mathcal{P}$ that would lead to satisfying g. If c is a precondition of some action $a_i \in A$, it must also be in the effects of some other action $a_j \in \mathcal{P}$. The opportunistic satisfaction of c may mean that the agent will now not have to execute a_j, freeing up resources that could allow it to resume trying to satisfy g.

Similarly, as discussed in Section 8.2, we can also use cue sharing when a goal is suspended specifically because of some unmet precondition. This might be helpful when resources are not a limiting factor in plan execution, or simply when plan actions fail to produce their expected effects.

The difficulty for both of these situations is in determining what conditions should be used for opportunity cues. As discussed in Section 4.2.1, we wish the agent to select those conditions that are most likely to lead to the suspended goal's satisfaction that are also conditions that other agents can help satisfy.

For instance, consider again the above example where agent $\mathbb{A}_i$ has determined that condition c_1 is necessary for the achievement of suspended goal g_1. Unlike the above discussion, it is not actually necessary for c_1 to be in the capability set of some other agent in the MAS. Rather, c_1 only has to be an effect of some action in another agent's plan to potentially be opportunistically satisfied. But, if it is not part of any known capability sets, it would be difficult for $\mathbb{A}_i$ to determine which, if any, other agent could help satisfy c_1.

In the information systems case the problem is a bit easier, since an unmet condition is a description of information the agent does not possess. If that condition is critical to satisfying the goal, then the condition is a good candidate for opportunistic assistance from other agents. Further, the infrastructure of a distributed information system must necessarily provide a way to locate agents that can provide given types of information. Thus if an agent can find another source for the information needs of an unmet condition, it should use it as an opportunity cue.

In general, however, identifying conditions that would make good opportunity cues in multi-agent opportunism is likely to be more complicated. It is possible that a form of diagnosis may be needed to determine exactly what conditions are impeding progress towards a particular goal, with further reasoning being required to determine if the other agents in the MAS can help satisfy it. Since finding a domain-independent mechanism to support cue sharing would greatly enhance our framework for multi-agent opportunism, it is an area we intend to study further.

9.3.4 Other Coordination Mechanisms

In our POPG-based approach to multi-agent opportunism, the MAS coordination mechanism is used for the discovery and distribution of information. In terms of discovery, the coordination mechanism is used to determine the capabilities of other agents in the MAS, and when possible, the goals that have been assigned to them. This information is used to decide which extra capabilities should be predictively encoded in each agent's plans. As for distribution, the coordination mechanism is used to determine which agents are capable of providing opportunistic assistance for suspended goals. In our specific approach, we chose to use a middle agent to coordinate the activities of the other agents. The experiments described in Chapter 7 indicate that this mechanism does allow the agents to obtain sufficient information about one another to make multi-agent opportunism feasible.

We would like to extend the experiments described in Chapter 7 by considering the performance impact of multi-agent opportunism when compared to using the middle agent to re-allocate suspend goals to some other agent in the MAS. The middle agent is in a good position to select another agent for the suspended goal, since it knows about each agent's capabilities, and in principle has some idea of each agent's current work load. We do, however, still expect multi-agent opportunism to result in better performance than this approach, for two key reasons.

First, if we maintain our assumption that the agents are not capable of runtime re-planning (otherwise this would just be reactive plan repair), then the broker would have to include the suspended goal in the next cycle of goal assignments. This would mean that either some agent would be assigned more than the expected k goals, or that some other goal that would otherwise have been assigned during that cycle would itself be delayed. If the choice is to assign more than k goals to some agent $\mathbb{A}$, then because the agents are already resource bound, it is likely that $\mathbb{A}$ will again not be able to satisfy all of its assigned goals, leading to more goals being suspended. Unless the broker is able to balance the load of the extra goal assignments, this could lead to a cascade of suspended goals, causing the system's performance to degrade. If instead the middle agent delays some goal in the current cycle, even if it prioritizes the goals by value, there is a possibility of starvation of some goals, meaning that no attempt at all will be made to satisfy them (even opportunistically).

Our second reason for expecting poorer performance when using the middle agent to reallocate a suspended goal to another agent is that it will limit the potential for opportunistic satisfaction. That is, the middle agent will select a single other agent to attempt to satisfy the suspended goal. By not informing other agents of the suspended goal, opportunities for satisfying that goal in a more cost effective way may be missed.

We would also like to examine multi-agent opportunism in the context of other MAS coordination mechanisms. We would expect that each coordination mechanism would have its own strengths and weakness in terms of supporting opportunism, even going beyond information the discovery and distribution. It is exactly these issues that we would like to explore.

For instance, the contract-net protocol (CNP) [Smith, 1980] is inherently opportunistic in the way goals can be dynamically re-allocated at runtime. That is, suppose some agent, $\mathbb{A}_1$, determines that it must suspend one of its goals, g, that it had contracted to satisfy for some value, V_g. Using the normal CNP mechanisms, $\mathbb{A}_1$ can solicit bids from the other agents to determine if any of them could satisfy g. If some other agent, $\mathbb{A}_2$, provides a bid to satisfy g at a cost less than V_g, then g can be opportunistically satisfied by $\mathbb{A}_2$.

CNP alone, however, does not provide a complete model of multi-agent opportunism. As we can see above, it does provide a mechanism for agents to determine which other agents in the MAS to share their opportunity-related information with. Similarly, the decision process that agents use to compute their bids for a given goal does allow them to determine if they should attempt to provide opportunistic assistance. But CNP does not specify how the agents compute their bids, and so we do not know if the agents are basing their decisions on their current intentions. If we assume that they use planning to make this decision, then it is reasonable to expect that they could try to opportunistically incorporate goals suspended by other agents into their current intentions, perhaps using a method like reactive plan repair described in Section 6.2.3. This would require the agents to be capable of runtime re-planning. While this does violate one of our key assumptions, it is not unreasonable in the context of CNP.

But even if the agents do compute bids for new goals based on their current intentions, without modification CNP does not allow the agents to attempt to incorporate a goal into their plans for *possible* opportunistic satisfaction. That is, just as with the plan-merging and coalition formation mechanisms described in Section 6.2.3, CNP requires agents to commit to satisfying their goals. As such, the agents do not, in general, have the flexibility to provide assistance on a truly opportunistic basis—*if and when* conditions permit. Rather, they agree to commit their resources and actions to satisfying the given goal. Further, it is possible that agents with losing bids may encounter opportunities to satisfy a suspended goal. But, since only the single agent with the winning bid commits to a given goal, these other potential opportunities would be missed.

Also, CNP alone does not provide a way for the agents to determine when (for which goals) they should request opportunistic assistance, nor does it provide any method for the agents to determine exactly what opportunity-related information they should share. If, as above, we assume we are working with a system of planning agents, then just as with our POPG-based approach the agents could use their plans to make these decisions. If we further assume that the agents use their plans to request opportunistic assistance for their suspended goals, then CNP does provide a way for the agents to determine when it would be cost effective to make that request. As described in the example above, if through the bidding process an agent determines that one or more other agents can possibly satisfy its suspended goal, g, for a cost less than the goal's value, V_g, then the agent should make the request for assistance.

In addition to analytically considering other MAS coordination mechanisms, we would also like to empirically examine multi-agent opportunism in the context of these other mechanisms. For example, in our existing experimental environment we could examine agents that use CNP for coordination, as described above, or a hierarchy as discussed in Section 8.1.1 For a hierarchy, unlike the approach described in Section 8.1.1, initially we would use a classical planner to keep the changes to the agents manageable.

Of particular interest, though, would be to consider the impact of multi-agent opportunism on agents coordinated using models of teamwork (e.g., [Tambe, 1997; Grosz and Kraus, 1996; Cohen and Levesque, 1990]). Our belief is that since the agents in these models already possess a great deal of shared information, they could more easily exploit multi-agent opportunities. If fact it may even be feasible for teams of agents to infer potential opportunities for one another. The focus of the research described in this dissertation has been on finding the low-end of shared information that still leads to performance improvements due to multi-agent opportunism. An examination of teamwork models could provide us with a better idea of the high-end, since the agents already share so much information.

9.4 Final Remarks

In this dissertation we have presented a framework for multi-agent opportunism that is applicable to open systems of heterogeneous planning agents. Our primary thesis question for this framework was simply: *Can multi-agent opportunism be effective in systems of heterogeneous agents with little or no shared knowledge?* That is, can open, real-world multi-agent systems benefit from being able to recognize and respond to opportunities for each other's goals? Or will the lack of shared information be too restrictive to allow cost-effective opportunistic assistance?

The analysis of our framework described in Chapter 7 specifically addresses this question. In that analysis, we examined the impact of multi-agent opportunism on a specific, but broadly representative, class of multi-agent systems. Our results indicate that even when the agents know only about the capabilities of other agents in the system, and even when no runtime re-planning is possible, multi-agent opportunism is indeed effective in improving the overall system performance. Further, when the agents are better informed with knowledge about what goals have been assigned to the other agents in the system, multi-agent opportunism is able to produce even better improvements in the system's performance. Alternatively, our results also indicate that, unless the agents are capable of runtime re-planning, at least some *a priori* shared information is needed to make multi-agent opportunism effective.

A second, related question that we have explored is: *Are agents that are capable of exploiting opportunities for themselves also capable, using the same mechanisms, of recognizing and responding to opportunities for other agents, given that they have adequate knowledge of each other?*

If one were to only consider the abstract part of our framework (Chapter 4), the answer to this question would be trivially yes. That is, if we assume that an agent is capable of recognizing and responding to potential opportunities for its own goals, it is not difficult to expect that they could also recognize and respond to potential opportunities for the goals of other agents *given adequate knowledge of these goals.* But through our specific POPG-based approach, we have gone a step further in that we have demonstrated in a concrete setting that agents can indeed use the same mechanism to recognize and respond to potential opportunities independent of whether they are for another agent's goal or one of their own. Through our specific approach we are also able to describe what is meant by "adequate knowledge" for a particular class of MAS. Further, and perhaps more interesting, we have shown that under some circumstances, multi-agent opportunism can be effective even when the agents are not capable of single-agent opportunism.

Finally, the contributions of our framework for multi-agent opportunism are both theoretical and practical. On the theoretical side, our framework provides an analysis of the critical issues that must be addressed in order to successfully exploit opportunities in a multi-agent system. These issues involve the type and degree of knowledge the agents must share, as well as the decision-making capabilities they must possess. This analysis provides MAS designers and developers important guidance to incorporate multi-agent opportunism into their own systems. It also provides the fundamental underpinnings of our own specific approach to multi-agent opportunism.

On the practical side, we have developed, implemented, and evaluated a specific approach to multi-agent opportunism for a particular class of multi-agent system. Our approach enables systems of planning agents coordinated through a middle agent to opportunistically assist one another achieve their goals, thus improving the overall system performance. We have achieved this by combining a particular form of single-agent opportunism known as predictive encoding with an approximate decision-theoretic planning mechanism that allows the agents to make critical information-sharing decisions. The planning mechanism uses an abstract plan representation known as Partial Order Plan Graphs (POPGs) that we developed specifically to support multi-agent opportunism. While the POPG representation was designed for this research project, it is still general enough to represent the features of most (if not all) techniques used in the area of classical (i.e., STRIPS-based) AI domain-independent planning.

We believe the contributions made by this approach are important for two key reasons. First, it should be applicable to many existing MASs. This is because we have leveraged well-understood technologies from both AI planning and single-agent opportunism, and also because middle agents are widely used in coordinating multi-agent systems. Second, our approach demonstrates that MASs can benefit from multi-agent opportunism even when the agents share little or no common knowledge of such things as plans, goals, and capabilities. This is because we have started from an assumption that the agents will be heterogeneous, implying that they would possess little or no shared knowledge. Our results thus provide a baseline, demonstrating that performance improvements can still be achieved even with this restrictive assumption.

Bibliography

José Luis Ambite and Craig A. Knoblock. Flexible and scalable cost-based query planning in mediators: A transformational approach. *Artificial Intelligence*, 118: 115–161, 2000.

José Luis Ambite and Craig A. Knoblock. Planning by Rewriting. *Journal of Artificial Intelligence Research*, 15:207–261, 2001.

DAML Services Coalition: A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, and H. Zeng. DAML-S: Semantic Markup for Web Services. In *Proceedings of the International Semantic Web Working Symposium (SWWS)*, July 30-August 1 2001.

H.R. Berenji and D. Vengerov. Cooperation and coordination between fuzzy reinforcement learning agents in continuous state partially observable Markov decision processes. In *Proceedings of the IEEE Fuzzy Systems Conference*, pages 621–627, Aug 1999.

Lawrence Birnbaum and Gregg Collins. Opportunistic Planning and Freudian Slips. In *Proceedings of the Sixth Conference of the Cognitive Science Society*, 1984.

D. Richard Blidberg and Steven G. Chappell. Guidance and control architecture for the EAVE vehicle. *IEEE Journal of Oceanic Engineering*, OE-11(4):449–461, 1986.

A. Blum and M. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90:281–300, 1997.

Craig Boutilier. Sequential Optimality and Coordination in Multiagent Systems. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 478–485, Stockholm, 1999.

F. Bub, W. Brown, P. Mupparapu, K. Jacobs, and B. Rogers. Hydrographics survey report: Convective overturn experiment (CONVEX): R/V endeavor cruise en-291. Technical report, University of New Hampshire Ocean Process Analysis Laboratory, 1997. (also at http://ekman.sr.unh.edu/OPAL/CONVEX/EN291/en291-report.html).

Paolo Busetta, Luciano Serafini, Dhirendra Singh, and Floriano Zini. Extending Multi-agent Cooperation by Overhearing. In *CooplS '01: Proceedings of the 9th International Conference on Cooperative Information Systems*, pages 40–52. Springer-Verlag, 2001. ISBN 3-540-42524-1.

T. Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1-2):165–204, 1994.

CARFAX. Carfax vehicle history report service. http://www.carfax.com/.

S.G. Chappell, R.M. Turner, E.H. Turner, and C.M. Grunden. Cooperative behavior in an autonomous oceanographic sampling network: MAUV Project update. In *Proceedings of the 10th International Symposium on Unmanned Untethered Submersible Technology (UUST)*, pages 375–384, September 1997.

Eric K. Clemons and Lorin M. Hitt. Poaching and the Misappropriation of Information: An Analysis of Relationship Risks in Information-Intensive Production. In *Proceedings of the 37th Hawaii International Conference on System Sciences (HICSS-04)*, January 5-8 2004.

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

Jeffrey S. Cox and Edmund H. Durfee. Discovering and Exploiting Synergy Between Hierarchical Planning Agents. In *Proceedings of Second International Joint Conference On Autonomous Agents and Multiagent Systems (AAMAS '03)*, Melbourne, Australia, July 2003.

T. Curtin, J. Bellingham, J. Catipovic, and D. Webb. Autonomous oceanographic sampling networks. *Oceanography*, 6(3), 1993.

Mathijs de Weerdt, Andre Bos, Hans Tonino, and Cees Witteveen. Multi-agent Cooperation in a Planning Framework. In *Proceedings of the Twelfth Belgium-Netherlands Artificial Intelligence Conference (BNAIC-00)*, pages 53–60, 2000.

R. Dearden, N. Meuleau, S. Ramakrishnan, D. Smith, and R. Washington. Contingency Planning for Planetary Rovers. In *3rd Int. NASA Workshop on Planning & Scheduling for Space*, Houston, 2002.

K. Decker and V. Lesser. Designing a Family of Coordination Algorithms. In M. Huhns and M. Singh, editors, *Readings in Agents*, pages 450–457. Morgan Kaufmann, 1998.

Carmel Domshlak and James Lawton. On Planning for Multi-Agent Opportunistic Execution. In *Proceedings of the IJCAI-03 Workshop on Issues in Designing Physical Agents for Dynamic Real-Time Environments*, Acapulco, Mexico, 2003.

Carmel Domshlak and James Lawton. Opportunistic Planning and Plan Execution. In *Proceedings of the European Conference on Artificial Intelligence (ECAI-04) Workshop on Agents in Dynamic Real-Time Environments*, Valencia, Spain, August 2004.

Edmund Durfee and Victor Lesser. Using partial global plans to coordinate distributed prolem solvers. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 875–883, 1987.

Kutluhan Erol, James Hendler, and Dana S. Nau. HTN Planning: Complexity and Expressivity. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, volume 2, pages 1123–1128, Seattle, Washington, USA, 1994. AAAI Press/MIT Press. ISBN 0-262-51078-2.

Tara Estlin and Daniel Gaines. Distributed Rovers Project, Planning and Scheduling Artificial Intelligence Group, JPL. http://www-aig.jpl.nasa.gov/public/planning/dist-rovers/, 2002.

Mark Fasciano. Real-Time Case-Based Reasoning in a Complex World. Technical Report TR-96-05, University of Chicago, 1996.

R. Fikes and N. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Articial Intelligence*, 2(3/4), 1971.

FIPA. FIPA Abstract Architecture Specification. Technical report, Foundation for Intelligent Physical Agents, 2002. http://www.fipa.org/specs/fipa00001/SC00001L.html.

M. Fox and D. Long. The Third International Planning Competition: Temporal and Metric Planning. In *Proc. of AIPS-02*, pages 333–335, 2002. URL `http://www.dur.ac.uk/d.p.long/competition.html`.

Anthony Francis. *Memory-Based Opportunistic Reasoning (Thesis Proposal).* PhD thesis, Georgia Tech, 1997. (ftp://ftp.cc.gatech.edu/pub/ai/students/centaur/proposal.ps.Z).

Claudia Goldman and Schlomo Zilberstein. Optimizing Information Exchange in Cooperative Multi-Agent Systems. In *Proceedings of the Second International Joint Conference on Autonomous Agent and Multi-Agent Systesm (AAMAS-03)*, Melbourne, Australia, July 2003.

B. Grosz and S. Kraus. Collaborative Plans for Complex Group Action. *Artificial Intelligence*, 86(2):269–357, 1996.

B. Grosz and C. Sidner. Plans for Discourse. In P. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*. MIT Press, 1990.

Kristian Hammond, Mark Fasciano, Daniel Fu, and Timothy Converse. Actualized Intelligence: Case-based Agency in Practice. Technical Report TR-96-06, University of Chicago, 1996.

Kristian J. Hammond, Timothy Converse, and Mitchell Marks. Opportunism and Leaning. *The Journal of Machine Learning*, 10(3), March 1993.

Barbara Hayes-Roth and Fredrick Hayes-Roth. A Cognitive Model of Planning. *Cognitive Science*, 3:275–310, 1979.

C. Hewitt. Open Information Systems Semantics for Distributed Artificial Intelligence. *Artificial Intelligence*, 47(1-3):79–106, 1991.

Jörg Hoffmann and Bernhard Nebel. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

Nicholas R. Jennings and Michael J. Wooldridge. Applications of Intelligent Agents. In Nicholas R. Jennings and Michael J. Wooldridge, editors, *Agent Technology Foundations, Applications and Markets*. Springer-Verlag, 1998.

N.R. Jennings. Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems using Joint Intentions. *Artificial Intelligence*, 75:195–240, 1995.

Matthias Klusch and Katia Sycara. Brokering and Matchmaking for Coordination of Agent Societies: A Survey. In A. Omicini et al., editor, *In Coordination of Internet Agents*. Springer-Verlag, 2001.

Craig A. Knoblock. Building a planner for information gathering: A report from the trenches. In *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems*, Edinburgh, Scotland, 1996.

Sarit Kraus. Negotiation and cooperation in multi-agent systems. *Artificial Intelligence*, 94:79–97, 1997.

James Lawton. Opportunism in Planning Systems: A Critical Review. http://cdps.umcs.maine.edu/ jhlawton/DepthExam/lit-survey-final.ps.gz, 1999.

James Lawton. Opportunism in Planning Agents. In *Proceedings of the Seventh World Multiconference on Systemics, Cybernetics and Informatics (SCI-03)*, Orlando, Florida, July 2003.

James Lawton and Carmel Domshlak. Towards Multi-Agent Opportunism with Planning Agents. In *Proceedings of the AAMAS-03 Workshop on Autonomy, Delegation, and Control*, Melbourne, 2003.

James Lawton and Carmel Domshlak. Multi-Agent Opportunistic Planning and Plan Execution. In *Proceedings of The 16th IEEE International Conference on Tools with Artificial Intelligence*, Boca Raton, Florida, November 2004a.

James Lawton and Carmel Domshlak. On the Role of Knowledge in Multi-Agent Opportunism. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems*, New York, 2004b.

James Lawton, Roy M. Turner, and Elise H. Turner. A Unified Long-Term Memory System. In *Proceedings of the Third International Conference on Case-Based Reasoning (ICCBR-99)*, 1999.

Linda Wills and Janet Kolodner. Towards more creative case-based design systems. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, 1994.

Amnon Lotem and Dana S. Nau. New Advances in GraphHTN: Identifying Independent Subproblems in Large HTN Domains. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS-00)*, Breckenridge, April 2000.

D. McAllester and D. Rosenblitt. Systematic nonlinear planning. In *Proc. of AAAI-91*, pages 634–639, 1991.

Hyacinth S. Nwana and Divine T. Ndumu. A Perspective on Software Agents Research. *The Knowledge Engineering Review*, 14(2):1–18, 1999.

Elth Ogston and Stamatis Vassiliadis. Matchmaking Among Minimal Agents Without a Facilitator. In *Proceedings of the Fifth International Conference on Autonomous Agents (Agents-01)*, pages 608–615, Montreal, Quebec, Canada, 2001.

E. Oliveira and P. Garrido. Cognitive Cooperation Facilitators. In *Proceedings of the 1995 IEEE International Conference on Intelligent Systems for the 21st Century*, pages 3806–3809, Oct 1995.

A. Patalano, C. Seifert, and K. Hammond. Predictive Encodings: Planning for Opportunities. In *Proc. of the 15th Conf. of the Cognitive Science Society*, pages 800–805, 1993.

Andrea Patalano, Colleen Seifert, and Kristian Hammond. Predictive Encodings: Planning for Opportunities. In *Proceedings of the Fifteenth Conference of the Cognitive Science Society*, 1991.

M. Pollack and M. Ringuette. Introducing the Tileworld: Experimentally Evaluating Agent Architectures. In *Proceedings of the Eigth National Conference on Artificial Intelligence (AAAI-90)*, 1990.

L. Pryor. Opportunity recognition in complex environments. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 1147–1152, 1996.

L. Pryor and G Collins. Reference features as guides to reasoning about opportunities. In *Proceedings of the Fourteenth Conference of the Cognitive Science Society*, 1992.

Ashwin Ram and Lawrence Hunter. The Use of Explicit Goals for Knowledge to Guide Inference and Learning. *Journal of Applied Intelligence*, 2(1):47–73, 1992.

E. Sacerdoti. *A Structure for Plans and Behavior.* Elsevier, North-Holland, New York, 1977.

R. Schank and R. Abelson. *Scripts, Plans, Goals, and Understanding.* Lawrence Erlbaum, Hillsdale, NJ, 1977.

O. Shehory and S. Kraus. Formation of Overlapping Coalitions for Precedence-ordered Task Execution among Autonomous Agents. In *Proc. of ICMAS-96*, pages 330–337, 1996.

Onn Shehory and Sarit Kraus. Methods for Task Allocation Via Agent Coalition Formation. *Artificial Intelligence*, 101(1-2):165–200, May 1998.

Onn Shehory, Gita Sukthankar, and Katia Sycara. Agent Aided Aircraft Maintenance. In *Proceedings of the Third International Conference on Autonomous Agents (Agents-99)*, pages 306–312, Seattle, Washington, 1999.

M. Simina, J. Kolodner, A. Ram, and M. Gorman. Invention as an Opportunistic Enterprise. In *Proceedings of The Nineteenth Conference of the Cognitive Science Society*, 1997.

M. Simina, J. Kolodner, A. Ram, and M. Gorman. Opportunistic Enterprises in Invention. In *Proceedings of The Twentieth Conference of the Cognitive Science Society*, 1998.

Marin D. Simina and Janet L. Kolodner. Opportunistic Reasoning: A Design Perspective. In *Proceedings of the Seventeenth Conference of the Cognitive Science Society*, 1995.

D. Smith. Special Issue on the 3rd International Planning Competition. *Journal of Artificial Intelligence Research*, 20, 2003.

R.G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, 29(12):1104–1113, 1980.

Katia Sycara, Jianguo Lu, Matthias Klusch, and Seth Widoff. Matchmaking Among Heterogeneous Agents on the Internet. *Journal of ACM SIGMOD Record: Special Issue on Semantic Interoperability in Global Information Systems (A.Ouksel and A. Sheth, eds.)*, 1999.

Milind Tambe. Towards Flexible Teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

Elise H. Turner and Roy M. Turner. A constraint-based approach to assigning system components to tasks. *International Journal of Applied Intelligence*, 10(2/3):155–172, 1999.

Roy M. Turner. *Adaptive Reasoning for Real-World Problems: A Schema-Based Approach.* Lawrence Earlbaum Associates, 1994.

Roy M. Turner and Elise H. Turner. Organization and reorganization of autonomous oceanographic sampling networks. In *Proceedings of the 1998 IEEE International Conference on Robotics and Automation (ICRA-98)*, pages 2060–2067, Leuven, Belgium, May 1998.

Roy M. Turner and Elise H. Turner. A two-level, protocol-based approach to controlling autonomous oceanographic sampling networks. *IEEE Journal of Oceanic Engineering*, 26(4):654–666, October 2001.

Roy M. Turner, Elise H. Turner, and D.R. Blidberg. Organization/Reorganization of AOSNs: The AUSI/UMaine MAUV Projects - Status and Plans. 1997. presented to the Office of Naval Research NOPP Grants Kickofff and Coordination Meeting.

F. von Martial. *Coordinating Plans of Autonomous Agents.* LNAI 610. Springer-Verlag New York, Inc., 1992.

D. S. Weld. Recent Advances in AI Planning. *AI Magazine*, 20(2):93–123, 1999.

Gerhard Wickler. *CDL: The Capability Description Language.* PhD thesis, Edinburgh University, 1998. http://www.aiai.ed.ac.uk/ oplan/cdl/.

Linda Wills and Janet Kolodner. Explaining Serendipitous Recognition in Design. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, Atlanta, GA, 1994.

Hao-Chi Wong and Katia Sycara. A Taxonomy of Middle-Agents for the Internet. In *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMSA-00)*, Boston, MA, 2000.

Q. Yang, D.S. Nau, and J. Hendler. Merging Separately Generated Plans with Restricted Interactions. *Computational Intelligence*, 8(2):648–676, 1992.