INTEGRATED RESOURCE ALLOCATION AND PLANNING IN STOCHASTIC MULTIAGENT ENVIRONMENTS

by

Dmitri A. Dolgov

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2006

To my parents and Anya.

ACKNOWLEDGEMENTS

First and foremost, I owe a lot of gratitude to my advisor, Ed Durfee. Without his support, patience, academic guidance, and technical advice, this dissertation would not have been possible. I especially want to thank Ed for giving me the freedom to explore and supporting me when I got excited about problems, even at times when he didn't quite share my enthusiasm. I truly believe Ed's top work priority is the growth of his students, and for that I offer my deepest thanks!

I'm very grateful to the members of my committee, Kang Shin, Satinder Singh, Demos Teneketsis, and Michael Wellman for their valuable advice and insightful comments on my work. Michael Wellman deserves special credit for doing a very thorough job reviewing my thesis.

I'm very thankful to my teachers at the University of Michigan, especially professors Martha Pollack and Kevin Compton.

The administrators of the Michigan AI Lab deserve many thanks for their help in navigating various financial and administrative jungles. The support of Kelly Cormier, Colleen Neilson, and Wendy Anderson was truly indispensable.

My friends in the AI Lab contributed significantly to making my life there fun, productive, and sometimes both. Thanks to all of you, and especially Jeff C., Joel W., Mike J., Thom B., Jianhui W., Stefan W., Chris K., Matt R., Bart P., Julie W., Vishal S., Dave W., Josh E., Kevin L., and Dan R.!

Among my non-AI friends, I thank Tami & Mike U., Mike S., Kat & Dave P., Brooke H., Bill R., Ken L., Khachik S., and Lisa & Derek C. for making my life in Ann Arbor much more enjoyable. ABC deserves a mention for continued and consistent support. Also, to all the people with whom I played soccer and went skiing, thanks. I would have long withered without those games and trips!

To my parents I owe so much more than can be expressed on paper or in words. Mom, Dad, thanks for everything!

Finally, I thank my fiance, friend, colleague, and editor, Anya, for her love and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Resource allocation is a ubiquitous problem that arises whenever scarce resources have to be distributed among multiple autonomous entities (e.g., people, companies, robots). Stochastic planning is also a very common problem that focuses on developing models and algorithms for behaving optimally in uncertain environments. The motivation for this dissertation is that the problems of resource allocation and stochastic planning are often very strongly intertwined: the utility for acquiring some resources is commonly determined by what goals can be achieved using these resources, while devising the best course of action for achieving these goals can involve solving a complex stochastic planning problem. An integrated approach to modeling and solving the problems of resource allocation and stochastic planning allows us to exploit problem structure that would otherwise be lost if the problems were considered separately.

The overarching goal of this dissertation is to develop computationally efficient mechanisms for allocating consumable and non-consumable resources among agents whose preferences for these resources are induced by stochastic planning problems. Towards this end, we develop new models of planning problems, based on the framework of Markov decision processes (MDPs), where the action sets are explicitly parameterized by the available resources. Given these models, we design algorithms based on linear and integer programming that simultaneously solve for optimal allocations of resources and strategies for acting in the stochastic environments. These algorithms then form the core of our mechanisms for allocating resources in cooperative as well as competitive multiagent settings. We show analytically and empirically that the integrated approach leads to drastic (in many cases, exponential) improvements in computational efficiency over methods that consider the problems separately.

To complement the above, we develop mechanisms that, in addition to exploiting structure in agents' preferences arising from regularities in the underlying planning problems, also exploit structure within the agents' MDPs. By utilizing and extending techniques based on approximate linear programming, we adapt our resource-allocation mechanisms to well-structured planning problems, represented as factored MDPs. This leads to algorithms that scale up to even larger resource-allocation problems, where the agents' preferences are induced by MDPs with extremely large state spaces.

CHAPTER 1

Introduction

The problem of resource allocation among multiple autonomous entities (agents) is ubiquitous in the modern world. A business manager has to distribute a limited budget among the departments of the company. A computing center faces the problem of allocating its limited resources among computational tasks. A government is in charge of distributing the scarce resources of its country (e.g., wireless spectrum) among its people and businesses. Making the right allocation decisions in these and other similar scenarios can be of critical importance.

However, what does it mean to allocate the resources in the "right" way? A good manager wants to maximize the long-term profitability of her business. A process that allocates resources to tasks in a computing environment should aim to get as much useful work done as possible. The goal of a professionally run and honest government is to do what is best for its people.

In all the above, the goal of the resource-allocation process is to maximize a measure of global utility that can be obtained by the agents in the system by using these resources. This, in turn, raises the question of what determines the value of a particular set of resources to an agent. Resources are used by the agents to pursue their goals and to obtain rewards on achieving the latter. However, in order to be able to achieve those goals, the agents often need to solve a nontrivial planning problem. For example, when making decisions regarding budget allocation, a manager needs to consider how the money will be used and what will be accomplished as a result. In a computing center, it is important to understand what sequence of steps is required to complete a task before resources can be allocated to it. Similarly, before a government can make an informed decision about how to allocate the wireless spectrum in the best possible way, someone has to evaluate the possible uses of the frequencies and the resulting benefits.

Further, in any realistic domain, such a planning process is complicated by the fact that an agent faces multiple interdependent objectives, whose achievement requires executing sequences of actions whose outcomes are uncertain. For instance, profitability of a business is subject to many external factors (demand, competition, legislation) that can seldom be predicted with certainty.

The focus of this dissertation is on the interconnected problems of resource allocation and sequential decision making under uncertainty about the dynamics of the environments the agents operate in. When viewed primarily from the resource allocation perspective, this work can be characterized as a study of algorithms for resource allocation where the values of the resources to the agents are defined by the agents' stochastic planning problems. Alternatively, if the planning problem is placed at the forefront, the work can be described as a study of multiagent planning under uncertainty, where the interactions between agents are defined by the resources that are being distributed among them. We focus on problems where the allocation of resources is done in a single step: the resources are distributed among the agents, and no re-allocation of resources is allowed during the plan-execution phase. This dissertation describes a formal framework of such resource-allocation and stochastic planning-problems, analyzes their properties, and develops computationally tractable solution algorithms.

1.1 Resource Allocation and Stochastic Planning

The problem of resource allocation among multiple agents arises in countless domains and is studied in many diverse research fields such as economics, operations research, and computer science. The main focus of the work done in the area of resource allocation is on developing mechanisms that distribute the resources among the agents in desirable ways, given the agents' preferences over sets of resources. In such problems, the characteristics of the agents' utility functions often have a significant bearing on the properties of the resource-allocation problem. However, although defining classes of utility functions that lead to well-behaved resource-allocation problems is a topic that has received a lot of attention, most work stays agnostic about eh underlying processes that define the agents' preferences for resources.

Stochastic planning, or sequential decision making under uncertainty, is also a very widely studied problem that has found application in many diverse areas. As a result, several formal mathematical frameworks (e.g., Markov decision processes) have emerged as popular tools for studying such problems. However, for the most part, such models do not have an explicit notion of resources and do not explicitly address the problem of planning under resource constraints.

The fundamental insight of the work in this dissertation is that these two classes of problems are strongly intertwined in ways that make analyzing and solving them in concert very beneficial. The motivation behind this work is that many real-world domains have both resource-allocation and stochastic-planning components to them, and the main hypothesis of this thesis is that by integrating these two problems and studying them in tandem, we can fruitfully exploit structure that is lost if the problems are considered in isolation. As we argue in this dissertation with support of analytical and empirical data, this conjecture does hold and the methods developed herein can be successfully applied to very large resource-allocation problems where agents' preferences are defined by the underlying stochastic planning problems.

1.2 Main Contributions

The main contribution of this dissertation is a class of new resource-allocation methods for problems where agents' utility functions are induced by Markov decision processes. The main result of this work is that it recognizes that there is a lot of structure in such MDP-induced preferences, which can be exploited to yield drastic (often exponential) reductions in computational complexity of the resource-allocation algorithm.

More specifically, the major contributions of the work presented in this dissertation are as follows (depicted schematically in Figure 1.1 with the labels in the figure corresponding to the numbering in the list below).

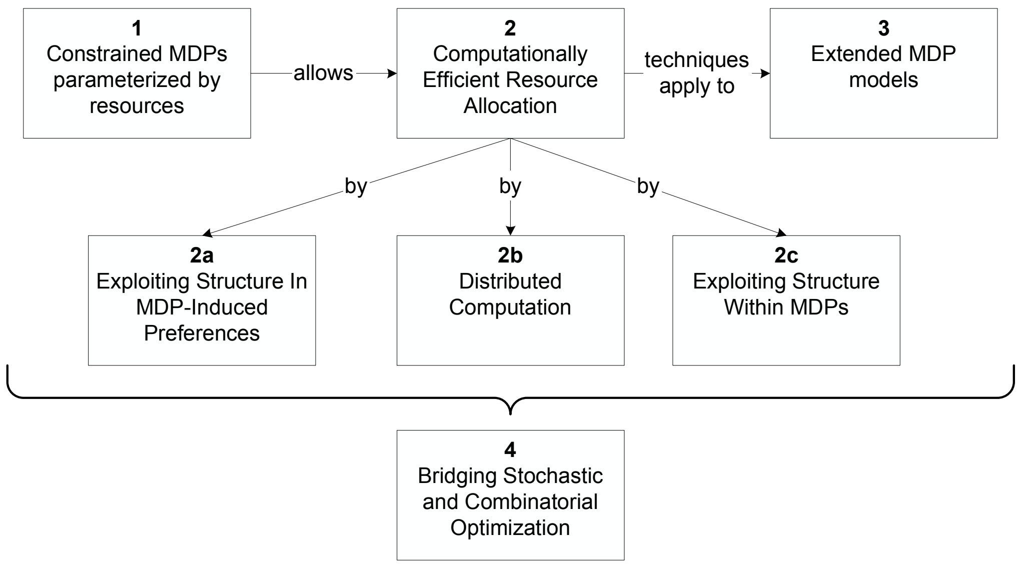

1
Constrained MDPs parameterized by resources
allows
2
Computationally Efficient Resource Allocation
techniques apply to
3
Extended MDP models
by
by
by
2a
Exploiting Structure In MDP-Induced Preferences
2b
Distributed Computation
2c
Exploiting Structure Within MDPs
4
Bridging Stochastic and Combinatorial Optimization

Figure 1.1: Contributions.

1. **Markov decision processes with resources and capacity constraints**
   We present new models based on the framework of Markov decision processes (MDPs) of stochastic-planning problems for agents whose capabilities are parameterized by the resources available to them. Further, the models capture situations where the agents have limited capacities that restrict what sets of resources they can make use of. The benefit of making the notion of resources and capacities explicit in the stochastic planning models is that is allows a parameterization of the planning problem that supports the development of efficient multiagent resource-allocation methods.

2. **Computationally efficient resource-allocation methods**
   We develop and evaluate a suite of computationally efficient resource-allocation methods for agents with preferences induced by MDPs. The computational efficiency is achieved through the use of the following techniques.

   (a) **Exploiting structure due to MDP-based preferences**
   The algorithms exploit the structure that stems from agents' MDP-based preference models. As we show in this thesis, making use of that structure and simultaneously solving the resource-allocation and planning problems leads to a drastic reduction in computational complexity. We present resource-allocation mechanisms that are broadly applicable to both cooperative and self-interested agents.

   (b) **Distributing the computation**
   In multiagent systems, a further increase in computational efficiency can come from distributing the resource-allocation and planning algorithms, thus offloading some of the computation to the participating agents. However, for groups of self-interested agents, care must be taken in passing information between agents, because they might not want to reveal their private information. We show that our resource-allocation mechanism can be effectively distributed and, further, for

domains involving self-interested agents, this can be accomplished without revealing agents' private information. The distributed version also maintains other important properties of the base mechanism, such as strategic simplicity for the participating agents (truth telling is a dominant strategy).

(c) **Exploiting structure within MDPs**
Markov decision processes are subject to the *curse of dimensionality*, a term introduced by Bellman (1961) to refer to the fact that the number of states of a system grows exponentially with the number of discrete variables that define the system state. To address this, *factored* models of MDPs have been proposed (Boutilier, Dearden, & Goldszmidt, 1995) that exploit the structure and sparsity within the MDPs for more compact representations and computationally efficient approximate solutions. By extending existing techniques based on *approximate linear programming* and developing new ones, we show how our new resource-allocation algorithms can be adapted to work with factored MDPs. This enables scaling to extremely large problems with hundreds of resource types, tens of agents, and billions of world states.

3. **Extending stochastic planning methods**
Several extensions to classical MDPs can be modeled as special cases or simple extensions of the stochastic planning algorithms that we develop in this dissertation for resource-bounded agents. In particular, we present algorithms for finding optimal (stationary deterministic) policies for MDPs with cost constraints and multiple discount factors, as well as algorithms for finding approximately optimal (stationary randomized and stationary deterministic) policies for MDPs with risk-sensitive constraints and objective functions. Such extensions to MDPs have been studied previously, but for some previously formulated models (e.g., MDPs with multiple discounts) no prior implementable solution algorithms have existed. Thus, this work furthers the field of stochastic planning by providing new algorithms for some extended models of MDPs.

4. **Bridge between stochastic and combinatorial optimization**
Another (perhaps more speculative) contribution of this thesis is that, by considering resource-allocation and planning problems concurrently, it strengthens the currently underdeveloped link between the research areas of combinatorial and stochastic optimization. This dissertation only begins to explore this broader connection, but our results indicate that the relationship can be very synergistic.

1.3 Overview of the Thesis

The rest of the thesis is organized as follows:

Chapter 2 begins the analysis of the problem for non-consumable resources (resources that enable the execution of actions, but are not themselves consumed in the processes). This chapter introduces our single-agent MDP-based model, where the action set of the MDP is parametrized by the resources available to the agent, and the resources that the agent can make use of are constrained by the agent's limited capacities. The properties of the single-agent optimization problem are discussed and a solution algorithm for obtaining optimal policies is presented.

Chapter 3 presents a mechanism for allocating scarce non-consumable resources among agents, whose local optimization problems are defined by the model discussed in the previous chapter. The chapter focuses on an auction-based mechanism for distributing resources among self-interested agents (the fully cooperative case follows trivially). A distributed, privacy-preserving, version of the mechanism is also described. The chapter also contains an empirical evaluation of the efficacy of this method.

Chapter 4 discusses how the policy-optimization algorithms developed in the previous two chapters for resource-constrained agents can be applied to other models of MDPs that have been previously formulated in the literature. In particular, we show how the algorithm from Chapter 2 can be adapted to produce optimal stationary deterministic policies for constrained MDPs with multiple discount factors, a problem for which no implementable algorithms have previously existed.

Chapter 5 deals with consumable resources, i.e., resources that are consumed during action execution. The biggest difficulty in this setting is due to the fact that the value of a resource bundle to an agent can be ambiguous, because the total consumption of a consumable resource in a stochastic environment is not deterministic. In this chapter we analyze two models. The first one captures the risk-neutral case, where the value of a particular set of resources to an agent is defined as the value of best policy whose resource requirements, on average, do not exceed the available amounts. The second model captures the more difficult risk-sensitive case, where the value of a set of resources to an agent is defined as the value of the best policy whose probability of exceeding the available resources does not exceed the specified bounds. The methodology developed in this chapter for risk-sensitive constraints can also be applied more generally to solve for optimal stationary policies in MDPs with arbitrary utility functions.

Chapter 6 considers the problem where the interactions between agents are not completely defined by the shared resources. Rather, the agents can affect the dynamics and rewards in each others' planning problems. This chapter analyzes the effects of model locality and sparseness on the solutions. The main question studied in this chapter is whether, under what conditions, and to what extent, can the compactness of the multiagent MDP model be maintained in the solution to the problem.

Chapter 7 extends the approach developed in the previous chapters to the case of well-structured MDPs (represented by factored models). This chapter demonstrates how we can design resource-allocation algorithms that effectively exploit both the structure due to the agents' MDP-induced preferences as well as the structure within the MDPs themselves to scale to very large domains.

Chapter 8 concludes with a summary of the main contributions of the thesis, its limitations, and a discussion of the questions that remain open.

CHAPTER 2

Non-Consumable Resources: Single-Agent Model

In this chapter we begin our analysis of resource-allocation and planning problems for domains that involve non-consumable resources. These resources enable the execution of actions, but are not consumed in the process. For example, an agent whose task is to transport goods needs vehicles to be able to make its deliveries. In that domain, a vehicle is an example of a non-consumable resource (unless, of course, its depreciation is taken into account). In this chapter we develop a single-agent model of the stochastic planning problem, where the action set of the agent depends on the resources available to it. The model and algorithms developed here form the foundation of the multiagent resource-allocation problem discussed in Chapter 3.[1]

We model an agent's decision-making problem as a Markov decision process (e.g., Puterman, 1994), where the action space of the MDP is defined by the resources available to the agent. In other words, an agent's MDP is parameterized by the available resources, and the agent's utility for a particular set of resources is defined as the expected value of the best policy that is realizable given the actions that it can execute using these resources.

Furthermore, a realistic agent has inherent limitations that constrain the sets of resources it can make use of (and thus what policies it can execute). For example, in a production setting, the size of a factory's workforce limits the amount of equipment that can be fully utilized. Therefore, the problem of acting optimally under the constraints of the agent's inherent limitations arises. This problem has been studied under various contexts and using different models of agents' limitations — some that directly restrict the agents' policy or strategy spaces (Russell & Subramanian, 1995; Bowling & Veloso, 2004), and some that model agents' limitations via the concept of resources (e.g., Benazera, Brafman, Mealeau, & Hansen, 2005). The latter approach typically makes more detailed assumptions about the structure of agents' constraints, which can often be exploited in practical algorithms. As viewed from the perspective of modeling of the agents' limitations, our work falls into the latter, resource-centric category.

Agents' resource limitations can be modeled within the agent's MDP, but this approach (while fully general) leads to an explosion in the size of the state space (Meuleau, Hauskrecht, Kim, Peshkin, Kaelbling, Dean, & Boutilier, 1998). In our model, we provide an alternative way of representing such limitations. We define, for every resource, a set of *capacity costs* and, for every agent, a set of *capacity limits.* We assume that capacity costs are additive and say that an agent can utilize only resource bundles whose total capacity

[1]The material in this chapter and the following one is largely based on the work that was originally reported in (Dolgov & Durfee, 2004c) and (Dolgov & Durfee, 2005a).

costs do not exceed the agent's capacity limits.[2]

The rest of this chapter is organized as follows. After a brief overview of the theory of classical unconstrained Markov decision processes, we introduce our model of the resource-constrained decision-making agent, and describe the single-agent stochastic policy-optimization problem that defines an agent's preferences over resources. We analyze the properties of this optimization problem (class of optimal policies, complexity, etc.) and present a solution algorithm.

2.1 Planning Under Uncertainty: Markov Decision Processes

Throughout this thesis the agents' decision problems are based on the model of discrete-time, fully-observable MDPs with finite state and action spaces. This section presents a very brief overview of the basic MDP results, a more detailed discussion of which can be found in the following texts: (Puterman, 1994; Bertsekas & Tsitsiklis, 1996; Sutton & Barto, 1998).

A classical single-agent, unconstrained, fully-observable MDP can be defined as a 4-tuple $\langle \mathcal{S}, \mathcal{A}, p, r \rangle$, where:

- $\mathcal{S} = \{s\}$ is a finite set of states the agent can be in.
- $\mathcal{A} = \{a\}$ is a finite set of actions the agent can execute.
- $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ defines the transition function. The probability that the agent goes to state σ if it executes action a in state s is $p(\sigma|s, a)$. We assume that, for any action, the corresponding transition matrix is stochastic: $\sum_\sigma p(\sigma|s, a) = 1 \ \forall s \in \mathcal{S}, a \in \mathcal{A}$.
- $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ defines the reward function. The agent obtains a reward of $r(s, a)$ if it executes action a in state s. We assume the rewards are bounded, i.e., $\exists r_{max} : |r(s, a)| \le r_{max} \ \forall s \in \mathcal{S}, a \in \mathcal{A}$.

In a discrete-time fully-observable MDP, at each time step, the agent observes the current state of the system and chooses an action that affects its immediate reward and the probabilities of future states. In an unconstrained MDP, the goal of the agent is to find a *policy* that maximizes some measure of aggregate reward the agent receives.

A policy is said to be *Markovian* (or *history-independent*) if the choice of action does not depend on the history of states and actions encountered in the past, but rather only on the current state and time. If, in addition to that, the policy does not depend on time, it is called *stationary*. By definition, a stationary policy is always Markovian. A *deterministic* policy always prescribes the execution of the same action in a state, while a *randomized* policy chooses actions according to a probability distribution.

Following the standard notation (Puterman, 1994), we refer to different classes of policies as Π^{xy}, where $x = \{H, M, S\}$ specifies whether a policy is History-dependent, Markovian, or Stationary, and $y = \{R, D\}$ specifies whether the policy is Randomized or Deterministic (e.g., the class of stationary deterministic policies is labeled Π^{SD}). Obviously, $\Pi^{\mathrm{H}y} \supset \Pi^{\mathrm{M}y} \supset \Pi^{\mathrm{S}y}$ and $\Pi^{x\mathrm{R}} \supset \Pi^{x\mathrm{D}}$, with history-dependent randomized policies Π^{HR} and stationary deterministic policies Π^{SD} being the most and the least general, respectively.

[2]As we discuss below, this model is general enough to model arbitrary non-decreasing utility functions.

A stationary randomized policy is thus a mapping of states to probability distributions over actions: $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0,1]$, where $\pi(s,a)$ defines the probability that action a is executed in state s. A stationary deterministic policy can be represented as a degenerate randomized policy for which there is only one action for each state that has a nonzero probability of being executed.

In an unconstrained discounted MDP, the goal is to find a policy that maximizes the total expected discounted reward over an infinite time horizon:[3]

$$U_\gamma(\pi,\alpha) = \mathbb{E}\Big[\sum_{t=0}^{\infty}(\gamma)^t r_t(\pi,\alpha)\Big], \tag{2.1}$$

where $\gamma \in [0,1)$ is the discount factor, and r_t is the (random) reward the agent receives at time t, whose distribution depends on the policy π and the initial distribution over the state space $\alpha : \mathcal{S} \mapsto [0,1]$.

One of the most important results in the theory of MDPs states that, for an unconstrained discounted MDP with the total expected reward optimization criterion, there always exists an optimal policy that is stationary, deterministic, and *uniformly optimal*, where the latter term means that the policy is optimal for all distributions over the starting state.

There are several commonly used ways of finding the optimal policy, and central to all of them is the concept of a *value function* of a policy, $v_\pi : \mathcal{S} \mapsto \mathbb{R}$, where $v_\pi(s)$ is the expected cumulative value of the reward the agent would receive if it started in state s and behaved according to policy π.

$$v_\pi(s) = U_\gamma\big(\pi, \alpha(s) = 1\big) \tag{2.2}$$

For a given policy π, the value of every state is the unique solution to the following system of $|\mathcal{S}|$ linear equations:

$$v_\pi(s) = \sum_a r(s,a)\pi(s,a) + \gamma \sum_\sigma p(\sigma|s,a)v_\pi(\sigma), \qquad \forall s \in \mathcal{S}. \tag{2.3}$$

or, equivalently, in vector form:

$$v_\pi = \mathcal{B}_\pi v_\pi, \tag{2.4}$$

where the linear operator $\mathcal{B}_\pi$ is often referred to as the *Bellman backup operator* of policy π. Thus, the value function v_π of a policy is the unique fixed point of the corresponding linear operator $\mathcal{B}_\pi$.

To find the optimal policy, it is handy to consider the *optimal value function* $v^* : \mathcal{S} \mapsto \mathbb{R}$, where $v^*(s)$ represents the value of state s, given that the agent behaves optimally. The optimal value function satisfies the following system of $|\mathcal{S}|$ nonlinear equations:

$$v^*(s) = \max_a \Big[r(s,a) + \gamma \sum_\sigma p(\sigma|s,a)v^*(\sigma)\Big], \qquad \forall s \in \mathcal{S}. \tag{2.5}$$

or:

$$v^* = \mathcal{B}^* v^*, \tag{2.6}$$

[3]Notation: here and in the rest of the thesis $(x)^y$ is an exponent, while x^y is a superscript.

where $\mathcal{B}^*$ is the nonlinear *Bellman operator*. Thus, the value function v^* of an optimal policy is the unique fixed point of the Bellman operator $\mathcal{B}^*$. Given the optimal value function v^* an optimal policy is to simply act greedily with respect to v^*:[4]

$$\pi(s,a) = \begin{cases} 1 & \text{if } \ a = \arg\max_a \left[r(s,a) + \gamma \sum_\sigma p(\sigma|s,a) v^*(\sigma) \right] \\ 0 & \text{otherwise.} \end{cases} \tag{2.7}$$

There are several well-known ways of computing the optimal value function. One approach is to use an iterative algorithm such as *value iteration* or *policy iteration* to solve (2.5). For example, value iteration iteratively updates the value function by repeatedly applying the Bellman operator:

$$v^{n+1} \leftarrow \mathcal{B}^* v^n, \tag{2.8}$$

which is guaranteed to converge to the optimal value function v^*, because $\mathcal{B}^*$ is a contraction, meaning that each update (2.8) reduces the L_∞ distance (component-wise max of absolute value) between v^n and v^*.

Another way of solving for the optimal value function is to formulate the nonlinear system (2.5) as a linear program (LP) with $|\mathcal{S}|$ optimization variables $v(s)$ and $|\mathcal{S}||\mathcal{A}|$ constraints: Intuitively, a nonlinear system of the form $z = \max\{c_1, c_2, \dots c_n\}$, can be formulated as an LP of the form $\min\{z\}$ subject to constraints $z \geq c_i \ \forall i$. For (2.5), this translates to the following minimization LP with $|\mathcal{S}|$ optimization variables $v(s)$ and $|\mathcal{S}||\mathcal{A}|$ constraints:

$$\begin{aligned} &\min \sum_s \alpha(s) v(s) \\ &\text{subject to:} \\ &v(s) \geq r(s,a) + \gamma \sum_\sigma p(\sigma|s,a) v(\sigma), \qquad \forall s \in \mathcal{S}, a \in \mathcal{A}, \end{aligned} \tag{2.9}$$

where α is an arbitrary constant vector with $|\mathcal{S}|$ strictly positive components ($\alpha(s) > 0 \, \forall s \in \mathcal{S}$).[5]

It is often very useful to consider the equivalent dual LP with $|\mathcal{S}||\mathcal{A}|$ optimization variables $x(s,a)$ and $|\mathcal{S}|$ constraints:[6]

$$\begin{aligned} &\max \sum_s \sum_a r(s,a) x(s,a) \\ &\text{subject to:} \\ &\sum_a x(\sigma,a) - \gamma \sum_s \sum_a x(s,a) p(\sigma|s,a) = \alpha(\sigma), \qquad \forall \sigma \in \mathcal{S}; \\ &x(s,a) \geq 0 \qquad \forall s \in \mathcal{S}, a \in \mathcal{A}. \end{aligned} \tag{2.10}$$

The optimization variables $x(s,a)$ are often called the *visitation frequencies* or the *occupation measure* of a policy. If we think of α as the initial probability distribution,

[4] Assuming there are no ties; if there are multiple optimal actions for a state, one can be selected arbitrarily.

[5] The overloading of α as the objective function coefficients here and the initial probability distribution of an MDP earlier is intentional and is explained shortly.

[6] Note that some authors (e.g., Altman, 1996, 1998) prefer the opposite convention, where (2.10) is called the dual, and (2.9), the primal.

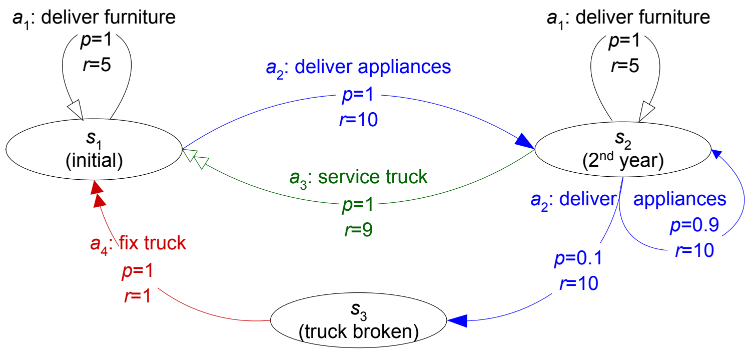


Figure 2.1: Unconstrained MDP example, delivery domain. Transition probabilities and rewards are shown on the diagram. Actions not shown result in transition to same state with no reward. There is also a noop action a_0 that corresponds to doing nothing; it does not change state and produces zero reward. Here and throughout the thesis, transitions corresponding to different actions are shown using different arrow styles.

then $x(s, a)$ can be interpreted as the total expected number of times action a is executed in state s. Then, $x(s) = \sum_a x(s, a)$ gives the total expected *flow* through state s, and the constraints in the above LP can be interpreted as the conservation of flow through each of the states.

An optimal policy can be computed from a solution to the above LP as:

$$\pi(s, a) = \frac{x(s, a)}{\sum_a x(s, a)}, \tag{2.11}$$

where non-negativity of α guarantees that $\sum_a x(s, a) > 0 \ \forall s \in \mathcal{S}$. In general, this appears to lead to randomized policies. However, a bounded LP with n constraints always has a *basic feasible solution* (e.g., (Bertsimas & Tsitsiklis, 1997)), which by definition has no more than n non-zero components. If α is strictly positive, a basic feasible solution to the LP (2.11) will have precisely $|S|$ nonzero components (one for each state), which guarantees an existence of an optimal deterministic policy. Such a policy can be easily obtained by most LP solvers (e.g., simplex will always produce solutions that map to deterministic policies).

Furthermore, as mentioned above, for unconstrained discounted MDPs, there always exist policies that are uniformly optimal (optimal for all initial distributions). The above LP (2.10) yields uniformly optimal policies if a strictly positive α is used. However, the solution (x) to the LP retains its interpretation as the expected number of times state s is visited and action a is executed only for the initial probability distribution α that was used in the LP. For MDP for which uniformly optimal solutions do not exist (as is the case for most problems in this thesis), an arbitrary initial distribution α can be used, but the resulting policy is only optimal for that α.

Example 2.1. *Consider a simple delivery domain, depicted in Figure 2.1, where the agent can obtain rewards for delivering furniture (action a_1) or delivering appliances (action a_2). Delivering appliances produces higher rewards (as shown on the diagram), but it does more* *damage to the delivery vehicle. If the agent only delivers furniture, the damage to the vehicle is negligible, whereas if the agent delivers appliances, the vehicle is guaranteed to function reliably for the first year (state s_1), but after that (state s_2) has a 10% probability of failure, per year. The vehicle can be serviced (action a_3), reseting its condition, but at the expense of lowering profits. If the truck does break (state s_3), it can be repaired (action a_4), but with a more significant impact on profits. We will assume a discount factor of* 0.9.

The optimal value function for this problem is as follows: $v^*(s_1) = 95.3, v^*(s_2) = 94.7, v^*(s_3) = 86.7$, *and the corresponding optimal occupation measure (assuming a uniform α) is the following (listing only the non-zero elements):* $x(s_1, a_2) = 4.9, x(s_2, a_3) = 4.8, x(s_3, a_4) = 0.33$. *This maps to the optimal policy that dictates that the agent is to start by delivering appliances (action a_2 in state s_1), then service the vehicle after the first year (action a_3 in state s_2), and fix the vehicle if it ever gets broken (a_4 in s_3) (the latter has zero probability of happening if the agent starts in state s_1 or s_2).* ■

A slight generalization of the discounted MDP described above is a *contracting* MDP model (van Nunen, 1976; Kallenberg, 1983). An MDP $\langle \mathcal{S}, \mathcal{A}, p, r \rangle$ is called contracting if its transition function satisfies the condition that there exists a vector μ with all positive components ($\mu(s) > 0$) and a scalar $\gamma \in [0, 1)$, such that

$$\sum_{\sigma} p(\sigma|s, a)\mu(\sigma) \leq \gamma\mu(s) \qquad \forall a \in \mathcal{A}, s \in \mathcal{S}. \tag{2.12}$$

It is easy to see that the discounted MDP is a special case of the contracting model. Indeed, a problem with a discount coefficient $\gamma \in [0, 1)$ and a stochastic transition function ($\sum_{\sigma} p(\sigma|s, a) = 1$) can be modeled as a contracting problem with a substochastic transition function ($\sum_{\sigma} p(\sigma|s, a) < 1$) by setting $p(\sigma|s, a) \leftarrow \gamma p(\sigma|s, a)$. Clearly, a problem with such a transition function satisfies (2.12) (just let $\mu(s) = 1\ \forall s$). A contracting MDP can be viewed simply as a discounted MDP with each state having its own discount coefficient that is "folded into" the transition function.

In majority of the thesis we use only the more popular discounted MDP model, although all of our results generalize directly to the contracting model, as well as other flavors of MDP models, such as the MDP with average per-step rewards. In Chapter 5, which deals with consumable resources, we sometimes refer to the contracting MDP, as the latter is more intuitive in some contexts.

2.2 Agent Model: MDPs with Resources and Capacity Constraints

It is often the case that an agent has many capabilities that are all in principle available to it, but not all combinations of these capabilities are realizable, because choosing to enable some of the capabilities might seize the resources needed to enable others. In other words, the space of feasible policies is constrained, since a particular policy might not be feasible if the agent's architecture does not support the combination of capabilities required for that policy.

We model the agent's resource-parametrized MDP as follows. The agent has a set of actions that are potentially executable, and each action requires a certain combination of resources. To capture local constraints on the sets of resources that an agent can use, we use the concept of capacities: each resource has capacity costs associated with it, and each agent has capacity constraints. For example, a delivery company needs vehicles and loading equipment (resources to be allocated) to make its deliveries (execute actions). However, all equipment costs money and requires manpower to operate it (the agent's local capacity costs). Therefore, the amount of equipment the agent can acquire and successfully utilize is constrained by factors such as its budget and limited manpower (agent's local capacity bounds). This two-layer model with capacities and resources represented separately might seem unnecessarily complex (why not fold them together or impose constraints directly on resources?), but the separation becomes evident and useful in the multiagent model discussed in Chapter 3. We emphasize the difference: the resources are the items being allocated between the agents, while the capacities are used to model the inherent limitations of the individual agents.

The agent's optimization problem is to choose a subset of the available resources that does not violate its capacity constraints, such that the best policy feasible under that bundle of resources yields the highest utility. In other words, the single-agent problem analyzed in this section has no constraints on the total resource amounts (they are introduced in the multiagent problem in the next section), and the constraints are only due to the agent's capacity limits.[7]

We can model this optimization problem as an n-tuple $\langle \mathcal{S}, \mathcal{A}, p, r, \mathcal{O}, \rho_o, \mathcal{C}, \kappa, \widehat{\kappa}, \alpha \rangle$, where:

- $\langle \mathcal{S}, \mathcal{A}, p, r \rangle$ are the standard components of an MDP, as defined earlier in Section 2.1.
- $\mathcal{O} = \{o\}$ is the set of resources (e.g., $\mathcal{O} = \{$production equipment, vehicle, $\ldots\}$).
- $\rho_o : \mathcal{A} \times \mathcal{O} \mapsto \mathbb{R}$ is a function that specifies the resource requirements of all actions; $\rho_o(a, o)$ defines how much of resource o action a needs to be executable (e.g., $\rho_o(a, \text{vehicle}) = 1$ means that action a requires one vehicle).[8]
- $\mathcal{C} = \{c\}$ is the set of capacities (e.g., $\mathcal{C} = \{$space, money, manpower, $\ldots\}$).
- $\kappa_o : \mathcal{O} \times \mathcal{C} \mapsto \mathbb{R}$ is a function that specifies the capacity costs of resources; $\kappa(o, c)$ defines how much of capacity c a unit of resource o consumes (e.g., $\kappa_o(\text{vehicle}, \text{money}) = \$50000$ defines the cost of a vehicle, while $\kappa_o(\text{vehicle}, \text{manpower}) = 2$ means that two people are required to operate the vehicle).
- $\widehat{\kappa} : \mathcal{C} \mapsto \mathbb{R}$ specifies the upper bound on the capacities; $\widehat{\kappa}$ gives the upper bound on capacity c (e.g., $\widehat{\kappa}(\text{money}) = \$1{,}000{,}000$ defines the budget constraint, $\widehat{\kappa}(\text{manpower}) = 7$ specifies the size of the workforce).
- $\alpha : \mathcal{S} \mapsto \mathbb{R}$ is the initial probability distribution; $\alpha(s)$ is the probability that the agent starts in state s.

In the above, actions are mapped to resources, and resources are mapped to capacity costs. This two-level model is needed for multiagent problems, where resources are shared by the

[7]Dealing with limited resource amounts in the single-agent setting is trivial; such constraints can be handled with a simple pruning of the agent's action space.

[8]In our model, the resource requirements of actions are independent of state, which is true in many domains (e.g., driving requires a vehicle, regardless of the location). For domains where this is not the case, the action set can be expanded, or the algorithms can be modified without much difficulty.

agents, but capacity costs are local. For single-agent problems, the two functions can be merged.

Our goal is to find a policy π that yields the highest expected reward, under the conditions that the resource requirements of that policy do not exceed the capacity bounds of the agent. In other words, we have to solve the following mathematical program (it assumes a stationary policy, which is supported by the argument in Section 2.3):

$$\begin{aligned} &\max U_\gamma(\pi, \alpha) \\ &\text{subject to:} \\ &\sum_o \kappa(o,c) \max_a \Big\{ \rho_o(a,o) H\big(\sum_s \pi(s,a)\big) \Big\} \le \widehat{\kappa}(c), \qquad \forall c \in \mathcal{C}, \end{aligned} \tag{2.13}$$

where H is the Heaviside "step" function of a nonnegative argument, defined as:

$$H(z) = \begin{cases} 0 & z = 0 \\ 1 & z > 0 \end{cases}$$

The constraint in (2.13) can be interpreted as follows. The argument of H is nonzero if the policy π assigns a nonzero probability of using action a in at least one state. Thus, $H(\sum_s \pi(s,a))$ serves as an indicator function that tells us whether the agent plans to use action a in its policy, and $\max\big\{\rho_o(a,o)H(\sum_s \pi(s,a))\big\}$ tells us how much of resource o the agent needs for its policy. We take a max with respect to a, because the same resource o can be used by different actions, such as in the delivery domain, where driving requires one person and loading/unloading appliances takes two people, so a policy for delivering appliances overall needs two people (and not three). Therefore, when summed over all resources o, the left-hand side gives us the total requirements of policy π in terms of capacity c, which has to be no greater than the bound $\widehat{\kappa}(c)$.

The following example illustrates the single-agent model.

Example 2.2. *Let us augment Example 2.1 as follows. Suppose the agent needs to obtain a truck to perform its delivery actions (a_1 and a_2). The truck is also required by the service and repair actions (a_3 and a_4). Further, to deliver appliances, it needs to acquire a forklift, and it needs to hire a mechanic to be able to repair the vehicle (a_4). The noop action a_0 requires no resources. This problem maps to a model with three resources (truck, forklift, and mechanic): $\mathcal{O} = \{o_t, o_f, o_m\}$, and the following action resource costs (listing only the non-zero ones):*

$$\begin{aligned} &\rho_o(a_1, o_t) = 1, \\ &\rho_o(a_2, o_t) = 1, \qquad \rho_o(a_2, o_f) = 1, \\ &\rho_o(a_3, o_t) = 1, \\ &\rho_o(a_4, o_t) = 1, \qquad \rho_o(a_4, o_m) = 1. \end{aligned}$$

Moreover, suppose that the resources (truck o_t, forklift o_f, or mechanic o_m) have the following capacity costs (there is only one capacity type, money: $\mathcal{C} = \{c_1\}$)

$$\kappa_o(o_t, c_1) = 2, \ \kappa_o(o_f, c_1) = 3, \ \kappa_o(o_m, c_1) = 4,$$

and the agent has a limited budget of $\widehat{\kappa} = 8$. It can, therefore acquire no more than two of the three resources, which means that the optimal solution to the unconstrained problem as in Example 2.1 is no longer feasible. ■

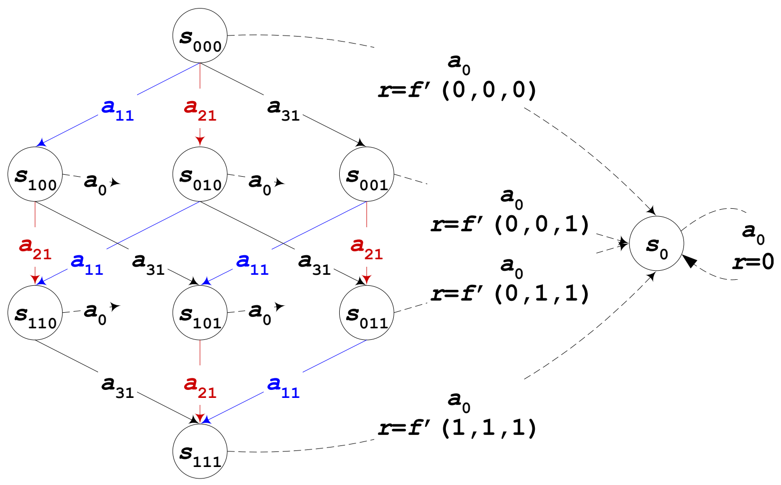


Figure 2.2: Creating an MDP with resources for an arbitrary non-decreasing utility function. The case shown has three binary resources. All transitions are deterministic.

The MDP-based model of agents' preferences presented above is fully general for discrete indivisible resources, i.e., any non-decreasing utility function over resource bundles can be represented via the resource-constrained MDP model described above.

Theorem 2.3. *Consider a finite set of n indivisible resources $\mathcal{O} = \{o_i\}$ ($i \in [1, n]$), with $m \in \mathbb{N}$ available units of each resource. Then, for any non-decreasing utility function defined over resource bundles $f : [0, m]^n \mapsto \mathbb{R}$, there exists a resource-constrained MDP $\langle \mathcal{S}, \mathcal{A}, p, r, \mathcal{O}, \rho_o, \mathcal{C}, \kappa, \widehat{\kappa}, \alpha\rangle$ (with the same resource set $\mathcal{O}$) whose induced utility function over the resource bundles is the same as f. In other words, for every resource bundle $\mathbf{z} \in [0, m]^n$, the value of the optimal policy among those whose resource requirements do not exceed $\mathbf{z}$ (call this set $\Pi(\mathbf{z})$) is the same as $f(\mathbf{z})$:*

$$\forall f : [0, m]^n \mapsto \mathbb{R},\ \exists\ \langle \mathcal{S}, \mathcal{A}, p, r, \mathcal{O}, \rho_o, \mathcal{C}, \kappa, \widehat{\kappa}, \alpha\rangle :$$

$$\forall\ \mathbf{z} \in [0, m]^n, \Pi(\mathbf{z}) = \Big\{\pi \Big| \max_a \Big[\rho_o(a, o_i) H\big(\sum_s \pi(s, a)\big)\Big] \le z_i\Big\} \implies$$

$$\max_{\pi\in\Pi(\mathbf{z})} U_\gamma(\pi, \alpha) = f(\mathbf{z}).$$

Proof. This statement can be shown via a straightforward construction of an MDP that has an exponential number (one per resource bundle) of states or actions. Below we present a different reduction with a linear number of actions and an exponential number of states. Our choice is due to the fact that, although the reverse mapping requiring two states and exponentially many actions is even more straightforward, such an MDP feels somewhat unnatural.

Given an arbitrary non-decreasing utility function f, a corresponding MDP can be constructed as follows (illustrated in Figure 2.2 for $n = 3$ and $m = 1$). The state space $\mathcal{S}$ of the MDP consists of $(m + 1)^n + 1$ states — one state ($s_\mathbf{z}$) for every resource bundle $\mathbf{z} \in [0, m]^n$, plus a sink state (s_0). Intuitively, a state in the MDP corresponds to the situation where the agent has the corresponding resource bundle.

The action space of the MDP $\mathcal{A} = a_0 \bigcup \{a_{ij}\},\ i \in [1,n],\ j \in [1,m]$ consists of $mn+1$ actions — m actions per each resource $o_i,\ i \in [1,n]$, plus an additional action a_0.

The transition function p is deterministic and is defined as follows. Action a_0 is applicable in every state and leads to the sink state s_0. Every other action a_{ij} is applicable in states $s_{\mathbf{z}}$, where $z_i = (j-1)$ and leads with certainty to the states where $z_i = j$:

$$p(\sigma|s,a) = \begin{cases} 1 & a = a_{ij}, s = s_{\mathbf{z}}, \sigma = s_{\mathbf{z}'}, z_i = (j-1), z'_i = j, \\ 1 & a = a_0, \sigma = s_0, \\ 0 & \text{otherwise.} \end{cases}$$

In other words, a_{ij} only applies in states that have $j-1$ units of resource i and leads to the state where the amount of i^{th} resource increases to j.

The reward function r is defined as follows. There are no rewards in state s_0, and action a_0 is the only action that produces rewards in other states:

$$r(s,a) = \begin{cases} f'(\mathbf{z}) & a = a_o, s = s_{\mathbf{z}},\ \forall \mathbf{z} \in [0,m]^n \\ 0 & \text{otherwise,} \end{cases}$$

where f' is a simple transformation of f that compensates for the effects of discounting:

$$f'(\mathbf{z}) = f(\mathbf{z})(\gamma)^{-\sum_i z_i}.$$

In other words, it takes $\sum_i z_i$ transitions to get to state $s_{\mathbf{z}}$, so the contribution of the above into the total discounted reward will be exactly $f(\mathbf{z})$.

The resource requirements ρ_o of actions are as follows: action a_0 does not require any resources, while every other action a_{ij} requires j units of resource o_i.

Finally, the initial conditions are $\alpha(s_{\mathbf{z}=\mathbf{0}}) = 1$, meaning that the agent always starts in the state that corresponds to the empty resource bundle (state s_{000} in Figure 2.2). The capacity costs κ_o and limits $\widehat{\kappa}$ are not used, so we set $\mathcal{C} = \varnothing$.

It is easy to see that in the MDP constructed above, given a resource bundle $\mathbf{z}$, any policy from the feasible set $\Pi(\mathbf{z})$ has zero probability of reaching any state $s_{\mathbf{z}'}$ for which $\mathbf{z}' > \mathbf{z}$ (for any component i). Furthermore, an optimal policy from the set $\Pi(\mathbf{z})$ will be to transition to state $s_{\mathbf{z}}$ (since $f(\mathbf{z})$ is non-decreasing) and then use action a_0, thus obtaining a total discounted reward of $f(\mathbf{z})$. ■

Let us comment that while Theorem 2.3 establishes the generality of the MDP-based preference model introduced in this section, the construction used in the proof is of little practical interest. Indeed, mapping arbitrary unstructured utility functions to exponentially large MDPs is quite useless. The view presented in this work is exactly the opposite: in many domains utility functions are induced by a stochastic decision-making process (modeled as an MDP), thus resulting in well-structured preferences over resources that can be exploited to create computationally efficient resource-allocation algorithms.

2.3 Properties of the Single-Agent Constrained MDP

In this section, we analyze the constrained policy-optimization problem (2.13). Namely, we show that stationary deterministic policies are optimal for this problem, meaning that it is not necessary to consider randomized, or history-dependent policies. However, solutions to problem (2.13) are not, in general, uniformly optimal (optimal for any initial distribution). Furthermore, we show that (2.13) is NP-hard, unlike the unconstrained MDPs, which can be solved in polynomial time ((Littman, Dean, & Kaelbling, 1995) and references therein).

We begin by showing optimality of stationary deterministic policies for the constrained optimization problem. Recall that Π^{HR} refers to the class of history-dependent randomized policies (the most general policies), and $\Pi^{\text{SD}} \subset \Pi^{\text{HR}}$ refers to the class of stationary deterministic policies.

Theorem 2.4. *Given an MDP with resource and capacity constraints $M = \langle \mathcal{S}, \mathcal{A}, p, r, \mathcal{O}, \rho_o, \mathcal{C}, \kappa, \widehat{\kappa}, \alpha \rangle$, if there exists a policy $\pi \in \Pi^{\text{HR}}$ that is a feasible solution for M, there exists a stationary deterministic policy $\pi^{\text{SD}} \in \Pi^{\text{SD}}$ that is also feasible, and the expected total reward of π^{SD} is no less than that of π:*

$$\forall\ \pi \in \Pi^{\text{HR}},\ \exists\ \pi^{\text{SD}} \in \Pi^{\text{SD}} : U_\gamma(\pi^{\text{SD}}, \alpha) \geq U_\gamma(\pi, \alpha).$$

Proof. Let us label $\mathcal{A}' \subseteq \mathcal{A}$ the set of all actions that have a non-zero probability of being executed according to π, i.e.,

$$\mathcal{A}' = \{a | \exists s : \pi(\{s_t, a_t\}, a) > 0\}, \qquad \forall \{s_t, a_t\},$$

where $\{s_t, a_t\}$ is a sequence of state-action pairs.

Let us also construct an unconstrained MDP: $M' = \langle \mathcal{S}, \mathcal{A}', p', r' \rangle$, where p' and r' are the restricted version of p and r with the action domain limited to $\mathcal{A}'$:

$$\begin{aligned}
&p' : \mathcal{S} \times \mathcal{A}' \times \mathcal{S} \mapsto [0, 1], \\
&r' : \mathcal{S} \times \mathcal{A}' \mapsto \mathbb{R}, \\
&p'(\sigma | s, a) = p(\sigma | s, a),\ r'(s, a) = r(s, a) \quad \forall s \in \mathcal{S}, \sigma \in \mathcal{S}, a \in \mathcal{A}'.
\end{aligned}$$

Due to a well-known property of unconstrained infinite-horizon MDPs with the total expected discounted reward optimization criterion, M' is guaranteed to have an optimal stationary deterministic solution (e.g., Theorem 6.2.10 in (Puterman, 1994)), which we label π^{SD}.

Consider π^{SD} as a potential solution to M. Clearly, π^{SD} is a feasible solution, because its actions come from the set $\mathcal{A}'$ that includes actions that π uses with non-zero probability, which means that the resource requirements (as in (2.13)) of π^{SD} can be no greater than those of π. Indeed:

$$\begin{aligned}
\max_{a \in \mathcal{A}'} \Big\{ \rho_o(a, o) H\big(\sum_s \pi^{\text{SD}}(s, a) \big) \Big\} &\leq \max_{a \in \mathcal{A}'} \rho_o(a, o) \\
&= \max_{a \in \mathcal{A}} \Big\{ \rho_o(a, o) H\big(\sum_s \pi(\{s_t, a_t\}, a) \big) \Big\},
\end{aligned} \tag{2.14}$$

where the first inequality is due to the fact that $H(z) \leq 1\ \forall z$, and the second equality follows from the definition of $\mathcal{A}'$.

Furthermore, observe that π^{SD} yields the same total reward under M' and M. Additionally, since π^{SD} is a uniformly optimal solution to M', it is, in particular, optimal for the initial conditions α of the constrained MDP M. Therefore, π^{SD} constitutes a feasible solution to M whose expected reward is greater than or equal to the expected reward of any feasible policy π. ■

The result of Theorem 2.4 is not at all surprising: intuitively, stationary deterministic policies are optimal, because history-dependence does not increase the utility of the policy, and using randomization can only increase resource costs. The latter is true because including an action in a policy incurs the same costs in terms of resources regardless of the probability of executing that action (or the expected number of times the action will be executed). This is true because we are dealing with non-consumable resources, and the same property does not hold for MDPs with consumable resources (as we discuss in more detail in Section 2.6).

We now show that uniformly optimal policies do not always exist for our constrained problem. This result is well known for another class of constrained MDPs (Altman & Shwartz, 1991; Altman, 1999; Kallenberg, 1983; Puterman, 1994), where constraints are imposed on the total *expected* costs that are proportional to the expected number of times the corresponding actions are executed (as is the case with consumable resources). Here, we establish the analogous result for problems with non-consumable resources and capacity constraints, for which the costs are incurred by the agent when it includes an action in its policy, regardless of how many times the action is actually executed.

Observation 2.5. *There do not always exist uniformly optimal solutions to (2.13). In other words, there exist two constrained MDPs that differ only in their initial conditions: $M = \langle \mathcal{S}, \mathcal{A}, p, r, \mathcal{O}, \rho_o, \mathcal{C}, \kappa_o, \widehat{\kappa}, \alpha \rangle$ and $M' = \langle \mathcal{S}, \mathcal{A}, p, r, \mathcal{O}, \rho_o, \mathcal{C}, \kappa_o, \widehat{\kappa}, \alpha' \rangle$, such that there is no policy that is optimal for both problems simultaneously, i.e., for any two policies π and π' that are optimal solutions to M and M', respectively, the following holds:*

$$U_\gamma(\pi, \alpha) > U_\gamma(\pi', \alpha), \quad U_\gamma(\pi, \alpha') < U_\gamma(\pi', \alpha'). \tag{2.15}$$

We demonstrate this observation by example.

Example 2.6. *Consider the resource-constrained problem as in Example 2.2. It is easy to see that if the initial conditions are $\alpha = [1, 0, 0]$ (the agent starts in state s_1 with certainty), the optimal policy for states s_1 and s_2 is the same as in Example 2.1 ($s_1 \to a_2$ and $s_2 \to a_3$), which given the initial conditions results in zero probability of reaching state s_3, which is assigned the noop a_0. This policy requires the truck and the forklift. However, if the agent starts in state s_3 ($\alpha = [0, 0, 1]$), the optimal policy is to fix the truck (execute a_4 in s_3), and to resort to furniture delivery (do a_1 in s_1 and assign the noop a_o to s_2, which is never visited). This policy requires the mechanic and the truck. These two policies are uniquely optimal for the corresponding initial conditions, and are suboptimal for other initial conditions, which demonstrates that no uniformly optimal policy exists for this example.* ■

We now analyze the computational complexity of the optimization problem (2.13).

Theorem 2.7. *The following decision problem is NP-complete. Given an instance of an MDP $\langle \mathcal{S}, \mathcal{A}, p, r, \mathcal{O}, \rho_o, \mathcal{C}, \kappa_o, \widehat{\kappa}, \alpha \rangle$ with resources and capacity constraints, and a rational number Y, does there exist a feasible policy π, whose expected total reward, given α, equals or exceeds Y?*

Proof. As shown in Theorem 2.4, there always exists an optimal policy for (2.13) that is stationary and deterministic. Therefore, the presence in NP is obvious, since we can, in

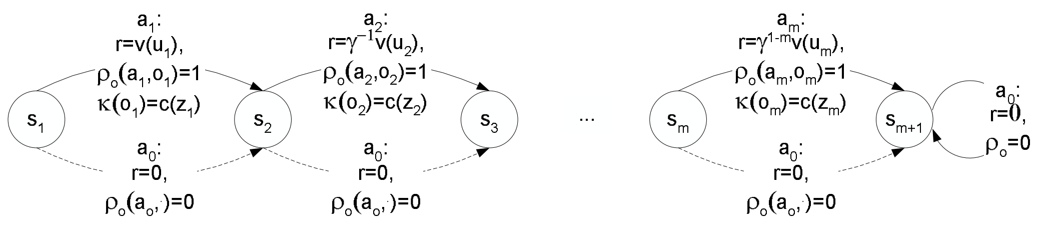


Figure 2.3: Reduction of KNAPSACK multiagent MDP with resources. All transitions are deterministic.

polynomial time, guess a stationary deterministic policy, verify that it satisfies the resource constraints, and calculate its expected total reward (the latter can be done by solving the standard system of linear Markov equations (2.3) on the values of all states).

To show NP-completeness of the problem, we use a reduction from KNAPSACK (Garey & Johnson, 1979). Recall that KNAPSACK is an NP-complete problem, which asks whether, for a given set of items $z \in \mathcal{Z}$, each of which has a cost $c(z)$ and a value $v(z)$, there exists a subset $\mathcal{Z}' \subseteq \mathcal{Z}$ such that the total value of all items in $\mathcal{Z}'$ is no less than some constant $\widehat{v}$, and the total cost of the items is no greater than another constant $\widehat{c}$, i.e., $\sum_{z\in\mathcal{Z}'} c(z) \leq \widehat{c}$ and $\sum_{z\in\mathcal{Z}'} v(z) \geq \widehat{v}$. Our reduction is illustrated in Figure 2.3 and proceeds as follows.

Given an instance of KNAPSACK with $|\mathcal{Z}| = m$, let us number all items as $z_i,\ i \in [1, m]$ as a notational convenience. For such an instance of KNAPSACK, we create an MDP with $m + 1$ states $\{s_1, s_2, \ldots s_{m+1}\}$, $m + 1$ actions $\{a_0, \ldots a_m\}$, m types of resources $\mathcal{O} = \{o_1, \ldots o_m\}$, and a single capacity $\mathcal{C} = \{c_1\}$.

The transition function on these states is defined as follows. Every state $s_i,\ i \in [1, m]$ has two transitions from it – corresponding to actions a_i and a_0. Both actions lead to state s_{i+1} with probability 1. State s_{m+1} is absorbing and all transitions from it lead back to itself.

The reward and the cost functions are constructed as follows. Every action $a_i,\ i \in [1, m]$ (which corresponds to item z_i in KNAPSACK) produces the reward of $v(z_i)(\gamma)^{1-i}$. Hence (given our transition function, which implies that state s_i is reached exactly at step $i - 1$), this means that if action $a_i, i \in [1, m]$ is ever executed, its contribution to the total discounted reward will be $v(z_i)(\gamma)^{1-i}(\gamma)^{i-1} = v(z_i)$. Action a_0 produces a reward of zero in all states.

The resource requirements of actions are defined as follows. Action $a_i,\ i \in [1, m]$ only needs resource o_i, i.e., $\rho_o(a_i, o_j) = 1 \iff i = j$. We set the cost of resource o_i to be the cost $c(z_i)$ of item i in the KNAPSACK problem. The "null" action a_0 requires no resources.

In order to complete the construction, we set the initial distribution $\alpha = [1, 0, \ldots]$, so that the agent starts in state s_1 with probability 1. We also define the decision parameter $Y = \widehat{v}$ and the upper bound on the single capacity $\widehat{\kappa} = \widehat{c}$.

Essentially, this construction allows the agent to choose action a_i or a_0 at every state s_i. Choosing action a_i is equivalent to putting item z_i into the knapsack, while action a_0 corresponds to the choice of not including z_i in the knapsack. Therefore, there exists a policy that has the expected payoff no less than $Y = \widehat{v}$ and uses no more than $\widehat{\kappa} = \widehat{c}$ of the shared resource if and only if there exists a solution to the original instance of

KNAPSACK. ■

Note that the above complexity result stems from the limited capacities of the agents and the fact that we define the resource requirements of a policy as the set of all resources needed to carry out all actions that have a nonzero probability of being executed. A different model, where the resource requirements of a policy depend on the total number of times the corresponding actions are executed would have very different properties; such a model is analyzed in Chapter 5, which deals with consumable resources.

2.4 Solution Algorithm

Now that we have analyzed the properties of the optimization problem (2.13), we present a reduction of (2.13) to a mixed integer linear program (MILP). Given that we have established NP-completeness of (2.13) in the previous section, MILP (also NP-complete) is a reasonable formulation that allows us to reap the benefits of a vast selection of efficient algorithms and tools (see, for example, (Wolsey, 1998) and references therein).

To this end, let us rewrite (2.13) in the occupation measure coordinates x. Adding the constraints from (2.13) to the standard LP in occupancy coordinates (2.10), and noticing that (for states with nonzero probability of being visited) $\pi(s,a)$ and $x(s,a)$ are either zero or nonzero simultaneously:

$$H\big(\sum_s \pi(s,a)\big) = H\big(\sum_s x(s,a)\big)$$

we get the following program in x:

$$\begin{aligned}
&\max \sum_s \sum_a x(s,a) r(s,a) \\
&\text{subject to:} \\
&\sum_a x(\sigma,a) - \gamma \sum_s \sum_a x(s,a) p(\sigma|s,a) = \alpha(\sigma), && \forall \sigma \in \mathcal{S}; \\
&\sum_o \kappa(o,c) \max_a \Big\{ \rho_o(a,o) H\big(\sum_s x(s,a)\big)\Big\} \le \widehat{\kappa}(c), && \forall c \in \mathcal{C}; \\
&x(s,a) \ge 0, && \forall s \in \mathcal{S}, a \in \mathcal{A}.
\end{aligned} \tag{2.16}$$

The challenge in solving this mathematical program is that the constraints are nonlinear due to the maximization over a and the Heaviside function H.

To get rid of the first source of nonlinearity, let us observe that a constraint with a single max over a finite set of discrete values can be trivially linearized by expanding out the set over which the maximization is taken:

$$\max_{z \in \mathcal{Z}} f(z) \le a \iff f(z) \le a \;\forall z \in \mathcal{Z}.$$

However, in (2.16), we have a finite sum of maximums, which is slightly more cumbersome, but can also be linearized by observing that the inequality

$$\sum_i^n g(u_i) \max_{z \in \mathcal{Z}} f(z,u_i) = g(u_1) \max_{z \in \mathcal{Z}} f(z,u_1) + \ldots + g(u_n) \max_{z \in \mathcal{Z}} f(z,u_n) \le a$$

is equivalent to the following system of $|\mathcal{Z}|^n$ linear inequalities:

$$g(u_1)f(z_1,u_1) + g(u_2)f(z_2,u_2) + \ldots + g(u_n)f(z_n,u_n) \leq a, \qquad \forall z_1, z_2, \ldots z_n \in \mathcal{Z}.$$

Applying this to the constraints from (2.16), we can express the original system of $|\mathcal{C}|$ nonlinear constraints (each of which has a max):

$$\sum_o \kappa(o,c) \max_a \Big\{ \rho_o(a,o) H\big(\sum_s x(s,a)\big) \Big\} \leq \widehat{\kappa}(c), \qquad \forall c \in \mathcal{C}$$

as the following system of $|\mathcal{C}||\mathcal{A}|^{|\mathcal{O}|}$ constraints where the max is removed:

$$\sum_o \kappa(o,c)\rho_o(a_o,o) H\big(\sum_s x(s,a)\big) \leq \widehat{\kappa}(c), \qquad \forall c \in \mathcal{C}, a_{o_1}, a_{o_2}, \ldots \in \mathcal{A}. \tag{2.17}$$

Notice that this straightforward way of eliminating the maximization exponentially increases the number of constraints, because the above expansion enumerates all possible actions for each resource (i.e., it enumerates policies where each resource o is used by action a_1, where it is used by action a_2, action a_3, etc.) However, in many problems not all resources are used by all actions. In such cases, most of the above constraints become redundant, and the number of constraints can be reduced from $|\mathcal{C}||\mathcal{A}|^{|\mathcal{O}|}$ to $|\mathcal{C}|\prod_o |\mathcal{A}_o|$, where $\mathcal{A}_o$ is the number of actions that use resource o. Furthermore, in a special case, where the resource requirements of actions are binary, the number of constraints can be reduced much more significantly, as discussed next in Section 2.5.

To linearize the Heaviside function, we augment the original optimization variables x with a set of $|\mathcal{A}|$ binary variables $\Delta(a) \in \{0,1\}$, where $\Delta(a) = H(\sum_s x(s,a))$. In other words, $\Delta(a)$ is an indicator variable that shows whether the agent plans to use action a in its policy. Using Δ, and expanding the max as above, we can rewrite the resource constraints in (2.16) as:

$$\sum_o \kappa(o,c)\rho_o(a_o,o)\Delta(a_o) \leq \widehat{\kappa}(c), \qquad \forall c \in \mathcal{C}, a_{o_1}, a_{o_2}, \ldots \in \mathcal{A}, \tag{2.18}$$

which are linear in Δ. We can then synchronize Δ and x via the following linear inequalities:

$$\frac{\sum_s x(s,a)}{X} \leq \Delta(a), \qquad \forall a \in \mathcal{A}, \tag{2.19}$$

where $X \geq \max \sum_s x(s,a)$ is some constant finite upper bound on the expected number of times action a is used, which exists for any discounted MDP. We can, for example, let $X = (1-\gamma)^{-1}$, since $\sum_{s,a} x(s,a) = (1-\gamma)^{-1}$ for any x that is a valid occupation measure for an MDP with discount factor γ.[9]

[9]Instead of using a single X for all resources, a different $X(a) \geq \max \sum_s x(s,a)$ can be used for every action, leading to more uniform normalization and potentially better numerical stability of the MILP solver.

Putting it all together, the problem (2.13) of finding optimal policies under resource constraints can be formulated as the following MILP:

$$\begin{aligned}
&\max \sum_s \sum_a x(s,a) r(s,a) \\
&\text{subject to:} \\
&\sum_a x(\sigma,a) - \gamma \sum_s \sum_a x(s,a) p(\sigma|s,a) = \alpha(\sigma), && \forall \sigma \in \mathcal{S}; \\
&\sum_o \kappa(o,c) \rho_o(a_o,o) \Delta(a_o) \le \widehat{\kappa}(c), && \forall c \in \mathcal{C}, a_{o_1}, a_{o_2}, \ldots \in \mathcal{A}; \\
&\sum_s x(s,a)/X \le \Delta(a), && \forall a \in \mathcal{A}; \\
&x(s,a) \ge 0, && \forall s \in \mathcal{S}, a \in \mathcal{A}; \\
&\Delta(a) \in \{0,1\}, && \forall a \in \mathcal{A};
\end{aligned} \tag{2.20}$$

We illustrate the reduction with an example.

Example 2.8. *Let us formulate the MILP for the constrained problem from Example 2.6. Recall that in that problem there are three resources $\mathcal{O} = \{o_t, o_f, o_m\}$ (truck, forklift, and mechanic), one capacity type $\mathcal{C} = \{c_1\}$ (money), and actions have the following resource requirements (listing only the nonzero ones):*

$$\begin{aligned}
\rho_o(a_1,o_t) = 1, \quad \rho_o(a_2,o_t) = 1, \rho_o(a_2,o_f) = 1, \\
\rho_o(a_3,o_t) = 1, \quad \rho_o(a_4,o_t) = 1, \rho_o(a_4,o_m) = 1.
\end{aligned}$$

The resources have the following capacity costs:

$$\kappa_o(o_t,c_1) = 2, \ \kappa_o(o_f,c_1) = 3, \ \kappa_o(o_m,c_1) = 4,$$

and the agent has a limited budget, i.e., a capacity bound $\widehat{\kappa}(c_1) = 8$.

To compute the optimal policy for an arbitrary α, we can formulate the problem as an MILP using the techniques described above. Using binary variables $\{\Delta(a_i)\} = \{\Delta_i\} = \{\Delta_1, \Delta_2, \Delta_3, \Delta_4\}$,[10] we can express the constraint on capacity cost as the following system of $|\mathcal{C}||\mathcal{A}|^{|\mathcal{O}|} = 1(4)^3 = 64$ linear constraints as in (2.18):

$$\begin{aligned}
&(2)(1)\Delta_1 + (3)(0)\Delta_1 + (4)(0)\Delta_1 \le 8, \\
&(2)(1)\Delta_1 + (3)(0)\Delta_1 + (4)(0)\Delta_2 \le 8, \\
&(2)(1)\Delta_1 + (3)(0)\Delta_1 + (4)(0)\Delta_3 \le 8, \\
&(2)(1)\Delta_1 + (3)(0)\Delta_1 + (4)(1)\Delta_4 \le 8, \\
&(2)(1)\Delta_1 + (3)(1)\Delta_2 + (4)(0)\Delta_1 \le 8, \\
&\ldots \\
&(2)(0)\Delta_4 + (3)(0)\Delta_4 + (4)(1)\Delta_4 \le 8.
\end{aligned}$$

It is easy to see that most of these constraints are redundant, and the fact that each action requires only a small subset of the resources allows us to greatly reduce the number

[10] We do not create a Δ_0 for the noop action a_0, as its resource costs are zero, and it drops out of all expressions.

of constraints. In fact, the only resource that is used by multiple actions is o_t. Therefore, in accordance with our earlier discussion, we need only $\prod_o |\mathcal{A}_o| = 1 \times 4 \times 1 = 4$ constraints:

$$\begin{aligned}
(2)(1)\Delta_1 + (3)(1)\Delta_2 + (4)(1)\Delta_4 \leq 8,\\
(2)(1)\Delta_2 + (3)(1)\Delta_2 + (4)(1)\Delta_4 \leq 8,\\
(2)(1)\Delta_3 + (3)(1)\Delta_2 + (4)(1)\Delta_4 \leq 8,\\
(2)(1)\Delta_4 + (3)(1)\Delta_2 + (4)(1)\Delta_4 \leq 8.
\end{aligned}$$

where the each of the four constraints corresponds to a case where the first resource (o_t) is used by a different action.

As mentioned earlier, we can set $X = (1-\gamma)^{-1}$ for the constraints that synchronize the occupation measure x and the binary indicators Δ. Combining this with other constraints from (2.20), we get an MILP with 12 continuous and 4 binary variables, and $|\mathcal{S}| + |\mathcal{C}| \prod_o |\mathcal{A}_o| + |\mathcal{A}| = 3 + 4 + 3 = 10$ constraints (not counting the last two sets of range constraints). ■

As mentioned earlier, even though solving such programs is, in general, an NP-complete task, there is a wide variety of very efficient algorithms and tools for doing so. Therefore, one of the benefits of reducing the optimization problem (2.13) to an MILP is that it allows us to make use of the existing highly efficient tools.

2.5 Binary Resource Costs

In some domains (as, for instance, was the case in the example from the previous section), the resource requirements of actions are binary ($\rho_o(a, o) = \{0, 1\}$). Under these conditions we can significantly simplify the optimization program (2.16).

First, let us observe that, when $\rho_o(a, o) = \{0, 1\}$, the total resource requirements of a policy can be simplified as follows:

$$\max_a \Big\{\rho_o(a, o) H\big(\sum_s x(s, a)\big)\Big\} = H\Big(\sum_a \rho_o(a, o) \sum_s x(s, a)\Big). \tag{2.21}$$

Therefore, instead of introducing $|\mathcal{A}|$ binary variables $\Delta(a)$ that tell us whether the agent plans to execute action a in its policy, we can get away with using $|\mathcal{O}|$ binary variables $\delta(o)$:

$$\delta(o) = H\Big(\sum_a \rho_o(a, o) \sum_s x(s, a)\Big), \tag{2.22}$$

where $\delta(o)$ tells us whether the agent's policy requires the use of resource o.

Given the above, we can use the same techniques as before to express the resource constraints as linear functions of δ, and to synchronize δ with x using a linear inequality to obtain the following simplified version of the optimization program (2.16):

$$\begin{aligned}
&\max \sum_s \sum_a x(s,a) r(s,a) \\
&\text{subject to:} \\
&\sum_a x(\sigma,a) - \gamma \sum_s \sum_a x(s,a) p(\sigma|s,a) = \alpha(\sigma), && \forall \sigma \in \mathcal{S}; \\
&\sum_o \kappa(o,c)\delta(o) \le \widehat{\kappa}(c), && \forall c \in \mathcal{C}; \\
&1/X' \sum_a \rho_o(a,o) \sum_s x(s,a) \le \delta(o), && \forall o \in \mathcal{O}; \\
&x(s,a) \ge 0, && \forall s \in \mathcal{S}, a \in \mathcal{A}; \\
&\delta(o) \in \{0,1\}, && \forall o \in \mathcal{O},
\end{aligned} \tag{2.23}$$

where $X' \ge \max \sum_a \rho_o(a,o) \sum_s x(s,a)$ is the normalization constant, which is the upper bound on the argument of $H()$. The bound X' is guaranteed to exist for discounted problems, and can be obtained in polynomial time, analogously to the upper bound $\max \sum_s x(s,a)$ from the previous section.

The above MILP (2.23) has far fewer constraints than (2.20): most noticeably, $|\mathcal{C}|$ instead of $|\mathcal{C}||\mathcal{A}|^{|\mathcal{O}|}$ (or, more precisely, $|\mathcal{C}| \prod_o |\mathcal{A}_o|$) capacity constraints. The number of binary variables is $|\mathcal{O}|$ instead of $|\mathcal{A}|$, which can also be significant for problems where the number of actions is much larger than the number of resources.

Example 2.9. *Following the above discussion, the optimization problem for Example 2.6, where all resource requirements are binary, can be formulated using three binary variables* $\delta(o) = \{\delta(o_t), \delta(o_f), \delta(o_m)\}$ *instead of four variables* $\Delta(a)$*, as was done in Example 2.8. Using these variables, the capacity constraints can be formulated as one inequality:*

$$2\delta(o_t) + 3\delta(o_f) + 4\delta(o_m) \le 8,$$

instead of the four inequalities in Example 2.8. ■

Given the reduction in the number of binary optimization variables, the MILP formulation presented in this section is more efficient for problems with binary resource requirements than the more general formulation described in the previous section. Furthermore, by expanding the resource set, any problem can be represented using binary resource requirements only. If the domain contains mostly binary requirements, it may be more effective to expand the non-binary resource requirements ρ_o by augmenting the resource set $\mathcal{O}$ and use the binary formulation of this section rather than directly applying the more-general formulation from Section 2.4.

2.6 Conclusions

In this chapter, we developed an MDP model, whose action set is parameterized on a set of non-consumable resources. The main contributions of the work presented in this chapter are the following:

- We presented an analysis of the properties of the resource-constrained MDP model in Section 2.2. We established that stationary deterministic policies are optimal for these MDPs, although there does not, in general, exist a policy that is optimal for all initial conditions. Further, we proved that solving these constrained MDPs to exceed a given threshold of policy value is an NP-complete task.

- We developed, in Section 2.4, a new algorithm based on the dual LP formulation for solving such MDPs with non-consumable resources and capacity constraints.

- Finally, in Section 2.5 we showed how the solution algorithm can be significantly simplified for domains where the resource requirements of all actions are binary.

While the model, analysis, and algorithms presented in this chapter are valuable for formulating and solving constrained single-agent stochastic planning problems, the main benefit of the explicit resource-centric treatment presented here becomes apparent in multiagent domains. Thus, an empirical evaluation of the approach is postponed until the following chapter, which discusses the multiagent problem.

CHAPTER 3

Allocation of Non-Consumable Resources

The previous chapter assumed that all resources are there for the taking, and those that the agent cannot obtain due to capacity constraints, or has no value for, are unused. In this chapter we consider the multiagent problem, where the resources taken by one agent influence what is available to be distributed among the others. For example, a delivery company with a finite fleet of vehicles might face the problem of allocating the vehicles among its operational branches, and if the vehicles are scarce, it is important to find an effective allocation. In this chapter we study the problem of allocating non-consumable resources among agents whose stochastic planning problems are represented via the constrained MDP model discussed in the previous chapter. We assume that the agents are self-interested, but the resource-allocation algorithms developed in this chapter are directly applicable to cooperative domains, where they can also lead to significant gains in computational efficiency.

The problem of resource allocation among multiple self-interested agents is ubiquitous, with applications ranging from decentralized scheduling (e.g., Wellman, Walsh, Wurman, & MacKie-Mason, 2001) and network routing (e.g., Feldmann, Gairing, Lucking, Monien, & Rode, 2003) to transportation logistics (e.g., Sheffi, 2004; Song & Regan, 2002) and bandwidth allocation (e.g., McMillan, 1994; McAfee & McMillan, 1996), just to name a few. A lot of work in this setting has focused on the problem of *mechanism design* (e.g., Mas-Colell, Whinston, & Green, 1995), the goal of which is to create mechanisms that allocate the resources to the agents in ways desirable to the system-designer, given that each participating agent is selfishly maximizing its own utility.

However, as mentioned in Chapter 1, the majority of this work assumes that the agents' utility functions are specified directly (see, for example, the survey (Chevaleyre, Dunne, Endriss, Lang, Lemaitre, Maudet, Padget, Phelps, Rodriguez-Aguilar, & Sousa, 2006)) and does not analyze the underlying processes that define the agents' preferences for resources. In this chapter we focus on the multiagent setting, and study mechanisms for efficient resource allocation, while taking a much closer look at agents' preferences. We show how having a detailed model of the processes that define agents' utility functions can be extremely advantageous from the point of view of the mechanism-design problem.

In particular, we analyze the problem of optimally allocating the limited shared resources among the agents, where each agent's utility function over the resources is defined by the MDP model with resources and capacity constraints, as described previously in Chapter 2. As is often done in the work on resource allocation, we assume that the utility function of each agent is only a function of its allotment of resources and its price, and it does not depend on what resources are given to other agents. In our MDP-based model, this means that the agents' planning problems become completely independent once the resources are allocated.

For domains that involve complex preferences that exhibit combinatorial effects between the resources, *combinatorial auctions* (e.g., de Vries & Vohra, 2003; Pekec & Rothkopf, 2003) are often used to solve the resource-allocation problems. In particular, Generalized Vickrey Auctions (GVAs) (MacKie-Mason & Varian, 1994) are widely studied because they produce socially optimal allocations and are simple from the strategic perspective (agents' strategic reasoning is simple, because they have dominant bidding strategies).

However, a weak point of GVAs, and combinatorial auctions (CAs) more generally, is the high computational complexity of the agents' valuation problem (how much to bid for resource bundles) (Parkes, 2001; Sandholm, 2002), and the winner-determination problem of the auctioneer (given the bids, how to allocate resources and how to set prices) (Rothkopf, Pekec, & Harstad, 1998). The high computational complexity stems from the fact that, in general, the agents need to specify preferences over an exponential number of resource bundles, and the auctioneer has to solve a combinatorial winner-determination problem on that space of bundle allocations.

Such computational challenges in combinatorial resource-allocation problems have recently lead to a move towards resource-allocation mechanisms that model the mechanisms underlying agents' resource preferences (e.g., Larson, 2004; Larson & Sandholm, 2005). In particular, in many domains, agents' preferences are well-structured, which suggests using concise preference-specification languages that exploit that structure. Towards that end, logical bidding languages have been proposed (Boutilier & Hoos, 2001; Sandholm, 2002) that allow agents to specify their valuations on logical formulas over allocations of goods, and efficient algorithms and systems have been developed that make use of such concise preference specifications (Boutilier, 2002; Hoos & Boutilier, 2000; Sandholm, 2000; Pennock & Wellman, 2000; Fortnow, Kilian, Pennock, & Wellman, 2005). In a similar fashion, we propose techniques for exploiting agents' MDP-induced preferences.

In the rest of this chapter, we develop a computationally efficient GVA-based resource-allocation mechanism. We also present a distributed version of the mechanism that leads to further gains in computational efficiency for the problem of winner determination. This speedup is achieved without sacrificing information privacy; it also maintains other important properties of the auction, such as socially optimal outcomes and excellent strategic complexity for the participating agents (the mechanism is strategy-proof). Finally, in Section 3.2, we analyze experimentally the computational efficiency of our resource-allocation mechanism.

3.1 Multiagent Resource Allocation

We assume that agents are weakly coupled (Meuleau et al., 1998), i.e., they only interact through the shared resources, and once the resources are allocated, the agents' transitions and rewards are independent.[1]

[1]If agents are cooperative, the assumption about weak coupling can be relaxed (at the expense of an increase in complexity), and the same MILP-based algorithm for simultaneously performing policy optimization and resource allocation can be applied if we consider the joint state spaces of the interacting agents. For self-interested agents, a violation of the weakly-coupled assumption would mean that the agents would be playing a Markov game (Shapley, 1953) once the resources

Also, to simplify the following discussion, we assume that agents' resource requirements are binary (as discussed in Section 2.5), but all results can be generalized to the non-binary case in a straightforward manner.

Formally, the input to the resource-allocation problem consists of the following:

- $\mathcal{M} = \{m\}$ is the set of agents.
- $\{\langle \mathcal{S}, \mathcal{A}, p^m, r^m, \alpha^m, \rho_o^m, \widehat{\kappa}^m \rangle\}$ is the collection of weakly coupled single-agent MDPs, as defined in the single-agent model in Section 2.2. For simplicity, but without loss of generality, we assume that all agents have the same state and actions spaces $\mathcal{S}$ and $\mathcal{A}$, but each has its own transition and reward functions p^m and r^m, initial conditions α^m, as well as its own resource requirements $\rho_o^m : \mathcal{A} \times \mathcal{O} \mapsto \{0, 1\}$ and capacity bounds $\widehat{\kappa}^m : \mathcal{C} \mapsto \mathbb{R}$.
- $\mathcal{O} = \{o\}, \mathcal{C} = \{c\}$ are the sets of resources and capacities, defined exactly as in the single-agent model in Section 2.2.
- $\kappa_o : \mathcal{O} \times \mathcal{C} \mapsto \mathbb{R}$ specifies the capacity costs of the resources, defined exactly as in the single-agent model in Section 2.2.
- $\widehat{\rho}_o : \mathcal{O} \mapsto \mathbb{R}$ specifies the upper bound on the amounts of the shared resources (this defines the additional bound for the multiagent problem).

Given the above, our goal is to design a mechanism for allocating the resources to the agents in an economically efficient way, i.e., in a way that maximizes the social welfare of the agents (one of the most often-used criteria in mechanism design). We would also like the mechanism to be efficient from the computational standpoint.

Example 3.1. *Suppose that there are two delivery agents. The MDP and capacity constraints of the first agent are exactly as defined in Examples 2.1 and 2.2. The MDP of the second agent is almost the same as that of the first agent, with the only difference that it gets a slightly higher reward for delivering appliances:* $r^2(s_1, a_2) = 12$ *(whereas* $r^1(s_1, a_2) = 10$ *for the first agent). Suppose there are two trucks, one forklift, and one mechanic that are shared by the two agents. These bounds are specified as follows:*

$$\widehat{\rho}_o(o_t) = 2, \ \widehat{\rho}_o(o_f) = 1, \ \widehat{\rho}_o(o_m) = 1.$$

Then the problem is to decide which agent should get the forklift, and which should get the mechanic (trucks are plentiful in this example). ■

3.1.1 Combinatorial Auctions

As previously mentioned, the problem of finding an optimal resource allocation among self-interested agents that have complex valuations over combinations of resources arises in many different domains (e.g., Ferguson, Nikolaou, Sairamesh, & Yemini, 1996; Wellman et al., 2001) and is often called a *combinatorial allocation problem*. A natural and widely used mechanism for solving such problems is a *combinatorial auction* (CA) (e.g., de Vries

are allocated, which would significantly complicate the strategic analysis of the agents' betting strategies during the initial resource allocation.

& Vohra, 2003). In a CA, each agent submits a set of bids for resource bundles to the auctioneer, who then decides what resources each agent will get and at what price.

Consider the problem of allocating among a set of agents $\mathcal{M}$ a set of indivisible resources $\mathcal{O}$, where the total quantity of resource $o \in \mathcal{O}$ is bounded by $\widehat{\rho_o}(o)$. Our earlier simplifying assumption that actions' resource requirements are binary implies that agents will be interested only in bundles that contain no more than one unit of a particular resource.

In a combinatorial auction, each agent $m \in \mathcal{M}$ submits a bid b_w^m (specifying how much the agent is willing to pay) for every bundle $w \in \mathcal{W}^m$ that has some value $u_w^m > 0$ to the agent. After collecting all the bids, the auctioneer solves the winner-determination problem (WDP), a solution to which prescribes how the resource bundles should be distributed among the agents and at what prices. If agent m wins bundle w at price q_w^m, its utility is $u_w^m - q_w^m$ (we are assuming risk-neutral agents with quasi-linear utility functions). Thus, the optimal bidding strategy of an agent depends on how the auctioneer allocates the resources and sets prices.

A Generalized Vickrey Auction (GVA) (MacKie-Mason & Varian, 1994), which is an extension of Vickrey-Clarke-Groves mechanisms (Vickrey, 1961; Clarke, 1971; Groves, 1973) to combinatorial auctions, allocates resources and sets prices as follows. Given the bids b_w^m of all agents, the auctioneer chooses an allocation that maximizes the sum of agents' bids. This problem is NP-complete (Rothkopf et al., 1998) (unless some severe restriction are placed on the admissible bids (Rothkopf et al., 1998; Sandholm & Suri, 2000)) and can be expressed as the following integer program, where the optimization variables $z_w^m = \{0, 1\}$ are indicator variables that show whether bundle w is assigned to agent m, and $n_{wo} = \{0, 1\}$ specifies whether bundle w contains o:[2]

$$\begin{aligned}
&\max \sum_{m\in\mathcal{M}} \sum_{w\in\mathcal{W}^m} z_w^m b_w^m \\
&\text{subject to:} \\
&\sum_{w\in\mathcal{W}^m} z_w^m \le 1, \qquad\qquad \forall m \in \mathcal{M}; \\
&\sum_{m\in\mathcal{M}} \sum_{w\in\mathcal{W}^m} z_w^m n_{wo} \le \widehat{\rho_o}(o), \qquad \forall o \in \mathcal{O}.
\end{aligned} \tag{3.1}$$

The first constraint in (3.1) says that no agent can receive more than one bundle, and the second constraint ensures that the total amount of resource o assigned to the agents does not exceed the total amount available.

A GVA assigns resources according to the optimal solution $\widetilde{z}$ to (3.1) and sets the payment for agent m to:

$$q_w^m = V_{-m}^* - \sum_{m'\neq m} \widetilde{z}_w^{m'} b_w^{m'}, \tag{3.2}$$

where V_{-m}^* is the value of (3.1) if m were to not participate in the auction (the optimal value if m does not submit any bids), and the second term is the sum of other agents' bids in the solution $\widetilde{z}$ to the WDP with m participating.

[2]There are other related algorithms for solving the WDP (e.g., Sandholm, 2002), but we will use the integer program (3.1) as a representative formulation for the class of algorithms that perform a search in the space of binary decisions on resource bundles.

A GVA has a number of nice properties. It is *strategy-proof*, meaning that the dominant strategy of every agent is to bid its true value for every bundle: $b^m_w = u^m_w$. The auction is economically efficient, meaning that it allocates the resources to maximize the social welfare of the agents (because, when agents bid their true values, the objective function of (3.1) becomes the social welfare). Finally, a GVA satisfies the *participation constraint*, meaning that no agent decreases its utility by participating in the auction.

A straightforward way to implement a GVA for our MDP-based problem is the following. Let each agent $m \in \mathcal{M}$ enumerate all possible resource bundles $\mathcal{W}^m$ that satisfy its local capacity constraints ($\widehat{\kappa}^m(c)$). For each bundle $w \in \mathcal{W}^m$, agent m would determine the feasible action set $\mathcal{A}(w)$ and formulate an MDP $\Lambda^m(w) = \langle \mathcal{S}, \mathcal{A}(w), p^m(w), r^m(w), \alpha^m \rangle$, where $p^m(w)$ and $r^m(w)$ are projections of the agent's transition and reward functions onto $\mathcal{A}(w)$. Every agent would then solve each $\Lambda^m(w)$ corresponding to a feasible bundle to find the optimal policy $\widetilde{\pi}^m(w)$, whose expected discounted reward would define the value of bundle w: $u^m_w = U^m_\gamma(\widetilde{\pi}^m(w), \alpha^m)$.

Then, the auction proceeds in the standard GVA manner. The agents submit their bids b^m_w to the auctioneer. Since this is just a special case of a GVA, the strategy-proof property implies that agents would not deviate from bidding their true values $b^m_w = u^m_w = U^m_\gamma(\widetilde{\pi}^m(w), \alpha^m)$. The auctioneer then solves the standard winner-determination problem (3.1) and sells the resources to the agents at prices (3.2). Again, by properties of the standard GVA, the mechanism will yield socially optimal resource allocations.

This mechanism suffers from two major complexity problems. First, the agents have to enumerate an exponential number of resource bundles and compute the value of each by solving the corresponding (possibly large) MDP. Second, the auctioneer has to solve an NP-complete winner-determination problem on exponential input. The following sections are devoted to tackling these complexity problems.

Example 3.2. *Consider the two-agent problem described in Example 3.1, where two trucks, one forklift, and the services of one mechanic are being auctioned off. Using the straightforward version of the GVA outlined above, each agent would have to consider* $2^{|\mathcal{O}|} = 2^3 = 8$ *possible resource bundles (since resource requirements of both agents are binary, neither agent is going to bid on a bundle that contains two trucks). For every resource bundle, each agent will have to formulate and solve the corresponding MDP to compute the utility of the bundle.*

For example, if we assume that both agents start in state s_1 *(different initial conditions would result in different expected rewards, and thus different utility functions), the value of the null resource bundle to both agents would be* 0 *(since the only action they would be able to execute is the noop* a_0*). On the other hand, the value of bundle* $[o_t, o_f, o_m] = [1, 1, 1]$ *that contains all the resources would be* 95.3 *to the first agent and* 112.4 *to the second one. The value of bundle* $[1, 1, 0]$ *to each agent would be the same as the value of* $[1, 1, 1]$ *(since their optimal policies for the initial conditions that put them in* s_1 *do not require the mechanic).*

Once the agents submit their bids to the auctioneer, it will have to solve the WDP via the integer program (3.1) with $|\mathcal{M}|2^{|\mathcal{O}|} = 2(2)^3 = 16$ *binary variables. Given the above, the optimal way to allocate the resources would be to assign a truck to each of the agents, the forklift to the second agent, and the mechanic to either (or neither) of the two. Thus, the agents would receive bundles* $[1, 0, 0]$ *and* $[1, 1, 0]$*, respectively, resulting in social welfare of* $50 + 112.4 = 162.4$*. If however, at least one of the agents had a non-zero probability of* *starting in state s_3, the value of the resource bundles involving the mechanic would change drastically, as would the optimal resource allocation and its social value.* ■

3.1.2 Avoiding Bundle Enumeration

A trivial way to simplify the agents' computational problems is by creating an auction where they submit the specifications of their constrained MDPs to the auctioneer as bids. This is an instance of a *direct revelation mechanism* (Myerson, 1979), where an agent's *type* that it submits to the auctioneer is defined by its constrained MDPs.[3] Clearly, this moves the burden of solving the valuation problem from the agents to the auctioneer, and, by itself, does not lead to any efficiency gains. Such a mechanism also has implications on information privacy issues, because the agents have to reveal their local MDPs to the auctioneer (which they might not want to do). Nevertheless, we can build on this idea to increase the efficiency of solving both the valuation and winner-determination problems and do so without sacrificing agents' private MDP information. We address ways of maintaining information privacy in the next section, and for the moment focus on improving the computational complexity of the agents' valuation and the auctioneer's winner-determination problems.

The question we pose in this section is as follows. Given that each agent bids its MDP, its resource information, and its capacity bounds $\langle \mathcal{S}, \mathcal{A}, p^m, r^m, \alpha^m, \rho_o^m, \widehat{\kappa}^m \rangle$, can the auctioneer formulate and solve the winner-determination problem more efficiently than by simply enumerating each agent's resource bundles and solving the standard integer program (3.1) with an exponential number of binary variables?

Therefore, the goal of the auctioneer is to find a joint policy (a collection of single-agent policies under our weak-coupling assumption) that maximizes the sum of the expected total discounted rewards for all agents, under the conditions that: i) no agent m is assigned a set of resources that violate its capacity bound $\widehat{\kappa}^m$ (i.e., no agent is assigned more resources than it can carry), and ii) the total amounts of resources assigned to all agents do not exceed the global resource bounds $\widehat{\rho_o}(o)$ (i.e., we cannot allocate to the agents more resources than are available). This problem can be expressed as the following mathematical program:

$$\begin{aligned}
&\max \sum_m U_\gamma^m(\pi^m, \alpha^m) \\
&\text{subject to:} \\
&\sum_o \kappa(o,c) H\big(\rho_o^m(a,o) \sum_s \pi^m(s,a)\big) \le \widehat{\kappa}^m(c), \qquad \forall c \in \mathcal{C}, m \in \mathcal{M}; \\
&\sum_m \rho_o^m(a,o) H\big(\sum_s \pi^m(s,a)\big) \le \widehat{\rho_o}(o), \qquad \forall o \in \mathcal{O}.
\end{aligned} \tag{3.3}$$

Obviously, the decision version of this problem is NP-complete, as it subsumes the single-agent MDP with capacity constraints, NP-completeness of which was shown in Section 2.3. Moreover, the problem remains NP-complete even in the absence of single-agent capacity constraints. Indeed, the global constraint on the amounts of the shared resources is sufficient to make the problem NP-complete, which can be shown with a

[3]Submitting a full utility function as in the previous section is also a direct revelation mechanism, where an agent's type is defined by its utility function.

straightforward reduction from KNAPSACK, similar to the one used in the single-agent case in Section 2.3.

We can linearize (3.3), similarly to the single-agent problem from Section 2.4, yielding the following MILP, which is simply a multiagent version of (2.23) (recall the assumption of this section that resource requirements are binary):

$$
\begin{aligned}
&\max \sum_{m,s,a} x^m(s,a) r^m(s,a) \\
&\text{subject to:} \\
&\sum_a x^m(\sigma,a) - \gamma \sum_{s,a} x^m(s,a) p^m(\sigma|s,a) = \alpha^m(\sigma), && \forall \sigma \in \mathcal{S}, m \in \mathcal{M}; \\
&\sum_o \kappa(o,c)\delta^m(o) \le \widehat{\kappa}^m(c), && \forall c \in \mathcal{C}, m \in \mathcal{M}; \\
&\sum_m \delta^m(o) \le \widehat{\rho_o}(o), && \forall o \in \mathcal{O}; \\
&1/X \sum_a \rho_o^m(a,o) \sum_s x^m(s,a) \le \delta^m(o), && \forall o \in \mathcal{O}, m \in \mathcal{M}; \\
&x^m(s,a) \ge 0, && \forall s \in \mathcal{S}, a \in \mathcal{A}, m \in \mathcal{M}; \\
&\delta^m(o) \in \{0,1\}, && \forall o \in \mathcal{O}, m \in \mathcal{M},
\end{aligned}
\tag{3.4}
$$

where $X \ge \max \sum_a \rho_o(a,o) \sum_s x(s,a)$ is an upper bound on the argument of $H(\cdot)$, used for normalization. As in the single-agent case, this bound is guaranteed to exist for discounted MDPs and can be obtained in polynomial time.

The MILP (3.4) allows the auctioneer to efficiently solve the WDP without having to enumerate the possible resource bundles. As compared to the standard WDP formulation (3.1), which has on the order of $|\mathcal{M}|2^{|\mathcal{O}|}$ binary variables, (3.4) has only $|\mathcal{M}||\mathcal{O}|$ binary variables. This exponential reduction is attained by exploiting the knowledge of the agents' MDP-based valuations and simultaneously solving the policy-optimization and resource-allocation problems. Given that the worst-case solution time for MILPs is exponential in the number of integer variables, this reduction has a significant impact on the worst-case performance of the algorithm. The average-case running time is also reduced drastically, as demonstrated by our experiments, presented in Section 3.2.

Example 3.3. *If we apply the algorithm discussed above to our running example as an alternative to the straightforward GVA presented in Example 3.2, the winner-determination MILP (3.4) will look as follows. It will have* $|\mathcal{M}||\mathcal{S}||\mathcal{A}| = (2)(3)(5) = 30$ *continuous occupation-measure variables* x^m*, and* $|\mathcal{M}||\mathcal{O}| = (2)(3) = 6$ *binary variables* $\delta^m(o)$*. It will have* $|\mathcal{M}||\mathcal{S}| = (2)(3) = 6$ *conservation-of-flow constraints that involve continuous variables only, as well as* $|\mathcal{M}||\mathcal{C}| + |\mathcal{O}| + |\mathcal{M}||\mathcal{O}| = (2)(1) + 3 + (2)(3) = 9$ *constraints that involve binary variables.*

The capacity constraints for the agents will be exactly as in the single-agent case described in Example 2.8, and the global resource constraints will be:

$$\delta^1(o_t) + \delta^2(o_t) \le 2, \qquad \delta^1(o_f) + \delta^2(o_f) \le 1, \qquad \delta^1(o_m) + \delta^2(o_m) \le 1.$$

Notice that in this example there is one binary decision variable per resource per agent (yielding 6 *such variables for this simple problem). This is exponentially fewer than the* *number of binary variables in the straightforward GVA formulation of Example 3.2, which requires one binary variable per resource bundle per agent (yielding* 16 *such variables for this problem). Given that MILPs are NP-complete in the number of integer variables, this reduction from* 16 *to* 6 *variables is noticeable even in a small problem like this one and can lead to drastic speedup for larger domains.* ■

The mechanism described above is an instantiation of the GVA, so by the well-known properties of VCG mechanisms, this auction is strategy-proof (the agents have no incentive to lie to the auctioneer about their MDPs), it attains the socially optimal resource allocation, and no agent decreases its utility by participating in the auction.

To sum up the results of this section: by having the agents submit their MDP information to the auctioneer instead of their valuations over resource bundles, we have essentially removed all computational burden from the agents and at the same time significantly simplified the auctioneer's winner-determination problem (the number of integer optimization variables is reduced exponentially).

3.1.3 Distributing the Winner-Determination Problem

Unlike the straightforward GVA implementation discussed earlier in Section 3.1.1, where the agents shared some computational burden with the auctioneer, in the mechanism from Section 3.1.2, the agents submit their information to the auctioneer and then just idle while waiting for a solution. This suggests further potential efficiency improvements. Indeed, given the complexity of MILPs, it would be beneficial to exploit the computational power of the agents to offload some of the computation from the auctioneer back to the agents (we assume that agents have no cost for "helping out" and would prefer for the outcome to be computed faster).[4] Thus, the goal of this section is to distribute the computation of the winner-determination problem (3.4).

For concreteness, we will base the algorithm of this section on the branch and bound method for solving MILPs (Wolsey, 1998), but exactly the same techniques will also work for other MILP algorithms (e.g., cutting planes) that perform a search in the space of *LP relaxations* of the MILP. In branch and bound for MILPs with binary variables, LP relaxations are created by choosing a binary variable and setting it to either 0 or 1, and relaxing the integrality constraints of other binary variables. If a solution to an LP relaxation happens to be integer-valued, it provides a lower bound on the value of the global solution. A non-integer solution provides an upper bound for the current subproblem, which (combined with other lower bounds) is used to prune the search space.

Thus, a simple way for the auctioneer to distribute the branch and bound algorithm is to simply farm out LP relaxations to other agents and ask them to solve the LPs. However, it is easy to see that this mechanism is not strategy-proof. Indeed, an agent that is tasked with solving parts of the global winner-determination problem could benefit from lying to the auctioneer, i.e., the agent might be better off optimizing a function that differs from the social welfare. This is a common phenomenon in distributed mechanism implementations (e.g., Parkes & Shneidman, 2004): whenever some WDP calculations are

[4]As observed by Parkes and Shneidman (2004), this assumption is a bit controversial, since a desire for efficient computation implies nonzero cost for computation, while the agents' cost for "helping out" is not modeled. It is, nonetheless, a common assumption in distributed mechanism implementations.

offloaded to an agent participating in the auction, the agent might be able to benefit from sabotaging the computation. As suggested by Parkes and Shneidman (2004), there are several approaches to ensuring the strategy-proofness of a distributed implementation. The approach best suited for our problem is based on the idea of *redundant computation*, where multiple agents are asked to do the same task and any disagreement is carefully punished to discourage lying. In the rest of this section, we demonstrate that this is very easy to implement in our case.

The basic idea is very simple: let the auctioneer distribute the LP relaxations to the agents, but check their solutions and re-solve the problems if the agents return incorrect solutions (this makes truthful computation a weakly dominant strategy). This strategy of the auctioneer removes the incentive for the agents to lie and yields exactly the same solution as the centralized algorithm. However, in order for this to be beneficial, the complexity of checking a solution must be significantly lower than the complexity of solving the problem. Fortunately, this is true for LPs.

Suppose the auctioneer has to solve the following LP, which can be written in two equivalent ways (let us refer to the one on the left as the primal, and to the one on the right as the dual):

$$\min \boldsymbol{\alpha}^T \mathbf{v} \,\Big|\, A^T\mathbf{v} \geq \mathbf{r}\ , \quad \max \mathbf{r}^T\mathbf{x} \left| \begin{array}{l} A\mathbf{x} = \boldsymbol{\alpha} \\ \mathbf{x} \geq 0 \end{array} \right. \tag{3.5}$$

By the strong duality property, if the primal LP has a solution $\mathbf{v}^*$, then the dual also has a solution $\mathbf{x}^*$, and $\boldsymbol{\alpha}^T\mathbf{v}^* = \mathbf{r}^T\mathbf{x}^*$. Furthermore, given a solution to the primal LP, it is easy to compute a solution to the dual: by complimentary slackness, $\mathbf{v}^{*T} = \mathbf{r}^T B^{-1}$ and $\mathbf{x}^* = B^{-1}\boldsymbol{\alpha}$, where B is a square invertible matrix, composed of columns of A that correspond to basic variables of the solution.

These well-known properties can be used by the auctioneer to quickly check optimality of solutions returned by the agents. Suppose that an agent returns $\mathbf{v}$ as a solution to the primal LP. The auctioneer can calculate the dual solution $\mathbf{v}^T = \mathbf{r}^T B^{-1}$ and check whether $\mathbf{r}^T\mathbf{x} = \boldsymbol{\alpha}^T\mathbf{v}$. Thus, the most expensive operation that the auctioneer has to perform is the inversion of B, which can be done in sub-cubic time. As a matter of fact, from the implementation perspective, it would be more efficient to ask the agents to return both the primal and the dual solutions, since many popular algorithms compute both in the process of solving LPs.

Thus, we have provided a simple method that allows us to effectively distribute the winner-determination problem, while maintaining strategy-proofness of the mechanism and with a negligible computation overhead for the auctioneer.

3.1.4 Preserving Information Privacy

The mechanism that we have discussed so far has the drawback that it requires agents to reveal complete information about their MDPs to the auctioneer. The problem is also exacerbated by the distributed WDP algorithm from the previous section, since not only does each agent reveal its MDP information to the auctioneer, but that information is then also spread to other agents via the LP relaxations of the global MILP. We now show how to alleviate this problem.

Let us note that, in saying that agents prefer not to reveal their local information, we are implicitly assuming that there is an external factor that affects agents' utilities, and it depends on how much information is revealed. A sensible way to measure the value of information is by how it changes one's decision-making process and its outcomes. Since this effect is not part of our model (in fact, it contradicts our weak-coupling assumption), we cannot in a domain-independent manner define what constitutes "useful" information, and how bad it is for an agent to reveal too much about its MDP. Modeling such effects and carefully analyzing them is an interesting research task, but it is outside the scope of this thesis. Thus, for the purposes of this section, we will be content with a mechanism that hides enough information to make it impossible for the auctioneer or an agent to uniquely determine the transition or reward function of any other agent (in fact, information revealed to any agent will map to infinitely many significantly different MDPs of other agents).[5] Many such transformations are possible; we present just one to illustrate the concept.

The main idea of our approach is to modify the previous mechanism so that the agents submit their private information to the auctioneer in an "encrypted" form that will allow the auctioneer to solve the winner-determination problem, but will not allow it to infer the agents' original MDPs.

First, note that, instead of passing an MDP to the auctioneer, each agent can submit an equivalent LP (2.10). So, the question becomes: can the agent transform its LP in such a way that the auctioneer will be able to solve it, but will not be able to infer the transition and reward functions of the originating MDP? In other words, the problem reduces to the following. Given an LP $\mathcal{L}_1$ (created from an MDP $\Lambda = \langle \mathcal{S}, \mathcal{A}, p, r, \alpha \rangle$ via (2.10)), we need to find a transformation $\mathcal{L}_1 \rightarrow \mathcal{L}_2$ such that a solution to the transformed LP $\mathcal{L}_2$ will uniquely map to a solution to the original LP $\mathcal{L}_1$, but $\mathcal{L}_2$ will not reveal the transition or the reward functions of the original MDP (p or r). We show that a simple change of variables suffices.

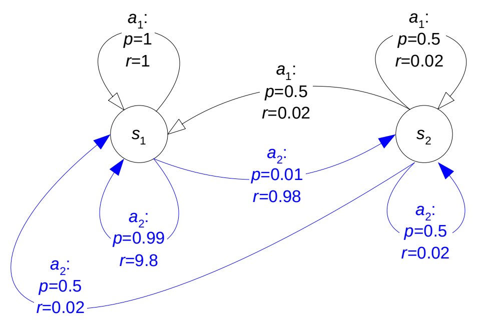
Suppose agent m_1 has an MDP-originated LP and is going to ask agent m_2 to solve it. In order to maintain the linearity of the problem (to keep it simple for m_2 to solve), m_1 should limit itself to linear transformations. Consider a linear, invertible transformation of the primal coordinates $\mathbf{u} = F\mathbf{v}$, and a linear, invertible transformation of the dual coordinates $\mathbf{y} = D\mathbf{x}$. Then, the LP from (3.5) will be transformed (by applying F, switching to the dual, and then applying D) to an equivalent LP in the new coordinates $\mathbf{y}$:

$$\max \mathbf{r}^T D^{-1}\mathbf{y} \left| \begin{array}{l} (F^{-1})^T A D^{-1}\mathbf{y} = (F^{-1})^T \boldsymbol{\alpha} \\ D^{-1}\mathbf{y} \geq 0 \end{array} \right. \tag{3.6}$$

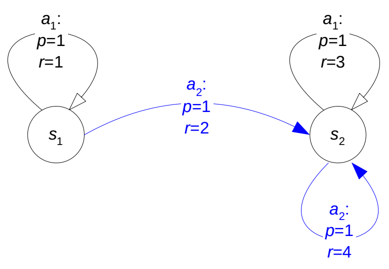
The value of the optimal solution to (3.6) will be the same as the value of the optimal solution to (3.5), and given an optimal solution $\mathbf{y}^*$ to (3.6), it is easy to compute the solution to the original: $\mathbf{x}^* = D^{-1}\mathbf{y}^*$. Indeed, from the perspective of the dual, the primal transformation F is equivalent to a linear transformation of the dual equality constraints $A\mathbf{x} = \alpha$, which (given that F is non-singular) has no effect on the solution or the objective function. Furthermore, the dual transformation D is equivalent to a change of variables that modifies the solution but not the value of the objective function.

However, a problem with the above transformations is that it gives away D^{-1}. Indeed, agent m_2 will be able to simply read (up to a set of multiplicative constants) the transformation off the constraints $D^{-1}\mathbf{y} \geq 0$. Therefore, only diagonal matrices with positive coefficients (which are equivalent to stretching the coordinate system) are not trivially deduced by m_2, since they also map to $\mathbf{y} \geq 0$.

[5] A more stringent condition would require that agents' preferences over resource bundles are not revealed (Parkes, 2001), but we set a lower bar here.



(a)



(b)

Figure 3.1: Preserving information privacy example. Two different MDPs that can lead to the same LP constraint matrix.

Let us demonstrate that, given any MDP Λ and the corresponding LP $\mathcal{L}_1$, we can choose D and F such that it will be impossible for m_2 to determine the coefficients of $\mathcal{L}_1$ (or equivalently the original transition and reward functions p and r). When agent m_2 receives $\mathcal{L}_2$ (as in (3.6)), all it knows is that $\mathcal{L}_2$ was created from an MDP, so the columns of the constraint matrix of the original LP $\mathcal{L}_1$ must sum to a constant:

$$\sum_j A_{ji} = 1 - \gamma \sum_\sigma p(\sigma|s,a) = 1 - \gamma. \tag{3.7}$$

This gives m_2 a system of $|\mathcal{S}|$ nonlinear equations for the diagonal D and arbitrary F, which have a total of $|\mathcal{S}||\mathcal{A}| + |\mathcal{S}|^2$ free parameters. For everything but the most degenerate cases (which can be easily handled by an appropriate choice of D and F), these equations are hugely under-constrained and will have infinitely many solutions. As a matter of fact, by sacrificing $|\mathcal{S}|$ of the free parameters, m_1 can choose D and F in such a way that the columns of constraints in $\mathcal{L}_2$ will also sum to a constant $c \in (0,1)$, which would have the effect of transforming $\mathcal{L}_1$ to an $\mathcal{L}_2$ that corresponds to another valid MDP Λ_2. Therefore, given an $\mathcal{L}_2$ there are infinitely many original MDPs Λ and transformations D, F that map to the same LP $\mathcal{L}_2$.

We also have to consider the connection of resource and capacity costs to agents' occupation measures in the global WDP (3.4). There are essentially two things that the auctioneer has to be able to do: i) determine the value of each agent's policy (to be able to maximize the social welfare), and ii) determine the resource requirements of the policies (to check the resource constraints). So, the question is, how does our transformation affect these? As noted earlier, the transformation does not change the objective function, so the first requirement holds. On the other hand, D does change the occupation measure $x^m(s,a)$ by arbitrary multipliers. However, a multiplicative factor of $x^m(s,a)$ has no effect on the usage of non-consumable resources, as it only matters whether the corresponding $x^m(s,a)$ is zero or not (step function H nullifies the scaling effect). Thus, the second condition also holds.

Example 3.4. *Consider the two-state MDP depicted in Figure 3.1a that represents the decision-making problem of a sales company, with the two states corresponding to possible market conditions, and the two actions corresponding to two possible sales strategies.*

Market conditions in state s_1 are much more favorable than in state s_2 (the rewards for both actions are higher). The transitions between the two states correspond to probabilities of market conditions changing and the rewards reflect expected profitability in these two states. Obtaining such numbers in a realistic scenario would require performing costly and time-consuming research, and the company might not want to make this information public.

Therefore, if the company were to participate in the resource-allocation mechanism described above, it would want to encrypt its MDP before submitting it to the auctioneer in order to hide its private estimate of the market conditions.

The MDP has the following reward function

$$r = (1,\ 19.622,\ 0.063,\ 0.084)^T, \tag{3.8}$$

and the following transition function:

$$p(a_1) = \begin{pmatrix} 1 & 0 \\ 0.5 & 0.5 \end{pmatrix}, \quad p(a_2) = \begin{pmatrix} 0.986 & 0.014 \\ 0.5 & 0.5 \end{pmatrix}. \tag{3.9}$$

Taking $\gamma = 0.8$, this corresponds to the following conservation of flow constraint matrix:

$$A = \begin{pmatrix} 0.2 & 0.212 & -0.4 & -0.4 \\ 0 & -0.012 & 0.6 & 0.6 \end{pmatrix}. \tag{3.10}$$

Before submitting its LP to the auctioneer, the agent applies the following transformations:

$$D = diag(1,\ 0.102,\ 47.619,\ 47.619), \quad F = \begin{pmatrix} 2 & 0 \\ -0.084 & 0.126 \end{pmatrix}, \tag{3.11}$$

yielding the following new constraint matrix:

$$A' = (F^{-1})^T A D^{-1} = \begin{pmatrix} 0.1 & 1 & 0 & 0 \\ 0 & -0.9 & 0.1 & 0.1 \end{pmatrix}. \tag{3.12}$$

However, the above constraint matrix A' corresponds to a non-transformed conservation of flow constraint for a different MDP (shown in Figure 3.1b) with $\gamma = 0.9$, the following reward function:

$$r = (1,\ 2,\ 3,\ 4)^T, \tag{3.13}$$

and the following transition function:

$$p(a_1) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad p(a_2) = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}. \tag{3.14}$$

Therefore, when the auctioneer receives the constraint matrix A', it has no way of knowing whether the agent has an MDP with transition function (3.14) that was transformed using (3.11) or the MDP with transition function (3.9) that was not transformed. Notice that the dynamics of the two MDPs vary significantly: both in transition probabilities and state connectivity. The second MDP does not reveal any information about the originating MDP and the corresponding market dynamics. In particular, it does not reveal the (proprietary) knowledge that the market conditions in s_1 are more favorable to profitability. ■

To sum up, we can, to a large extent, maintain information privacy in our mechanism by allowing agents to apply linear transformations to their original LPs. The information that is revealed by our mechanism consists of agents' resource costs $\rho_o^m(a, o)$, capacity bounds $\widehat{\kappa}^m(c)$, and the sizes of their state and action spaces (the latter can be hidden by adding dummy states and actions to the MDP).

The revealed information can be used to infer agents' preferences and resource requirements. Further, numeric policies are revealed, but the lack of information about transition and reward functions renders this information worthless (as just illustrated in Example 3.4, there could be multiple originating MDPs with very different properties).

3.2 Experimental Evaluation

In this section we present an empirical analysis of the computational efficiency of the resource-allocation mechanism described in Section 3.1. We report results on the computational efficiency of the mechanism from Section 3.1.2, where the agents submit their MDPs to the auctioneer, who then simultaneously solves the resource-allocation and policy-optimization problems. Regarding the additional speedup achieved by distributing the WDP, as described in Section 3.1.3, we do not report empirical results, since it is well-established in the parallel programming literature that parallel versions of branch-and-bound MILP solvers consistently achieve linear speedup (Eckstein, Phillips, & Hart, 2000). This is due to the fact that branch-and-bound algorithms require very little inter-process communication.

For our experiments, we implemented a multiagent delivery problem (based on the multiagent rover domain in (Dolgov & Durfee, 2004c)), where agents operate in a stochastic grid world with delivery locations randomly placed throughout the grid. Each delivery task requires a set of resources, and there are limited quantities of the resources. There are random delivery locations on the grid, and each location has a set of deliveries that it accepts. Each resource takes up some space (capacity cost), and each delivery agent has bounded space to hold the resources (limited capacity). The agents participate in an auction where they bid on delivery resources. In this setting, the value of a resource depends on what other resources the agent acquires and what other deliveries it can make. Given a bundle of resources, an agent's policy optimization problem is to find the optimal delivery plan. The exact parameters used in our experiments are not critical for the trends seen in the results presented below, but for the sake of reproducibility the domain is described below.[6]

3.2.1 Experimental Setup

For a delivery domain with $|\mathcal{M}|$ agents operating on an n-by-n grid and sharing $|\mathcal{O}|$ resource types, we used the following parameters.

The resources enable agents to carry out delivery tasks. For a problem with $|\mathcal{O}|$ resource types, there are $|\mathcal{O}|$ delivery actions, and performing action $i \in [1, |\mathcal{O}|]$ requires a random subset of resources from $\mathcal{O}$ (where the number of resources required by an action is an important parameter, whose effect on complexity is discussed in Section 3.2). The

[6]We also investigated other, randomly generated domains, and the results were qualitatively the same.

probability that task $i \in [1, |\mathcal{O}|]$ can be carried out at a location is $0.1+0.4(|\mathcal{O}|-i)/(|\mathcal{O}|-1)$, i.e., uniformly distributed between 0.1 and 0.5, as a function of the action ID i (actions with lower IDs are more rewarding, per the definition of the reward function below, but can be executed at fewer locations).

There are $n^2/5$ possible delivery locations randomly placed on the grid. Each delivery location is assigned a set of delivery tasks that can be executed there (a single location can be used for multiple delivery tasks, and a single task can be carried out at any of several locations). The assignment of tasks to locations is done randomly.

Each agent has $4 + |\mathcal{O}|$ actions: drive in any of the four perpendicular directions and execute one of the delivery tasks. The drive actions result in movement in the intended direction with probability of 0.8 and with probability of 0.2 produce no change of location. All movement actions incur a negative reward, the amount of which depends on the size of the agent. For a problem with $|\mathcal{M}|$ agents, the movement penalty incurred by agent $m \in [1, |\mathcal{M}|]$ is $-1 - 9(m-1)/(|\mathcal{M}|-1)$, i.e., distributed uniformly on $[-1, -10]$ as a function of the agent's ID.

Execution of an action corresponding to a delivery task $i \in [1, |\mathcal{O}|]$ in a location to which the task is assigned produces a reward of $100i/|\mathcal{O}|$ and moves the agent to a new random location on the grid. The new location is chosen randomly at problem generation (thus known to agent), but the transition is deterministic, which induces a topology with nearby and remote locations. Attempting execution of a delivery task in an incorrect location does not change state and produces zero reward.

The agents bid for delivery resources of $|\mathcal{O}|$ types. There are $c_{glob}|\mathcal{M}|$ units of each resource, where c_{glob} is the global constraint level (set to 0.5 for most of our experiments, as described in more detail in Section 3.2.2). There is one capacity type: size. The size requirements for making deliveries of type $i \in [1, |\mathcal{O}|]$ is i. The capacity limit of agent m is $1/2c_{loc}|\mathcal{O}|(|\mathcal{O}|+1)$, where c_{loc} is the local constraint level (set to 0.5 for most of our experiments, as was described in more detail in Section 3.2).

The initial location of each agent is randomly selected from a uniform distribution. The discount factor is $\gamma = 0.95$.

3.2.2 Experimental Results

Computational complexity of constrained optimization problems can vary greatly as constraints are tightened or relaxed. Therefore, as our first step in the analysis of empirical computational complexity of our mechanism, we investigate how its running time depends on the capacity constraints of each agent and on the bounds on the total amounts of resources shared by the agents. As is common with other types of constrained optimization or constraint-satisfaction problems, it is natural to expect that the winner-determination MILP will be easy to solve when the problem is over- or under-constrained in either the capacity or the resource bounds. To empirically verify this, we varied local capacity constraint levels from 0 (meaning agents cannot use any resources) to 1 (meaning each agent has the capacity to use enough resources to execute its optimal unconstrained policy), as well as the global constraint levels for which 0 meant that no resources were available to the agents, and 1 meant that there were enough resources to assign to each agent its most desired resource bundle. In all of our experiments the part of the MILP solver was played by CPLEX 8.1 on a P-4. A typical running-time profile is shown in Figure 3.2. The

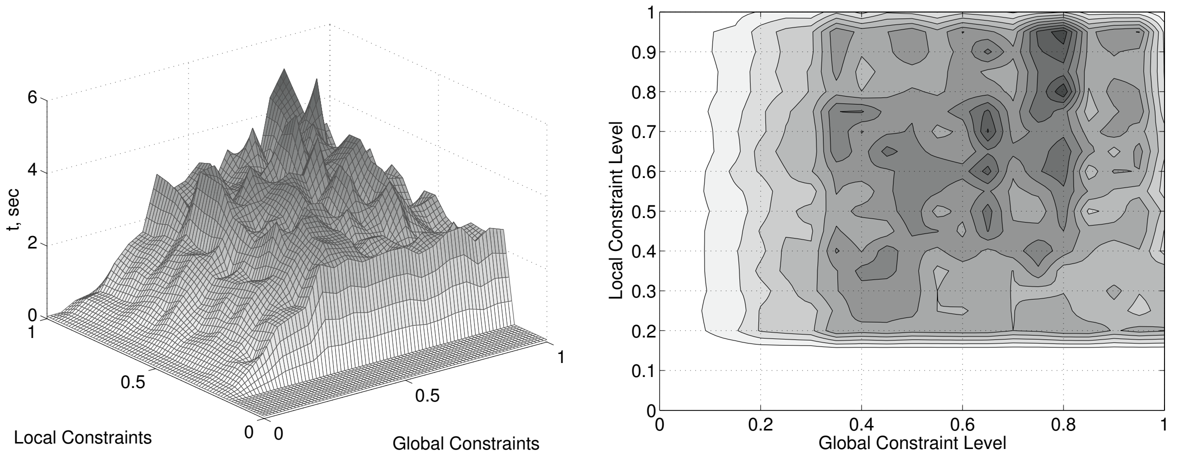


Figure 3.2: Running time for MDP-based WDP for different constraint levels of global ($\widehat{\rho_o}$) and local ($\widehat{\kappa}^m$) constraints. The constraint levels are fractions of utopian levels that are needed to implement optimal unconstrained policies. Problems involved 10 agents, each operating on a 5-by-5 grid, with 10 shared resource types. Each data point shown is an average of 10 runs over randomly generated problems.

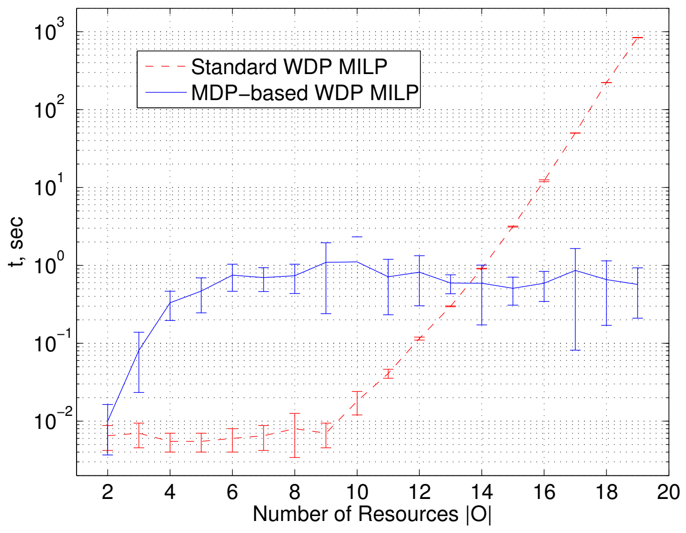


Figure 3.3: Comparison of the running time of MDP-based and flat combinatorial auctions. The latter does not include the time for solving the MDPs to compute resource-bundle values. Error bars show the standard deviation over ten runs.

problem is very easy when it is over-constrained, becomes more difficult as the constraints are relaxed and then abruptly becomes easy again when capacity and resource levels start to approach utopia.

In all of the following experiments we aim to avoid the easy regions of constraint levels. Therefore, given the complexity profiles, we set the constraint levels to 0.5 for both local capacity and global resource bounds.

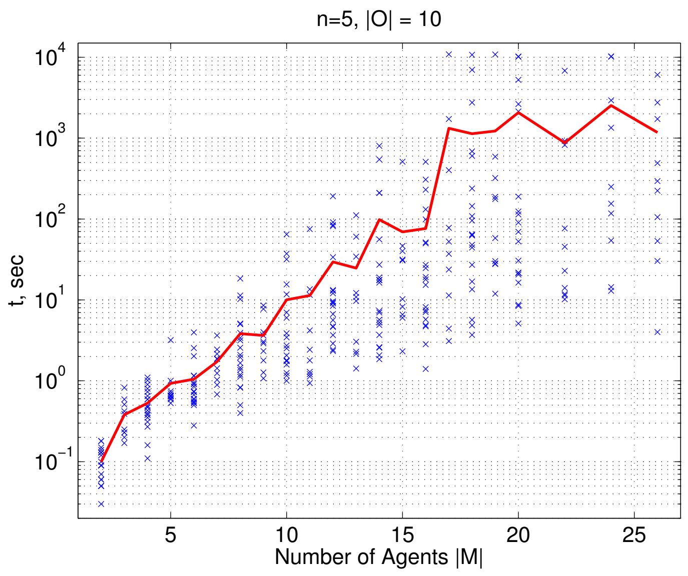
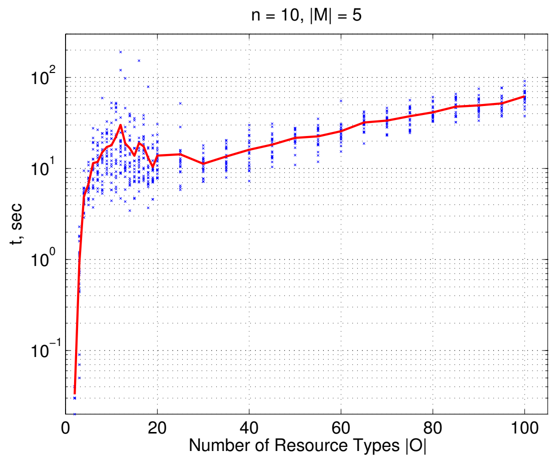


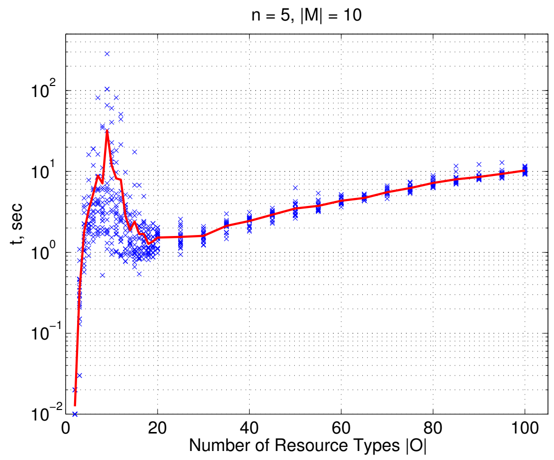
Figure 3.4: Scaling the MDP-based winner-determination MILP to more agents. Agents operated on 5-by-5 grids and shared 10 types of resources.

We begin by comparing the performance of our MDP-based auction (Section 3.1.2) to the performance of the standard CA with flat preferences. The results are summarized in Figure 3.3, which compares the time it takes to solve the standard winner-determination problem on the space of all resource bundles (3.1) to the time needed to solve the combined MDP-WDP problem (3.4) used in our mechanism, as the number of resources is increased (with 5 agents, on a 5-by-5 grid). Despite the fact that both algorithms have exponential worst-case running time, the number of integer variables in (3.1) is exponentially larger than in our MILP (3.4), the effect of which is clearly demonstrated in Figure 3.3. Furthermore, this comparison gives an extremely optimistic view of the performance of the standard CA, as it does not take into account the additional complexity of the valuation problem, which requires formulating and solving a very large number of MDPs (one per resource bundle). On the other hand, the latter is embedded into the WDP of our mechanism (3.4), thus including the time for solving the valuation problem in the comparison would only magnify the effect. In fact, in our experiments, the time required to solve the MDPs for the valuation problem was significantly greater than the time for solving the resulting winner-determination MILP. However, we do not present quantitative results to that effect here because of the difference in implementation (iterating over resource bundles and solving MDPs was done via a straightforward implementation in Matlab, while MILPs were solved using highly optimized CPLEX code). No parallelization of the WDP was performed for these experiments for either algorithm.

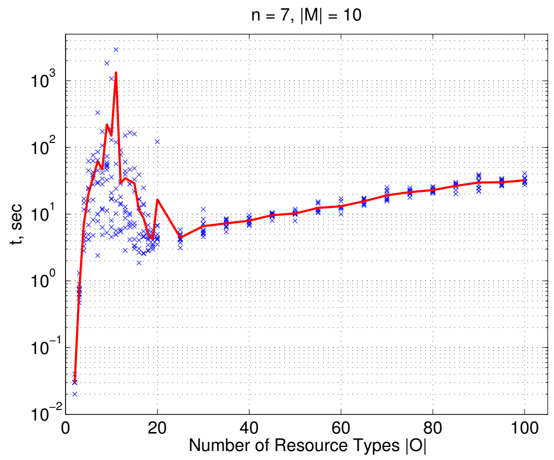
Below we analyze the performance of our algorithm on larger problems infeasible for the straightforward CA. Figure 3.4 illustrates the scaling effect as the number of agents participating in the auction is increased. Here and below, each point on the plot corresponds to a single run of the experiment (with no less than ten runs performed for every value of parameters), and the solid line is the mean. Recall that the size of the WDP scales linearly with the number of agents. The graph therefore reflects a rather standard



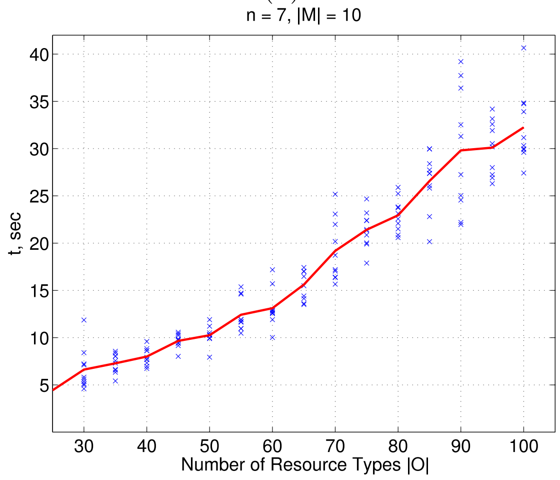
(a)



(b)



(c)



(d)

Figure 3.5: Scaling of the MDP-based winner-determination MILP with the number of resource types on three sets of problems with different grid sizes (n) and different numbers of agents ($|\mathcal{M}|$) is shown in (a,b,c); (d) shows a linear-scale plot of the tail of the data in (c).

scaling effect for an NP-complete problem. As can be seen from the plot, problems with 25 agents and 10 resource types are well within the reach of the method, on average taking around 30 minutes.

Next, we analyze how the method scales with the number of resource types. Figure 3.5 shows the solution time as a function of the number of resource types for three different sets of problems. In these problems, the number of actions scaled linearly with the number of resource types, but each action required a constant number of resources, i.e., the number of nonzero $\rho_o(a, o)$ per action was constant (two) regardless of the total number of resource types. These problems exhibit an interesting trait wherein the running time peaks for relatively low numbers of resource types then falls quickly, and then increases much more slowly as the number of resource types increases (as illustrated in Figure 3.5d, which uses a linear scale). This is due to the fact that when the total number of resource types is

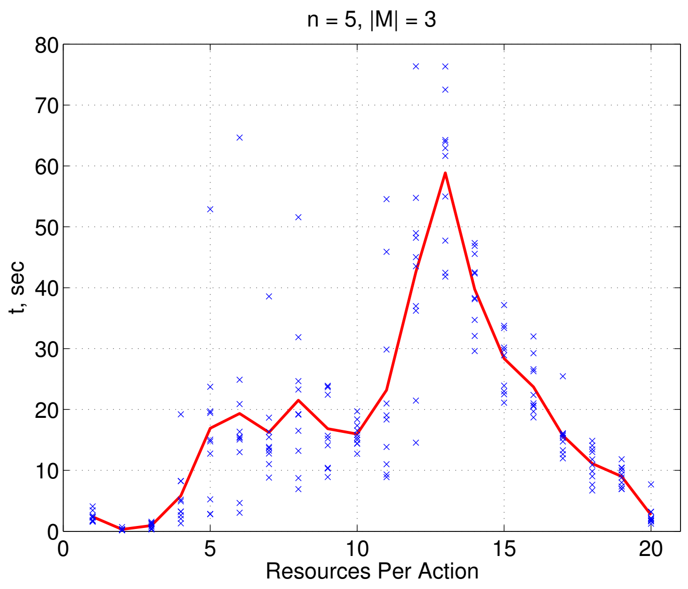


Figure 3.6: Complexity of MDP-based winner-determination MILP as a function of complexity of actions' resource requirements.

much higher than the number of resources required by any action, there is less contention for a particular resource among the agents and between one agent's actions. Therefore, the problems become relatively under-constrained and the solution time increases slowly.

To better illustrate this effect, we ran a set of experiments complementary to the ones shown in Figure 3.5: we kept the total number of resource types constant and increased the number of resource types required by each action. The results are shown in Figure 3.6. The running-time profile is similar to what we observed earlier when we varied the local and global constraints: when the total number of resources per action is low or high, the problem is under- or over-constrained and is relatively easy to solve, but its complexity increases significantly when the number of resources required by each action is in the range of 50-80% of the total number of resource types.

Based on the above, we would expect that if the actions' resource requirements increased with the total number of resource types, the problem would not scale as gracefully as in Figure 3.5. For example, Figure 3.7 illustrates the running time for problems where the number of resources required by each action scales linearly with the total number of resources. There, the complexity does increase significantly faster. However, it is not unreasonable to assume that in many domains the number of actions does not, in fact, increase with the total number of resource types involved. Indeed, it is natural to assume that the total number of resource types increases as the problem becomes more complicated and the number of tasks the agent can perform increases. However, why should the resource requirements of an action increase as well? If the delivery agent from our running example acquires the ability to deliver pizza, it might need new resources to perform actions related to this new activity, but one would not expect the resource requirements for delivering furniture or appliances to change. Therefore, we believe that in many real applications, our method will scale up gracefully with the total number of resource types.

The above experiments illustrate the point that for domains where agents have

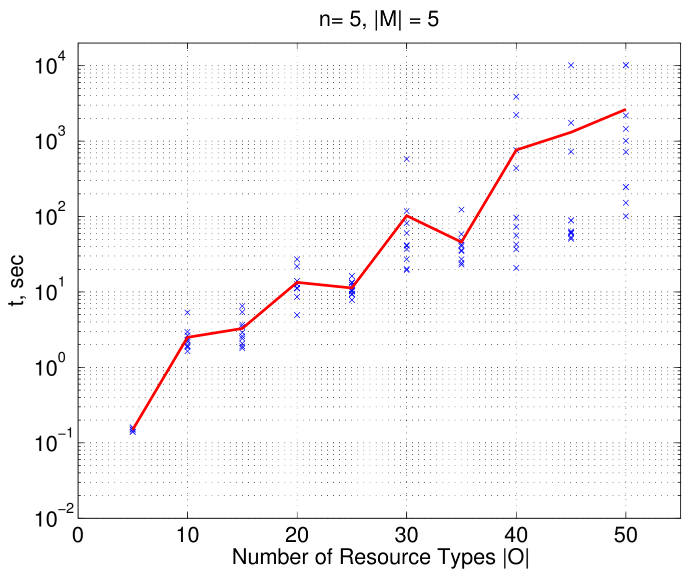


Figure 3.7: Complexity of MDP-based winner-determination MILP when resource requirements scale with the number of resource types. Actions' resource requirements grow proportionally to the total number of resource types. The number of resource types needed by each action is 10% of the total number of resource types $|\mathcal{O}|$.

preferences that are defined by underlying Markov decision processes, the resource-allocation mechanism developed in this chapter can lead to significant computational advantages. As shown in Figure 3.5, the method can be successfully applied to very large problems that, we argue, are well beyond the reach of combinatorial resource-allocation mechanisms with flat preferences. As our experiments show (Figure 3.3), even for small problems, combinatorial resource allocation mechanisms with flat preferences can be time-consuming, and our attempts to empirically evaluate those simpler mechanisms on larger problems proved futile. For instance, our method takes under one minute to solve a problem that, in the standard CA, requires the agents to enumerate up to 2^{100} bundles and the auctioneer to solve an NP-complete problem with an input of that size.

3.3 Conclusions

In this chapter, we presented a computationally efficient variant of a GVA for resource allocation among self-interested agents. The main contributions of the work presented in this chapter include the following:

- We developed a new auction-based mechanism for resource allocation among self-interested agents whose valuations for resource bundles are defined via the constrained MDP model from Chapter 2. The benefit of this mechanism is that it allows one to avoid an explicit enumeration of the exponentially large set of resource bundles in both the agents' valuation and the auctioneer's winner-determination problems (as described in Section 3.1.2).

- We showed analytically and empirically the drastic reduction in the complexity of the resource-allocation mechanism that can be achieved by exploiting the structure in agents' MDP-based preferences over resources.

- We described how the solution algorithm for the winner-determination problem can be effectively distributed among the agents to achieve further gains in computational complexity, while ensuring that the mechanism remains strategy-proof.

- We showed how the resource-allocation mechanism can be implemented so that information privacy is maintained to a large extent, by having the agents "encrypt" their MDPs before submitting them to the auctioneer.

The treatment of this chapter focused on the problem of resource allocation among self-interested agents, but the algorithms also apply to cooperative MDPs with weakly interacting agents. Moreover, in the cooperative setting, the concept of resources can be viewed as a compact way to model inter-agent constraints and their inability to include some combinations of joint actions in their policies. Such weakly coupled MDPs, where agents have independent transition and reward functions, but certain combinations of joint actions are not feasible, is a widely used model of agents' interactions (e.g., Singh & Cohn, 1998; Ghavamzadeh & Mahadevan, 2002). Our model was resource-centric, but more direct models are also certainly possible. For example, agents can use SAT formulas to describe valid combinations of joint actions. This case can be easily handled via simple modifications to the single and multiagent MILPs (2.20) and (3.4). Indeed, any SAT formula can be expressed as a set of linear inequalities on binary variables $\Delta(a)$ (or $\Delta^m(a)$ in the multiagent case), which can be directly added to the corresponding MILP.

Overall, we believe that the models and solution algorithms described in this chapter significantly further the applicability of combinatorial resource-allocation mechanisms to practical problems, where the utility functions for resource bundles are defined by sequential stochastic decision-making problems.

CHAPTER 4

Constrained MDPs with Multiple Discount Factors

In this chapter, we deviate from the standard discounted infinite-horizon MDP model and consider an extended model that has cost constraints and multiple discount factors. The purpose of this chapter is twofold. First, we show how the ideas developed in the previous chapters can be applied to create algorithms for a different MDP model that has been formulated previously, but for which no implementable algorithms have been devised. Second, we illustrate the generality of the resource-allocation mechanism from Chapter 3 by applying it to problems where the agents' planning problems are formulated as such extended MDPs with multiple discount factors.[1]

4.1 Justification for Costs and Multiple Discounts

The classical MDP models, such as the discounted infinite-horizon MDP discussed earlier, aim to maximize a measure of the aggregate reward received by the agent. Such MDPs have a number of very nice properties: they are subject to the principle of local optimality, according to which the optimal action for a state is independent of the choice of actions for other states, and, as discussed in Section 2.1, optimal policies for such MDPs are stationary, deterministic, and uniformly optimal. These properties translate into very efficient dynamic and linear programming solution algorithms, and policies that are easy to implement in standard agent architectures.

However, there are numerous domains where the classical MDP model proves inadequate, because it can be very difficult to fold all the relevant feedback from the environment (i.e., rewards the agent receives and costs it incurs) into a single scalar reward function. In particular, the agent's actions, in addition to producing rewards, might also incur costs that might be measured very differently from the rewards, making it hard or impossible to express both on the same scale. For example, a natural problem for a delivery agent is to maximize aggregate reward for making deliveries, subject to constraints on the total time spent en route. Problems naturally modeled as constrained MDPs also often arise in other domains: for example, in telecommunication applications (e.g., (Lazar, 1983)), where it is desirable to maximize throughput subject to delay constraints.

Another situation where the classical MDP model is not expressive enough is where an agent receives multiple reward streams and incurs multiple costs, each with a different discount factor. For example, the delivery agent could face a rush-hour situation where the

[1]This chapter is based on the work that originally appeared in (Dolgov & Durfee, 2005c).

rewards for making deliveries decrease as a function of time (same delivery action produces lower reward if it is executed at a later time), while the traffic conditions improve with time (same delivery action can be executed faster at a later time). If the rewards decrease and traffic conditions improve on different time scales, the problem can be naturally modeled with two discount factors, allowing the agent to evaluate the tradeoffs between early and late delivery. Problems with multiple discount factors also frequently arise in other domains: for example, an agent can be involved in several financial ventures with different risk levels and time scales, where a model with multiple discount factors would allow the decision maker to quantitatively weigh the tradeoffs between shorter- and longer-term investments. Feinberg and Shwartz (1999) describe more examples and provide further justification for constrained models with several discount factors.

The price we have to pay for extending the classical model by introducing constraints and several discount factors is that stationary deterministic policies are no longer guaranteed to be optimal (Feinberg & Shwartz, 1994, 1995). Searching for an optimal policy in the larger class of non-stationary randomized policies can dramatically increase problem complexity; in fact, the complexity of finding optimal policies for this broad class of constrained MDPs with multiple costs, rewards, and discount factors is not known, and no solution algorithms exist (aside from some very special cases (Feinberg & Shwartz, 1996)). Furthermore, even if they could be found, these non-stationary randomized policies might not be reliably executable by basic agent architectures. For example, (Paruchuri, Tambe, Ordonez, & Kraus, 2004) described how executing randomized policies in multiagent systems can be problematic.

In this chapter, we therefore, focus on finding optimal stationary deterministic policies for MDPs with multiple rewards, costs, and discount factors. This problem has been studied before and has been proven to be NP-complete by Feinberg (2000), who formulated it as a non-linear and non-convex mathematical program. Unfortunately, aside from intractable techniques of general non-convex optimization, these problems have heretofore not been practically solvable.

4.2 Stationary Deterministic Policies for Constrained MDPs

We begin by analyzing the constrained MDP with a single reward, multiple costs, and one discount factor. Next, in Section 4.3, we expand this to the case with multiple rewards and costs, and several discount factors.

Suppose that the agent, besides getting rewards for executing actions, also incurs costs: $c^k : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R},\ k \in [1, K]$, where $c^k(s,a)$ is the cost of type k incurred for executing action a in state s (e.g., actions might take time and cost money, in which case we would say that there are two types of costs). Then, a natural problem to pose is to maximize the expected discounted reward subject to some upper bounds on the total expected discounted costs.

We refer to the total expected discounted reward as:

$$R_\gamma(\pi, \alpha) = \sum_{t=0}^{\infty} (\gamma)^t r_t(\pi, \alpha), \tag{4.1}$$

where $r_t(\pi, \alpha)$ is the (random) cost that the agent incurs at time t.

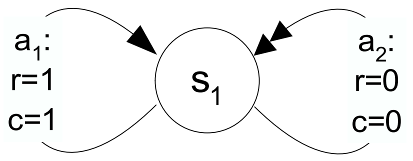


Figure 4.1: Example: MDP with constraint on expected cost.

Similarly, we label the total discounted cost of type $k \in [1, K]$ as:

$$C_\gamma^k = \sum_{t=0}^{\infty} (\gamma)^t c_t^k(\pi, \alpha), \tag{4.2}$$

where $c_t^k(\pi, \alpha)$ is the (random) cost that the agent incurs at time t. Then, the constrained discounted MDP can be written as

$$\begin{aligned} &\max_\pi \mathbb{E}_\alpha\big[R_\gamma(\pi)\big] \\ &\text{subject to:} \\ &\mathbb{E}_\alpha\big[C_\gamma^k(\pi)\big] \le \widehat{c}^k, \qquad \forall k \in [1, K], \end{aligned} \tag{4.3}$$

where $\widehat{c}^k$ is the upper bound on the cost of type k.

4.2.1 Problem Properties

If problem (4.3) is feasible, then there always exists an optimal policy $\pi \in \Pi^{\mathrm{SR}}$ that is stationary, but, in general, randomized and not uniformly optimal (Kallenberg, 1983; Heyman & Sobel, 1984). The problem of finding the optimal stationary randomized policy for the given initial conditions α is P-Complete[2], and can be formulated as the following LP (Kallenberg, 1983; Heyman & Sobel, 1984):

$$\begin{aligned} &\max \sum_{s,a} r(s,a)x(s,a) \\ &\text{subject to:} \\ &\sum_a x(\sigma, a) - \gamma \sum_{s,a} x(s,a)p(\sigma|s,a) = \alpha(\sigma), && \forall \sigma \in \mathcal{S}; \\ &\sum_{s,a} c^k(s,a)x(s,a) \le \widehat{c}^k, && \forall k \in [1, K]; \\ &x(s,a) \ge 0, && \forall s \in \mathcal{S}, a \in \mathcal{A}. \end{aligned} \tag{4.4}$$

Example 4.1. *Consider the problem shown in Figure 4.1, which has a single state (s_1), two actions (a_1 and a_2), and a single cost function (c^1). Let $\gamma = 0.9$. Then, the obvious optimal policy for the unconstrained problem is to always execute the first action: $\pi(s_1, a_1) = 1$, yielding a total expected discounted reward of* 10 *(and the same total cost).*

[2]Presence in P follows from the LP formulation below, and completeness follows from a trivial reduction from an unconstrained MDP.

Now, consider a constrained problem where the total cost is bounded by $\widehat{C} = 5$. Clearly, the solution to the unconstrained problem does not satisfy the constraint. The optimal solution to the constrained problem is to randomize between a_1 and a_2 to lower the total expected cost.

The constraints in the LP (4.4) then become:

$$\begin{aligned}
&x(s_1, a_1) + x(s_1, a_2) - 0.9x(s_1, a_1) - 0.9x(s_1, a_2) = 1; \\
&x(s_1, a_1) + x(s_1, a_2) \leq 5.
\end{aligned}$$

The optimal feasible solution that maximizes $x(s_1, a_1)$ is then $x(s_1, a_1) = x(s_1, a_2) = 5$, which maps to the policy that equally randomizes between the two actions. ■

As argued in Section 4.1, randomized policies are often more difficult to implement than deterministic ones, thus, it can be desirable to compute optimal solutions to (4.3) from the class of stationary deterministic policies:

$$\begin{aligned}
&\max_{\pi} \mathbb{E}_{\alpha}\big[R_{\gamma}(\pi)\big] \\
&\text{subject to:} \\
&\quad \mathbb{E}_{\alpha}\big[C^k_{\gamma}(\pi)\big] \leq \widehat{c}^k, \qquad \forall k \in [1, K]; \\
&\quad \pi \in \Pi^{\text{SD}}.
\end{aligned} \tag{4.5}$$

This, however, is more difficult: as shown by Feinberg (2000) (using a reduction similar to that of Filar and Krass (1994)), the decision version of (4.5) is NP-complete. Feinberg (2000) also reduced (4.5) to a mathematical program by augmenting (4.4) with the following constraint, ensuring that only one $x(s, a)$ per state is nonzero:

$$|x(s, a) - x(s, a')| = x(s, a) + x(s, a'), \quad \forall s \in \mathcal{S}; a, a' \in \mathcal{A}. \tag{4.6}$$

However, the resulting program (4.4,4.6) is neither linear nor convex, and thus presents significant computational challenges.

In related work that establishes connections between Hamiltonian cycles and MDPs, Ejov, Filar, and Gondzio (2004) consider the inverse task of solving HC problems by reducing them to constrained MDPs with average rewards, and solving the latter for optimal stationary deterministic policies. Their approach for solving the constrained MDP is based on formulating it as an indefinite quadratic program. This non-convex quadratic program can then be approximated by "nearly convex" quadratic functions to yield an approximately optimal deterministic policy, which can then be converted to an HC for the original problem.

4.2.2 Solution Algorithm

We show how (4.5) can be reduced to a mixed integer linear program. The following lemma provides the basis for our reduction.

Lemma 4.2. *Consider an MDP $\langle \mathcal{S}, \mathcal{A}, p, r, \alpha \rangle$, a policy π, its corresponding occupation measure x (given α), a constant $X \geq x(s, a)$ $\forall s \in \mathcal{S}, a \in \mathcal{A}$, and a set of binary variables $\Delta(s, a) = \{0, 1\}$, $\forall s \in \mathcal{S}, a \in \mathcal{A}$.*

If x and Δ satisfy the following conditions

$$\sum_a \Delta(s,a) \leq 1, \quad \forall s \in \mathcal{S}, \tag{4.7}$$

$$x(s,a)/X \leq \Delta(s,a), \quad \forall s \in \mathcal{S}, a \in \mathcal{A}, \tag{4.8}$$

then, for all states s that, under π and α, have a nonzero probability of being visited ($\sum_a x(s,a) > 0$), π is deterministic, and the following holds:

$$\Delta(s,a) = 1 \Leftrightarrow x(s,a) > 0. \tag{4.9}$$

Proof. Consider a state s^* that, under policy π and initial distribution α, has a nonzero probability of being visited, i.e., $\sum_a x(s^*,a) > 0$. Then, since the occupation measure is non-negative, there must be at least one action in this state that has a non-zero occupation measure:

$$\exists a^* \in \mathcal{A} \quad \text{s.t.} \quad x(s^*,a^*) > 0.$$

Then, (4.8) forces $\Delta(s^* a^*) = 1$, which, due to (4.7), forces zero values for all other Δ's for state s^*:

$$\Delta(s^* a) = 0 \quad \forall a \neq a^*.$$

Given (4.8), this in turn, means that the occupation measure for all other actions has to be zero:

$$x(s^*,a) = 0 \quad \forall a \neq a^*,$$

which, per (2.11), translates into the fact that the policy π is deterministic and $\Delta(s,a) = 1 \Leftrightarrow x(s,a) > 0$. ■

Lemma 4.2 immediately leads to the following MILP that is equivalent to (4.5):

$$\begin{aligned}
&\max \sum_s \sum_a x(s,a) r(s,a) \\
&\text{subject to:} \\
&\sum_a x(\sigma,a) - \gamma \sum_{s,a} x(s,a) p(\sigma|s,a) = \alpha(\sigma), && \forall \sigma \in \mathcal{S}, a \in \mathcal{A}; \\
&\sum_{s,a} c^k(s,a) x(s,a) \leq \widehat{c}^k, && \forall k \in [1,K]; \\
&\sum_a \Delta(s,a) \leq 1, && \forall s \in \mathcal{S}; \\
&x(s,a)/X \leq \Delta(s,a), && \forall s \in \mathcal{S}, a \in \mathcal{A}; \\
&x(s,a) \geq 0, && \forall s \in \mathcal{S}, a \in \mathcal{A}; \\
&\Delta(s,a) \in \{0,1\}, && \forall s \in \mathcal{S}, a \in \mathcal{A}.
\end{aligned} \tag{4.10}$$

The normalization constant X in this MILP can be computed in polynomial time by, for example, solving the LP (2.10) with the objective function replaced by $\max \sum_{s,a} x(s,a)$ and setting X to its maximum value.

The above reduction to an MILP is most valuable for domains where it is difficult to implement a randomized stationary policy because of an agent's architectural limitations. It is also of interest for domains where such limitations are not present, as it can be used for evaluating the quality vs. implementation-difficulty tradeoffs between randomized and deterministic policies during the agent-design phase.

Example 4.3. *Again, consider the problem from Example 4.1 that was shown in Figure 4.1. Using the same cost constraint and discount factor as in Example 4.1, the MILP (4.10) becomes:*

$$\begin{aligned}
&\max x(s_1, a_1) \\
&\textit{subject to:} \\
&\quad x(s_1, a_1) + x(s_1, a_2) - 0.9x(s_1, a_1) - 0.9x(s_1, a_2) = 1; \\
&\quad x(s_1, a_1) + x(s_1, a_2) \le 5; \\
&\quad \Delta(s_1, a_1) + \Delta(s_1, a_2) \le 1; \\
&\quad 1/10x(s_1, a_1) \le \Delta(s_1, a_1); \\
&\quad 1/10x(s_1, a_2) \le \Delta(s_1, a_2); \\
&\quad x(s_1, a_1) \ge 0; \quad x(s_1, a_2) \ge 0; \quad \Delta(s_1, a_1) \in \{0, 1\}; \quad \Delta(s_1, a_2) \in \{0, 1\}.
\end{aligned}$$

An optimal solution to which is $x(s_1, a_1) = 1, x(s_1, a_1) = 0, \Delta(s_1, a_1) = 1, \Delta(s_1, a_2) = 0$, which maps to the deterministic policy that always executes a_1. ■

4.3 Stationary Deterministic Policies for MDPs with Multiple Discounts

We now turn our attention to the more general case of MDPs with multiple streams of rewards and costs, each with its own discount factor $\gamma_n,\ n \in [1, N]$. In this model, the total discounted reward is a weighted sum of the N discounted reward streams:

$$R(\pi) = \sum_n \beta_n R_{\gamma_n}(\pi) = \sum_n \beta_n \sum_{t=0}^{\infty} (\gamma_n)^t r_t(\pi, \alpha), \tag{4.11}$$

where β_n is the weight of the n^{th} reward stream that is defined by the reward function $r_n : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$.

Similarly, each of the K total expected costs is a weighted sum of N discounted cost streams:

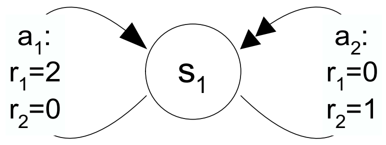
$$C^k(\pi) = \sum_n \beta_{kn} C^k_{\gamma_n} = \sum_n \beta_{kn} \sum_{t=0}^{\infty} (\gamma_n)^t c^{kn}_t(\pi, \alpha), \tag{4.12}$$

where β_{kn} is the weight of the n^{th} discounted stream of cost of type k, defined by the cost function $c^{kn} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$.

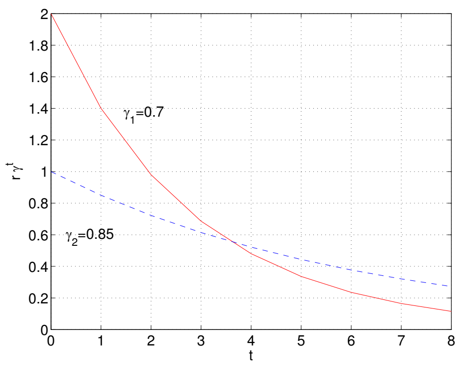
Notice that in the definition of the MDP with multiple discount factors, there are N reward functions and KN cost functions (unlike the constrained MDP from the previous section that had 1 and K reward and cost functions, respectively).

The goal is to maximize the total weighted discounted reward, subject to constraints on weighted discounted costs:

$$\begin{aligned}
&\max_\pi \mathbb{E}_\alpha \Big[\sum_n \beta_n R_{\gamma_n}(\pi) \Big] \\
&\text{subject to:} \\
&\mathbb{E}_\alpha \Big[\sum_n \beta_{kn} C^k_{\gamma_n}(\pi) \Big] \le \widehat{c}^k, \qquad \forall k \in [1, K].
\end{aligned} \tag{4.13}$$



(a)



(b)

Figure 4.2: Example: MDP with two discount factors.

4.3.1 Problem Properties

Feinberg and Shwartz (1994, 1995) developed a general theory of constrained MDPs with multiple discount factors. The challenge with this class of problems is that optimal solutions to (4.13) are, in general, not stationary, as illustrated by the following example.

Example 4.4. *Consider the simple problem shown in Figure 4.2a. There is one state (s_1), two actions (a_1 and a_2), and two reward functions (r_1 and r_2). The first action produces rewards $r_1 = 2$ and $r_2 = 0$, while the second action yields $r_1 = 0$ and $r_2 = 1$. In this example, if $\gamma_1 \geq \gamma_2$, the optimal policy is to always execute a_1, leading to a total reward if $1/(1-\gamma_1)$. However, if $\gamma_1 < \gamma_2$, the optimal policy is to switch to a_2 at time t^* for which $\gamma_2^{t^*} \geq 2\gamma_1^{t^*}$, and execute a_2 from there on. Figure 4.2b shows the discounted rewards for $\gamma_1 = 0.7$ and $\gamma_2 = 0.85$. For these discount factors, the optimal policy is to switch from a_1 to a_2 staring at time $t = 4$.* ■

Unfortunately, except for some special cases (Feinberg & Shwartz, 1996), there are no implementable algorithms for finding optimal policies for such problems. Because of this, and given the complexity of implementing non-stationary randomized policies (even if they could be found), it is worthwhile to consider the following problem of finding solutions from the class of stationary policies:

$$\begin{aligned}
&\max_{\pi} \mathbb{E}_\alpha\Big[\sum_n \beta_n R_{\gamma_n}(\pi)\Big] \\
&\text{subject to:} \\
&\mathbb{E}_\alpha\Big[\sum_n \beta_{kn} C^k_{\gamma_n}(\pi)\Big] \leq \widehat{c}^k, \qquad \forall k \in [1, K]; \\
&\pi \in \Pi^{\mathrm{SR}}.
\end{aligned} \tag{4.14}$$

Feinberg (2000) showed that the decision version of this problem is NP-complete, and he also formulated (4.14) as the following mathematical program (again, based on (4.4)):

$$
\begin{aligned}
&\max \sum_n \beta_n \sum_{s,a} r_n(s,a) x_n(s,a) \\
&\text{subject to:} \\
&\sum_a x_n(\sigma,a) - \gamma_n \sum_{s,a} x_n(s,a) p(\sigma|s,a) = \alpha(\sigma), && \forall \sigma \in \mathcal{S}, a \in \mathcal{A}, n \in [1,N]; \\
&\sum_n \beta_{kn} \sum_{s,a} c^{kn}(s,a) x_n(s,a) \leq \hat{c}^k, && \forall k \in [1,K]; \\
&x_n(s,a) / \sum_a x_n(s,a) = x^{n+1}(s,a) / \sum_a x^{n+1}(s,a), && \forall s \in \mathcal{S}, a \in \mathcal{A}, n \in [1,N]; \\
&x_n(s,a) \geq 0, && \forall s \in \mathcal{S}, a \in \mathcal{A}, n \in [1,N].
\end{aligned}
\tag{4.15}
$$

This program has an occupation measure x_n for each discount factor γ_n, $n \in [1,N]$ and expresses the total reward and total costs as weighted linear functions of these occupation measures. The first set of constraints contains the conservation of flow constraints for each of the N occupation measures, and the third set of constraints ensures that all occupation measures map to the same policy (recall (2.11)).

Furthermore, Feinberg (2000) also proved that if we limit the search to stationary deterministic policies Π^{SD}, the problem remains NP-complete. Under the presence of constraints, this result directly follows from the NP-completeness of the problem of finding optimal stationary deterministic policies for constrained MDPs with a single discount factor (4.5). However, this complexity result also holds even for unconstrained problems with multiple discount factors.

As in the previous section, if we are interested in limiting the search to policies in Π^{SD}, this can be accomplished by imposing the following additional constraint on the occupation measures in (4.15) (Feinberg, 2000):

$$
\Big| \sum_n \big(x_n(s,a) - x_n(s,a') \big) \Big| = \sum_n \big(x_n(s,a) + x_n(s,a') \big), \qquad \forall s \in \mathcal{S}; a, a' \in \mathcal{A}. \tag{4.16}
$$

Because of the synchronization of the different occupation measures and the constraint that forces deterministic policies, this program (4.15,4.16) is non-linear and non-convex, and is thus difficult to solve.

4.3.2 Solution Algorithm

For the problem of finding optimal stationary deterministic policies, we present a reduction of (4.15,4.16) to a MILP. As before, a benefit of this reduction is that it opens the possibility of using a wide array of efficient solution techniques and tools. Our reduction is based on the following lemma.

Lemma 4.5. *Consider an MDP $\langle \mathcal{S}, \mathcal{A}, p, r, \alpha \rangle$ with several discount factors γ_n, for $n \in [1,N]$, a set of policies π_n, $n \in [1,N]$ with their corresponding occupation measures x_n (policy π_n and discount factor γ_n define x_n), a constant $X \geq x_n(s,a)$ $\forall n \in [1,N], s \in \mathcal{S}, a \in \mathcal{A}$, and a set of binary variables $\Delta(s,a) = \{0,1\}$.*

If x_n and Δ satisfy the following conditions

$$\sum_a \Delta(s,a) \le 1, \quad \forall s \in \mathcal{S}, \tag{4.17}$$

$$x_n(s,a)/X \le \Delta(s,a), \quad \forall n \in [1,N], s \in \mathcal{S}, a \in \mathcal{A}, \tag{4.18}$$

then, the sets of reachable states $\mathcal{I}_n = \{s : \sum_a x_n(s,a) > 0\}$ defined by all occupation measures are the same, i.e., $\mathcal{I}_n = \mathcal{I}_{n'}$, $\forall n, n' \in [1,N]$. Furthermore, all π_n are deterministic on $\mathcal{I}_n$, and $\pi_n(s,a) = \pi^{n'}(s,a)$ $\forall n, n' \in [1,N]$.

Proof. Consider an initial state s^* (i.e., $\alpha(s^*) > 0$). Following the argument of Lemma 4.2, the policy for that state is deterministic:

$$\exists a^* : \quad x_n(s^*,a^*) > 0, \Delta(s^* a^*) = 1; \; x_n(s^*,a) = 0, \Delta(s^* a) = 0 \; \forall a \neq a^*.$$

This implies that all N occupation measures x_n must prescribe the execution of the same deterministic action a^* for state s^*, because all $x_n(s,a)$ are tied to the same $\Delta(s,a)$ via (4.18).

Therefore, all occupation measures x_n correspond to the same deterministic policy on the initial states $\mathcal{I}_0 = \{s : \alpha(s) > 0\}$. We can then expand this statement by induction to all reachable states. Indeed, the set of states $\mathcal{I}_1$ that are reachable from $\mathcal{I}_0$ in one step will be the same for all x_n. Then, by the same argument as above, all x_n map to the same deterministic policy on $\mathcal{I}_1$, and so forth. ■

It immediately follows from Lemma 4.5 that the problem of finding optimal stationary deterministic policies for an MDP with weighted discounted rewards and constraints can be formulated as the following MILP:

$$\begin{aligned}
&\max \sum_n \beta_n \sum_{s,a} r_n(s,a) x(s,a)^n \\
&\text{subject to:} \\
&\sum_a x(\sigma,a)^n - \gamma_n \sum_{s,a} x(s,a)^n p(\sigma|s,a) = \alpha(\sigma), && \forall \sigma \in \mathcal{S}, a \in \mathcal{A}, n \in [1,N]; \\
&\sum_n \beta_{kn} \sum_{s,a} c^{kn}(s,a) x(s,a)^n \le \widehat{c}^k, && \forall k \in [1,K]; \\
&\sum_a \Delta(s,a) \le 1, && \forall s \in \mathcal{S}; \\
&x(s,a)^n / X \le \Delta(s,a), && \forall s \in \mathcal{S}, n \in [1,N]; \\
&x(s,a)^n \ge 0, && \forall s \in \mathcal{S}, a \in \mathcal{A}, n \in [1,N]; \\
&\Delta(s,a) \in \{0,1\}, && \forall s \in \mathcal{S}, a \in \mathcal{A}.
\end{aligned} \tag{4.19}$$

where $X \ge \max x(s,a)^n$ is a constant, as in Lemma 4.5.

Although this MILP produces policies that are only optimal in the class of stationary deterministic ones, at present time there are (to the best of our knowledge) no practical algorithms for finding optimal solutions from any larger policy class for constrained MDPs with multiple discount factors.

4.4 Experimental Observations

We have implemented the MILP algorithm for finding optimal stationary deterministic policies for constrained MDPs discussed in this chapter and empirically evaluated it on a class of test problems. In the following discussion, we focus on the constrained MDPs from Section 4.2, because these problems are better studied, and the existing algorithms for finding optimal randomized policies can serve as benchmarks, whereas there are no alternative algorithms for finding policies that are optimal for general constrained MDPs with multiple discount factors.

In our empirical analysis, we tried to answer the following questions: i) how well do deterministic policies perform, compared to optimal randomized ones, and ii) what is the average-case complexity of the resulting MILPs. The answers to these questions are obviously domain-dependent, so the following discussion should not be viewed as a comprehensive characterization of the behavior of our algorithms on constrained MDPs. However, we believe that our experiments provide some interesting observations about such problems.

We experimented with a large set of randomly generated problems and with a more meaningful manually constructed domain, which we randomly perturbed in several ways. The big picture resulting from the experiments on the randomly generated domains was very similar to the one from the manually constructed example, providing a certain measure of comfort about the stability and validity of our observations. We report here the results for the manually constructed domain.

For our test domain, we used a model of an autonomous delivery agent, similar to the one used for the resource-allocations experiments in Chapter 3. In this single-agent variant of the domain, the agent is operating in a grid world with delivery sites placed randomly throughout the grid. The agent moves around the grid (incurring small negative rewards for every move) and receives positive rewards for making deliveries. The agent's movement is non-deterministic, and the agent has some probability of getting stuck in randomly placed dangerous locations. The agent also incurs a scalar cost (e.g., time) per move, and the objective is to maximize the total expected discounted reward subject to an upper bound on the total expected discounted cost.

The results of our experiments are summarized below. First, we analyzed the values of randomized and deterministic policies as functions of the constraint level ($[0, 1]$), where 0 means that only policies that incur zero cost are feasible (strictest possible constraint), whereas 1 means that the upper bound on cost equals the cost of the optimal unconstrained policy (agent is not constrained at all). The first observation, as illustrated in Figure 4.3, is that the value of stationary deterministic policies for constrained problems is reasonably close to optimal. We can also observe that the value of deterministic policies changes in a very discrete manner (i.e., it jumps up at certain constraint levels), whereas the value of randomized policies changes continuously. This is, of course, only natural, given that the space of randomized policies is continuous, and randomized policies can gradually increase the probability of taking "better" actions as cost constraints are relaxed. On the other hand, the space of deterministic policies is discrete, and their quality jumps when the constraints are relaxed to permit the agent to switch to a better action. While the number and the size of these jumps in the value function depends on the dynamics of the MDP, the high-level picture was the same in all of our experiments.

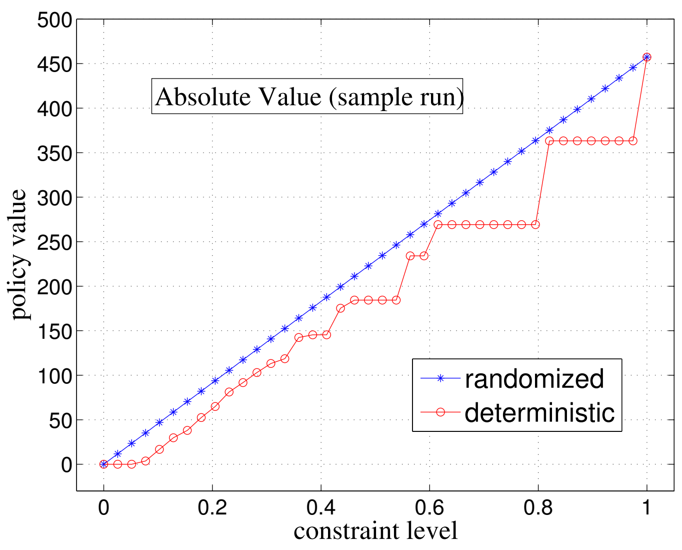


Figure 4.3: Value of deterministic and randomized policies for constrained MDPs.

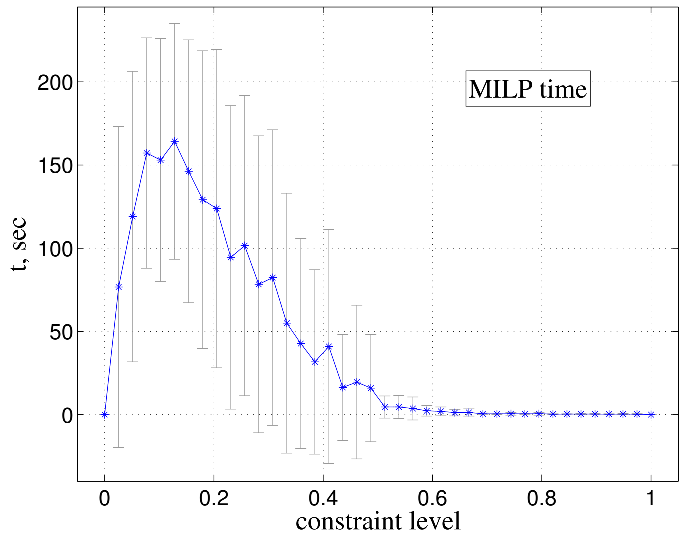


Figure 4.4: Complexity profile for constrained MDPs as a function of constraint level.

Figure 4.4 shows the running time of the MILP solver as a function of the tightness of the constraints (here and in Figure 4.5 the plots contain values averaged over 100 runs, with the error bars showing the standard deviation). The data indicates that our MILPs (4.10) have an easy-hard-easy complexity profile, although without a sharp phase transition from hard to easy, i.e., the problems very quickly become hard, and then gradually get easier as cost constraints are relaxed.

This complexity profile gives rise to the question regarding the source of the difficulty for solving MILPs in the "hard" region: is it difficult to find good feasible solutions, or is it time-consuming to prove their optimality? Figure 4.5 suggests that the latter is the case, which can be considered as the more fortunate outcome, since algorithms with such performance profiles can be successfully used in an anytime manner. The figure contains a

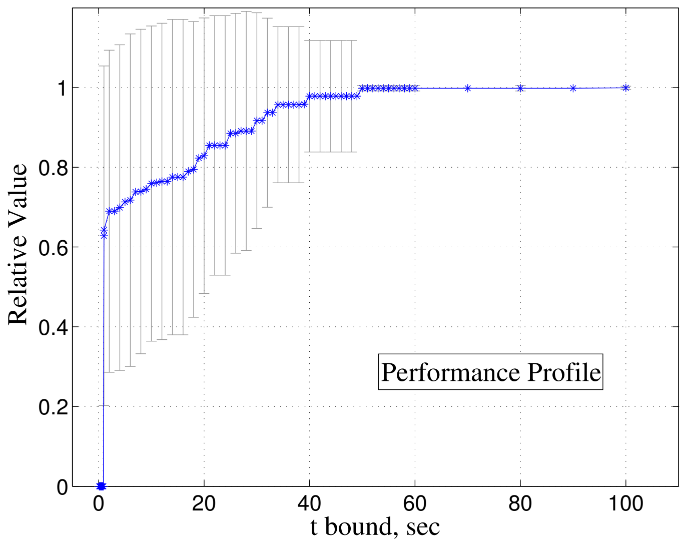


Figure 4.5: Performance profile for finding optimal deterministic policies for constrained MDPs.

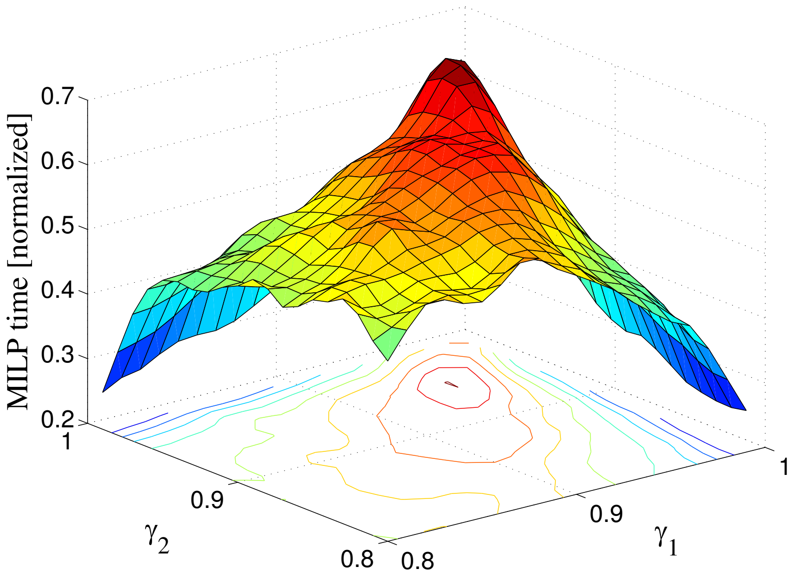


Figure 4.6: Solution time for MDP with two discount factors.

plot of the quality of the best solution found as a function of the time bound imposed on the MILP solver[3] for problems in the hardest constraint region (constraint tightness level of 0.13). As the graph shows, very good policies are usually produced quickly.

We conclude with a somewhat intriguing observation about the MILP solution time for constrained MDPs with multiple discount factors (Section 4.3). We generated and solved a large number of random MDPs with two discount factors and plotted (after cubic smoothing) the average solution time (shown in Figure 4.6). An interesting observation

[3]CPLEX 8.1 on a P4 performed the role of the MILP solver.

about this plot is that the problem instances where the two discount factors are equal (or close) appear to be the hardest (notice the contours in the γ_1-γ_2 plane). This is counterintuitive, because such MDPs are equivalent to standard MDPs with one discount factor. A possible explanation might be that when discount factors are far apart, one of the reward functions dominates the other and the problem becomes simpler, while when the discount factors are close, the tradeoffs become more complicated (with the equivalence to a standard MDP hidden in the MILP translation). However, this is speculation and a more careful analysis would be needed to explain this phenomenon.

4.5 Discussion and Generalization

We have presented algorithms for finding optimal deterministic policies for two classes of constrained MDPs, and in both cases we were maximizing a measure of the total expected discounted reward subject to constraints on the total expected discounted costs. However, our technique of finding optimal stationary deterministic policies via mixed integer programming also applies to other classes of MDPs.

In particular, the same methodology applies to MDPs with average per-time rewards and constraints (e.g., (Puterman, 1994)). Similarly to the constrained total-reward discounted MDP model described in Section 4.2, the MDP with average rewards and constraints can also be formulated as an LP (similar to (4.4)) that yields optimal stationary randomized policies. The problem of finding optimal stationary deterministic policies for such MDPs is also known to be NP-complete (Filar & Krass, 1994). Our MILP reduction of Section 4.2 carries through with almost no changes and can thus be used to find optimal stationary deterministic policies for such MDPs.

We have presented a stand-alone treatment of the problem of finding stationary deterministic policies for MDPs with multiple discounts. However, this problem is a special case of the resource-based MDP problem introduced in Chapter 2. In fact, both problems studied in this chapter, (4.5) and (4.14), can be formulated as a special case of the MDP with resources and capacity constraints (2.13). If we create a unique resource type o per every state-action pair $\langle s, a\rangle$, then the binary variables $\Delta(s, a)$ will map exactly to the $\delta(o)$ used in the resource-based model. We can then create a capacity type per state and set the capacity costs such that only one action choice per state is admissible.

Finally, we note that the resource-allocation MILP from Chapter 3 can be easily adapted to work with the MDP model with multiple discounts that was discussed in this chapter. Indeed, it is straightforward to combine the multiagent resource-allocation MILP (3.4) and the policy-optimization problem (4.19) by simply merging the constraints.

4.6 Conclusions

In this chapter, we have developed a general integer programming method for finding optimal stationary deterministic policies in constrained MDPs. The main contributions of the work described in this chapter are the following:

- For the first problem of finding stationary deterministic policies for constrained MDPs, our methodology is of most value for domains where randomized policies (which work better and are easier to compute) are undesirable or difficult to implement because of an agent's architectural limitations. However, even in the absence of such limitations, the approach is useful in situations where it is desirable to compare the quality of randomized and deterministic policies, such as when an agent is being designed for a particular task and it is necessary to weigh the cost of implementing a more complex policy-execution mechanism against the gain in expected performance.

- For the second problem of finding optimal stationary deterministic policies for constrained MDPs with multiple discount factors, to the best of our knowledge, no feasible algorithms have been reported for finding optimal solutions in any interesting policy class, and thus our MILP approach for finding optimal stationary deterministic policies provides the first practical approach to dealing with constrained MDPs with multiple discount factors.

CHAPTER 5

Consumable Resources

In this chapter, we consider the case of allocating consumable resources, i.e., resources that are used during action execution. For example, time, money, hard-drive space, equipment that depreciates when used are all typical instances of consumable resources.[1]

The main difficulty that we encounter when dealing with consumable resources is that the definition of the value of a bundle of resources is ambiguous. This is due to the fact that for a fixed policy (and even a fixed initial state), the total consumption of a resource is nondeterministic. Indeed, the agent consumes a constant quantity of a resource every time it executes an action, and the total number of times the agent executes the action is a random variable whose realization depends on the trajectory of the system. This fundamentally differs from the case of non-consumable resources studied in the earlier chapters, where the resource requirements of a policy and its capacity costs could be computed deterministically.

Therefore, given the uncertainty in the total resource requirements of a policy, there are several ways that the value of a resource bundle can be defined. For instance, for some domains it might be appropriate to say that the value of a resource bundle is the expected utility of the best policy whose resource consumption *on average* does not exceed the available resource amounts. This is most appropriate when the decision-maker has control over multiple identical systems and is concerned with the average resource consumption per system (e.g., a cluster of computers that share data storage). In this case over-usage by some systems is compensated by under-usage by others; when the number of systems is sufficiently large, the total consumption follows a normal distribution, for which it is easy to compute a reasonable safety margin. However, such risk-neutral approaches are not always applicable and expressive enough, as pointed out, for example, by Ross and Chen (1988) in the telecommunication domain. In particular, under some conditions it might be desirable to consider the worst-case scenario and define the value of a resource bundle as the expected utility of the best policy whose resource usage *never* exceeds the available resource amounts. This might be more appropriate when the resource in question is, for example, fuel, and exceeding the available amounts might have catastrophic effects (for instance, in an airplane). In yet other domains, it might be desirable to take a middle-ground approach with a probabilistic view that defines the value of a resource bundle as the expected utility of the best policy whose *probability* of exceeding the available resource amounts is bounded.

[1]This chapter is based on the material that was originally published in (Dolgov & Durfee, 2003) and (Dolgov & Durfee, 2004a).

In this chapter we examine the various scenarios, analyze the corresponding resource-allocation and policy optimization problems and provide solution algorithms. As with the case of non-consumable resources, we begin the discussion with a study of the single-agent policy optimization and then discuss the multiagent resource-allocation problems.

5.1 Single-Agent Problem Formulation

The single-agent model can be defined as $\langle \mathcal{S}, \mathcal{A}, p, r, \mathcal{Q}, \rho_q, \mathcal{C}, \kappa_q, \widehat{\kappa}, \alpha \rangle$, where:

- $\langle \mathcal{S}, \mathcal{A}, r, p, \alpha \rangle$ are the standard components of an MDP, as defined in Section 2.1.
- $\mathcal{Q}$ is the set of consumable resources (e.g., $\mathcal{Q} = \{\text{memory}, \text{fuel}, \ldots\}$).
- $\rho_q : \mathcal{A} \times \mathcal{Q} \mapsto \mathbb{R}$ is a function that specifies the resource requirements of all actions; $\rho_q(a, q)$ defines how much of resource q action a needs *per execution* (e.g., $\rho_q(a, \text{fuel}) = 1$ means that action a requires one unit of fuel every time it is carried out).[2]
- $\mathcal{C}$ is the set of capacities of our agent, defined exactly as in the previous chapters dealing with non-consumable resources (e.g., $\mathcal{C} = \{\text{weight}, \text{space}, \ldots\}$).
- $\kappa_q : \mathcal{Q} \times \mathcal{C} \mapsto \mathbb{R}$ is a function that specifies the capacity costs of resources; $\kappa_q(q, c)$ defines how much of capacity c a unit of resource q consumes (e.g., $\kappa_q(\text{fuel}, \text{weight}) = 0.9\text{kg}$ specifies that one unit (e.g., liter) of fuel weighs 0.9kg.
- $\widehat{\kappa} : \mathcal{C} \mapsto \mathbb{R}$ specifies the upper bound of the capacities, defined exactly as in the chapters dealing with non-consumable resources; $\widehat{\kappa}(c)$ gives the upper bound on capacity c (e.g., $\widehat{\kappa}(\text{weight}) = 10\text{kg}$ means that the agent cannot carry more than 10kg of weight).

In the following sections we present an analysis of the problem under the constraints on the *expected* capacity costs, and we also discuss probabilistic constraints, where we bound the *probability* of exceeding a pre-specified limit on the total capacity costs. Note that the worst-case scenario mentioned above is a special case of the probabilistic constraint, where the probability of exceeding the available resource amounts is bounded by 0.

We use $X_\rho(q)$ to denote the (random) total discounted usage of resource q.[3] We will then be interested in solving the following two problems. In the first problem, we will consider the expected resource costs, so our goal is to find a policy π that yields the highest expected reward, under the conditions that the expected resource requirements of that policy do not exceed the capacity bounds of the agent:

$$\begin{aligned} &\max U_\gamma(\pi, \alpha) \\ &\text{subject to:} \\ &\sum_q \kappa_q(q, c) \mathbb{E}\Big[X_\rho(q) | \pi, \alpha\Big] \le \widehat{\kappa}(c), \qquad \forall c \in \mathcal{C}. \end{aligned} \tag{5.1}$$

[2]For simplicity and consistency with the model of non-consumable resources, we map actions directly to resources. It is also possible to use a resource function $\rho_q : \mathcal{A} \times \mathcal{S} \times \mathcal{Q} \mapsto \mathbb{R}$ that assigns different resource costs, depending on the current state. All of the results of this chapter directly carry over to that case.

[3]When dealing with consumable resources, discounting can be interpreted as the probability of exiting the system at each time step. If we adopt the contracting MDP model as defined in Section 2.1, we can talk about the total (undiscounted) resource consumption, which will always be bounded.

In the second problem that we will consider, we will want to bound the probability that the total capacity costs exceed the capacity limits of the agent:

$$\begin{aligned}
&\max U_\gamma(\pi, \alpha) \\
&\text{subject to:} \\
&\mathbb{P}\Big[\sum_q \kappa_q(q,c) X_\rho(q) > \widehat{\kappa}(c) \Big| \pi, \alpha\Big] \le P_0(c), \quad \forall c \in \mathcal{C},
\end{aligned} \tag{5.2}$$

where $P_0(c)$ is the user-specified upper bound on the probability that the total consumption of capacity c exceeds $\widehat{\kappa}(c)$. For this problem, we will focus on finding optimal stationary randomized policies, and will also briefly discuss the problem of finding optimal deterministic policies.

Example 5.1. *Consider the delivery domain from Chapter 2, as introduced in Example 2.1. Suppose that it costs money to service the truck and, as it often happens, the money for servicing it comes from a separate budget of the company running the deliveries. For the purpose of this example, we are going to assume that fixing the truck if it breaks down does not dig into this budget (perhaps the money comes out of a different budget).*

Under these conditions, the optimization problem could be to maximize rewards for making the deliveries, subject to budget constraints for servicing the vehicle. This problem could be modeled by augmenting the unconstrained problem in Example 2.1 as follows. There is only one consumable resource (the number of times the vehicle is serviced): $\mathcal{Q} = \{v\}$, *and each service action incurs a unit cost:*

$$\rho_q(a_3, v) = 1.$$

Further, there is a single capacity cost (money): $\kappa = \{m\}$; *the cost of servicing the vehicle once is* \$1000*:*

$$\kappa_q(v, m) = 1000,$$

and the agent has a total budget of \$3000*:*

$$\widehat{\kappa}(m) = 3000.$$

Then, the two formulations (5.1) and (5.2) would map to problems where the agent wants to find a policy that maximizes rewards, subject to conditions that either: i) the total expected discounted cost for servicing the vehicle does not exceed \$3000, *or ii) the probability of the total service cost exceeding* \$3000 *is bounded by some probability threshold. The former constraint might be sensible if the company has a fleet of delivery vehicles and is concerned with the total service cost, where the global budget constraints are satisfied if the expected service cost for one vehicle is bounded as above.*

Notice that we can immediately conclude that the optimal unconstrained policy from Example 2.1 violates the constraints on the expected service cost, because for that policy and uniform initial conditions, $x(s_1, a_2) = 4.9$, *which means that it incurs an expected cost of* 4900. ■

5.2 Problem Properties and Complexity

We begin by discussing problem (5.1) where constraints are imposed on the expected capacity costs. First of all, let us note that (5.1) is a slight generalization of a well-studied (e.g., (Kallenberg, 1983; Puterman, 1994)) constrained MDP with a linear cost function, which is defined as an n-tuple $\langle \mathcal{S}, \mathcal{A}, p, r, w, \widehat{w}, \alpha \rangle$, where $\mathcal{S}, \mathcal{A}, p, r, \alpha$ are the standard components of an MDP as discussed above, $w : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is a cost function defined for all state-actions pairs, and $\widehat{w}$ is the bound on the expected total cost, which defines feasible policies. A solution to this problem is a policy that maximizes the expected total reward, while ensuring that the total expected cost does not exceed $\widehat{w}$.

This is almost identical to (5.1), with the only difference that our model has an additional mapping of resources to capacities (with bounds on capacities), instead of costs defined directly on state-action-pairs (with bounds imposed directly on costs).

Clearly, (5.1) is a generalization of the standard problem, since we can always reduce the standard constrained MDP to (5.1) by defining one capacity per resource and by setting all capacity costs to one. Since ours is a very simple generalization, all properties of the standard constrained MDP carry over to our problem. We briefly discuss these known results here for completeness, as they apply to our model and problem formulation.

Kallenberg (1983) showed that a standard contracting MDP with linear constraints always has an optimal stationary policy. However, the existence of uniformly optimal policies is not guaranteed. Furthermore, unlike the unconstrained problems and our problem with non-consumable resources, deterministic policies are not always optimal for constrained MDPs with linear constraints. We summarize these existing and well-known results (using the terms of our problem) in the following statement.

Theorem 5.2. *Given an MDP $M = \langle \mathcal{S}, \mathcal{A}, p, r, \mathcal{Q}, \rho_q, \mathcal{C}, \kappa_q, \widehat{\kappa}, \alpha \rangle$ with consumable resources and capacity constraints, the policy optimization problem (5.1) always has an optimal stationary randomized policy, but there does not always exist a policy that is uniformly optimal.*

We illustrate this statement by a simple example that does not have a uniformly optimal policy, and for which, under some initial conditions, the optimal policy requires randomization.

Example 5.3. *Consider the simple MDP from Figure 5.1. The MDP has two states (s_1, s_2), three action (a_1, a_2, a_3), and a sub-stochastic transition matrix with probability of* 0.1 *of leaving the system at each step (which is equivalent to an MDP with a stochastic matrix and a discount factor of* 0.9*). The optimal unconstrained policy for this problem is $\pi_u = [(0,0,1),(0,0,1)]$, (i.e., to execute action a_3 in both states), which yields an expected discounted reward of* 5 *if the agent starts in s_1, and a reward of* 10 *if the agent starts in s_2.*

Let us now add resource constraints to the problem. Suppose that there is a single resource (q_1), and action costs are as shown in Figure 5.1:

$$\rho_q(a_1, q_1) = 0,\ \ \rho_q(a_2, q_1) = 1,\ \ \rho_q(a_3, q_1) = 2,$$

Let us also say that there is a single capacity cost (c_1), the resource q_1 has a unit cost ($\kappa_q(q_1, c_1) = 1$), and the upper bound on the expected total cost is $\widehat{\kappa}(c_1) = 12$.

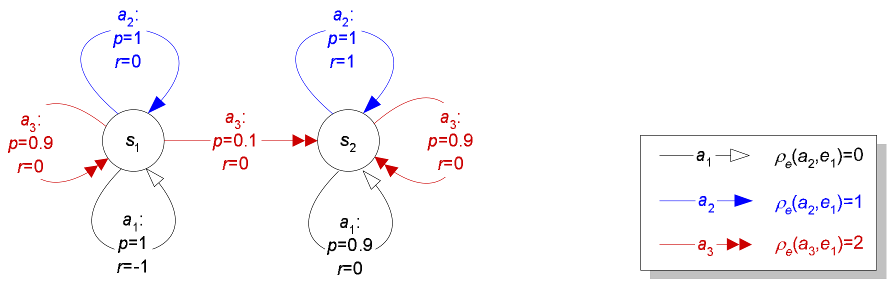


Figure 5.1: Optimal policies for problems with constraints on consumable resources require randomization and are not uniformly optimal.

Let us show that this example does not have a uniformly optimal solution, and that under some initial conditions deterministic policies are suboptimal. Suppose the agent starts in state s_2 *(*$\alpha = [0,1]$*). Then, the optimal unconstrained policy* π_u *will have the expected capacity cost of* 10*, which satisfies the constraint (expected cost is less than* $\widehat{\kappa}(c_1)$*).*

However, if the agent starts in s_1*, the cost of the unconstrained optimal policy will be higher, because it will incur a high cost for executing* a_3 *in* s_1 *before reaching* s_2*. In fact,* π_u *will have an expected cost of* 15*, which violates the constraint (*$15 > \widehat{\kappa}(c_1)$*). Under such conditions, the optimal policy will be* $\pi = [(0,0,1),(0.6,0.4,0)]$*, i.e., it will be necessary to randomize in* s_2 *between the free but useless* a_1 *and the expensive but rewarding* a_2*.* ■

It is well known (Kallenberg, 1983; Puterman, 1994) that the standard constrained MDPs with constraints on the expected cost can be solved in polynomial time using linear programming. In a similar manner, we can show that (5.1) can be solved in polynomial time using linear programming (the LP itself is given in the next section).

We now analyze the problem with probabilistic constraints (5.2).

Theorem 5.4. *The following problem is NP-hard. Given a discounted MDP* $M = \langle \mathcal{S}, \mathcal{A}, p, r, \mathcal{Q}, \rho_q, \mathcal{C}, \kappa_q, \widehat{\kappa}, \alpha \rangle$ *with consumable resources and capacity constraints and discount factor* γ*, a rational number* Y*, and probability bounds* $P_0(c)$*, does there exist a stationary policy* π*, such that it satisfies the probabilistic bounds on the capacity costs and the expected total reward of* π*, given* α*, equals or exceeds* Y*? In other words, does there exist a stationary policy* $\pi \in \Pi^{\text{SR}}$ *such that:*

$$\mathbb{P}\Big[\sum_q \kappa_q(q,c) X_\rho(q) > \widehat{\kappa}(c) \Big| \pi, \alpha\Big] \leq P_0(c) \quad \forall c \in \mathcal{C}, \tag{5.3}$$

and

$$U_\gamma(\pi,\alpha) \geq Y. \tag{5.4}$$

Proof. We show NP-hardness of the problem via a reduction from Hamiltonian cycle (HC) (Garey & Johnson, 1979). HC asks whether, for a given directed graph $G(\mathcal{V},\mathcal{Z})$, there exists a path of length $|\mathcal{V}|$ that begins in a given starting state v_1, visits every state exactly once, and returns back to the starting state. Our reduction is illustrated in Figure 5.2 and proceeds as follows.

For any graph $G(\mathcal{V},\mathcal{Z})$ with $\mathcal{V} = \{v_i\}$, $i = [1,n]$ and $\mathcal{Z} = \{z_k\}$, $k \in [1,m]$ edges, let us construct an MDP with a state space $\mathcal{S} = \{s_i\}$, $i \in [0,n]$ that has a state s_i that

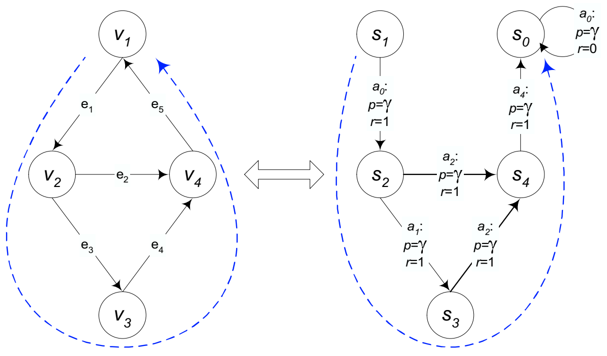


Figure 5.2: Reduction of HC to MDP with consumable resources and probabilistic constraints.

corresponds to every vertex v_i and one additional state s_0. Similarly, let the action space be $\mathcal{A} = \{a_k\},\ i \in [0, m]$, i.e., there is one action for every edge z_k and an additional action a_0. As a convenience, from now on, let us say that, for any state s_i, only the actions that correspond to edges that lead from v_i are executable in s_i (in practice, we can always set the rewards and costs of our MDP in a way that prohibits the execution of all other actions in s_i).

Let us define the transition function as follows. For every edge $z_k = (v_i, v_j),\ j \neq 1$ (i.e., all edges except those leading into v_1), we add a deterministic transition from s_i to s_j via action a_k with probability 1:

$$\forall\ i \in [1, n],\ j \in [1, n]:$$
$$p(s_j|s_i, a_k) = \begin{cases} 1 & \text{if } z_k = (v_i, v_j) \text{ i.e., there is an edge } (v_i, v_j); \\ 0 & \text{otherwise.} \end{cases} \tag{5.5}$$

For every edge $z_k = (v_i, v_1)$ leading into v_1, we add a transition that leads from s_i to our "extra" state s_0:

$$\forall\ i \in [1, n]:$$
$$p(s_0|s_i, a_k) = \begin{cases} \gamma & \text{if } z_k = (v_i, v_1) \text{ i.e., } a_k \text{ corresponds to edge } (v_i, v_1); \\ 0 & \text{otherwise.} \end{cases} \tag{5.6}$$

Similarly to the above, the only action executable in s_0 is a_0, and it leads back to s_0:

$$\forall\ j \in [0, n]: \quad p(s_j|s_0, a_0) = \begin{cases} 1 & \text{if } j = 0; \\ 0 & \text{otherwise.} \end{cases} \tag{5.7}$$

The above creates a unique action a_k for every edge $z_k = (v_i, v_j)$ in the HC with a transition from state s_i to state s_j, except the edges that lead into the starting vertex v_1, which are mapped to actions that lead to s_0 instead. We have thus split the starting vertex into two states: s_1 and s_0, where s_1 inherited all outgoing edges from v_1, and s_0 got all the edges incoming into v_1. State s_0 is a "sink" state, and this construction helps us reduce the finite-step HC to an infinite horizon MDP.

We now define the cost functions ρ_q and κ_q that will allow us to ensure that each state is visited no more than once. Let us define the resource set $\mathcal{Q} = \{q_i\},\ i \in [1,n]$ so that there is a unique resource q_i for every vertex v_i in the HC. Let us also define a unique capacity cost for every resource: $\mathcal{C} = \{c_i\},\ i \in [1,n]$, and let each resource only require a unit of the corresponding capacity:

$$\forall i \in [1,n]\ j \in [1,n]: \quad \kappa_q(q_i, c_j) = 1 \iff i = j. \tag{5.8}$$

The resource costs of all actions are defined so that all actions that correspond to edges from vertex v_i require a unit of resource q_i and none of the other resources:

$$\forall\ k \in [1,m],\ i \in [1,n]: \quad \rho_q(a_k, q_i) = \begin{cases} 1 & \text{if } z_k = (v_i, v_j); \\ 0 & \text{otherwise.} \end{cases} \tag{5.9}$$

Also, we let a_0 (only executable in s_0) be a "free" action that does not require any resources:

$$\forall\ i \in [1,n]: \quad \rho_q(a_0, q_i) = 0. \tag{5.10}$$

Given our definition of the cost functions, it is easy to see that the total capacity cost c_i of any trajectory will equal the total number of visits to state s_i under that policy. Indeed, the only resource that requires c_i is q_i, and q_i is only consumed by actions performed in state s_i.

Given the resource requirements and capacity costs as defined above, we can formulate a probabilistic constraint that ensures that every state has a zero probability of being visited more than once:

$$\mathbb{P}\Big[\sum_q \kappa_q(q,c) X_\rho(q) > 1 \Big| \pi, \alpha\Big] \le 0, \quad \forall c \in \mathcal{C}. \tag{5.11}$$

The above constraint is violated if and only if there are some states (recall the one-to-one correspondence between states and capacities c) that have positive probability of being visited more than once.

The above allows us to rule out policies that might visit states more than once. To complete the reduction we prohibit policies that might not visit some states at all by setting the rewards r and the threshold Y as follows. For this purpose, it will suffice to choose the threshold

$$Y = \sum_{i=0}^{n} \gamma^i = \frac{1-\gamma^{n+1}}{1-\gamma}, \tag{5.12}$$

and a trivial reward function that assigns a unit reward to every state action pair (except a_0 and s_0, which produces no reward).

$$\begin{aligned} r(s_i, a_k) &= 1, \qquad \forall\ i \in [1,n],\ k \in [1,m]; \\ r(s_0, a_0) &= 0. \end{aligned} \tag{5.13}$$

As shown above, our cost functions ρ_q and κ_q ensure that the only feasible policies are transient policies that have zero probability of visiting the same state twice (except, of course, s_0). Furthermore, the reward function ensures that the only policies that yield an expected payoff of more than Y are policies that visit at least n states. Therefore, our

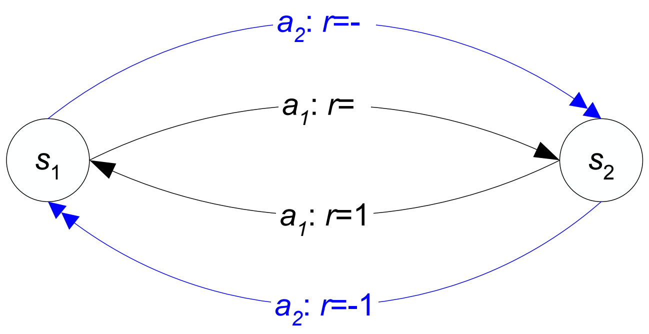


Figure 5.3: MDPs with probabilistic constraints are non-linear and non-convex in the occupation measure coordinates.

constrained MDP has a solution whose value is no less than Y if and only if there exists a transient policy that starts in state s_1, has a nonzero probability of visiting all states (but no more than once), and ends in the sink state s_0. This, in turn, can happen if and only if the original graph $G(\mathcal{V}, \mathcal{Z})$ has a Hamiltonian cycle. ■

Notice that Theorem 5.4 only provides a lower bound on the complexity of solving MDPs with probabilistic constraints, as we did not show the problem to be *in* NP. The reason for this is that simply checking whether the probabilistic constraints are violated is a hard problem in itself (since it requires reasoning about the pdf of the total cost, instead of just the mean values), and we do not have a polynomial-time algorithm for it. However, the exact complexity of checking probabilistic constraints is currently not known (if the latter were known to be in some class X, the complexity of the problem could be bounded from above by NP^X). Although there exist algorithms (some even polynomial) for computing the pdf of the cumulative reward for MDPs for finite time intervals (Donatiello & Grassi, 1991; de Souza e Silva & Gail, 1989; de Souza e Silva, Gail, & Campos, 1995), to the best of our knowledge there are no general algorithms and complexity results for infinite-horizons.

Finally, let us make the following observation, regarding the complexity of the MDP with probabilistic constraints.

Observation 5.5. *The optimization problem (5.2) is non-linear and non-convex.*

The observation is demonstrated via the following example.

Example 5.6. *Consider the simple two-state MDP with deterministic transitions shown in Figure 5.3 and two policies:*

$$\pi_1(s_1, a_1) = 1, \qquad \pi_1(s_2, a_2) = 1,$$
$$\pi_2(s_1, a_2) = 1, \qquad \pi_2(s_2, a_1) = 1.$$

The distribution of total discounted (with factor γ) rewards for both policies is $\delta(0)$, i.e., both policies achieve zero total reward with certainty.

Now consider a mixture of the two policies:

$$\pi_3(s, a) = \frac{1}{2}\pi_1(s, a) + \frac{1}{2}\pi_2(s, a),$$

which maps to the same mixture of the corresponding occupation measures.

The distribution of the total reward for π_3 is a discrete approximation of a normal distribution (for sufficiently large γ) with mean 0 and nonzero variance. Therefore, for any $\widehat{R}$, the probability that π_3 achieves a total reward greater than $\widehat{R}$ is nonzero, which shows that the constraint is non-linear. Further, there exists a p_0 such that a constraint $\mathbb{P}[R > \widehat{R}] < p_0$ is satisfied for both π_1 and π_2, but not π_3, which shows that the constraint is non-convex. ■

5.3 Solution for MDPs with Constraints on the Expected Resource Costs

In this section and the following one, we discuss ways of solving problems with consumable resources and capacity constraints whose properties and complexity were discussed in the previous section.

We start with the simplest problem with constraints on the expected capacity costs (5.1), which we reproduce here for convenience (recall that we use $X_\rho(q)$ to denote the random total usage of resource q):

$$\begin{aligned}&\max U_\gamma(\pi, \alpha)\\ &\text{subject to:}\\ &\sum_q \kappa_q(q,c)\mathbb{E}\Big[X_\rho(q)|\pi,\alpha\Big] \leq \widehat{\kappa}(c), \quad \forall c \in \mathcal{C}.\end{aligned}$$

As mentioned in the previous section, this is only a slight generalization of the thoroughly studied standard constrained MDP (Kallenberg, 1983; Puterman, 1994) with scalar costs defined on the state-action pairs, and bounds on the expected total values of these costs. There exists a well-known linear programming solution to a contracting MDP with constraints on the expected total costs (Altman & Shwartz, 1991; Altman, 1999; Kallenberg, 1983; Puterman, 1994). In fact, since the total expected costs can be expressed as linear functions of the occupancy measure (just like the total expected reward), the constraints on the expected costs are linear and can simply be added to the standard dual LP (2.10) without breaking its linearity.

Similarly to this well-known method, we can formulate our constraints on the expected capacity costs as linear functions of the occupation measure x and formulate (5.1) as an LP:

$$\begin{aligned}&\max \sum_s \sum_a r(s,a)x(s,a)\\ &\text{subject to:}\\ &\sum_a x(\sigma,a) - \gamma \sum_s \sum_a x(s,a)p(\sigma|s,a) = \alpha(\sigma), && \forall \sigma \in \mathcal{S};\\ &\sum_q \kappa_q(q,c) \sum_a \rho_q(a,q) \sum_s x(s,a) \leq \widehat{\kappa}(c), && \forall c \in \mathcal{C};\\ &x(s,a) \geq 0, && \forall s \in \mathcal{S}, a \in \mathcal{A}.\end{aligned} \tag{5.14}$$

An optimal policy can be obtained from this LP in the standard way (2.11), which, in general, yields randomized policies (because a basic feasible solution to (5.14) has more that $|\mathcal{S}|$ nonzero variables).

Example 5.7. *Let us consider the problem with constraints on the total expected capacity costs from Example 5.1. The constraint on the capacity cost from the LP (5.14) becomes:*

$$(1000)(1)x(s_2, a_3) \leq 3000,$$

and the optimal solution to the LP (if the agent starts in s_1) is (showing only the non-zero components of the occupation measure):

$$\begin{aligned} &x(s_1, a_2) = 3.93, \\ &x(s_2, a_2) = 2.82, \qquad x(s_2, a_3) = 3.00, \\ &x(s_3, a_4) = 0.25. \end{aligned}$$

This maps to a policy that randomizes between a_2 and a_3 (delivering appliances and servicing the truck) in state s_2 as follows:

$$\pi(s_2, a_2) = 0.48, \qquad \pi(s_2, a_3) = 0.52$$

and yields an expected reward of 94.7 (while the optimal unconstrained policy produced a reward of 95.3 for these initial conditions).

Notice that the randomization in state s_3 does not necessarily mean that each delivery agent has to flip a coin before deciding whether to deliver appliances(a_2) or service the truck (a_3). For example, in a company with a large pool of vehicles, the above would be equivalent to having 48% of vehicles assigned to delivering appliances, while 52% of the vehicles get serviced after the first year of exploitation (action a_2 in state s_2).

In general, by allowing randomized policies, we avoid the need to include information about the current incurred cost in the state, while formulating and solving the optimization problem. The actual policy that is executed can base its decisions deterministically on other hidden factors (e.g., each vehicle can have a deterministic schedule for doing different deliveries on different days), as long as the new policy is equivalent to the randomized policy computed as the solution to the MDP (e.g., the resulting frequencies of appliance deliveries and vehicle servicing are 48% and 52%, respectively). ■

5.4 Probabilistic Constraints: Linear Approximation

We now discuss possible approaches to solving the risk-sensitive problem (5.2) where constraints are imposed on the probability that the total capacity cost exceeds a given bound. We reproduce the problem here for convenience (recall that $X_\rho(q)$ is the random total consumption of resource q, and $P_0(c)$ is the user-defined bound on the probability that the total capacity cost c exceeds $\widehat{\kappa}(c)$):

$$\begin{aligned} &\max U_\gamma(\pi, \alpha) \\ &\text{subject to:} \\ &\mathbb{P}\Big[\sum_q \kappa_q(q, c) X_\rho(q) \geq \widehat{\kappa}(c) \Big| \pi, \alpha\Big] \leq P_0(c), \quad \forall c \in \mathcal{C}. \end{aligned}$$

As shown in the earlier section, this problem is NP-hard, and unfortunately we do not have a good way of solving for the exact solution. The challenge is that the pdf of the total cost $X_\rho(q)$ is not easily available. Indeed, the total cost is a sum of a random number of dependent random variables (cost consumed at each time step), and computing the pdf of their sum is not an easy task.

In the rest of the chapter, we present two approximations to (5.2): a linear approximation in the current section, and a polynomial approximation in Section 5.5.

In the rest of this section we assume that resources are non-replenishable (i.e., $\rho_q(a,q) \geq 0 \ \forall a,q$) and that capacity costs are nonnegative (i.e., $\kappa_q(q,c) \geq 0 \ \forall q,c$). Under these assumptions, we can employ the Markov inequality to bound the probability that the total cost exceeds the given limit. Indeed, for a nonnegative random variable Z, the following holds (where $b > 0$ is some constant):

$$\mathbb{P}\Big[Z \geq b\Big] \leq \frac{\mathbb{E}[Z]}{b}. \tag{5.15}$$

Thus, we can bound the probability that the total capacity cost exceeds $\widehat{\kappa}(c)$ as follows:

$$\mathbb{P}\Big[\sum_q \kappa_q(q,c) X_\rho(q) \geq \widehat{\kappa}(c) \Big| \pi, \alpha\Big] \leq \frac{1}{\widehat{\kappa}(c)} \sum_q \kappa_q(q,c) \mathbb{E}\Big[X_\rho(q) \Big| \pi, \alpha\Big], \tag{5.16}$$

where the expected value of the total resource cost X_ρ can be expressed as a linear function of the occupation measure $x(s,a)$:

$$\mathbb{E}\Big[X_\rho(q)\Big|\pi,\alpha\Big] = \sum_a \rho_q(a,q) \sum_s x(s,a). \tag{5.17}$$

Therefore, if we bound the right-hand side of (5.16) by $P_0(c)$, we will have the following linear approximation to the original probabilistic constraint:

$$\frac{1}{\widehat{\kappa}(c)} \sum_q \kappa_q(q,c) \sum_a \rho_q(a,q) \sum_s x(s,a) \leq P_0(c). \tag{5.18}$$

To sum up, we can construct the following linear approximation to (5.2):

$$\begin{aligned}
&\max \sum_s \sum_a r(s,a) x(s,a) \\
&\text{subject to:} \\
&\sum_a x(\sigma,a) - \gamma \sum_s \sum_a x(s,a) p(\sigma|s,a) = \alpha(\sigma), && \forall \sigma \in \mathcal{S}; \\
&\frac{1}{\widehat{\kappa}(c)} \sum_q \kappa_q(q,c) \sum_a \rho_q(a,q) \sum_s x(s,a) \leq P_0(c), && \forall c \in \mathcal{C}; \\
&x(s,a) \geq 0, && \forall s \in \mathcal{S}, a \in \mathcal{A};
\end{aligned} \tag{5.19}$$

However, since the Markov inequality provides a very rough upper bound, the left-hand side of (5.18) will tend to overestimate the probability of exceeding the cost bounds. Therefore, (5.19) will produce suboptimal policies that will be very conservative in their resource usage.

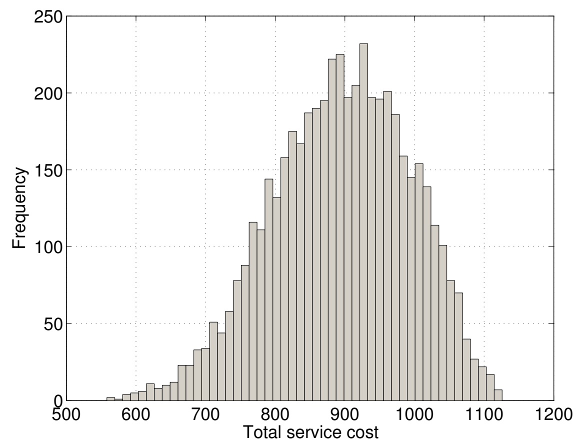


Figure 5.4: Sub-optimality of linear Markov approximation for the delivery example.

```
Algorithm 1: Iterative approach to solving (5.2)
    repeat
        Solve (5.19) to get policy $\pi$
        Evaluate actual probability that cost bounds exceed $P_0(c)$ under $\pi$
        Adjust $P_0(c)$
    until $\pi$ is good enough or $P_0(c)$ converges
```

Example 5.8. *Let us consider the delivery problem from earlier examples 5.1 and 5.7. Suppose that the constraint for this problem is that the probability of violating the budget limitations should be no greater than* $P_0 = 0.3$.

If we apply the linear Markov approximation to this problem, we obtain the following solution:

$$x(s_1, a_2) = 2.3,$$
$$x(s_2, a_2) = 0.62, \qquad x(s_2, a_3) = 0.9,$$
$$x(s_3, a_4) = 0.56.$$

This occupation measure corresponds to the policy that mixes between a_2 *and* a_3 *much more conservatively than in Example 5.7:*

$$\pi(s_2, a_2) = 0.87, \qquad \pi(s_2, a_3) = 0.13,$$

which is to be expected.

However, if we consider the resulting distribution of the total cost for this policy (shown in Figure 5.4), it is evident that the policy is overly conservative: its probability of violating the budget constraints is practically zero (while the problem formulation allowed $P_0 = 0.3$*), which leads to a sub-optimal reward of* 94.05. ■

To alleviate this problem that arises from the overly pessimistic nature of the Markov bound, we use a simple generate-and-test approach, where we iteratively change the bound P_0 to find a reasonable approximation to the original problem. The generic process is illustrated in Algorithm 1.

The individual steps in Algorithm 1 can be accomplished as follows. The simplest way of evaluating a policy in step (2) is to run a Monte Carlo simulation of the Markov chain

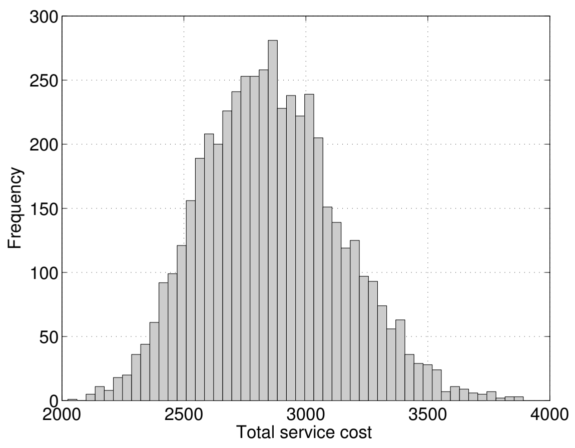


Figure 5.5: Iterative Markov approximation for the delivery example.

that corresponds to π and to check the pdf of the total cost. There are several ways one can adjust $P_0(c)$ in step (3). Again, one of the simplest procedures is to do a binary search on $P_0(c)$, decreasing it when the result of step (2) is too high, and increasing it otherwise.

Example 5.9. *Let us apply the iterative procedure in Algorithm 1 to the problem in Example 5.8. The algorithm converges to the following policy:*

$$\begin{aligned} &\pi(s_1, a_2) = 1, \\ &\pi(s_2, a_2) = 0.51, \qquad \pi(s_2, a_3) = 0.49, \\ &\pi(s_3, a_4) = 1, \end{aligned}$$

which is significantly more aggressive than the policy resulting from the single-shot Markov approximation in Example 5.8. Figure 5.5 shows the resulting cost distribution for that policy; its probability of violating the budget constraints is 0.2994 *(compare to the specified bound of* $P_0 = 0.3$*, and the practically-zero probability for the single-shot Markov approximation). The new policy also has a higher total expected reward:* 94.68 *instead of* 94.05*, obtained from the single-shot approximation.*[4] ■

An important thing to note about Algorithm 1 is that, in general, the Markov inequality does not provide a monotonic bound on the probability of exceeding the cost bounds. In other words, there could be two policies π and π' such that the actual probability of exceeding the cost bounds is lower for π than for π', but the Markov bound $1/\widehat{\kappa}(c)\mathbb{E}[X_\rho(q)|\pi, \alpha]$ for π is higher than the one for π'. However, the bound is monotonic under some conditions: for example, if the total cost is distributed normally with a constant variance for all policies. It can be shown that for a recurrent Markov chain, as the discount factor γ approaches 1, the distribution of the total reward approaches the normal distribution, and in our experiments (discussed in the next section), the distributions were very close to normal and the change in variance was not very high.

However, even if the above condition on monotonicity of the Markov bound does hold, Algorithm 1 is still not guaranteed to converge to the optimal solution of (5.2), as the

[4]The absolute difference is small in this example, because a_3 is free and yields only a slightly lower reward than a_2. In other domains, the benefit could be arbitrarily large.

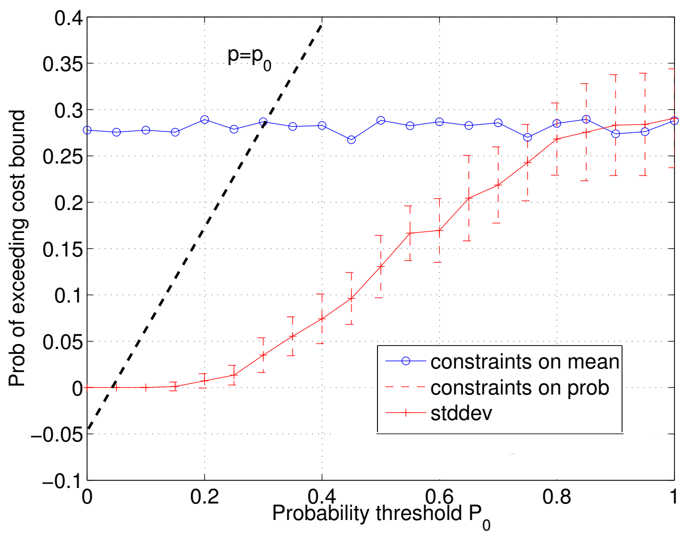


Figure 5.6: Probability of exceeding cost bounds for the linear Markov approximation.

latter is non-convex and cannot be approximated via a sequence of increasingly relaxed linear programs, as done in Algorithm 1. However, informally speaking, monotonicity does mean that Algorithm 1 finds the "tightest" linear approximation of (5.2). In Section 5.5 we present a polynomial approximation that creates a better fit for the feasible region (but at the expense of a significant increase in complexity).

5.4.1 Experimental Evaluation

In this section we present some experimental observation about the linear Markov approximation of problem (5.2) as well as the benefits of the iterative bound adjustment described in Algorithm 1.

The goal of our first set of experiments is to see how harsh of a bound the Markov inequality provides. To answer this question, we have conducted experiments on a set of randomly generated problems. The generated MDPs shared some common properties, among which the most interesting ones are the following (the values for our main experiments are given in parentheses):

- $|\mathcal{S}|, |\mathcal{A}|, |\mathcal{Q}|$. The total number of states, actions, and resources, respectively. (20, 20, 2)
- $M_R = \max(r)$ Maximum reward. Rewards are assigned from a uniform distribution on $[0, M_R]$. (10)
- $M_{\rho_q} = \max(\rho_q)$. Maximum resource cost. Resource costs are assigned from $[0, M_{\rho_q}]$ based on a distribution of r and $\phi(\rho, r)$ (described below). (10)
- $\phi = \phi(\rho_q, r)$. Correlation between rewards and resource costs; better actions are typically more costly. ([0.8,1])
- κ_q. Capacity costs of the resources. All resources had unit costs. (1)

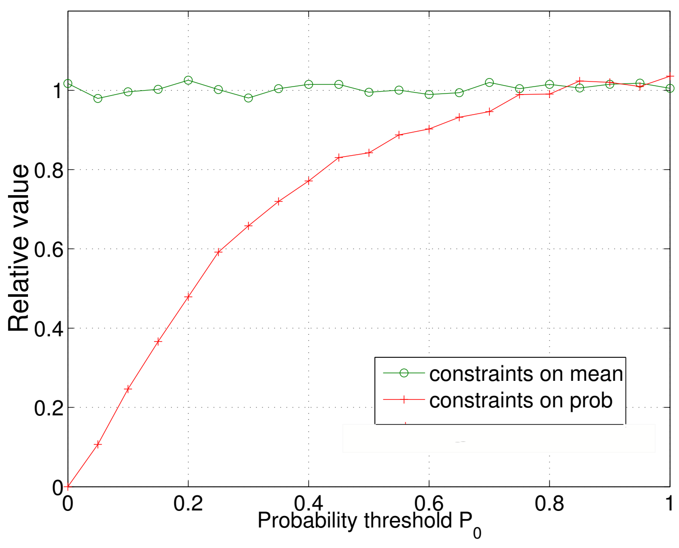


Figure 5.7: Quality of single-shot Markov approximation.

- $[\widehat{\kappa}_{min}, \widehat{\kappa}_{max}]$. Upper bounds on resource amounts are assigned according to a uniform distribution from this range. ([200, 300])
- γ. Discount factor.([0.95,0.99])

We are interested in the behavior of the Markov approximation as a function of the probability threshold P_0. Therefore, we have run a number of experiments for various values of $P_0 \in [0, 1]$. We have gradually increased P_0 from 0 to 1 in increments of 0.05, and for each value, generated 50 random models and solved them using two methods: i) an MDP with constraints on the mean cost, and ii) a Markov approximation of the probabilistic constraint as discussed above. We then evaluated via a Monte-Carlo simulation each of the resulting policies in terms of expected reward and probability of exceeding the cost threshold.

Figure 5.6 shows a plot of the actual probability of exceeding the capacity cost bounds as a function of the probability threshold P_0. The data points are averaged over the (ten) runs for a particular value of P_0. The curve that corresponds to the Markov approximation also shows the standard deviation for the runs. The other data have very similar variance, so for readability we omit the standard deviation from these curves.

Obviously, P_0 has no effect on the MDP with constraints on the mean cost, but it does affect to a large extent the solutions to the problem with probabilistic constraints. We can see that the probability of exceeding the cost threshold for the solutions produced by the linear Markov approximation is always below P_0 (as it should be). Also, it is clear that the Markov bound is a very strong condition and it greatly underestimates the probability of exceeding the cost threshold.

The total expected (normalized) rewards obtained by these policies are shown in Figure 5.7. For the policies obtained by the Markov approximation, the actual rewards are shown that do not necessarily equal the expected rewards that are used during the optimization process. This is because only the runs that did not exceed the capacity costs were included in the plot, and policies were not penalized for violating the cost constraints.

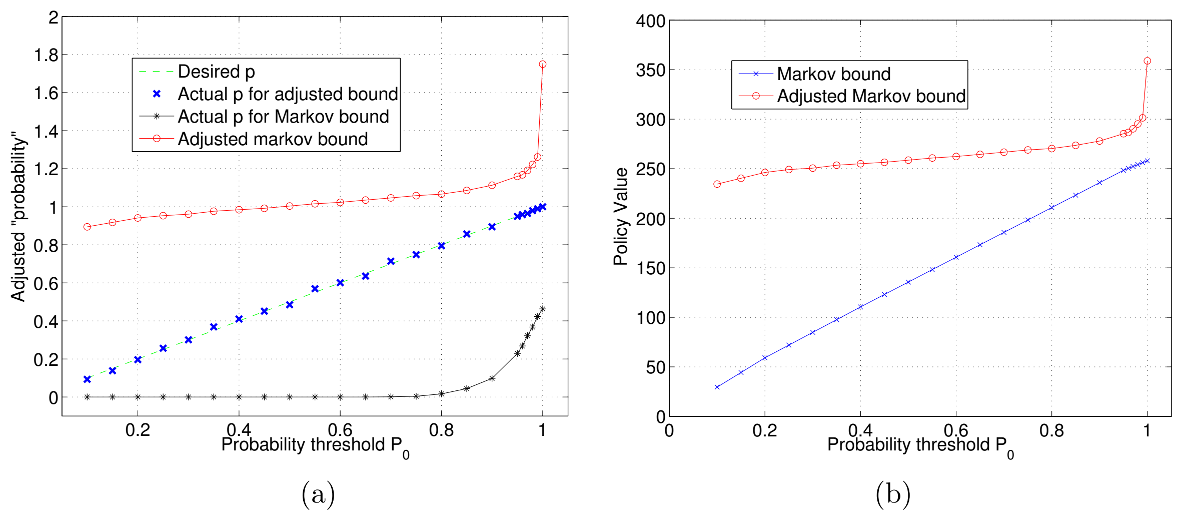


Figure 5.8: Effect of iteratively adjusting the Markov bound via Algorithm 1. (a): The bottom curve is the actual probability of exceeding the cost bounds, as computed by a single-shot Markov approximation. The dotted line is $p = p_0$ and indicates the desired constraint level. The "x"s show the actual probability of exceeding cost bounds for the Markov bound adjusted via Algorithm 1, and the value of the adjusted parameter P_0 is given by the top line with "o" markers. (b): the corresponding effect on policy value.

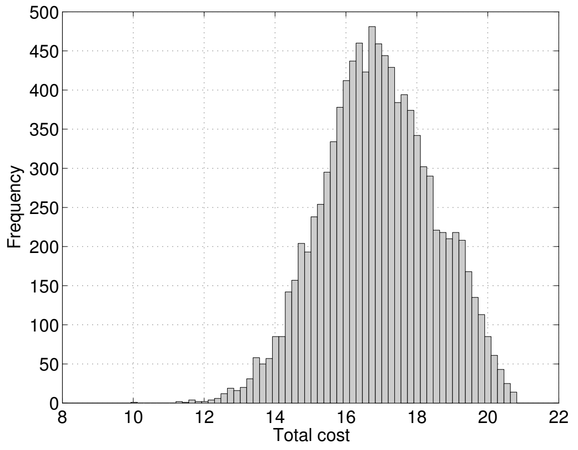
This plot shows the effects of the Markov approximation on the total expected rewards received by the agent.

The bottom line is that the actual probability of exceeding cost bounds for the Markov approximation is significantly lower than the threshold P_0 (the Markov curve is much lower than the admissible $p = P_0$ curve in Figure 5.6), which translates into low total reward, as illustrated in Figure 5.7.

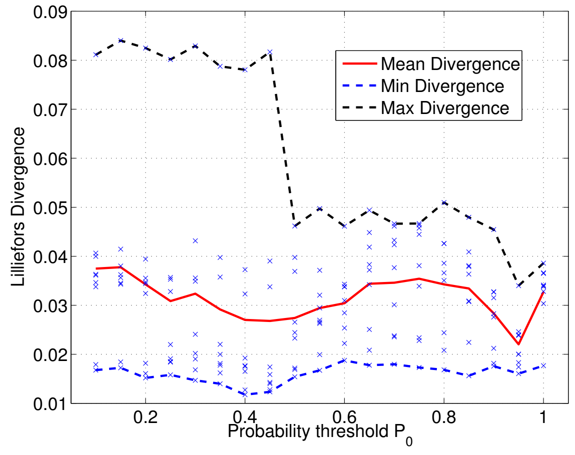
The purpose of Algorithm 1 was to address this shortcoming of the Markov approximation and improve the quality of the resulting solution. We experimentally evaluated the benefit of using this iterative adjustment procedure on the single-agent grid-world delivery domain from Section 3.2.1. For this version of the domain, we got rid of the non-consumable resources and added a single consumable resource, time, and imposed a constraint that the expected total time for a policy is bounded by one half of the time required for the optimal unconstrained policy. Figure 5.8 shows the results. Figure 5.8a demonstrates that the approach of iteratively adjusting the Markov bound does provide a much more accurate estimate of the actual probability of exceeding the capacity cost bounds than the single-shot Markov approximation. Figure 5.8b illustrates the corresponding effect on policy value — the policies obtained via Algorithm 1 perform much better than the ones resulting from the single-shot linear approximation (5.19).

Finally, as a justification for Algorithm 1, we provide some evidence that the Markov bound is almost monotonic. To that effect, Figure 5.9 presents evidence that the distributions over the total cost in all of our experiments were almost normal and without large deviations in their variance.

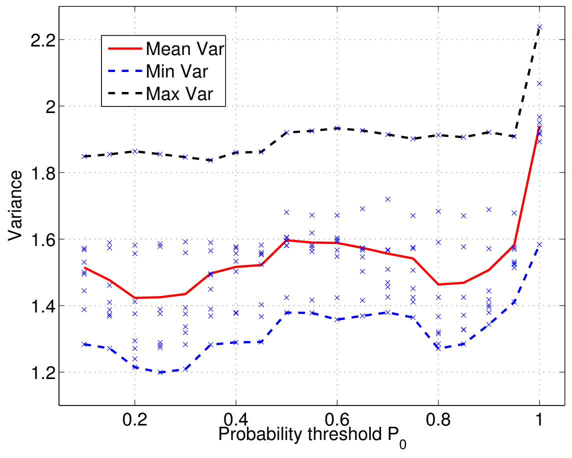
As mentioned earlier, this monotonicity of the Markov bound does not guarantee



(a)



(b)



(c)

Figure 5.9: Distribution of total cost, comparison to normal. (a): A typical distribution of total discounted cost. (b): Lilliefors divergence for the empirical data (the Kolmogorov-Smirnoff divergence between the empirical data and a normal with the empirical mean and standard deviation). (c): Values of variance in the distribution of total cost. All distributions are very close to normal, and the spread in variance is not very large.

convergence of Algorithm 1 to the optimal solution of the MDP with probabilistic constraints (5.2) because of the non-convexity of the latter. The above empirical evidence is merely meant to indicate that the linear approximation obtained via the iterative procedure of Algorithm 1 is "tight". The following section presents methods for dealing with the non-linearity and non-convexity of the probabilistic constraints.

5.5 Probabilistic Constraints: Polynomial Approximation

In this section we present a more general polynomial approximation to the problem with probabilistic constraints (5.2). To simplify the following discussion, in this section we assume that there is a single resource type q, there is a single capacity c, and the resource has unit capacity cost: $\kappa_q(q, c) = 1$. This is done purely for notational convenience, and all of the results of this section generalize in a straightforward way to the more general case of multiple resources and capacities. Given the above assumptions, the problem (5.2) reduces to the following:

$$\begin{aligned} &\max U_\gamma(\pi, \alpha) \\ &\text{subject to:} \\ &\mathbb{P}\Big[C > C_0 \Big| \pi, \alpha\Big] \leq P_0, \end{aligned} \tag{5.20}$$

where we $C = X_\rho(q)$ is the random total resource cost and C_0 is the given threshold.

In the rest of the section, we focus on solving (5.20), but the methodology presented here is more general and supports reasoning about *probability distributions* of rewards and costs instead of just their *expected values*. Reasoning about distributions of rewards and costs is a valuable tool that allows us to attack a rich variety of MDPs. For instance, it can be applied to the following classes of problems.

- MDPs with general utility functions, where instead of maximizing the expected total discounted reward: $\max_\pi \mathbb{E}_\alpha\big[R_\gamma(\pi)\big]$, the goal is to maximize the expected value of a given utility function of the reward: $\max_\pi \mathbb{E}_\alpha\big[U(R_\gamma(\pi))\big]$. This MDP model can be used to express nonlinear preferences and variable risk attitudes.

- MDPs with the optimization criterion that maximizes the probability that the total reward exceeds a threshold value: $\max_\pi \mathbb{P}_\alpha\big[R_\gamma(\pi) \geq \widehat{R}\big]$. This MDP model is useful in *satisficing* (Simon, 1957) scenarios, where the utility for increasing the expected reward above a threshold is negligible, while increasing the probability of exceeding that threshold is of vital importance.

- MDPs with constraints on cost variance. In particular, such models have been studied in the easier context of MDPs with average-per-step rewards and costs. For example, Sobel (1985) analyzed the model that constrains the expected cost and maximizes the mean-variance ratio of the reward. Huang and Kallenberg (1994) developed a unified approach to handling variance via an algorithm based on parametric linear programming. However, dealing with variance in infinite horizon MDPs with long-term rewards and costs is a more difficult task, and (to the best of our knowledge) no algorithms exist for such models.

Also, let us note that the problem (5.20), which we address in this section, is similar to models that have been previously studied in the context of average-per-step MDPs. Ross and Varadarajan (1989, 1991) developed an approach where constraints are placed on the actual *sample-path* costs of a policy. In their work, the space of feasible solutions is constrained to the set of policies whose probability of violating the constraints asymptotically approaches zero. This model does not allow arbitrary probability thresholds, and furthermore, the model with total rewards and costs is considerably more difficult than the average-per-step model.

Also, several approaches (Howard & Matheson, 1972; Koenig & Simmons, 1994; Marcus, Fernandez-Gaucherand, Hernandez-Hernandez, Colaruppi, & Fard, 1997) to modeling risk-sensitive utility functions have been proposed that work by transforming risk-sensitive problems into equivalent risk-neutral problems, which can then be solved by dynamic programming. However, this transformation only works for a certain class of utility functions. Namely, this has been done for exponential utility functions that are characteristic of agents that have "constant local risk aversion" (Pratt, 1964) or obey the "delta property" (Howard & Matheson, 1972), which says that a decision maker's risk sensitivity is independent of his current wealth. This approximation has a number of very nice analytical properties, but is generally considered somewhat unrealistic (Howard & Matheson, 1972). The method developed in this section attempts to address this issue via approximate modeling of a more general class of utility functions and constraints.

5.5.1 Calculating the Probability of Exceeding Cost Bounds

In this section, for compactness, we use a slightly different notation than in the rest of this thesis:

$$\begin{aligned} &x_{sa} = x(s,a), \qquad \alpha_s = \alpha(s), \qquad \pi_{sa} = \pi(s,a), \\ &P^a_{s\sigma} = p(\sigma|s,a), \qquad \widetilde{p}_{s\sigma} = \sum_a P^a_{s\sigma}\pi_{sa} = \sum_a p(\sigma|s,a)\pi(s,a). \end{aligned}$$

To find the probability of exceeding the cost bounds, it would be very useful to know the probability density function (pdf) $f_C(C)$.[5] Then, the probability of exceeding the cost bounds could be expressed simply as

$$\mathbb{P}[C \geq C_0] = \int_{C_0}^{\infty} f_C(C)dC. \tag{5.21}$$

Unfortunately, $f_C(C)$ is not easily available. However, it is a well-known fact that under some conditions the moments of a random variable completely specify its distribution (Papoulis, 1984).[6] The k^{th} moment of a random variable x is defined as the expected value of x^k:

$$E^k_x = \mathbb{E}[(x)^k] = \int_{-\infty}^{\infty} f_x(x)(x)^k dx \tag{5.22}$$

One way to compute the pdf $f_x(x)$, given the moments E^k_x is via an inverse Legendre transform.[7] Indeed, the Legendre polynomials

$$\mathcal{P}_l(x) = \frac{1}{(2)^l l!}\frac{d^l}{dx^l}\Big((x)^2 - 1\Big)^l \tag{5.23}$$

form a complete orthogonal set on the interval $[-1, 1]$:

$$\int_{-1}^{1} \mathcal{P}_l \mathcal{P}_m = \frac{2}{2l+1}\delta_{lm}. \tag{5.24}$$

[5]Note that, in general, for an MDP with finite state and action spaces, the total costs have a discrete distribution. However, we make no assumptions about the continuity of the pdf $f_C(C)$, and our analysis carries through for both continuous and discrete density functions; in the latter case, $f_C(C)$ can be represented as a sum of Dirac delta functions: $f_C(C) = \sum_k p_k\delta(x - p_k)$.

[6]This is true when the power series of the moments that specifies the characteristic function converges, which holds in our case due to the contracting nature of the discounted Markov process and the fact that costs are finite.

[7]A more common and natural way involves inverting the characteristic function of x via a Fourier transform, but the method does not work for this problem.

Therefore, a function on that interval $[-1, 1]$ can be approximated as a weighted sum of Legendre polynomials (Abramowitz & Stegun, 1965):

$$f(x) = \sum_{l=0}^{\infty} b_l \mathcal{P}_l(x), \tag{5.25}$$

where $\mathcal{P}_l(x)$ is the l^{th} Legendre polynomial, and b_l is a constant coefficient, obtained by multiplying the polynomials by $f(x)$, integrating over $[-1, 1]$, and using the orthogonality condition:

$$b_l = \frac{2l+1}{2} \int_{-1}^{1} f(x)\mathcal{P}_l(x)dx = \sum_k a_{kl} E_x^k \tag{5.26}$$

Realizing that $\int f(x)\mathcal{P}_l(x)dx$ is just a linear combination of several moments E_x^k, we get:

$$f(x) = \sum_{l=0}^{\infty} \sum_k a_{kl} E_x^k \mathcal{P}_l(x), \tag{5.27}$$

where in the second summation, the index k runs over all powers of x present in $\mathcal{P}_l$. Therefore, for an $x \in [-1, 1]$, we can express the probability that x is greater than some x_0 as a linear function of the moments:

$$\mathbb{P}[x \geq x_0] = \int_{x_0}^{1} f_x(x)dx = \int_{x_0}^{1} \sum_{l=0}^{\infty} \sum_k a_{kl} E_x^k \mathcal{P}_l(x)dx = \sum_k \psi_k(x_0) E_x^k, \tag{5.28}$$

where $\psi_k(x_0) = \sum_l a_{kl} \int_{x_0}^{1} \mathcal{P}_l(x)dx$, in which the index l runs over all polynomials that include the k^{th} power of x.

Therefore, if we normalize C to be in the interval $[-1, 1]$, we could use the above method to express $\mathbb{P}[C \geq C_0]$ as a linear function of the moments E_C^k. Now, if we could come up with a system of coordinates $\mathbf{y}$, such that the moments E_C^k could be expressed via $\mathbf{y}$, we might be able to formulate a manageable approximation to (5.20). However, it is important to note that unless we use an infinite number of moments, the resulting program will be an approximation to the original one.

5.5.2 Computing the Moments

As mentioned in the previous section, the properties of the pdf of the total cost are not immediately obvious, as the total cost is a sum of a random number of dependent random variables. We do, however, know how the system evolves with time, i.e. given the initial probability distribution, a policy, and the corresponding transition probabilities over states, we know the probability that the system is in state s at time t — it is simply $(\widetilde{\mathbf{P}}^t\alpha)_s$, where $\widetilde{\mathbf{P}} = (\widetilde{p}_{s\sigma})$ is the probability transition matrix induced by the policy ($\widetilde{p}_{s\sigma} = \sum_a \pi_{sa} P_{s\sigma}^a$). In other words, we know the probability distribution for the random variables $n_s(t) = \{0, 1\}$, where $n_s(t) = 1$ if state s is visited at time t, and 0 otherwise.

Let us also define for every state a random variable $N_s = \sum_{t=0}^{\infty} n_s(t)$ that specifies the total number of times state i is visited. Then, the moments E_C^k of C can be expressed as linear functions of the cross-moments $E_{s_1 \ldots s_k} = \langle N_{s_1} N_{s_2} \cdots N_{s_k} \rangle$ (the expected value of the product) as follows:

$$
\begin{aligned}
E_C^1 &= \langle \sum_s c_s N_s \rangle = \sum_s c_s \langle N_s \rangle = \sum_s c_s E_s \\
E_C^2 &= \langle \Big(\sum_s c_s N_s \Big)^2 \rangle = \sum_s \sum_\sigma c_s c_\sigma \langle N_s N_\sigma \rangle = \sum_s \sum_\sigma c_s c_\sigma E_{s\sigma} \\
&\dots \\
E_C^k &= \sum_{s_1} \sum_{s_2} \dots \sum_{s_k} c_{s_1} c_{s_2} \cdots c_{s_k} E_{s_1 s_2 \dots s_k}
\end{aligned}
\tag{5.29}
$$

Let us now compute the first moments

$$
E_s = \langle \sum_{t=0}^{\infty} n_s(t) \rangle = \sum_{t=0}^{\infty} \langle n_s(t) \rangle. \tag{5.30}
$$

Recalling that $n_s(t) = \{0, 1\}$, and, therefore, its mean equals the probability that $n_s(t)$ is 1:[8]

$$
E_s = \sum_{t=0}^{\infty} \mathbb{P}[n_s(t)] = \sum_{t=0}^{\infty} (\widetilde{\mathbf{P}}^t \boldsymbol{\alpha})_s = ((\mathbf{I} - \widetilde{\mathbf{P}})^{-1} \boldsymbol{\alpha})_s, \tag{5.31}
$$

where $\mathbf{I}$ is the identity matrix, and

$$
\sum_{t=0}^{\infty} \widetilde{\mathbf{P}}^t = (\mathbf{I} - \widetilde{\mathbf{P}})^{-1}
$$

holds, because $\lim_{t \to \infty} \widetilde{\mathbf{P}}^t = 0$ for our contracting system. Multiplying by $(\mathbf{I} - \mathbf{P})$, we get:

$$
E_s - \sum_\sigma \widetilde{\mathbf{P}}_{\sigma s} E_\sigma = \alpha_s \tag{5.32}
$$

Note that the above is exactly the "conservation of probability" constraint as in the standard dual MDP LP (2.10). Indeed, since $\widetilde{p}_{s\sigma} = \sum_a P^a_{s\sigma} \pi_{sa}$ and $x_{sa} = E_s \pi_{sa}$, the two are identical. Let us now compute the second moments in a similar fashion:

$$
\begin{aligned}
E_{s\sigma} =& \langle N_s N_\sigma \rangle = \Big\langle \Big(\sum_{t_1=0}^{\infty} n_s(t_1) \Big) \Big(\sum_{t_2=0}^{\infty} n_\sigma(t_2) \Big) \Big\rangle = \sum_{t_1=0}^{\infty} \sum_{t_2=0}^{\infty} \langle n_s(t_1) n_\sigma(t_2) \rangle \\
=& \sum_{t_1=0}^{\infty} \sum_{t_2=t_1}^{\infty} \langle n_s(t_1) n_\sigma(t_2) \rangle + \sum_{t_2=0}^{\infty} \sum_{t_1=t_2}^{\infty} \langle n_s(t_1) n_\sigma(t_2) \rangle - \sum_{t=0}^{\infty} \langle n_s(t) n_\sigma(t) \rangle
\end{aligned}
\tag{5.33}
$$

Once again, recalling that $n_s(t)$ are binary variables, and since the system can only be in one state at a particular time, the mean of their product is:

$$
\langle n_s(t_1) n_\sigma(t_2) \rangle = \begin{cases} \mathbb{P}[n_s(t_1), n_\sigma(t_2)], & \text{if } t_1 \neq t_2 \\ \delta_{s\sigma} \mathbb{P}[n_s(t_1)], & \text{if } t_1 = t_2, \end{cases} \tag{5.34}
$$

[8]Hereafter we use the notation $\mathbb{P}[x]$ for binary variables as a shorthand for $\mathbb{P}[x = 1]$, and $\mathbb{P}[x, y]$ for $\mathbb{P}[(x = 1) \wedge (y = 1)]$

where $\mathbb{P}[n_s(t_1), n_\sigma(t_2)]$ is the probability that state i is visited at time t_1 and state σ is visited at time t_2. Also, since the system is Markovian, for $t_1 \leq t_2$, we have:

$$\begin{aligned}\mathbb{P}[n_s(t_1), n_\sigma(t_2)] =&\mathbb{P}[n_s(t_2)|n_\sigma(t_1)]\mathbb{P}[n_\sigma(t_1)]\\ =&\sum_\zeta \mathbb{P}[n_s(t_2)|n_\zeta(t_2-1)]\mathbb{P}[n_\zeta(t_2-1)|n_\sigma(t_1)]\mathbb{P}[n_\sigma(t_1)]\\ =&(\widetilde{p}^{t_2-t_1})_{s\sigma}\mathbb{P}[n_s(t_1)]\end{aligned} \tag{5.35}$$

Substituting, we obtain:

$$\begin{aligned}E_{s\sigma} =&\sum_{t_1=0}^{\infty}\sum_{t_2=t_1}^{\infty}(\widetilde{p}^{t_2-t_1})_{s\sigma}\mathbb{P}[n_s(t_1)] + \sum_{t_2=0}^{\infty}\sum_{t_1=t_2}^{\infty}(\widetilde{p}^{t_1-t_2})_{s\sigma}\mathbb{P}[n_\sigma(t_2)] - \sum_{t=0}^{\infty}\delta_{s\sigma}\mathbb{P}[n_s(t_1)]\\ =&\sum_{t_1=0}^{\infty}\mathbb{P}[n_s(t_1)]\sum_{\Delta t=0}^{\infty}(\widetilde{p}^{\Delta t})_{s\sigma} + \sum_{t_2=0}^{\infty}\mathbb{P}[n_\sigma(t_2)]\sum_{\Delta t=0}^{\infty}(\widetilde{p}^{\Delta t})_{\sigma s} - \delta_{s\sigma}\sum_{t=0}^{\infty}\mathbb{P}[n_s(t_1)]\\ =&(\mathbf{I}-\widetilde{\mathbf{P}})^{-1}_{s\sigma}E_s + (\mathbf{I}-\widetilde{\mathbf{P}})^{-1}_{\sigma s}E_\sigma - \delta_{s\sigma}E_s\end{aligned} \tag{5.36}$$

Unfortunately, as can be seen from the above, the second-order moments cannot be expressed in terms of the first-order moments via a linear function. Therefore, we cannot use the moments as the optimization variables directly. Instead, we are going to work with the following asymmetric terms, where the order of indexes of M corresponds to a temporal ordering of the terms in the sums:

$$\begin{aligned}M_i =&\sum_{t=0}^{\infty}\langle n_s(t)\rangle = \sum_\sigma \alpha_\sigma(\mathbf{I}-\widetilde{\mathbf{P}})^{-1}_{s\sigma}\\ M_{s\sigma} =&\sum_{t_1=0}^{\infty}\sum_{t_2=t_1}^{\infty}\langle n_s(t_1)n_\sigma(t_2)\rangle = \sum_{t_1=0}^{\infty}\sum_{t_2=t_1}^{\infty}\mathbb{P}[n_\sigma(t_2)|n_s(t_1)]\mathbb{P}[n_s(t_1)]\\ =&M_s(\mathbf{I}-\widetilde{\mathbf{P}})^{-1}_{s\sigma}\\ M_{s\sigma\zeta} =&\sum_{t_1=0}^{\infty}\sum_{t_2=t_1}^{\infty}\sum_{t_3=t_2}^{\infty}\langle n_s(t_1)n_\sigma(t_2)n_\zeta(t_3)\rangle\\ =&\sum_{t_1=0}^{\infty}\sum_{t_2=t_1}^{\infty}\sum_{t_3=t_2}^{\infty}\mathbb{P}[n_\zeta(t_3)|n_\sigma(t_2)]\mathbb{P}[n_\sigma(t2), n_s(1)]\\ =&M_{s\sigma}(\mathbf{I}-\widetilde{\mathbf{P}})^{-1}_{\sigma\zeta}\\ &\dots\\ M_{s_1s_2\dots s_{k-1}s_k} =&M_{s_1\dots s_{k-1}}(\mathbf{I}-\widetilde{\mathbf{P}})^{-1}_{s_{k-1}s_k}\end{aligned} \tag{5.37}$$

We will refer to the above terms as the *asymmetric moments* (although they do not correspond to moments of any real variables). Clearly, all of the k^{th} order asymmetric moments can be expressed as a linear function of the asymmetric moments of order $k-1$ by moving the $(\mathbf{I}-\widetilde{\mathbf{P}})$ term to the left-hand side. For example, for the second moments, this step can be easily done by rewriting (5.37) in matrix form:

$$\mathbf{M}'' = \mathbf{D}(M_s)(\mathbf{I}-\widetilde{\mathbf{P}})^{-1}, \tag{5.38}$$

where $\mathbf{M}'' = [M_{s\sigma}]$ is the matrix of second order asymmetric moments, and $\mathbf{D}(M_s)$ is a diagonal matrix, with values of the first order moment M_s on the diagonal. Multiplying by $(\mathbf{I} - \widetilde{\mathbf{P}})$ on the right, we get $\mathbf{M}'' - \mathbf{M}''\widetilde{\mathbf{P}} = \mathbf{D}(M_s)$. Similarly, for the other moments we get:

$$
\begin{aligned}
&M_s - \sum_\sigma M_\sigma \widetilde{p}_{\sigma s} = \alpha_s \\
&M_{s\sigma} - \sum_k M_{s\zeta} \widetilde{p}_{\zeta\sigma} = \delta_{s\sigma} M_s \\
&M_{s\sigma\zeta} - \sum_\varsigma M_{s\sigma\varsigma} \widetilde{p}_{\varsigma\zeta} = \delta_{s\zeta} M_{s\sigma} \\
&\dots \\
&M_{s_1 s_2 \dots s_{k-1} s_k} - \sum_\sigma M_{s_1 s_2 \dots s_{k-1}} \widetilde{p}_{s_{k-1} j} = \delta_{s_1 s_k} M_{s_1 s_2 \dots s_{k-1}}
\end{aligned}
\tag{5.39}
$$

Furthermore, it can be seen that the true moments of order k can be expressed as linear functions of the asymmetric moments of orders 1 through k:

$$
\begin{aligned}
E_s =& M_s, \\
E_{s\sigma} =& M_{s\sigma} + M_{\sigma s} - \delta_{s\sigma} M_s \\
E_{s\sigma\zeta} =& M_{s\sigma\zeta} + M_{s\zeta\sigma} + M_{\sigma s\zeta} + M_{\sigma\zeta s} + M_{\zeta s\sigma} + M_{\zeta\sigma s} - \delta_{s\sigma} M_{s\zeta} - \delta_{s\sigma} M_{\zeta s} - \\
& \delta_{s\zeta} M_{s\sigma} - \delta_{s\zeta} M_{\sigma s} - \delta_{\sigma\zeta} M_{\sigma s} - \delta_{\sigma\zeta} M_{s\sigma} + \delta_{s\sigma\zeta} M_s
\end{aligned}
\tag{5.40}
$$

Indeed, for the first moments, there is only one state index and therefore the first asymmetric moment equals the true moment. For the second moments, both $M_{s\sigma}$ and $M_{\sigma s}$ include the term $(t_1 = t_2)$, and thus we have to subtract the term

$$\sum_{t_1} \sum_{t_2 = t_1} \langle n_s(t_1) n_\sigma(t_2) \rangle = \delta_{s\sigma} M_s.$$

The expressions for other moments are obtained in a similar manner.

The last step that remains in formulating the optimization program is to substitute the transition probabilities $\widetilde{p}_{s\sigma} = \sum_a P_{s\sigma} \pi_{sa}$ and to define the actual optimization variables. This is where we hit a problem that breaks the linearity of the program. Recall that for the standard constrained MDP, the optimization variables are defined as $x_{sa} = E_s \pi_{sa} = M_s \pi_{sa}$ and have the interpretation of the expected total number of times action a is executed in state i. As mentioned earlier, this allows one to express the first-order constraint from (5.39) as a linear function of x_{sa}. Indeed, since $\sum_a \pi_{sa} = 1$, the first moments are simply $M_s = \sum_a x_{sa}$, and recalling that $x_{sa} = E_s \pi_{sa} = M_s \pi_{sa}$, we have for the first order constraint:

$$M_s - \sum_\sigma M_\sigma \widetilde{p}_{\sigma s} = \sum_a x_{sa} - \sum_\sigma M_\sigma \sum_a P_{\sigma s} \pi_{\sigma a} = \sum_a x_{sa} - \sum_\sigma \sum_a P_{\sigma s} x_{\sigma a} \tag{5.41}$$

Unfortunately, the same trick does not work for the higher order constraints. If we were to define similar variables for the higher-order moments (e.g., $x_{s\sigma a} = M_{s\sigma} \pi_{\sigma a}$), we could, of course, rewrite (5.39) as linear functions of these variables. However, by doing this, we would introduce too many free parameters into the program. To retain the original desired interpretation of the variables, we would also have to add constraints to ensure that the policy $\boldsymbol{\pi}$ implied by the higher-order variables is the same as the one computed from the first-order x_{sa}. Clearly, these new constraints would be quadratic:

$$\frac{x_{\sigma a_1}}{x_{\sigma a_2}} = \frac{x_{s\sigma a_1}}{x_{s\sigma a_2}} = \ldots = \frac{\pi_{\sigma a_1}}{\pi_{\sigma a_2}} \tag{5.42}$$

Hence, it appears that there is no easy way to avoid the quadratic expressions $(M_{s\sigma}\pi_{\sigma a})$ in the constraints on the moments:

$$\begin{aligned}
&\sum_a x_{sa} - \sum_\sigma \sum_a x_{sa} P^a_{\sigma s} = \alpha_s \\
&M_{s\sigma} - \sum_k M_{ik} \sum_a P^a_{kj} \pi_{ka} = \delta_{s\sigma} M_s \\
&M_{s\sigma\zeta} - \sum_\varsigma M_{s\sigma\varsigma} \sum_a P^a_{\varsigma\zeta} \pi_{\varsigma a} = \delta_{s\zeta} M_{s\sigma} \\
&\ldots \\
&M_{s_1 s_2 \ldots s_{k-1} s_k} - \sum_\sigma M_{s_1 s_2 \ldots s_{k-1}} \sum_a P^a_{s_{k-1} j} \pi_{s_{k-1} a} = \delta_{s_1 s_k} M_{s_1 s_2 \ldots s_{k-1}} \\
&\pi_{sa} x_{i0} = \pi_{i0} x_{sa}
\end{aligned} \tag{5.43}$$

We are therefore left with an optimization program in x_{sa} and the asymmetric moments $(M_{s\sigma}, M_{\sigma s}, M_{s\sigma\zeta}, \ldots)$ that has: a linear objective function $\sum_s \sum_a x_{sa} r_{sa}$, a constraint on the probability of the total cost exceeding a given threshold, which is linear in the moments E (5.28), which are linear in M (5.40), and a system of quadratic constraints that synchronizes the moments (5.43).

The complexity of solving the resulting optimization problem is quite high: it is quadratic and non-convex. In Section 5.5.4, we show how the problem can be simplified if we restrict attention to stationary deterministic policies.

5.5.3 An Example

As an example of the use of the method presented in the previous sections, let us consider a simple problem, for which we can analytically compute the distribution of the total cost and formulate a constrained optimization program for it using the first three moments of the total cost. In this section we present a more careful derivation of the optimization program, paying more attention to some steps that were just briefly described in Section 5.5.1. We also present some empirical evidence that shows how closely our model approximates the true cumulative probability distribution function of the total cost.

Consider the problem depicted in Figure 5.10. There are two states, one of which (s_2) is a sink state. If the agent starts in state 1 ($\boldsymbol{\alpha} = [1, 0]$), the total received reward is the same as the total incurred cost, and both equal the total number of visits to state s_1. The obvious optimal policy for the unconstrained problem is to always execute action a_1 in state s_1 ($\boldsymbol{\pi} = [1, 0, 1, 0]$). If we set the upper bound on the cost as $C_0 = 10$, the unconstrained optimal policy has about a 30% chance of exceeding that bound. As we decrease the acceptable probability of exceeding the cost bound, the policies should become more conservative, i.e. they should prescribe higher probabilities of executing action a_2 in state s_1.

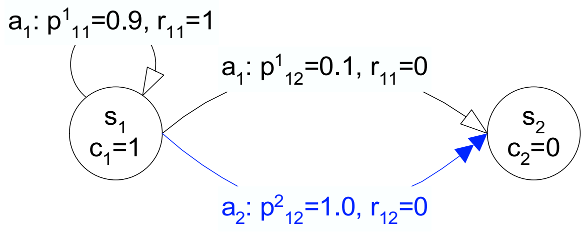


Figure 5.10: Simple problem with two states and two actions.

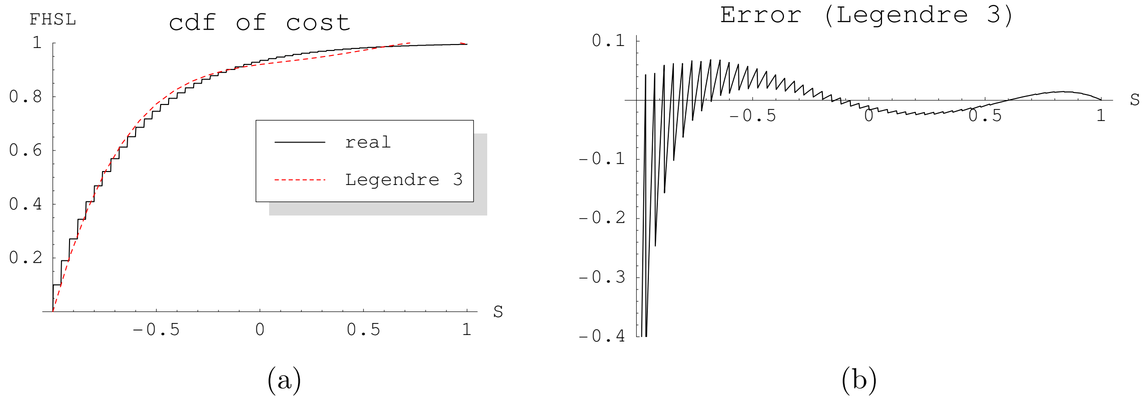


Figure 5.11: Quality of third-degree Legendre approximation of a cdf. (a) — the actual cdf of the total cost and the cdf computed from a third-degree Lagrange approximation of the pdf of the cost; (c) – relative error of the approximation in (b).

In order to apply the Legendre approximation from Section 5.5.1, we have to ensure that the total cost C is in the range $[-1, 1]$. Clearly, this is not generally the case for our problem. Therefore, to satisfy this condition, we apply a transformation $S = 2C/C_{max} - 1$ with $C_{max} = 50$ (a reasonable approximation, as $\mathbb{P}[C \geq 50] = 0.004$).[9] Figure 5.11a shows the resulting cumulative distribution function $F_S(S)$ for the unconstrained optimal policy and the cdf computed from a third-degree approximation of the pdf $f(S)$. Figure 5.11b shows the relative error of the third-degree approximation and serves to show that we can expect to get a reasonably good approximation of the cdf using just the first three moments. Note that the pdf for this problem is discrete (thus, harder to approximate with continuous Legendre polynomials), but we can still get a reasonably good approximation of the cdf, which is what we really care about, since our goal is to estimate $\mathbb{P}[S > S_0] = 1 - F_S(S_0)$. Let us now compute a third-order approximation to $f_S(S)$ for an arbitrary policy by computing the coefficients b_l (5.26):

$$b_0 = \frac{1}{2}E_S^0, \quad b_1 = \frac{3}{2}E_S^1, \quad b_2 = -\frac{5}{4}E_S^0 + \frac{15}{4}E_S^2, \quad b_3 = -\frac{21}{4}E_S^1 + \frac{35}{4}E_S^3. \tag{5.44}$$

[9]For a contracting system with bounded rewards, $\lim_{C_0 \to \infty} \mathbb{P}[C > C_0] = 0$. Thus one can always compute or estimate a reasonable C_{max} for which $\mathbb{P}[C > C_{max}]$ is arbitrary small.

Notice that here we have to use the moments of $S \in [-1, 1]$. However, the constraints in (5.43) operate on moments of C, and since our optimization variables are going to be the moments of $C \in [0, C_{max}]$, we have to be able to express the former via the latter. This can be easily done by solving the following linear system of equations for E_S:

$$\begin{aligned} E_C^0 &= \int_0^{C_{max}} f_C(C)dC = \frac{1}{2}C_{max}\int_{-1}^{1} f_S(S)dS = \frac{1}{2}C_{max}E_S^0, \\ E_C^1 &= \frac{1}{4}C_{max}^2(E_S^1 + E_S^0), \\ E_C^2 &= \frac{1}{8}C_{max}^3(E_S^2 + 2E_S^1 + E_S^0), \\ E_C^3 &= \frac{1}{16}C_{max}^4(E_S^3 + 3E_S^2 + 3E_S^1 + E_S^0) \end{aligned} \tag{5.45}$$

Now, the probability of exceeding the cost bounds is

$$\mathbb{P}[C \geq C_0] = \int_{C_0}^{C_{max}} f_C(C)dC = \frac{C_{max}}{2}\int_{S_0}^{1} f_C(S)dS,$$

where $S_0 = 2C_0/C_{max} - 1$.

In order to see how closely the approximations used in our model match the true distribution of the total cost, we have investigated a number of increasingly more constrained scenarios for our example problem, and for each one plotted the true pdf corresponding to the optimal policy, as well as the third-order approximation used in our optimization program. The results are shown in Figure 5.12. One has to note that calculating the "true" pdf for a given policy is not a trivial task, since the relationship cannot be expressed in simple terms, and the whole point of using the approximations is to be able to reason about the pdf while doing the optimization. As seen from the diagrams, our polynomial approximation of the pdf can sometimes lead to invalid negative values, which is, of course, undesirable, and an approximation that is restricted to non-negative values would be preferable. However, incorporating such constraints into our approach is a nontrivial task. Furthermore, as can be seen from the diagrams, the approximations become worse as policies become more and more conservative. This is due to the fact that Legendre polynomials have difficulty approximating horizontal lines. A possible solution to this problem might be to dynamically change the upper bound on the total cost C_{max}.

5.5.4 Restricting to Stationary Deterministic Policies

In this section, we present a simplification of the quadratic optimization problem discussed above, under the additional constraint that the optimal policy is stationary deterministic. In other words, we develop a solution to the following problem:

$$\begin{aligned} &\max U_\gamma(\pi, \alpha) \\ &\text{subject to:} \\ &\mathbb{P}\Big[C > C_0\Big|\pi, \alpha\Big] \leq P_0, \\ &\pi \in \Pi^{\text{SD}}. \end{aligned} \tag{5.46}$$

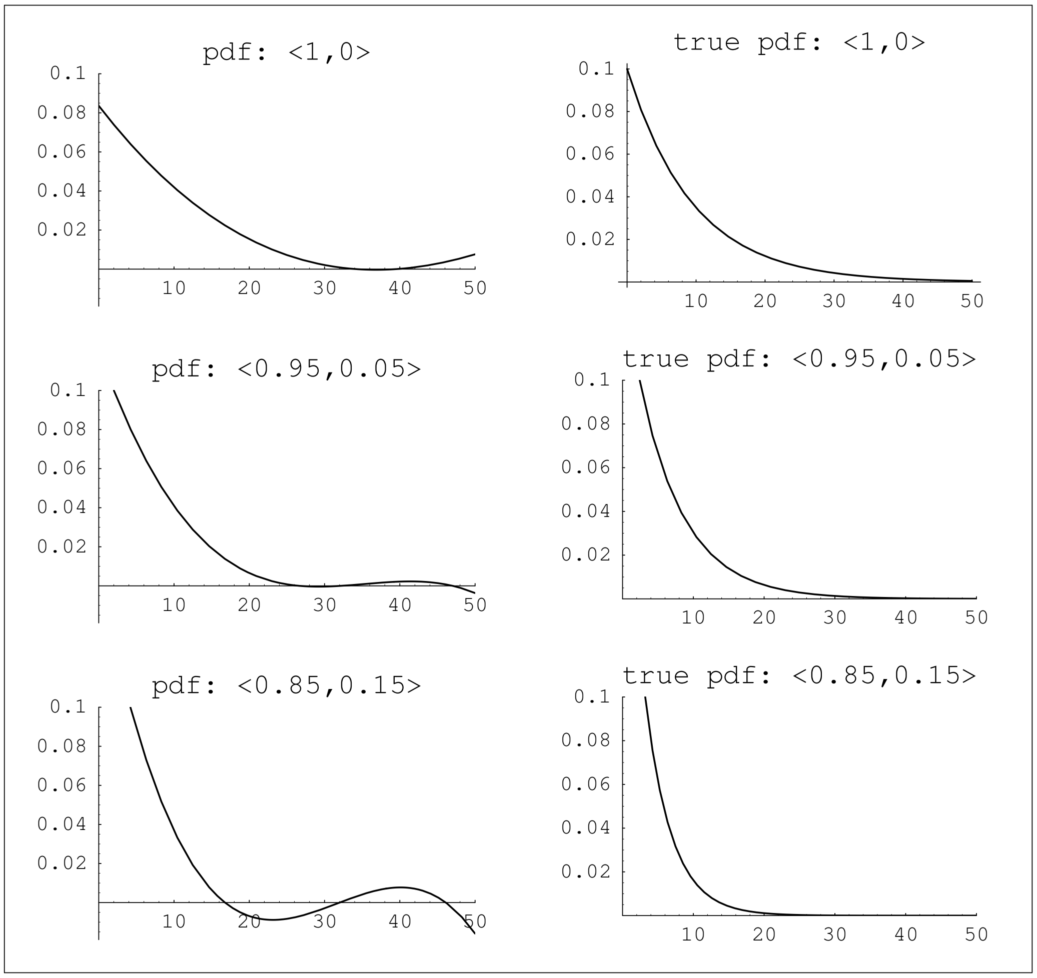


Figure 5.12: Polynomial approximation of a pdf for several policies for increasingly more conservative policies. Diagrams on the left show the third-order approximations used in the model, whereas the plots on the right show the true functions that correspond to the same policies. The numbers in angle-brackets show the policy's probability of executing in state s_1 actions a_1 and a_2, respectively. The policy for state s_2 is not shown, as it is irrelevant, since the latter is a sink state.

Recall that the only non-linear constraints in the optimization problem developed in Section 5.5.2 were the following constraints that ensured that the occupation measures of various orders:

$$\frac{x_{\sigma a_1}}{x_{\sigma a_2}} = \frac{x_{s\sigma a_1}}{x_{s\sigma a_2}} = \ldots = \frac{\pi_{\sigma a_1}}{\pi_{\sigma a_2}}$$

However, if we restrict attention to stationary deterministic policies, these constraints can be linearized by introducing a set of binary variables $\Delta_{sa} \in \{0, 1\}$, analogously to how it was done in Section 4.3 for the MDP with multiple discount factors.

Indeed, we can ensure that the occupation measure of different orders map to the same deterministic policy by replacing the above quadratic constraints with the following linear ones:

$$\begin{aligned}
&\sum_a x_{sa_1} \le \Delta_{sa}, && \forall s \in \mathcal{S}, a \in \mathcal{A};\\
&\sum_a x_{s\sigma a_1} \le \Delta_{\sigma a}, && \forall s, \sigma \in \mathcal{S}, a \in \mathcal{A};\\
&\dots && \\
&\sum_a x_{s_1 s_2 \dots s_k a_1} \le \Delta_{s_k a}, && \forall s_1, s_2, \dots s_k \in \mathcal{S}, a \in \mathcal{A};\\
&\sum_a \Delta_{sa} \le 1 && \forall s \in \mathcal{S},
\end{aligned} \tag{5.47}$$

and the moments M can be expressed via the occupation measures as follows:

$$M_{s\sigma} = \sum_a x_{s\sigma a}, \quad M_{s\sigma\zeta} = \sum_a x_{s\sigma\zeta a}, \quad \dots \quad M_{s_1 s_2 \dots s_n} = \sum_a x_{s_1 s_2 \dots s_n a}. \tag{5.48}$$

Thus, if we assume that $\pi \in \Pi^{\mathrm{SD}}$, the problem (5.46) can be approximated as an MILP, as opposed to the non-convex quadratic program developed in previous sections. This MILP has a linear objective function $\sum_s \sum_a x_{sa} r_{sa}$, and the feasible region is defined by the following linear constraints: a constraint on the probability of the total cost exceeding a given threshold, which is linear in the moments E (5.28). These moments E are themselves expressed in terms of the asymmetric moments M via the linear equalities (5.40). Finally, the asymmetric moments are expressed in terms of the higher-order occupation measures x via (5.48), and (5.47) ensures that all occupation measures correspond to the same policy.

5.6 Multiagent Resource Allocation

We now discuss the multiagent problem of distributing consumable resources among agents whose utilities for these resources are defined by the constrained MDP model discussed earlier in this chapter (Section 5.1). We focus on the risk-neutral model, where the value of a resource bundle is defined as the total expected discounted reward of the best policy whose expected cost does not exceed the available resource amounts (which corresponds to the model defined in Section 5.1). However, for cooperative agents, a resource allocation algorithm analogous to the one described below can also be implemented based on the risk-sensitive approaches discussed in Sections 5.4 and 5.5. For self-interested agents, the use of approximate techniques for solving the policy optimization problem is problematic, as it in general introduces an incentive for the agents to deviate from disclosing true information in their bids (e.g., Kfir-Dahav, Monderer, & Tennenholtz, 2000; Parkes, 2001). However, the theoretical possibility of manipulating a multiagent mechanism is often thwarted by the computational complexity of doing so (e.g., Sanghvi & Parkes, 2004; Bartholdi, Tovey, & Trick, 1989; Rothkopf et al., 1998; Procaccia & Rosenschein, 2006). Although we believe that manipulating a mechanism whose winner determination procedure requires solving a collection of NP-hard risk-sensitive MDPs is unlikely to be computationally tractable, an analysis of this issue is outside the scope of this thesis.

Analogously to the case of non-consumable resources analyzed in Chapter 3, the input to the resource-allocation problem consists of the following components:

- $\mathcal{M} = \{m\}$ is the set of agents.

- $\{\langle \mathcal{S}, \mathcal{A}, p^m, r^m, \alpha^m, \rho_q^m, \widehat{\kappa}^m \rangle\}$ is the collection of weakly coupled single-agent MDPs. As before, we assume that all agents have the same state and actions spaces $\mathcal{S}$ and $\mathcal{A}$, but each has its own transition and reward functions p^m and r^m, initial conditions α^m, as well as its own resource requirements $\rho_q^m : \mathcal{A} \times \mathcal{Q} \mapsto \mathbb{R}$ and capacity bounds $\widehat{\kappa}^m : \mathcal{C} \mapsto \mathbb{R}$.

- $\mathcal{Q} = \{q\}$ and $\mathcal{C} = \{c\}$ are the sets of resources and capacities, defined exactly as in the single-agent model in Section 5.1.

- $\kappa_q : \mathcal{Q} \times \mathcal{C} \mapsto \mathbb{R}$ specifies the capacity costs of the resources, defined exactly as in the single-agent model in Section 5.1.

- $\widehat{\rho_o} : \mathcal{O} \mapsto \mathbb{R}$ specifies the upper bound on the amounts of the shared resources, analogously to how it was defined in the case of non-consumable resource in Chapter 3.

Given the above, our goal is to construct a mechanism that allocates the consumable resources among the (self-interested) agents in a socially optimal manner. As before (in the case of non-consumable resources discussed in Chapter 3), a combinatorial auction seems to be a very good candidate for a solution approach. However, if we try to implement a straightforward combinatorial auction for this problem, a challenge immediately arises, because the number of possible resource bundles with nonzero utility for the agent is uncountably infinite. Indeed, for any (continuous) amount of a resource, an agent (in general) has a different randomized policy that maximizes its utility. Notice that this difference between consumable and non-consumable resources arises regardless of whether the resource-requirements (ρ_o and ρ_q) are integer or real-valued. With non-consumable resources (even if the resource requirements are real-valued), the number of possible bundles that have different utility for the agent is bounded by $|\mathcal{A}|^{|\mathcal{O}|}$. However, in the case of consumable resources, because the total usage is additive and depends on the system trajectory, the total number of bundles with non-zero utility is infinite.

One approach that can be used to alleviate the above difficulty is to only allocate resources to the agents in discrete amounts. This would allow us to implement a standard combinatorial auction for this problem (as described earlier in Section 3.1.1). However, such a discretization would introduce a source of approximation error and would negatively affect the complexity of the mechanism: $\sim N^{|\mathcal{Q}|}$ bundles would need to be numerated, where N is the number of discretized quantities for one resource type. Given that the WDP problem in a combinatorial auction is NP-complete, this approach is clearly undesirable.

Below we present a mechanism that does not require such a discretization or enumeration and furthermore allows the WDP to be solved in polynomial time. We note that in some situations, it is a property of the domain that the resources can be allocated to the agents only in discrete amounts, in which case the above argument about infinite support for agents' utility functions would not be valid. However it would still be desirable to avoid the exponential enumeration of resource bundles, which is accomplished by our mechanism. Furthermore, our mechanism can be easily adapted to the discrete scenario without requiring a complete enumeration of resource bundles, as described below (although, in this case the WDP becomes NP-complete).

The basic idea of the mechanism is the same as in the auction for non-consumable resources described in Chapter 3. The agents submit their MDPs, and the auctioneer solves the resource allocation and the policy-optimization problems simultaneously. The latter can be accomplished by solving the following LP (compare to the MILP (3.4)):

$$
\begin{aligned}
& \max \sum_{m,s,a} x^m(s,a) r^m(s,a) \\
& \text{subject to:} \\
& \sum_a x^m(\sigma,a) - \gamma \sum_{s,a} x^m(s,a) p^m(\sigma|s,a) = \alpha^m(\sigma), && \forall \sigma \in \mathcal{S}, m \in \mathcal{M}; \\
& \sum_q \kappa_q(q,c) \sum_a \rho_q(a,q) \sum_s x^m(s,a) \le \widehat{\kappa}^m(c), && \forall c \in \mathcal{C}, m \in \mathcal{M}; \\
& \sum_m \sum_a \rho_q(a,q) \sum_s x^m(s,a) \le \widehat{\rho_q}(q), && \forall q \in \mathcal{Q}; \\
& x^m(s,a) \ge 0, && \forall s \in \mathcal{S}, a \in \mathcal{A}, m \in \mathcal{M};
\end{aligned}
\tag{5.49}
$$

Notice that for the problem with consumable resources, where the utility of a resource bundle is defined as the value of the best policy whose expected resource usage does not exceed the available amounts (and whose capacity costs are not violated), the WDP can be solved in polynomial time. Further, notice that the LP is formulated purely on the occupation measure coordinates and does not require any special "resource-allocation" variables (compare with the case of non-consumable resources, where we had to augment the optimization with binary $\delta^m(o)$).

For the case where resources can be allocated to the agents in discrete amounts, the program (5.49) can be adapted as follows. Suppose that resource q can be given out in quantities $\{\lambda_1(q), \lambda_2(q), \dots, \lambda_n(q)\}$. Then, the WDP can be formulated as the following MILP:

$$
\begin{aligned}
& \max \sum_{m,s,a} x^m(s,a) r^m(s,a) \\
& \text{subject to:} \\
& \sum_a x^m(\sigma,a) - \gamma \sum_{s,a} x^m(s,a) p^m(\sigma|s,a) = \alpha^m(\sigma), && \forall \sigma \in \mathcal{S}, m \in \mathcal{M}; \\
& \sum_a \rho_q(a,q) \sum_s x^m(s,a) \le \sum_i \theta_i^m(q) \lambda_i(q) && \forall q \in \mathcal{Q}, m \in \mathcal{M} \\
& \sum_i \theta_i^m(q) = 1 && \forall q \in \mathcal{Q}, m \in \mathcal{M} \\
& \sum_q \kappa_q(q,c) \sum_a \rho_q(a,q) \sum_s x^m(s,a) \le \widehat{\kappa}^m(c), && \forall c \in \mathcal{C}, m \in \mathcal{M}; \\
& \sum_m \sum_i \theta_u^m(q) \lambda_i(q) \le \widehat{\rho_q}(q), && \forall q \in \mathcal{Q}; \\
& x^m(s,a) \ge 0, && \forall s \in \mathcal{S}, a \in \mathcal{A}, m \in \mathcal{M}; \\
& \theta_i^m(q) \in \{0,1\} && \forall i \in [1,n], m \in \mathcal{M};
\end{aligned}
\tag{5.50}
$$

Besides the standard occupation measure coordinates, this MILP has $n|\mathcal{M}||\mathcal{Q}|$ binary variables, as opposed to the $|\mathcal{M}|n^{|\mathcal{Q}|}$ that would be required for a straightforward combinatorial auction outlined above.

As in the case of the auction for non-consumable resources, the weak point of that mechanism is that agents have to reveal their private MDP information to the auctioneer, and if a distributed solution to the MILP (5.50) is desired,[10] the information can also propagate to other agents. To alleviate this, the agents can employ exactly the same MDP-encryption technique that was described in detail in Section 3.1.4. The only difference is that, given the transformation $\mathbf{y} = D\mathbf{x}$, the resource cost function ρ_q^m will also have to be scaled by D^{-1} (since the total consumption of consumable resources is proportional to the occupation measure). This has the additional benefit of hiding the resource cost functions (unlike the case of non-consumable resources where they were revealed).

5.6.1 Experimental Evaluation

In this section we compare the performance of the mechanism for distributing consumable resources to the performance of the mechanism for non-consumable resources that was developed in Chapter 3. For our experiments, we slightly modified the delivery domain (detailed in Section 3.2.1) that we previously used for the case of non-consumable resources.

The domain that we used in this section inherited all of the properties of the domain described in Section 3.2.1: the delivery tasks, the transition function, and the rewards were all unchanged. The only new feature was that instead of non-consumable resources, a problem with N delivery tasks had N types of consumable resources ($|\mathcal{Q}| = N$). Each action used a constant number of resource types that were selected at random at problem generation, and one execution of an action consumed a single unit of each of the corresponding resources. As in the case of non-consumable resources, we varied the number of resource types that were required for any action, and the resource types were assigned to actions from a uniformly random distribution. As before, there was a single capacity cost, and the bounds on the capacity of each agent were set to half of the amount needed for the optimal unconstrained policy. The global resource constraints were also chosen to correspond to half of the utopian levels.

Figure 5.13a shows how the resource-allocation problem for consumable resources (5.49) scales with the number of agents, as compared to the resource-allocation problem for non-consumable resources (3.4) (from Section 5.6). In fact, for this domain, the complexity of (5.49) grows linearly with the number of agents, as is more clearly visible from Figure 5.13b.

Figure 5.14 shows how the mechanism for consumable resources scales with the number of resource types. Figure 5.14a compares the running time of the mechanism for consumable resources to that of the mechanism for non-consumable ones. Again, for this domain, the algorithm for non-consumable resources scales linearly with the number of resource types, as is evident from Figure 5.14b. Further, an interesting observation is that the running time for the algorithm for consumable resources increases monotonically and does not exhibit the easy-hard-easy profile that was characteristic for non-consumable resources. Therefore, we can conclude that for consumable resources, unlike non-consumable ones, the number of resource types used by each action does not have a critical bearing on the complexity of the resource-allocation algorithm.

To confirm the above point and get a better understanding of the effect of actions' resource requirements on problem complexity, we evaluated the performance of the algorithm on domains where the number of resource types used by each action scaled proportionally to the total number of resource types used in the problem. The results are

[10]The benefits of parallelizing the LP (5.49) are less pronounced than for the MILPs.

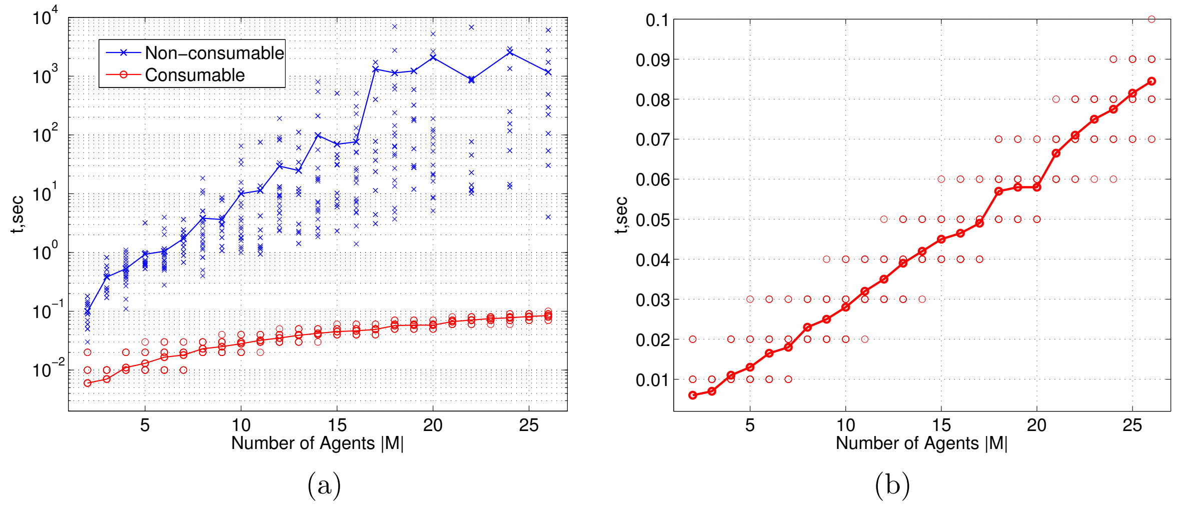


Figure 5.13: Scaling with the number of agents: consumable and non-consumable resources. (a): Consumable and non-consumable resources. (b): Consumable resources for larger problems.

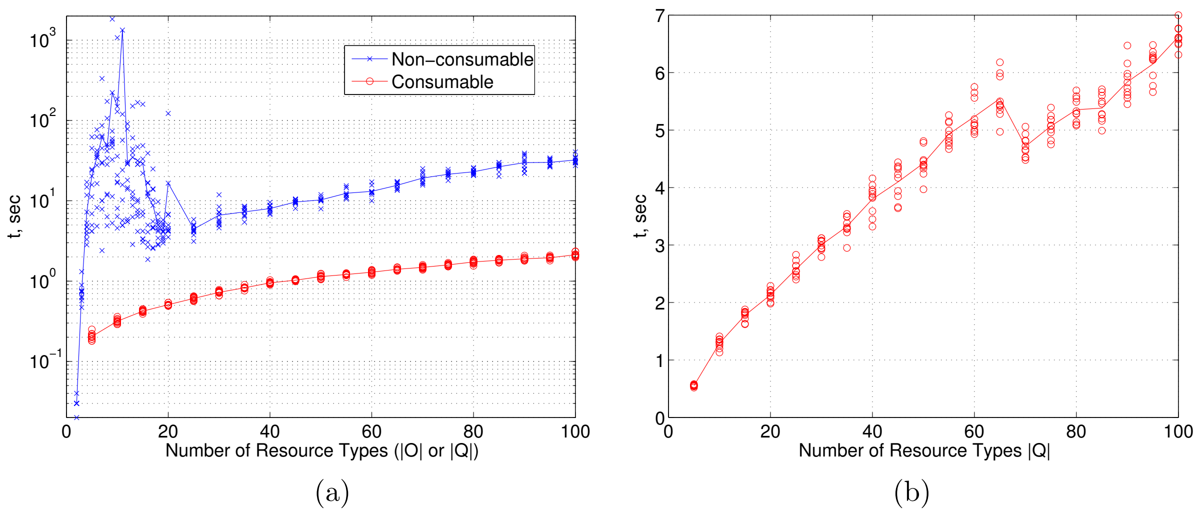
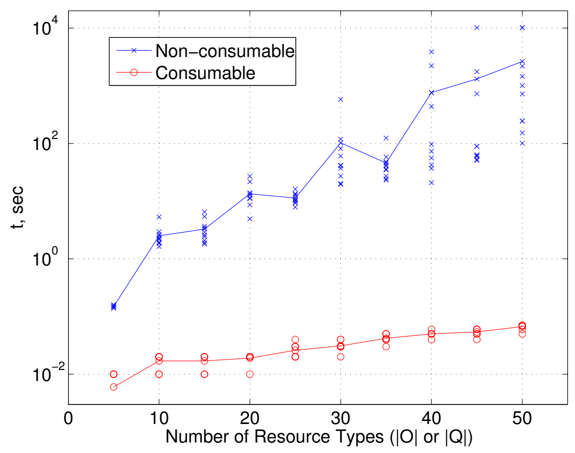


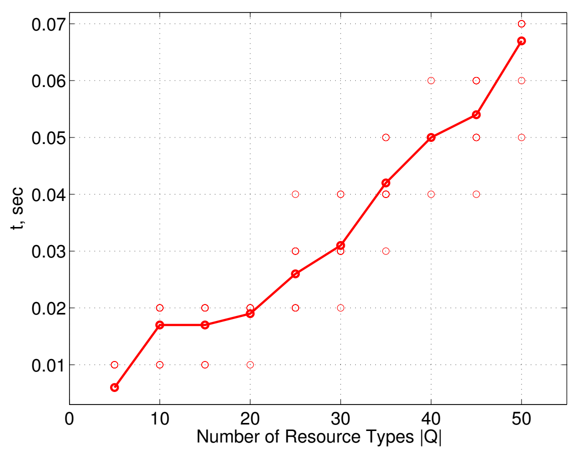
Figure 5.14: Scaling with the number of resources types: consumable and non-consumable resources. (a): Consumable and non-consumable resources ($n = 7, |\mathcal{M}| = 10$). (b): Consumable resources on larger problems ($n = 10, |\mathcal{M}| = 20$).

summarized in Figure 5.15. As can be seen from Figure 5.15a and Figure 5.15b (larger problems), even if the actions' resource requirements increase with the total number of resource types, the algorithm for consumable resources scales much more gracefully than the one for non-consumable resources.

Overall, the above experiments confirm the analytical claims we made previously that a very efficient polynomial-time resource-allocation mechanism can be implemented for consumable resources (when the utility of a resource bundle is based on policies with bounds on expected resource usage and capacity costs).



(a)



(b)

Figure 5.15: Scaling with the number of resource types and action resource requirements. Action resource requirements scale proportionally with total resource types. (a): Consumable and non-consumable resources ($n = 5, |\mathcal{M}| = 5$). (b): Consumable resources on larger problems ($n = 10, |\mathcal{M}| = 10$).

5.7 Conclusions

In this chapter we analyzed the policy-optimization and resource-allocation problems for domains involving consumable resources. For such problems, the definition of the value of a resource bundle is ambiguous, and we discussed both the risk-neutral case (based on the expected resource consumption), as well as the risk-sensitive case (with constraints on probability of exceeding resource limitations). The main contributions of the work presented in this chapter include the following:

- We presented an analysis of the properties of the MDP with probabilistic constraints. In particular, in Section 5.2 we obtained a new complexity result showing this problem to be NP-hard.
- We presented a linear approximation of the MDP with probabilistic constraints, based on the Markov inequality. This linear approximation is based on a rather strict bound, and we proposed and analyzed experimentally a procedure for iteratively adjusting the bound, which allowed us to improve the quality of the approximation, as demonstrated empirically in Section 5.4.1.
- We presented a new approximation of the probabilistic constraints via a set of Legendre polynomials. This approximation technique is very general and, as discussed in Section 5.5, can be used to model several other extensions to MDPs, such as MDPs with general utility functions. However, the computational complexity of this method is high: it needs a large number of optimization variables, and computing randomized policies maps to a quadratic non-convex program, while deterministic ones require a mixed integer program. Therefore, we believe this technique is mostly of theoretical interest.

- We analyzed the multiagent resource allocation problem for the case of risk-neutral MDPs, where the value of a resource bundle is defined as the value of the best policy that satisfies constraints on the expected resource usage. We showed that the auction-based resource-allocation mechanism developed in Chapter 3 can be adapted to work with consumable resources as well. As we discussed in Section 5.6, the complexity of the winner-determination problem used in the mechanism can be significantly lower for the case of consumable resources. Further, the mechanism is strategy-proof and maintains information privacy.

- In section 5.6.1, we experimentally tested and verified the above claims about the efficiency of the resource-allocation mechanism for risk-neutral preferences based on expected resource usage.

Finally, we note that for problems that involve both non-consumable and consumable resources, a resource-allocation mechanism can be constructed by directly (and without any modifications) combining the approaches presented in this chapter and in Chapter 3. For such domains, the winner-determination problem can be formulated as an MILP with the objective function that is common to (3.4) and (5.49), and with the feasible region that is defined simply by the union of constraints in (3.4) and (5.49) (if capacities are shared by consumable and non-consumable resources, the corresponding local constraints have to be adjusted to bound the sum of capacity costs).

CHAPTER 6

Multiagent MDPs with Local and Asymmetric Dependencies

In previous chapters, we have assumed that the agents' stochastic planning problems are only coupled via the shared resources. As discussed in Chapter 3, this model allows us to represent constraints on the space of joint actions of the agents, but it assumes that once the resources are allocated, the reward and transition functions of the agents are independent. In other words, the actions or transitions of one agent do not affect the transitions or rewards of other agents. In this chapter we study the effects of relaxing this assumption. The material of this chapter is not required for the following ones, but the results presented here do provide a larger context for the multiagent MDPs discussed in this thesis, as well as a justification for the approximate methods discussed in the following chapter.[1]

In general, in the absence of any restrictions on how agents influence each other, it becomes necessary to consider the joint state and action spaces of all agents. Consequently, as the number of agents in a multiagent system increases, the size of the MDP increases exponentially, which means that very quickly it becomes impossible to even model the problem, let alone solve it.

Fortunately, in many real-world domains, the interactions between agents are not arbitrary. In this chapter, we focus on multiagent MDPs where the interactions between agents are localized (each agent only affects a small number of neighbors). Central to the model that we use in this chapter is the concept of a *dependency graph* that describes the relationships between agents. The idea is very similar to other graphical models, e.g., *graphical games* (Kearns, Littman, & Singh, 2001), *coordination graphs* (Guestrin, Koller, Parr, & Venkataraman, 2003), and *multiagent influence diagrams* (Koller & Milch, 2003), where graphs are used to more compactly represent the interactions between agents to avoid the exponential explosion in problem size. Similarly, our representation of a multiagent MDP is exponential only in the degree of the dependency graph, and can be exponentially smaller than the size of the flat MDP defined on the joint state and action spaces of all agents. A distinguishing characteristic of the graphical representation that we use is that it makes more fine-grained distinctions about how agents affect each other: we distinguish between agents' effects on other agents' reward functions from their effects on other agents' transition functions.

We focus on asymmetric dependency graphs, where the influences that agents exert on each other do not have to be mutual. Such interactions are characteristic of large-scale

[1]The work in this chapter was originally reported in (Dolgov & Durfee, 2004b), with an extended version also appearing in (Dolgov & Durfee, 2005b).

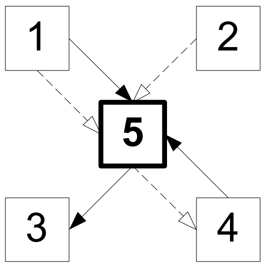


Figure 6.1: Agent dependency graph

multiagent domains with authority-based relationships between agents, i.e., low-authority agents have no control over higher-authority ones. As we discuss below, there are problem classes where this asymmetry has important positive implications on the structure of optimal multiagent policies and the problem complexity.

For any compact problem representation, an important question is whether the compactness of problem representation can be maintained in the solutions, and if so, whether it can be exploited to devise more efficient solution methods. Here we study these questions using a graphical representation of the interactions between agents. To that end, we analyze the effects of optimization criteria and topologies of dependency graphs on the structure of optimal policies, and for problems where some compactness can be maintained in the solution, we present algorithms that make use of the graphical representation. The main contribution of the work presented in this chapter is that it answers, for several classes of multiagent MDPs, the question of whether optimal policies can be represented compactly.

We note, however, that in this chapter we only study the structure and complexity of *optimal* solutions, and the claims do not apply to approximation techniques that exploit compact MDP representations (e.g., (Guestrin et al., 2003; de Farias & Van Roy, 2003, 2004; St-Aubin, Hoey, & Boutilier, 2000)). As such, this chapter provides a complexity analysis of multiagent MDPs under various conditions and can serve as a guide to where it is necessary to resort to approximate algorithms for large-scale multiagent policy optimization problems. Approximate solution techniques are discussed below in Chapter 7.

6.1 Graphical Multiagent MDPs

In many multiagent domains, the interactions between agents are only local, meaning that the rewards and transitions of an agent are not directly influenced by all other agents, but rather only by a small subset of them. To exploit the sparseness in agents' interactions, we use a compact representation that is analogous to the Bayesian network representation of joint probability distributions of several random variables. Given its similarity to other graphical models (e.g., (Jordan, 2004; Kearns et al., 2001)), we label the representation a *graphical multiagent MDP* (graphical MMDP).

Central to the definition of a graphical MMDP is the notion of a *dependency graph* (Figure 6.1), which shows how agents affect each other. The graph has a vertex for every agent in the multiagent MDP. There is a directed edge from vertex k to vertex m if agent k has an influence on agent m. The concept is very similar to *coordination graphs* (Guestrin et al., 2003), but we distinguish between two ways agents can influence each other: (1) an agent can affect another agent's transitions, in which case we use a solid arrow to depict this relationship in the dependency graph, and (2) an agent can affect another agent's rewards, in which case we use a dashed arrow in the dependency graph. For example, if an agent is trying to go through a doorway, and a second agent is controlling the state of the door, the transition probabilities of the first agent are affected by whether the door is open (state of the second agent), in which case we use a solid transition-related arrow from the second agent to the first. In a different scenario, if the door is always open, but the second agent sometimes rewards agents for going through the doorway, the transition probabilities of the first agent are not affected by the state of the second agent, but the reward obtained by the first agent depends on the state of the second agent. In this case, we use a dashed reward-related arrow from the second agent to the first.

To simplify the following discussion of graphical multiagent MDPs, we also introduce some additional concepts and notation pertaining to the structure of the dependency graph. Consider an agent $m \in \mathcal{M}$ and its neighbors. We are going to refer to the agent's upstream neighbors (parents) as $\mathcal{N}_m^-$ and downstream neighbors (children) as $\mathcal{N}_m^+$. To distinguish between reward and transition dependencies, we label all agents that directly affect m's transitions as $\mathcal{N}_m^-(P)$ (parents of m with respect to transition function P), and all agents whose transitions are directly affected by m as $\mathcal{N}_m^+(P)$ (children of m with respect to transition function P). Similarly, we use $\mathcal{N}_m^-(R)$ to refer to agents that directly affect m's rewards, and $\mathcal{N}_m^+(R)$ to refer to agents whose rewards are directly affected by m. Thus, in the graph shown in Figure 6.1, $\mathcal{N}_5^-(P) = \{1, 4\}$, $\mathcal{N}_5^-(R) = \{1, 2\}$, $\mathcal{N}_5^+(P) = \{3\}$, $\mathcal{N}_5^+(R) = \{4\}$, and $\mathcal{N}_m^- = \{1, 2, 4\}$, and $\mathcal{N}_5^+ = \{3, 4\}$. We use the terms "transition-related" and "reward-related" parents and children to distinguish between the two categories. Sometimes it will also be helpful to talk about the union of transition-related and reward-related parents or children, in which case we use $\mathcal{N}_m^- = \mathcal{N}_m^-(P) \bigcup \mathcal{N}_m^-(R)$ and $\mathcal{N}_m^+ = \mathcal{N}_m^+(P) \bigcup \mathcal{N}_m^+(R)$. Furthermore, let us label the set of all *ancestors* of m (all agents from which m is reachable) with respect to transition-related and reward-related dependencies as $\mathcal{O}_m^-(P)$ and $\mathcal{O}_m^-(R)$, respectively. Similarly, let us label the *descendants* of m (all agents reachable from m) as $\mathcal{O}_m^+(P)$ and $\mathcal{O}_m^+(R)$. Finally, $\mathcal{O}_m^- = \mathcal{O}_m^-(P) \bigcup \mathcal{O}_m^-(R)$ and $\mathcal{O}_m^+ = \mathcal{O}_m^+(P) \bigcup \mathcal{O}_m^+(R)$ refer to the unions of all ancestors and descendants, respectively.

We define a graphical MMDP with a set of agents ($\mathcal{M}$) as follows. Associated with each agent $m \in \mathcal{M}$ is a n-tuple $\langle \mathcal{S}^m, \mathcal{A}^m, p^m, r^m \rangle$, where the state space $\mathcal{S}^m$ and the action space $\mathcal{A}^m$ are defined exactly as before, but the transition and the reward functions are defined as follows:

$$\begin{aligned} p^m &: \mathcal{S}^{\mathcal{N}_m^-(P)} \times \mathcal{S}^m \times \mathcal{A}^m \times \mathcal{S}^m \mapsto [0, 1] \\ r^m &: \mathcal{S}^{\mathcal{N}_m^-(R)} \times \mathcal{S}^m \mapsto \mathbb{R}, \end{aligned} \tag{6.1}$$

where $\mathcal{S}^{\mathcal{N}_m^-(P)}$ and $\mathcal{S}^{\mathcal{N}_m^-(R)}$ are the joint state spaces of the transition-related and reward-related parents of m, respectively. In other words, the transition function of agent m specifies a probability distribution over its next states $\mathcal{S}^m$ as a function of its own current state $\mathcal{S}^m$, the current states of all of its parents $\mathcal{S}^{\mathcal{N}_m^-(P)}$, and its own action $\mathcal{A}^m$. That is $p(s^{\mathcal{N}_m^-(P)}, s^m, a^m, \sigma^m)$ is the probability that agent m goes to state σ^m if it executes action a^m when its current state is s^m and the states of its transition-related parents are $s^{\mathcal{N}_m^-(P)}$. The reward function is defined analogously on the current states of the agent itself and the reward-related parents (i.e., $r(s^{\mathcal{N}_m^-(R)}, s^m)$ is the reward that agent m gets when it is in state s^m and its parents are in states $s^{\mathcal{N}_m^-(R)}$).

Notice that, in (6.1), the transition function of m does not depend on actions of m's parents, but only on their current states. This is done for notational convenience and to simplify the discussion. It does not limit the generality of our model, as (6.1) can be used to model the more general case, by encoding the information about the last executed action in the state. Such an encoding might not be desirable for efficiency reasons, in which case the alternative is to modify our model, which should not present any fundamental difficulties.

Also notice that we allow cycles in the agent dependency graph, and moreover the same agent can both influence and be influenced by some other agent (e.g., agents 4 and 5 in Figure 6.1). We also allow for *asymmetric* influences between agents, i.e., it could be the case that one agent affects the other, but not *vice versa* (e.g., agent 5 in Figure 6.1 is influenced by agent 1, but the opposite is not true). This is often the case in domains where the relationships between agents are authority-based. It turns out that the existence of such asymmetry has important implications on the compactness of the solution and the complexity of the solution algorithms. We return to the discussion of the consequences of this asymmetry in the following sections.

It is important to note that, in this representation, each transition and reward function only specifies the rewards and transition probabilities of agent m, and contains no information about the rewards and transitions of other agents. This implies that the reward and next state of agent m are conditionally independent of the rewards and the next states of other agents, given the current action of m and the state of m and its parents $\mathcal{N}_m^-$. Therefore, this model does not allow for correlations between the rewards or the next states of different agents. For example, we cannot model the situation where two agents are trying to go through the same door and whether one agent makes it depends on whether the other one does; we can only represent, for each agent, the probability that it makes it, independently of the other. This is analogous to the commonly made simplifying assumption that variables in a dynamic Bayesian network are independent of other variables within the same time slice (e.g., Guestrin et al., 2003). This limitation of the model can be overcome by "lumping together" groups of agents that are correlated in such ways into a single agent as in the flat multiagent MDP formulation. In fact, we could have allowed for such dependencies in our model, but it would have complicated the presentation. Instead, we assume that all such correlations have already been dealt with, and the resulting problem only consists of agents (perhaps composite ones) whose states and rewards have this conditional independence property.

It is easy to see that the size of a problem represented in this fashion is exponential in the maximum number of parents of any agent, but unlike the flat model, it does not depend on the total number of agents. Therefore, for large-scale multiagent problems where each agent has a small number of parents, space savings can be significant. In particular, this can lead to exponential (in terms of the number of agents) savings for domains where the number of parents of any agent is bounded by a constant.

6.2 Properties of Graphical Multiagent MDPs

Given the compact representation of multiagent MDPs described above, two important questions arise. First, can we compactly represent the solutions to these problems? And second, if so, can we exploit the compact representations of the problems and the solutions to improve the efficiency of the solution algorithms? Positive answers to these questions would be important indications of the value of this graphical problem representation. However, before we attempt to answer these questions and get into a more detailed analysis of the related issues, let us lay down some groundwork that will simplify the following discussion.

First of all, let us note that a graphical multiagent MDP is just a compact representation, and any graphical MMDP can be easily converted to a flat multiagent MDP, analogously to how a compact Bayesian network can be converted to a joint probability distribution. For example, for a problem where all agents in a graphical MDP are maximizing the social welfare of the team (sum of rewards of all agents), this graphical MMDP is equivalent to the following flat MDP:

$$\begin{aligned}
&\mathcal{S}^{\mathcal{M}} = \mathcal{S}^1 \times \ldots \times \mathcal{S}^m, \\
&\mathcal{A}^{\mathcal{M}} = \mathcal{A}^1 \times \ldots \times \mathcal{A}^m, \\
&p^{\mathcal{M}}(s^{\mathcal{M}}) = \sum_{m \in \mathcal{M}} r^m(s^{\mathcal{N}_m^-(R)}, s^m), \\
&p^{\mathcal{M}}(s^{\mathcal{M}}, a^{\mathcal{M}}, \sigma^{\mathcal{M}}) = \prod_{m \in \mathcal{M}} p^m(s^{\mathcal{N}_m^-(P)}, s^m, a^m, \sigma^m)
\end{aligned} \tag{6.2}$$

Therefore, all properties of solutions to flat multiagent MDPs (e.g., stationarity, history-independence, etc.) also hold for equivalent problems that are formulated as graphical MMDPs. However, in general, it is not possible to convert a flat multiagent MDP to a graphical MMDP without "lumping" together all agents into one by taking cross products of their state and action spaces. This suggests that it might be possible to more compactly represent the class of policies that are optimal for problems that *are* representable as graphical MMDPs.

Let us make the following simple observation that defines the notation and sets the stage for the following discussion.

Observation 6.1. *For a graphical MMDP $\langle \mathcal{S}^m, \mathcal{A}^m, p^m, r^m \rangle$, $m \in \mathcal{M}$, with an optimization criterion for which optimal policies are Markov, stationary, and deterministic,*[2] *such policies can be represented as $\pi^m : \mathcal{S}^{\mathcal{X}^m} \mapsto \mathcal{A}^m$, where $\mathcal{S}^{\mathcal{X}^m}$ is a cross product of the state spaces of some subset of all agents ($\mathcal{X}^m \subseteq \mathcal{M}$).*

Clearly, this observation does not say much about the compactness of policies, since it allows $\mathcal{X}^m = \mathcal{M}$, which corresponds to a solution where an agent has to consider the states of all other agents when deciding on an action. If that were always the case, using this compact graphical representation for the problem would not (by itself) be beneficial, because the solution would not retain the compactness and would be exponential in the number of agents. However, for some problems, $\mathcal{X}^m$ can be significantly smaller than $\mathcal{M}$. Thus we are interested in determining, for every agent m, the *minimal* set of agents whose states m's policy has to depend on.

[2]We will implicitly assume that optimal policies are Markov, stationary, and deterministic from now on.

Definition 6.2. *In a graphical MMDP, a set of agents $\mathcal{X}^m$ is a* minimal domain *of an optimal policy $\pi^m : \mathcal{S}^{\mathcal{X}^m} \mapsto \mathcal{A}^m$ of agent m if and only if, for any set of agents $\mathcal{Y}$ and any policy $\pi^{m\prime} : \mathcal{S}^{\mathcal{Y}} \mapsto \mathcal{A}^m$, the following implications hold:*

$$\begin{aligned} \mathcal{Y} \subset \mathcal{X}^m &\implies U(\pi^{m\prime}) < U(\pi^m) \\ \mathcal{Y} \supseteq \mathcal{X}^m &\implies U(\pi^{m\prime}) \leq U(\pi^m), \end{aligned}$$

where $U(\boldsymbol{\pi})$ is the payoff that is being maximized.

In words, any policy that is defined on the states of a subset of the minimal domain $\mathcal{X}^m$ will be strictly worse than π^m, and no policy defined on a superset of $\mathcal{X}^m$ can be better than π^m.

In essence, this definition allows us to talk about the sets of agents whose joint state space is necessary and sufficient for determining optimal actions of agent m. From now on, whenever we use the notation $\pi^m : \mathcal{S}^{\mathcal{X}^m} \mapsto \mathcal{A}^m$, we implicitly assume that $\mathcal{X}^m$ is the minimal domain of π^m.

6.2.1 Assumptions

As mentioned earlier, one of the main goals of the following sections will be to characterize the minimal domains of agents' policies under various conditions. We will be interested in analyzing the worst-case complexity of policies (i.e., the structure of policies for the most difficult examples from a given class of multiagent MDPs). One way to perform such an analysis is by studying examples of such worst-case scenarios. However, we take a different route which we believe is more illustrative: we first make some assumptions about properties of minimal domains that allow us to rule out some non-interesting degenerate special cases, and then rely on these assumptions to derive our complexity results. As such, these assumptions do not limit the general complexity results that follow, as the latter only require that there exist *some* problems for which the assumptions hold. In the rest of the chapter, we implicitly assume that they hold.

Central to our future discussion will be an analysis of which random variables (rewards, states, etc.) depend on which others. It will be very useful to talk about the conditional independence of future values of some variables, given the current values of others.

Definition 6.3. *We say that a random variable X is* Markov *on the joint state space $\mathcal{S}^{\mathcal{Y}}$ of some set of agents $\mathcal{Y}$ if, given the current values of all states in $\mathcal{S}^{\mathcal{Y}}$, the future values of X are independent of any past information. If that property does not hold, we say that X is non-Markov on $\mathcal{S}^{\mathcal{Y}}$.*

We make the following assumptions:

Assumption 6.4. *For a minimal domain $\mathcal{X}^m$ of agent m's optimal policy, and a set of agents $\mathcal{Y}$, the following hold:*

1. *$\mathcal{X}^m$ is unique*
2. *$m \in \mathcal{X}^m$*
3. *$l \in \mathcal{X}^m \implies \mathcal{S}^l$ is Markov on $\mathcal{S}^{\mathcal{X}^m}$*
4. *$\mathcal{S}^m$ is Markov on $\mathcal{S}^{\mathcal{Y}} \iff \mathcal{Y} \supseteq \mathcal{X}^m$*

The first assumption allows us to avoid some special cases where there are sets of agents whose states are 100% correlated, and equivalent policies can be constructed as functions of either of the sets.

The second assumption states that the domain of an optimal policy of an agent should include its own state, which is true for all but the most trivial cases.

The third assumption says that the state space of any agent l that is in the minimal domain of m must be Markov on the state space of the minimal domain. Since the state space of agent l is in the minimal domain of m, it must influence m's rewards in a non-trivial manner. Thus, if $\mathcal{S}^l$ is non-Markov on $\mathcal{S}^{\mathcal{X}^m}$, agent m will, in general, be able to increase its payoff by expanding the domain of its policy to make $\mathcal{S}^l$ Markov (since that will allow it to better predict future rewards).

The fourth assumption says that the agent's state is Markov only on supersets of its minimal domain. Indeed, if there exists a smaller domain $\mathcal{Z} \subset \mathcal{X}^m$ such that the agent's state space $\mathcal{S}^m$ is Markov on $\mathcal{Z}$, the agent should be able to implement the same policy on $\mathcal{Z}$, contradicting the definition of the minimal domain. Conversely, $\mathcal{S}^m$ must be Markov on the minimal domain, since if the opposite were true, the agent would, in general, benefit from expanding the domain of its policy to better predict future rewards. Clearly, if $\mathcal{S}^m$ is Markov on $\mathcal{X}^m$, it must be Markov on any superset of $\mathcal{X}^m$.

These assumptions are slightly redundant (e.g., 4 could be deduced from weaker conditions), but we use this form for brevity and clarity.

We can combine Assumptions 6.4.1 and 6.4.4 into the following useful result.

Corollary 6.5. *For a minimal domain $\mathcal{X}^m$ of agent m's optimal policy and a set of agents $\mathcal{Y}$, such that $\mathcal{Y} \not\supseteq \mathcal{X}^m$, the following holds: $\mathcal{S}^m$ is non-Markov on $\mathcal{S}^{\mathcal{Y}}$.*

Indeed, if the above did not hold, meaning that $\mathcal{S}^m$ were Markov on $\mathcal{S}^{\mathcal{Y}}$, by Assumption 6.4.4, $\mathcal{Y}$ would be a superset of some minimal domain $\mathcal{X}^{m\prime} \neq \mathcal{X}^m$, which would violate the uniqueness assumption 6.4.1.

6.2.2 Transitivity

Using the assumptions of the previous sections, we can derive an important property of minimal domains that will significantly simplify the analysis that follows.

Theorem 6.6. *Consider two agents $m, l \in \mathcal{M}$, where the optimal policies of m and l have minimal domains of $\mathcal{X}^m$ and $\mathcal{X}^l$, respectively ($\pi^m : \mathcal{S}^{\mathcal{X}^m} \mapsto \mathcal{A}^m$, $\pi^l : \mathcal{S}^{\mathcal{X}^l} \mapsto \mathcal{A}^l$). Then, under Assumption 6.4, the following holds:*

$$l \in \mathcal{X}^m \implies \mathcal{X}^l \subseteq \mathcal{X}^m,$$

i.e., if the minimal domain $\mathcal{X}^m$ of agent m's policy includes agent l, then $\mathcal{X}^m$ must also include the minimal domain of l.

Proof. We will show this by contradiction. Let us consider an agent from l's minimal domain: $k \in \mathcal{X}^l$. Let us assume (contradicting the statement of the proposition) that $l \in \mathcal{X}^m$, but $k \notin \mathcal{X}^m$. Consider the set of agents that consists of the union of the two minimal domains $\mathcal{X}^m$ and $\mathcal{X}^l$, but with agent k removed:

$$\mathcal{Y}^m = \mathcal{X}^m \bigcup (\mathcal{X}^l \setminus k).$$

Then, since $\mathcal{Y}^m \not\supseteq \mathcal{X}^l$, Assumption 6.4.4 implies that $\mathcal{S}^l$ is non-Markov on $\mathcal{S}^{\mathcal{Y}^m}$. Thus, Assumption 6.4.3 implies $l \notin \mathcal{X}^m$, which contradicts our earlier assumption. ■

Essentially, this proposition says that the minimal domains have a certain "transitive" property: if agent m needs to base its action choices on the state of agent l, then, in general, m also needs to base its actions on the states of all agents in the minimal domain of l. As such, this proposition will help us to establish lower bounds on policy sizes.

Intuitively, the proposition says that since m's policy depends on l's state, and the trajectory of l's state depends on $\mathcal{X}^l$, it makes sense for agent m to base its actions on the states of all agents in $\mathcal{X}^l$. Otherwise, the evolution of m's own state might not be Markov and agent m might not be able to predict the future as well as it could, leading to suboptimal policies. To illustrate, let us once again refer to our doorway example, where one agent, m, needs to go through a doorway that is being controlled by a second agent, l. Naturally, the optimal action of the first agent, m, depends on the state of the second agent, l, implying that $l \in \mathcal{X}^m$ (second agent is in the minimal domain of the first one). Now, suppose that the door-opening policy of the second agent l depends on the state of a third agent k (perhaps the third agent controls the power to the building), which by definition means that $k \in \mathcal{X}^l$. Under these conditions the first agent m should base its action choices on the state of agent k (e.g., no sense pursuing a policy that requires going through the door if there is no power and no chance of the door opening). Thus, agent m should expand its domain to include all the external factors which affect the policy of the door-controlling agent l, which in this case, includes the state of the power-controlling agent k.

In the rest of the chapter, we analyze some classes of problems to see how large the minimal domains are under various conditions and assumptions, and for domains where minimal domains are not prohibitively large, we outline solution algorithms that exploit graphical structure. In what follows, we focus on two common scenarios: one, where the agents work as a team and aim to maximize the social welfare of the group (sum of individual payoffs), and the other, where each agent only maximizes its own individual payoff.

6.3 Maximizing Social Welfare

The following proposition characterizes the structure of the optimal solutions to graphical multiagent MDPs under the social welfare optimization criterion, and as such serves as an indication of whether the compactness of this particular representation can be exploited to devise an efficient solution algorithm for such problems. We demonstrate that, in general, when the social welfare of the group is considered, the optimal actions of each agent depend on the states of all other agents (unless the dependency graph is disconnected). Let us point out that this scenario where all agents are maximizing the same objective function is equivalent to a single-agent factored MDP, and our results for this case are analogous to the fact that the value function in a single-agent factored MDP does not, in general, retain the structure of the problem (Koller & Parr, 1999).

The implication of these results is that for large-scale cooperative MDPs where all agents are maximizing the social welfare of the group, the complexity and size of optimal solutions very quickly becomes prohibitive. Therefore, for such problems it is necessary to

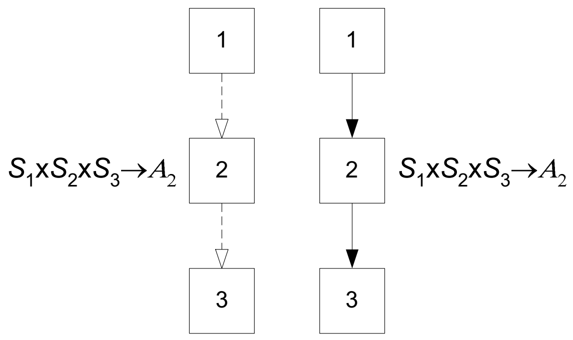

1
1
$S_1 \times S_2 \times S_3 \to A_2$
2
2
$S_1 \times S_2 \times S_3 \to A_2$
3
3

Figure 6.2: Illustration for Theorem 6.7

resort to approximate solution methods (St-Aubin et al., 2000; Guestrin et al., 2003; de Farias & Van Roy, 2003, 2004; Dolgov & Durfee, 2006).

Theorem 6.7. *For a graphical MMDP with a connected (ignoring edge directionality) dependency graph, under the optimization criterion that maximizes the social welfare of all agents, an optimal policy π^m of agent m, in general, depends on the states of all other agents, i.e.,*

$$\pi^m : \mathcal{S}^{\mathcal{M}} \mapsto \mathcal{A}^m.$$

Proof. Agent m must, at the minimum, base its action decisions on the states of its immediate (both transition- and reward-related) parents and children (as illustrated in Figure 6.2). Indeed, agent m should worry about the states of its transition-related parents, $\mathcal{N}_m^-(P)$, because their states affect the one-step transition probabilities of m, which certainly have a bearing on m's payoff. Agent m should also include in the domain of its policy the states of its reward-related parents, $\mathcal{N}_m^-(R)$, because they affect m's immediate rewards and agent m might need to adjust its behavior depending on the states of its parents. Similarly, since the agent cares about the social welfare of all agents, it will need to consider the effects of its actions on the states and rewards of its immediate children, and must thus base its policy on the states of its immediate children $\mathcal{N}_m^+(P)$ and $\mathcal{N}_m^+(R)$ to potentially "set them up" to get higher rewards.

Having established that the minimal domain of each agent must include the immediate children and parents of the agent, we can use the transitivity property from the previous section to extend this result. Although Theorem 6.6 only holds under the conditions of Assumption 6.4, for our purpose of determining the complexity of policies *in general*, it is sufficient that there exist problems for which Assumption 6.4 holds. It follows from Theorem 6.6 that the minimal domain of agent m must include all parents and children of m's parents and children, and so forth. For a connected dependency graph, this expands the minimal domain of each agent to all other agents in $\mathcal{M}$. ■

The above result should not be too surprising, as it makes clear, intuitive sense. Indeed, let us consider a simple example, shown in Figure 6.3, that has a flavor of a common production scenario (or a multi-stage delivery domain). The agents are operating an assembly line, where several tasks have to be done in sequence to build the output product. Each agent has to perform two operations in order for the whole process to succeed (e.g., in Figure 6.3, agent 2 has to perform operations 2 and 7). Furthermore, each

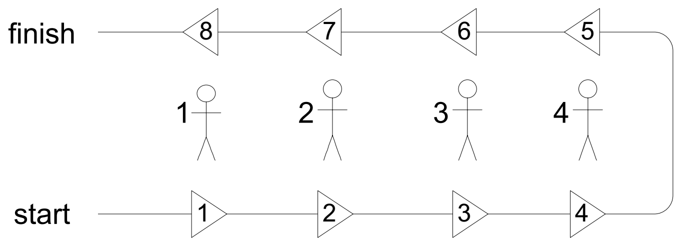


Figure 6.3: Assembly line example.

agent can choose to participate in the assembly line, yielding a very high reward if all agent cooperate, or it can concentrate on a local task that does not require the cooperation of other agents, but which has a much lower social payoff (alternatively, the agent might have a cost for participating in the assembly line, causing it to prefer doing nothing unless they all cooperate). The interactions between the agents in the assembly line are only local, i.e., each agent receives the product from a previous agent, modifies it, and passes it on to the next agent. Let us now suppose that each agent has a certain probability of breaking down, and if that happens to at least one of the agents, the assembly line fails. In such an example, the optimal policy for the agents would be to act as follows. If all agents are healthy, participate in the assembly line. If an agent that is downstream in the production line fails, immediately stop participating in the assembly and switch to the local task. If the failed agent is upstream in the production line, finish processing the items already in the pipeline and then switch to the local task. Clearly, in this example, agents' policies are functions of the states of *all* other agents.

The take-home message of this section is that, when the agents care about the social welfare of the group, even when the interactions between the agents are only local, the agents' policies depend on the joint state space of all agents. The reason for this is that a state change of one agent might lead all other agents to want to *immediately* modify their behavior. Therefore, the compact problem representation (by itself and without additional restrictions) does not lead to compact solutions.

6.4 Maximizing Own Welfare

In this section, we analyze problems where each of the agents maximizes its own payoff. Under this assumption, unlike the discouraging scenario of the previous section, the complexity of agents' policies is slightly less frightening. The following result characterizes the sizes of the minimal domains of optimal policies for problems where each agent maximizes its own utility. It states that the policy of every agent depends on the states of all of its ancestors.

Theorem 6.8. *For a graphical MMDP with an optimization criterion where each agent maximizes its own reward, the minimal domain of m's policy consists of m itself and all of its transition- and reward-related ancestors: $\mathcal{X}^m = \mathcal{E}_m^-$, where we use $\mathcal{E}_m^- = m \bigcup \mathcal{O}_m^- = m \bigcup \mathcal{O}_m^-(P) \bigcup \mathcal{O}_m^-(R)$ to refer to m and all of its ancestors.*

Proof. To show the correctness of the proposition, we need to prove that, (1) the minimal domain must include at least m itself and its ancestors ($\mathcal{X}^m \supseteq \mathcal{E}_m^-$), and (2) that $\mathcal{X}^m$ does not include any other agents ($\mathcal{X}^m \subseteq \mathcal{E}_m^-$).

We can show (1) by once again applying the transitivity property. Clearly, an agent's policy should be a function of the states of the agent's reward-related and transition-related parents, because they affect the one-step transition probabilities and rewards of the agent. Then, by Theorem 6.6, the minimal domain of the agent's policy must also include all of its ancestors.

We establish (2) as follows. We assume that it holds for all ancestors of m, and show that it must then hold for m. We then expand the statement to all agents by induction.

Let us fix the policies π^k of all agents except m. Then, consider the tuple $\langle \mathcal{S}^{\mathcal{E}_m^-}, \mathcal{A}^m, \widetilde{p}_{\mathcal{E}_m^-}, \widetilde{r}_{\mathcal{E}_m^-} \rangle$, where $\widetilde{p}_{\mathcal{E}_m^-}$ and $\widetilde{r}_{\mathcal{E}_m^-}$ are functions with the following domains and ranges:

$$\begin{aligned} \widetilde{p}_{\mathcal{E}_m^-} &: \mathcal{S}^{\mathcal{E}_m^-} \times \mathcal{A}^m \times \mathcal{S}^{\mathcal{E}_m^-} \mapsto [0,1] \\ \widetilde{r}_{\mathcal{E}_m^-} &: \mathcal{S}^{\mathcal{E}_m^-} \mapsto \mathbb{R} \end{aligned} \tag{6.3}$$

and are defined as follows:

$$\begin{aligned} \widetilde{p}_{\mathcal{E}_m^-}(s^{\mathcal{E}_m^-}, a^m, \sigma^{\mathcal{E}_m^-}) =& p^m(s^{\mathcal{N}_m^-(P)}, s^m, a^m, \sigma^m) \\ & \prod_{k \in \mathcal{O}_m^-} p^k\left(s^{\mathcal{N}_k^-(P)}, s^k, \pi^k(s^{\mathcal{E}_k^-}), \sigma^k\right) \\ \widetilde{r}_{\mathcal{E}_m^-}(s^{\mathcal{E}_m^-}) =& r^m(s^{\mathcal{N}_m^-(R)}, s^m) \end{aligned} \tag{6.4}$$

The above constitutes a fully-observable MDP on $\mathcal{S}^{\mathcal{E}_m^-}$ and $\mathcal{A}^m$ with transition function $\widetilde{p}_m$ and reward function $\widetilde{r}_m$. Let us label this decision process MDP_1. By properties of fully-observable MDPs, there exists an optimal stationary deterministic solution π_1^m of the form $\pi_1^m : \mathcal{S}^{\mathcal{E}_m^-} \mapsto \mathcal{A}^m$.

Also consider the following MDP on an augmented state space that includes the joint state space of all the agents (and not just m's ancestors): $MDP_2 = \langle \mathcal{S}^{\mathcal{M}}, \mathcal{A}^m, \widehat{p}_{\mathcal{M}}, \widehat{r}_{\mathcal{M}} \rangle$, where $\widehat{p}_{\mathcal{M}}$ and $\widehat{r}_{\mathcal{M}}$ are functions with the following domains and ranges:

$$\begin{aligned} \widehat{p} &: \mathcal{S}^{\mathcal{M}} \times \mathcal{A}^m \times \mathcal{S}^{\mathcal{M}} \mapsto [0,1] \\ \widehat{r} &: \mathcal{S}^{\mathcal{M}} \mapsto \mathbb{R} \end{aligned} \tag{6.5}$$

and are defined as follows:

$$\begin{aligned} \widehat{p}_{\mathcal{M}}(s^{\mathcal{M}}, a^m, \sigma^{\mathcal{M}}) =& p^m(s^{\mathcal{N}_m^-(P)}, s^m, a^m, \sigma^m) \\ & \prod_{k \in \mathcal{O}_m^-} p^k\left(s^{\mathcal{N}_k^-(P)}, s^k, \pi^k(s^{\mathcal{E}_k^-}), \sigma^k\right) \\ & \prod_{k \in \mathcal{M} \setminus (m \cup \mathcal{O}_m^-)} p^k\left(s^{\mathcal{N}_k^-(P)}, s^k, \pi^k(s^{\mathcal{M}}), \sigma^k\right) \\ \widehat{r}_{\mathcal{M}}(s^{\mathcal{M}}) =& r^m(s^{\mathcal{N}_m^-(R)}, s^m) \end{aligned} \tag{6.6}$$

Basically, we have now constructed two fully-observable MDPs: MDP_1 that is defined on $\mathcal{S}^{\mathcal{E}_m^-}$, and MDP_2 that is defined on $\mathcal{S}^{\mathcal{M}}$, where MDP_1 is essentially a "projection" of

MDP_2 onto $\mathcal{S}^{\mathcal{E}_m^-}$. We need to show that no solution to MDP_2 can have a higher value[3] than the optimal solution to MDP_1. Let us refer to the optimal solution to MDP_1 as π_1^m. Suppose there exists a solution π_2^m to MDP_2 that has a higher value than π_1^m. The policy π_2^m defines a stochastic trajectory for the system over the state space $\mathcal{S}^{\mathcal{M}}$ (for any fixed initial distribution over the state space). Let us label the distribution over the state space at time t as $\rho(s^{\mathcal{M}}, t)$ and its projection onto $\mathcal{S}^{\mathcal{E}_m^-}$ as $\rho(s^{\mathcal{E}_m^-}, t)$. Under our assumptions we can always construct a non-stationary randomized policy $\widetilde{\boldsymbol{\pi}}_m^1(t) : \mathcal{S}^{\mathcal{E}_m^-} \times \mathcal{A}^m \mapsto [0, 1]$ for MDP_1 that yields the same distribution $\rho(s^{\mathcal{E}_m^-}, t)$ over the state space $\mathcal{S}^{\mathcal{E}_m^-}$ as the one produced by π_2^m. Thus, there exists a randomized non-stationary solution to MDP_1 that has a higher payoff than π_1^m, which is a contradiction, since we assumed that π_1^m was optimal for MDP_1.

We have therefore shown that, given that the policies of all ancestors of m depend only on their own states and the states of their ancestors, there always exists a policy that maps the state space of m and its ancestors ($\mathcal{S}^{\mathcal{E}_m^-}$) to m's actions ($\mathcal{A}^m$) that is at least as good as any policy that maps the joint space of all agents ($\mathcal{S}^{\mathcal{M}}$) to m's actions. Then, by using induction, we can expand this statement to all agents (for acyclic graphs we use the root nodes as the base case, and for cyclic graphs, we use agents that do not have any ancestors that are not simultaneously their descendants). ■

The intuition behind the above result is very simple: if an agent is maximizing its own welfare, it should not worry about the state of agents that have no bearing on its future rewards and transitions (the descendants). It does, however, need to worry about all of its reward and transition-related ancestors, because otherwise the agent's state or reward sequence might not be Markov on the state space of its minimal domain, in which case its policy will, in general, be suboptimal.

The implication of the above proposition is that, for situations where each agent maximizes its own utility, the optimal actions of each agent do not have to depend on the states of all other agents, but rather only on its own state and the states of its ancestors. In contrast to the conclusions of Section 6.3, this result is more encouraging. For example, for dependency graphs that are trees (typical of authority-driven organizational structures), the number of ancestors of any agent equals the depth of the tree, which is logarithmic in the number of agents. Therefore, if each agent maximizes its own welfare, the size of its policy will be exponential in the depth of the tree, but only linear in the number of agents.

6.4.1 Acyclic Dependency Graphs

Thus far we have shown that problems where agents optimize their own welfare can allow for more compact policy representations. We now describe an algorithm that exploits the compactness of the problem representation to more efficiently solve such policy optimization problems for domains with acyclic dependency graphs.

It is a distributed algorithm where the agents exchange information, and each one solves its own policy optimization problem. The algorithm is very straightforward and works as follows. First, the root nodes of the graph (the ones with no parents) compute their optimal policies that are simply mappings of their own states to their own actions.

[3]The proof does not rely on the actual type of optimization criterion used by each agent and holds for any criterion that is a function only of the agents' trajectories.

Algorithm 2: Solving acyclic multiagent MDPs.

```
Function SolveAcyclicMDP()
in      : $\langle \mathcal{S}^m, \mathcal{A}^m, p^m, r^m \rangle$ — MDP of agent $m$
        : $\mathcal{N}_m^-$ — parents of agent $m$
        : $\mathcal{N}_m^+$ — children of agent $m$
out     : optimal policy $\pi^m$ for agent $m$

wait for policies $\pi^k$ of all ancestors ($k \in \mathcal{O}_m^-$) from parents $\mathcal{N}_m^-$
form MDP M=$\langle \mathcal{S}^{\mathcal{E}_m^-}, \mathcal{A}^m, \widetilde{p}_{\mathcal{E}_m^-}, \widetilde{r}_{\mathcal{E}_m^-} \rangle$ per (6.4)
$\pi^m \leftarrow$ solve MDP $\langle \mathcal{S}^{\mathcal{E}_m^-}, \mathcal{A}^m, \widetilde{p}_{\mathcal{E}_m^-}, \widetilde{r}_{\mathcal{E}_m^-} \rangle$
send own policy $\pi^m$ and $\pi^k$ to children $\mathcal{N}_m^+$
```

Once a root agent computes a policy that maximizes its welfare, it sends the policy to all of its children. Each child waits to receive the policies π^k, $k \in \mathcal{N}_m^-$ from its ancestors, then forms a MDP on the state space of itself and its ancestors as in (6.4). It then solves this MDP $\langle \mathcal{S}^{\mathcal{E}_m^-}, \mathcal{A}^m, \widetilde{p}_{\mathcal{E}_m^-}, \widetilde{r}_{\mathcal{E}_m^-} \rangle$ to produce a policy $\pi^m : \mathcal{E}_m^- \mapsto \mathcal{A}^m$, at which point it sends the policy and the policies of its ancestors to its children. The process repeats until all agents compute their optimal policies. Essentially, this algorithm performs, in a distributed manner, a topological sort of the dependency graph, and computes a policy for every agent. Let us note that parents have no incentive to hide their policies from the children, since the children cannot influence the parents' utility, because of the asymmetry.

6.4.2 Cyclic Dependency Graphs

We now turn our attention to the case of dependency graphs with cycles. Note that the complexity result of Theorem 6.8 still applies, because no assumptions about the cyclic or acyclic nature of dependency graphs were made in the statement or proof of the proposition. Thus, the minimal domain of an agent's policy is still the set of its ancestors.

The problem is, however, that the solution algorithm of the previous section is inappropriate for cyclic graphs, because it will deadlock on agents that are part of a cycle, since these agents will be waiting to receive policies from each other. Indeed, when self-interested agents mutually affect each other, it is not clear how they should go about constructing their policies. Moreover, in general, for such agents there might not even exist a set of stationary deterministic policies that are in equilibrium, i.e., since the agents mutually affect each other, the best responses of agents to each others' policies might not be in equilibrium.

A careful analysis of this case falls in the realm of stochastic games (Shapley, 1953; Owen, 1982; Wal, 1981; Littman, 1994), and is beyond the scope of this thesis. However, if we assume that there exists an equilibrium in stationary deterministic policies, and that the agents in a cycle have some "black-box" way of collectively constructing their joint policies, we can formulate an algorithm for computing optimal policies, by modifying the algorithm from the previous section as follows. The agents begin by finding the largest cycle they are a part of, and then, after the agents receive policies from their parents who are not also their descendants, the agents proceed to devise an optimal joint policy for their cycle, which they then pass to their children.

Algorithm 3: Solving cyclic multiagent MDPs.

```
Function SolveCyclicMDP()
in      : $\langle \mathcal{S}^m, \mathcal{A}^m, p^m, r^m \rangle$ — MDP of agent $m$
        : $\mathcal{N}_m^-$ — parents of agent $m$
        : $\mathcal{N}_m^+$ — children of agent $m$
out     : optimal policy $\pi^m$ for agent $m$

$\mathcal{G}_m \leftarrow$ find all your peers
wait for policies $\pi^k$ of all ancestors not in $\mathcal{G}_m$
$p_0^m \leftarrow$ local transition function from $p^m$ and $\{\pi^k\}$
form a joint MDP $\langle \mathcal{S}^{\mathcal{E}_m^-}, \mathcal{A}^{\mathcal{G}_m}, p^{\mathcal{G}_m}, p^{\mathcal{G}_m} \rangle$
$\pi^m \leftarrow$ solve joint MDP $\langle \mathcal{S}^{\mathcal{G}_m}, \mathcal{A}^{\mathcal{G}_m}, p^{\mathcal{G}_m}, p^{\mathcal{G}_m} \rangle$
send own policy $\pi^m$ to children $\mathcal{N}_m^+$
```

Notice that the algorithm relies on a way for each agent m to find all other agents that are a part of a cycle that contains m. Since the set of agents that are in a cycle with m is the intersection of the ancestors and descendants of m, finding all peers of an agent can be done in polynomial time (in the number of agents) via a simple algorithm that performs a traversal of the dependency graph.

6.5 Additive Rewards

In our earlier analysis, a reward function r^m of an agent could depend in an arbitrary way on the current states of the agent and its parents (6.1). In fact, this is why agents, in general, needed to adjust their behavior depending on the states of their parents (and children in the social welfare case), which, in turn, was why the effects of reward-related dependencies propagated just as the transition-related ones did.

In this section, we consider a subclass of reward functions for which locality is better maintained. Namely, we focus on *additively separable* reward functions:

$$r^m(s^{\mathcal{N}_m^-(R)}, s^m) = r_{mm}(s^m) + \sum_{k \in \mathcal{N}_m^-(R)} r_{mk}(s^k), \tag{6.7}$$

where r_{mk} is a function ($r_{mk} : \mathcal{S}^k \mapsto \mathbb{R}$) that specifies the contribution of agent k to m's reward. In words, we assume that a reward of agent m can be expressed as a sum of several terms, each of which depends on the state of only one agent.

Furthermore, the results of this section are only valid under certain assumptions about the optimization criteria the agents use. Let us say that if an agent receives a history of rewards $\mathcal{H}(r) = \{r(t)\} = \{r(0), r(1), \ldots\}$, its payoff is $U\big(\mathcal{H}(r)\big) = U\big(r(0), r(1), \ldots\big)$. Then, in order for our results to hold, U has to be *linear additive*:

$$U\big(\mathcal{H}(r_1 + r_2)\big) = U\big(\mathcal{H}(r_1)\big) + U\big(\mathcal{H}(r_2)\big) \tag{6.8}$$

Notice that this assumption holds for the commonly used risk-neutral MDP optimization criteria, such as expected total reward, expected total discounted reward, and average per-step reward (Puterman, 1994), and is, therefore, not greatly limiting.

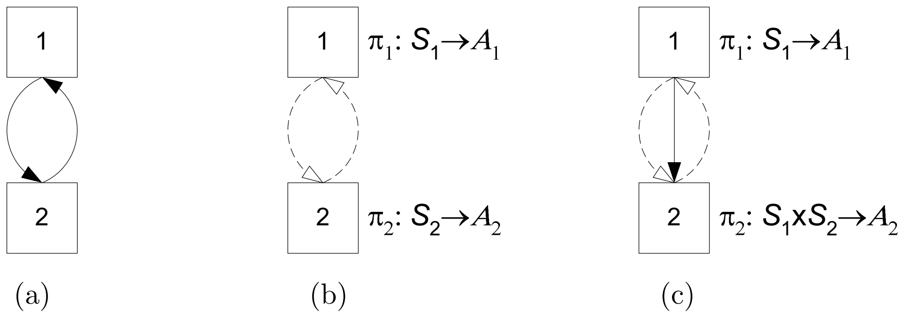


Figure 6.4: Additive rewards. Two-agent problems.

In the rest of this section we examine problems with reward functions that are subject to these conditions. We begin by analyzing some important special cases with only two agents (shown in Figure 6.4) and then discuss whether and how these results can be extended to problems with more than two agents.

First, let us observe that all problems in Figure 6.4 have cyclic dependency graphs. Therefore, if the reward functions of the agents were not additively separable, per our earlier results of Section 6.4, there would be no guarantee that there would exist an equilibrium in stationary deterministic policies. The problem in Figure 6.4a has a cycle in transition-related dependencies, and our assumptions about the reward functions will not help us with the existence of equilibria. Therefore, in this section, we will only consider problems where there are no cycles due to transition-related dependencies. Under these conditions, as we show below, our assumption about the additivity of the reward functions ensures that an equilibrium always exists for problems such as the ones in Figure 6.4b and Figure 6.4c.

Let us consider the case in Figure 6.4b. Clearly, the policy of neither agent affects the transition function of the other. Thus, given our assumptions about additivity of rewards and utility functions, it is easy to see that the problem of maximizing the payoff is separable for each agent. For example, for agent 1 we have:

$$\max_{\pi^1,\pi^2} U_1\big(\mathcal{H}(R_1(s^1,s^2)) = \max_{\pi^1} U\big(\mathcal{H}(r_{11}(s^1))\big) + \max_{\pi^2} U\big(\mathcal{H}(r_{21}(s^2))\big) \tag{6.9}$$

Thus, regardless of what policy agent 2 chooses, agent 1 should adopt a policy that maximizes the first term in (6.9). In game-theoretic terms, each of the agents has a (weakly) dominant strategy, and will adopt that strategy, regardless of what the other agent does. This is what guarantees the above-mentioned equilibrium.

Given that each agent needs to optimize a function of only that agent's own states and actions, each agent can construct its optimal policy independently. Indeed, each agent has to solve a standard MDP on its own state and action space with a slightly modified reward function:

$$\widetilde{r^m}(s^m) = r^{mm}(s^m),$$

which differs from the original reward function (6.7) in that it ignores the contribution of m's parents to its reward.

Let us now analyze the case in Figure 6.4c, where the state of agent 1 affects the transition probabilities and rewards of agent 2, and the state of agent 2 affects the rewards of agent 1. Again, without the assumption that rewards are additive, this cycle would have caused the policies of both agents to depend on the cross product of their state spaces $\mathcal{S}^1 \times \mathcal{S}^2$, and furthermore the existence of equilibria in stationary deterministic policies between self-interested agents would not be guaranteed. However, when rewards are additive, the problem is simpler. Indeed, due to our assumptions, we can write the optimization problems of the two agents as:

$$\begin{aligned}
\max_{\pi^1,\pi^2} U_1(\ldots) &= \max_{\pi^1} U_1(\mathcal{H}(r_{11})) + \max_{\pi^1,\pi^2} U_1(\mathcal{H}(r_{12})) \\
\max_{\pi^1,\pi^2} U_2(\ldots) &= \max_{\pi^1} U_2(\mathcal{H}(r_{21})) + \max_{\pi^1,\pi^2} U_2(\mathcal{H}(r_{22}))
\end{aligned} \tag{6.10}$$

Notice that here the problems are no longer separable (as they were in the previous case of Figure 6.4b), so neither agent is guaranteed to have a dominant strategy.

However, we can make an additional assumption about the structure of agents' rewards that will guarantee an existence of a Nash equilibrium in stationary deterministic policies. Namely, let us assume that agents' reward functions are subject to the following condition:

$$r_{mk}(s^k) = l_{mk}(r_{kk}(s^k)), \tag{6.11}$$

where l_{mk} is a positive linear function ($l_{mk}(x) = \alpha x + \beta,\ \alpha > 0, \beta \geq 0$). This condition implies that agents' preferences over each other states are positively (and linearly) correlated, i.e., when an agent increases its local reward, its contribution to the rewards of its reward-related children also increases linearly.

Under that assumption, (6.10) will always have an equilibrium solution in stationary deterministic policies. This is due to the fact that a positive linear transformation of the reward function of a MDP does not change its optimal policy, as demonstrated below (for concreteness we show this for MDPs with the total expected discounted reward optimization criterion, but the statement is true more generally).

Lemma 6.9. *Consider two MDPs: $\Lambda = \langle \mathcal{S}, \mathcal{A}, r, p\rangle$ and $\Lambda' = \langle \mathcal{S}, \mathcal{A}, r', p\rangle$, $r'(s) = \alpha r(s) + \beta$, where $\alpha > 0$ and $\beta \geq 0$. Then, a policy π is optimal for Λ under the total expected discounted optimization criterion if and only if it is optimal for Λ'.*

Proof. Let us consider how the linear transformation of the reward function will affect the Q function of the MDP. It is easy to see that the linear transformation $r'(s) = \alpha r(s) + \beta$ of the reward function will lead to a linear transformation of the Q function, where $Q'(s,a) = \alpha Q(s,a) + \beta(1-\gamma)^{-1}$, where γ is the discount factor.

Indeed, suppose that this is true. Then, the new Bellman equations for the transformed MDP Λ' will have the form:

$$Q'(s,a) = r'(s) + \gamma \sum_{\sigma} p(\sigma|s,a) \max_a Q'(\sigma, a)$$

or, under our hypothesis about the transformation of the Q function:

$$\alpha Q(s,a) + \frac{\beta}{1-\gamma} = \alpha r(s) + \beta + \gamma \sum_{\sigma} p(\sigma|s,a) \max_a \left(\alpha Q(\sigma,a) + \frac{\beta}{1-\gamma}\right)$$

After a trivial algebraic manipulation, the above can be expressed as

$$Q(s,a) = r(s) + \gamma \sum_{\sigma} p(\sigma|s,a) \max_a Q(\sigma,a) + \frac{\beta}{\alpha} + \frac{\gamma\beta}{\alpha(1-\gamma)} - \frac{\beta}{\alpha(1-\gamma)},$$

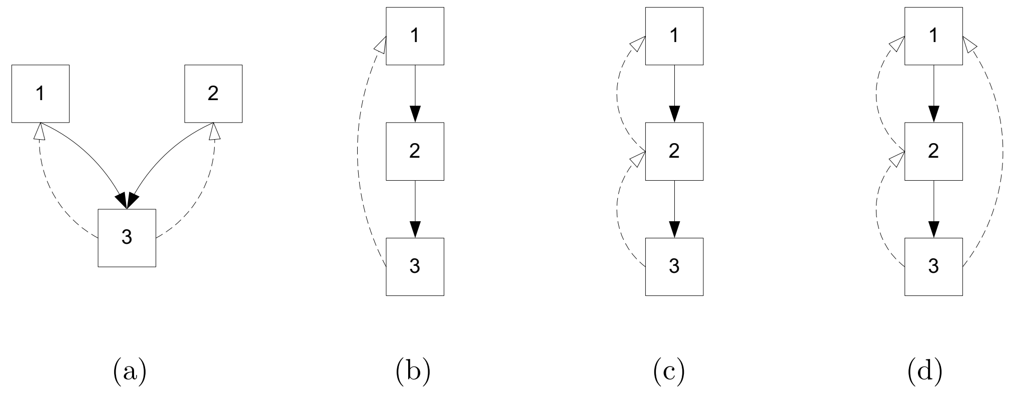


Figure 6.5: Multiagent problems, additive rewards: existence of equilibrium strategies.

where the last terms cancel out, yielding exactly the Bellman equation for the original MDP Λ:

$$Q(s,a) = r(s) + \gamma \sum_{\sigma} p(\sigma|s,a) \max_a Q(\sigma,a)$$

Therefore, since the agent computes the optimal policy as

$$\pi(s,a) = 1 \iff \\ a = \operatorname*{argmax}_a Q'(s,a) = \operatorname*{argmax}_a \alpha Q(s,a) + \beta(1-\gamma)^{-1} = \operatorname*{argmax}_a Q(s,a),$$

a policy π is optimal for Λ if and only if it is optimal for Λ'. ■

Lemma 6.9 implies that, for any policy π_1, a policy π_2 that maximizes the second term of U_1 in (6.10) will be simultaneously maximizing (given π_1) the second term of U_2 in (6.10). In other words, given any π_1, both agents will agree on the choice of π_2. Therefore, agent 1 can find the pair $\langle \pi_1, \pi_2 \rangle$ that maximizes its payoff U_1 and adopt that π_1. Then, agent 2 will adopt the corresponding π_2, since deviating from it cannot increase its utility because π_2 is simultaneously maximizing the second terms in (6.10) for both agents.

Let us now consider whether these results carry over to problems with more than two agents. Unfortunately, there is no trivial extension of the analysis to problems with arbitrary numbers of agents and general dependency graphs, because the question of the existence of equilibria in stationary deterministic strategies becomes more complicated. To illustrate the issue, let us consider a few more special cases shown in Figure 6.5.

Consider the dependency graph in Figure 6.5a. The optimization problems the agents face are as follows.

$$\begin{aligned} \max U_1(\ldots) &= \max_{\pi^1} U_1(\mathcal{H}(r_{11})) + \max_{\pi^1,\pi^3} U_1(\mathcal{H}(r_{13})) \\ \max U_2(\ldots) &= \max_{\pi^2} U_2(\mathcal{H}(r_{22})) + \max_{\pi^2,\pi^3} U_2(\mathcal{H}(r_{23})) \\ \max U_3(\ldots) &= \max_{\pi^1,\pi^1,\pi^3} U_3(\mathcal{H}(r_{33})) \end{aligned} \tag{6.12}$$

It is easy to see that the existence of a Nash equilibrium is not guaranteed in this case, because agents 1 and 2 might want agent 3 to behave in different ways and there might not exist a set of stationary, deterministic strategies $\langle \pi^1, \pi^2, \pi^3 \rangle$ that are in equilibrium (i.e., one of the agents might want to deviate). The problem with this example is due to the fact that agent 3 has multiple transition-related parents, which suggests that problems with tree-like transition dependency graphs might be better-behaved.

Let us, therefore, consider the example in Figure 6.5b, whose transition dependency graph is a tree. The optimization problems of the agents are:

$$\begin{aligned}
\max U_1(\ldots) &= \max_{\pi^1} U_1(\mathcal{H}(r_{11})) + \max_{\pi^1,\pi^2,\pi^3} U_1(\mathcal{H}(r_{13})) \\
\max U_2(\ldots) &= \max_{\pi^1,\pi^2} U_2(\mathcal{H}(r_{22})) \\
\max U_3(\ldots) &= \max_{\pi^1,\pi^2,\pi^3} U_3(\mathcal{H}(r_{33}))
\end{aligned} \tag{6.13}$$

Here, the existence of an equilibrium is also not guaranteed because, even though agents 1 and 3 will always agree on π^3 (given any π^1), agents 1 and 2 might not have an equilibrium. In other words, given a π^1 (which defines the transition function and thus the optimization problem of agent 2), agent 2 can find its best policy $\pi^{2*}(\pi^1) = \arg\max_{\pi^2} U_2(\mathcal{H}(r_{22}))$. However, given π^{2*}, agent 1 might want to change its π^1 to improve its reward via agent 3. Here, the problem is due to the fact that agent 2 has control of agent 3 who does not contribute to 2's rewards directly, but does affect the rewards of a parent of 2.

The above suggest that perhaps limiting reward loops to immediate transition-related children and parents (as in the case of two agents) might lead to equilibria. To investigate, let us consider the example from Figure 6.5c. The agents' optimization problems are:

$$\begin{aligned}
\max U_1(\ldots) &= \max_{\pi^1} U_1(\mathcal{H}(r_{11})) + \max_{\pi^1,\pi^2} U_1(\mathcal{H}(r_{12})) \\
\max U_2(\ldots) &= \max_{\pi^1,\pi^2} U_2(\mathcal{H}(r_{22})) + \max_{\pi^1,\pi^2,\pi^3} U_2(\mathcal{H}(r_{23})) \\
\max U_3(\ldots) &= \max_{\pi^1,\pi^2,\pi^3} U_3(\mathcal{H}(r_{33}))
\end{aligned} \tag{6.14}$$

Alas, here a Nash is also not guaranteed, because once again the interests of agents 1 and 2 might conflict. For example, the term $U_1(\mathcal{H}(r_{12}))$ might be the most important for agent 1, whereas agent 2 might want to choose π^2 to increase $U_2(\mathcal{H}(r_{23}))$ above all else.

A condition that does ensure the existence of an equilibrium is illustrated by the example in Figure 6.5d, where the optimization problems of the agents are:

$$\begin{aligned}
\max U_1(\ldots) &= \max_{\pi^1} U_1(\mathcal{H}(r_{11})) + \max_{\pi^1,\pi^2} U_1(\mathcal{H}(r_{12})) + \max_{\pi^1,\pi^2,\pi^3} U_1(\mathcal{H}(r_{13})) \\
\max U_2(\ldots) &= \max_{\pi^1,\pi^2} U_2(\mathcal{H}(r_{22})) + \max_{\pi^1,\pi^2,\pi^3} U_2(\mathcal{H}(r_{23})) \\
\max U_3(\ldots) &= \max_{\pi^1,\pi^2,\pi^3} U_3(\mathcal{H}(r_{33}))
\end{aligned} \tag{6.15}$$

Here, an equilibrium exists, because each agent maximizes a subset of reward terms that its parent is maximizing, i.e., given any π^1 and π^2, all three agents will agree on the choice of π^3; similarly, given a π^1 the agents will agree on the choice of π^2 and π^3.

Thus, just like in the two-agent problems discussed earlier, if the contributions of agents to each other's rewards are aligned (as in (6.11)), and the maximization problem of each agent includes reward terms that are a subset of the terms of each of its parents, an equilibrium strategy profile exists. In this case, the agents can formulate their optimal policies via algorithms similar to the ones described in Section 6.4.1.

An interesting question is whether this is a necessary condition for the existence of equilibria in stationary deterministic strategies for problems with arbitrary dependency graphs and numbers of agents, or whether weaker assumptions would be sufficient. An analysis of this issue is one of the directions of possible future work.

6.6 Conclusions

We analyzed the use of a particular compact, graphical representation for a class of multiagent MDPs with local, asymmetric influences between agents. As is the case with other graphical models, the representation studied in this chapter can lead to exponential savings in problem representation. However, in general, because the effects of agents' influences on each other propagate with time, the compactness of the problem representation is not fully preserved in the solution. The main contributions of this chapter include the following:

- We showed that for multiagent problems with the social welfare optimization criterion, problem structure is not preserved in the solution. These problems are equivalent to single-agent problems, for which similar results are known (Koller & Parr, 1999). Because optimal policies for such problems do not retain any of the structure of the original problem (agents' policies depend on the states of all other agents), exact solution methods are infeasible, and approximate solution techniques appear well-justified (as discussed in more detail in Chapter 7).

- We also analyzed multiagent problems with self-interested agents, and showed the complexity of solutions to be less prohibitive in some cases (acyclic dependency graphs). We demonstrated that under further restrictions on agents' effects on each others' rewards (additively separable, positive linear functions), locality is preserved to a greater extent. Under these conditions, equilibria in stationary deterministic strategies can exist even for graphs with reward-related cycles.

The take-home message of this chapter is that very strong conditions are required in order for an optimal solution to a well-structured multiagent MDP to retain the structure. Thus, in the next chapter, we discuss approximate solution techniques that forcefully impose structure onto the space of solutions they consider to ensure computational tractability.

CHAPTER 7

Approximate Planning and Resource Allocation with Factored MDP

In the earlier chapters of this dissertation, we have focused on exploiting the structure in agents' preferences that stems from the regularities of the underlying stochastic planning problems (modeled as MDPs). The main goal of doing so was to combat the complexity of combinatorial resource-allocation mechanisms, and we have shown that exploiting MDP-induced structure can lead to drastic gains in computational efficiency for the resource-allocation problem. In that part of the work, we have assumed that agents are weakly coupled via the resources. Then, in Chapter 6, we have analyzed multiagent MDP models with stronger coupling and discussed methods of exploiting the structure in the inter-agent dependencies for computational efficiency. However, as we described in Chapter 6, very strong assumptions are needed in order for such structure to lead to tractable solutions. In this chapter we focus on yet another source of problem structure: structure within the single-agent MDPs. Here, we show how the resource-allocation methods developed in Chapters 3 and 5, which exploit the structure due to agents' MDP-induced preferences, can be adapted to also exploit the structure that is present within the agents' MDPs.[1]

The traditional *flat* representation of an MDP requires an explicit enumeration of all possible states of the system, which subjects them to the curse of dimensionality, a term introduced by Bellman (1961) to refer to an exponential blow-up of the state space in terms of the number of *state features* that define the problem. To address this issue, *factored* MDP models have been proposed (Boutilier et al., 1995) that model the state space of an MDP as a cross product of state features, represent the transition function as a dynamic Bayesian network, and assume the reward function can be additively decomposed into functions with compact domains.

Unfortunately, as discussed in the previous chapter as well as elsewhere in the literature (e.g., Koller & Parr, 1999), a solution to a well-structured MDP does not, in general, maintain the structure and compactness of the problem, which forces the need for approximate solution algorithms. Approximate linear programming (ALP) (Schweitzer & Seidmann, 1985; de Farias & Van Roy, 2003) is a promising approach, with principled foundations and efficient solution techniques (de Farias & Van Roy, 2003, 2004; Guestrin, 2003; Guestrin et al., 2003; Patrascu, Poupart, Schuurmans, Boutilier, & Guestrin, 2002). ALP work has mostly focused on the *primal* LP (defined in equation (2.9) in Section 2.1), which operates on the space of value functions. Significantly less effort has been invested in approximating the *dual* LP (2.10), which operates on occupation measure coordinates and

[1]This chapter is based on the material presented in (Dolgov & Durfee, 2006) and (Dolgov & Durfee, 2005a).

forms the basis of our integrated resource-allocation and planning approach (Chapters 2, 3, and 5).

Developing efficient solutions to factored MDPs with constraints requires an approximate version of the dual LP, and there are two obvious ways of potentially achieving this: take the Dual of the Approximated version of the primal LP (*DALP*), or Approximate the Dual LP directly (*ADLP*). The former formulation (the DALP), proposed and analyzed by Guestrin (2003), is well suited to our needs and, in Section 7.2, we use it as the basis for constructing a resource-allocation mechanism for factored MDPs. However, this approach does have a weakness: as detailed in Section 7.1.3, it scales exponentially with the induced width of the associated cluster graph, which in some cases can grow very large (especially for constrained MDPs, where the cost functions increase the interactions between state features).

The second, ADLP approach instead approximates the dual LP directly. Unfortunately, as we demonstrate in Section 7.1.2, linear approximations of the optimization variables do not interact with the dual LP as well as they do with the primal, because the constraint coefficients cannot be computed efficiently. To address this, in Section 7.3, we develop a *composite ALP* that symmetrically approximates both the primal and the dual optimization coordinates (the value function and the occupation measure), which is equivalent to approximating both the objective functions and the feasible regions of the LPs. This method provides an efficient approximation to constrained MDPs, and also performs well on unconstrained problems, as we empirically show in Section 7.3.1.

Overall, the main contribution of this chapter is that it extends the resource-allocation algorithms presented earlier in Chapters 3 and 5 to factored MDP models. The mechanisms presented in this chapter effectively exploit both the structure in agents' preferences that is *due to the underlying MDPs*, as well as the structure *within the MDPs* themselves, leading to a very computationally efficient approximate resource-allocation mechanism. The mechanism scales well both in terms of the size of the resource-allocation problem (as the number of resources grows) and the size of the policy-optimization problem (as the number of state features in an MDP increases), as a consequence also allowing scaling to larger numbers of agents.

All methods developed in this chapter are approximate. Therefore, applying them to auction-based mechanisms for resource-allocation among self-interested agents might introduce complications, because the resulting auction might no longer be strategy-proof. As such, these methods are more appropriate for a cooperative multiagent setting.

7.1 Factored MDPs and Approximate Linear Programming

The classical MDP model requires an enumeration of all possible system states and thus scales very poorly. To combat this problem, a compact MDP representation has been proposed (Boutilier et al., 1995; Boutilier, Dearden, & Goldszmidt, 2000) that defines the state space as the cross-product of the state features: $\mathcal{S} = z_1 \times z_2 \dots z_N$, and uses a factored transition function and an additively separable reward function.

The transition function is specified as a two-layer dynamic Bayesian network (DBN) (Dean & Kanazawa, 1989), with the current state features viewed as the parents of the next time-step features:

$$p(\sigma|s,a) = p(\mathbf{z}(\sigma)|\mathbf{z}(s),a) = \prod_{j=1}^{J_p} p_j(z_n(\sigma)|a,\mathbf{z}_{p_j}(s)), \tag{7.1}$$

where $\mathbf{z}(\cdot)$ is the instantiation of all $\mathcal{Z}$ features corresponding to a state, $z_j(\cdot)$ denotes the value of the j^{th} state feature of a state, and $\mathbf{z}_{p_j}(\cdot)$ is the instantiation of the set of features $\mathcal{Z}_{p_j}$ that are the parents of z_j in the transition DBN. Likewise, in the rest of this chapter, we will use $\mathcal{Z}_\varphi$ to refer to the set of features in the domain of function φ, and $\mathbf{z}_\varphi$ to refer to an instantiation of these features.

The reward function for a factored MDP is compactly defined as

$$r(s,a) = \sum_{j=1}^{J_r} r_j(\mathbf{z}_{r_j}(s),a), \tag{7.2}$$

where $\mathbf{z}_{r_j}(\cdot)$ is an instantiation of a subset of state features $\mathcal{Z}_{r_j} \subseteq \mathcal{Z}$ that are in the domain of the j^{th} local reward function r_j.

Clearly, this factored representation is only beneficial if the local transition functions $p_j(z_j(\sigma)|\mathbf{z}_{p_j},a)$ and local reward functions $r_j(\mathbf{z}_{r_j},a)$ have small domains: $|\mathcal{Z}_{p_j}| \ll |\mathcal{Z}|$ and $|\mathcal{Z}_{r_j}| \ll |\mathcal{Z}|$, i.e., each function depends only on a small subset of all state features $\mathcal{Z}$.

7.1.1 Primal Approximation (ALP)

Approximate linear programming (Schweitzer & Seidmann, 1985; de Farias & Van Roy, 2003) lowers the dimensionality of the primal LP (2.9) by restricting the optimization to the space of value functions that are linear combinations of a predefined set of K basis functions h:

$$v(s) = v(\mathbf{z}(s)) = \sum_{k=1}^{K} h_k(\mathbf{z}_{h_k}(s))w_k, \tag{7.3}$$

where $h_k(\mathbf{z}_{h_k})$ is the k^{th} basis function defined on a small subset of the state features $\mathcal{Z}_{h_k} \subset \mathcal{Z}$, and w are the new optimization variables. This technique is similar to linear regression, where a function is approximated as a linear combination of a given (in general, non-linear) basis. The difference, however, is that here instead of minimizing a measure of error for the given data points, the goal is to minimize the measure to the unknown optimal value function. For the approximation to be computationally effective, the domain of each basis function has to be small ($|\mathcal{Z}_{h_k}| \ll |\mathcal{Z}|$).

As a notational convenience, we can rewrite the above as $v = Hw$, where H is a $|\mathcal{S}| \times |w|$ matrix composed of basis functions h_k.[2]

Thus, the LP (2.9) becomes:

$$\begin{aligned} &\min \alpha^T Hw \\ &\text{subject to:} \\ &AHw \geq r, \end{aligned} \tag{7.4}$$

[2]While using this notation, it is important to keep in mind that the exponentially sized H would never be explicitly written out, because each column is a basis function that can be represented compactly.

where we define the constraint matrix $A_{sa,\sigma} = \delta_{s\sigma} - \gamma p(\sigma|s,a)$ (where $\delta_{s\sigma}$ is the Kronecker delta, $\delta_{s\sigma} = 1 \Leftrightarrow s = \sigma$).

For this method to be effective, we need to be able to efficiently compute the objective function $\alpha^T H$ and the constraints AH, which can be done as described in (Guestrin et al., 2003). Consider a factored initial distribution:

$$\alpha(s) = \prod_{j=1}^{J_\alpha} \mu_j(\mathbf{z}_{\mu_j}(s)),$$

where, as usual, the domain of each factor μ_j is taken to be small ($|\mathcal{Z}_{\mu_j}| \ll |\mathcal{Z}|$). Then, the objective function can be computed as:[3]

$$\begin{aligned}(\alpha^T H)_k =& \sum_s \alpha(s) H_{sk} = \sum_s \prod_i \mu_j(\mathbf{z}_{\mu_j}(s)) h_k(\mathbf{z}_{h_k}(s)) \\ =& \sum_{\mathbf{z}'} \prod_{m'} \mu_{m'}(\mathbf{z}'_{\mu_{m'}}) h_k(\mathbf{z}'_{h_k}),\end{aligned}$$

where $\mathbf{z}'$ iterates over all features in the union of the domain of h_k and the domains of those $\mu_{m'}$ that have a non-zero intersection with the domain of h_k: $\mathbf{z}' = \{\mathbf{z}_{\mu_j} \cup \mathbf{z}_{h_k} : \mathcal{Z}_{\mu_j} \cap \mathcal{Z}_{h_k} \neq \varnothing\}$, because all μ_i that do not have any variables in common with h_k factor out and their sum is 1 (since it is a sum of a probability distribution over its domain). This computation is illustrated in the following example.

Example 7.1. *Consider a state space $\mathcal{S} = z_1 \times z_2 \times z_3$, a set of basis functions $H = [h_1(z_1), h_2(z_2, z_3), h_3(z_3)]$, and an initial distribution $\alpha = \mu_1(z_1)\mu_2(z_2)\mu_3(z_2, z_3)$. Then,*

$$\begin{aligned}(\alpha^T H)_1 =& \sum_{z_1, z_2, z_3} \mu_1(z_1)\mu_2(z_2)\mu_3(z_2, z_3) h_1(z_1) \\ =& \sum_{z_1} \mu_1(z_1) h_1(z_1) \sum_{z_2, z_3} \mu_2(z_2)\mu_3(z_2, z_3) \\ =& \sum_{z_1} \mu_1(z_1) h_1(z_1),\end{aligned}$$

which can be computed efficiently by summing over all values of z_1 instead of $z_1 \times z_2 \times z_3$. Similarly, both $(\alpha^T H)_2$ and $(\alpha^T H)_3$ can be computed by summing over $z_2 \times z_3$. ■

The constraint coefficients in (7.4) can also be computed efficiently:

$$\begin{aligned}(AH)_{sa,k} =& \sum_\sigma A_{sa,\sigma} h_k(\sigma) = \sum_\sigma (\delta_{s\sigma} - \gamma p(\sigma|s,a)) h_k(\sigma) \\ =& \sum_{\mathbf{z}} \delta(\mathbf{z}(s), \mathbf{z}) h_k(\mathbf{z}) - \gamma \sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{z}(s), a) h_k(\mathbf{z}).\end{aligned}$$

The first sum can be computed efficiently, because it is simply $h_k(\mathbf{z}_{h_k}(s))$, since δ is nonzero only at $\mathbf{z}(\sigma) = \mathbf{z}(s)$. The second term can also be computed efficiently, since $p(\mathbf{z}(\sigma)|\mathbf{z}(s), a)$ is a factored probability distribution (just like in the case of α above). The following example illustrates the computation.

[3] Here and below the general expression is followed by a simple example; some readers might find it beneficial to switch this order.

Example 7.2. *Consider $\mathcal{S}$ and H as in Example 7.1 and the transition model (with actions omitted):*

$$p(z_1', z_2', z_3' | z_1, z_2, z_3) = p_1(z_1'|z_1) p_2(z_2'|z_1, z_2) p_2(z_3'|z_3)$$

Then, the second term in AH, shown for $k = 3$, becomes:

$$\begin{aligned}
\sum_\sigma p(\sigma|s,a) h_3(\sigma) &= \sum_\sigma p(z_1'(\sigma), z_2'(\sigma), z_3'(\sigma)|z_1, z_2, z_3) h_3(\sigma) \\
&= \sum_{z_1', z_2', z_3'} p_1\big(z_1'|z_1(s)\big) p_2\big(z_2'|z_1(s), z_2(s)\big) p_3\big(z_3'|z_3(s)\big) h_3(z_1') \\
&= \sum_{z_3'} p_3\big(z_3'|z_3(s)\big) h_3(z_3') \sum_{z_1', z_2'} p_1\big(z_1'|z_1(s)\big) p_2\big(z_2'|z_1(s)\big) \\
&= \sum_{z_3'} p_3\big(z_3'|z_3(s)\big) h_3(z_3')
\end{aligned}$$

which can be efficiently computed by summing over z_3'. ■

The primal ALP described above reduces the number of optimization variables from $|\mathcal{S}|$ to $|w| = K$, and, as just illustrated, the coefficients of the objective function and every constraint row can be computed efficiently. However, the number of rows in the constraint matrix remains exponential at $|\mathcal{S}||\mathcal{A}|$, so the ALP has to undergo some additional transformation (or approximation) to become feasible. To address this issue, several techniques have been proposed, such as sampling (de Farias & Van Roy, 2004) and exploiting problem structure (Guestrin et al., 2003).

7.1.2 Approximation of the Dual LP

Another way we can try to construct an ALP suitable for constrained MDPs involved in resource-allocation mechanisms is to approximate the variables of the dual LP (2.10) using the primal ALP techniques. The focus of this section is on the negative result that shows that this approximation, by itself, is not computationally feasible. However, the analysis of this section also paves the way for the approximation presented in Section 7.3.

By straightforwardly applying the techniques from the primal ALP, we could restrict the optimization in (2.10) to a subset of the occupation measures that belong to a certain basis $Q = [q_l]$, $l \in [1, L]$, reducing the number of optimization variables from $|\mathcal{S}||\mathcal{A}|$ to $|y| = L$:

$$\begin{aligned}
&\max r^T Q y \\
&\text{subject to:} \\
&A^T Q y = \alpha, \\
&Q y \geq 0.
\end{aligned} \tag{7.5}$$

For this approximation to be practical, we need to efficiently compute the objective function $r^T Q$ and the constraint matrix $A^T Q$, as well as deal with the exponential number of constraints. The objective-function coefficients can be computed efficiently:

$$
\begin{aligned}
(r^T Q)_l =& \sum_{s,a} (r^T)_{sa} Q_{sa,l} = \sum_{s,a} \sum_{j=1}^{J_r} r_j\big(\mathbf{z}_{r_j}(s)\big) q_l(\mathbf{z}_{q_l}) \\
=& \beta \sum_{j=1}^{J_r} \Big[\sum_{\mathbf{z}_{r_j} \bigcup \mathbf{z}_{q_l}} r_j\big(\mathbf{z}_{r_j}(s)\big) q_l(\mathbf{z}_{q_l}) \Big],
\end{aligned}
$$

where $\beta = |\mathcal{Z} \setminus (\mathcal{Z}_{r_j} \bigcup \mathcal{Z}_{q_l})|$ is the normalization constant that is the size of the domain not included in the summation. Each of the M terms above can be efficiently computed by summing over the state variables in the union $\mathcal{Z}_{r_j} \bigcup \mathcal{Z}_{q_l}$. Unfortunately, the same is not true for the constraint coefficients, and therein lies the biggest problem of the dual ALP:

$$
(A^T Q)_{\sigma,l} = \sum_{s,a} \delta_{s\sigma} q_l(s,a) - \sum_{s,a} \gamma p(\sigma|s,a) q_l(s,a) \tag{7.6}
$$

The first term can be calculated efficiently, as in the case of the primal ALP, since it is simply $q_l(\mathbf{z}_{h_k}(j))$. However, the second term presents problems, as demonstrated below.

Example 7.3. *Consider $\mathcal{S}$ and p as in the previous examples. The problematic second term in (7.6), for $Q = [q_1(z_1, z_2, a), q_2(z_2, a), q_3(z_3, a)]$, becomes ($l = 3$, with actions a omitted for brevity):*

$$
\begin{aligned}
&\sum_{z_1,z_2,z_3} q_3(z_3) p_1(z_1'(j)|z_1) p_2(z_2'(j)|z_1, z_2) p_3(z_3'(j)|z_3) \\
&= \sum_{z_3} q_3(z_3) p_3(z_3'(j)|z_3) \sum_{z_1,z_2} p_1(z_1'(j)|z_1) p_2(z_1'(j)|z_2, z_2)
\end{aligned}
$$

and computing the last term requires summing over the whole state space $z_1 \times z_2 \times z_3$. ■

This example demonstrates the critical difference between the primal and dual ALPs, due to the difference between the left- and the right-hand-side operators $A(\cdot)$ and $(\cdot)A$, used in the primal and dual ALPs, respectively. The former can be computed efficiently, because $\sum_a P(a|b) = 1$ and their product drops out of the computation, while the latter cannot, since a product of terms of the form $\sum_b P(a|b)$ is hard to compute efficiently. Therefore, the drawback of the dual ALP (7.5) is that it has an exponential number of constraints, and computing the coefficients for *each one* of them scales exponentially.

7.1.3 Dual ALP

An alternative way of constructing a computationally tractable LP in the dual coordinates is to consider the dual of the primal ALP (7.4) (as done in (Guestrin, 2003)):

$$
\begin{aligned}
&\max r^T x \\
&\text{subject to:} \\
&H^T A^T x = H^T \alpha.
\end{aligned} \tag{7.7}
$$

This LP has $|\mathcal{S}||\mathcal{A}|$ variables (occupation-measures) and $|w| = K$ constraints (approximated flow conservation), where each of the K constraints corresponds to a linear combination of

$|\mathcal{Z}_{h_k}|$ real flow constraints, with h_k defining the weights with which they are aggregated into the new approximate constraint. (7.7) is always feasible, since the constraints in (7.7) are a relaxed version of the exact conservation-of-flow constraints in (2.10). Furthermore, as shown in (Guestrin, 2003), a solution to (7.7) constitutes a valid density function (summing to $(1-\gamma)^{-1}$), from which it follows that the LP is always bounded.

A problem with the LP (7.7) is that it has an exponential number of optimization variables. However, as proposed by Guestrin (2003), the exact occupation measure x can be represented more compactly by using *marginal occupation measures* (or marginal visitation frequencies) ξ, which define the occupation measure over subsets of the state features. Indeed, let us define a marginal occupation measure over each of the following clusters of variables: the domain of every primal basis function $\mathcal{Z}_{h_k}$, the domain of every local reward function $\mathcal{Z}_{r_j}$, and the domains of all backprojections of the basis functions g_k. The challenge then becomes in ensuring that the marginal occupation measures ξ are consistent. Fortunately, global consistency (but not necessarily correspondence to a valid flat occupation measure x) can often be achieved via local marginal consistency constraints that ensure that marginal occupation measures ξ_1 and ξ_2 agree on the values of features they share (with non-overlapping features marginalized out):

$$\sum_{a}\sum_{\widetilde{\mathbf{z}}_j,a}\xi_j(\mathbf{z}_j,a)=\sum_{a}\sum_{\widetilde{\mathbf{z}}_i,a}\xi_i(\mathbf{z}_i,a), \tag{7.8}$$

where $\widetilde{\mathcal{Z}_i}=\mathcal{Z}_j\setminus\mathcal{Z}_i$, and $\widetilde{\mathcal{Z}_j}=\mathcal{Z}_i\setminus\mathcal{Z}_j$. As is the case in graphical models (Jordan, 2004) more generally, and as observed for this particular case (Guestrin, 2003), when the cluster graph associated with the domains of the marginal occupation measures ξ forms a *junction tree*,[4] local consistency implies global consistency. If the original cluster graph does not form a junction tree, it can be made into one by applying the widely used techniques of moralization and triangulation.[5]

Using the marginal occupation measure as described above, the LP (7.7) can be represented more compactly as the following LP defined on marginal occupation measures $\xi_j(\mathbf{z}_j)$, one per each cluster in $\mathcal{Z}_{h_k}$, $\mathcal{Z}_{g_k}$, $\mathcal{Z}_{r_j}$:[6]

[4]A junction tree is an undirected tree with clusters of state features as nodes, with the property that if a feature appears in clusters A and B, it also appears in any cluster in the path from A to B.

[5]Note that while the problem of computing the junction tree with the smallest width is an NP-hard problem (Arnborg, Corneil, & Proskurowski, 1987), there are efficient approximations.

[6]Here and below, for simplicity we assume the cluster graph is a junction tree. If not, the LP will have to undergo a preprocessing step, as described above.

$$\max \sum_{j=1}^{J_r} \sum_a \sum_{\mathbf{z}_{r_j}} r_j(\mathbf{z}_{r_j}, a)\xi_{r_j}(\mathbf{z}_{r_j}, a)$$

subject to:

$$\begin{aligned}
&\sum_a \xi_{h_k}(\mathbf{z}_{h_k}, a)h_k(\mathbf{z}_{h_k}) - \gamma \sum_a g_k(a)\xi_{g_k}(\mathbf{z}_{h_k}, a) = \sum_{\mathbf{z}_{h_k}} \alpha(\mathbf{z}_{h_k})h_k(\mathbf{z}_{h_k}) && \forall k \in [1, K];\\
&\sum_{\widetilde{\mathbf{z}_j}} \xi_j(\mathbf{z}_j, a) = \sum_{\widetilde{\mathbf{z}_i}} \xi_i(\mathbf{z}_i, a) && \forall \xi_j, \xi_i, \mathbf{z}_j, \mathbf{z}_i,\ \widetilde{\mathcal{Z}_i} = \mathcal{Z}_j \setminus \mathcal{Z}_i,\ \widetilde{\mathcal{Z}_j} = \mathcal{Z}_i \setminus \mathcal{Z}_j;\\
&\sum_a \sum_{\mathbf{z}_j} \xi_j(\mathbf{z}_j, a) = (1-\gamma)^{-1} && \forall j,\ \mathbf{z}_j;\\
&\xi_j(\mathbf{z}_j, a) \geq 0, && \forall a, j,\ \mathbf{z}_j.
\end{aligned} \tag{7.9}$$

Given a solution $\xi_j(\mathbf{z}_j)$ to the above LP, one way (detailed in (Guestrin, 2003)) to compute the optimal policy for any given state is as follows. Given the dual solution $\xi_j(\mathbf{z}_j)$, compute the primal solution w that defines the value-function approximation. Given the current state, do a one-step lookahead and, for each state, compute the value-function estimate based on w, and greedily select the action that maximizes the sum of the one-step reward and the expected value of the next step (as in the standard MDP).

Another way of obtaining the policy is to calculate it directly from the marginal occupation measures ξ. Each marginal occupation measure forms a valid distribution on its domain, so we can use a variant of the junction-tree inference algorithm for calculating the true occupation measures for various actions in the current state. The procedure is described in Algorithm 4 and works by simply traversing the cluster tree from the root and factoring in the marginal occupation measures along the way, while normalizing appropriately. Note that because the marginal occupation measures (even if globally consistent) do not necessarily correspond to a valid occupation measure x, there is not a one-to-one correspondence between the marginals ξ and policy π, as in the flat case.

Example 7.4. *Consider the cluster graph in Figure 7.1. The marginal occupation measures are: $\xi(z_1, z_2, a)$, $\xi(z_1, z_3, a)$, and $\xi(z_2, z_4, a)$. This graph forms a junction tree, so global consistency of marginal occupation measures is guaranteed whenever local consistency constraints are satisfied:*

$$\sum_{z_2} \xi(z_1, z_2, a) = \sum_{z_3} \xi(z_1, z_3, a)$$
$$\sum_{z_1} \xi(z_1, z_2, a) = \sum_{z_4} \xi(z_2, z_4, a)$$

The true occupation measure for this problem is:

$$x(z_1, z_2, z_3, z_4, a) = \frac{\xi(z_1, z_2, a)\xi(z_1, z_3, a)\xi(z_2, z_4, a)}{\sum_{z_1} \xi(z_1, z_2, a) \sum_{z_2} \xi(z_1, z_2, a)},$$

which is what would be computed by Algorithm 4. ■

Algorithm 4: Obtaining policy from marginal occupation measures.

```
Function PolicyForState
in    : $\mathbf{z}$ — current state
      : $n$ — cluster id
out   : $\pi(\mathbf{z}, a)$ — policy for current state

return PolicyForStateRecursive($\mathbf{z}$, root-cluster, $\mathbf{1}$, $\varnothing$)

Function PolicyForStateRecursive
in    : $\mathbf{z}$ — current state
      : $n$ — cluster id
      : $\pi$ — marginal policy
      : $\theta$ — features already factored into policy
out   : $\pi'(\mathbf{z}, a)$ — marginal policy
      : $\theta'$ — features already factored into policy

/*update features already factored in */
$\theta' \leftarrow \theta \bigcup \mathcal{Z}_{\xi_j}$
/*factor in $n^{\text{th}}$ marginal */
$\pi'(\mathbf{z}, a) \leftarrow \pi(\mathbf{z}, a)\xi(\mathbf{z}_j, a)$
/*normalize by overlapping features */
$\mathcal{Z}' \leftarrow \theta \bigcap \mathcal{Z}_{\xi_j}$
$\pi'(\mathbf{z}, a) \leftarrow \pi'(\mathbf{z}, a)/\xi(z')$
/*factor in children */
forall $i \in Children(j)$ do
    $\pi'' \leftarrow PolicyForStateRecursive(j, \pi', \theta)$
    $\pi'(\mathbf{z}, a) \leftarrow \pi'(\mathbf{z}, a)\pi''(\mathbf{z}, a)$
end
```

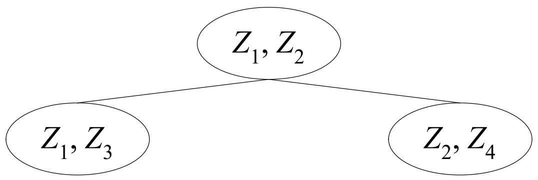


Figure 7.1: Example of a simple cluster graph that forms a junction tree.

7.2 Resource Allocation with Factored MDPs

We now show how the resource-allocation algorithms developed earlier for non-consumable resources (Chapters 2 and 3) and consumable resources (Chapter 5) can be adapted to work with factored MDP models.

We begin by considering the case of non-consumable resources ($\mathcal{O}$). For notational simplicity, we discuss the case of binary resource requirements ($\rho_o(a, o) = \{0, 1\}$), but all the results presented below generalize directly to the non-binary case. Recall from

Chapter 3 that the optimal allocation of resources and optimal policies were computed via MILP (3.4), defined on a set of occupation measures and augmented with binary indicator variables. Using the latter, we could formulate each agent's capacity constraints and global resource constraints as linear functions of $\delta^m(o)$:

$$\begin{aligned} \sum_o \kappa(o,c)\delta^m(o) &\leq \widehat{\kappa}(c), && \forall c \in \mathcal{C}, m \in \mathcal{M}; \\ \sum_m \delta^m(o) &\leq \widehat{\rho_o}(o), && \forall o \in \mathcal{O}. \end{aligned}$$

In Chapter 2, we have shown how the relationship between binary indicator variables $\delta^m(o)$ and the continuous occupation-measure variables $x^m(s,a)$ can be captured via a set of linear constraints. Below we show how the same can be accomplished for $\delta^m(o)$ and the marginal occupation measures $\xi_j(\mathbf{z}_j, a)$.

As in Chapter 2, let us first consider the slightly simpler case of synchronizing $\Delta(a)$ (binary indicators on whether action a is used in the policy) and the marginal occupation measures $\xi_j(\mathbf{z}_j, a)$. This can be accomplished via the following constraints (compare with (2.19)):

$$\xi_j(\mathbf{z}_j, a)(1-\gamma) \leq \Delta(a), \qquad \forall \xi, \mathbf{z}_j. \tag{7.10}$$

Indeed, $\xi_j(\mathbf{z}_j, a)(1-\gamma) \leq 1$, because $\sum_{\mathbf{z}_j,a} \xi_j(\mathbf{z}_j, a) = (1-\gamma)^{-1}$, as ensured by constraints on the marginal occupation measures in (7.9). Thus, constraint (7.10) ensures that when any marginal $\xi(\cdot, a)$ assigns a non-zero probability to action a, this drives the corresponding $\Delta(a)$ to be 1. On the other hand, when all $\xi(\cdot, a)$ are 0 for a given a, both $\Delta(a) = 0$ and $\Delta(a) = 1$ satisfy (7.10), in which case $\Delta(a)$ will be driven down to $\Delta(a) = 0$ by the capacity constraints (if they are binding; if they are not, the value of $\Delta(a)$ is irrelevant).

Under the assumption that the resource requirements are binary, instead of using $|\mathcal{A}|$ binary variables $\Delta(a)$ that indicate whether the agent plans to execute action a in its policy, it suffices to use $|\mathcal{O}|$ binary variables, where $\delta(o)$ tells us whether the agent's policy requires the use of resource o.

The relationship between the binary variables $\delta(o)$ and the marginals ξ can be expressed similarly to how it was done for $\Delta(a)$ above, leading to the following linear constraint:

$$(1-\gamma)\sum_a \rho_o(a,o) \sum_j \sum_{\mathbf{z}_j} \xi_j(\mathbf{z}_j, a) \leq \delta(o), \qquad \forall o. \tag{7.11}$$

The above allows us to construct an equivalent of the multiagent resource-allocation and policy-optimization MILP (3.4) for factored MDPs (the full MILP is presented below).

We now discuss the case of consumable resources ($\mathcal{Q}$), which is even simpler than the case of non-consumable resources discussed above. Consider a set of linearly decomposable resource-cost functions ρ_q (defined analogously to the reward functions in (7.2)):

$$\rho_q(a,q) = \sum_{j=1}^{J_q} \rho_j(\mathbf{z}_{\rho_j}(s), a). \tag{7.12}$$

Then the total resource requirements of a policy of agent m, as defined by a set of marginals ξ^m is simply:

$$\sum_{j=1}^{J_q} \sum_a \sum_{\mathbf{z}^m_{\rho_j}} \rho^m_j(\mathbf{z}^m_{\rho_j}, a)\xi^m_{\rho_j}(\mathbf{z}^m_{\rho_j}, a), \tag{7.13}$$

which allows us to bound the local capacity costs as well as the global resource costs via linear functions of the marginal occupation measures.

Putting it all together, we arrive at the following MILP that has, for every agent m, a set of marginal occupation measures ξ^m and a set of binary variables $\delta^m(o) = \{0,1\}$. Each agent has its own basis functions h_k^m, and therefore its own backprojections g_k^m.

$$\max \sum_m \sum_{j=1}^{J_r} \sum_a \sum_{\mathbf{z}_{r_j}^m} r_j^m(\mathbf{z}_{r_j}^m, a) \xi_{r_j}^m(\mathbf{z}_{r_j}^m, a)$$

subject to:

$$\begin{aligned}
&\sum_a \xi_{h_k}^m(\mathbf{z}_{h_k}^m, a) h_k^m(\mathbf{z}_{h_k}^m) - \gamma \sum_a g_k^m(a) \xi_{g_k}^m(\mathbf{z}_{h_k}^m, a) = \sum_{\mathbf{z}_{h_k}} \alpha^m(\mathbf{z}_{h_k}^m) h_k^m(\mathbf{z}_{h_k}^m) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \forall k \in [1, K],\ m \in \mathcal{M}; \\
&\sum_{\widetilde{\mathbf{z}_j^m}} \xi_j^m(\mathbf{z}_j^m, a) = \sum_{\widetilde{\mathbf{z}_i^m}} \xi_i^m(\mathbf{z}_i^m, a) \\
&\qquad\qquad \forall m \in \mathcal{M},\ \xi_j^m, \xi_i^m, \mathbf{z}_j^m, \mathbf{z}_i^m,\ \widetilde{\mathcal{Z}_i^m} = \mathcal{Z}_j^m \setminus \mathcal{Z}_i^m,\ \widetilde{\mathcal{Z}_j^m} = \mathcal{Z}_i^m \setminus \mathcal{Z}_j^m; \\
&\sum_q \kappa_q(q, c) \sum_a \sum_j^{J_q} \rho_j^m(\mathbf{z}_{\rho_j}^m, a) \sum_{\mathbf{z}_{\rho_j}} \xi_j^m(\mathbf{z}_{\rho_j}^m, a) + \sum_o \kappa(o, c) \delta^m(o) \le \widehat{\kappa}^m(c), \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \forall m \in \mathcal{M}, c \in \mathcal{C}; \\
&\sum_a \sum_{\mathbf{z}_j^m} \xi_j^m(\mathbf{z}_j^m, a) = (1-\gamma)^{-1} \qquad\qquad \forall j,\ \mathbf{z}_j; \\
&(1-\gamma) \sum_a \rho_o^m(a, o) \xi_j^m(\mathbf{z}^a g_j) \le \delta^m(o), \qquad\qquad \forall \xi^m, \mathbf{z}_j^m, o; \\
&\xi_j^m(\mathbf{z}_j^m, a) \ge 0, \qquad\qquad \forall a, j,\ \mathbf{z}_j; \\
&\delta^m(o) = \{0, 1\}, \qquad\qquad \forall m \in \mathcal{M}, o \in \mathcal{O}.
\end{aligned} \tag{7.14}$$

As in the flat-MDP model presented in Chapter 3, this MILP has $|\mathcal{M}||\mathcal{O}|$ binary variables, which is an exponential reduction from a naive resource-allocation scheme that has a decision variable per resource bundle (thus requiring on the order of $|\mathcal{M}|2^{|\mathcal{O}|}$ variables). Furthermore, the number of occupation-measure coordinates is also drastically reduced in this factored MILP, compared to the flat variant (3.4). The actual gains in efficiency and the associated losses in quality depend on what basis functions h_k are used, and on how well-structured the MDP is (the latter ultimately defines the domain sizes of the marginal occupation measures ξ).

7.2.1 Experimental Evaluation

In this section, we present an empirical evaluation of the efficacy of the factored resource-allocation MILP (7.14) developed in the previous section. For our experiments, we used an extended version of the factored "SysAdmin" problem (Guestrin, Koller, & Parr, 2001), which has become one of the popular test problems in the factored MDP literature (e.g., Kveton & Hauskrecht, 2004; Guestrin et al., 2003).

The problem involves a network of N computers, each of which might fail with probability that depends on the status of the neighboring computers. The job of the decision-maker (the SysAdmin) is to keep the network running. The state of the system is defined by a set of N binary features, each corresponding to the status of a computer. At each time step the SysAdmin can reboot at most one of the computers and receives a reward for every computer that is functioning properly. We used a network with a unidirectional-ring topology, where each computer influenced and was influenced by the status of a single other computer. The parameters of the transition and reward model are described in detail in (Guestrin et al., 2003).

We extended the domain to the multiagent setting with resources as follows. Each SysAdmin agent was in charge of running its own network of N computers (all agents had the same number of computers they were individually responsible for). In our domain, computer failures were caused by more serious malfunctions (e.g., a virus infection) that could not be fixed by simply rebooting the computer, and thus performing the recovery actions required resources (e.g., specialized software, hardware, or trained personnel). Further, the computers on the network were heterogeneous (mail servers, web servers, data storage, etc.), and thus the recovery of each computer required a different subset of the resources. The total quantity of the resources was limited (e.g., the total number of anti-virus specialists for hire was limited) and, each SysAdmin also had a limited budget, which constrained the amounts of resources it could acquire.

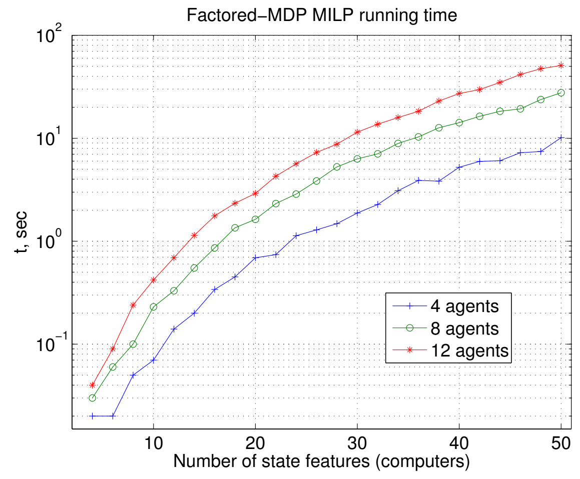
For a local MDP with N computers, there were $|\mathcal{O}| = N$ resources, and each recovery action required two resources, which were randomly selected at the time of problem generation. There was a single capacity cost c (money) and all resources had a unit capacity cost $\kappa(o, c) = 1$ (all recovery tools cost the same amount of money).

A thorough empirical analysis of ALP solutions to the SysAdmin problem is presented in (Guestrin et al., 2003), and it analyzes the quality of solutions and the computational complexity of solving the ALP for various network topologies and different ways of selecting the basis functions. In particular, the experiments in (Guestrin et al., 2003) demonstrate that using basis functions defined over singletons of features leads to high-quality approximations at a very low computational cost. Therefore, in the following discussion, we use the same basis functions and focus on the question of how significantly using the factored representation and ALP affects the computational complexity of the resource-allocation MILP (7.14).

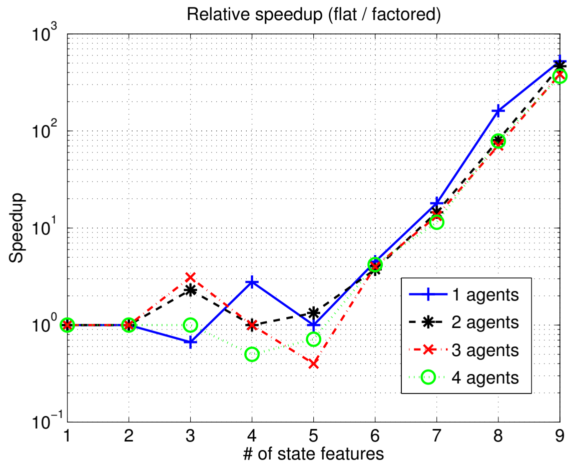
Figure 7.2a shows the running time of the factored resource-allocation MILP as a function of the number of state features (computers) in the problem. As can be seen, the factored MDP approach allows us to easily scale the resource-allocation problem up to MDPs with more than 10^{15} states.

Figure 7.2b demonstrates the corresponding exponential gain in efficiency of the factored MILP (7.14), compared to the one using a flat-MDP model. Recall that the flat MILP (3.4) presented in Chapter 3 by itself commonly achieves exponential speedup over a straightforward resource-allocation mechanism that enumerates all resource bundles (as shown in Section 3.2).

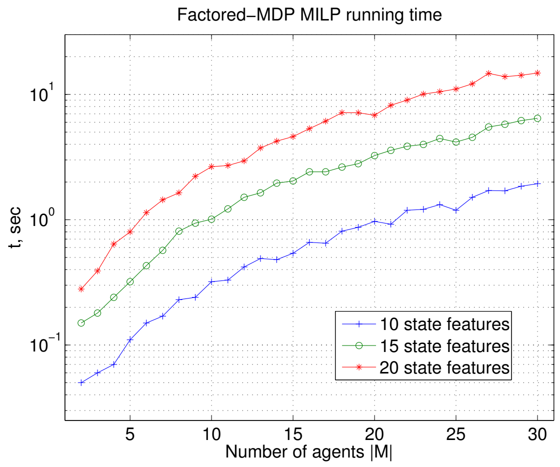
Finally, Figure 7.2c shows the complexity of the factored MILP as a function of the number of agents participating in the resource allocation. The graph demonstrates that although the number of binary variables in the MILP is proportional to the number of agents, the approach shows promise for scaling to large numbers of agents.



(a)



(b)



(c)

Figure 7.2: Efficiency of resource allocation with factored MDPs. (a): Running time of factored resource-allocation MILP as a function of the number of state features. (b): Relative speedup of factored-MDP-based MILP, compared to the flat-MDP MILP; ratio of running times. (c): Running time of the factored resource-allocation MILP as a function of the number of agents.

7.3 Composite ALP

The dual ALP approach discussed in the previous sections can lead to very computationally efficient approximations for MDPs (and the associated resource-allocation problems), but the method does have a weak point. As discussed in Section 7.1.3, to guarantee global consistency of marginal occupation measures, it is necessary to convert the cluster graph associated with the domains of the marginals into a junction tree. The size of the resulting marginals is exponential in the induced width of the original cluster graph, which in some cases can lead to poor scaling. For example, if in the SysAdmin problem discussed in the previous section, the additive terms in the reward function were defined over all possible pairs of features (instead of just singles), the size of the factored

MDP model would be quadratic in the number of state features, but the size of the DALP approximation would be exponential (defining basis function over pairs of features would have the same effect). This tendency is exacerbated in MDPs with consumable resources, where the resource-cost functions $\rho_q(a, q)$ aid with the propagation of interdependencies between state features.

In this section we present an alternative ALP formulation, the *composite ALP*, which works with the occupation-measure coordinates and is thus well-suited for constrained MDPs, but avoids the "curse of the induced width".[7]

The ADLP (7.5) approximates the dual variables x, which is equivalent to approximating the feasible region of the primal LP (7.4); the primal ALP does the opposite. We can combine the two by applying the dual approximation $x = Qy$ to the DALP (7.7):

$$\begin{aligned} &\max r^T Qy \\ &\text{subject to:} \\ &\quad H^T A^T Q y = H^T \alpha, \\ &\quad Qy \geq 0. \end{aligned} \tag{7.15}$$

This ALP still has an exponential number ($|\mathcal{S}||\mathcal{A}|$) of constraints in $Qy \geq 0$, but this can be resolved in several ways. These constraints can be reformulated using the non-serial dynamic programming approach (Bertele & Brioschi, 1972) (analogously to its application in (Guestrin et al., 2003)), yielding an equivalent, but smaller, constraint set. Another approach is to simply restrict attention to non-negative basis functions Q and replace the constraints with a stricter condition $y \geq 0$ (introducing another source of approximation error). We will adopt the latter approach (which works quite well, as shown by our experiments, presented in Section 7.3.1), leading to the following LP:

$$\begin{aligned} &\max r^T Qy \\ &\text{subject to:} \\ &\quad H^T A^T Q y = H^T \alpha, \\ &\quad y \geq 0. \end{aligned} \tag{7.16}$$

The above gives the dual form of the composite ALP. The equivalent primal form is:

$$\begin{aligned} &\min \alpha^T H w \\ &\text{subject to:} \\ &\quad Q^T A H w \geq Q^T r. \end{aligned} \tag{7.17}$$

The primal form has K variables (one per primal basis function h_k) and L constraints (one per dual basis function q_l); the dual form has L variables and K constraints. Thus, the composite ALP combines the efficiency gains of the primal and the dual approximations. However, as in the case of the primal and the dual ALPs, the usefulness of the composite ALP is contingent upon our ability to efficiently compute the coefficients of its objective function and constraints. The objective functions of the two forms are the same as in the primal and the dual ALPs, respectively, so both can be computed efficiently as described

[7]To our knowledge, this term was coined by Guestrin (2003).

in the earlier sections. Thus, the important question is whether the constraint coefficients can be computed efficiently.

A first glance at the constraints conveys some pessimism because of the term AQ, which was the stumbling block in directly approximating the dual LP (Section 7.1.2). However, despite that, the computation can be carried out efficiently if we apply the primal approximation first and then the dual approximation to the result. Consider the primal approximation:

$$\begin{aligned}(AH)_{sa,k} &= h_k\big(\mathbf{z}_{h_k}(s)\big) - \gamma \sum_{\mathbf{z}_{h_k}} \prod_{n:z_j \in \mathcal{Z}_{h_k}} p_j\big(z_j|\mathbf{z}_{p_j}(s),a\big) h_k(\mathbf{z}_{h_k}) \\ &= h_k(\mathbf{z}_{h_k}) - \psi(\mathbf{z}_{\psi_k},a),\end{aligned}$$

where we introduced ψ_k to refer to the second term, which is a compact function whose domain is the union of the DBN parents of all features that are in the domain of the k^{th} basis function: $\mathcal{Z}_{\psi_k} = \bigcup_{j:z_j\in\mathcal{Z}_{h_k}} \mathcal{Z}_{p_j}$. Applying the dual approximation to the result:

$$\begin{aligned}\big(Q^T(AH)\big)_{l,k} &= \sum_{s,a} q_l(s,a)\Big(h_k(\mathbf{z}_{h_k}(s)) - \psi(\mathbf{z}_{\psi_k},a)\Big) \\ &= \sum_{\mathbf{z}_{h_k}\bigcup \mathbf{z}_{q_l}} q_l(\mathbf{z}_{q_l},a) h_k(\mathbf{z}_{h_k}) - \sum_{\mathbf{z}_{\psi_k}\bigcup \mathbf{z}_{q_l},a} q_l(\mathbf{z}_{q_l},a)\psi(\mathbf{z}_{\psi_k},a)\end{aligned}$$

The first term can be computed efficiently by summing over $\mathcal{Z}_{h_k}\bigcup\mathcal{Z}_{q_l}$, the union of the domains of the k^{th} primal and the l^{th} dual basis function. The second term is obtained by summing over the action space and $\mathcal{Z}_{\psi_k}\bigcup\mathcal{Z}_{ql} = \big(\bigcup_{j:z_j\in\mathcal{Z}_{h_k}}\mathcal{Z}_{p_j}\big)\bigcup\mathcal{Z}_{ql}$, the domain of the dual basis function q_l and the union of the DBN parents of all features in the domain of h_k. Therefore, the coefficients of the composite constraint matrix can be computed efficiently by summing over relatively small domains (assuming the domains of all basis functions are small and the transition DBN is well-structured).

Example 7.5. *Consider $\mathcal{S}$, H, and p, as in the earlier examples from Section 7.1.1: $\mathcal{S} = z_1 \times z_2 \times z_3$, $H = [h_1(z_1), h_2(z_2,z_3), h_3(z_3)]$, and*

$$p(z_1',z_2',z_3'|z_1,z_2,z_3) = p_1(z_1'|z_1)p_2(z_2'|z_1,z_2)p_2(z_3'|z_3).$$

Then, for $k=3$, we have:

$$(AH)_{sa,3} = h_3\big(z_3(s)\big) - \gamma\sum_{z_3'} p\big(z_3'|z_3(s),a\big)h_3(z_3')$$

Thus, $\mathcal{Z}_{\psi_3} = \{z_3\}$, and ψ_3 can be computed efficiently by summing over z_3'. Multiplying by Q, we get for $l=2$:

$$\begin{aligned}(Q^TAH)_{2,3} &= \sum_{s,a}(Q^T)_{2,sa}(AH)_{sa,3} \\ &= \sum_{z_2,z_3} q_2(z_2)h_3(z_3) - \gamma\sum_{z_2,z_3} q_2(z_2)\sum_{z_3'} p\big(z_3'|z_3,a\big)h_3(z_3')\end{aligned}$$

which can be computed by summing over $z_2\times z_3$. ■

Another important issue to consider is the feasibility and boundedness of the composite ALPs (7.17) and (7.16). All ALPs that approximate only the optimization variables (primal ALP (7.4); its dual, DALP (7.7); the dual approximation, ADLP (7.5)) are bounded, because the approximation limits the search to a subset of possible solutions. Feasibility of the primal ALP (7.4) can also be ensured by adding a constant to H (de Farias & Van Roy, 2003). In the case of the composite ALP, where both the feasible region and the objective function are approximated, ensuring boundedness and feasibility is slightly more complicated.

Lemma 7.6. *The primal form of the composite ALP (7.17) is feasible for any dual basis $Q \geq 0$ and any primal basis H that contains a constant function $h_k(\mathbf{z}_{h_k}) = 1$.*

Proof. By the results of de Farias and Van Roy (de Farias & Van Roy, 2003), the primal ALP (7.4) is feasible whenever the primal basis H contains a constant. Call a feasible solution to the primal ALP w^*. By definition, w^* satisfies

$$AHw^* \geq r.$$

Then, for any $Q \geq 0$, $Q^T AHw^* \geq Q^T r$ also holds, meaning that (7.17) also has a feasible solution. ■

In other words, introducing a dual approximation Q only enlarges the feasible region of the primal form of the composite ALP (7.17), thus ensuring its feasibility. Unfortunately, (7.17) is not, in general, bounded, because the dual basis Q might contain too few constraints. Intuitively, to bound (7.17), we need at least as many constraints as optimization variables. Therefore, an important question is: Given a primal basis H, how big must the dual basis Q be to ensure the boundedness of the primal form (7.17), or, equivalently, the feasibility of the dual form (7.16)?

Lemma 7.7. *For any primal basis H ($|\mathcal{S}| \times K$), there exists a dual basis Q ($|\mathcal{S}||\mathcal{A}| \times L$), such that the number of dual basis functions does not exceed the number of primal functions ($L \leq K$), and the dual form of the composite ALP (7.16) is feasible for H and Q.*

Proof. A flat set of constraints $A^T x = \alpha$ is always feasible, thus there also always exists a solution to

$$H^T A^T x = H^T \alpha.$$

Let $\text{rank}(H^T A^T) = m \leq K$. Then, let us reorder rows and columns such that the upper-left $m \times m$ corner of $H^T A^T$ is non-singular. Let the dual basis contain m linearly independent functions and reorder the rows of Q such that the top m rows are also linearly independent. Then, $H^T A^T Q$ will be $K \times m$, with a non-singular $m \times m$ matrix in the upper-left corner. The remaining rows and the corresponding coefficients of the right-hand side $H^T \alpha$ will be their linear combinations. Thus, the resulting system $H^T A^T Q y = H^T \alpha$ will have a solution. ■

Therefore, for any primal basis H with K functions, there exists a dual basis Q with $L \leq K$ functions, which guarantees feasibility of the dual form of the composite ALP (7.16). By standard properties of LPs, this ensures the boundedness of the primal form (7.16), thus ensuring a feasible and bounded solution to both. Intuitively, the program (7.16) has more variables than equations ($L > K$), thus it is usually feasible. Therefore, from a practical standpoint, ensuring boundedness and feasibility of the composite ALPs is not difficult (when $L > K$, all but the most degenerate systems are underconstrained). This was confirmed by our experiments (presented in the next section), where using a meaningful dual basis with several times more functions than the primal (but on the same order of magnitude), resulted in feasible instantiations of the program (7.16).

7.3.1 Experimental Evaluation

As mentioned earlier, the main driving force behind the composite ALP was to construct an efficient ALP in the dual coordinates that was not exponential in the induced width of the cluster graph defined by the marginals. However, the approach can also be applied to unconstrained problems. Therefore, since there is a wider variety of algorithms for unconstrained MDPs, we focus on unconstrained domains in our analysis. This ignores one of the advantages of the composite ALP, but gives a more direct and clearer comparison to other methods.

We evaluated the composite ALP on the unconstrained variant of the "SysAdmin" problem (Guestrin et al., 2003), which was earlier described in Section 7.2.1. Figure 7.3a compares the values of policies for a problem with a unidirectional-ring network. The values of policies obtained by the following methods are compared:

1. **Optimal**: the optimal policy, which was obtained by "flattening out" the factored MDP and using the exact LP to solve the resulting MDP.

2. **Primal ALP**: the primal ALP (7.4) with basis functions over all pairs of features. The size of problem input grew quadratically with the number of state features. Note that for unconstrained problems, the primal ALP (7.4) is equivalent to the DALP (7.7).

3. **Composite**: the composite ALP (7.17) with the same primal basis as in the primal ALP above and a dual basis defined over triplets of neighbors. The size of problem input grew quadratically with the number of state features.

4. **Primal ALP (singles)**: the primal ALP (7.4) with basis functions over single features. The size of problem input grew linearly with the number of state features.

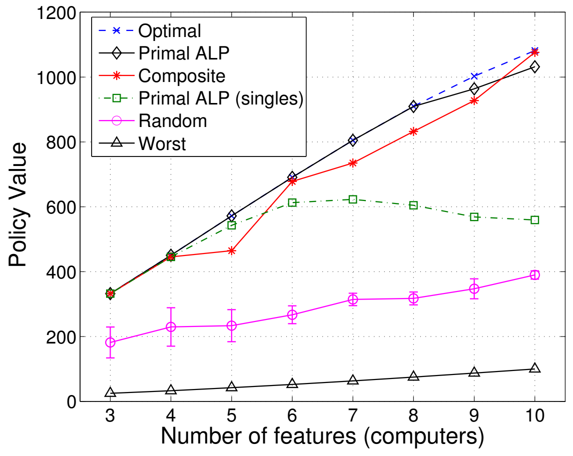
5. **Random**: a policy obtained by selecting a random deterministic action for every state.

6. **Worst**: a policy, obtained by negating the reward function.

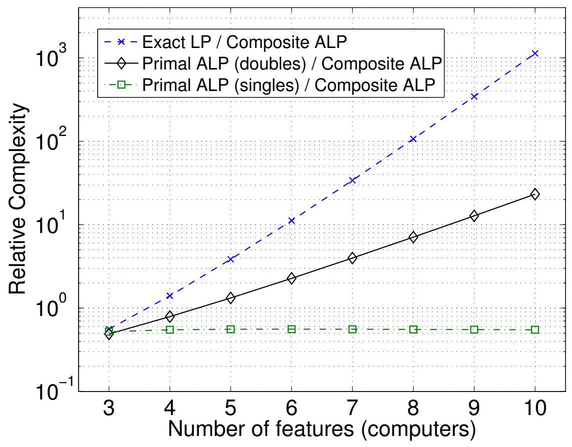
All policies were evaluated in closed form using a "flattened out" version of the MDP.

We performed no optimizations of the basis functions, and only used very simple functions, such as constants, binary indicators, and identity matrices. The plot shows the actual values of policies (not the value functions), which is a more accurate metric, as constraint approximation in the composite ALP can lead to unrealizable value functions.[8]

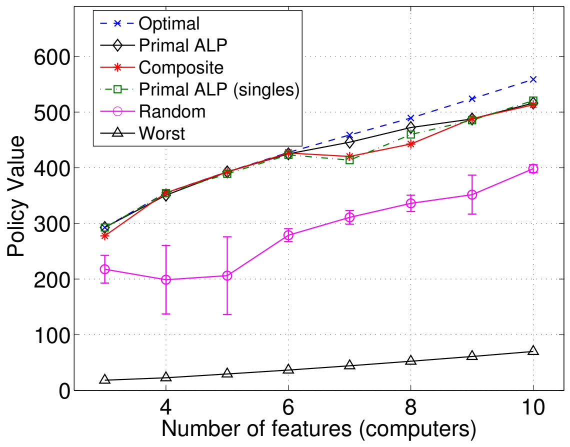
[8]Given the same primal basis, the composite ALP will, in general, produce lower-quality solutions than the primal ALP, because it also approximates the feasible region. The data point in Figure 7.3a, corresponding to 10 computers, is unusual. The value function of the composite ALP maps to a better policy than a more accurate value function of the primal ALP.



(a)



(b)



(c)

Figure 7.3: Comparison of ALP methods. (a): quality, unidirectional ring; (b): relative efficiency; (c): quality, bidirectional ring.

Figure 7.3b shows the efficiency gains of the composite ALP, relative to the exact LP and the two primal ALP variations. A problem where each pair of variables appears in at least one function has an induced width that equals the total number of state variables. Thus, the composite ALP achieves exponential speedup, compared to a primal ALP or a DALP with a basis set defined on all pairs of features, but without a significant loss in quality (Figure 7.3a). The complexity of the composite ALP in these experiments roughly matches the complexity of the primal ALP (or the DALP) with basis functions over single features (Figure 7.3b), but the composite ALP produces noticeably better policies (Figure 7.3a). Of course, for better-structured problems, basis functions over single features might suffice, such as for the more symmetric bidirectional-ring network (Figure 7.3c).

Overall, our experiments indicate that the composite ALP can be a viable alternative to the other ALP approaches. By symmetrically, but independently, approximating the space of solutions and the feasible regions of the LPs, the composite ALP avoids the exponential explosion in the induced width of the cluster graph of the problem. Overall, our results confirm the intuition behind composite ALPs: if the dimensionality of the problem is greatly reduced via an approximation of the solution space, then using the exact feasible region can be wasteful.

7.4 Discussion: Folding Resources into MDPs

Throughout this thesis, the resources are modeled as external to the planning problem. This is a requirement for resource-allocation in competitive multiagent settings, where each of the agents had its own MDP and there is a central authority in charge of allocating the resources. On the other hand, in cooperative environments, an alternative strategy is to model the resource information within the multiagent MDP and provide the agents with actions for obtaining the resources. A solution to such an MDP would yield an optimal policy for allocating the resources as well as acting in the stochastic environment.

However, this approach is not computationally tractable within flat MDP models, as it would lead to an increase in the size of the state space that is both exponential in the number of agents and the number of resources (the transitions would become coupled once the resources are folded into the state space). Avoiding this exponential scaling in the reduction of the resource-allocation problem to an unconstrained MDP would be impossible, because the opposite would mean a polynomial-time solution to an NP-complete problem.

The above argues against folding the resource information into the state space of a flat MDP, but it does not preclude the possibility of representing the resulting multiagent MDP in a factored manner, leading to compact MDPs that were studied in this chapter. However, the challenge with this approach is that, given the global resource bounds and the local capacity constraints, the transition function of the resulting MDP does not seem to factor well. Indeed, the dynamics of a feature that corresponds to the status of one resource will be affected by the values of all other resource-related features. Further, even if a factored decomposition of the resulting multiagent problem were possible, a high-quality computationally efficient ALP solution could only be obtained for domains where the effects of the resources on each other were nearly additive (with all non-additive effects being captured via the basis functions defined over small domains).

While we believe that this approach of folding the resource information into a large multiagent MDP is not the most appropriate for the problems that are the focus of this thesis (doing the latter would actually lead to loss of information about structure), a more detailed investigation of the factored-MDP approach and a careful comparison to our methodology could be a useful direction of future work.

7.5 Conclusions

In this chapter we showed how the techniques that we developed earlier for solving constrained MDPs and resource-allocation problems can be adapted to work with factored MDPs. The main contributions of the work presented here are as follows:

- We formulated an MILP for approximately allocating consumable and non-consumable resources for problems where the agents' preferences over the resources are defined by factored MDPs. These factored analogs of the algorithms developed earlier in

Chapters 3 and 5 allow scaling to much larger stochastic planning problems in the resource allocation mechanisms, as we demonstrated empirically in Section 7.2.1.

- We analyzed a linear approximation of the dual LP and presented a negative result that demonstrates that linear approximations do not interact with the dual formulation as well as they do with the primal.

- Finally, we developed a new ALP method, the composite ALP, which symmetrically approximates the primal and the dual optimization variables. A benefit of the composite ALP is that it is not subject to the "curse of the induced width", which plagues the primal ALP (7.4) and its dual (7.7), and as we showed experimentally in Section 7.3.1, it can be an efficient alternative to other ALP methods.

CHAPTER 8

Conclusions

This thesis considers the problem of resource allocation for domains where the preferences of agents for the resources are induced by stochastic-planning problems (modeled as Markov decision processes). We demonstrated that, by taking an integrated approach to resource allocation and planning, and by exploiting structure in agents' preferences due to the underlying MDPs as well as the structure within the MDPs, we can develop computationally efficient resource-allocation mechanisms that scale up to very large domains.

In this chapter we conclude with a summary of the main contributions of the work, along with a discussion of its limitations and an overview of future directions and open problems.

8.1 Summary of Contributions

Conceptually, this dissertation can be broken into three parts. The first part, consisting of Chapters 2, 3, and 4, deals with models involving non-consumable resources. The second part, which corresponds to Chapter 5, discusses policy-optimization and resource-allocation problems with consumable resources. Finally, the third part, consisting of Chapters 6 and 7, focuses on graphical and factored MDP models. The overarching goal of all three parts is on developing efficient planning and resource-allocation algorithms by exploiting problem structure: parts one and two exploit structure due to agents' MDP-induced preferences (for non-consumable and consumable resources, respectively), and part three discusses methods for exploiting structure present in MDPs themselves. Below, we briefly summarize the main results and contributions of this work, with an emphasis on the connections between the parts.

I. **Non-consumable Resources**

   (i) **Single-Agent Constrained MDP Model**
   In Chapter 2, we developed a new MDP model, where the action space was parameterized by the available resources, and the space of feasible policies was constrained by capacity limits.

   - We showed that the model is fully general, meaning that it captures arbitrary non-decreasing combinatorial preferences over resources.
   - We analyzed the resulting constrained MDP. In particular, we showed that the problem is NP-complete, optimal policies for this constrained MDP are stationary deterministic, and uniformly optimal policies do not, in general, exist.

  - We developed a new MILP-based formulation for this NP-complete constrained MDP. Further, we showed how the model can be significantly simplified for the case of binary resource requirements.

(ii) **Multiagent Resource-Allocation Mechanism**
In Chapter 3, we developed a new computationally efficient auction-based resource-allocation mechanism for non-consumable resources that exploits the structure in agents' MDP-induced preferences, as defined by the model of Chapter 2.

  - We showed analytically and empirically that the mechanism achieves substantial (often exponential) speedup over a straightforward combinatorial resource-allocation mechanism that requires a flat enumeration of all valuable resource bundles. In our mechanism, the winner-determination problem remains NP-complete, but the structure in agents' preferences allows us to exponentially reduce the number of integer decision variables in the optimization.
  - We developed a distributed version of the winner-determination problem that leads to further speedup. The distributed version of the mechanism remains strategy-proof and (to a large extent) maintains information privacy.

(iii) **Constrained MDPs with Multiple Discounts**
In Chapter 4, we demonstrated how techniques, similar to the ones developed in Chapter 2 for solving resource-constrained MDPs, can be applied to other constrained MDP models that have been previously formulated in the literature, but for which no implementable algorithms had been known.

  - We presented an MILP formulation for the problem of finding optimal stationary deterministic policies for infinite-horizon MDPs with constraints on expected total discounted cost. This NP-complete problem had been studied before by Feinberg (2000), but only non-linear, non-convex formulations had been available for this problem.
  - We presented an MILP formulation for the problem of finding optimal stationary deterministic policies for constrained MDPs with multiple differently discounted streams of rewards and costs. This problem had also been analyzed prior to this work (Feinberg & Shwartz, 1994, 1995, 1996), had been shown to be NP-complete, but only non-convex formulations had been previously known.

II. **Consumable Resources**

In Chapter 5, we analyzed the resource-allocation and planning problems involving consumable resources. The main challenge in these problems was due to the fact that the definition of the value of a resource bundle is ambiguous, because the total resource consumption of a given policy is non-deterministic. We analyzed both the risk-neutral case (constraints are imposed on the expected resource consumption) as well as the risk-sensitive model (constraints are imposed on the probability of exceeding allowable resource thresholds).

- For the MDP with probabilistic constraints on resource usage, we obtained a new complexity result showing this problem to be NP-hard. We also developed a linear approximation to this problem based on the Markov bound as well as an iterative procedure for adjusting the bound for obtaining higher-quality policies.

- We presented a non-linear approximation to the MDP with probabilistic constraints, based on Legendre polynomials. In the end, the problem of finding optimal stationary randomized policies was reduced to a quadratic optimization problem, and the problem of finding optimal stationary deterministic policies was reduced to an MILP. This Legendre approximation can also be applied more generally to MDPs with arbitrary utility functions.
- We developed a multiagent resource-allocation mechanism for distributing consumable resources, based on the risk-neutral MDP model. As we showed analytically and empirically, the complexity of the resource-allocation mechanism for consumable resources is significantly lower than that of its counterpart for non-consumable resources, developed in Chapter 3. The resource-allocation mechanism developed in that chapter can be directly integrated with the mechanism for non-consumable resources.

III. **Factored and Graphical MDP Models.**

(i) **Graphical MDPs with Local and Asymmetric Dependencies**
In Chapter 6, we analyzed a graphical model of multiagent MDPs where the agents' effects on each other are asymmetric and are confined to local neighborhoods. We considered MDPs with the social-welfare optimization criterion as well as multiagent problems with self-interested agents. The main message of that chapter was that very strong conditions are needed in order for a well-structured MDP to have a well-structured optimal solution. The purpose of that chapter was to lay the groundwork and provide justification for the approximate methods of the following chapter.

(ii) **Approximate Planning with Factored MDPs**
In Chapter 7, we showed that the resource-allocation methods discussed in this thesis are compatible with factored MDP representations and solution algorithms, which allows scaling up to stochastic planning problems with very large state spaces.

- We developed a factored counterpart to the resource-allocation mechanisms discussed in Chapters 3 and 5 for non-consumable and consumable resources, respectively.
- We presented a negative result, which shows that linear approximations do not interact as well with the dual LP variables as they do with the primal.
- We developed a new ALP method for factored MDPs that symmetrically approximates the primal and the dual coordinates of the MDP (the value function and the occupation measure). This method is suitable for constrained problems, scales better than other ALP approaches, and as we showed experimentally, can be used to produce high-quality solutions.

8.2 Open Questions and Future Directions

There are many questions that remain open in the area explored by this dissertation. Below, we briefly outline a few of the possible directions of future work and discuss preliminary ideas on pursuing these topics.

Continuous state and action spaces
We used MDPs with discrete and finite state and action spaces as our model of the agents' stochastic planning problems. In many realistic cases, however, the state space is defined by continuous quantities, and there is a continuous spectrum of actions available to the agent. There is a significant body of prior work on MDPs with continuous state and action spaces, including work on constrained MDPs (Altman & Shwartz, 1991; Altman, 1999), and factored MDPs with continuous state and action features (Hauskrecht & Kveton, 2006; Kveton & Hauskrecht, 2005; Guestrin, Hauskrecht, & Kveton, 2004). The linear programming formulation of MDPs at the core of all planning and resource-allocation algorithms presented in this thesis also serves as the basis for much of the above-mentioned work on continuous MDPs. Therefore, the approaches used in our resource-allocation methods and the techniques used for solving continuous MDPs appear compatible and complementary. Extending our MDP-based resource-allocation mechanisms to continuous MDPs appears to be a viable and fruitful direction.

Partial observability
Another possible direction along the lines of relaxing assumptions on the model of the agents' planning problem is to extend the work to partially observable MDPs (POMDPs) (Smallwood & Sondik, 1973). Traditional planning methods for POMDPs are based on dynamic programming, which hinders the incorporation of constraints into the model, as needed by our resource-allocation mechanism. However, the recent work on approximate linear programming formulations for factored MDPs with continuous state and action variables (mentioned in the previous item) could also pave the way to extending our resource-allocation mechanisms to partially-observable environments. A possible approach is to reduce the partially-observable MDP to an observable MDP defined on the continuous belief space, with one continuous variable for every state in the original POMDP, and to apply the methods for continuous factored MDPs to the resulting problem.

Context-specific structure in factored MDPs
In Chapter 7, we showed how our resource-allocation methods can be adapted to work with factored MDPs under linear approximation. The ALP methods that we used for solving factored MDPs allow one to exploit additive structure in MDPs. There is also another type of structure that is often present in MDPs: the context-specific structure (Boutilier et al., 1995; Dearden & Boutilier, 1997; Boutilier, Dean, & Hanks, 1999; Boutilier et al., 2000). This type of structure stems from the fact that some state or action features are irrelevant in some contexts (defined by instantiations of other features). Developing resource-allocation mechanisms that can exploit both additive structure and context-specific structure would further the applicability of these methods to domains with even larger planning problems. Of potential relevance to this task is the work by Guestrin (2003) that develops ALP solutions that exploit both forms of structure.

Heuristic methods for non-consumable resources
For the case of non-consumable resources, we focused on exact methods for obtaining optimal resource allocations and policies. However, the resulting optimization problems are NP-complete, and exploring approximate solutions would be a worthwhile effort. In (Dolgov & Durfee, 2004c), we performed a preliminary investigation of search-based

heuristic approximations to the NP-complete winner-determination problem for non-consumable resources. However, more work needs to be done in exploring the properties of such approximations and developing smarter heuristics.

Approximate mechanisms and non-cooperative agents
Most of the optimal resource-allocation and planning methods developed in this thesis are applicable to both cooperative and competitive multiagent environments. However, applying the approximate algorithms in competitive environments might be problematic, because the resulting mechanism might no longer be strategy-proof (Kfir-Dahav et al., 2000; Parkes, 2001). That is, agents might be able to strategically exploit the source of the approximation error to their benefit by deviating from truth-telling strategies. However, as mentioned earlier in Chapter 5, exploiting the mechanism in one's favor might be computationally intractable (e.g., Sanghvi & Parkes, 2004; Bartholdi et al., 1989; Rothkopf et al., 1998; Procaccia & Rosenschein, 2006). An analysis of such issues for the approximate resource-allocation mechanisms discussed in this thesis could provide the justification for using these approximate methods in competitive settings.

8.3 Closing Remarks

Both combinatorial resource allocation and stochastic planning are, in general, not tractable: MDPs are subject to the curse of dimensionality and combinatorial resource allocation suffers from the curse of the exponential bundle space. Despite (or perhaps because of) that, both resource allocation and stochastic planning are active areas of research and significant effort is being invested in the development of computationally feasible algorithms for these problems that would allow them to scale up to practical settings. In particular, methods that aim to reduce computational complexity by lowering the dimensionality of the problem are gaining momentum (for these problems as well as in other areas of AI). The work presented in this dissertation falls within the scope of this larger effort: we demonstrated that by integrating the problems of resource allocation and stochastic planning, we can fruitfully exploit structure within these problems to significantly lower their computational complexity.

While many open problems remain (a small number of which were outlined in the previous section), this thesis demonstrates the effectiveness of the integrated approach to resource allocation and planning in stochastic environments. Therefore, we believe that the models and solution algorithms developed in this dissertation significantly further the applicability and general usefulness of both resource-allocation mechanisms and multiagent stochastic planning in practical settings.

Bibliography

Abramowitz, M., & Stegun, I. A. (1965). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover Publications, Inc., New York.

Altman, E. (1996). Constrained Markov decision processes with total cost criteria: Occupation measures and primal LP. *Methods and Models in Operations Research*, *43*(1), 45–72.

Altman, E. (1998). Constrained Markov decision processes with total cost criteria: Lagrange approach and dual LP. *Methods and Models in Operations Research*, *48*, 387–417.

Altman, E., & Shwartz, A. (1991). Adaptive control of constrained Markov chains: Criteria and policies. *Annals of Operations Research, special issue on Markov Decision Processes*, *28*, 101–134.

Altman, E. (1999). *Constrained Markov Decision Processes*. Chapman and HALL/CRC.

Arnborg, S., Corneil, D. G., & Proskurowski, A. (1987). Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods*, *8*(2), 277–284.

Bartholdi, J. J., Tovey, C. A., & Trick, M. A. (1989). The computational difficulty of manipulating an election. *Social Choice and Welfare*, *6*(3), 227–241.

Bellman, R. (1961). *Adaptive Control Processes: A Guided Tour*. Princeton University Press.

Benazera, M. E., Brafman, R. I., Mealeau, N., & Hansen, E. (2005). Planning with continuous resources in stochastic domains. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, pp. 1244 – 1251.

Bertele, U., & Brioschi, F. (1972). *Nonserial Dynamic Programming*. Academic Press.

Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific.

Bertsimas, D., & Tsitsiklis, J. (1997). *Introduction to Linear Optimization*. Athena Scientific.

Boutilier, C. (2002). Solving concisely expressed combinatorial auction problems. In *Eighteenth National Conference on Artificial Intelligence*, pp. 359–366.

Boutilier, C., Dean, T., & Hanks, S. (1999). Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, *11*, 1–94.

Boutilier, C., Dearden, R., & Goldszmidt, M. (1995). Exploiting structure in policy construction. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1104–1111.

Boutilier, C., Dearden, R., & Goldszmidt, M. (2000). Stochastic dynamic programming with factored representations. *Artificial Intelligence*, *121*(1-2), 49–107.

Boutilier, C., & Hoos, H. H. (2001). Bidding languages for combinatorial auctions. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pp. 1211–1217.

Bowling, M., & Veloso, M. (2004). Existence of multiagent equilibria with limited agents. *Journal of Artificial Intelligence Research*, *22*, 353–384.

Chevaleyre, Y., Dunne, P. E., Endriss, U., Lang, J., Lemaitre, M., Maudet, N., Padget, J., Phelps, S., Rodriguez-Aguilar, J. A., & Sousa, P. (2006). Issues in multiagent resource allocation. *Informatica*.

Clarke, E. H. (1971). Multipart pricing of public goods. *Public Choice*, *18*, 19–33.

de Farias, D. P., & Van Roy, B. (2003). The linear programming approach to approximate dynamic programming. *Operations Research*, *51*(6), 850–856.

de Farias, D., & Van Roy, B. (2004). On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of Operations Research*, *29*(3), 462–478.

de Souza e Silva, E., & Gail, H. R. (1989). Calculating availability and performability measures of repairable computer systems using randomization. *Journal of the ACM*, *36*(1), 171–193.

de Souza e Silva, E., Gail, H. R., & Campos, R. V. (1995). Calculating transient distributions of cumulative reward. In *Proceedings of the 1995 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, pp. 231–240. ACM Press.

de Vries, S., & Vohra, R. V. (2003). Combinatorial auctions: A survey. *INFORMS J. on Computing*, *15*(3), 284–309.

Dean, T., & Kanazawa, K. (1989). A model for reasoning about persistence and causation. *Computational Intelligence*, *5*(3), 142–150.

Dearden, R., & Boutilier, C. (1997). Abstraction and approximate decision-theoretic planning. *Artificial Intelligence*, *89*(1-2), 219–283.

Dolgov, D. A., & Durfee, E. H. (2003). Approximating optimal policies for agents with limited execution resources. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pp. 1107–1112.

Dolgov, D. A., & Durfee, E. H. (2004a). Approximate probabilistic constraints and risk-sensitive optimization criteria in Markov decision processes. In *Proceedings of the Eighth International Symposiums on Artificial Intelligence and Mathematics (AI&M 7-2004)*, Florida.

Dolgov, D. A., & Durfee, E. H. (2004b). Graphical models in local, asymmetric multi-agent Markov decision processes. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-04)*, pp. 956–963.

Dolgov, D. A., & Durfee, E. H. (2004c). Optimal resource allocation and policy formulation in loosely-coupled Markov decision processes. In *Proceedings of the Fourteenth*

International Conference on Automated Planning and Scheduling (ICAPS 04), pp. 315–324.

Dolgov, D. A., & Durfee, E. H. (2005a). Computationally-efficient combinatorial auctions for resource allocation in weakly-coupled MDPs. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-05)*, pp. 657–664, New York, NY, USA. ACM Press.

Dolgov, D. A., & Durfee, E. H. (2005b). Locality and asymmetry in large-scale multiagent MDPs. In Scerri, P., Vincent, R., & Mailler, R. (Eds.), *Coordination of Large-Scale Multiagent Systems*. Springer.

Dolgov, D. A., & Durfee, E. H. (2005c). Stationary deterministic policies for constrained MDPs with multiple rewards, costs, and discount factors. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, pp. 1326–1331.

Dolgov, D. A., & Durfee, E. H. (2006). Symmetric primal-dual approximate linear programming for factored MDPs. In *Proceedings of the Ninth International Symposiums on Artificial Intelligence and Mathematics (AI&M 2006)*, Florida.

Donatiello, L., & Grassi, V. (1991). On evaluating the cumulative performance distribution of fault-tolerant computer systems. *IEEE Trans. Comput.*, *40*(11), 1301–1307.

Eckstein, J., Phillips, C., & Hart, W. (2000). Pico: An object-oriented framework for parallel branch and bound. In *Proceedings of the Workshop on Inherently Parallel Algorithms in Optimization and Feasibility and their Applications*.

Ejov, V., Filar, J., & Gondzio, J. (2004). An interior point heuristic for the Hamiltonian cycle problem via Markov decision processes. *Journal of Global Optimization*, *29*(3), 315 – 334.

Feinberg, E., & Shwartz, A. (1996). Constrained discounted dynamic programming. *Mathematics of Operations Research*, *21*, 922–945.

Feinberg, E., & Shwartz, A. (1999). Constrained dynamic programming with two discount factors: Applications and an algorithm. *IEEE Transactions on Automatic Control*, 628–630.

Feinberg, E. A., & Shwartz, A. (1994). Markov decision processes with weighted discounted criteria. *Mathematics of Operations Research*, *19*, 152–168.

Feinberg, E. A., & Shwartz, A. (1995). Constrained Markov decision processes with weighted discounted criteria. *Mathematics of Operations Research*, *20*, 302–320.

Feinberg, E. A. (2000). Constrained discounted Markov decision processes and Hamiltonian cycles. *Mathematics of Operations Research*, *25*(1), 130–140.

Feldmann, R., Gairing, M., Lucking, T., Monien, B., & Rode, M. (2003). Selfish routing in non-cooperative networks: A survey. In *28th International Symposium on Mathematical Foundations of Computer Science (MFCS-2003)*, pp. 21–45. Springer-Verlag.

Ferguson, D., Nikolaou, C., Sairamesh, J., & Yemini, Y. (1996). Economic models for allocating resources in computer systems. In Clearwater, S. (Ed.), *Market-Based*

Control: A Paradigm for Distributed Resource Allocation, pp. 156–183, Hong Kong. World Scientific.

Filar, J. A., & Krass, D. (1994). Hamiltonian cycles and Markov chains. *Mathematics of Operations Research*, *19*, 223–237.

Fortnow, L., Kilian, J., Pennock, D. M., & Wellman, M. P. (2005). Betting boolean-style: A framework for trading in securities based on logical formulas. *Decision Support Systems*, *39*(1), 87 – 104.

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co.

Ghavamzadeh, M., & Mahadevan, S. (2002). A multiagent reinforcement learning algorithm by dynamically merging Markov decision processes. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems AAMAS-02*, pp. 845–846, New York, NY, USA. ACM Press.

Groves, T. (1973). Incentives in teams. *Econometrica*, *41*(4), 617–631.

Guestrin, C., Koller, D., & Parr, R. (2001). Multiagent planning with factored MDPs. In *Proceedings of the 14th Neural Information Processing Systems (NIPS-14)*.

Guestrin, C., Koller, D., Parr, R., & Venkataraman, S. (2003). Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, *19*, 399–468.

Guestrin, C. (2003). *Planning Under Uncertainty in Complex Structured Environments.* Ph.D. thesis, Computer Science Department, Stanford University.

Guestrin, C., Hauskrecht, M., & Kveton, B. (2004). Solving factored MDPs with continuous and discrete variables. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pp. 235–242.

Hauskrecht, M., & Kveton, B. (2006). Approximate linear programming for solving hybrid factored MDPs. In *Proceedings of the 9th International Symposium on Artificial Intelligence and Mathematics*.

Heyman, D. P., & Sobel, M. J. (1984). *Volume II: Stochastic Models in Operations Research.* McGraw-Hill, New York.

Hoos, H. H., & Boutilier, C. (2000). Solving combinatorial auctions using stochastic local search. In *AAAI/IAAI*, pp. 22–29.

Howard, R., & Matheson, J. (1972). Risk-sensitive Markov decision processes. *Management Science*, *18*(7), 356–369.

Huang, Y., & Kallenberg, L. (1994). On finding optimal policies for Markov decision chains. *Mathematics of Operations Research*, *19*, 434–448.

Jordan, M. I. (2004). Graphical models. *Statistical Science (Special Issue on Bayesian Statistics)*, *19*, 140–155.

Kallenberg, L. (1983). *Linear Programming and Finite Markovian Control Problems.* Math. Centrum, Amsterdam.

Kearns, M., Littman, M. L., & Singh, S. (2001). Graphical models for game theory. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence (UAI01)*, pp. 253–260.

Kfir-Dahav, N. E., Monderer, D., & Tennenholtz, M. (2000). Mechanism design for resource bounded agents. In *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS-00)*, pp. 309–316.

Koenig, S., & Simmons, R. G. (1994). Risk-sensitive planning with probabilistic decision graphs. In Doyle, J., Sandewall, E., & Torasso, P. (Eds.), *KR'94: Principles of Knowledge Representation and Reasoning*, pp. 363–373. Morgan Kaufmann, San Francisco, California.

Koller, D., & Milch, B. (2003). Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior*, *45*(1), 181–221.

Koller, D., & Parr, R. (1999). Computing factored value functions for policies in structured MDPs. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence IJCAI-99*, pp. 1332–1339.

Kveton, B., & Hauskrecht, M. (2004). Heuristic refinements of approximate linear programming for factored continuous-state Markov decision processes. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling*, pp. 306–314.

Kveton, B., & Hauskrecht, M. (2005). An MCMC approach to solving hybrid factored MDPs. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pp. 1346–1351.

Larson, K. (2004). *Mechanism Design for Computationally Limited Agents.* Ph.D. thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA.

Larson, K., & Sandholm, T. (2005). Mechanism design and deliberative agents. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS '05)*, pp. 650–656, New York, NY, USA. ACM Press.

Lazar, A. (1983). Optimal flow control of a class of queuing networks in equilibrium. *IEEE Transactions on Automatic Control*, *28*(11), 1001–1007.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pp. 157–163.

Littman, M. L., Dean, T. L., & Kaelbling, L. P. (1995). On the complexity of solving Markov decision problems. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI–95)*, pp. 394–402, Montreal.

MacKie-Mason, J. K., & Varian, H. (1994). Generalized Vickrey auctions. Tech. rep., University of Michigan.

Marcus, S., Fernandez-Gaucherand, E., Hernandez-Hernandez, D., Colaruppi, S., & Fard, P. (1997). Risk-sensitive Markov decision processes. In et. al, C. B. (Ed.), *Systems and Control in the Twenty-First Century*, pp. 263–279. Birkhauser.

Mas-Colell, A., Whinston, M. D., & Green, J. R. (1995). *Microeconomic Theory.* Oxford University Press, New York.

McAfee, R. P., & McMillan, J. (1996). Analyzing the airwaves auction. *Journal of Economic Perspectives*, *10*(1), 159–75.

McMillan, J. (1994). Selling spectrum rights. *Journal of Economic Perspectives*, *8*(3), 145–62.

Meuleau, N., Hauskrecht, M., Kim, K.-E., Peshkin, L., Kaelbling, L., Dean, T., & Boutilier, C. (1998). Solving very large weakly coupled Markov decision processes. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pp. 165–172.

Myerson, R. B. (1979). Incentive compatibility and the bargaining problem. *Econometrica*, *47*, 61–74.

Owen, G. (1982). *Game Theory* (Second edition). Academic Press, Orlando, Florida.

Papoulis, A. (1984). *Probability, Random Variables, and Stochastic Processes* (Second edition). McGraw-Hill, New York, NY, USA.

Parkes, D. (2001). *Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency*. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania.

Parkes, D. C., & Shneidman, J. (2004). Distributed implementations of Vickrey-Clarke-Groves mechanisms. In *Proc. 3rd Int. Joint Conf. on Autonomous Agents and Multi Agent Systems*, pp. 261–268.

Paruchuri, P., Tambe, M., Ordonez, F., & Kraus, S. (2004). Towards a formalization of teamwork with resource constraints. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 596–603.

Patrascu, R., Poupart, P., Schuurmans, D., Boutilier, C., & Guestrin, C. (2002). Greedy linear value-approximation for factored Markov decision processes. In *Eighteenth National Conference on Artificial Intelligence*, pp. 285–291. American Association for Artificial Intelligence.

Pekec, A., & Rothkopf, M. H. (2003). Combinatorial auction design. *Management Science*, *49*(11), 1485–1503.

Pennock, D. M., & Wellman, M. P. (2000). Compact securities markets for Pareto optimal reallocation of risk. In *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pp. 481–488, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Pratt, J. (1964). Risk aversion in the small and in the large. *Econometrica*, *32*(1-2), 122–136.

Procaccia, A. D., & Rosenschein, J. S. (2006). Junta distributions and the average-case complexity of manipulating elections. In *The Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, Hakodate, Japan. To appear.

Puterman, M. L. (1994). *Markov Decision Processes*. John Wiley & Sons, New York.

Ross, K., & Chen, B. (1988). Optimal scheduling of interactive and non-interactive traffic in telecommunication systems. *IEEE Transactions on Auto Control*, *33*, 261–267.

Ross, K., & Varadarajan, R. (1989). Markov decision processes with sample path constraints: the communicating case. *Operations Research*, *37*, 780–790.

Ross, K., & Varadarajan, R. (1991). Multichain Markov decision processes with a sample path constraint: A decomposition approach. *Mathematics of Operations Research*, *16*, 195–207.

Rothkopf, M. H., Pekec, A., & Harstad, R. M. (1998). Computationally manageable combinational auctions. *Management Science*, *44*(8), 1131–1147.

Russell, S. J., & Subramanian, D. (1995). Provably bounded optimal agents. *Journal of Artificial Intelligence Research*, *2*, 575–609.

Sandholm, T. (2000). eMediator : a next generation electronic commerce server. In *International Conference on Autonomous Agents*, pp. 341–348.

Sandholm, T. (2002). Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, *135*(1-2), 1–54.

Sandholm, T., & Suri, S. (2000). Improved algorithms for optimal winner determination in combinatorial auctions and generalizations. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pp. 90–97. AAAI Press / The MIT Press.

Sanghvi, S., & Parkes, D. C. (2004). Hard-to-manipulate combinatorial auctions. Tech. rep., Harvard University.

Schweitzer, P., & Seidmann, A. (1985). Generalized polynomial approximations in Markovian decision processes. *Journal of Mathematical Analysis and Applications*, *110*, 568–582.

Shapley, L. S. (1953). Stochastic games. *Proceedings of National Academy of Science, USA*, *39*, 1095–1100.

Sheffi, Y. (2004). Combinatorial auctions in the procurement of transportation services. *Interfaces*, *34*(4), 245–252.

Simon, H. (1957). A behavioural model of rational choice. In Simon, H. (Ed.), *Models of Man: Social and Rational; Mathematical Essays on Rational Human Behavior in a Social Setting*, pp. 241–260. J. Wiley, New York.

Singh, S., & Cohn, D. (1998). How to dynamically merge Markov decision processes. In Jordan, M. I., Kearns, M. J., & Solla, S. A. (Eds.), *Advances in Neural Information Processing Systems*, Vol. 10, pp. 1057–1063. The MIT Press.

Smallwood, R. D., & Sondik, E. J. (1973). The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, *21*, 1071–1088.

Sobel, M. (1985). Maximal mean/standard deviation ratio in undiscounted MDP. *Operations Research Letters*, *4*, 157–188.

Song, J., & Regan, A. (2002). Combinatorial auctions for transportation service procurement: The carrier perspective. *Transportation Research Record*, *1833*, 40–46.

St-Aubin, R., Hoey, J., & Boutilier, C. (2000). Apricodd: Approximate policy construction using decision diagrams. In *Proceedings of the 14th Neural Information Processing Systems*, pp. 1089–1095.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning I: An Introduction.* MIT Press.

van Nunen, J. (1976). *Contracting Markov Decision Processes.* Mathematisch Centrum, Amsterdam.

Vickrey, W. (1961). Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, *16*, 8–37.

Wal, J. v. d. (1981). Stochastic dynamic programming. *Mathematical Centre Tracts*, *139*.

Wellman, M. P., Walsh, W. E., Wurman, P. R., & MacKie-Mason, J. K. (2001). Auction protocols for decentralized scheduling. *Games and Economic Behavior*, *35*, 271–303.

Wolsey, L. (1998). *Integer Programming.* John Wiley & Sons.