Percentile and inverse optimization in Markov decision processes with extensions to convex programs

Zahra Ghatrani

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2021

Reading Committee:
Archis Ghate, Chair
Zelda Zabinsky
Chaoyue Zhao

Program Authorized to Offer Degree:
Industrial & Systems Engineering

University of Washington

Abstract

Percentile and inverse optimization in Markov decision processes with extensions to convex programs

Zahra Ghatrani

Chair of the Supervisory Committee:
Professor Archis Ghate
Industrial & Systems Engineering

Infinite-horizon stationary Markov decision processes (MDPs) have been studied extensively in the literature. Over the last sixty years, they have found applications in a broad range of areas such as healthcare, telecommunications, transportation, revenue management, supply chain and inventory management, scheduling, resource allocation, autonomous systems, and reinforcement learning. An MDP is described as follows. At the beginning of each time-step, a decision-maker observes the state of a stochastic system. The decision-maker then chooses an action. The system probabilistically evolves into a new state and the decision-maker earns a reward. The transition probability and the reward both depend on the current state and the action chosen therein. The decision-maker's goal is to choose actions such that the expected total discounted reward over an infinite horizon is maximized.

It is often assumed in the literature that the transition probabilities and the rewards are known to the decision-maker. In practice, however, these are estimated subject to errors. The question then arises as to how the decision-maker can incorporate its incomplete knowledge about these parameters into the decision-making process. At least three different approaches have been proposed: robust optimization, inverse optimization, and percentile optimization. This dissertation makes methodological contributions to two of these three: percentile and inverse optimization.

Percentile optimization in MDPs accounts for uncertainty in reward parameters by choosing decisions that maximize the β-percentile of the expected total discounted reward over an infinite horizon. This approach is similar to chance-constrained optimization. It turns out that, when rewards are multivariate Gaussian, the percentile optimization problem in MDPs can be reformulated as a second-order cone program (SOCP). Multi-armed bandit (MAB) problems are perhaps the most studied special case of MDPs. Unfortunately, an as-is application of the existing percentile optimization methodology is intractable for MABs. In particular, the resulting SOCP suffers from the curse-of-dimensionality because its size is exponential in the number of arms.

The idea in inverse optimization is to recover implied parameter values from observed decisions. In the context of MDPs, this approach has been applied to recover reward values that render observed decisions optimal. It is known that this results in a linear program (LP) that can be solved efficiently. A counterpart of this approach is not available for imputing transition probabilities. The challenge is that since transition probabilities appear on the left hand side as constraint coefficients in an LP formulation of an infinite-horizon MDP, the inverse problem turns out to be nonconvex bilinear. More generally, there are only a few studies in the literature that focus on applying inverse optimization framework to LPs or convex programs with unknown constraint parameters.

The research objective of this dissertation is to apply convex optimization methods to efficiently compute approximate solutions of (i) percentile optimization problems in MABs, and more generally, in weakly coupled MDPs under Gaussian rewards; (ii) inverse problems in MDPs with unknown transition probabilities; and (iii) inverse semidefinite programs (SDPs). The dissertation is organized as follows.

Percentile optimization in MAB problems: the *first* chapter focuses on MAB problems whose traditional version can be described as follows. At each time-step, a decision-maker selects one arm from a finite set, after observing the states of all arms. A reward is

earned from this arm and its state evolves stochastically. No reward is earned from other arms, and their states do not change. The goal is to determine an arm-pulling policy that maximizes the expected total discounted reward over an infinite-horizon.

The chapter considers the more challenging case where rewards are multivariate Gaussian with possible correlations across states, to account for estimation errors. This is motivated by recent work on percentile optimization in MDPs. We demonstrate that, when applied to MABs, this yields an intractable SOCP with size exponential in the number of arms. The chapter proposes a Lagrangian relaxation method to break this curse-of-dimensionality. This relaxation dualizes the restriction that exactly one arm must be played at each time-step. The optimal value of the relaxed problem provides an upper bound on the exact percentile problem. Moreover, the relaxation achieves a decomposition across arms, which exponentially reduces the computational complexity. The chapter then applies convex strong duality to formulate the problem of finding the tightest upper bound (and the corresponding best Lagrange multiplier) as a tractable SOCP. We propose three approaches to recover feasible arm-pulling decisions during run-time from an off-line optimal solution of this SOCP. Numerical experiments suggest that one of these three method appears to be more effective than the other two. This methodology is also extended to a broader class of problems called weakly coupled MDPs. There, we propose four methods to recover run-time decisions from an off-line optimal solution of an SOCP. Our numerical results suggest that three of these methods perform better than the fourth one, and that one of these three methods seems to work better for larger problems than the other two.

Inverse MDPs with unknown transition probabilities: the *second* chapter considers two variants of this problem. In the first variant, the decision-maker wonders whether there exist transition probabilities and corresponding decisions that would attain a given expected total discounted reward over an infinite-horizon (the so-called value function). An easy-to-verify necessary and sufficient condition for this existence is derived. The chapter

demonstrates that when this condition is met, the requisite transition probabilities and decisions can be imputed by solving a feasibility LP. These ideas are then extended to the case when the decision-maker wishes to render the given value function optimal. The chapter then turns to the more difficult problem of imputing transition probabilities that make given decisions optimal. LP strong duality is applied to this problem to derive a nonconvex bilinear program. Tailored versions of two heuristics that exploit the structure of this bilinear program are proposed. The first one is rooted in a so-called convex-concave procedure (CCP) for a class of problems called "difference of convex" programs. The second one is called sequential linear programming (SLP). The performance of these two methods is compared via numerical experiments against an exact global optimization method based on generalized Bender's decomposition. Computational experiments on randomly generated inverse generic MDP problems reveal that SLP outperforms the other two methods in both runtime and objective values. Further insights into SLP's performance are derived via numerical experiments on inverse inventory control, equipment replacement, and multi-armed bandit problems.

Inverse semidefinite programs with unknown constraint parameters: The *third* chapter focuses on extension of the methods discussed in the second chapter to the broader class of convex optimization problems called semi-definite programs (SDPs). An inverse optimization methodology for SDPs with unknown constraint parameters is currently not available. Similar to the second chapter, when constraint parameters are unknown, the resulting inverse SDP problem is nonconvex bilinear. This chapter focuses on six variants of this inverse problem constructed based on which parameters and decision variables are known. In each case, we will identify when the inverse problem is trivial to solve and when it is not. We will show that in one variant, the inverse SDP problem reduces to solving a group of SDPs. In two of the other variants, we provide conditions under which the inverse problem can be reduced to a tractable SDP via a variable transformation. In all other cases, we apply

tailored versions of the heuristics proposed in the second chapter to obtain an approximate solution. Another heuristic called Alternate Convex Search (ACS) is also implemented in some cases. The performance of these methods is compared via numerical experiments.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

PERCENTILE OPTIMIZATION IN MULTI-ARMED BANDIT PROBLEMS

1.1 Introduction

The term multi-armed bandit (MAB) is derived from the problem faced by a gambler/player on a slot machine with multiple levers (one-armed bandits). At each time-step, the gambler has to decide which lever (arm) to pull, in order to maximize the expected total discounted reward over an infinite horizon. MABs are used as an abstract model for sequential decision making, where a limited resource needs to be allocated between different reward generating processes.

In this chapter, we consider the classical MAB problem, which is described as follows [43, 58, 82, 84]. There is a finite set of arms $\mathcal{N} = \{1, \cdots, N\}$. At each time-step $t \in \{0, 1, 2, ...\}$, each arm $n \in \mathcal{N}$ is in some state $s_n \in \mathcal{S}_n$, where $\mathcal{S}_n$ represents the finite state space of arm n. The player is restricted to pulling one arm at a time. If arm n is pulled, s_n evolves to s'_n with probability $p_n(s'_n|s_n)$, yielding a corresponding reward $r_n(s_n)$. No reward is earned from arms that are not pulled and their states do not change. The player's goal is to choose an arm to play in each state $s = (s_1, \cdots, s_N)$, so as to maximize the expected total discounted reward over an infinite horizon. The discount factor is $0 < \alpha < 1$. MABs arise in various applications including recommender systems, dynamic pricing, advertising, clinical trials, and job scheduling [17, 50].

In principle, MAB problems can be formulated as a Markov decision Process (MDP) and solved via Bellman's equations of dynamic programming [70]. However, due to the curse-of-dimensionality, this is computationally intractable — the state space of the MAB is of exponential size in N. In a celebrated paper, Gittins [43] proved that instead of solving the

resulting N-dimensional dynamic program, an optimal solution for the MAB problem can be characterized by an index type of policy. That is, at each time step, and for each arm, we need to compute a dynamic index called the Gittins index. It is optimal to play the arm with the highest Gittins index at each time-step. The index for one arm is independent of others. This decomposes the problem so that the computational effort reduces from exponential in N to linear. After the original paper by Gittins, other researchers proposed various approaches to prove the optimality of this index policy [50, 78, 82, 84].

In the above MAB problem, the reward values are assumed to be known to the decision-maker. In practice, however, rewards are estimated from data subject to errors. The decision-maker's incomplete knowledge or uncertainty about the rewards can negatively impact the performance of an arm-pulling policy [59]. To address this issue in generic MDPs, Delage and Mannor [31] proposed a framework that they termed *percentile optimization*. In their approach, roughly speaking, the decision-maker tries to find a policy whose expected total discounted reward over an infinite horizon is sufficiently large with at least a given probability. This is akin to the classic idea of chance-constrained programming [29, 69], and was proposed as a less conservative alternative to robust optimization in MDPs [54, 63]. Delage and Mannor [31] showed that the percentile optimization problem for generic MDPs with multivariate Gaussian uncertainty in rewards can be reformulated as a tractable second-order cone program (SOCP) [5]. The research objective of this study is to build upon this idea in the specific context of MAB problems.

The decision-maker's philosophy in this chapter is to find a solution whose reward is sufficiently high with a high probability. That is, roughly speaking, the decision-maker solves a chance-constrained problem of the form max y, subject to $\mathbb{P}(\text{reward} \geq y) \geq \beta$, for some given confidence level $\beta \in (0, 1)$. Here, the probability is computed with respect to the uncertainty in reward. Such a percentile approach was pioneered for MDPs by Delage and Mannor [31]. They showed that the percentile optimization problem for MDPs with multivariate Gaussian uncertainty in rewards can be reformulated as a second-order cone program (SOCP) [5]. We build upon this idea in the context of MAB problems.

Unfortunately, an as-is application of the Delage and Mannor framework to MAB problems is computationally intractable because the number of variables and constraints in the resulting SOCP is exponential in the number of arms N. In this chapter, we demonstrate how to break this curse-of-dimensionality via an approximate solution method rooted in Lagrangian relaxation. Lagrangian relaxation was applied to MAB problems (with known rewards) by Hawkins [50] in his doctoral dissertation. He showed that the curse-of-dimensionality can be broken by relaxing (or dualizing) the constraint that only one arm can be pulled at a time. This again decomposes the problems into single-arm subproblems. In fact, Hawkins showed that solutions recovered from this approach coincide with the Gittins index approach. Caro and Gupta [22] applied Lagrangian relaxation to MAB problems with uncertain transition probabilities within a robust optimization framework. We build upon this existing literature to develop our Lagrangian relaxation methodology for percentile optimization in MAB problems. We show that the relaxed problem decomposes across arms, thus yielding a new SOCP whose size grows only linearly in N. We also propose two lookahead methods and a normalization method to recover feasible arm-pulling decisions for the original MAB problem, from the solution of this relaxed SOCP. Numerical results show that on average, the normalization method exhibits the best performance in terms of percentile values, and is the most robust to the magnitude of expected value and covariance of rewards. We also extend the Lagrangian relaxation methodology to a broader class of percentile problems in weakly coupled MDPs. We derive an SOCP formulation of the relaxed problem. Finally, we propose three deterministic methods and a randomized method to recover feasible decisions for the weakly coupled MDP from a solution of this SOCP. Our numerical experiments show that the randomized method is dominated by the deterministic ones. Furthermore, while the deterministic methods perform similarly on smaller weakly coupled MDPs, the third deterministic approach outperforms the first two on larger problems.

The remainder of this chapter is organized as follows. In Section 1.2, we describe a problem statement for the nominal MAB problem (with known rewards). We describe a percentile formulation in Section 1.3, and formalize our Lagrangian relaxation approach

in Section 1.4. In Section 1.5, we propose three heuristics to retrieve feasible decisions. Numerical results for percentile MAB problems are presented in Section 1.6. In Section 1.7, we extend our methodology to percentile weakly coupled MDPs and also provide numerical results. Conclusions are outlined in Section 1.8.

1.2 Nominal formulation

The action space of each individual arm $n \in \mathcal{N}$ is $\mathcal{A}_n = \{a_n \in \{0,1\}\}$. Here, actions $a_n = 1$ and $a_n = 0$ represent pulling and not pulling arm n, respectively. The state space of the MAB problem is defined as $\mathcal{S} = \bigtimes_{n=1}^{N} \mathcal{S}_n$. The action space in every state $s \in \mathcal{S}$ is $\mathcal{A} = \bigtimes_{n=1}^{N} \mathcal{A}_n = \{0,1\}^N$. The reward earned by playing any action $a = (a_1, \ldots, a_n) \in \mathcal{A}$ in state $s = (s_1, \cdots, s_N) \in \mathcal{S}$ is

$$r(s,a) = \sum_{n \in \mathcal{N}} r_n(s_n) a_n. \tag{1.1}$$

Since actions are binary, this expression includes the reward earned from any arm that is pulled and does not include rewards for arms that are not pulled. The transition probability induced by any action $a = (a_1, \ldots, a_n) \in \mathcal{A}$ in state $s = (s_1, \cdots, s_N) \in \mathcal{S}$ is

$$p(s'|s,a) = \prod_{n=1}^{N} \Big(a_n p_n(s'_n|s_n) + (1-a_n)\gamma_n(s'_n|s_n) \Big). \tag{1.2}$$

Here, we have utilized the notation $\gamma_n(s'_n|s_n) = 1$ if $s'_n = s_n$, and 0 otherwise. The expression in (1.2) succinctly captures various events that can occur at a time-step. If arm n is pulled ($a_n = 1$), only the term $p_n(s'_n|s_n)$ is "in effect". This determines the transition probability associated with the n^{th} arm. If arm n is not pulled ($a_n = 0$), only the term $\gamma_n(s'_n|s_n)$ is in effect. Moreover, this term is 1 only if the state remains unchanged ($s'_n = s_n$); otherwise it is 0. The combined transition probability is obtained by multiplying probabilities for individual arms. As discussed earlier, the player is restricted to pull exactly one arm at every time-step.

Therefore, the set of feasible actions is given by

$$\bar{\mathcal{A}} = \big\{ a \in \mathcal{A} \big| \sum_{n \in \mathcal{N}} a_n = 1 \big\}. \tag{1.3}$$

Observe that $|\bar{\mathcal{A}}| = N$, and each element in $\bar{\mathcal{A}}$ corresponds to choosing a particular arm.

A (deterministic Markovian) policy π is a decision rule that assigns actions $\pi(s) \in \bar{\mathcal{A}}$ to states $s \in \mathcal{S}$. This policy can be stored in a matrix of size $|\mathcal{S}| \times N$, wherein, the entry in the s^{th} row and n^{th} column is 1 if arm n is played in state s; the entry is 0 otherwise. Let Π denote the finite set of all such policies. The expected total discounted value of a policy $\pi \in \Pi$ upon starting in state $s \in \mathcal{S}$ is defined as

$$J^{\pi}(s) = \mathbb{E}\left(\sum_{t=0}^{\infty} \alpha^t r(\mathbf{s}^t, \pi(\mathbf{s}^t)) \Big| \mathbf{s}^0 = s \right). \tag{1.4}$$

Here, $\mathbf{s}^t$ is the (stochastic) state at time-step t, induced by policy π and initial state s. We use $J^{\pi} \in \mathbb{R}^{|\mathcal{S}|}$ to denote the column vector whose s^{th} entry is $J^{\pi}(s)$. Suppose the initial state s_n of arm n is sampled, independently of all other arms, according to a probability mass function (pmf) q_n. Thus, the initial state pmf q (column vector of size $|\mathcal{S}|$) is given by $q(s) = \prod_{n=1}^{N} q_n(s_n)$. This is a natural assumption, given the independence of distinct arms. We define the value function J_q^{π} as the expectation of J^{π} with respect to q. That is, $J_q^{\pi} = \sum_{s \in \mathcal{S}} q(s) J^{\pi}(s)$. Let $r^{\pi} \in \mathbb{R}^{|\mathcal{S}|}$ and $\mathbb{P}_{\pi} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ represent the immediate reward vector and transition probability matrix associated with policy π, respectively. That is, $r^{\pi}(s) = r(s, \pi(s))$ and the entry in the s^{th} row and s'^{th} column of $\mathbb{P}_{\pi}$ equals $p\big(s' | s, \pi(s)\big)$. Using Chapmann-Kolmogorov equations, we obtain,

$$J_q^{\pi} = \sum_{s \in \mathcal{S}} q(s) J^{\pi}(s) = q^T \underbrace{\left[\sum_{t=0}^{\infty} (\alpha \mathbb{P}_{\pi})^t \right]}_{J^{\pi}} r^{\pi}. \tag{1.5}$$

The decision-maker's goal is to find a policy that solves

$$J_q = \max_{\pi \in \Pi} \; J_q^\pi = \max_{\pi \in \Pi} \; q^T \left[\sum_{t=0}^{\infty} (\alpha \mathbb{P}_\pi)^t \right] r^\pi. \tag{1.6}$$

As stated in Section 1.1, optimal solutions to this problem can be characterized via Bellman's equations of dynamic programming. However, these equations are intractable to solve because the cardinality of $\mathcal{S}$ is exponential in N. Lagrangian relaxation and Gittins index calculation are two (equivalent) approaches that break this curse-of-dimensionality.

In the next section, we will introduce a percentile optimization variant of the nominal MAB problem described here; a Lagrangian relaxation methodology will be devised in Section 1.4 to overcome the curse-of-dimensionality.

1.3 Percentile formulation

Delage and Mannor [31] showed that optimal actions in a percentile MDP might be randomized. We accordingly expand the action space of the percentile MAB problem to pmfs on $\bar{\mathcal{A}}$. The action space is thus now denoted

$$\Delta = \left\{ \delta \in \mathbb{R}_+^{|\bar{\mathcal{A}}|} | \sum_{a \in \bar{\mathcal{A}}} \delta(a) = 1 \right\}. \tag{1.7}$$

Here, $\delta(a)$ is the probability that action $a \in \bar{\mathcal{A}}$ is selected. Note that since the set $\bar{\mathcal{A}}$ only includes action vectors that allow one arm to be pulled at a time, the player is still restricted to doing so even under randomized actions. The expected immediate reward of the randomized action $\delta \in \Delta$ in state $s = (s_1, \cdots, s_N) \in \mathcal{S}$ is defined as

$$r(s, \delta) = \sum_{a \in \bar{\mathcal{A}}} \delta(a) r(s, a). \tag{1.8}$$

The transition probabilities induced by the randomized action $\delta \in \Delta$ in state $s = (s_1, \cdots, s_N) \in \mathcal{S}$ are given by

$$p(s'|s, \delta) = \sum_{a \in \bar{\mathcal{A}}} \delta(a) p(s'|s, a). \tag{1.9}$$

A randomized policy ρ is a mapping from $\mathcal{S}$ to Δ such that $\rho(s) \in \Delta$ specifies the probabilities assigned to each action in $\bar{\mathcal{A}}$. In particular, $\rho(s, a)$ is the probability of choosing action $a \in \bar{\mathcal{A}}$ in state $s \in \mathcal{S}$. The (uncountable) set of all randomized policies is denoted by $\mathcal{P}$. Let $\mathbb{P}_\rho \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ denote the transition probability matrix associated with the randomized policy ρ. The element in row s and column s' is calculated, based on (1.9), as $\mathbb{P}_\rho(s'|s) = \sum_{a \in \bar{\mathcal{A}}} \rho(s, a) p(s'|s, a)$. Similarly, the vector $r^\rho \in \mathbb{R}^{|\mathcal{S}|}$ contains the immediate rewards earned under policy ρ in all states $s \in \mathcal{S}$. Based on (1.8), its components can be written as $r^\rho(s) = \sum_{a \in \bar{\mathcal{A}}} \rho(s, a) r(s, a)$. Similar to the deterministic case, the expected total discounted reward of the randomized policy $\rho \in \mathcal{P}$ can be written as

$$J_q^\rho = q^T \left[\sum_{t=0}^{\infty} (\alpha \mathbb{P}_\rho)^t \right] r^\rho. \tag{1.10}$$

Since actions in $\bar{\mathcal{A}}$ uniquely identify a single arm that is played, an alternative view of a randomized policy is that of an $\mathcal{S} \times N$ matrix; this alternative way of writing turns out to be more convenient for developing the percentile methodology. The entry in row s and column n denotes the probability of playing arm n in state s. We denote this entry by $\rho_n(s)$. Consequently, we use ρ_n to denote the corresponding $|\mathcal{S}|$-dimensional column vector. Using this notation, and the equivalence of actions in $\bar{\mathcal{A}}$ with arms in $\mathcal{N}$, the above expression for $r^\rho(s)$ can be rewritten as $r^\rho(s) = \sum_{n \in \mathcal{N}} \rho_n(s) r_n(s)$. Here, with some abuse of notation, we have used $r_n(s)$ to denote the reward that would be earned if arm n were played in state s, although we know that this reward only depends on s_n. These observations help us write the vector $r^\rho \in \mathbb{R}^{|\mathcal{S}|}$ of rewards earned by policy ρ, in matrix form, as $r^\rho = \sum_{n \in \mathcal{N}} \text{diag}(\rho_n) r_n$. Again, here, with some abuse of notation, $r_n \in \mathbb{R}^{|\mathcal{S}|}$(we later use $\tilde{r}_n \in \mathbb{R}^{|\mathcal{S}_n|}$ for individual

arms) contains the rewards obtained by playing arm n in various states $s \in \mathcal{S}$. Furthermore $\text{diag}(\rho_n)$ is the $|\mathcal{S}| \times |\mathcal{S}|$ diagonal matrix whose diagonal entries are the $|\mathcal{S}|$ elements of the column vector ρ_n. Substituting this back into (1.10), we obtain,

$$J_q^\rho = q^T \left[\sum_{t=0}^{\infty} (\alpha \mathbb{P}_\rho)^t\right] \sum_{n \in \mathcal{N}} \text{diag}(\rho_n) r_n = \sum_{n \in \mathcal{N}} q^T \underbrace{\left[\sum_{t=0}^{\infty} (\alpha \mathbb{P}_\rho)^t\right] \text{diag}(\rho_n) r_n}_{J_n^\rho} = \sum_{n \in \mathcal{N}} J_{q,n}^\rho. \quad (1.11)$$

Here, $J_n^\rho \in \mathbb{R}^{|\mathcal{S}|}$ is a column vector whose component $J_n^\rho(s)$ is the expected total discounted reward earned from all plays only of arm n, under policy ρ, when the initial state is $s \in \mathcal{S}$. Similarly, $J_{q,n}^\rho$ is the expected total discounted reward earned from all plays only of arm n, under policy ρ, when the initial state pmf is q.

We apply one relevant feature of the generic linear programming (LP) approach to MDPs from [70, Section 6.9]. Consider the set of feasible solutions $x \in \mathbb{R}^{|\mathcal{S}| \times |\bar{\mathcal{A}}|}$ to the system

$$\sum_{a \in \bar{\mathcal{A}}} x(s,a) - \alpha \sum_{s' \in \mathcal{S}} \sum_{a \in \bar{\mathcal{A}}} p(s|s',a) x(s',a) = q(s), \quad \forall s \in \mathcal{S} \quad (1.12a)$$

$$x(s,a) \geq 0, \quad \forall s \in \mathcal{S}, a \in \bar{\mathcal{A}}. \quad (1.12b)$$

We denote this set by $\mathcal{X}$. Theorem 6.9.1 and Corollary 6.9.2 in [70] imply that there is a one-to-one correspondence between randomized policies ρ and feasible solutions to (1.12). Specifically, formulas 6.9.3 and 6.9.4 in [70] define this one-to-one correspondence. In addition, formulas 6.9.8 and 6.9.9 in [70] imply that (i) for every feasible x, there is a randomized policy ρ_x such that $J_{q,n}^{\rho_x} = \sum_{s \in \mathcal{S}} \sum_{a \in \bar{\mathcal{A}}} r_n(s_n) a_n x(s,a)$; and (ii) for every randomized policy ρ, there is a feasible x_ρ such that $J_{q,n}^\rho = \sum_{s \in \mathcal{S}} \sum_{a \in \bar{\mathcal{A}}} r_n(s_n) a_n x_\rho(s,a)$.

Following Delage and Mannor [31], we now assume that the rewards are multivariate Gaussian. We denote this as $\tilde{r}_n \sim \text{Normal}(\mu_n, \Theta_n)$. Here, $\mu_n \in \mathbb{R}^{|\mathcal{S}_n|}$ is the mean vector and $\Theta_n \in \mathbb{R}^{|\mathcal{S}_n| \times |\mathcal{S}_n|}$ is the covariance matrix of rewards obtained by playing arm n. The rewards of an arm across different states may be correlated, while the rewards of different

arms are assumed to be independent. Under this uncertainty model, quantities $J^{\rho}_{q,n}$ are now random variables. Let $\beta_n \in [0.5, 1), \forall n \in \mathcal{N}$, be given fractions. We formulate the percentile optimization problem as

$$y^* = \max_{y \in \mathbb{R}^N, \rho \in \mathcal{P}} \sum_{n \in \mathcal{N}} y_n \tag{1.13a}$$

$$\mathbb{P}_{\tilde{r}_n}\Big(J^{\rho}_{q,n} \geq y_n\Big) \geq \beta_n, \qquad \forall n \in \mathcal{N}. \tag{1.13b}$$

The subscript $\tilde{r}_n$ emphasizes that the probability is computed with respect to the uncertainty in rewards for arm n.

The aforementioned one-to-one correspondence between randomized policies and feasible solutions to the system (1.12) implies that, when rewards are random, the event "$J^{\rho}_{q,n} \geq y_n$, for some $\rho \in \mathcal{P}$" is equivalent to the event "$\sum_{s \in \mathcal{S}} \sum_{a \in \bar{\mathcal{A}}} \tilde{r}_n(s_n) a_n x(s,a) \geq y_n$, for some $x \in \mathcal{X}$." Consequently, problem (1.13) is equivalent to the problem

$$y^* = \max_{y \in \mathbb{R}^N, x \in \mathbb{R}^{|\mathcal{S}| \times |\bar{\mathcal{A}}|}} \sum_{n \in \mathcal{N}} y_n \tag{1.14a}$$

$$\mathbb{P}_{\tilde{r}_n}\left(\sum_{s \in \mathcal{S}} \sum_{a \in \bar{\mathcal{A}}} \tilde{r}_n(s_n) a_n x(s,a) \geq y_n\right) \geq \beta_n, \qquad \forall n \in \mathcal{N} \tag{1.14b}$$

(1.12a) – (1.12b).

The next lemma allows us to further simplify this formulation.

Lemma 1.3.1. *The distribution of the expected total discounted reward earned from plays of the n^{th} arm is given by*

$$\sum_{s \in \mathcal{S}} \sum_{a \in \bar{\mathcal{A}}} \tilde{r}_n(s_n) a_n x(s,a) \sim \mathit{Normal}\Big(\sum_{s \in \mathcal{S}} \sum_{a \in \bar{\mathcal{A}}} \mu_n(s_n) a_n x(s,a), \sum_{s \in \mathcal{S}} \sum_{s' \in \mathcal{S}} \sum_{a \in \bar{\mathcal{A}}} \sum_{a' \in \bar{\mathcal{A}}} x(s,a) x(s',a') a_n a'_n \Theta_n(s_n, s'_n)\Big). \tag{1.15}$$

Proof. Since reward vector $\tilde{r}_n$ is multivariate Gaussian, we know that any linear combination of its components is univariate Normal. Recalling that $\mathcal{S}$ is the Cartesian product $\bigtimes_{n=1}^{N} \mathcal{S}_n$,

we observe that

$$\sum_{s\in\mathcal{S}}\sum_{a\in\bar{\mathcal{A}}}\tilde{r}_n(s_n)a_nx(s,a) = \sum_{s_1\in\mathcal{S}_1}\cdots\sum_{s_n\in\mathcal{S}_n}\cdots\sum_{s_N\in\mathcal{S}_N}\sum_{a\in\bar{\mathcal{A}}}\tilde{r}_n(s_n)a_nx(s,a)$$
$$= \sum_{s_n\in\mathcal{S}_n}\sum_{s_1\in\mathcal{S}_1}\cdots\sum_{s_{n-1}\in\mathcal{S}_{n-1}}\sum_{s_{n+1}\in\mathcal{S}_{n+1}}\cdots\sum_{s_N\in\mathcal{S}_N}\sum_{a\in\bar{\mathcal{A}}}\tilde{r}_n(s_n)a_nx(s,a)$$
$$= \sum_{s_n\in\mathcal{S}_n}\tilde{r}_n(s_n)\underbrace{\left(\sum_{s_1\in\mathcal{S}_1}\cdots\sum_{s_{n-1}\in\mathcal{S}_{n-1}}\sum_{s_{n+1}\in\mathcal{S}_{n+1}}\cdots\sum_{s_N\in\mathcal{S}_N}\sum_{a\in\bar{\mathcal{A}}}a_nx(s,a)\right)}_{c_n(s_n)}.$$

Here, the second equality follows by reordering sums and the third equality follows by factoring out $\tilde{r}_n(s_n)$ out of the other sums. Now, since the expression inside the big parentheses, denoted $c_n(s_n)$, only depends on s_n, the whole expression is indeed a linear combination of $\tilde{r}_n(s_n)$. It is, therefore, univariate Normal. By linearity of expectation, its mean is $\sum_{s\in\mathcal{S}}\sum_{a\in\bar{\mathcal{A}}}\mu_n(s_n)a_nx(s,a)$. Further, its variance is given by

$$\sum_{s_n\in\mathcal{S}_n}\sum_{s'_n\in\mathcal{S}_n}c_n(s_n)c_n(s'_n)\Theta_n(s_n,s'_n)$$
$$= \sum_{s_n\in\mathcal{S}_n}\sum_{s'_n\in\mathcal{S}_n}\left(\sum_{s_1\in\mathcal{S}_1}\cdots\sum_{s_{n-1}\in\mathcal{S}_{n-1}}\sum_{s_{n+1}\in\mathcal{S}_{n+1}}\cdots\sum_{s_N\in\mathcal{S}_N}\sum_{a\in\bar{\mathcal{A}}}a_nx(s,a)\right)$$
$$\left(\sum_{s'_1\in\mathcal{S}_1}\cdots\sum_{s'_{n-1}\in\mathcal{S}_{n-1}}\sum_{s'_{n+1}\in\mathcal{S}_{n+1}}\cdots\sum_{s'_N\in\mathcal{S}_N}\sum_{a'\in\bar{\mathcal{A}}}a'_nx(s',a')\right)\Theta_n(s_n,s'_n)$$
$$= \sum_{s\in\mathcal{S}}\sum_{s'\in\mathcal{S}}\sum_{a\in\bar{\mathcal{A}}}\sum_{a'\in\bar{\mathcal{A}}}x(s,a)a_n\Theta_n(s_n,s'_n)x(s',a')a'_n.$$

Here, the last equality follows by reordering sums and then utilizing the earlier observation that S is the Cartesian product $\bigtimes_{n=1}^{N}\mathcal{S}_n$. This completes the proof. □

Note that $\max\{y\in\mathbb{R}|\mathbb{P}(\text{Normal}(\mu,\sigma^2)\geq y)\geq\beta\} = \mu-\Phi^{-1}(\beta)\sigma$, where Φ is the cumulative distribution function of a standard Normal random variable. Using this and Lemma 1.3.1, problem (1.14) can be reformulated as the SOCP

$$y^* = \max_{x \in \mathbb{R}^{|\mathcal{S}|\times|\bar{\mathcal{A}}|}} \sum_{n\in\mathcal{N}} \left\{ \sum_{s\in\mathcal{S}} \sum_{a\in\bar{\mathcal{A}}} \mu_n(s_n) a_n x(s,a) - \Phi^{-1}(\beta_n) \sqrt{\sum_{s\in\mathcal{S}} \sum_{s'\in\mathcal{S}} \sum_{a\in\bar{\mathcal{A}}} \sum_{a'\in\bar{\mathcal{A}}} x(s,a)x(s',a') a_n a'_n \Theta_n(s_n, s'_n)} \right\} \tag{1.16a}$$

(1.12a) – (1.12b).

This is the analog, in our specific context, of the generic SOCP derived by Delage and Mannor [31]. SOCPs are considered computationally tractable because polynomial-time interior point algorithms are available for their solution [5, 57]. However, solving (1.16) is computationally expensive because it includes $\prod_{n\in\mathcal{N}} |\mathcal{S}_n| \times N + \prod_{n\in\mathcal{N}} |\mathcal{S}_n|$ constraints and $N \times \prod_{n\in\mathcal{N}} |\mathcal{S}_n|$ variables. Thus, the numbers of constraints and variables grow exponentially with the number of arms. In the next section, we propose an approximation method rooted in Lagrangian relaxation to tackle this difficulty.

1.4 Lagrangian relaxation

We employ a Lagrangian approach to relax constraints in (1.3) that couple different arms. That is, for any real number $\lambda \in \mathbb{R}$ (not necessarily nonnegative), define the Lagrangian problem by relaxing the constraint $\sum_{n\in\mathcal{N}} a_n = 1$ and adding the corresponding term $\lambda(1 - \sum_{n\in\mathcal{N}} a_n)$ to the reward $r(s,a)$ from (1.1), as described in [1]. In the Lagrangian relaxation, for any $\lambda \in \mathbb{R}$, the reward earned upon choosing action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$ is given by $\lambda + \sum_{n=1}^{N} \underbrace{\big(r_n(s_n) - \lambda\big)a_n}_{r_n^\lambda(s_n,a_n)}$. We write this compactly as $\bar{r}^\lambda(s,a) = \lambda + r^\lambda(s,a)$, where $r^\lambda(s,a)$ is defined as $\sum_{n=1}^{N} r_n^\lambda(s_n,a_n)$.

The action space in the nominal Lagrangian relaxed problem is $\mathcal{A}$. For the percentile formulation, we need to expand this to include randomized actions. Thus, the randomized action space is now given by $\Delta' = \big\{\delta' \in \mathbb{R}_+^{2^N} \,|\, \sum_{a\in\mathcal{A}} \delta'(a) = 1\big\}$. Here, 2^N is the cardinality of

the action space $\mathcal{A}$. The expected reward earned by the randomized action $\delta' \in \Delta'$ is

$$\bar{r}^{\lambda}(s,\delta') = \sum_{a\in\mathcal{A}} \delta'(a)\bar{r}^{\lambda}(s,a) = \lambda + \underbrace{\sum_{a\in\mathcal{A}} \delta'(a) r^{\lambda}(s,a)}_{r^{\lambda}(s,\delta')}.$$

The transition probabilities under the randomized action $\delta' \in \Delta'$ are given by

$$p(s'|s,\delta') = \sum_{a\in\mathcal{A}} \delta'(a) p(s'|s,a), \text{ for } s,s' \in \mathcal{S}, \tag{1.17}$$

where $p(s'|s,a)$ are defined in (1.2).

Randomized policies ν assign actions $\delta' \in \Delta'$ to states $s \in \mathcal{S}$. The uncountable set of all such policies is denoted by $\mathcal{Q}$. The value of implementing a randomized policy $\nu \in \mathcal{Q}$ upon starting the relaxed problem in state $s \in \mathcal{S}$ is defined as

$$J^{\nu,\lambda}(s) = \mathbb{E}\left(\sum_{t=0}^{\infty} \alpha^t(\lambda + r^{\lambda}(\mathbf{s}^t, \nu(\mathbf{s}^t)))|\mathbf{s}^0 = s\right) = \frac{\lambda}{1-\alpha} + \mathbb{E}\left(\sum_{t=0}^{\infty} \alpha^t r^{\lambda}(\mathbf{s}^t, \nu(\mathbf{s}^t))\Big|\mathbf{s}^0 = s\right).$$

Let $\mathbb{P}_{\nu} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ denote the transition probability matrix under randomized policy ν. The entry in row s and column s' of this matrix equals $p(s'|s,\nu(s))$. Moreover, column vector $r^{\nu,\lambda} \in \mathbb{R}^{|\mathcal{S}|}$ contains the values of $r^{\lambda}\big(s,\nu(s)\big)$, for all states $s \in \mathcal{S}$. Similar to the previous section, this further yields

$$J_q^{\nu,\lambda} = \sum_{s\in\mathcal{S}} q(s) J^{\nu,\lambda}(s) = \frac{\lambda}{1-\alpha} + q^T \left[\sum_{t=0}^{\infty} (\alpha \mathbb{P}_{\nu})^t\right] r^{\nu,\lambda}. \tag{1.18}$$

Let ν_a denote the column of the $|\mathcal{S}| \times 2^N$ policy matrix ν, associated with action $a \in \mathcal{A}$. Similarly, $r^{a,\lambda} \in \mathbb{R}^{|\mathcal{S}|}$ denotes the column vector containing values of $r^{\lambda}(s,a)$, for all $s \in \mathcal{S}$. The vector $r^{\nu,\lambda}$ can then be written in matrix form as

$$r^{\nu,\lambda} = \sum_{a\in\mathcal{A}} \text{diag}(\nu_a) r^{a,\lambda}. \tag{1.19}$$

By writing $r^{a,\lambda} = \sum_{n\in\mathcal{N}} a_n(r_n - \lambda\vec{1})$, we express (1.19) as

$$\begin{aligned} r^{\nu,\lambda} &= \sum_{a\in\mathcal{A}} \mathrm{diag}(\nu_a)\left[\sum_{n\in\mathcal{N}} a_n(r_n - \lambda\vec{1})\right] = \sum_{n\in\mathcal{N}}\sum_{a\in\mathcal{A}} a_n \mathrm{diag}(\nu_a)(r_n - \lambda\vec{1}) \\ &= \sum_{n\in\mathcal{N}} \sum_{a\in\mathcal{A}|a_n=1} \mathrm{diag}(\nu_a)(r_n - \lambda\vec{1}). \end{aligned} \tag{1.20}$$

Substituting this into (1.18), the expected total discounted reward $J_q^{\nu,\lambda}$ can be split into a sum of $\frac{\lambda}{1-\alpha}$ and the expected total discounted rewards obtained from each arm $n \in \mathcal{N}$ as

$$\begin{aligned} J_q^{\nu,\lambda} &= \frac{\lambda}{1-\alpha} + q^T\left[\sum_{t=0}^{\infty}(\alpha\mathbb{P}_\nu)^t\right]\left[\sum_{n\in\mathcal{N}}\sum_{a\in\mathcal{A}|a_n=1} \mathrm{diag}(\nu_a)(r_n - \lambda\vec{1})\right] \\ &= \frac{\lambda}{1-\alpha} + \sum_{n\in\mathcal{N}} \underbrace{q^T\left[\sum_{t=0}^{\infty}(\alpha\mathbb{P}_\nu)^t\right]\sum_{a\in\mathcal{A}|a_n=1} \mathrm{diag}(\nu_a)(r_n - \lambda\vec{1})}_{J_n^{\nu,\lambda}} = \frac{\lambda}{1-\alpha} + \sum_{n\in\mathcal{N}} J_{q,n}^{\nu,\lambda}. \end{aligned} \tag{1.21}$$

Here, $J_n^{\nu,\lambda} \in \mathbb{R}^{|\mathcal{S}|}$ is a column vector whose entry $J_n^{\nu,\lambda}(s)$ equals the expected total discounted reward earned (excluding $\lambda/(1-\alpha)$) only by plays of arm n under policy $\nu \in \mathcal{Q}$ starting in state s, when the Lagrange multiplier is fixed at λ. Moreover, $J_{q,n}^{\nu,\lambda}$ is the expectation of these rewards with respect to the initial state pmf q. We write the Lagrangian relaxation of the percentile optimization problem (1.13) as

$$y^*(\lambda) = \frac{\lambda}{1-\alpha} + \max_{y(\lambda)\in\mathbb{R}^N, \nu\in\mathcal{Q}} \sum_{n\in\mathcal{N}} y_n(\lambda) \tag{1.22a}$$

$$\mathbb{P}_{\tilde{r}_n}\Big(J_{q,n}^{\nu,\lambda} \geq y_n(\lambda)\Big) \geq \beta_n, \qquad \forall n \in \mathcal{N}. \tag{1.22b}$$

We next show that this Lagrangian relaxation provides an upper bound on (1.13). Although this conclusion is perhaps not surprising, we include a complete proof because it does rely on problem-specific details that are typically not relevant in generic proofs of Lagrangian bounds.

Proposition 1.4.1. *For any $\lambda \in \mathbb{R}$, we have, $y^*(\lambda) \geq y^*$.*

Proof. We first show that the set of policies $\mathcal{Q}$ for the relaxed problem includes the set of policies $\mathcal{P}$ for the percentile MAB problem, lifted into the higher dimensional space $\mathbb{R}^{2^N}$. Suppose $\rho \in \mathcal{P}$. Consider any state $s \in \mathcal{S}$. Then, by definition, $\sum_{a \in \bar{\mathcal{A}}} \rho(s,a) = 1$. By (1.3), we have, $\bar{\mathcal{A}} \in \mathcal{A}$. Hence, $\rho(s)$ can be lifted into $\mathbb{R}^{2^N}$ by assigning $\rho(s,a) = 0$ for actions $a \in \mathcal{A} \setminus \bar{\mathcal{A}}$ while the expected rewards and transition probabilities remain unchanged. The lifted randomized action is denoted by $\hat{\rho}(s) \in \Delta'$ and satisfies $\sum_{a \in \mathcal{A}} \hat{\rho}(s,a) = 1$. Since this holds for all $s \in \mathcal{S}$, we obtain $\hat{\rho} \in \mathcal{Q}$.

Suppose policy $\rho^* \in \mathcal{P}$ is optimal to problem (1.13), with corresponding optimal β_n-percentile values $y_n^{\rho^*}$. Lift this policy to $\hat{\rho}^*$. Define values $y_n^{\hat{\rho}^*}(\lambda)$ as

$$y_n^{\hat{\rho}^*}(\lambda) = y_n^{\rho^*} - \lambda q^T \left[\sum_{t=0}^{\infty} (\alpha \mathbb{P}_{\rho^*})^t \right] \rho_n^*. \tag{1.23}$$

We will now show that the pair of variables $\hat{\rho}^*$ and $y^{\hat{\rho}^*}(\lambda) = \frac{\lambda}{1-\alpha} + \sum_{n \in \mathcal{N}} y_n^{\hat{\rho}^*}(\lambda)$ is feasible to problem (1.22), and that the objective function value of this pair in problem (1.22) is identical to y^{ρ^*} — the optimal value in problem (1.13). This will then immediately yield the required bound.

To establish feasibility of the above pair, we use the matrix form of the rewards vector as defined in (1.20). The Lagrangian expected total discounted reward obtained from arm n

under policy $\hat{\rho}^*$

$$
\begin{aligned}
J_{q,n}^{\hat{\rho}^*,\lambda} =& q^T \left[\sum_{t=0}^{\infty}(\alpha \mathbb{P}_{\hat{\rho}^*})^t\right] r_n^{\hat{\rho}^*,\lambda} \\
=& q^T \left[\sum_{t=0}^{\infty}(\alpha \mathbb{P}_{\hat{\rho}^*})^t\right] \sum_{a\in\mathcal{A}|a_n=1} \operatorname{diag}\big(\hat{\rho}_a^*\big)(r_n - \lambda\vec{1}) && \text{(from (1.20))} \\
=& q^T \left[\sum_{t=0}^{\infty}(\alpha \mathbb{P}_{\hat{\rho}^*})^t\right] \operatorname{diag}\Big(\sum_{a\in\mathcal{A}|a_n=1} \hat{\rho}_a^*\Big)(r_n - \lambda\vec{1}) \\
=& q^T \left[\sum_{t=0}^{\infty}(\alpha \mathbb{P}_{\rho^*})^t\right] \operatorname{diag}(\rho_n^*)(r_n - \lambda\vec{1}) && \text{(by definition of the lifted policy } \hat{\rho}^*) \\
=& q^T \left[\sum_{t=0}^{\infty}(\alpha \mathbb{P}_{\rho^*})^t\right] \operatorname{diag}(\rho_n^*) r_n - \lambda q^T \left[\sum_{t=0}^{\infty}(\alpha \mathbb{P}_{\rho^*})^t\right] \operatorname{diag}(\rho_n^*)\vec{1} = J_{q,n}^{\rho^*} - \lambda q^T \left[\sum_{t=0}^{\infty}(\alpha \mathbb{P}_{\rho^*})^t\right] \rho_n^*.
\end{aligned}
$$

Using this, and the $y_n^{\hat{\rho}^*}(\lambda)$ defined in (1.23), the probabilistic constraint in the Lagrangian relaxed percentile problem is written as

$$
\begin{aligned}
&\mathbb{P}_{\tilde{r}_n}\left(J_{q,n}^{\hat{\rho}^*,\lambda} \geq y_n^{\hat{\rho}^*}(\lambda)\right) = \mathbb{P}_{\tilde{r}_n}\left(J_{q,n}^{\rho^*} - \lambda q^T \left[\sum_{t=0}^{\infty}(\alpha \mathbb{P}_{\rho^*})^t\right] \rho_n^* \geq y_n^{\rho^*} - \lambda q^T \left[\sum_{t=0}^{\infty}(\alpha \mathbb{P}_{\rho^*})^t\right] \rho_n^*\right) \\
&= \mathbb{P}_{\tilde{r}_n}\left(J_{q,n}^{\rho^*} \geq y_n^{\rho^*}\right) = \beta_n.
\end{aligned}
$$

The last inequality follows because $y_n^{\rho^*}$ is the β_n-percentile value associated with policy ρ^* in problem (1.13). This proves feasibility.

It remains to compare the objective values. We have,

$$\begin{aligned}
y^{\hat{\rho}^*}(\lambda) =& \frac{\lambda}{1-\alpha} + \sum_{n\in\mathcal{N}} y_n^{\hat{\rho}^*}(\lambda) = \frac{\lambda}{1-\alpha} + \sum_{n\in\mathcal{N}} y_n^{\rho^*} - \lambda \sum_{n\in\mathcal{N}} q^T \left[\sum_{t=0}^{\infty} (\alpha \mathbb{P}_{\rho^*})^t\right] \rho_n^* \\
=& \frac{\lambda}{1-\alpha} + \sum_{n\in\mathcal{N}} y_n^{\rho^*} - \lambda q^T \left[\sum_{t=0}^{\infty} (\alpha \mathbb{P}_{\rho^*})^t\right] \sum_{n\in\mathcal{N}} \rho_n^* = \frac{\lambda}{1-\alpha} + \sum_{n\in\mathcal{N}} y_n^{\rho^*} - \lambda q^T \left[\sum_{t=0}^{\infty} (\alpha \mathbb{P}_{\rho^*})^t\right] \vec{1} \\
=& \frac{\lambda}{1-\alpha} + \sum_{n\in\mathcal{N}} y_n^{\rho^*} - \lambda q^T \left[\sum_{t=0}^{\infty} \alpha^t \mathbb{P}_{\rho^*}^t \cancel{\vec{1}}^{\vec{1}}\right] = \frac{\lambda}{1-\alpha} + \sum_{n\in\mathcal{N}} y_n^{\rho^*} - \lambda q^T \left[\sum_{t=0}^{\infty} \alpha^t \vec{1}\right] \\
=& \frac{\lambda}{1-\alpha} + \sum_{n\in\mathcal{N}} y_n^{\rho^*} - \frac{\lambda}{1-\alpha} q^T \cancel{1}^{1} = \sum_{n\in\mathcal{N}} y_n^{\rho^*} = y^{\rho^*}.
\end{aligned}$$

We then have, $\max_{\rho\in\mathcal{P}} y^{\rho} = y^{\rho^*} = y^{\hat{\rho}^*}(\lambda) \leq \max_{\nu\in\mathcal{Q}} y^{\nu}(\lambda) = y^*(\lambda)$. Here, the inequality follows because $\hat{\rho}^* \in \mathcal{Q}$. This completes the proof. □

Similar to the previous section, using properties of the multivariate Gaussian distribution, the Lagrangian relaxed percentile problem (1.22) can be reformulated as the SOCP

$$\begin{aligned}
y^*(\lambda) = \frac{\lambda}{1-\alpha} + \max_x \sum_{n\in\mathcal{N}} \Bigg[& \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} x(s,a)\big(\mu_n(s_n) - \lambda\big) a_n \\
& - \Phi^{-1}(\beta_n) \sqrt{\sum_{s\in\mathcal{S}} \sum_{s'\in\mathcal{S}} \sum_{a\in\mathcal{A}} \sum_{a'\in\mathcal{A}} x(s,a) x(s',a') a_n a'_n \Theta_n(s_n, s'_n)} \Bigg] && \text{(1.24a)} \\
& \sum_{a\in\mathcal{A}} x(s,a) - \alpha \sum_{s'\in\mathcal{S}} \sum_{a\in\mathcal{A}} p(s|s',a) x(s',a) = q(s), \quad \forall s \in \mathcal{S} && \text{(1.24b)} \\
& x(s,a) \geq 0, \quad \forall s \in \mathcal{S}, a \in \mathcal{A}. && \text{(1.24c)}
\end{aligned}$$

In Proposition 1.4.2 below, we will show that problem (1.22) decomposes across individual arms. Toward that end, we first introduce randomized policies for individual arms. These policies assign probabilities $\delta'_n \in \Delta'_n$ to states $s_n \in \mathcal{S}_n$, where Δ'_n is defined as $\Delta'_n = \Big\{\delta'_n \in \mathbb{R}^2_+ | \sum_{a_n\in\{0,1\}} \delta'_n(a_n) = 1\Big\}$. The set of all such individual randomized policies is denoted by $\mathcal{Q}_n$. Consider such an individual randomized policy $\nu_n \in \mathcal{Q}_n$, which assigns probabilities

$\nu_n(s_n, 0)$ and $\nu_n(s_n, 1)$ to not playing and playing arm n in state $s_n \in \mathcal{S}_n$. Let $r_n^{\nu_n,\lambda} \in \mathbb{R}^{|\mathcal{S}_n|}$ denote the reward vector that contains values

$$\begin{aligned} r_n^{\lambda}(s_n, \nu_n(s_n)) &= \sum_{a_n \in \{0,1\}} \nu_n(s_n, a_n) r_n^{\lambda}(s_n, a_n) = \sum_{a_n \in \{0,1\}} \nu_n(s_n, a_n)(r_n(s_n) - \lambda) a_n \\ &= \nu_n(s_n, 1)(r_n(s_n) - \lambda), \end{aligned}$$

for all $s_n \in \mathcal{S}_n$. Similarly, $\mathbb{P}_{\nu_n} \in \mathbb{R}^{|\mathcal{S}_n| \times |\mathcal{S}_n|}$ denotes the transition probability matrix induced by policy $\nu_n \in \mathcal{Q}_n$ for arm n. The entry in row s_n and column s_n' of this matrix is given by

$$p_n(s_n'|s_n, \nu_n(s_n)) = \sum_{a_n \in \{0,1\}} \nu_n(s_n, a_n) p_n(s_n'|s_n, a_n).$$

When the Lagrange multiplier is fixed at λ, the value of implementing randomized policy $\nu_n \in \mathcal{Q}_n$ for arm n upon starting in state s_n is given by

$$J^{\nu_n,\lambda}(s_n) = \mathbb{E}\left(\sum_{t=0}^{\infty} \alpha^t r^{\lambda}(\mathbf{s}_n^t, \nu_n(\mathbf{s}_n^t)) | \mathbf{s}_n^0 = s_n \right).$$

The expectation of $J^{\nu_n,\lambda}$ with respect to the initial state distribution q_n is defined as

$$J_{q_n}^{\nu_n,\lambda} = \sum_{s_n \in \mathcal{S}_n} q_n(s_n) J^{\nu_n,\lambda}(s_n) = \underbrace{q_n^T \left[\sum_{t=0}^{\infty} (\alpha \mathbb{P}_{\nu_n})^t \right]}_{J^{\nu_n,\lambda}} r_n^{\nu_n,\lambda}.$$

Here, $J_{q_n}^{\nu_n,\lambda} \in \mathbb{R}^{|\mathcal{S}_n|}$, whose entry $J_{q_n}^{\nu_n,\lambda}(s_n)$ equals the total expected discounted reward obtained from arm n under immediate rewards $r_n^{\nu_n,\lambda} \in \mathbb{R}^{|\mathcal{S}_n|}$, initial state distribution q_n, and policy ν_n, when the Lagrange multiplier is fixed at λ. It is important to note the difference between $J_n^{\nu,\lambda}$ that was defined in (1.21) and $J^{\nu_n,\lambda}$ defined above. The former is defined over $\mathbb{R}^{|\mathcal{S}|}$; the latter is defined over $\mathbb{R}^{|\mathcal{S}_n|}$. We define the percentile optimization problem for each

arm $n \in \mathcal{N}$ as

$$y_n^*(\lambda) = \max_{y_n(\lambda), \nu_n \in \mathcal{Q}_n} y_n(\lambda) \tag{1.25a}$$

$$\mathbb{P}_{\tilde{r}_n}\Big(J_{q_n}^{\nu_n,\lambda} \geq y_n(\lambda)\Big) \geq \beta_n. \tag{1.25b}$$

As before, the percentile subproblem for arm n is equivalent to the SOCP

$$y_n^*(\lambda) = \max_{x_n} \sum_{s_n \in \mathcal{S}_n} x_n(s_n, 1)\big(\mu_n(s_n) - \lambda\big) - \Phi^{-1}(\beta_n)\sqrt{\sum_{s_n \in \mathcal{S}_n}\sum_{s_n' \in \mathcal{S}_n} x_n(s_n,1)x_n(s_n',1)\Theta_n(s_n,s_n')} \tag{1.26a}$$

$$\sum_{a_n \in \{0,1\}} x_n(s_n, a_n) - \alpha \sum_{s_n' \in \mathcal{S}_n}\sum_{a_n \in \{0,1\}} p_n(s_n|s_n', a_n)x_n(s_n', a_n) = q_n(s_n), \quad \forall s_n \in \mathcal{S}_n \tag{1.26b}$$

$$x_n(s_n, a_n) \geq 0, \quad \forall s_n \in \mathcal{S}_n, a_n \in \{0,1\}. \tag{1.26c}$$

Note that the terms $x_n(s_n, 0)$ and $x_n(s_n', 0)$ do not appear in the objective function, because they would be multiplied by $a_n = 0$ and $a_n' = 0$.

Proposition 1.4.2. *The optimal value in the Lagrangian relaxed percentile problem (1.22) can be expressed as*

$$y^*(\lambda) = \frac{\lambda}{1-\alpha} + \sum_{n \in \mathcal{N}} y_n^*(\lambda), \tag{1.27}$$

where $y_n^*(\lambda)$ are the optimal values in problems (1.25) for arms $n \in \mathcal{N}$.

Proof. Let x be a feasible solution to problem (1.24). Let

$$x_n(\hat{s}_n, \hat{a}_n) = \sum_{a \in \mathcal{A} | a_n = \hat{a}_n} \sum_{s \in \mathcal{S} | s_n = \hat{s}_n} x(s, a), \quad \forall \hat{s}_n \in \mathcal{S}_n, \hat{a}_n \in \{0,1\}. \tag{1.28}$$

We show that this x_n is feasible to problem (1.26). It is clearly nonnegative, and hence feasible to (1.26c), because x is nonnegative by (1.24c). To check feasibility of x_n to constraint (1.26b)

for $\hat{s}_n$, we write,

$$\begin{aligned}
&\sum_{\hat{a}_n\in\{0,1\}} x_n(\hat{s}_n,\hat{a}_n) - \alpha \sum_{\hat{s}'_n\in\mathcal{S}_n}\sum_{\hat{a}_n\in\{0,1\}} p_n(\hat{s}_n|\hat{s}'_n,\hat{a}_n)x_n(\hat{s}'_n,\hat{a}_n) \\
&= \sum_{\hat{a}_n\in\{0,1\}}\sum_{a|a_n=\hat{a}_n}\sum_{s|s_n=\hat{s}_n} x(s,a) - \alpha \sum_{\hat{s}'_n\in\mathcal{S}_n}\sum_{\hat{a}_n\in\{0,1\}} p_n(\hat{s}_n|\hat{s}'_n,\hat{a}_n)\left[\sum_{a|a_n=\hat{a}_n}\sum_{s'|s'_n=\hat{s}'_n} x(s',a)\right] \\
&= \sum_{s|s_n=\hat{s}_n}\sum_{a\in\mathcal{A}} x(s,a) - \alpha\sum_{s'\in\mathcal{S}}\sum_{a\in\mathcal{A}} p_n\big(\hat{s}_n|s'_n,a_n\big)x(s',a). \qquad (1.29)
\end{aligned}$$

The expression above can be further simplified by observing, for any $s'\in\mathcal{S}$ and $a\in\mathcal{A}$, that

$$\begin{aligned}
&\sum_{s|s_n=\hat{s}_n} p(s|s',a) = \sum_{s_1\in\mathcal{S}_1}\cdots\sum_{s_{n-1}\in\mathcal{S}_{n-1}}\sum_{s_{n+1}\in\mathcal{S}_{n+1}}\cdots\sum_{s_N\in\mathcal{S}_N} p(s_1,\ldots,s_{n-1},\hat{s}_n,s_{n+1},\ldots,s_N|s',a) \\
&= (a_n p_n(\hat{s}_n|s'_n) + (1-a_n)\gamma_n(\hat{s}_n|s'_n)) \\
&\underbrace{\sum_{s_1\in\mathcal{S}_1}\cdots\sum_{s_{n-1}\in\mathcal{S}_{n-1}}\sum_{s_{n+1}\in\mathcal{S}_{n+1}}\cdots\sum_{s_N\in\mathcal{S}_N}\prod_{\substack{i=1\\ i\neq n}}^{N}(a_i p_i(s_i|s'_i)+(1-a_i)\gamma_i(s_i|s'_i))}_{1} \\
&= (a_n p_n(\hat{s}_n|s'_n) + (1-a_n)\gamma_n(\hat{s}_n|s'_n)) = p_n(\hat{s}_n|s'_n,a_n).
\end{aligned}$$

Here, the second equality follows from (1.2). Substituting this into (1.29) yields

$$\begin{aligned}
&\sum_{s|s_n=\hat{s}_n}\sum_{a\in\mathcal{A}} x(s,a) - \alpha\sum_{s'\in\mathcal{S}}\sum_{a\in\mathcal{A}}\left(\sum_{s|s_n=\hat{s}_n} p(s|s',a)\right)x(s',a) \\
&= \sum_{s|s_n=\hat{s}_n}\left[\sum_{a\in\mathcal{A}} x(s,a) - \alpha\sum_{s'\in\mathcal{S}}\sum_{a\in\mathcal{A}} p(s|s',a)x(s',a)\right].
\end{aligned}$$

Observe that this expression is the sum of constraints (1.24*b*) over all states $s\in\mathcal{S}$ whose n^{th} element equals $\hat{s}_n\in\mathcal{S}_n$. Therefore, it equals the sum of the corresponding right-hand-sides,

given by

$$\sum_{s|s_n=\hat{s}_n} q(s) = \sum_{s|s_n=\hat{s}_n} \left(\prod_{i=1}^{N} q_i(s_i) \right) = \sum_{s|s_n=\hat{s}_n} \left(q_n(\hat{s}_n) \prod_{\substack{i=1 \\ i\neq n}}^{N} q_i(s_i) \right) = q_n(\hat{s}_n) \prod_{\substack{i=1 \\ i\neq n}}^{N} \left(\sum_{s_i \in \mathcal{S}_i} q_i(s_i) \right)^{1}$$

$$= q_n(\hat{s}_n).$$

This is equal to the right-hand-side of constraint (1.26b). Therefore, feasibility holds.

Consider problem (1.26) for any fixed n. Its objective value includes two components. The expected value part and the variance part. The expected value part is

$$\begin{aligned}
\sum_{\hat{s}_n \in \mathcal{S}_n} x_n(\hat{s}_n, 1)\big(\mu_n(\hat{s}_n) - \lambda\big) &= \sum_{\hat{s}_n \in \mathcal{S}_n} \left(\sum_{s\in\mathcal{S}|s_n=\hat{s}_n} \sum_{a\in\mathcal{A}|a_n=1} x(s,a) \right) \big(\mu_n(\hat{s}_n) - \lambda\big) \quad \text{(from (1.28))} \\
&= \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}|a_n=1} x(s,a)\big(\mu_n(s_n) - \lambda\big) \\
&= \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} x(s,a)\big(\mu_n(s_n) - \lambda\big) a_n. \qquad (1.30)
\end{aligned}$$

The variance part is

$$\begin{aligned}
&\sum_{\hat{s}_n \in \mathcal{S}_n} \sum_{\hat{s}'_n \in \mathcal{S}_n} x_n(\hat{s}_n, 1) x_n(\hat{s}'_n, 1) \Theta_n(\hat{s}_n, \hat{s}'_n) \\
&= \sum_{\hat{s}_n \in \mathcal{S}_n} \sum_{\hat{s}'_n \in \mathcal{S}_n} \left[\sum_{s\in\mathcal{S}|s_n=\hat{s}_n} \sum_{a\in\mathcal{A}|a_n=1} x(s,a) \right] \left[\sum_{s'\in\mathcal{S}|s'_n=\hat{s}'_n} \sum_{a'\in\mathcal{A}|a'_n=1} x(s',a') \right] \Theta_n(\hat{s}_n, \hat{s}'_n) \\
&= \sum_{s\in\mathcal{S}} \sum_{s'\in\mathcal{S}} \sum_{a\in\mathcal{A}|a_n=1} \sum_{a'\in\mathcal{A}|a'_n=1} x(s,a) x(s',a') \Theta_n(s_n, s'_n) \\
&= \sum_{s\in\mathcal{S}} \sum_{s'\in\mathcal{S}} \sum_{a\in\mathcal{A}} \sum_{a'\in\mathcal{A}} x(s,a) x(s',a') a_n a'_n \Theta_n(s_n, s'_n). \qquad (1.31)
\end{aligned}$$

Now, let x^* be an optimal solution to (1.24) and let x^*_n be the corresponding feasible solutions to problems (1.26), constructed as per (1.28). Thus, the optimal objective value in

(1.24) is

$$
\begin{aligned}
y^*(\lambda) &= \frac{\lambda}{1-\alpha} + \sum_{n\in\mathcal{N}} \Bigg[\sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} x^*(s,a)\big(\mu_n(s_n) - \lambda\big) a_n \\
&\quad - \Phi^{-1}(\beta_n) \sqrt{\sum_{s\in\mathcal{S}} \sum_{s'\in\mathcal{S}} \sum_{a\in\mathcal{A}} \sum_{a'\in\mathcal{A}} x^*(s,a) x^*(s',a') a_n a'_n \Theta_n(s_n, s'_n)} \Bigg] \\
&= \frac{\lambda}{1-\alpha} + \sum_{n\in\mathcal{N}} \Bigg[\sum_{\hat{s}_n\in\mathcal{S}_n} x^*_n(\hat{s}_n, 1)\big(\mu_n(\hat{s}_n) - \lambda\big) \\
&\qquad - \Phi^{-1}(\beta_n) \sqrt{\sum_{\hat{s}_n\in\mathcal{S}_n} \sum_{\hat{s}'_n\in\mathcal{S}_n} x^*_n(\hat{s}_n, 1) x^*_n(\hat{s}'_n, 1) \Theta_n(\hat{s}_n, \hat{s}'_n)} \Bigg] \quad \text{(from (1.30) and (1.31))} \\
&\leq \frac{\lambda}{1-\alpha} + \sum_{n\in\mathcal{N}} y^*_n(\lambda). \qquad (1.32)
\end{aligned}
$$

Here, the inequality holds because on the left-hand-side we have feasible objective values for subproblems (1.26) but on the right-hand-side we have optimal objective values $y^*_n(\lambda)$.

Conversely, consider policies $\nu^*_n \in \mathcal{Q}_n$ optimal for the decomposed Lagrangian relaxed problems (1.25), for $n \in \mathcal{N}$. Construct a policy $\nu^* \in \mathcal{Q}$ for the original Lagrangian relaxed problem (1.22) using the formula $\nu^*(s,a) = \prod_{n\in\mathcal{N}} \nu^*_n(s_n, a_n)$, for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$. Since this policy is multiplicatively separable and since the initial state probabilities $q(s)$ equal $\prod_{n\in\mathcal{N}} q_n(s_n)$, we have,

$$
J^{\nu^*,\lambda}_{q,n} = J^{\nu^*_n,\lambda}_{q_n}, \qquad (1.33)
$$

for every realization of $\tilde{r}_n$ and all arms $n \in \mathcal{N}$. This shows that

$$
\begin{aligned}
y^*(\lambda) &= \frac{\lambda}{1-\alpha} + \max_{y(\lambda)\in\mathbb{R}^N, \nu\in\mathcal{Q}} \left\{ \sum_{n\in\mathcal{N}} y_n(\lambda) \middle| \mathbb{P}_{\tilde{r}_n}\Big(J_{q,n}^{\nu,\lambda} \geq y_n(\lambda) \Big) \geq \beta_n, \ \forall n \in \mathcal{N} \right\} \\
&\geq \frac{\lambda}{1-\alpha} + \max_{y(\lambda)\in\mathbb{R}^N} \left\{ \sum_{n\in\mathcal{N}} y_n(\lambda) \middle| \mathbb{P}_{\tilde{r}_n}\Big(J_{q,n}^{\nu^*,\lambda} \geq y_n(\lambda) \Big) \geq \beta_n, \ \forall n \in \mathcal{N} \right\} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(feasibility of policy } \nu^*) \\
&= \frac{\lambda}{1-\alpha} + \max_{y(\lambda)\in\mathbb{R}^N} \left\{ \sum_{n\in\mathcal{N}} y_n(\lambda) \middle| \mathbb{P}_{\tilde{r}_n}\Big(J_{q,n}^{\nu_n^*,\lambda} \geq y_n(\lambda) \Big) \geq \beta_n, \ \forall n \in \mathcal{N} \right\} \qquad \text{(from (1.33))} \\
&= \frac{\lambda}{1-\alpha} + \sum_{n\in\mathcal{N}} y_n^*(\lambda). \qquad (1.34)
\end{aligned}
$$

Here, the last equality holds by optimality of policies ν_n^* to problems (1.25), for $n \in \mathcal{N}$. Combining inequalities (1.32) and (1.34) establishes the claim. □

Since Proposition 1.4.1 established that $y^*(\lambda) \geq y^*$ for any $\lambda \in \mathbb{R}$, we attempt to find a λ that yields the smallest value of $y^*(\lambda)$. This would lead to the tightest upper bound on y^*. We substitute $y_n^*(\lambda)$ from (1.26) into formula (1.27) for $y^*(\lambda)$ from Proposition 1.4.2. We then minimize the resulting expression for $y^*(\lambda)$ over $\lambda \in \mathbb{R}$. This yields the problem

$$
\begin{aligned}
\min_{\lambda\in\mathbb{R}} \ y^*(\lambda) = \min_{\lambda} \Bigg\{ & \frac{\lambda}{1-\alpha} + \sum_{n\in\mathcal{N}} \max_{x_n} \sum_{s_n\in\mathcal{S}_n} x_n(s_n,1)\big(\mu_n(s_n) - \lambda\big) \\
& \qquad - \Phi^{-1}(\beta_n) \sqrt{\sum_{s_n\in\mathcal{S}_n} \sum_{s_n'\in\mathcal{S}_n} x_n(s_n,1) x_n(s_n',1) \Theta_n(s_n,s_n')} \qquad (1.35a) \\
& \sum_{a_n\in\{0,1\}} x_n(s_n,a_n) - \alpha \sum_{s_n'\in\mathcal{S}_n} \sum_{a_n\in\{0,1\}} p_n(s_n|s_n',a_n) x_n(s_n',a_n) = q_n(s_n), \quad \forall s_n \in \mathcal{S}_n \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (1.35b) \\
& x_n(s_n,a_n) \geq 0, \quad \forall s_n \in \mathcal{S}_n, a_n \in \{0,1\} \Bigg\}. \qquad (1.35c)
\end{aligned}
$$

Proposition 1.4.3. *Using strong duality, the inner maximization problems in (1.35) can be*

replaced with their duals. As a result, problem (1.35) can be converted into the SOCP

$$\min_{\lambda\in\mathbb{R}} \; y^*(\lambda) = \min_{\lambda,V,f} \; \frac{\lambda}{1-\alpha} + \sum_{n\in\mathcal{N}} q_n^T V_n \tag{1.36a}$$

$$\big(\Theta_n\big)^{1/2} f_n + (\mathbb{I} - \alpha\mathbb{P}_n)V_n \geq \mu_n - \lambda\vec{1}, \qquad \forall n \in \mathcal{N} \tag{1.36b}$$

$$\|f_n\|_2 \leq \Phi^{-1}(\beta_n), \qquad \forall n \in \mathcal{N} \tag{1.36c}$$

$$V_n \geq 0, \qquad \forall n \in \mathcal{N}. \tag{1.36d}$$

Here, V and f denote the collection of decision variables $\{V_n\}_{n\in\mathcal{N}}$ and $\{f_n\}_{n\in\mathcal{N}}$, where both $V_n, f_n \in \mathbb{R}^{|\mathcal{S}_n|}$.

Proof. For simplicity, we focus only on the inner problem associated with one arm $n \in \mathcal{N}$. The inner problem for arm n, which we view as the primal problem, is

$$\max_{x_n} \sum_{s_n\in\mathcal{S}_n} x_n(s_n,1)\big(\mu_n(s_n) - \lambda\big) - \Phi^{-1}(\beta_n)\sqrt{\sum_{s_n\in\mathcal{S}_n}\sum_{s_n'\in\mathcal{S}_n} x_n(s_n,1)x_n(s_n',1)\Theta_n(s_n,s_n')} \tag{1.37a}$$

$$\sum_{a_n\in\{0,1\}} x_n(s_n,a_n) - \alpha\sum_{s_n'\in\mathcal{S}_n}\sum_{a_n\in\{0,1\}} p_n(s_n|s_n',a_n)x_n(s_n',a_n) = q_n(s_n), \quad \forall s_n \in \mathcal{S}_n \tag{1.37b}$$

$$x_n(s_n,a_n) \geq 0, \quad \forall s_n \in \mathcal{S}_n, a_n \in \{0,1\}. \tag{1.37c}$$

We use column vector $x_0 \in \mathbb{R}^{|\mathcal{S}_n|}$ with entries $x_n(s_n,0)$, for $s_n \in \mathcal{S}_n$, as a shorthand notation. Similarly, we fill column vector $x_1 \in \mathbb{R}^{|\mathcal{S}_n|}$ with entries $x_n(s_n,1)$, for $s_n \in \mathcal{S}_n$. The matrix form of the above primal problem is

$$\max_{x_0,x_1} \; x_1^T(\mu_n - \lambda\vec{1}) - \Phi^{-1}(\beta_n)\|\Theta_n^{\frac{1}{2}}x_1\|_2 \tag{1.38a}$$

$$(\mathbb{I} - \alpha\mathbb{P}_n^T)x_1 + (1-\alpha)x_0 = q_n \tag{1.38b}$$

$$x_0, x_1 \geq 0. \tag{1.38c}$$

Here, column vector $\mu_n \in \mathbb{R}^{|\mathcal{S}_n|}$ is comprised of entries $\mu_n(s_n)$, for $s_n \in \mathcal{S}_n$; and matrix $\mathbb{P}_n \in \mathbb{R}^{|\mathcal{S}_n|\times|\mathcal{S}_n|}$ stores the transition probabilities associated with pulling arm n. Since the

objective function is concave and constraints are linear, this maximization problem is convex.

We introduce the variable transformation $\Theta_n^{\frac{1}{2}} x_1 = z_n \in \mathbb{R}^{|\mathcal{S}_n|}$ and denote its associated dual variable as $f_n \in \mathbb{R}^{|\mathcal{S}_n|}$. The dual variable associated with constraint (1.38b) is denoted by $V_n \in \mathbb{R}^{|\mathcal{S}_n|}$. The Lagrangian is

$$\begin{aligned}
&\mathcal{L}(x_0, x_1, z_n; V_n, f_n) \\
&= x_1^T(\mu_n - \lambda\vec{1}) - \Phi^{-1}(\beta_n)\|z_n\|_2 + V_n^T\left(q_n - (\mathbb{I} - \alpha\mathbb{P}_n^T)x_1 - (1-\alpha)x_0\right) + f_n^T(z_n - \Theta_n^{\frac{1}{2}} x_1).
\end{aligned}$$

The Lagrangian dual problem is given by

$$\begin{aligned}
&\min_{V_n, f_n} \max_{x_0 \geq 0, x_1 \geq 0, z_n} \mathcal{L}(x_0, x_1, z_n; V_n, f_n) \\
&= \min_{V_n, f_n} \Bigg(q_n^T V_n + \max_{x_0 \geq 0}\left\{-(1-\alpha)x_0^T V_n\right\} + \max_{x_1 \geq 0}\left\{x_1^T\left(\mu_n - \lambda\vec{1} - (\mathbb{I} - \alpha\mathbb{P}_n)V_n - \Theta_n^{\frac{1}{2}} f_n\right)\right\} \\
&\qquad + \max_{z_n}\left\{f_n^T z_n - \Phi^{-1}(\beta_n)\|z_n\|_2\right\}\Bigg).
\end{aligned}$$

The value of the first inner maximization problem over x_0 is

$$\max_{x_0 \geq 0}\Big\{ -(1-\alpha)x_0^T V_n\Big\} = \begin{cases} 0 & \text{if } V_n \geq 0 \\ +\infty & \text{otherwise.} \end{cases}$$

Similarly, the value of the second inner maximization problem over x_1 can be written as

$$\max_{x_1 \geq 0}\Big\{x_1^T\big(\mu_n - \lambda\vec{1} - (\mathbb{I} - \alpha\mathbb{P}_n)V_n - \Theta_n^{\frac{1}{2}} f_n\big)\Big\} = \begin{cases} 0 & \text{if } \mu_n - \lambda\vec{1} - (\mathbb{I} - \alpha\mathbb{P}_n)V_n - \Theta_n^{\frac{1}{2}} f_n \leq 0 \\ +\infty & \text{otherwise.} \end{cases}$$

Lastly, the value of the third inner maximization problem in variable z_n is given by

$$\begin{aligned}\max_{z_n}\Big\{f_n^T z_n - \Phi^{-1}(\beta_n)\|z_n\|_2\Big\} &= \max_{z_n}\Big\{\|f_n\|_2\|z_n\|_2 - \Phi^{-1}(\beta_n)\|z_n\|_2\Big\}\\ &= \max_{z_n}\Big\{\Big(\|f_n\|_2 - \Phi^{-1}(\beta_n)\Big)\|z_n\|_2\Big\}\\ &= \begin{cases} 0 & \text{if } \|f_n\|_2 - \Phi^{-1}(\beta_n) \leq 0\\ +\infty & \text{otherwise.}\end{cases}\end{aligned}$$

Using these, the dual problem can be written as

$$\min_{V_n, f_n}\ q_n^T V_n \tag{1.39a}$$

$$(\mathbb{I} - \alpha\mathbb{P}_n)V_n + \Theta_n^{1/2} f_n \geq \mu_n - \lambda\vec{1} \tag{1.39b}$$

$$\|f_n\|_2 \leq \Phi^{-1}(\beta_n) \tag{1.39c}$$

$$V_n \geq 0. \tag{1.39d}$$

Since the objective function of problem (1.37) is concave and all constraints are linear, Proposition 5.2.1 in [10] implies that the duality gap is zero. This means that solving problem (1.37) is equivalent to solving problem (1.39). This completes the proof. □

Problem (1.36) has $1 + 2 \times \sum_{n\in\mathcal{N}} |\mathcal{S}_n|$ variables and $2\sum_{n\in\mathcal{N}} |\mathcal{S}_n| + N$ constraints. Thus, its size grows linearly with N. Contrasting this with the original exact SOCP (1.16), Lagrangian relaxation has thus achieved an exponential reduction in problem size. In the next section, we describe three methods to retrieve feasible arm-pulling decisions for the original percentile MAB problem (1.13) in run-time, using an optimal solution to problem (1.36).

1.5 Recovering feasible decisions in run-time

Let $\lambda^* \in \mathbb{R}$; $V_n^* \in \mathbb{R}^{|\mathcal{S}_n|}$ and $f_n^* \in \mathbb{R}^{|\mathcal{S}_n|}$, for all $n \in \mathcal{N}$, denote an optimal solution to problem (1.36). Let $y^*(\lambda^*)$ denote the corresponding optimal objective value. Suppose the MAB

problem is in some state $s = (s_1, \ldots, s_N)$ at some time-step $t = 0, 1, 2, \ldots$. The decision-maker then needs to select an arm to pull in this state. The next three sections describe three competing approaches to achieve this.

1.5.1 One-step lookahead

Let π_n denote the probability of playing arm n. The immediate random reward earned upon playing arm n is then $\pi_n \tilde{r}_n(s_n) + (1 - \pi_n)0 = \pi_n \tilde{r}_n(s_n)$. The next state for arm n then equals $s'_n \in \mathcal{S}_n$ with probability $\pi_n p_n(s'_n|s_n)$ and it equals s_n with probability $(1 - \pi_n)$. We approximate the expected total discounted reward earned from all plays of arm n starting from these states with $V_n^*(s'_n)$ and $V_n^*(s_n)$, respectively. This means that the approximate random total discounted reward earned from all plays of arm n equals

$$\pi_n \tilde{r}_n(s_n) + \alpha \sum_{s'_n \in \mathcal{S}} \pi_n p_n(s'_n|s_n) V_n^*(s'_n) + \alpha(1 - \pi_n) V_n^*(s_n).$$

The idea in one-step lookahead is to find probabilities π_n, for all arms n, that maximize the sum of β_n-percentiles of these approximations. This idea is motivated by one-step lookahead in dynamic programming, where value functions on the right-hand-side of Bellman's equations are replaced with their approximations [11]. This yields the problem

$$\max_{y \in \mathbb{R}^N, \pi \in \mathbb{R}^N} \sum_{n \in \mathcal{N}} y_n \tag{1.40a}$$

$$\mathbb{P}_{\tilde{r}_n(s_n)}\Big(\pi_n \tilde{r}_n(s_n) + \alpha \sum_{s'_n \in \mathcal{S}} \pi_n p_n(s'_n|s_n) V_n^*(s'_n) + \alpha(1 - \pi_n) V_n^*(s_n) \geq y_n\Big) \geq \beta_n, \quad \forall n \in \mathcal{N} \tag{1.40b}$$

$$\sum_{n \in \mathcal{N}} \pi_n = 1 \tag{1.40c}$$

$$\pi_n \geq 0, \quad \forall n \in \mathcal{N}. \tag{1.40d}$$

Since $\tilde{r}_n(s_n)$ is univariate Normal, the β_n-percentile above can be written in closed form as described earlier. After some algebraic simplification and dropping constant terms from the objective, the above problem therefore reduces to the LP

$$\max_{\pi} \sum_{n\in\mathcal{N}} \pi_n \Big(\mu_n(s_n) + \alpha \sum_{s'_n\in\mathcal{S}} p_n(s'_n|s_n)V^*_n(s'_n) - \alpha V^*_n(s_n) - \Phi^{-1}(\beta_n)\sqrt{\Theta_n(s_n,s_n)}\Big) \quad (1.41a)$$

$$\sum_{n\in\mathcal{N}} \pi_n = 1 \quad (1.41b)$$

$$\pi_n \geq 0, \quad \forall n \in \mathcal{N}. \quad (1.41c)$$

Observe that in this LP, it is optimal to set $\pi_n = 1$ for any one arm with the largest value of $\Big\{\mu_n(s_n) + \alpha \sum_{s'_n\in\mathcal{S}} p_n(s'_n|s_n)V^*_n(s'_n) - \alpha V^*_n(s_n) - \Phi^{-1}(\beta_n)\sqrt{\Theta_n(s_n,s_n)}\Big\}$; and to set $\pi_n = 0$ for all other arms. That is, it is optimal to play any one arm from the set

$$\operatorname*{argmax}_{n\in\mathcal{N}} \Big\{\mu_n(s_n) + \alpha \sum_{s'_n\in\mathcal{S}} p_n(s'_n|s_n)V^*_n(s'_n) - \alpha V^*_n(s_n) - \Phi^{-1}(\beta_n)\sqrt{\Theta_n(s_n,s_n)}\Big\}.$$

Thus, in this case, the recovered decision is of the deterministic index-type.

1.5.2 One-step lookahead with a single chance constraint

The one-step lookahead approach in the above section produces a feasible deterministic action. To recover a potentially randomized feasible decision, we solve

$$\max_{z\in\mathbb{R},\pi\in\mathbb{R}^N} \; z \quad (1.42a)$$

$$\mathbb{P}_{\tilde{r}}\left(\sum_{n\in\mathcal{N}} \pi_n \big[\tilde{r}_n(s_n) + \alpha \sum_{s'_n\in\mathcal{S}_n} p_n(s'_n|s_n)J(s_1,\ldots,s'_n,\ldots,s_N)\big] \geq z\right) \geq \beta \quad (1.42b)$$

$$\sum_{n\in\mathcal{N}} \pi_n = 1 \quad (1.42c)$$

$$\pi_n \geq 0, \quad \forall n \in \mathcal{N}. \quad (1.42d)$$

Here, we have used the shorthand notations $\beta = \prod_{n\in\mathcal{N}} \beta_n$, and

$$J(u) = \sum_{n\in\mathcal{N}} V_n^*(u_n),\ \forall u = (u_1, \ldots, u_N) \in \mathcal{S}. \tag{1.43}$$

As an aside, we note that problem (1.42) is a relaxation of problem (1.40). In particular, for any feasible solution $y \in \mathbb{R}^N, \pi \in \mathbb{R}^N$ for (1.40), the solution $z = \sum_{n\in\mathcal{N}} y_n$ and π is feasible to (1.42) with an identical objective value as in (1.40). Problem (1.42) also has another interpretation. If arm n is played in state s_n, the immediate reward is $\tilde{r}_n(s_n)$ and then the next state equals $(s_1, \ldots, s_{n-1}, s'_n, s_{n+1}, \ldots, s_N)$ with probability $p_n(s'_n|s_n)$. The expression in (1.43) approximates the expected total discounted reward upon starting in this state with $V_n^*(s'_n) + \sum_{\substack{m\in\mathcal{N}\\ m\neq n}} V_m^*(s_m)$. This approximation is employed in constraint (1.42b) of the one-step lookahead problem (1.42).

Again, using properties of multivariate Gaussian random vectors, (1.42) simplifies to the SOCP

$$\max_{\pi} \sum_{n\in\mathcal{N}} \pi_n\Big(\mu_n(s_n) + \alpha \sum_{s'_n\in\mathcal{S}} p_n(s'_n|s_n) J(s_1, \ldots, s'_n, \ldots, s_N)\Big) - \Phi^{-1}(\beta)\sqrt{\sum_{n\in\mathcal{N}} \pi_n^2 \Theta_n(s_n, s_n)} \tag{1.44a}$$

$$\sum_{n\in\mathcal{N}} \pi_n = 1 \tag{1.44b}$$

$$\pi_n \geq 0, \quad \forall n \in \mathcal{N}, \tag{1.44c}$$

if $\beta \geq 0.5$.

1.5.3 Normalized decisions

Instead of using the optimal dual variable values V_n^*, we can use the optimal values of primal variables x_n^* from problem (1.24) at $\lambda = \lambda^*$ to recover feasible decisions for the original percentile MAB problem. Using these values of x_n^*, we define the (unconstrained)

probabilities of playing arm n in state $s_n \in \mathcal{S}_n$ as

$$\pi_n^*(s_n, 1) = \frac{x_n^*(s_n, 1)}{x_n^*(s_n, 1) + x_n^*(s_n, 0)}, \quad \forall s_n \in \mathcal{S}_n, n \in \mathcal{N}. \tag{1.45}$$

However, the probabilities of playing different arms should add to 1. We therefore define a feasible randomized action via normalization as

$$\hat{\pi}_n(s, 1) = \begin{cases} \frac{\pi_n^*(s_n,1)}{\sum_{j\in\mathcal{N}} \pi_j^*(s_j,1)} & \forall n \in \mathcal{N}, \quad \text{if } \sum_{j\in\mathcal{N}} \pi_j^*(s_j, 1) > 0, \\ \frac{1}{N}, & \text{otherwise.} \end{cases} \tag{1.46}$$

Here, $\hat{\pi}_n(s, 1)$ denotes the probability of playing arm n.

1.6 Numerical results

The objective of our numerical experiments is to compare the above three methods to recover feasible actions, and to conduct sensitivity analyses to gain insights into the effect of problem parameters. Throughout this chapter, all convex programs were solved using the cvxpy package on Python 3.8.

1.6.1 Comparison of decision recovery methods

Our simulation study included 5 sets of MAB problems with different number of arms and states per arm. For each set of problems we generated 20 instances as follows. For each arm, the initial state distributions q_n and transitions probabilities $\mathbb{P}_n$ were sampled from a uniform $[0, 1]$ distribution and then normalized. The expected rewards μ_n were sampled from the uniform distribution on $[1, 20]$. To ensure that the randomly generated covariance matrices were positive semi-definite, we first sampled an $n \times n$ matrix denoted by A_n from a uniform $[-5, 5]$ distribution, for all $n \in \mathcal{N}$. The covariance matrix was then generated as $\Theta_n = A_n(A_n)^T$.

Recall that the Lagrangian bound obtained by solving problem (1.36) is denoted by

$y^*(\lambda^*)$. We use $\hat{y}^1$, $\hat{y}^2$, and $\hat{y}^3$, respectively, to denote the estimated percentile values earned by implementing the decision recovery methods described in Sections 1.5.1, 1.5.2, and 1.5.3. These values were computed using the simulation process summarized in Algorithm 1. To derive insights into the performance of the three decision recovery methods, we compared them against a myopic decision approach, where an arm $\hat{n} \in \underset{n \in \mathcal{N}}{\operatorname{argmax}}\{\mu_n(s_n)\}$ with the largest expected immediate reward is played in state $s = (s_1, \ldots, s_N)$. Its estimated percentile value is denoted by $\hat{y}^{\text{myopic}}$.

Since the one-step lookahead method in Section 1.5.1 and the myopic approach return deterministic decisions, we ran a larger number of simulation threads for their evaluation in the hope that a large variety of states were visited. For these two methods, we employed $M = 10,000$ threads whereas for the methods in Sections 1.5.2 and 1.5.3 we used $M = 1,000$ threads. Each simulation thread contained either 40 time-steps (for discount factor $\alpha = 0.80$) or 175 time-steps (for $\alpha = 0.95$), noting that $0.80^{40} \approx 0.95^{175} \leq 10^{-4}$ [1].

Estimated percentile values, averaged over 20 instance within each problem set, are plotted in Figure 1.1. Each problem set is identified by a pair $(N, |\mathcal{S}_n|)$ on the x-axis. Recall here that, N denotes the number of arms; and $|\mathcal{S}_n|$ denotes the number of states per arm, assumed to be invariant across arms. The figure only includes results for $\alpha = 0.95$ as the qualitative patterns were similar when $\alpha = 0.80$.

Algorithm 1 Simulation to compare performance of decision recovery methods

Input: $(\mu_n, \Theta_n, \mathbb{P}_n, q_n, \beta_n)$ for all arms $n \in \mathcal{N}$; number M of simulation threads; number T of time-steps per simulation thread; name of feasible decision recovery method

Solve the dual Lagrangian relaxed problem (1.36) to compute λ^*; $V_n^*, y_n^*(\lambda^*)$, for all $n \in \mathcal{N}$
Using λ^*, solve the primal Lagrangian relaxed problem (1.26) to get values x_n^*, for all $n \in \mathcal{N}$
for $n \in \mathcal{N}$ **do**
  Sample a random reward vector $r_n \in \mathbb{R}^{|\mathcal{S}_n|}$ from the multivariate Normal(μ_n, Θ_n) distribution
  Sample initial state s_n of arm n from the pmf q_n
end
$\mathbf{s}^0 \leftarrow (s_1, \cdots, s_N)$
Initialize $J_n^{stored} = [\,]$ for all arms $n \in \mathcal{N}$
for *simulation thread* $m \in \{1, \cdots, M\}$ **do**
  Initialize $J_n = 0$ for all arms $n \in \mathcal{N}$

  for *time-step* $t \in \{0, \cdots, T-1\}$ **do**
    if *feasible decision recovery method = One-step lookahead* **then**
      Select an action $\hat{n} \in \underset{n \in \mathcal{N}}{\text{argmax}} \Big\{ \mu_n(s_n) + \alpha \sum_{s_n' \in \mathcal{S}} p_n(s_n'|s_n) V_n^*(s_n') - \alpha V_n^*(s_n) - \Phi^{-1}(\beta_n) \sqrt{\Theta_n(s_n, s_n)} \Big\}$
    end
    if *feasible decision recovery method = One-step lookahead with a single chance constraint* **then**
      $\beta \leftarrow max\left(0.5, \prod_{n \in \mathcal{N}} \beta_n\right)$
      Compute an optimal pmf $\hat{\pi}$ to the SOCP (1.44)
      Select the action $\hat{n}$ by sampling an arm according to $\hat{\pi}$
    end
    if *feasible decision recovery method = Normalized actions* **then**
      Compute pmf $\hat{\pi}$ using equations (1.46)
      Select action $\hat{n}$ by sampling an arm according to $\hat{\pi}$
    end
    if *feasible decision recovery method = myopic* **then**
      Select an action $\hat{n} \in \underset{n \in \mathcal{N}}{\text{argmax}} \{\mu_n(s_n)\}$
    end
    $J_{\hat{n}} \leftarrow J_{\hat{n}} + \alpha^t r_{\hat{n}}(s_{\hat{n}})$
    Sample the next state $s_{\hat{n}}'$ of arm $\hat{n}$ according to transition probabilities $p_{\hat{n}}(s_{\hat{n}}'|s_{\hat{n}})$
    $\mathbf{s}_{t+1} \leftarrow (s_1, \ldots, s_{\hat{n}-1}, s_{\hat{n}}', s_{\hat{n}+1}, \ldots, s_N)$
  end
  for $n \in \mathcal{N}$ **do**
    J_n^{stored}.append(J_n)
  end
end
Estimate sum of percentile of rewards for the implemented decisions as $\sum_{n \in \mathcal{N}} \beta_n$-percentile$(J_n^{\text{stored}})$

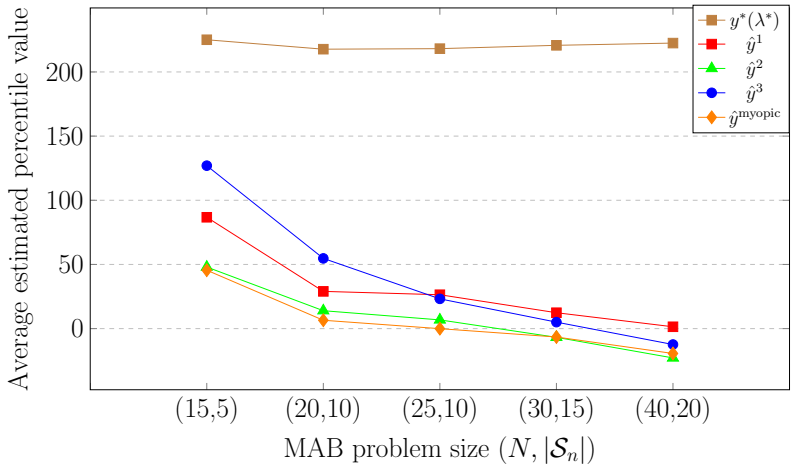


Figure 1.1: A comparison of the performance of four decision recovery methods against the Lagrangian bound, averaged over 20 instances within each problem set.

Figure 1.1 shows that the one-step lookahead method with single chance constraint in Section 1.5.2 and the myopic approach are dominated by the other two methods in all problem sets. As the problem size increases, the gap between the Lagrangian upper bound and the estimate of a lower bound provided by the feasible decisions increases. Furthermore, while the normalized decisions from Section 1.5.3 exhibit the largest percentile values for smaller problem sizes, the one-step lookahead method from Section 1.5.1 outperforms the normalized decisions for larger problems. In the next subsection, we will perform additional tests on smaller instances where we can compute the exact value of the percentile problem to explore the root cause of these patterns.

Unlike the one-step lookahead methods, which involve solving an LP or an SOCP at each time-step, the normalized decisions from Section 1.5.3 do not require solving any optimization problems in run-time. This speeds-up computation. Thus, normalized decisions are good candidates in terms of both performance and run-time.

To test the effect of covariance matrices Θ_n on the performance of the decision recovery methods, we designed 3 experiments with different covariance magnitudes. In particular, entries of matrices A_n described earlier were sampled uniformly from $[-5, 5]$ for low covariance,

$[-20, 20]$ for medium covariance, and $[-40, 40]$ for large covariance. We fixed the discount factor at $\alpha = 0.80$, and focused on a 5-armed bandit problem with 5 states per arm. Figure 1.2 shows the estimated percentile values averaged over 30 instances of these randomly generated $(5, 5)$ MAB problems. The figure suggests that by increasing the covariance magnitude, while the gaps between the Lagrangian bound and the decision recovery methods increase, the dominance relationship between the decision recovery methods does not change. In particular, the normalized decisions method outperforms the other decision methods at all covariance levels. Again, in the next subsection, will perform experiments with smaller instances of MAB problems to study the root cause of this pattern.

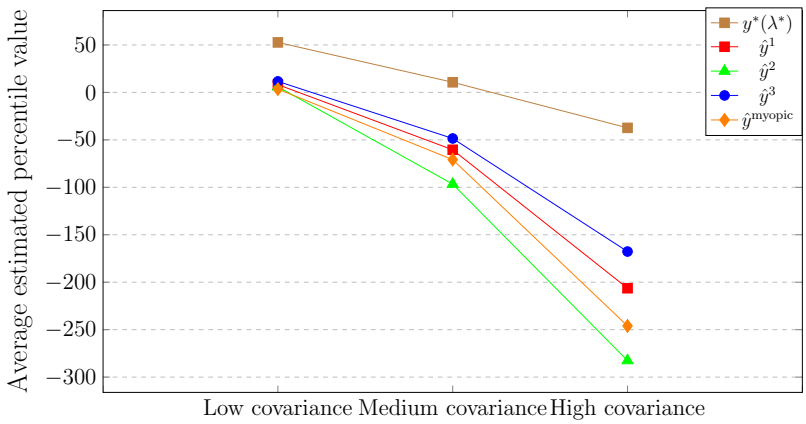


Figure 1.2: A comparison of the performance of four decision recovery methods against the Lagrangian bound, averaged over 30 instances within each problem set. Each problem set includes randomly generated $(5, 5)$ MABs at one specific level of covariance.

1.6.2 Additional tests on smaller MAB instances

We performed additional tests on MAB problems of smaller sizes, wherein the percentile problem (1.16) can be solved exactly to obtain its optimal value y^*. Similarly, decisions prescribed by any one of the three methods in Sections 1.5.1-1.5.3 can be calculated in every state of the MAB problem to construct a corresponding policy. Formula 6.9.3 in [70] can then be applied to calculate the corresponding values of $x(s, a)$ for every state s and feasible

action a for the MAB. By substituting this into problem (1.16) as a feasible solution, we then obtain the value of the policy under consideration. Similarly for the myopic approach. We denote the exact values of the four decision methods obtained this way by $y^1, y^2, y^3, y^{\text{myopic}}$, respectively. The purpose of these experiments is to better-compare the decision recovery methods by eliminating any estimation errors introduced by the simulation process.

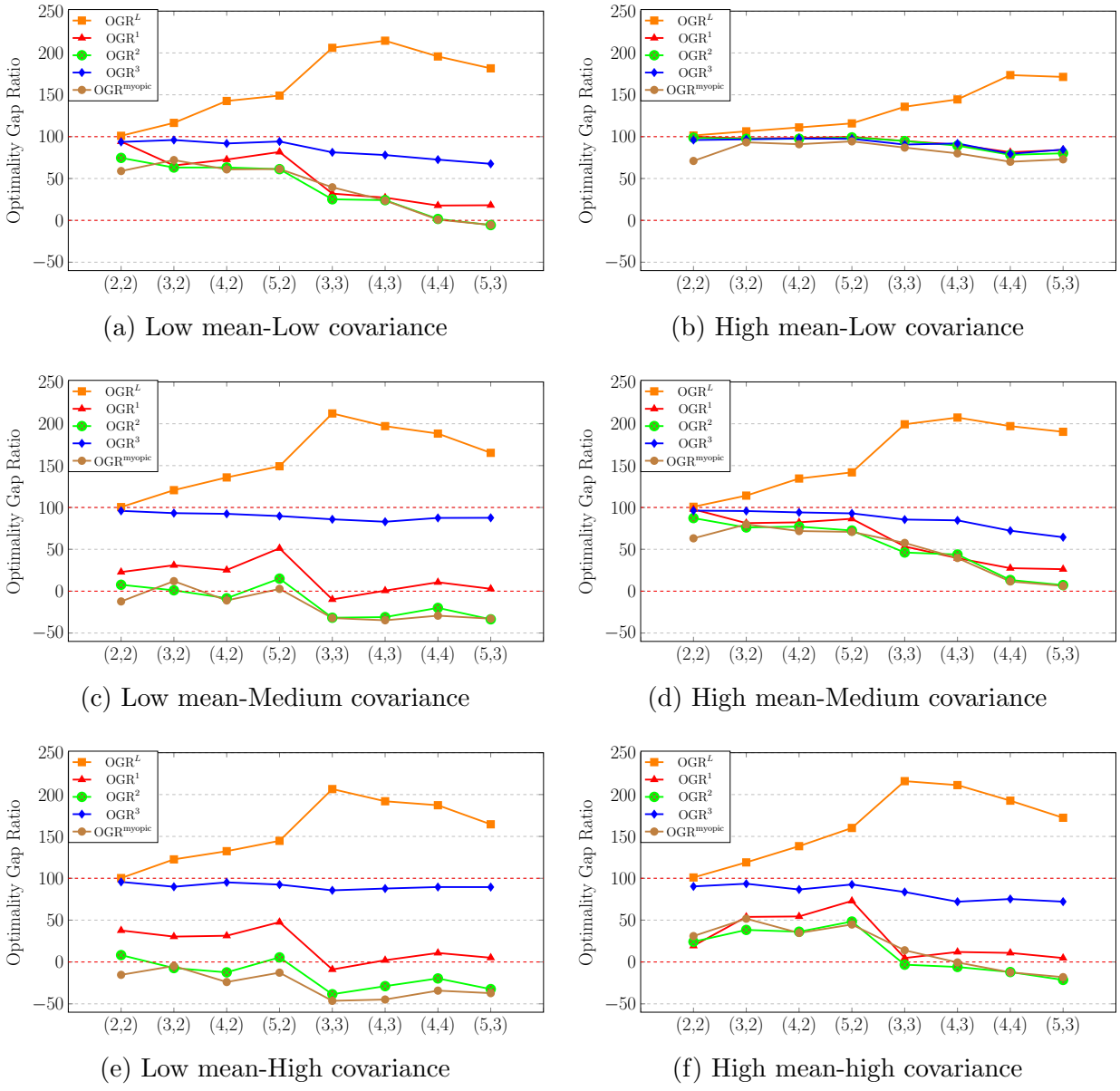
In all experiments, the discount factor was set at $\alpha = 0.80$. To study the effect of the rewards' mean and covariance, 6 groups of experiments were designed using 2 levels of mean and 3 levels for covariance values. In particular, the expected rewards were sampled uniformly from $[1, 20]$ for the low mean case and from $[100, 200]$ for the high mean case. Furthermore, entries of matrices A_n described earlier were sampled from the uniform $[-5, 5]$ distribution for low covariance, uniform $[-20, 20]$ for medium covariance, and uniform $[-40, 40]$ for large covariance.

To evaluate the quality of bounds provided by various methods, we define a metric called Optimality Gap Ratio (OGR) as

$$\text{OGR}^i = \frac{y^i - y^0}{y^* - y^0} \times 100, \;\; i = 1, 2, 3, \text{myopic}$$
$$\text{OGR}^L = \frac{y^*(\lambda^*) - y^0}{y^* - y^0} \times 100.$$

Here, y^0 denotes the objective value of problem (1.16) associated with the policy of playing an arm drawn uniformly at random, in all states. This metric shifts and scales the bounds such that the OGR of y^0 would be 0 and that of y^* would be 100. As such, OGR values close to 100 are better than values far from 100. Also observe that $\text{OGR}^L \geq 100$ because $y^*(\lambda^*) \geq y^*$. Figure 1.3 shows the results of these 6 simulation studies, averaged over 20 instances within each problem set.

Figure 1.3 shows that the normalized decisions method from Section 1.5.3 is the most robust against the rewards' means and covariances, as well as problem sizes. A comparison of sub-figures in the same row suggests that an increase in the magnitude of rewards' mean improves the quality of the lower-bounds provided by the one-step lookahead method and the



(a) Low mean-Low covariance

(b) High mean-Low covariance

(c) Low mean-Medium covariance

(d) High mean-Medium covariance

(e) Low mean-High covariance

(f) High mean-high covariance

Figure 1.3: Optimality Gap Ratio (OGR) values for 6 experiments with various mean and covariance levels. The x-axis identifies distinct problem sets with their sizes ($N, |\mathcal{S}_n|$ for each n). The OGR values were averaged over 20 instances for each problem set.

myopic approach, when the covariance is fixed. On the other hand, comparing sub-figures in the same column reveals that an increase in the rewards' covariance deteriorates the performance of the decision recovery methods, when the rewards' means are fixed. Specifically, the one-step lookahead method with a single chance constraint performs even worse than the baseline method in many problem sets when the mean is low and covariance is either medium or high. One possible explanation for this is that these two methods treat future rewards as being fixed thus ignoring their variability.

Plot 1.3b shows the case where the relative magnitude of the mean values to the covariance values is the highest, among the 6 sub-figures. Therefore, that experiment is the most similar to a deterministic MAB problem. The results in this sub-figure suggest that all three decision methods perform very well and better than the myopic approach, compared to the other five sub-figures. Lastly, the trend of OGR values in each sub-figure suggests that increase in problem size adversely affects all methods. However, the normalized decisions method from Section 1.5.3 appears to be the most robust in this regard as well.

Similar to Figure 1.2 from Section 1.6.1, Figure 1.3 also suggests that the normalized decisions method from Section 1.5.3 outperforms the other methods at all covariance levels. Furthermore, unlike Figure 1.1, which shows better percentile values for the one-step lookahead method compared to the normalized decisions approach, results in Figure 1.3a suggest that the latter method is the most reliable in terms of OGR values when the problem size increases. A possible cause for this can be overestimation of the lower bound provided by the one-step lookahead actions in Figure 1.1 which is caused by simulation error due to insufficient number of simulation threads.

1.7 Extension to weakly coupled MDPs

Weakly coupled MDPs [1, 50] are a generalization of MABs discussed thus far in this chapter. In a weakly coupled MDP, several otherwise independent MDPs, called sub-MDPs, are linked via constraints on their joint actions. The rewards are additively separable and the transition probabilities are multiplicatively separable over the sub-MDPs.

A generic weakly coupled MDP can be described as follows. Suppose there are N sub-MDPs, where $\mathcal{S}_n$ denotes the state space of the n^{th} sub-MDP. For every state $s_n \in \mathcal{S}_n$ the action space of the n^{th} sub-MDP is denoted by $\mathcal{A}_n(s_n)$. When action $a_n \in \mathcal{A}_n(s_n)$ in selected in state $s_n \in \mathcal{S}_n$, the n^{th} sub-MDP transitions into a new state $s'_n \in \mathcal{S}_n$ with probability $p_n(s'_n|s_n, a_n)$ and the decision maker receives a reward $r_n(s_n, a_n)$. The state space of the weakly coupled MDP is $\mathcal{S} = \bigtimes_{n=1}^{N} \mathcal{S}_n$. The action space in state $s \in \mathcal{S}$ is $\mathcal{A}(s) = \bigtimes_{n=1}^{N} \mathcal{A}_n(s_n)$. Transition probabilities and rewards for state $s \in \mathcal{S}$ under action $a \in \mathcal{A}(s)$ are given by $p(s'|s, a) = \prod_{n=1}^{N} p_n(s'_n|s_n, a_n)$ and $r(s, a) = \sum_{n=1}^{N} r_n(s_n, a_n)$. The joint actions of the sub-MDPs are, however, restricted by a set of linking constraints defined as $\sum_{n=1}^{N} D_n^m(s_n, a_n) \leq b_m$ for $m \in \{1, \cdots, M\}$. These can be viewed as a set of M resource consumption constraints, where b_m is the amount of resource m that is available and where $D_n^m(s_n, a_n)$ denotes the amount of resource m consumed by the n^{th} sub-MDP when action $a_n \in \mathcal{A}_n(s_n)$ is selected in state $s_n \in \mathcal{S}_n$. The set of feasible joint actions in state $s \in \mathcal{S}$ is $\bar{\mathcal{A}}(s) = \Big\{a \in \mathcal{A}(s) | \sum_{n=1}^{N} D_n^m(s_n, a_n) \leq b_m, \forall m \in \{1, \cdots, M\}\Big\}$. In the special case of MAB problems, each arm represents a sub-MDP and the linking constraint is that only one arm can be played at a time. We will demonstrate that our results for MAB problems can be extended to the broader class of weakly coupled MDPs. This will expand the applicability of our work to various scheduling, resource allocation, queuing, supply chain, and restless bandit problems [1, 6, 14, 34, 44, 45, 46, 50, 66, 67, 68, 85].

We begin by introducing the sets

$$\mathcal{A} = \bigcup_{s \in \mathcal{S}} \mathcal{A}(s) \qquad \mathcal{S}(a) = \{s \in \mathcal{S} | a \in \mathcal{A}(s)\}$$

$$\bar{\mathcal{A}} = \bigcup_{s \in \mathcal{S}} \bar{\mathcal{A}}(s) \qquad \bar{\mathcal{S}}(a) = \{s \in \mathcal{S} | a \in \bar{\mathcal{A}}(s)\}$$

$$\mathcal{A}_n = \bigcup_{s_n \in \mathcal{S}_n} \mathcal{A}_n(s_n) \qquad \mathcal{S}_n(a_n) = \{s_n \in \mathcal{S}_n | a_n \in \mathcal{A}(s_n)\}.$$

Similar to MAB problems, a randomized policy ρ assigns probabilities $\rho(s, a)$ of selecting feasible joint actions $a \in \bar{\mathcal{A}}(s)$, for each $s \in \mathcal{S}$. The uncountable set of all such randomized

policies is denoted by $\mathcal{P}$. The transition probabilities under a randomized policy $\rho \in \mathcal{P}$ are given by

$$\mathbb{P}_\rho(s'|s) = \sum_{a \in \bar{\mathcal{A}}(s)} \rho(s,a) p(s'|s,a). \tag{1.47}$$

Similarly, the expected immediate reward earned in state $s \in \mathcal{S}$ under a randomized policy ρ is given by

$$r^\rho(s) = \sum_{a \in \bar{\mathcal{A}}(s)} \rho(s,a) r(s,a). \tag{1.48}$$

We let $r^\rho \in \mathbb{R}^{|\mathcal{S}|}$ denote the column vector with entries $r^\rho(s)$, for $s \in \mathcal{S}$. For a feasible action $a \in \bar{\mathcal{A}}$, we will also use $\rho_a \in \mathbb{R}^{|\mathcal{S}|}$ to denote the column vector containing values of $\rho(s,a)$ for all $s \in \bar{\mathcal{S}}(a)$, and 0 for $s \in \mathcal{S} \setminus \bar{\mathcal{S}}(a)$. Similarly, we will use $r^a \in \mathbb{R}^{|\mathcal{S}|}$ to denote the column vector containing values of $r(s,a)$ for $s \in \bar{\mathcal{S}}(a)$, and 0 for $s \in \mathcal{S} \setminus \bar{\mathcal{S}}(a)$. The reward vector r^ρ can be written in matrix form as

$$r^\rho = \sum_{a \in \bar{\mathcal{A}}} \mathrm{diag}(\rho_a) r^a = \sum_{a \in \bar{\mathcal{A}}} \sum_{n \in \mathcal{N}} \mathrm{diag}(\rho_a) r^a_n. \tag{1.49}$$

Here, $r^a_n \in \mathbb{R}^{|\mathcal{S}|}$ contains the rewards earned from the n^{th} sub-MDP upon taking action $a \in \bar{\mathcal{A}}$ for states $s \in \bar{\mathcal{S}}(a)$, and 0 for $s \in \mathcal{S} \setminus \bar{\mathcal{S}}(a)$.

Similar to MAB problems, the expected total discounted reward for a randomized policy $\rho \in \mathcal{P}$ can be written as

$$J^\rho_q = q^T \left[\sum_{t=0}^{\infty} (\alpha \mathbb{P}_\rho)^t \right] r^\rho. \tag{1.50}$$

Substituting for r^ρ from (1.49), we get

$$J^\rho_q = q^T \left[\sum_{t=0}^{\infty} (\alpha \mathbb{P}_\rho)^t \right] \sum_{a \in \bar{\mathcal{A}}} \sum_{n \in \mathcal{N}} \mathrm{diag}(\rho_a) r^a_n = \sum_{n \in \mathcal{N}} q^T \left[\sum_{t=0}^{\infty} (\alpha \mathbb{P}_\rho)^t \right] \sum_{a \in \bar{\mathcal{A}}} \mathrm{diag}(\rho_a) r^a_n = \sum_{n \in \mathcal{N}} J^\rho_{q,n}.$$

Here, $J^{\rho}_{q,n}$ denotes the expected total discounted reward earned by policy ρ from the n^{th} sub-MDP when the initial state distribution is q.

Now suppose that the immediate rewards of all actions $a_n \in \mathcal{A}_n$ in all sub-MDPs follow multivariate Gaussian distributions. That is, $\tilde{r}_{a_n} \sim \text{Normal}\big(\mu_{a_n}, \Theta_{a_n}\big)$, where $\mu_{a_n} \in \mathbb{R}^{|\mathcal{S}_n(a_n)|}$ denotes the mean reward vector and $\Theta_{a_n} \in \mathbb{R}^{|\mathcal{S}_n(a_n)|\times|\mathcal{S}_n(a_n)|}$ denotes the covariance matrix of immediate rewards obtained by selecting action a_n in the n^{th} sub-MDP. Throughout this section, for simplicity of notations and formulations, we consider the case where the immediate rewards across actions in a sub-MDP are independent. However, the results of this section can be easily extended to the case where immediate rewards of different actions in a sub-MDP are correlated.

We now formulate the percentile weakly coupled MDP problem as

$$y^*_w = \max_{y\in\mathbb{R}^N, \rho\in\mathcal{P}} \sum_{n\in\mathcal{N}} y_n \tag{1.51a}$$

$$\mathbb{P}_{\tilde{r}_{a_n}}\Big(J^{\rho}_{q,n} \geq y_n\Big) \geq \beta_n, \qquad \forall n \in \mathcal{N}. \tag{1.51b}$$

This is the counterpart of the percentile MAB problem (1.13). Using the same ideas as in Section 1.3, we can reformulate this percentile problem as the SOCP

$$\max_x \sum_{n\in\mathcal{N}} \left\{ \sum_{s\in\mathcal{S}} \sum_{a\in\bar{\mathcal{A}}(s)} \mu_{a_n}(s_n) x(s,a) - \Phi^{-1}(\beta_n) \sqrt{\sum_{s\in\mathcal{S}} \sum_{s'\in\mathcal{S}} \sum_{a\in\bar{\mathcal{A}}(s)} \sum_{a'\in\bar{\mathcal{A}}(s')|a'_n=a_n} x(s,a)x(s',a')\Theta_{a_n}(s_n,s'_n)} \right\} \tag{1.52a}$$

$$\sum_{a\in\bar{\mathcal{A}}(s)} x(s,a) - \alpha \sum_{s'\in\mathcal{S}} \sum_{a\in\bar{\mathcal{A}}(s')} p(s|s',a)x(s',a) = q(s), \quad \forall s \in \mathcal{S} \tag{1.52b}$$

$$x(s,a) \geq 0, \quad \forall s \in \mathcal{S}, a \in \bar{\mathcal{A}}(s). \tag{1.52c}$$

This is the counterpart of the SOCP (1.16) for the percentile MAB problem. Problem (1.52) has $\sum_{s\in\mathcal{S}} |\bar{\mathcal{A}}(s)|$ variables and $\prod_{n\in\mathcal{N}} |\mathcal{S}_n| + \sum_{s\in\mathcal{S}} |\bar{\mathcal{A}}(s)|$ constraints, where $\sum_{s\in\mathcal{S}} |\bar{\mathcal{A}}(s)|$ can be as high as $\prod_{n\in\mathcal{N}} \sum_{s_n\in\mathcal{S}_n} |\mathcal{A}_n(s_n)|$. Therefore, its size is exponential in N, which renders it computationally

challenging to solve.

To alleviate this curse of dimensionality, we define a Lagrangian relaxation by removing the coupling constraints and adding them to the immediate rewards. The action space in state $s \in \mathcal{S}$ of the deterministic Lagrangian relaxation problem is $\mathcal{A}(s)$. Let λ be any vector of Lagrange multipliers in $\mathbb{R}^M_+$. In the Lagrangian relaxation, the reward earned in state $s \in \mathcal{S}$ by choosing action $a \in \mathcal{A}(s)$ is denoted by $\bar{r}^\lambda(s,a)$ and is given by

$$\bar{r}^\lambda(s,a) = \sum_{n=1}^{N} r_n(s_n,a_n) + \lambda^T(b - \sum_{n=1}^{N} D_n(s_n,a_n)) = \lambda^T b + \sum_{n=1}^{N} \underbrace{\Big(r_n(s_n,a_n) - \lambda^T D_n(s_n,a_n)\Big)}_{r^\lambda(s_n,a_n)}$$
$$= \lambda^T b + r^\lambda(s,a).$$

Here $D_n(s_n,a_n)$ denotes the M-dimensional column vector containing values of $D_n^m(s_n,a_n)$ for $m = 1,\ldots,M$. A relaxed randomized policy ν assigns probabilities $\nu(s,a)$ to joint actions $a \in \mathcal{A}(s)$. The uncountable set of all such randomized policies is denoted $\mathcal{Q}$. The transition probabilities and immediate reward under policy ν in state $s \in \mathcal{S}$ are defined as

$$\mathbb{P}_\nu(s'|s) = \sum_{a \in \mathcal{A}(s)} \nu(s,a) p(s'|s,a) \tag{1.53a}$$

$$r^{\nu,\lambda}(s) = \sum_{a \in \mathcal{A}(s)} \nu(s,a) r^\lambda(s,a) \tag{1.53b}$$

Let $\mathbb{P}_\nu \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ and $r^{\nu,\lambda} \in \mathbb{R}^{|\mathcal{S}|}$ denote the transition probability matrix and immediate reward vector under relaxed randomized policy ν. For each action $a \in \mathcal{A}$, the column vector $r_n^{a,\lambda} \in \mathbb{R}^{|\mathcal{S}|}$ contains the values of $r^\lambda(s_n,a_n)$ for all states $s \in \mathcal{S}(a)$, and 0 for $s \in \mathcal{S} \setminus \mathcal{S}(a)$. Furthermore, let $\nu_a \in \mathbb{R}^{|\mathcal{S}|}$ denote the column vector comprised of values $\nu(s,a)$, for all $s \in \mathcal{S}(a)$, and 0 for $s \in \mathcal{S} \setminus \mathcal{S}(a)$. The expected total discounted reward of policy $\nu \in \mathcal{Q}$ can

be written as

$$J_q^{\nu,\lambda} = \frac{\lambda^T b}{1-\alpha} + q^T \left[\sum_{t=0}^{\infty} (\alpha \mathbb{P}_\nu)^t\right] r^{\nu,\lambda} = \frac{\lambda^T b}{1-\alpha} + q^T \left[\sum_{t=0}^{\infty} (\alpha \mathbb{P}_\nu)^t\right] \sum_{n\in\mathcal{N}} \sum_{a\in\mathcal{A}} \operatorname{diag}(\nu_a) r_n^{a,\lambda}$$
$$= \frac{\lambda^T b}{1-\alpha} + \sum_{n\in\mathcal{N}} q^T \left[\sum_{t=0}^{\infty} (\alpha \mathbb{P}_\nu)^t\right] \sum_{a\in\mathcal{A}} \operatorname{diag}(\nu_a) r_n^{a,\lambda} = \frac{\lambda^T b}{1-\alpha} + \sum_{n\in\mathcal{N}} J_{q,n}^{\nu,\lambda}.$$

Here, $J_{q,n}^{\nu,\lambda}$ denotes the expected total discounted reward from the n^{th} sub-MDP when the initial state distribution is q and the immediate reward vector is $r^{\nu,\lambda}$ under the relaxed randomized policy ν.

The Lagrangian relaxation of the percentile problem (1.51) is then given by

$$y_w^*(\lambda) = \frac{\lambda^T b}{1-\alpha} + \max_{y(\lambda)\in\mathbb{R}^N, \nu\in\mathcal{Q}} \sum_{n\in\mathcal{N}} y_n(\lambda) \tag{1.54a}$$
$$\mathbb{P}_{\tilde{r}_{a_n}}\left(J_{q,n}^{\nu,\lambda} \geq y_n(\lambda)\right) \geq \beta_n \qquad \forall n \in \mathcal{N}. \tag{1.54b}$$

This is the counterpart of problem (1.22). Again, using properties of the multivariate Gaussian distribution, this Lagrangian relaxed percentile problem can be written as the SOCP

$$\frac{\lambda^T b}{1-\alpha} + \max_x \sum_{n\in\mathcal{N}} \Bigg\{ \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}(s)} \big(\mu_{a_n}(s_n) - \lambda^T D_n(s_n, a_n)\big) x(s,a) - \Phi^{-1}(\beta_n) \sqrt{\sum_{s\in\mathcal{S}} \sum_{s'\in\mathcal{S}} \sum_{a\in\mathcal{A}(s)} \sum_{a'\in\mathcal{A}(s')|a'_n=a_n} x(s,a)x(s',a')\Theta_{a_n}(s_n,s'_n)} \Bigg\} \tag{1.55a}$$
$$\sum_{a\in\mathcal{A}(s)} x(s,a) - \alpha \sum_{s'\in\mathcal{S}} \sum_{a\in\mathcal{A}(s')} p(s|s',a)x(s',a) = q(s), \quad \forall s \in \mathcal{S} \tag{1.55b}$$
$$x(s,a) \geq 0, \quad \forall s \in \mathcal{S}, a \in \mathcal{A}(s). \tag{1.55c}$$

This is the counterpart of problem (1.24).

Let ν_n denote a randomized policy for the n^{th} sub-MDP. That is, $\nu_n(s_n, a_n)$ is the probability assigned to $a_n \in \mathcal{A}_n(s_n)$. We denote the uncountable set of all such randomized policies for the n^{th} sub-MDP by $\mathcal{Q}_n$. Let $\mathbb{P}_{\nu_n} \in \mathbb{R}^{|\mathcal{S}_n|\times|\mathcal{S}_n|}$ and $r_n^{\nu_n,\lambda} \in \mathbb{R}^{|\mathcal{S}_n|}$ denote the transi-

tion probability matrix and reward vector induced by policy $\nu_n \in \mathcal{Q}_n$ where their components are defined similar to equations (1.53) for a fixed sub-MDP. In each sub-MDP $n \in \mathcal{N}$, the expected total discounted reward earned in state $s_n \in \mathcal{S}_n$ under the randomized policy ν_n when the initial state distribution is q_n and the immediate reward vector is $r_n^{\nu_n,\lambda}$, is given by

$$J_{q_n}^{\nu_n,\lambda} = \sum_{s_n \in \mathcal{S}_n} q_n(s_n) J^{\nu_n,\lambda}(s_n) = q_n^T \left[\sum_{t=0}^{\infty} (\alpha \mathbb{P}_{\nu_n})^t \right] r_n^{\nu_n,\lambda}.$$

As in Section 1.4, it can be shown that the optimal value in the Lagrangian relaxed percentile problem (1.54) can be expressed as

$$y_w^*(\lambda) = \frac{\lambda^T b}{1-\alpha} + \sum_{n \in \mathcal{N}} y_{w,n}^*(\lambda), \tag{1.56}$$

where $y_{w,n}^*(\lambda)$ is the optimal solution to the following sub-problem

$$y_{w,n}^*(\lambda) = \max_{y_n(\lambda), \nu_n \in \mathcal{Q}_n} \; y_n(\lambda) \tag{1.57a}$$

$$\mathbb{P}_{\tilde{r}_{an}} \left(J_{q_n}^{\nu_n,\lambda} \geq y_n(\lambda) \right) \geq \beta_n. \tag{1.57b}$$

Using the same logic as in the proof of Proposition 1.4.1, we can show that $y_w^*(\lambda) \geq y_w^*$, for any $\lambda \in \mathbb{R}_+^M$. Writing each sub-problem (1.57) in SOCP form and then combining it with the problem of finding the best upper bound yields

$$\min_{\lambda \in \mathbb{R}_+^M} y_w^*(\lambda)$$

$$= \min_{\lambda \in \mathbb{R}^M_+} \left\{ \frac{\lambda^T b}{1-\alpha} + \sum_{n \in \mathcal{N}} \max_{x_n} \sum_{s_n \in \mathcal{S}_n} \sum_{a_n \in \mathcal{A}_n(s_n)} \big(\mu_{a_n}(s_n) - \lambda^T D_n(s_n, a_n)\big) x_n(s_n, a_n) \right.$$
$$\left. - \Phi^{-1}(\beta_n) \sqrt{\sum_{s_n \in \mathcal{S}_n} \sum_{s'_n \in \mathcal{S}_n} \sum_{a_n \in \mathcal{A}_n(s_n) \cap \mathcal{A}_n(s'_n)} x_n(s_n, a_n) x_n(s'_n, a_n) \Theta_{a_n}(s_n, s'_n)} \right.$$
(1.58a)

$$\sum_{a_n \in \mathcal{A}_n(s_n)} x_n(s_n, a_n) - \alpha \sum_{s'_n \in \mathcal{S}_n} \sum_{a_n \in \mathcal{A}_n(s'_n)} p_n(s_n | s'_n, a_n) x_n(s'_n, a_n) = q_n(s_n), \quad \forall s_n \in \mathcal{S}_n$$
(1.58b)

$$x_n(s_n, a_n) \geq 0, \quad \forall s_n \in \mathcal{S}_n, a_n \in \mathcal{A}_n(s_n) \Big\}. \tag{1.58c}$$

This is the counterpart of primal problem (1.35).

We can replace the inner maximization problems in (1.58) above with their duals. Strong duality then implies that problem (1.58) is equivalent to

$$\min_{\substack{\lambda \in \mathbb{R}^M_+ \\ V_n \in \mathbb{R}^{|\mathcal{S}_n|} \\ f_n(a_n) \in \mathbb{R}^{|\mathcal{S}_n(a_n)|}}} \quad \frac{\lambda^T b}{1-\alpha} + \sum_{n \in \mathcal{N}} \sum_{s_n \in \mathcal{S}_n} q_n(s_n) V_n(s_n) \tag{1.59a}$$

$$V_n(s_n) - \alpha \sum_{s'_n \in \mathcal{S}_n} p_n(s'_n | s_n, a_n) V_n(s'_n)$$
$$+ \sum_{s'_n \in \mathcal{S}_n} \Theta^{\frac{1}{2}}_{a_n}(s_n, s'_n) f_n(a_n, s'_n) \geq \mu_{a_n}(s_n) - \lambda^T D_n(s_n, a_n), \quad \forall s_n \in \mathcal{S}_n, a_n \in \mathcal{A}_n(s_n), n \in \mathcal{N}$$
(1.59b)

$$\sqrt{\sum_{a_n \in \mathcal{A}_n} \|f_n(a_n)\|_2^2} \leq \Phi^{-1}(\beta_n), \qquad \forall n \in \mathcal{N}. \tag{1.59c}$$

This is the counterpart of problem (1.36) from Proposition 1.4.3 and it is derived using the same logic. Observe that problem (1.59) has $M + \sum_{n \in \mathcal{N}} |\mathcal{S}_n| + \sum_{n \in \mathcal{N}} \sum_{s_n \in \mathcal{S}_n} |\mathcal{A}_n(s_n)|$ variables and $N + \sum_{n \in \mathcal{N}} \sum_{s_n \in \mathcal{S}_n} |\mathcal{A}_n(s_n)|$ constraints. Thus, the size of (1.59) grows only linearly with the number of sub-MDPs — an exponential reduction compared to (1.52). It remains to recover feasible decisions for the percentile weakly coupled MDP using the solution of the Lagrangian relaxation.

We denote optimal values of decision variables in the Lagrangian relaxed percentile weakly coupled problem (1.59) by $\lambda^* \in \mathbb{R}_+^M$ and $V_n^* \in \mathbb{R}^{|\mathcal{S}_n|}$, for all $n \in \mathcal{N}$. Recall that the size of the set of feasible actions in an MAB problem is simply equal to N, the number of arms. However, in a weakly coupled MDP, the set of feasible actions, which is denoted by $\bar{A}(s)$, for each $s \in \mathcal{S}$, is not readily available. For the generic weakly coupled MDP discussed in this section, finding this set of all feasible actions amounts to finding all feasible solutions to a generalized multi-dimensional knapsack problem — an intractable task in general. Therefore, direct extensions of decision recovery approaches for MABs, which assume that the set of feasible actions is known, does not appear viable. We propose the following alternative methods for recovering feasible decisions in run-time. These methods do not need the set of feasible actions.

1.7.1 Deterministic one-step lookahead with Bellman's equations

When the weakly coupled MDP is in state $s = (s_1, \cdots, s_N)$, this method replaces the immediate rewards of each sub-MDP with their expected values $\mu_n(s_n)$, and the future rewards by $V_n^*(s_n)$. This results in a deterministic weakly coupled MDP, where deterministic feasible action $\hat{a} \in \bar{\mathcal{A}}(s)$ is recovered using the following Bellman's equation

$$\hat{a} = \underset{a \in \bar{\mathcal{A}}(s)}{\operatorname{argmax}} \left[\sum_{n \in \mathcal{N}} \left(\mu_{a_n}(s_n) + \alpha \sum_{s_n \in \mathcal{S}_n} p_n(s_n' | s_n, a_n) V_n^*(s_n') \right) \right]. \tag{1.60}$$

We solve this using the integer program

$$\max_{z} \sum_{n\in\mathcal{N}} \sum_{a_n\in\mathcal{A}_n(s_n)} z_n(a_n) \left[\mu_{a_n}(s_n) + \alpha \sum_{s_n\in\mathcal{S}_n} p_n(s'_n|s_n,a_n)V_n^*(s'_n)\right] \tag{1.61a}$$

$$\sum_{n\in\mathcal{N}} \sum_{a_n\in\mathcal{A}_n(s_n)} D_n^m(s_n,a_n)z_n(a_n) \le b_m, \quad \forall m\in\{1,\cdots,M\} \tag{1.61b}$$

$$\sum_{a_n\in\mathcal{A}_n(s_n)} z_n(a_n) = 1, \quad \forall n\in\mathcal{N} \tag{1.61c}$$

$$z_n(a_n)\in\{0,1\}, \quad \forall a_n\in\mathcal{A}_n(s_n), n\in\mathcal{N}. \tag{1.61d}$$

This is a generalized multidimensional knapsack problem where integer variable $z_n(a_n)$ takes value 1, if action a_n is selected. Constraint (1.61c) restricts the number of actions selected for each sub-MDP to 1, resulting in a deterministic action for the weakly coupled MDP. The knapsack constraints (1.61b) are the coupling constraints that ensure the selected action $\hat{a}$ is in the set of feasible actions $\bar{A}(s)$.

1.7.2 Deterministic one-step lookahead with modified Bellman's equations

This method treats immediate rewards earned from the n^{th} sub-MDP as Normally distributed random variables, but approximates future rewards with V_n^*. It then aims to select a feasible action with the largest sum of β_n-percentiles from all sub-MDPs. This problem can be stated as the chance-constrained program

$$\max_{y\in\mathbb{R}^N, a\in\bar{A}(s)} \sum_{n\in\mathcal{N}} y_n \tag{1.62a}$$

$$\mathbb{P}_{\tilde{r}_{a_n}(s_n)}\left(\tilde{r}_{a_n}(s_n) + \alpha \sum_{s'_n\in\mathcal{S}_n} p_n(s'_n|s_n,a_n)V_n^*(s'_n) \ge y_n\right) \ge \beta_n, \quad \forall n\in\mathcal{N}, \tag{1.62b}$$

which is equivalent to the modified Bellman's equation

$$\hat{a} = \underset{a\in\bar{\mathcal{A}}(s)}{\operatorname{argmax}} \left[\sum_{n\in\mathcal{N}} \left(\mu_{a_n}(s_n) - \Phi^{-1}(\beta_n)\sqrt{\Theta_{a_n}(s_n,s_n)} + \alpha \sum_{s'_n\in\mathcal{S}_n} p_n(s'_n|s_n,a_n)V_n^*(s'_n) \right) \right].$$

This problem can be solved via the integer program

$$\begin{aligned}
\max_{z} & \sum_{n\in\mathcal{N}} \sum_{a_n\in\mathcal{A}_n(s_n)} z_n(a_n) \left[\mu_{a_n}(s_n) - \Phi^{-1}(\beta_n)\sqrt{\Theta_{a_n}(s_n,s_n)} + \alpha \sum_{s_n\in\mathcal{S}_n} p_n(s'_n|s_n,a_n)V_n^*(s'_n) \right] \\
& \sum_{n\in\mathcal{N}} \sum_{a_n\in\mathcal{A}_n(s_n)} D_n^m(s_n,a_n)z_n(a_n) \le b_m, \quad \forall m\in\{1,\cdots,M\} \\
& \sum_{a_n\in\mathcal{A}_n(s_n)} z_n(a_n) = 1, \quad \forall n\in\mathcal{N} \\
& z_n(a_n)\in\{0,1\}, \quad \forall a_n\in\mathcal{A}_n(s_n), n\in\mathcal{N}.
\end{aligned}$$

This is also a generalized multi-dimensional knapsack problem with constraints identical to problem (1.61). The objective function here, however, includes the extra covariance terms in the coefficient of $z_n(a_n)$.

1.7.3 Deterministic actions by estimating the distributions of future rewards

By splitting the expected total discounted reward of a policy ρ into sum of immediate and future rewards, when the initial state is $s = (s_1,\cdots,s_N)$ the weakly coupled percentile problem (1.51) can be rewritten as

$$y_w^* = \max_{y\in\mathbb{R}^N,\rho\in\mathcal{P}} \sum_{n\in\mathcal{N}} y_n \tag{1.63a}$$

$$\mathbb{P}_{\tilde{r}_{a_n}}\left(r_n^\rho(s) + \sum_{s'\in\mathcal{S}} \mathbb{P}_\rho(s'|s)J_n^\rho(s') \ge y_n \right) \ge \beta_n, \qquad \forall n\in\mathcal{N}. \tag{1.63b}$$

Here, $r_n^\rho(s) = \sum_{a\in\bar{\mathcal{A}}} \rho(s,a)r_n(s_n,a_n)$, and $J_n^\rho(s')$ denotes entry s' of $J_{q,n}^\rho$ defined earlier, when q is the unit vector whose s' component is 1. Note that when rewards are random, for any

policy $\rho \in \mathcal{P}$, both $r_n^\rho(s)$ and J_n^ρ are random variables. Unlike the previous two methods, which replace future rewards $J_n^\rho(s')$ by constants, in this method, we approximate their distributions using the solution of the Lagrangian relaxed percentile problem. Let π_n^* denote an optimal policy for the Lagrangian relaxed percentile problem (1.58) for sub-MDP $n \in \mathcal{N}$. Its associated (random) value function is denoted by $\tilde{V}_n^{\pi_n^*}$. We approximate the distribution of $J_n^\rho(s')$ in (1.63b) by the distribution of $\tilde{V}_n^{\pi_n^*}(s'_n)$. Furthermore, we solve the percentile problem (1.63) only over the set of deterministic actions. Therefore, for a feasible deterministic action $a \in \bar{\mathcal{A}}(s)$, the (random) expected future rewards from the n^{th} sub-MDP are replaced by

$$\begin{aligned}
&\sum_{s' \in \mathcal{S}} p(s'|s,a)\tilde{V}_n^{\pi_n^*}(s'_n) = \sum_{s'_1 \in \mathcal{S}_1} \cdots \sum_{s'_N \in \mathcal{S}_N} \prod_{i \in \mathcal{N}} p_i(s'_i|s_i,a_i)\tilde{V}_n^{\pi_n^*}(s'_n) \\
&= \sum_{s'_n \in \mathcal{S}_n} p_n(s'_n|s_n,a_n)\tilde{V}_n^{\pi_n^*}(s'_n) \underbrace{\sum_{s'_1 \in \mathcal{S}_1} \cdots \sum_{s'_{n-1} \in \mathcal{S}_{n-1}} \sum_{s'_{n+1} \in \mathcal{S}_{n+1}} \cdots \sum_{s'_N \in \mathcal{S}_N} \prod_{\substack{i \in \mathcal{N} \\ i \neq n}} p_i(s'_i|s_i,a_i)}_{1} \\
&= \sum_{s'_n \in \mathcal{S}_n} p_n(s'_n|s_n,a_n)\tilde{V}_n^{\pi_n^*}(s'_n).
\end{aligned}$$

This results in problem

$$\max_{y \in \mathbb{R}^N, a \in \bar{A}(s)} \sum_{n \in \mathcal{N}} y_n \tag{1.64a}$$

$$\mathbb{P}_{\tilde{r}_{a_n}} \left(\underbrace{\tilde{r}_{a_n}(s_n) + \alpha \sum_{s'_n \in \mathcal{S}_n} p_n(s'_n|s_n,a_n)\tilde{V}_n^{\pi_n^*}(s'_n)}_{Y_n} \geq y_n \right) \geq \beta_n, \quad \forall n \in \mathcal{N}. \tag{1.64b}$$

The optimal policy for problem (1.58) is given by $\pi_n^*(s_n,a_n) = \frac{x_n^*(s_n,a_n)}{\sum_{a_n \in \mathcal{A}_n(s_n)} x_n^*(s_n,a_n)}$. Recall that the value function of π_n^* can be evaluated using the matrix multiplication $V_n^{\pi_n^*} = (\mathbb{I} - \alpha \mathbb{P}_{\pi_n^*})^{-1} r^{\pi_n^*}$, where $r^{\pi_n^*} \in \mathbb{R}^{|\mathcal{S}_n|}$ and $\mathbb{P}_{\pi_n^*} \in \mathbb{R}^{|\mathcal{S}_n| \times |\mathcal{S}_n|}$ are the immediate rewards and transition probability matrix associated with the (randomized) policy π_n^* defined according to equations (1.49) and (1.47). Since the immediate rewards are Gaussian, the distribution of $\tilde{V}_n^{\pi_n^*} \in \mathbb{R}^{|\mathcal{S}_n|}$

is

$$\tilde{V}_n^{\pi_n^*} \sim \text{Normal}\left(\underbrace{(\mathbb{I}-\alpha\mathbb{P}_{\pi_n^*})^{-1}\mu^{\pi_n^*}}_{E[V_n^*]}, \underbrace{(\mathbb{I}-\alpha\mathbb{P}_{\pi_n^*}^T)^{-1}\Theta^{\pi_n^*}(\mathbb{I}-\alpha\mathbb{P}_{\pi_n^*})^{-1}}_{\text{Cov}[V_n^*]}\right),$$

where $\mu^{\pi_n^*} \in \mathbb{R}^{|\mathcal{S}_n|}$ and $\Theta^{\pi_n^*} \in \mathbb{R}^{|\mathcal{S}_n|\times|\mathcal{S}_n|}$ are defined using the definition of immediate reward of a randomized policy (1.49) as

$$\begin{aligned}\mu^{\pi_n^*} &= \sum_{a_n\in\mathcal{A}_n(s_n)} \text{diag}(\pi_{a_n}^*)\mu_{a_n}\\ \Theta^{\pi_n^*} &= \sum_{a_n\in\mathcal{A}_n(s_n)} \text{diag}(\pi_{a_n}^*)\Theta_{a_n}\text{diag}(\pi_{a_n}^*).\end{aligned}$$

By disregarding the covariance of immediate and expected future rewards, distribution of the random variable Y_n in (1.64b) is approximated as

$$Y_n \sim \text{Normal}\left(\mu_{a_n}(s_n) + \alpha p_n(s_n,a_n)^T E[V_n^*], \Theta_{a_n}(s_n,s_n) + \alpha^2 p_n(s_n,a_n)^T \text{Cov}[V_n^*] p_n(s_n,a_n)\right).$$

Here, $p_n(s_n,a_n)$ denotes the column vector containing values of $p_n(s_n'|s_n,a_n)$ for all $s_n' \in \mathcal{S}_n$. Problem (1.64) can then be reformulated as the integer program

$$\max_z \sum_{n\in\mathcal{N}}\sum_{a_n\in\mathcal{A}_n(s_n)} z_n(a_n)\Big[\mu_{a_n}(s_n) + \alpha p_n(s_n,a_n)^T E[V_n^*] - \Phi^{-1}(\beta_n)\sqrt{\Theta_{a_n}(s_n,s_n) + \alpha^2 p_n(s_n,a_n)^T \text{Cov}[V_n^*] p_n(s_n,a_n)}\Big] \tag{1.65a}$$

$$\sum_{n\in\mathcal{N}}\sum_{a_n\in\mathcal{A}_n(s_n)} D_n^m(s_n,a_n) z_n(a_n) \le b_m, \quad \forall m \in \{1,\cdots,M\} \tag{1.65b}$$

$$\sum_{a_n\in\mathcal{A}_n(s_n)} z_n(a_n) = 1, \quad \forall n \in \mathcal{N} \tag{1.65c}$$

$$z_n(a_n) \in \{0,1\}, \quad \forall a_n \in \mathcal{A}_n(s_n), n \in \mathcal{N}. \tag{1.65d}$$

The feasible region of this problem is the same as those in the previous two sections. The coefficient of $z_n(a_n)$ in the objective function, however, includes extra terms within the covariance expression.

1.7.4 Decomposable randomized one-step lookahead actions

To allow for randomized actions, we first adjust the above method by eliminating constraint (1.65c), which restricts the number of actions a_n with $z_n(a_n) = 1$ to exactly one. For every such feasible solution z and each sub-MDP $n \in \mathcal{N}$, let $\tilde{\mathcal{A}}_n = \{a_n \in \mathcal{A}_n(s_n) | z_n(a_n) = 1\}$. Given that we are only allowed to select one action per sub-MDP, we assume that every feasible z is associated with the randomized action of assigning probability $\frac{1}{|\tilde{\mathcal{A}}_n|}$ to each a_n with $z_n(a_n) = 1$. This implies that the total number of possible joint actions for the weakly coupled MDP resulting from a feasible solution z would be $\prod_{n \in \mathcal{N}} |\tilde{\mathcal{A}}_n|$. To ensure that all of the possible resulting joint actions satisfy coupling constraints, for each resource m, we modify constraint (1.65b) to restrict sum of the largest possible resource consumption over all sub-MDPs. This yields the integer program

$$\max_{z} \sum_{n \in \mathcal{N}} \sum_{a_n \in \mathcal{A}_n(s_n)} z_n(a_n) \Big[\mu_{a_n}(s_n) + \alpha p_n(s_n, a_n)^T E[V_n^*] - \Phi^{-1}(\beta_n) \sqrt{\Theta_{a_n}(s_n, s_n) + \alpha^2 p_n(s_n, a_n)^T \mathrm{Cov}[V_n^*] p_n(s_n, a_n)} \Big] \tag{1.66a}$$

$$\sum_{n \in \mathcal{N}} \max_{a_n \in \tilde{\mathcal{A}}_n} \left[D_n^m(s_n, a_n) \right] \leq b_m, \quad \forall m \in \{1, \cdots, M\} \tag{1.66b}$$

$$z_n(a_n) \in \{0, 1\}, \quad \forall a_n \in \mathcal{A}_n(s_n), n \in \mathcal{N}. \tag{1.66c}$$

To write (1.66b) as a linear constraint, we introduce variables $g_n^m(s_n)$ for all $n \in \mathcal{N}$ and $m \in \{1, \cdots, M\}$. Constraint (1.66b) is then replaced by

$$\sum_{n \in \mathcal{N}} g_n^m(s_n) \leq b_m, \quad \forall m \in \{1, \cdots, M\}$$
$$D_n^m(s_n, a_n) z_n(a_n) \leq g_n^m(s_n), \quad \forall a_n \in \mathcal{A}_n(s_n).$$

1.7.5 Numerical results

In this section, we test performance of the above four decision recovery methods. We did this for 4 problem sets, each comprised of 20 randomly generated instances. In all instances, the discount factor was fixed at $\alpha = 0.8$. Each simulation included 1000 threads. Each thread included 40 time-steps. Percentile values of various decision recovery methods were estimated via a simulation process similar to Algorithm 1.

In all our experiments, we assumed that $|\mathcal{S}_n|$ is invariant across sub-MDPs n. We also assumed that $\mathcal{A}_n(s_n) = \mathcal{A}_n$, for all $s_n \in \mathcal{S}_n$. In the n^{th} sub-MDPs, the initial state distribution q_n, the transition probabilities, the expected rewards μ_{a_n}, and the covariance matrix Θ_{a_n} were generated exactly as in Section 1.6. Each problem set is represented by a tuple (N,$|\mathcal{S}_n|$, $|\mathcal{A}_n|$, M). The coupling constraints were defined as $D_n^m(s_n, a_n) = d_n^m \times a_n$, where actions a_n are assumed to be integers from the set $\mathcal{A}_n = \{0, \cdots, |\mathcal{A}_n| - 1\}$. The left hand side parameters d_n^m were sampled from a uniform distribution over the interval [0,20]. Following Chu and Beasley [30], the right hand side parameters b_m were set to $\sum_{n \in \mathcal{N}} d_n^m \times (|\mathcal{A}_n| - 1) \times \epsilon$, where ϵ is called the tightness ratio. Tightness ratios were selected from $\epsilon \in \{0.25, 0.50, 0.75\}$. Results are presented in Table 1.1.

Problem Set									
N	$\lvert\mathcal{S}_n\rvert$	$\lvert\mathcal{A}_n\rvert$	M	ϵ	y^L	$\hat{y}^1$	$\hat{y}^2$	$\hat{y}^3$	$\hat{y}^4$
3	3	3	2	0.25	155.693	106.785	107.483	106.989	78.844
				0.5	169.597	134.036	134.297	135.24	88.359
				0.75	173.184	148.828	147.59	147.86	89.254
4	4	4	3	0.25	214.919	150.485	149.763	151.594	96.776
				0.5	239.816	191.363	190.189	193.556	100.423
				0.75	242.04	202.84	202.11	205.445	107.125
6	4	4	3	0.25	335.936	260.099	259.701	262.594	152.963
				0.5	366.52	305.818	305.189	308.005	165.877
				0.75	367.701	312.924	314.041	316.28	164.124
8	5	5	3	0.25	472.338	376.338	377.77	380.711	209.301
				0.5	521.96	433.111	435.307	438.362	218.513
				0.75	522.977	439.38	443.607	448.047	227.233

Table 1.1: Average estimated percentile values of four decision recovery methods from Sections 1.7.1-1.7.4. These are denoted by $\hat{y}^1$, $\hat{y}^2$, $\hat{y}^3$, and $\hat{y}^4$, respectively. The notation y^L represents the average optimal value of the Lagrangian relaxed percentile problem (1.59). All averages were calculated over 20 randomly generated instances for each one of the 4 problem sets.

The table shows that $\hat{y}^4$ is dominated by $\hat{y}^1, \hat{y}^2, \hat{y}^3$, in all problem sets. While the three deterministic decisions from Sections 1.7.1-1.7.3 seem to be comparable in terms of percentile values, a closer inspection reveals that $\hat{y}^3$ dominates $\hat{y}^1, \hat{y}^2$, as the problem size grows. Furthermore, increasing the tightness ratio increases the percentile values for all methods in the same problem set. This is consistent with the definition of the tightness ratio — higher tightness ratios allows for larger sets of feasible actions. The tightness ratio does not appear to have a big impact on the relative performance of the four decision approaches. In conclusion, given that the three deterministic decision methods have comparable computation

times, the method from Section 1.7.3 appears to be a good candidate.

1.8 Conclusions

In this chapter, we studied the classical multi-armed bandit problem with multivariate Gaussian rewards. We pursued a percentile optimization approach wherein the goal was to maximize the sum of individual β_n-percentiles of expected total discounted infinite-horizon rewards earned by playing distinct arms. This problem is equivalent to an SOCP, whose size is exponential in the number of arms. We therefore proposed a Lagrangian relaxation method to break the resulting curse-of-dimensionality. We showed that the optimal value of the relaxed problem provides an upper bound on the exact percentile problem. Moreover, we demonstrated that the relaxed problem can be reformulated as another SOCP using strong duality. The size of this SOCP is linear in the number of arms, and is thus exponentially smaller than the original exact SOCP. Two one-step lookahead methods as well as a normalization method were proposed to recover decisions in run-time using offline solution of this Lagrangian relaxed SOCP. The performance of these methods was compared via numerical experiments. These experiments showed that the normalization method achieves higher percentile values on average. It also maintains this performance advantage over a range of mean vectors and covariance matrices for the multivariate Gaussian rewards. As such, it is the most robust among the three methods. We then extended the Lagrangian relaxation approach to the more general class of weakly coupled MDPs. We proposed three deterministic methods and one randomized method to recover feasible decisions in run-time using an off-line solution of the relaxed problem. Numerical experiments showed that while the three deterministic methods perform similarly, the third method outperforms the other two in terms of percentile values for larger problems. Applying these decision recovery methods to weakly coupled MDPs that arise in problems such as scheduling, resource allocation, and restless bandits, will provide an interesting direction for future research.

Chapter 2

INVERSE MARKOV DECISION PROCESSES WITH UNKNOWN TRANSITION PROBABILITIES

2.1 Introduction and literature review

Conventional optimization problems aim to find reward-maximizing values of decision variables given the values of a model's parameters. This is called forward optimization. Inverse optimization, on the other hand, refers to the problem of determining the model's parameters using given values of decision variables [2, 52, 53]. Two approaches are typically pursued: search for parameters that are, in some sense, close to nominal "guess" values [2, 53]; or search for parameters that minimize some measure of sub-optimality for the given solution [13, 24]. The inverse optimization philosophy is applicable in two types of situations. The first is where it is difficult or impossible to estimate parameter values, but a particular decision is implemented in practice nevertheless. The practitioner then wonders about the "implied" parameter values. This is often the case in medicine, for instance, where parameters that characterize a patient's response to treatment or the utility of various health conditions are unknown. Nevertheless, doctors routinely make treatment choices and may be interested in imputing these parameter values [35, 40]. Such implied parameter values can help them test the validity of their decisions and perform comparative analyses. The second is where a planner wishes a desired solution to materialize, and is wondering what values of model parameters would achieve it. This is the case in transportation congestion pricing, for instance, where planners wish to see a particular traffic pattern and need to determine what tolling mechanism would attain it.

Inverse optimization was studied first by geophysicists [62, 64, 75, 86]. Other applications of inverse optimization include transportation [32, 33], demand management [23],

auctions [8], production planning [77], healthcare [7, 27, 35], and finance [12]. The mathematical programming community has worked on inverse problems for shortest paths [20, 21], finite-dimensional linear programs (LPs) [2, 25, 76], infinite-dimensional LPs [39, 41], conic programs [53], combinatorial optimization [52], network optimization [65, 87, 88], multi-objective optimization [26, 24, 37, 60, 71], integer programs [72], mixed-integer programs [81], separable convex programs with linear constraints [89], polynomial optimization [55], and machine learning/statistics [28, 74].

In this chapter, we study inverse optimization in the context of infinite-horizon Markov decision processes (MDPs) [70]. MDPs are commonly used to model sequential decision-making problems under uncertainty in many areas such as revenue management, healthcare, robot navigation, supply chain management, transportation, asset management, inventory control, and queuing systems. See [16] for an extensive list of classic and modern applications.

MDPs can be described as follows. At each time-step $t \in \{0, 1, 2, \cdots\}$, a system is in some state $s \in \mathcal{S}$, where $\mathcal{S}$ is a finite set. After observing this state, the decision-maker chooses an action $a \in \mathcal{A}$, where $\mathcal{A}$ is a finite set. The decision-maker then receives an immediate reward $r(s, a)$. The system then transitions into a new state $s' \in \mathcal{S}$ with probability $p(s'|s, a)$. For simplicity, we focus on the case where immediate rewards $r(s, a)$ do not depend on transition probabilities $p(s'|s, a)$. However, all our results can be extended if $r(s, a) = \sum_{s' \in \mathcal{S}} p(s'|s, a) r(s'|s, a)$, where $r(s'|s, a)$ is the immediate reward earned upon choosing action a in state s and the system transitions to state s'. A policy π is a mapping from $\mathcal{S}$ to $\mathcal{A}$ such that $\pi(s) \in \mathcal{A}$ is the action prescribed in state s. The expected total infinite-horizon discounted reward of a policy π starting in state $s \in \mathcal{S}$ is defined as

$$J^\pi(s) = \mathbb{E}\left(\sum_{t=0}^{\infty} \alpha^t r(\mathbf{S}_t, \pi(\mathbf{S}_t)) \big| \mathbf{S}_0 = s\right), \tag{2.1}$$

where $0 < \alpha < 1$ is the discount factor. Here, $\mathbf{S}_t$ is the (stochastic) state at time-step $t = 0, 1, 2, \cdots$, induced by policy π and initial state s. The vector $J^\pi \in \Re^{|\mathcal{S}|}$ with components $J^\pi(s)$ is often called the value function of policy π. The finite set of all policies is denoted by

Π. The decision-maker's goal is to find a policy that (simultaneously, for all initial states) maximizes the expected total infinite-horizon discounted reward. That is, the decision-maker wishes to solve

$$\pi^* \in \underset{\pi \in \Pi}{\operatorname{argmax}} \ \{J^\pi(s), \ \forall s \in \mathcal{S}\}. \tag{2.2}$$

The corresponding maximum expected total discounted reward is denoted by $J^*(s) = J^{\pi^*}(s)$, for all $s \in \mathcal{S}$. The vector $J^* \in \Re^{|\mathcal{S}|}$ is called the optimal value function.

There are three standard methods for solving infinite horizon MDPs: value iteration, policy iteration, and LP [70]. All of them essentially focus on finding an optimal value function that satisfies Bellman's equations of optimality, and recovering a policy that attains this optimal value function [70]. In particular, J^* is the optimal value function if, and only if, it is the (unique) solution of Bellman's equations of optimality

$$J^*(s) = \max_{a \in \mathcal{A}} \left\{ r(s,a) + \alpha \sum_{s' \in \mathcal{S}} p(s'|s,a) J^*(s') \right\}, \quad \forall s \in \mathcal{S}. \tag{2.3}$$

An action that achieves the above maximum in state s is optimal in that state. Similarly, J^π is the value function for policy π if, and only if, it is the (unique) solution of the system

$$J^\pi(s) = r\big(s, \pi(s)\big) + \alpha \sum_{s' \in \mathcal{S}} p\big(s'|s, \pi(s)\big) J^\pi(s'), \quad \forall s \in \mathcal{S}. \tag{2.4}$$

These are called Bellman's equations of policy evaluation.

There is only one paper in the literature that focuses on inverse optimization in MDPs. Erkin et al. [35] considered a healthcare application and studied the problem of imputing reward values $r(s,a)$ that would make a given policy optimal. This problem is not too difficult to solve, for two main reasons. First, it turns out that it is always possible to find rewards that make any given policy optimal. For instance, setting all rewards to 0 suffices. In this sense, the inverse problem is always feasible. The problem is still nontrivial because the decision-maker is usually interested in finding rewards that are close to a nominal guess. Fortunately,

the inverse optimization framework for finite-dimensional LPs from Ahuja and Orlin [2] can be applied in a straightforward manner to impute rewards in MDPs. Specifically, rewards $r(s, a)$ form the objective function coefficients in the LP formulation of the MDP. Ahuja and Orlin [2] proposed a simple approach that uses complementary slackness to recover objective function coefficients that make a given LP solution optimal. Their key conclusion was that if the decision-maker utilizes the ℓ_1 or the ℓ_∞ norm to evaluate closeness of rewards, the resulting inverse problem is also an LP. Erkin et al. [35] applied this approach to reformulate their inverse MDP problem as an LP.

Unlike Erkin et al. [35], we investigate the inverse problem of imputing transition probabilities $p(s'|s, a)$ in this study. As a motivating example, consider a queuing system whereby a planner is executing a particular routing or admission control policy. The planner wants to impute the characteristics of the stochastic arrival process that determines the transition probabilities. Similarly, imagine an inventory manager who is implementing an ordering policy. The manager wants to impute characteristics of the demand process that determines the transition probabilities. Finally, in healthcare, our setup applies to a doctor who is implementing a particular treatment protocol, and wants to impute the probabilities according to which health conditions evolve in response to treatment.

We utilize the aforementioned LP formulation of a forward MDP to model the inverse problem. In the forward LP formulation, transition probabilities appear on the left hand side as constraint coefficients. As in Ahuja and Orlin [2], we apply duality to formulate the inverse problem. It is known that when such an approach is applied to a forward LP with unknown left hand side coefficients, the resulting inverse problem is a computationally difficult, nonconvex, bilinear program [25]. Consequently, there are only a few papers that attempt to tackle inverse LPs with unknown left hand side coefficients. These papers utilize application-specific problem-structure to remove bilinearities from the inverse formulation. For instance, Birge et al. [15] utilized inverse optimization to recover market structure using market outcomes. Brucker and Shakhlevich [19] applied inverse optimization to the maximum lateness problem to recover the values of processing times or due dates. Chan and

Kaw [25] studied inverse LPs with unknown left hand side coefficients as well as unknown cost coefficients. They proposed two approaches to solve the inverse problem either in closed form or by tackling a polynomial number (in the number of constraints) of linear or convex programs. In another study, Ghobadi and Mahmoudzadeh [42] investigated the problem of recovering both the left hand side coefficients and the right hand side bounds such that a given set of solutions would be feasible while making a preferred subset of solutions optimal. Unfortunately, as we shall explain later, neither the Chan and Kaw [25] method nor the Ghobadi and Mahmoudzadeh [42] method is applicable in our context.

Given these gaps in the literature, the research objective of this chapter is to impute unknown transition probabilities $p(s'|s,a)$ by applying MDP theory and convex optimization methods to exactly or approximately solve the relevant bilinear problem. In fact, we study two variants of the inverse MDP problem. In the first variant (Section 2.2), the decision-maker is interested in finding transition probabilities and a corresponding policy whose value function equals a given target. We show that the resulting inverse MDP problem reduces to a linear feasibility problem. We also formulate another linear feasibility problem to find transitions probabilities that render the target value function optimal. We thus reach the conclusion that this first variant is easy to tackle. This is achieved entirely by relying on the aforementioned Bellman's equations. In the second variant (Section 2.3), we study the case when a policy is given. The decision-maker is interested in identifying transition probabilities that make this policy optimal. We apply two heuristics that are popular for Quadratically Constrained Quadratic Programs (QCQPs), since the resulting problem is nonconvex bilinear. Note that bilinear programs are special cases of QCQPs. The heuristics are called Convex-Concave Procedure (CCP) [56] (Section 2.3.1) and Sequential Linear Programming (SLP) [49] (Section 2.3.2). We compare them against a classic exact global optimization algorithm proposed by Floudas and Visweswaran [36]. This method is known as GOP (Global Optimization Procedure) and is based on generalized Bender's decomposition. For inverse MDP problems, this method turned out to be too slow to be useful in practice, and is only employed as a benchmark in this chapter. Computational experiments are presented in Sec-

tion 2.4. The overall conclusion of these experiments is that SLP appears to be a sound heuristic for inverse MDPs with unknown transition probabilities, and it seems to be more effective than CCP. Directions for future research are outlined in Section 2.5. We assume throughout this chapter that rewards $r(s, a)$ are known, because inverse problems where both the transition probabilities and the rewards are unknown turn out to be vacuously trivial.

2.2 Inverse problem with a given value function

We first consider in Section 2.2.1 the following problem: given any $\hat{V} \in \Re^{|\mathcal{S}|}$, does there exist transition probabilities $p \in \Re^{|\mathcal{S}|\times|\mathcal{S}|\times|\mathcal{A}|}$ and a corresponding policy $\pi \in \Pi$ such that $J^{\pi}(s) = \hat{V}(s)$, for all $s \in \mathcal{S}$. Here, $\hat{V}$ can be viewed as a target value function. The decision-maker also wishes to find such transition probabilities and policy, if they exist.

First note that such a policy may not exist. As a trivial example, consider an MDP with a single state and a single action [and thus transition probability is simply equal to 1]. Suppose $r(s, a) = 1$ for the single state s and the single action a. This problem includes a single policy — the one that executes action a in state s. It is easy to see that its value is $1/(1 - \alpha)$. Consequently, there is no policy whose value equals 1.

If the requisite policy and transition probabilities do exist, the next natural question is whether $\hat{V}$ can be the optimal value function for any combination of transition probabilities and policy. Furthermore, how can the decision-maker find such a combination? These questions are answered in Section 2.2.2.

A high-level summary of the complete procedure is provided in Figure 2.1 as a quick reference.

2.2.1 Enforcing $\hat{V} \in \Re^{|\mathcal{S}|}$ to be the value function of some policy

Given a $\hat{V} \in \Re^{|\mathcal{S}|}$, we study the question of whether or not there exist transition probabilities $p \in \Re^{|\mathcal{S}|\times|\mathcal{S}|\times|\mathcal{A}|}$ and policy π such that $\hat{V}$ is the value function for this policy. The next proposition provides a necessary and sufficient condition for this.

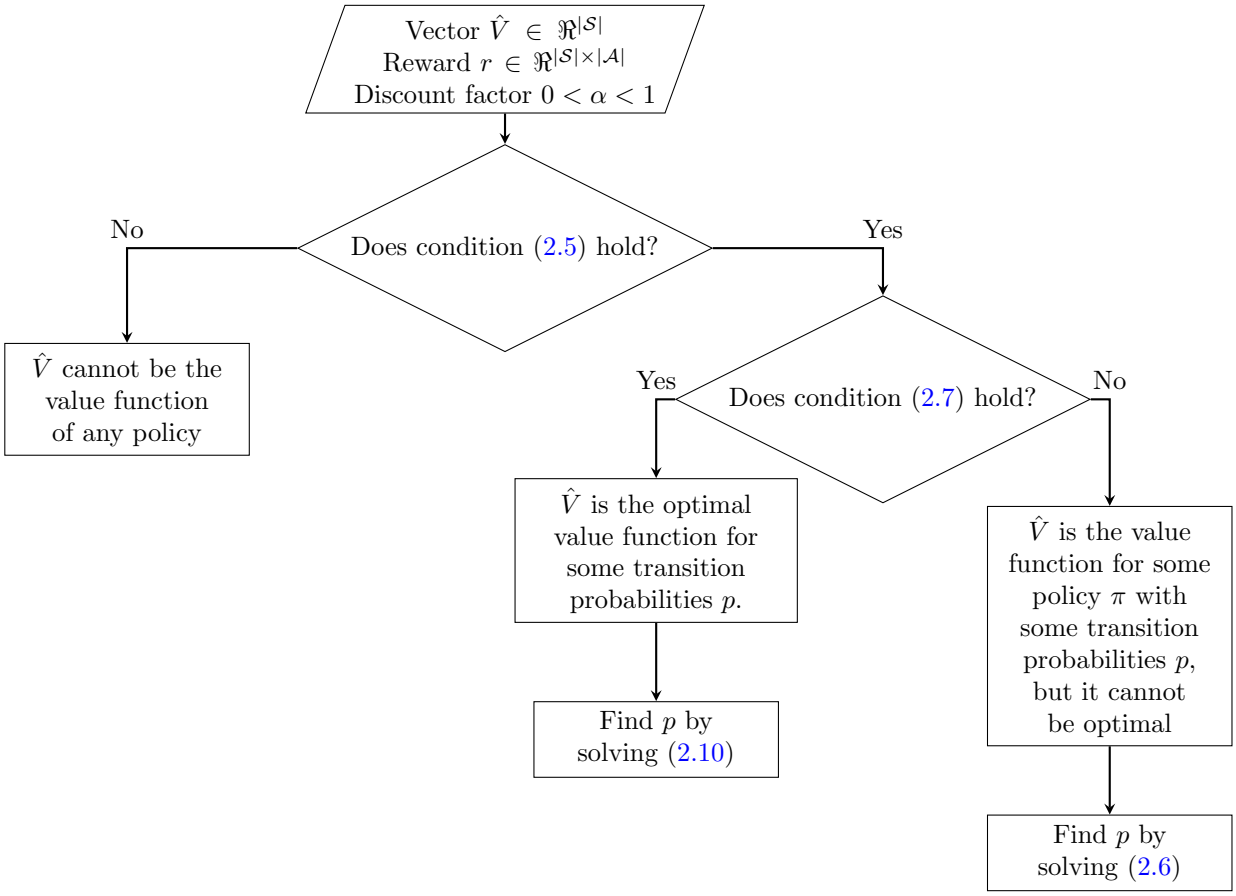


Figure 2.1: Flow chart for recovering transition probabilities when $\hat{V}$ is given.

Proposition 2.2.1. *There exist policy π and transition probabilities $p \in \Re^{|\mathcal{S}|\times|\mathcal{S}|\times|\mathcal{A}|}$ such that $\hat{V} = J^\pi$ if, and only if, for every state $s \in \mathcal{S}$, there is an action $a \in \mathcal{A}$ such that*

$$\hat{V}(s) - \alpha \max_{s' \in \mathcal{S}} \hat{V}(s') \leq r(s,a) \leq \hat{V}(s) - \alpha \min_{s' \in \mathcal{S}} \hat{V}(s'). \tag{2.5}$$

Proof. For the "only if" part, suppose that either (i) there exists a state s such that $\hat{V}(s) - \alpha \max_{s' \in \mathcal{S}} \hat{V}(s') > r(s,a)$ for all $a \in \mathcal{A}$, or (ii) there exists a state s such that $r(s,a) > \hat{V}(s) - \alpha \min_{s' \in \mathcal{S}} \hat{V}(s')$ for all $a \in \mathcal{A}$. In the first case, consider any arbitrary policy π and any

corresponding arbitrary transition pmf $p\big(\,\cdot\,|s,\pi(s)\big)$. We have,

$$r\big(s,\pi(s)\big)+\alpha\sum_{s'\in\mathcal{S}}p\big(s'|s,\pi(s)\big)\hat{V}(s')\leq r\big(s,\pi(s)\big)+\alpha\max_{s'\in\mathcal{S}}\hat{V}(s')\cancelto{1}{\sum_{s'\in\mathcal{S}}p\big(s'|s,\pi(s)\big)}$$
$$<\hat{V}(s).$$

Thus, $\hat{V}$ does not satisfy Bellman's evaluation equations (2.4). It therefore cannot be the value function of π. In the second case, consider any arbitrary policy π and any corresponding arbitrary transition pmf $p(\cdot|s,\pi(s))$. We have,

$$r\big(s,\pi(s)\big)+\alpha\sum_{s'\in\mathcal{S}}p\big(s'|s,\pi(s)\big)\hat{V}(s')\geq r\big(s,\pi(s)\big)+\alpha\min_{s'\in\mathcal{S}}\hat{V}(s')\cancelto{1}{\sum_{s'\in\mathcal{S}}p\big(s'|s,\pi(s)\big)}$$
$$>\hat{V}(s).$$

Thus, $\hat{V}$ does not satisfy Bellman's evaluation equations (2.4). It therefore cannot be the value function of π. Since π was arbitrary in both cases, the proof of the "only if" part is complete.

Now, for the "if" part, let $a^*(s)$, for each state $s\in\mathcal{S}$, denote an action that satisfies (2.5). For each $s\in\mathcal{S}$, define $f\big(p(\cdot|s,a^*(s))\big)=r(s,a^*(s))+\alpha\sum_{s'\in\mathcal{S}}p(s'|s,a^*(s))\hat{V}(s')$ as a real-valued function of the pmf $p(\cdot|s,a^*(s))$ over $\Delta^{|\mathcal{S}|}$. Here, $\Delta^{|\mathcal{S}|}$ is the usual probability simplex in $\Re^{|\mathcal{S}|}$, which is convex and hence path-connected. This f is a linear and hence continuous function. Let $\tilde{s}$ be a state such that $\hat{V}(\tilde{s})=\max_{s'\in\mathcal{S}}\hat{V}(s')$. Let $\tilde{p}(\cdot|s,a^*(s))$ be a degenerate pmf such that $\tilde{p}(\tilde{s}|s,a^*(s))=1$ and $\tilde{p}(s'|s,a^*(s))=0$ for all $s'\neq\tilde{s}$. Then,

$$f\big(\tilde{p}(\cdot|s,a^*(s))\big)=r(s,a^*(s))+\alpha\sum_{s'\in\mathcal{S}}\tilde{p}(s'|s,a^*(s))\hat{V}(s')=r(s,a^*(s))+\alpha\hat{V}(\tilde{s})$$
$$=r(s,a^*(s))+\alpha\max_{s'\in\mathcal{S}}\hat{V}(s')\geq\hat{V}(s).$$

Similarly, let $\hat{s}$ be a state such that $\hat{V}(\hat{s})=\min_{s'\in\mathcal{S}}\hat{V}(s')$. Let $\hat{p}(\cdot|s,a^*(s))$ be a degenerate pmf

such that $\hat{p}(\hat{s}|s, a^*(s)) = 1$ and $\hat{p}(s'|s, a^*(s)) = 0$ for all $s' \neq \hat{s}$. Then,

$$f\big(\hat{p}(\cdot|s, a^*(s))\big) = r(s, a^*(s)) + \alpha \sum_{s' \in \mathcal{S}} \hat{p}(s'|s, a^*(s))\hat{V}(s') = r(s, a^*(s)) + \alpha \hat{V}(\hat{s})$$
$$= r(s, a^*(s)) + \alpha \min_{s' \in \mathcal{S}} \hat{V}(s') \leq \hat{V}(s).$$

Then, by the intermediate value theorem, there exists a pmf $p^*(\cdot|s, a^*(s))$ such that $f\big(p^*(\cdot|s, a^*(s))\big) = \hat{V}(s)$. Specifically, $\hat{V}(s)$ satisfies Bellman's evaluation equations (2.4) for state s, action $a^*(s)$, and pmf $p^*(\cdot|s, a^*(s))$. In other words, there exist transition probabilities such that $\hat{V}$ is the value function for policy π with $\pi(s) = a^*(s)$. This completes the proof. □

Now suppose that $\hat{V}$ is such that condition (2.5) holds. In particular, as in the proof of the above proposition, let $a^*(s)$, for each $s \in \mathcal{S}$, be any action that satisfies this condition. Then, it only remains to find transition probabilities for which the policy defined by actions $a^*(s)$ satisfies Bellman's evaluation equations (2.4) with $\hat{V}$. This can be done simply by solving the feasibility LP.

$$\min_{p \in \Re^{|\mathcal{S}| \times |\mathcal{S}| \times |\mathcal{A}|}} 0$$
$$\hat{V}(s) - \alpha \sum_{s' \in \mathcal{S}} p\big(s'|s, a^*(s)\big)\hat{V}(s') = r\big(s, a^*(s)\big), \quad \forall s \in \mathcal{S} \tag{2.6a}$$
$$\sum_{s' \in \mathcal{S}} p\big(s'|s, a^*(s)\big) = 1, \quad \forall s \in \mathcal{S} \tag{2.6b}$$
$$p\big(s'|s, a^*(s)\big) \geq 0, \quad \forall s, s' \in \mathcal{S} \tag{2.6c}$$
$$\sum_{s' \in \mathcal{S}} p\big(s'|s, a\big) = 1, \quad \forall s \in \mathcal{S},\ a^*(s) \neq a \in \mathcal{A} \tag{2.6d}$$
$$p\big(s'|s, a\big) \geq 0, \quad \forall s, s' \in \mathcal{S},\ a^*(s) \neq a \in \mathcal{A}. \tag{2.6e}$$

Note that pmfs $p(\cdot|s, a)$, for each state $s \in \mathcal{S}$ and actions $a \neq a^*(s)$, can be set arbitrarily as they do not affect either the state evolution or the value function. Constraints (2.6d)-(2.6e) capture this.

2.2.2 *Enforcing $\hat{V} \in \Re^{|\mathcal{S}|}$ to be the optimal value function*

Now the next question is whether or not there exist a policy π and transition probabilities p for which $\hat{V}$ is the optimal value function. The next proposition provides a necessary and sufficient condition for this.

Proposition 2.2.2. *There exist policy π and transition probabilities $p \in \Re^{|\mathcal{S}|\times|\mathcal{S}|\times|\mathcal{A}|}$ such that $\hat{V} = J^*$ if, and only if, condition (2.5) in Proposition 2.2.1 holds AND*

$$\hat{V}(s) - \alpha \min_{s' \in \mathcal{S}} \hat{V}(s') \geq r\big(s,a\big), \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}. \tag{2.7}$$

Proof. First consider the "only if" part of the conclusion. Proposition 2.2.1 asserts that if condition (2.5) does not hold, then $\hat{V}$ cannot even equal the value function of any policy for any transition probabilities. Therefore, condition (2.5) is necessary for $\hat{V}$ to be the optimal value function for some policy and some transition probabilities. We now show that condition (2.7) is also necessary. So suppose that condition (2.7) does not hold. In other words, suppose $\hat{V}(s) - \alpha \min_{s' \in \mathcal{S}} \hat{V}(s') < r\big(s,a\big)$, for some $(s,a) \in \mathcal{S} \times \mathcal{A}$. Consider any such state-action pair (s,a). Then, for any transition probabilities p, we have,

$$r(s,a) + \alpha \sum_{s' \in \mathcal{S}} p(s'|s,a)\hat{V}(s') \geq r(s,a) + \alpha \min_{s' \in \mathcal{S}} \hat{V}(s') \sum_{s' \in \mathcal{S}} p(s'|s,a) \overset{1}{\cancel{}} > \hat{V}(s).$$

Consequently,

$$\max_{a \in \mathcal{A}} \left\{ r(s,a) + \alpha \sum_{s' \in \mathcal{S}} p(s'|s,a)\hat{V}(s') \right\} > \hat{V}(s).$$

Thus, $\hat{V}$ does not satisfy Bellman's equations of optimality (2.3). It therefore cannot equal the optimal value function J^*.

Now consider the "if" part. Proposition 2.2.1 asserts that if condition (2.5) holds, then there exists a policy and corresponding transition probabilities for that policy with $\hat{V}$ as the value function. Recall that the actions prescribed by this policy were denoted $a^*(s)$, and we

established the existence of transition probabilities $p^*(\cdot|s, a^*(s))$ such that

$$\hat{V}(s) = r(s, a^*(s)) + \alpha \sum_{s' \in \mathcal{S}} p^*(s'|s, a^*(s))\hat{V}(s'), \quad \forall s \in \mathcal{S}. \tag{2.8}$$

Now consider any state $s \in \mathcal{S}$ and any action $a \neq a^*(s)$. Let $\hat{s} \in \mathcal{S}$ be such that $\hat{V}(\hat{s}) = \min_{s' \in \mathcal{S}} \hat{V}(s')$. Consider the degenerate transition pmf $\hat{p}(\hat{s}|s, a) = 1$ and $\hat{p}(s'|s, a) = 0$ for $s' \neq \hat{s}$. Then,

$$r(s, a) + \alpha \sum_{s' \in \mathcal{S}} \hat{p}(s'|s, a)\hat{V}(s') = r(s, a) + \alpha \hat{V}(\hat{s}) = r(s, a) + \alpha \min_{s' \in \mathcal{S}} \hat{V}(s') \leq \hat{V}(s).$$

In other words, since s was arbitrary, we have established that

$$\hat{V}(s) \geq \max_{a^*(s) \neq a \in \mathcal{A}} \left\{ r(s, a) + \alpha \sum_{s' \in \mathcal{S}} \hat{p}(s'|s, a)\hat{V}(s') \right\}, \quad \forall s \in \mathcal{S}. \tag{2.9}$$

Combining (2.8) and (2.9), we see that $\hat{V}$ satisfies Bellman's equations of optimality (2.3). It therefore equals the optimal value function J^*. □

Now suppose that $\hat{V}$ is such that conditions (2.5) and (2.7) hold. In particular, as in the proofs of Propositions 2.2.1 and 2.2.2, let $a^*(s)$, for each $s \in \mathcal{S}$, be any action that satisfies condition (2.5). We then need to find transition probabilities for which the policy defined by actions $a^*(s)$ satisfies Bellman's equations of optimality (2.3) with $\hat{V}$. This can be done

simply by solving the feasibility LP

$$\min_{p \in \Re^{|\mathcal{S}| \times |\mathcal{S}| \times |\mathcal{A}|}} \quad 0$$

$$\hat{V}(s) - \alpha \sum_{s' \in \mathcal{S}} p(s'|s,a) \hat{V}(s') \geq r(s,a), \qquad \forall s \in \mathcal{S}, a \in \mathcal{A} \tag{2.10a}$$

$$\hat{V}(s) - \alpha \sum_{s' \in \mathcal{S}} p\big(s'|s,a^*(s)\big) \hat{V}(s') = r\big(s,a^*(s)\big), \qquad \forall s \in \mathcal{S} \tag{2.10b}$$

$$\sum_{s' \in \mathcal{S}} p(s'|s,a) = 1, \qquad \forall s \in \mathcal{S}, a \in \mathcal{A} \tag{2.10c}$$

$$p(s'|s,a) \geq 0, \qquad \forall s, s' \in \mathcal{S}, a \in \mathcal{A}. \tag{2.10d}$$

For all $s \in \mathcal{S}$ and $a^*(s) \neq a \in \mathcal{A}$, the degenerate pmf $\hat{p}(\cdot|s,a)$ defined in the proof of Proposition 2.2.2 does satisfy constraints (2.10a). Nevertheless, instead of forcing this feasible solution, we leave it to an LP solver to discover a feasible pmf, for additional flexibility. This is the reason why the degenerate pmf does not appear anywhere in problem (2.10).

We now turn to the more difficult problem of finding transition probabilities that make a given policy optimal.

2.3 Rendering a given policy optimal

This section utilizes LP formulations of infinite-horizon MDPs. These are reviewed briefly next.

Consider the LP

$$\min_{V \in \Re^{|\mathcal{S}|}} \quad \sum_{s \in \mathcal{S}} V(s)$$

$$V(s) - \alpha \sum_{s' \in \mathcal{S}} p(s'|s,a) V(s') \geq r(s,a), \qquad \forall s \in \mathcal{S}, a \in \mathcal{A}. \tag{2.11a}$$

Here, $V(s)$ is the optimization variable associated with the expected total discounted reward earned when the initial state is s. This LP has a unique optimal solution V^* that equals J^*,

the optimal value function. The dual of (2.11) is

$$\max_{x \in \Re^{|\mathcal{S}|\times|\mathcal{A}|}} \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} r(s,a)x(s,a)$$

$$\sum_{a\in\mathcal{A}} x(s,a) - \alpha \sum_{s'\in\mathcal{S}} \sum_{a\in\mathcal{A}} p(s|s',a)x(s',a) = 1, \qquad \forall s \in \mathcal{S} \tag{2.12a}$$

$$x(s,a) \geq 0, \qquad \forall s \in \mathcal{S}, a \in \mathcal{A}. \tag{2.12b}$$

An interesting and computationally useful property of this dual LP is that there is a one-to-one correspondence between its extreme points (that is, basic feasible solutions [BFSs]) and policies for the MDP [70, Proposition 6.9.3]. Moreover, every BFS of (2.12) has the following property: for every state $s \in \mathcal{S}$, there exists a unique action $a(s) \in \mathcal{A}$ such that $x\big(s, a(s)\big) > 0$; for all other actions $a(s) \neq a \in \mathcal{A}$, $x(s,a) = 0$. This implies that $\sum_{a\in\mathcal{A}} x(s,a) = x\big(s, a(s)\big)$. The unique policy corresponding to any BFS of (2.12) can be identified as follows: for each state s, prescribe the unique action $a(s)$ for which $x\big(s, a(s)\big) > 0$. Consequently, if the Simplex algorithm returns x^* as an optimal BFS to (2.12), it also delivers a corresponding policy. In fact, this policy is optimal to problem (2.2). Conversely, if a policy π^* is optimal to problem (2.2), then a BFS that is optimal to (2.12) exists such that, for all $s \in \mathcal{S}$, we have, $x(s, \pi^*(s)) > 0$ and $x(s,a) = 0$, for all $a \neq \pi^*(s)$. See [70, Theorem 6.9.4]. To solve an infinite-horizon MDP problem, one can solve either the primal (2.11) or the dual (2.12). An optimal solution to one problem can be employed to obtain an optimal solution to the other problem, via complementary slackness. In the remainder of this chapter, these two problems are viewed as "forward problems."

We assume in this section that all rewards $r(s,a)$ are nonnegative. This is without loss of generality. This holds because if there is a state-action pair with negative reward, we can construct an alternative MDP by adding the absolute value of the most negative reward to all original rewards. This new MDP has nonnegative rewards but the set of optimal policies is the same as the original MDP. This holds because adding a constant c to all rewards in an MDP increases the value $J^\pi(s)$ by an amount $c/(1-\alpha)$ for all policies π and all states

$s \in \mathcal{S}$. As such, the difference between value functions of different policies is not affected. To make these statements rigorous, fix any policy π and let $\tilde{J}^\pi$ denote the value function of the alternative MDP. Then, from equation (2.1), we have,

$$\begin{aligned}\tilde{J}^\pi(s) &= \mathbb{E}\left(\sum_{t=0}^{\infty} \alpha^t \Big[r\big(\mathbf{S}_t, \pi(\mathbf{S}_t)\big) + c\Big] \Big| \mathbf{S}_0 = s\right) = \mathbb{E}\left(\sum_{t=0}^{\infty} \alpha^t r\big(\mathbf{S}_t, \pi(\mathbf{S}_t)\big) \Big| \mathbf{S}_0 = s\right) + \left(\sum_{t=0}^{\infty} \alpha^t c\right) \\ &= J^\pi(s) + \frac{c}{1-\alpha}.\end{aligned}$$

Since all rewards are nonnegative, it is easy to see from equation (2.1) that the value function of any policy is nonnegative. In particular, the optimal value function is nonnegative. We can therefore add the nonnegativity constraint $V \geq 0$ to the primal MDP problem (2.11) without loss of optimality. This means that dual constraints (2.12a) should also be written in the inequality form "$\leq$". We will use this inequality form because it is essential for the CCP heuristic (Section 2.3.1) and more convenient for SLP and GOP. We reproduce these two problems here for convenience.

$$\min_{V \in \Re^{|\mathcal{S}|}} \sum_{s \in \mathcal{S}} V(s)$$

$$V(s) - \alpha \sum_{s' \in \mathcal{S}} p(s'|s,a) V(s') \geq r(s,a), \qquad \forall s \in \mathcal{S}, a \in \mathcal{A} \tag{2.13a}$$

$$V(s) \geq 0, \qquad \forall s \in \mathcal{S}. \tag{2.13b}$$

$$\max_{x \in \Re^{|\mathcal{S}|\times|\mathcal{A}|}} \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} r(s,a) x(s,a)$$

$$\sum_{a \in \mathcal{A}} x(s,a) - \alpha \sum_{s' \in \mathcal{S}} \sum_{a \in \mathcal{A}} p(s|s',a) x(s',a) \leq 1, \qquad \forall s \in \mathcal{S} \tag{2.14a}$$

$$x(s,a) \geq 0, \qquad \forall s \in \mathcal{S}, a \in \mathcal{A}. \tag{2.14b}$$

Suppose an x^* with the property that, for every $s \in \mathcal{S}$, $x^*\big(s, a^*(s)\big) > 0$ for some unique

$a^*(s) \in \mathcal{A}$ and $x^*(s,a) = 0$ for all $a \neq a^*(s)$, is optimal to (2.14). Then constraints (2.14a) must be active at this optimal solution, for every $s \in \mathcal{S}$. This can be established by contradiction, utilizing the fact that rewards are nonnegative. This also shows that x^* is, in fact, optimal to (2.12).

Now suppose that the decision-maker is given a policy $\hat{\pi} \in \Pi$. The decision-maker wants to find transition probabilities that render $\hat{\pi}$ optimal. However, depending on the values of rewards, this may not always be achievable. The example in Figure 2.2 illustrates this point.

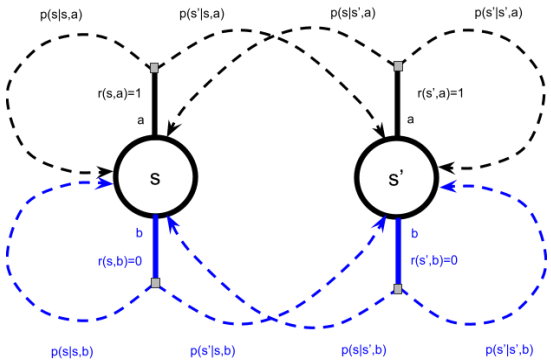


Figure 2.2: An MDP with $\mathcal{S} = \{s, s'\}$ and $\mathcal{A} = \{a, b\}$. Entities related to action a are shown in black line; blue lines for action b. Dotted curved arrows depict probabilistic state transitions. Rewards earned upon choosing actions a, b are $1, 0$, respectively, irrespective of state. Thus, policy $\pi(s) = \pi(s') = a$ is uniquely optimal regardless of the transition probability values. In other words, no values of transition probabilities can render the policy $\pi(s) = \pi(s') = b$ optimal.

We therefore formulate the inverse MDP problem as

$$\min_{p,x,V} \sum_{s\in\mathcal{S}} V(s) - \sum_{s\in\mathcal{S}} r\big(s,\hat{\pi}(s)\big)x\big(s,\hat{\pi}(s)\big)$$

$$V(s) - \alpha\sum_{s'\in\mathcal{S}} p(s'|s,a)V(s') \geq r(s,a), \qquad \forall s\in\mathcal{S}, a\in\mathcal{A} \tag{2.15a}$$

$$V(s) \geq 0, \qquad \forall s\in\mathcal{S} \tag{2.15b}$$

$$x\big(s,\hat{\pi}(s)\big) - \alpha\sum_{s'\in\mathcal{S}} p\big(s|s',\hat{\pi}(s')\big)x\big(s',\hat{\pi}(s')\big) \leq 1, \qquad \forall s\in\mathcal{S} \tag{2.15c}$$

$$x\big(s,\hat{\pi}(s)\big) \geq 0, \qquad \forall s\in\mathcal{S} \tag{2.15d}$$

$$\sum_{s'\in\mathcal{S}} p(s'|s,a) = 1, \qquad \forall s\in\mathcal{S}, a\in\mathcal{A} \tag{2.15e}$$

$$p(s'|s,a) \geq 0, \qquad \forall s,s'\in\mathcal{S}, a\in\mathcal{A}. \tag{2.15f}$$

Constraints (2.15a)-(2.15b) enforce primal feasibility and constraints (2.15c)-(2.15d) ensure dual feasibility. Constraints (2.15e)-(2.15f) ensure that the transition probabilities form genuine pmfs. It is important to note that decision variable x in this problem belongs to $\Re^{|\mathcal{S}|}$ instead of to $\Re^{|\mathcal{S}|\times|\mathcal{A}|}$. This is because the values of $x(s,a)$ were set to 0, for all $a \neq \hat{\pi}(s)$. By weak duality in the pair of problems (2.13)-(2.14), we know that the optimal objective value in (2.15) is nonnegative. The objective minimizes the duality gap between problems (2.13)-(2.14) under this setting. We now establish an important motivating property of this problem.

Proposition 2.3.1. *Suppose the triplet $p^* \in \Re^{|\mathcal{S}|\times|\mathcal{S}|\times|\mathcal{A}|}$, $x^* \in \Re^{|\mathcal{S}|}$, $V^* \in \Re^{|\mathcal{S}|}$ is optimal to problem (2.15), and the corresponding optimal objective value is 0. Then policy $\hat{\pi}$ is optimal to problem (2.2), with transition probabilities p^*. Conversely, if policy $\hat{\pi}$ is the optimal policy of an MDP problem under transition probability p^*, there exist values of x^* and V^* such that (p^*, x^*, V^*) is optimal to problem (2.15) with optimal value 0.*

Proof. If the optimal objective value is 0, then, by strong duality, V^* is optimal to (2.13) and x^* is optimal to (2.14), both with p^* as the transition probabilities.

We claim that $x^*(s, \hat{\pi}(s))$ must be > 0, for each $s \in \mathcal{S}$. For if not, then let s be the smallest state in $\mathcal{S}$ such that $x^*(s, \hat{\pi}(s)) = 0$. We then write the left hand side of constraint (2.15c) as

$$\cancelto{0}{x^*(s, \hat{\pi}(s))}(1 - \alpha p^*(s|s, \hat{\pi}(s))) - \alpha \sum_{\substack{s' \in \mathcal{S} \\ s' \neq s}} p^*(s|s', \hat{\pi}(s')))x^*(s', \hat{\pi}(s')) < 1.$$

Thus, this constraint is inactive. We can therefore increase $x^*(s, \hat{\pi}(s))$ to a positive value to make the constraint active and at the same time decrease the objective value (recall that rewards are nonnegative). Also note that increasing this $x^*(s, \hat{\pi}(s))$ does not violate the feasibility of any other constraint because this $x^*(s, \hat{\pi}(s))$ may only appear with a negative coefficient on the left hand side of other constraints of the form "$\leq$". By recursively applying this procedure, we obtain an alternative feasible solution with a smaller objective value than that of x^*, thus yielding a contradiction. This proves the claim.

Moreover, by following a similar argument, we can show that constraint (2.15c) is active, for every $s \in \mathcal{S}$ (also recall the discussion in the paragraph after problem (2.14)). This implies that the pair V^*, x^* in fact satisfies strong duality for problems (2.11) and (2.12), both with transition probabilities p^*. Recall that this ensures that policy $\hat{\pi}$ is optimal to problem (2.2), with transition probabilities p^*, as claimed.

To prove the converse, fix $x^*(s, a) = 0$, for all $s \in \mathcal{S}$ and $a \neq \hat{\pi}(s) \in \mathcal{A}$. Let V^* and x^* be optimal solutions to the MDP primal and dual problems (2.11) and (2.12), respectively, with transition probabilities p^*. The triplet (p^*, x^*, V^*) is feasible to the inverse problem, and optimality of $\hat{\pi}$ implies by strong duality that the corresponding objective of the inverse problem is zero. □

Problem (2.15) is nonconvex wherein constraints (2.15a) and (2.15c) are bilinear in variable pairs $p - V$ and $p - x$, respectively. Existing studies in the literature on inverse LPs with unknown constraint coefficients are not applicable to our problem, owing to the peculiar structure of MDPs. In inverse LP studies such as Chan and Kaw [25] and Ghobadi and

Mahmoudzadeh [42], the first requirement for a candidate constraint coefficient is to make the observed solutions feasible. In inverse MDPs, however, every policy is feasible regardless of the values of transition probabilities. Furthermore, while in these studies the observed solutions are in the form of primal decision variables, in our case the observed solutions are policies. A policy corresponds to the indices of active primal LP constraints at the optimal solution. Therefore, unlike the inverse problems studied by Chan and Kaw [25] and Ghobadi and Mahmoudzadeh [42], where only the dual feasibility constraints contain bilinear terms, in our context both the primal and the dual feasibility constraints are bilinear.

Bilinear programs can have many local optima [48]. Tailored algorithms in the literature for solving bilinear programs often utilize the branch-and-bound technique. For instance, Al-Khayyal [3] proposed a branch-and-bound technique with linear subproblems defined using the so-called convex envelopes. Ben-Tal et al. [9] proposed another branch-and-bound algorithm where the lower bound was calculated by partitioning the feasible region of the Lagrangian dual problem. Both methods were shown to converge to the optimal solution under certain conditions.

The structure of problem (2.15) is particularly suitable for applying an exact branch-and-bound method proposed by Floudas and Visweswaran [36] called GOP (Global Optimization Procedure). The algorithm iteratively solves primal and relaxed dual subproblems. This provides upper and lower bounds on the optimal value of the original problem. The algorithm starts by partitioning the variable set into complicating and non-complicating variables. In our problem, transition probabilities p are chosen as the complicating and x, V as the non-complicating variables. In every iteration, the primal problem is defined by fixing the value of the complicating variable at the current feasible solution. This yields an LP in our case.

In iteration k, the linear primal problem is defined as

$$\min_{x,V} \sum_{s\in\mathcal{S}} V(s) - \sum_{s\in\mathcal{S}} r\big(s,\hat{\pi}(s)\big)x\big(s,\hat{\pi}(s)\big)$$

$$V(s) - \alpha \sum_{s'\in\mathcal{S}} p^k(s'|s,a)V(s') \geq r(s,a), \qquad \forall s\in\mathcal{S}, a\in\mathcal{A} \tag{2.16a}$$

$$x\big(s,\hat{\pi}(s)\big) - \alpha \sum_{s'\in\mathcal{S}} p^k\big(s|s',\hat{\pi}(s')\big)x\big(s',\hat{\pi}(s')\big) \leq 1, \qquad \forall s\in\mathcal{S} \tag{2.16b}$$

$$V(s), x\big(s,\hat{\pi}(s)\big) \geq 0, \qquad \forall s\in\mathcal{S}. \tag{2.16c}$$

Here, p^k denotes the feasible value of the transition probabilities at iteration k. The relaxed dual subproblems are defined using the generalized Benders algorithm [38]. These are still difficult to solve since they contain an inner-problem over the set of non-complicating variables in every constraint. However, since the inverse MDP problem is bilinear, each of these inner-problems are linear in one of the non-complicating variables x, V. Assuming that these variables are bounded, the solution of each inner-problem lies at a bound (lower/upper) of x or V. As a consequence of this property, it is sufficient to solve the relaxed dual subproblems over all possible combinations of bounds for the non-complicating variables x and V. A bound on $V(s)$, for any $s\in\mathcal{S}$, can be included in (2.15) via the following thought process. It turns out that, for every $s\in\mathcal{S}$, constraint (2.15a) is active for at least one action $a\in\mathcal{A}$ in an optimal solution to (2.15). This can be proven via an argument similar to that in the proof of Proposition 2.3.1. A policy can then be defined by assigning one such "active" action to that state. The corresponding collection of active constraints (2.15a) then satisfies Bellman's evaluation equations (2.4). In other words, any optimal solution V^* to problem (2.15) equals the value function J^π of some policy π. It is easy to see from (2.1) that

$$\min_{a\in\mathcal{A}} r(s,a) + \frac{\alpha \min\limits_{s'\in\mathcal{S},a\in\mathcal{A}} r(s',a)}{1-\alpha} \leq J^\pi(s) \leq \max_{a\in\mathcal{A}} r(s,a) + \frac{\alpha \max\limits_{s'\in\mathcal{S},a\in\mathcal{A}} r(s',a)}{1-\alpha}.$$

This provides lower and upper bounds on variables $V(s)$ that can be included in problem

(2.15), without loss of optimality. Now recall from the proof of Proposition 2.3.1 that, for every $s \in \mathcal{S}$, constraint (2.15c) is active at any optimal solution x^*. That is, we have,

$$x^*\big(s, \hat{\pi}(s)\big) - \alpha \sum_{s' \in \mathcal{S}} p\big(s|s', \hat{\pi}(s')\big) x^*\big(s', \hat{\pi}(s')\big) = 1. \tag{2.17}$$

Thus, $x^*\big(s, \hat{\pi}(s)\big) \geq 1$. Also, adding these equations over $s \in \mathcal{S}$ yields

$$\begin{aligned}
&\sum_{s \in \mathcal{S}} x^*\big(s, \hat{\pi}(s)\big) - \alpha \sum_{s \in \mathcal{S}} \sum_{s' \in \mathcal{S}} p\big(s|s', \hat{\pi}(s')\big) x^*\big(s', \hat{\pi}(s')\big) \\
&= \sum_{s \in \mathcal{S}} x^*\big(s, \hat{\pi}(s)\big) - \alpha \sum_{s' \in \mathcal{S}} x^*\big(s', \hat{\pi}(s')\big) \cancelto{1}{\sum_{s \in \mathcal{S}} p\big(s|s', \hat{\pi}(s')\big)} = \sum_{s \in \mathcal{S}} x^*\big(s, \hat{\pi}(s)\big) - \alpha \sum_{s' \in \mathcal{S}} x^*\big(s', \hat{\pi}(s')\big) \\
&= |\mathcal{S}|.
\end{aligned}$$

This implies that

$$\sum_{s \in \mathcal{S}} x^*\big(s, \hat{\pi}(s)\big) = \frac{|\mathcal{S}|}{1 - \alpha}. \tag{2.18}$$

Plugging this back into (2.17) and utilizing the fact that $p\big(s|s', \hat{\pi}(s')\big) \leq 1$ yields $x^*\big(s, \hat{\pi}(s)\big) \leq 1 + \frac{\alpha|\mathcal{S}|}{1-\alpha}$. In summary, this discussion implies that the lower and upper bounds

$$1 \leq x^*(s, \hat{\pi}(s)) \leq 1 + \frac{\alpha|\mathcal{S}|}{1 - \alpha}$$

can be included in problem (2.15) without loss of optimality. Despite these favorable structural properties, the GOP algorithm is computationally expensive because, in every iteration of the algorithm, up to $2^{2|\mathcal{S}|}$ LPs need to be solved. Therefore, we propose two other heuristic methods called Convex-Concave Procedure (CCP) [56] and Sequential Linear Programming (SLP) [49]. We then computationally compare them against the exact GOP method as a benchmark.

2.3.1 Convex-Concave Procedure (CCP)

Convex-Concave Procedure (CCP) is a heuristic for finding local optimum solutions to the so-called difference of convex (DC) programming problems. DC programming problems have the following form:

$$\begin{aligned} \min_{x \in \Re^n} \; & f_0(x) - g_0(x) \\ & f_i(x) - g_i(x) \leq 0, \quad i = 1, \cdots, m. \end{aligned}$$

Here, $f_i : \mathbb{R}^n \to \mathbb{R}$ and $g_i : \mathbb{R}^n \to \mathbb{R}$ are convex functions, for $i = 0, \cdots, m$. The basic idea of CCP is as follows: denote the feasible solution obtained in iteration k by x^k. In iteration $k+1$, the concave parts, $g_i(x)$, of the objective function and constraints are replaced by their first-order approximations around x^k. The resulting problem is then convex. The algorithm thus solves a sequence of convex programs until a stopping criterion is met.

The first-order approximation of a convex function is a global under-estimator of the function [18]. This means that the resulting constraints are tighter than the original constraints. Thus, the resulting feasible region is a convex subset of the original (nonconvex) feasible region. A similar idea holds for the objective function as well. Therefore, the resulting convex problem is a restriction of the original DC problem, and its solution provides an upper bound on the optimal value of the DC problem. It is known that the CCP algorithm monotonically decreases the value of the objective function in every iteration [56, Section 1.3]. Therefore, it provides tighter upper bounds as iterations progress. Sriperumbudur and Lanckriet [73] proved that when all functions are differentiable, the algorithm converges to stationary points of the DC program. This result applies to the functions that arise in our inverse MDP problem (2.15). That is, CCP is guaranteed to converge to stationary points in our case.

The standard CCP is not suitable for DC programs with nonconvex equality constraints. It can be proven that in this case, the CCP algorithm gets stuck at the current feasible

solution. This is the reason why we converted the equality constraints (2.12a) into the inequality form (2.14a) without loss of optimality.

To apply the CCP algorithm to the inverse MDP problem (2.15), we first need to write it as a DC program. There are multiple ways to write a bilinear term as a difference of two convex functions. We choose the form

$$xy = \frac{1}{4}\left[(x+y)^2 - (x-y)^2\right].$$

The resulting procedure is described in Algorithm 2.

Algorithm 2 shows that the CCP method reduces in our context to solving a sequence of linear programs (2.16) and convex quadratically constrained programs with linear objective (2.19).

2.3.2 Sequential Linear Programming (SLP)

As the name suggests, Sequential Linear Programming (SLP) is an iterative procedure to find local optima of nonlinear optimization problems via a sequence of LPs. At each iteration, the method solves a first-order approximation of the model around the current solution. Unlike CCP, SLP replaces all nonlinear functions with their linear approximations. Therefore, all higher-order information is lost. While CCP converges to stationary points of the inverse MDP problem and provides improving bounds in every iteration, the objective function values found by SLP do not satisfy any monotonicity property and may not converge to stationary points of the problem. Another drawback of SLP is that it might fail in an iteration to find a feasible solution to the original problem. However, using the bilinearity property of our inverse MDP problem, we are able to avoid this pitfall in every iteration. In particular, in every iteration k, we first solve the inverse MDP problem with the complicating variable fixed at $p = p^k$ (2.16). Here, p^k are the transition probabilities delivered from the previous iteration. This yields an LP and the corresponding values V^k, x^k of the non-complicating variables. The inverse MDP problem (2.15) is then linearized using first-order

Algorithm 2 Convex-Concave Procedure (CCP) for inverse MDP problem (2.15)

input: Initial transition probability vector $p^0 \in \Re^{|\mathcal{S}|\times|\mathcal{S}|\times|\mathcal{A}|}$
initialize: $k := 0$
while *Stopping criterion not satisfied* **do**

1. Solve the inverse MDP problem at $p = p^k$ (problem (2.16)). Store the optimal values of variables as x^k and V^k.
2. Set the value of p^{k+1} to the optimal solution of the following convexified inverse MDP problem:

$$\begin{aligned}
&\min_{p,x,V} \sum_{s\in\mathcal{S}} V(s) - \sum_{s\in\mathcal{S}} r\big(s,\hat{\pi}(s)\big)x\big(s,\hat{\pi}(s)\big) \\
&\quad V(s) - \frac{\alpha}{4}\sum_{s'\in\mathcal{S}}\Big(p(s'|s,a)+V(s')\Big)^2 \\
&\qquad + \frac{\alpha}{4}\sum_{s'\in\mathcal{S}}\Big(p(s'|s,a)-V(s')\Big)^2\Big|^{lin}_{(p^k,V^k)} \geq r(s,a), && \forall s\in\mathcal{S}, a\in\mathcal{A} && \text{(2.19a)}\\
&\quad x\big(s,\hat{\pi}(s)\big) + \frac{\alpha}{4}\sum_{s'\in\mathcal{S}}\Big(p\big(s|s',\hat{\pi}(s')\big) - x\big(s',\hat{\pi}(s')\big)\Big)^2 \\
&\qquad - \frac{\alpha}{4}\sum_{s'\in\mathcal{S}}\Big(p\big(s|s',\hat{\pi}(s')\big) + x\big(s',\hat{\pi}(s')\big)\Big)^2\Big|^{lin}_{(p^k,x^k)} \leq 1, && \forall s\in\mathcal{S} && \text{(2.19b)}\\
&\quad V(s), x\big(s,\hat{\pi}(s)\big) \geq 0, && \forall s\in\mathcal{S} && \text{(2.19c)}\\
&\quad \sum_{s'\in\mathcal{S}} p(s'|s,a) = 1, && \forall s\in\mathcal{S}, a\in\mathcal{A} && \text{(2.19d)}\\
&\quad p(s'|s,a) \geq 0, && \forall s,s'\in\mathcal{S}, a\in\mathcal{A}. && \text{(2.19e)}
\end{aligned}$$

Here, the functions $(x+y)^2|^{lin}_{(x^k,y^k)}$ and $(x-y)^2|^{lin}_{(x^k,y^k)}$ refer to the first-order approximations of $(x+y)^2$ and $(x-y)^2$ around the point (x^k, y^k). That is,

$$\begin{aligned}
(x+y)^2|^{lin}_{(x^k,y^k)} &= (x^k+y^k)^2 + 2(x^k+y^k)(x-x^k) + 2(x^k+y^k)(y-y^k), \text{ and}\\
(x-y)^2|^{lin}_{(x^k,y^k)} &= (x^k-y^k)^2 + 2(x^k-y^k)(x-x^k) + 2(-x^k+y^k)(y-y^k).
\end{aligned}$$

3. Update: $k \leftarrow k+1$.

end

approximations of all bilinear terms at (p^k, x^k, V^k) to obtain problem (2.20). Its optimal solution then provides the values p^{k+1} of the complicating variable to be utilized in the next iteration. Note that the first-order approximation of the bilinear term xy around (x_0, y_0) is

$$xy\big|^{lin}_{(x_0,y_0)} = x_0 y + y_0 x - x_0 y_0$$

The benefit in each iteration of SLP over CCP in our case is that it only solves LPs instead of quadratic programs. The SLP procedure is summarized in Algorithm 3.

2.4 Computational experiments

In the following subsections, we present computational experiments for solving inverse MDP problems described in Section 2.3. We do this for generic MDPs, inventory control, equipment replacement, and multi-armed bandit problems. The discount factor was fixed at $\alpha = 0.95$ in all test instances. We set a maximum runtime for all algorithms. If any algorithm obtained the same objective value for 10 consecutive iterations, we stopped that algorithm. We compared the best objective function values reached by each algorithm. An initial guess for the transition probabilities and a policy $\hat{\pi}$ were generated and provided as input to all algorithms. Let DG_0 denote the optimal objective value in problem (2.15), associated with the given policy $\hat{\pi}$ and the initial guess transition probabilities. Note that when the transition probabilities are fixed, that problem is an LP and thus DG_0 is easy to compute. Let $DG_{algorithm}$ denote the best objective value reached by some algorithm. To compare the performance of different algorithms, we define a metric to quantify the percentage improvement in the objective value obtained by each algorithm as

$$\frac{DG_0 - DG_{algorithm}}{DG_0} \times 100. \tag{2.21}$$

This value equals 100% if, and only if, $DG_{algorithm} = 0$. That is, if and only if, an algorithm is able to reach the optimal objective value of 0. As such, the values of this metric range

Algorithm 3 Sequential Linear Programming (SLP) for inverse MDP problem (2.15)

input: Initial transition probability vector $p^0 \in \Re^{|\mathcal{S}|\times|\mathcal{S}|\times|\mathcal{A}|}$
initialize: $k := 0$
while *Stopping criterion not satisfied* **do**

1. Solve the inverse MDP problem at $p = p^k$ (problem (2.16)). Store the optimal values of variables as x^k and V^k.
2. Solve the inverse MDP problem where the bilinear terms are replaced with their first-order approximations around (p^k, x^k, V^k)

$$\min_{p,x,V} \sum_{s\in\mathcal{S}} V(s) - \sum_{s\in\mathcal{S}} r\big(s,\hat{\pi}(s)\big)x\big(s,\hat{\pi}(s)\big)$$

$$V(s) - \alpha \sum_{s'\in\mathcal{S}} \Big[p(s'|s,a)V^k(s') + p^k(s'|s,a)V(s') - p^k(s'|s,a)V^k(s')\Big] \geq r(s,a), \quad \forall s\in\mathcal{S}, a\in\mathcal{A} \tag{2.20a}$$

$$V(s), x\big(s,\hat{\pi}(s)\big) - \alpha \sum_{s'\in\mathcal{S}} \Big[p\big(s|s',\hat{\pi}(s')\big)x^k\big(s',\hat{\pi}(s')\big) + p^k\big(s|s',\hat{\pi}(s')\big)x\big(s',\hat{\pi}(s')\big) - p^k\big(s|s',\hat{\pi}(s')\big)x^k\big(s',\hat{\pi}(s')\big)\Big] \leq 1, \quad \forall s\in\mathcal{S} \tag{2.20b}$$

$$x\big(s,\hat{\pi}(s)\big) \geq 0, \quad \forall s\in\mathcal{S} \tag{2.20c}$$

$$\sum_{s'\in\mathcal{S}} p(s'|s,a) = 1, \quad \forall s\in\mathcal{S}, a\in\mathcal{A} \tag{2.20d}$$

$$p(s'|s,a) \geq 0, \quad \forall s,s'\in\mathcal{S}, a\in\mathcal{A}. \tag{2.20e}$$

Set the value of p^{k+1} to the optimal solution of this problem.

3. Update: $k \leftarrow k+1$.

end

between 0 and 100, and values closer to 100 are better. All mathematical programs were solved using the cvxpy package on Python 3.8.

2.4.1 Generic MDPs

For inverse generic MDP problems, we used the exact GOP method as a benchmark and compared the performance of the CCP and SLP heuristics against it. We considered 4 sets of randomly generated MDPs with different numbers of states and actions. For each problem set, we generated 20 instances. For each instance, the immediate rewards were sampled from a uniform distribution over the interval [1,50], and the policy $\hat{\pi}$ was sampled according to a discrete uniform distribution on the set of actions. For each action $a \in \mathcal{A}$, entries of an initial transition probability matrix were sampled from a uniform [0,1] distribution, and then appropriately normalized so that they summed to 1. The maximum runtime for each problem instance was set as $0.5 \times$ number of states $\times$ number of actions, in minutes.

Results of our simulations are presented in Figure 2.3. The filled markers show the average performance of each algorithm over 20 instances, while the hollow markers show the performance in individual instances. The figure shows that the SLP method reached the optimal objective value of 0 for each problem instance. That is, SLP was able to find transition probabilities under which the given policy $\hat{\pi}$ is optimal, in every instance.

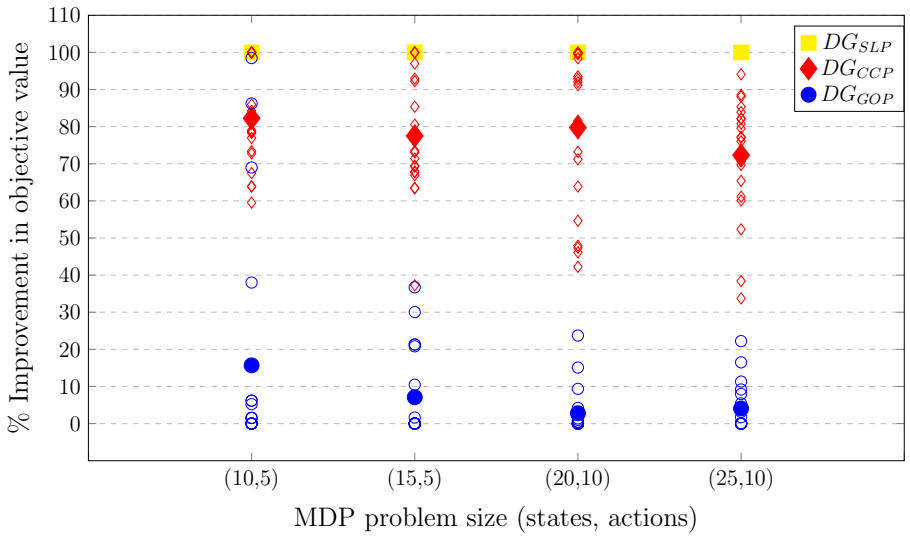


Figure 2.3: Performance of 3 algorithms in solving inverse generic MDPs. Each filled marker represents the average over 20 instances; individual instances are shown as hollow markers.

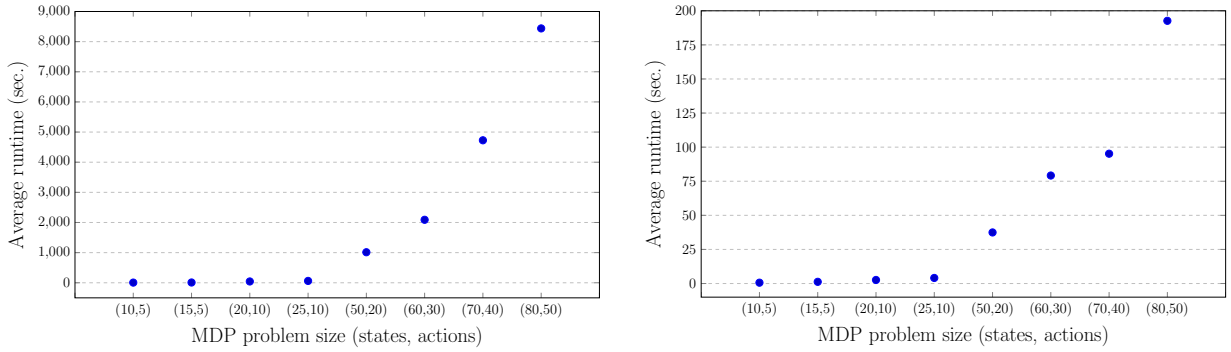
Figure 2.3 shows that the SLP heuristic performed the best among the three methods on our problem instances. Recall that in addition to the linear primal problem (2.16) which is solved by all algorithms in every iteration, SLP solves one LP in every iteration, while GOP solves $2^{2|\mathcal{S}|}$ LPs, and CCP solves a quadratically constrained program with linear objective in every iteration. Therefore, in the given limited runtime, SLP performed the highest number of iterations.

While the CCP and GOP algorithms reached the maximum runtime before finding an optimal solution in most test instances, SLP found an optimal solution before hitting the maximum runtime. To further explore the performance of SLP, we expanded our experiments to 4 sets of larger MDPs. Again, for each set, we generated 20 instances. The maximum runtime for these larger instances was set as 0.05 $\times$ number of states $\times$ number of actions, in minutes. Unlike the smaller instances, SLP did not always find the objective function value of 0 in these larger instances with randomly generated policies $\hat{\pi}$. In such cases, the optimal objective value is unknown and we do not know whether or not the given policy $\hat{\pi}$ can be rendered optimal. Therefore, we defined another policy generation method that

guarantees the generated policies can be rendered optimal. This is done using a reverse engineering technique. We randomly generated the initial transition probability and solved the corresponding LP derived from (2.15) to find an associated optimal policy. We repeated this process 1000 times and selected as $\hat{\pi}$ a policy that is optimal in the least number of instances. We call policies generated in this manner "zero-DG policies". Informally, the 1000 repetitions helped identify a policy that does not have a high probability of being optimal. Here, a high probability of a policy being optimal means that there are more choices of transition probabilities that render it optimal.

Figure 2.4 shows the average runtime of SLP. Figure 2.4a shows the average runtime of SLP for randomly generated policies with $DG_{SLP} = 0$. We generated random policies until we could collect 20 instances with DG_{SLP}=0. Figure 2.4b shows the average runtime of SLP for zero-DG policies. In all these instances, SLP reached an optimal objective value of 0. As expected, the figure shows that the runtime of SLP increases with problem size. For the zero-DG policies, SLP found the optimal objective value of 0 in less than 200 seconds, on average. However, for the randomly generated policies, the runtimes are larger. Recall that a zero-DG policy appears as the optimal policy at least once among the 1000 random MDP instances generated. The 1000 repetitions were implemented to find a policy with smaller likelihood of being optimal. However, in larger problems with large policy space, we might need many more repetitions to observe the entire set of policies that can obtain objective value of 0 in the inverse problem. Therefore, in most of problem instances included in Figure 2.4, the random policies that are selected from the entire set of policies with 0 duality gap, are still less likely to be optimal compared to the zero-DG policies, resulting in longer runtime for SLP. Overall, these fast runtimes demonstrate the potential of SLP not only for problems where it can find an optimal solution but perhaps also as an excellent method to quickly generate high-quality solutions as initial guesses for other heuristic or exact methods.

Due to the large degrees of freedom in inverse generic MDP problems, they can have multiple optimal solutions. However, in problems wherein transition probabilities are required to satisfy other side constraints, the degrees of freedom could be lower. To explore the perfor-



(a) Random policies

(b) Zero-DG policies

Figure 2.4: Runtime of SLP. Each point represents average runtime over 20 instances.

mance of SLP in such situations, we conducted numerical experiments on inverse inventory control problems, equipment replacement problems, and multi-armed bandit problems. In all these problems, policies $\hat{\pi}$ were generated using the aforementioned reverse engineering technique for zero-DG policies. SLP's performance on these problems is described in the next three subsections.

2.4.2 Inventory control problems

We studied SLP's performance on an infinite-horizon version of the inventory control problem described in [11, Example 1.3.2]. The problem assumes a maximum inventory level $I_{\max}$, a maximum demand $w_{\max}$, and a quadratic penalty associated with both excess inventory and unmet demand at the end of each period. The integer-valued inventory level at the beginning of a time-period is denoted by s, the integer-valued order amount is denoted by a, and the integer-valued random demand is denoted by w, with pmf $p(w)$. These three quantities determine the inventory level at the beginning of the next time-period. The ordering cost is assumed to equal 1 per unit ordered, and the cost function thus equals $a+(s+a-w)^2$. Note that in this case, the immediate costs depend on transition probabilities. As a result, the objective function of the inverse inventory control problem is bilinear in variables x, p. This

bilinearity in the objective function is handled similar to that in the constraints. The only allowable order quantities are such that $a \leq I_{\max} - s$. For these feasible order quantities, the transition probabilities $p(s'|s,a)$ are nonzero only if $s' \leq s+a$ and satisfy the side constraints

$$p(s'|s,a) = \begin{cases} \sum_{w=s+a}^{w_{\max}} p(w), & \text{if } s' = 0 \\ p(s+a-s'), & \text{otherwise.} \end{cases}$$

We implemented SLP on 5 sets of inventory control problems. For each set, we generated 20 random instances. For each instance, the maximum runtime was set as $0.5 \times I_{\max}$, in minutes. In all instances, SLP found the optimal objective value of 0 before hitting the maximum runtime. A summary of the results is provided in Table 2.1. The table shows that SLP quickly obtained 100% improvement in the initial objective value in all problem sets.

$w_{\max}$	$I_{\max}$	average % improvement in DG_0	average runtime (sec.)
20	20	100.00	9.90
30	30	100.00	29.01
40	40	100.00	57.17
50	50	100.00	111.52
60	60	100.00	171.60

Table 2.1: Performance of the SLP algorithm while solving 5 sets of inverse inventory control problems. Each problem set included 20 instances.

2.4.3 Equipment replacement problems

We studied SLP's performance on the equipment replacement problem described in the book by Puterman [70, Section 4.7.5], with one change. We considered a finite-state version of the problem instead of the countably infinite version described in Puterman's book. In particular, we included a worst possible condition for the equipment, which served as an

absorbing state if the equipment is not replaced. Therefore, the state space is defined as $\mathcal{S} = \{0, 1, \cdots, s_{\max}\}$, where higher values of s correspond to worse conditions of the equipment and $s = 0$ corresponds to a new equipment. The equipment deteriorates by i states with probability $p(i)$ if it is not replaced (action $a = 0$). The transition probabilities thus satisfy the side constraints

$$p(s'|s,0) = \begin{cases} 0, & \text{if } s' < s \\ p(s'-s), & \text{if } s \leq s' < s_{\max} \\ 1 - \sum_{s'=s}^{s_{\max}-1} p(s'-s), & \text{if } s' = s_{\max}. \end{cases}$$

The transition probabilities associated with replacing the equipment ($a = 1$) are simply $p(s'|s,1) = p(s')$. This is because state 0 corresponds to a new equipment, which deteriorates by s' states with probability $p(s')$. The operating cost in state s is $h(s) = 2s$, and the replacement cost K was sampled from a uniform distribution over the interval $[h(0), 1.2 \times h(s_{\max})]$. We set the maximum runtime as $0.5 \times s_{max}$, in minutes. However, in all test instances, SLP stopped before hitting the maximum runtime either because it found the optimal objective value of 0 or because it obtained the same objective value in 10 consecutive iterations.

Table 2.2 shows that on average, SLP did not find the optimal objective function value of 0 in some instances. However, the results show that SLP did obtain an approximately 90% improvement in the initial objective value quickly.

$\|\mathcal{S}\|$	average % improvement in DG_0	average runtime (sec.)
10	84.76	1.35
20	95.19	2.99
30	93.36	7.26
40	91.48	8.38
50	91.12	14.48

Table 2.2: Performance of SLP algorithm in solving 5 sets of inverse equipment replacement problems. Each problem set included 20 instances. The number of states$|\mathcal{S}| = s_{\max} + 1$, because the state space $\mathcal{S} = \{0, 1, \cdots, s_{\max}\}$.

2.4.4 Multi-armed bandit problems

We consider the classical multi-armed bandit (MAB) problem, which is described as follows. There is a finite set of arms $\mathcal{N} = \{1, \cdots, N\}$. At each time-step $t \in \{0, 1, 2, ...\}$, each arm $n \in \mathcal{N}$ is in some state $s_n \in \mathcal{S}_n$, where $\mathcal{S}_n$ represents the finite state-space of arm n. The player is restricted to pulling one arm at a time. If arm n is pulled, s_n evolves to s'_n with probability $p_n(s'_n|s_n)$, yielding a corresponding reward $r_n(s_n)$. No reward is earned from arms that are not pulled and their states do not change. The player's goal is to choose an arm to play in each state $s = (s_1, \cdots, s_n)$ to maximize the expected total discounted reward over an infinite horizon. Therefore, the transition probabilities of this model associated with the action of pulling arm n in state s satisfy the following side constraints,

$$p(s'|s, n) = \begin{cases} p_n(s'_n|s_n), & \text{if } s'_j = s_j \text{ for all } j \in \mathcal{N} \setminus \{n\} \\ 0, & \text{otherwise.} \end{cases}$$

We implemented SLP on 7 inverse multi-armed bandit problem sets, each with 20 randomly generated instances. The immediate rewards were sampled from a uniform distribution over the interval [1,50]. The number of states per arm was assumed to be constant across

all arms, and is denoted by $|\mathcal{S}|$. The maximum runtime for each MAB instance was set to $N \times |\mathcal{S}|$, in minutes. SLP's performance is summarized in Table 2.3. The total number of states in the multi-armed bandit problem is $|\mathcal{S}|^N$, which increases exponentially with the number of arms N. Thus, the size of the LP solved by SLP in every iteration is exponential in the number of arms. The average % improvement results in Table 2.3 suggest that as the total number of states gets larger, the average performance of SLP deteriorates. A closer look at the results revealed that in instances where SLP did not find the optimal objective value of 0, it had hit the maximum runtime limit. This means that SLP's performance was mainly hurt by the artificially set runtime. This is expected, given the curse of dimensionality in inverse multi-armed bandit problems.

N	$\lvert\mathcal{S}\rvert$	total number of states	average % improvement in DG_0	average runtime (sec.)
5	3	243	100.00	152.66
4	4	256	100.00	150.83
4	5	625	99.98	621.88
5	4	1024	99.46	1033.01
4	6	1296	84.70	1378.21
7	3	2187	83.66	1284.72
5	5	3125	59.99	1721.19

Table 2.3: Performance of SLP algorithm in solving 7 sets of inverse multi-armed bandit problems. Each problem set included 20 instances.

2.5 Conclusions

This chapter provided an inverse optimization approach for recovering transition probabilities of an MDP. We considered two variants of the problem. In the first variant, the decision-maker wonders whether there exist transition probabilities and corresponding decisions that would attain a given value function. We derived an easy-to-verify necessary and sufficient

condition for this existence. When this condition is met, the requisite transition probabilities and decisions can be imputed by solving a feasibility LP. These ideas were then extended to the case when the decision-maker wishes to render the given value function optimal. In the second variant, we studied the more difficult problem of imputing transition probabilities that make a given policy optimal. We applied two heuristics called CCP and SLP to solve the resulting nonconvex bilinear program. The performance of these two methods was compared via numerical experiments against an exact global optimization method. These experiments showed that SLP outperforms the other two methods in terms of both the runtimes and objective values in generic MDP problem instances. Results of applying SLP to inverse inventory control, equipment replacement, and multi-armed bandit problems revealed its potential for finding high-quality solutions in a reasonable amount of time.

We focused on inverse MDPs with unknown transition probabilities when the immediate rewards are known. It can be shown that when both transition probabilities and rewards are unknown and there is no side constraints on their values, it is trivial to find values of these unknown parameters that render any value function or policy optimal. While our work can handle the case where both transition probabilities and rewards are unknown but have side constraints or need to be close to a nominal guess value, future work could consider more tailored approaches to handle the non-convexity in this problem.

Chapter 3

INVERSE OPTIMIZATION IN SEMI-DEFINITE PROGRAMS WITH UNKNOWN CONSTRAINT PARAMETERS

3.1 Introduction

Semi-definite programs (SDPs) form a class of convex optimization problems where the objective function is affine, and the feasible region is defined by a linear matrix inequality (LMIs). A linear matrix inequality requires that an affine combination of symmetric matrices be positive semi-definite (PSD) [79]. SDPs are a special case of conic programs and include a variety of convex problems such as linear and convex quadratic programs. Specifically, SDPs can be viewed as an extension of LPs where the usual element-wise vector inequalities are replaced by LMIs [79]. Efficient algorithms are available for solving SDPs. For instance, interior point methods are known to exhibit polynomial worst-case complexity [4, 61]. SDPs arise in many applications such as control theory, robust optimization, combinatorial optimization, and statistics [47, 51, 80].

In Chapter 2, we developed an inverse optimization framework for MDPs with unknown transition probabilities. Some of the ideas discussed in that chapter can naturally be extended to LPs. In this chapter, however, we take a step further, and extend the inverse optimization framework to the more general case of SDPs with unknown constraint parameters.

We will study six variants of the inverse SDP problem, depending on which parameters and decision variable values are known to the decision-maker. Similar to Chapter 2, in all cases, the resulting inverse problem is a nonconvex bilinear program. In each case, we will seek conditions that render the problem trivial, and conditions that result in a tractable convex reformulation. When such conditions are not met, we will apply tailored versions of three heuristics called the Convex-Concave Procedure (CCP), Sequential Semidefinite Program-

ming (SSP), and Alternate Convex Search (ACS) for approximate solution. The numerical results for problems studied in subsections 3.3.1 and 3.3.3 are presented. Our experiments show the CCP method has the best performance in solving inverse SDP problems with fixed value of cost coefficient and right hand side parameters. In the 3 other experimented variants, while all the algorithms are able to reach to the optimal solution, ACS is the fastest, and thus the winning method.

This chapter is organized as follows. In Section 3.2, we introduce an SDP formulation. In Section 3.3, we study three variants of the inverse SDP problem when a solution $\hat{X}$ to the primal SDP is given. Similarly, in Section 3.4, we consider three variants of the inverse SDP problem when a solution to the dual SDP is given. Numerical results are provided in Section 3.5. Directions for future research are discussed in Section 3.6.

We will use the following notation throughout this chapter. The $n \times n$ identity matrix is denoted by $\mathbb{I}_n$. Let S^n denote the set of $n \times n$ symmetric matrices. The set of $n \times n$ PSD matrices is denoted by S^n_+. We sometimes use A to refer to the collection of matrices $\{A_i\}_{i \in I}$, where I is a finite index set. Matrices 0_m, $0_{m,n}$, J_m, $J_{m,n}$ represent the $m \times m$ matrix of all zeros, $m \times n$ matrix of all zeros, $m \times m$ matrix of all ones, and $m \times n$ matrix of all ones, respectively. The pair (i, j) refers to the i^{th} row and j^{th} column of a matrix. Finally, e_i refers to the unit vector with the i^{th} element equal to 1; and $e_{i,j}$ refers to a matrix of appropriate size with the entry in position (i, j) equal to 1, and the rest equal to 0.

3.2 Problem statement

In this chapter, we consider convex semidefinite programs (SDP) of the form

$$\min_X \operatorname{Tr}(CX) \tag{3.1a}$$

$$\operatorname{Tr}(A_i X) \geq b_i, \quad \forall i \in I \tag{3.1b}$$

$$X \succeq 0, \tag{3.1c}$$

where $|I| = m$, and $b \in \mathbb{R}^m$. The matrices $A_1, \cdots, A_m, C \in S^n$. Observe that the variable X is in the positive semidefinite cone.

We denote the dual variable associated with constraints (3.1b) by $\pi \in \mathbb{R}^m$. The dual SDP is

$$\max_{\pi} \ \pi^T b \tag{3.2a}$$

$$\sum_{i \in I} \pi_i A_i \preceq C \tag{3.2b}$$

$$\pi \geq 0, \tag{3.2c}$$

where constraint (3.2b) is an LMI.

Given an observed/desired solution to the primal SDP problem (3.1) or the dual SDP problem (3.2), we aim to find values of constraint matrices $A_1, \cdots, A_m$ that minimize the duality gap for the given solution. These matrices may also need to satisfy a set of side constraints. In this study, we consider side constraints of the following LMI form

$$\sum_{i \in I} [F_0^i + \sum_{j=1}^{n} \sum_{k=1}^{n} a_{jk}^i F_{jk}^i] \succeq 0. \tag{3.3}$$

Here, a_{jk}^i denotes the component (j,k) of matrix A^i. Matrices $F_0^i, F_{jk}^i \in S^q$ for all $i \in I, j, k \in \{1, \cdots, n\}$ are given parameters of the problem. Note that multiple LMIs can always be expressed as a single LMI, using block-diagonal matrices. This is why we defined the side constraints as a single LMI. If matrices F_0^i, F_{jk}^i are diagonal matrices for all $i \in I$, constraint (3.3) simplifies to a set of q linear inequality constraints. A few examples of constraints that are special cases of (3.3), and their associated parameters are provided here.

- $l \leq \text{Tr}(A_i) \leq u$
  $F_0^i = \begin{pmatrix} u & 0 \\ 0 & -l \end{pmatrix}$, $F_{jj}^i = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$ for all $1 \leq j \leq n$, $F_{jk}^i = 0_2$ if $j \neq k$, and $F_0^{i'}, F_{j,k}^{i'} = 0_2$ for $i' \neq i$.

- $A_i \in S^n$

  The number of linear constraints needed to impose symmetry is $q = \frac{n(n-1)}{2}$, which is the number of upper diagonal elements in an $n \times n$ matrix. For $j < k$ define $F^i_{jk} = e_{s,s} \in S^q$ where $s = \sum_{r \leq j-1} (n-r) + (k-j)$ (number of upper-diagonal elements before component (j,k)). Set $F^i_{kj} = -F^i_{jk}$. $F^i_{jj} = 0_q$ for all j. $F^{i'}_0, F^{i'}_{j,k} = 0_q$ for all $i' \neq i$.

- $A_i \succeq 0$

  For all j, k define $F^i_{jk} = e_{i,j} \in S^n$. $F^{i'}_0, F^{i'}_{j,k} = 0_n$ for all $i' \neq i$.

Note that in these examples we defined F matrices assuming that the given constraints are the only side constraints of the problem. As discussed before, additional side constraints can be incorporated by defining the F matrices as block diagonal matrices where each block represents the coefficient matrix associated with each LMI.

In the following sections, we study six variants of the inverse SDP problem with different types of given solution and known parameters. We will show that in all cases, the inverse SDP problems are nonconvex bilinear problems. Whenever the bilinear problem can not be reformulated as a tractable problem, an approximate solution is obtained using the convex concave procedure (CCP), and sequential semidefinite programming (SSP) defined similar to Chapter 2.

Since first-order approximations of functions are more accurate near the point around which the approximation is implemented, in this chapter, we add trust region constraints to the SSP algorithm. In iteration k of SSP, trust region constraints restrict the decision variables to be in the vicinity of the current feasible solution. These constraints need an input parameter called trust region radius denoted by ρ, which defines the maximum allowed distance from the current solution. As discussed in Chapter 2, a drawback of the SLP algorithm is that the solution of the first-order approximated problem may not be feasible to the original problem. This is also true for the SSP algorithm. To overcome this, every time the first-order approximated inverse problem returns an infeasible solution, we cut it from the feasible region of the approximated problem by appropriately adjusting the trust region

radius, and re-solving the problem. For example, if at iteration k, the approximated problem returns the infeasible solution Y^* while the trust region constraint was defined as $\|Y - Y^k\| \leq \rho$, the adjusted trust region radius denoted by ρ' can be defined as $\rho' = 0.9\|Y^* - Y^k\|$ to cut the solution Y from the feasible region. In addition to the CCP and SSP heuristics, we will use another known heuristic for solving bilinear problems, called alternate convex search (ACS) [83] when applicable. The algorithm partitions the set of variables into B disjoint sets. In every step, the algorithm solves B subproblems where each subproblem is defined similar to the original problem over one set of variables while the other sets of variables are fixed. A summary of the CCP, SSP, and ACS methods for solving the bilinear inverse SDP problems are provided in algorithms 4, 5, 6 respectively.

Algorithm 4 Convex-Concave Procedure (CCP) for inverse SDP problem

input: Initial coefficient matrices $\{A_i^0\}_{i \in I}$

initialize: $k := 0$

while *Stopping criterion not satisfied* **do**

1. Solve the inverse SDP problem at $A = A^k$. Store the optimal values of variables as X^k in inverse problems discussed in section 3.3, or π^k in inverse problems discussed in section 3.4.
2. Set the value of A^{k+1} to the optimal solution of the convexified inverse SDP problem around the current solution.
3. Update: $k \leftarrow k + 1$.

end

Algorithm 5 Sequential semidefinite programming (SSP) for inverse SDP problem

input: Initial coefficient matrices $\{A_i^0\}_{i\in I}$, Euclidean trust region radius ρ.

initialize: $k := 0$

while *Stopping criterion not satisfied* **do**

1. Solve the inverse SDP problem at $A = A^k$. Store the optimal values of variables as X^k in inverse problems discussed in section 3.3, or π^k in inverse problems discussed in section 3.4.
2. Set the value of A^{k+1} to the optimal solution of the first-order approximated inverse SDP problem around the current solution with trust region constraints. If the problem is infeasible, repeat step 2 with appropriately adjusted radius ρ'.
3. Update: $k \leftarrow k + 1$.

end

Algorithm 6 Alternate convex search (ACS) for inverse SDP problem

input: Initial coefficient matrices $\{A_i^0\}_{i\in I}$.

initialize: $k := 0$

while *Stopping criterion not satisfied* **do**

1. Solve the inverse SDP problem at $A = A^k$. Store the optimal values of variables as X^k in inverse problems discussed in section 3.3, or π^k in inverse problems discussed in section 3.4.
2. Solve the inverse SDP problem at X^k in inverse problems discussed in section 3.3, or π^k in inverse problems discussed in section 3.4. Store the optimal values of variables as A^{k+1}.
3. Update: $k \leftarrow k + 1$.

end

In the rest of this chapter we use "convexified inverse SDP" to refer to the problem derived by first writing the inverse SDP problem in a DC (difference of convex) programming form and then convexified through first-order approximation of the nonconvex terms. On the other hand, "first-order approximated inverse SDP" refers to the problem derived by directly replacing the nonconvex terms in the inverse SDP problem with their first-order approximations.

3.3 Inverse SDP problem given $\hat{X}$

In the following three subsections, we will study three variants of the inverse SDP problem when a solution $\hat{X} \in S^n_+$ to the primal SDP problem (3.1) is given, and we aim to find constraint matrices that minimize the duality gap for this given solution.

3.3.1 Known Parameters C and b

When cost parameters and right hand side parameters of the primal SDP problem are known, given an observed solution $\hat{X} \in S^n_+$ to the primal SDP problem, the inverse SDP problem is given by

$$\min_{A,\pi} \operatorname{Tr}(C\hat{X}) - \pi^T b \tag{3.4a}$$

$$\operatorname{Tr}(A_i\hat{X}) \geq b_i, \quad \forall i \in I \tag{3.4b}$$

$$\sum_{i\in I} \pi_i A_i \preceq C \tag{3.4c}$$

$$\pi \geq 0 \tag{3.4d}$$

$$\sum_{i\in I} [F^i_0 + \sum_{j=1}^{n}\sum_{k=1}^{n} a^i_{jk} F^i_{jk}] \succeq 0. \tag{3.4e}$$

Here, constraints (3.4b) represent primal feasibility; (3.4c)-(3.4d) ensure dual feasibility; and (3.4e) represent side constraints on matrices A_i. The objective function (3.4a) minimizes the duality gap for the given solution $\hat{X}$. Observe that the inverse problem is a nonconvex program where constraint (3.4c) is a bilinear matrix inequality (BMI). Note that the term $\operatorname{Tr}(C\hat{X})$ is a constant, but we keep it in the objective for consistency with other variants.

To apply the CCP method to this nonconvex bilinear problem, we first need to write the nonconvex constraint (3.4c) in a DC form. There are multiple ways to write a bilinear term

as a difference of two convex functions. Throughout this chapter, we choose the form

$$\sum_{i\in I}\pi_i A_i = \sum_{i\in I}\pi_i \mathbb{I}_n A_i = \sum_{i\in I}\frac{1}{4}\big[(\pi_i\mathbb{I}_n + A_i)^2 - (\pi_i\mathbb{I}_n - A_i)^2\big].$$

We denote the feasible solution in iteration k by (A^k, π^k). The first-order approximation of the nonconvex quadratic piece around the point (A^k, π^k) is

$$(\pi_i\mathbb{I}_n - A_i)^2 \approx 2(\pi_i^k\mathbb{I}_n - A_i^k)(\pi_i\mathbb{I}_n - A_i) - (\pi_i^k\mathbb{I}_n - A_i^k)^2.$$

Using this, in iteration k of the CCP algorithm, to derive the convexified inverse SDP problem, constraint(3.4c) is replaced by the following convex constraint

$$\sum_{i\in I}\big[(\pi_i\mathbb{I}_n + A_i)^2 - 2(\pi_i^k\mathbb{I}_n - A_i^k)(\pi_i\mathbb{I}_n - A_i) + (\pi_i^k\mathbb{I}_n - A_i^k)^2\big] \preceq 4C. \tag{3.5}$$

In semidefinite programs, nonlinear (convex) inequalities are converted into an LMI form by using Schur complements. The LMI form of the convexified constraint (3.5) using Schur complement is

$$\left(\begin{array}{c|c} \mathbb{I}_{mn} & \begin{pmatrix}\pi_1\mathbb{I}_n + A_1\\ \pi_2\mathbb{I}_n + A_2\\ \vdots\\ \pi_m\mathbb{I}_n + A_m\end{pmatrix} \\ \hline \begin{pmatrix}\pi_1\mathbb{I}_n + A_1\\ \pi_2\mathbb{I}_n + A_2\\ \vdots\\ \pi_m\mathbb{I}_n + A_m\end{pmatrix}^T & \begin{array}{c} 4C \\ +\sum_{i\in I} 2(\pi_i^k\mathbb{I}_n - A_i^k)(\pi_i\mathbb{I}_n - A_i) \\ -\sum_{i\in I}(\pi_i^k\mathbb{I}_n - A_i^k)^2 \end{array} \end{array}\right) \succeq 0. \tag{3.6}$$

Similarly, to derive the first-order approximated inverse SDP problem for the SSP method, at iteration k of the algorithm, we need to replace the nonconvex bilinear terms $\pi_i A_i$ in

constraint (3.4c) with their first-order approximations around point (A^k, π^k) as

$$\sum_{i\in I} \pi_i A_i \approx \sum_{i\in I} \left[\pi_i^k A_i + \pi_i A_i^k - \pi_i^k A_i^k\right] \tag{3.7}$$

Using this, in the first-order approximated inverse problem, constraint (3.4c) is replaced by the following LMI

$$\sum_{i\in I} \left[\pi_i^k A_i + \pi_i A_i^k - \pi_i^k A_i^k\right] \preceq 0. \tag{3.8}$$

To apply the ACS procedure, the set of variables should be partitioned into A and π. This reduces the original inverse problem into two tractable convex SDP subproblems. However, since the variables A do not appear in the objective function, optimizing over A when π is fixed, becomes a feasibility problem, which does not improve the value of objective function in in the inverse problem.

So far, we discussed how the nonconvex inverse SDP problem can be approximately solved using the two proposed heuristics. The following proposition states a variable transformation technique to convert this nonconvex bilinear problem into a tractable SDP when the side constraints have a special form.

Proposition 3.3.1. *Suppose either that there are no side constraints on the values of matrices A_i, or that each A_i independently needs to satisfy the side constraints*

$$F_0^i + \sum_{j=1}^{n}\sum_{k=1}^{n} a_{jk}^i F_{jk}^i \succeq 0, \quad \forall i \in I \tag{3.9a}$$

$$l^i J_n \leq A_i \leq u^i J_n, \quad \forall i \in I, \tag{3.9b}$$

where u^i , $l^i \in \mathbb{R}$ are scalars. Then, the nonconvex inverse SDP problem (3.4) is equivalent

to the SDP

$$\min_{Y,\pi} \ \mathrm{Tr}(C\hat{X}) - \pi^T b \tag{3.10a}$$

$$\mathrm{Tr}(Y_i\hat{X}) \geq b_i\pi_i, \quad \forall i \in I \tag{3.10b}$$

$$\sum_{i\in I} Y_i \preceq C \tag{3.10c}$$

$$\pi \geq 0 \tag{3.10d}$$

$$F_0^i\pi_i + \sum_{j=1}^{n}\sum_{k=1}^{n} y_{jk}^i F_{jk}^i \succeq 0, \quad \forall i \in I \tag{3.10e}$$

$$l^i\pi_i J_n \leq Y_i \leq u^i\pi_i J_n, \quad \forall i \in I. \tag{3.10f}$$

Here, constraints (3.9b) and (3.10f) express element-wise bounds on matrices A_i and Y_i. Using the optimal solution (Y^*, π^*) to this problem, the optimal values of coefficient matrices A_i^* are recovered as

$$A_i^* = \begin{cases} \frac{1}{\pi_i^*} Y_i^*, & \text{if } \pi_i^* > 0 \\ \underset{A_i}{\text{argmin}} \ 0 & \text{if } \pi_i^* = 0 \\ \quad \mathrm{Tr}(A_i\hat{X}) \geq b_i, & \\ \quad F_0^i + \sum_{j=1}^{n}\sum_{k=1}^{n} a_{jk}^i F_{jk}^i \succeq 0 & \\ \quad l^i J_n \leq A_i \leq u^i J_n. & \end{cases}$$

Proof. To establish equivalence of the two problems, we need to show that given a feasible solution to each problem, we can construct a feasible solution to the other problem with the same objective value.

First, let $(\tilde{A}, \tilde{\pi})$ be a feasible solution to problem (3.4) with side constraints of the form

(3.9). Construct a feasible solution $(\bar{Y}, \bar{\pi})$ to problem (3.10) as

$$\bar{\pi} \overset{\text{def}}{=} \tilde{\pi} \tag{3.11}$$

$$\bar{Y}_i \overset{\text{def}}{=} \tilde{\pi}_i \tilde{A}_i, \quad \forall i \in I. \tag{3.12}$$

To check feasibility of $(\bar{Y}, \bar{\pi})$, we plug it back into constraints of problem (3.10) as

Constraint (3.10b): For every $i \in I$

$$\operatorname{Tr}(\bar{Y}_i \hat{X}) - b_i \bar{\pi}_i \overset{(3.11),(3.12)}{=} \operatorname{Tr}(\tilde{\pi}_i \tilde{A}_i \hat{X}) - b_i \tilde{\pi}_i = \tilde{\pi}_i \big(\operatorname{Tr}(\tilde{A}_i \hat{X}) - b_i \big) \overset{(3.4b),(3.4d)}{\geq} 0.$$

Constraint (3.10c): $\sum_{i \in I} \bar{Y}_i \overset{(3.12)}{=} \sum_{i \in I} \tilde{\pi}_i \tilde{A}_i \overset{(3.4c)}{\preceq} C.$

Constraint (3.10e): For every $i \in I$

$$F_0^i \bar{\pi}_i + \sum_{j=1}^{n} \sum_{k=1}^{n} \bar{y}_{jk}^i F_{jk}^i \overset{(3.11),(3.12)}{=} F_0^i \tilde{\pi}_i + \sum_{j=1}^{n} \sum_{k=1}^{n} \tilde{\pi}_i \tilde{a}_{jk}^i F_{jk}^i = \tilde{\pi}_i (F_0^i + \sum_{j=1}^{n} \sum_{k=1}^{n} \tilde{a}_{jk}^i F_{jk}^i) \overset{(3.4d),(3.9a)}{\succeq} 0.$$

Constraints (3.10f): For every $i \in I$.

$$\bar{Y}_i - u^i \bar{\pi}_i J_n \overset{(3.11),(3.12)}{=} \tilde{\pi}_i \tilde{A}_i - u^i \tilde{\pi}_i J_n = \tilde{\pi}_i (\tilde{A}_i - u^i J_n) \overset{(3.4d),(3.9b)}{\leq} 0,$$

$$\bar{Y}_i - l^i \bar{\pi}_i J_n \overset{(3.11),(3.12)}{=} \tilde{\pi}_i \tilde{A}_i - l^i \tilde{\pi}_i J_n = \tilde{\pi}_i (\tilde{A}_i - l^i J_n) \overset{(3.4d),(3.9b)}{\geq} 0.$$

Given definition of $\bar{\pi}$ in (3.11), constraint (3.10d) is trivially satisfied and it is clear that the objective values of problems (3.4) and (3.10) are equal.

Conversely, assume that $(\bar{Y}, \bar{\pi})$ is a feasible solution to problem (3.10). Define the set I' as $I' = \{i \in I : \bar{\pi}_i > 0\}$. Note that given constraint (3.10f), for all $i \in I \setminus I'$, $Y_i = 0_n$. We can construct a feasible solution to the inverse SDP problem (3.4) as

$$\tilde{\pi} \overset{\text{def}}{=} \bar{\pi} \tag{3.13}$$

$$\tilde{A}_i \overset{\text{def}}{=} \frac{1}{\bar{\pi}_i} \bar{Y}_i, \quad \forall i \in I'. \tag{3.14}$$

Also, for $i \in I \setminus I'$, set values of $\tilde{A}_i$ to the solution of the following feasibility SDP problem

$$\max_{A_i} \; 0 \tag{3.15a}$$

$$\mathrm{Tr}(A_i \hat{X}) \geq b_i \tag{3.15b}$$

$$F_0^i + \sum_{j=1}^{n}\sum_{k=1}^{n} a_{jk}^i F_{jk}^i \succeq 0 \tag{3.15c}$$

$$l^i J_n \leq A_i \leq u^i J_n. \tag{3.15d}$$

To show feasibility of $(\tilde{A}, \tilde{\pi})$ to the inverse problem (3.4), we substitute it in the constraints as

Constraint (3.4b): For $i \in I'$

$$\mathrm{Tr}(\tilde{A}_i \hat{X}) - b_i \overset{(3.14)}{=} \mathrm{Tr}(\frac{1}{\bar{\pi}_i}\bar{Y}_i \hat{X}) - b_i = \frac{1}{\bar{\pi}_i}\big(\mathrm{Tr}(\bar{Y}_i \hat{X}) - \bar{\pi}_i b_i\big) \overset{(3.10b),(3.10d)}{\geq} 0.$$

Given problem (3.15), this constraint trivially holds for $i \in I \setminus I'$.

Constraint (3.4c):

$$\sum_{i\in I} \tilde{\pi}_i \tilde{A}_i - C \overset{(3.13)}{=} \sum_{i\in I'} \bar{\pi}_i \tilde{A}_i + \sum_{i\in I\setminus I'} \cancelto{0}{\bar{\pi}_i} \tilde{A}_i - C \overset{(3.14)}{=} \sum_{i\in I'} \bar{\pi}_i \frac{1}{\bar{\pi}_i}\bar{Y}_i - C = \sum_{i\in I'} \bar{Y}_i - C = \sum_{i\in I} \bar{Y}_i - C \overset{(3.10c)}{\preceq} 0.$$

Constraint (3.9a): For $i \in I'$

$$F_0^i + \sum_{j=1}^{n}\sum_{k=1}^{n} \tilde{a}_{jk}^i F_{jk}^i \overset{(3.14)}{=} F_0^i + \sum_{j=1}^{n}\sum_{k=1}^{n} \frac{1}{\bar{\pi}_i}\bar{y}_{jk}^i F_{jk}^i = \frac{1}{\bar{\pi}_i}(F_0^i \bar{\pi}_i + \sum_{j=1}^{n}\sum_{k=1}^{n} \bar{y}_{jk}^i F_{jk}^i) \overset{(3.10d),(3.10e)}{\succeq} 0.$$

Also, given problem (3.15), this constraint holds for $i \in I \setminus I'$.

Constraint (3.9b): For $i \in I'$

$$\tilde{A}_i - u^i J_n \overset{(3.14)}{=} \frac{1}{\bar{\pi}_i}\bar{Y}_i - u^i J_n = \frac{1}{\bar{\pi}_i}(\bar{Y}_i - u^i \bar{\pi}_i J_n) \overset{(3.9b)}{\leq} 0,$$
$$\tilde{A}_i - l^i J_n \overset{(3.14)}{=} \frac{1}{\bar{\pi}_i}\bar{Y}_i - l^i J_n = \frac{1}{\bar{\pi}_i}(\bar{Y}_i - l^i \bar{\pi}_i J_n) \overset{(3.9b)}{\geq} 0.$$

For all $i \in I \setminus I'$, given problem (3.15) for recovering values of A_i, this constraint is satisfied.

Similarly, given definition of $\tilde{\pi}$ in (3.13), it is clear that the objective values of problems (3.4) and (3.10) are equal. This completes the proof. □

3.3.2 Unknown Parameter C, Known Parameter b

When the cost coefficient parameter C is also unknown, given an observed solution $\hat{X} \in S^n_+$ to the primal SDP problem, the inverse SDP problem is

$$\min_{A,\pi,C} \operatorname{Tr}(C\hat{X}) - \pi^T b \tag{3.16a}$$
$$\operatorname{Tr}(A_i\hat{X}) \geq b_i, \quad \forall i \in I \tag{3.16b}$$
$$\sum_{i\in I} \pi_i A_i \preceq C \tag{3.16c}$$
$$\pi \geq 0 \tag{3.16d}$$
$$\sum_{i\in I} \pi_i = 1 \tag{3.16e}$$
$$\sum_{i\in I} [F_0^i + \sum_{j=1}^{n}\sum_{k=1}^{n} a_{jk}^i F_{jk}^i] \succeq 0. \tag{3.16f}$$

Here, constraint (3.16e) is added to ensure that the trivial solution $(\pi, C) = (0_{m,1}, 0_n)$ with objective value 0 is not feasible to the inverse problem. Note that we can add this normalizing constraint because $C \in S^n$ is assumed to be an unconstrained variable. In general, if variable C is assumed to be unconstrained, or it only needs to satisfy a set of homogeneous constraints, we can add a normalizing constraint without loss of optimality.

Lemma 3.3.1. *The inverse SDP problem (3.16) can be trivially solved if there are no side constraints on the coefficient matrices A_i (i.e. if constraints (3.16f) are removed).*

Proof. Let $\hat{A}_i = \frac{b_i}{\mathrm{Tr}(\hat{X})}\mathbb{I}_n$ for all $i \in I$. For any arbitrary $\hat{\pi}$ satisfying (3.16d)-(3.16e) define $\hat{C} = \sum_{i\in I} \hat{\pi}_i \hat{A}_i$. Then $(\hat{A}, \hat{\pi}, \hat{C})$ is feasible to (3.16), and it can be shown that its associated objective value is 0. □

The following proposition shows that the nonconvex inverse SDP problem (3.16) can be reduced to solving m convex optimization problems.

Proposition 3.3.2. *The optimal value of the inverse SDP problem (3.16) denoted by v^* can be expressed as*

$$v^* = \min_{i'\in I} v^*_{i'},$$

where

$$\begin{aligned} v^*_{i'} = \min_{A} \ & \mathrm{Tr}(A_{i'}\hat{X}) - b_{i'} \\ & \mathrm{Tr}(A_i\hat{X}) \geq b_i, \quad \forall i \in I \\ & \sum_{i\in I}[F^i_0 + \sum_{j=1}^{n}\sum_{k=1}^{n} a^i_{jk}F^i_{jk}] \succeq 0. \end{aligned}$$

Proof. Let us rewrite constraint (3.16c) in equality form as $\sum_{i\in I} \pi_i A_i + S = C$ where $S \in S^n_+$ is a slack variable. By substituting this in the objective function we get

$$\mathrm{Tr}(C\hat{X}) - \pi^T b = \mathrm{Tr}\left((\sum_{i\in I} \pi_i A_i + S)\hat{X}\right) - \pi^T b = \sum_{i\in I} \pi_i \big(\mathrm{Tr}(A_i\hat{X}) - b_i\big) + \mathrm{Tr}(S\hat{X}).$$

Using this, the inverse problem (3.16) is reformulated as

$$\min_{A,\pi,S} \sum_{i\in I} \pi_i \big(\operatorname{Tr}(A_i \hat{X}) - b_i \big) + \operatorname{Tr}(S\hat{X}) \tag{3.17a}$$

$$\operatorname{Tr}(A_i \hat{X}) \geq b_i, \quad \forall i \in I \tag{3.17b}$$

$$\pi \geq 0 \tag{3.17c}$$

$$\sum_{i\in I} \pi_i = 1 \tag{3.17d}$$

$$\sum_{i\in I} [F_0^i + \sum_{j=1}^{n} \sum_{k=1}^{n} a_{jk}^i F_{jk}^i] \succeq 0 \tag{3.17e}$$

$$S \succeq 0. \tag{3.17f}$$

It is clear that the optimal S^* is the 0_n matrix or any positive semidefinite matrix that satisfies $\operatorname{Tr}(S^*\hat{X}) = 0$. Furthermore, π^* is the unit vector where the value of π_{i^*} associated with $i^* = \underset{i\in I}{\operatorname{argmin}} \operatorname{Tr}(A_i\hat{X}) - b_i$ is set to 1, and the rest to zero. Therefore, problem (3.17) is equivalent to

$$\min_{i'\in I} \Big\{ \min_{A} \ \operatorname{Tr}(A_{i'}\hat{X}) - b_{i'}$$
$$\operatorname{Tr}(A_i \hat{X}) \geq b_i, \quad \forall i \in I$$
$$\sum_{i\in I} [F_0^i + \sum_{j=1}^{n} \sum_{k=1}^{n} a_{jk}^i F_{jk}^i] \succeq 0 \Big\}.$$

Using this, the optimal value of the cost coefficient is $C^* = A_{i^*}$. □

3.3.3 Known Parameter C, Unknown Parameter b

If the right hand side parameter b is unknown, given a feasible solution $\hat{X} \in S^n_+$ the inverse SDP problem is given by

$$\min_{A,\pi,b} \operatorname{Tr}(C\hat{X}) - \pi^T b \tag{3.18a}$$

$$\operatorname{Tr}(A_i\hat{X}) \geq b_i, \quad \forall i \in I \tag{3.18b}$$

$$\sum_{i\in I} \pi_i A_i \preceq C, \tag{3.18c}$$

$$\pi \geq 0, \tag{3.18d}$$

$$\sum_{i\in I} [F^i_0 + \sum_{j=1}^{n}\sum_{k=1}^{n} a^i_{jk} F^i_{jk}] \succeq 0. \tag{3.18e}$$

Observe that both the objective function (3.18a) as well as the constraints (3.18c) are bilinear.

Lemma 3.3.2. *The inverse SDP problem (3.18) can be trivially solved if there are no side constraints on the coefficient matrices A_i.*

Proof. Let $\hat{\pi} = 1_{m,1}$, $\hat{A}_i = \frac{1}{m}C$, $\hat{b}_i = \frac{1}{m}\operatorname{Tr}(C\hat{X})$ for all $i \in I$. Then $(\hat{A}, \hat{\pi}, \hat{b})$ is feasible to (3.18) with objective value of 0. □

The following proposition provides a bilinear reformulation of this inverse problem over variables A and π.

Proposition 3.3.3. *Problem (3.18) is equivalent to the following bilinear SDP*

$$\min_{A,\pi} \operatorname{Tr}\big((C - \sum_{i\in I} \pi_i A_i)\hat{X}\big) \tag{3.19a}$$

$$\sum_{i\in I} \pi_i A_i \preceq C, \tag{3.19b}$$

$$\pi \geq 0, \tag{3.19c}$$

$$\sum_{i\in I} [F_0^i + \sum_{j=1}^{n}\sum_{k=1}^{n} a_{jk}^i F_{jk}^i] \succeq 0. \tag{3.19d}$$

Proof. Let us rewrite constraints (3.18b) in equality form as $\operatorname{Tr}(A_i\hat{X}) - s_i = b_i$ by introducing the new slack variable $s \in \mathbb{R}_+^m$. By substituting these constraints in the objective function we get

$$\operatorname{Tr}(C\hat{X}) - \pi^T b = \operatorname{Tr}(C\hat{X}) - \sum_{i\in I} \pi_i\big(\operatorname{Tr}(A_i\hat{X}) - s_i\big) = \operatorname{Tr}\big((C - \sum_{i\in I} \pi_i A_i)\hat{X}\big) - \sum_{i\in I} \pi_i s_i.$$

Since the slack variables only need to satisfy the nonnegativity constraints $s \geq 0$, it is clear that in the optimal solution, for any value of π^*, the optimal value of s^* would be $0_{m,1}$ or any vector in $\mathbb{R}_+^m$ that satisfies $\sum_{i\in I} \pi_i^* s_i^* = 0$. Therefore, without loss of optimality, we can remove this term from the objective function of inverse SDP problem, and write it as $\operatorname{Tr}\big((C - \sum_{i\in I} \pi_i A_i)\hat{X}\big)$. □

To apply the CCP method, the bilinear constraint (3.19b) and objective function (3.19a) should be convexified. We showed earlier in (3.6) how to convexify constraint (3.19b). To convexify the objective function (3.19a) we first write it in the DC form

$$\operatorname{Tr}\big((C - \sum_{i\in I} \pi_i A_i)\hat{X}\big) = \operatorname{Tr}(C\hat{X}) - \operatorname{Tr}(\sum_{i\in I} \pi_i A_i\hat{X}) = \operatorname{Tr}(C\hat{X}) - \sum_{i\in I} \operatorname{Tr}(\pi_i A_i\hat{X}) =$$
$$\operatorname{Tr}(C\hat{X}) - \sum_{i\in I} \operatorname{Tr}(\pi_i\hat{X}A_i) = \operatorname{Tr}(C\hat{X}) - \frac{1}{4}\sum_{i\in I}\Big[\operatorname{Tr}\big((\pi_i\hat{X} + A_i)^2\big) - \operatorname{Tr}\big((\pi_i\hat{X} - A_i)^2\big)\Big].$$

By replacing the nonconvex piece with the first-order approximation around point (A^k, π^k)

at iteration k of CCP, the objective function is replaced by the following convex function

$$\mathrm{Tr}(C\hat{X}) - \frac{1}{4}\sum_{i\in I}\Big[2\,\mathrm{Tr}\left((\pi_i\hat{X}+A_i)(\pi_i^k\hat{X}+A_i^k)\right) - \mathrm{Tr}\left((\pi_i^k\hat{X}+A_i^k)^2\right) - \mathrm{Tr}\left((\pi_i\hat{X}-A_i)^2\right)\Big]. \tag{3.20}$$

Similarly, we showed earlier in (3.8) how to write the first-order approximation of constraint (3.19b) for the SSP method. The first-order approximation of the objective function (3.19a) around point (A^k, π^k) is

$$\mathrm{Tr}(C\hat{X}) - \sum_{i\in I}\mathrm{Tr}\left((\pi_i^k A_i + \pi_i A_i^k - \pi_i^k A_i^k)\hat{X}\right). \tag{3.21}$$

Finally, the ACS procedure for this inverse problem is implemented by defining two subproblems wherein one variable A is fixed and in the second one the variable π is fixed. Note that both of these subproblems are convex SDP problems.

So far we discussed how to solve the inverse problem with a general form LMI side constraint. The following proposition provides a tractable SDP reformulation of problem (3.19) when the side constraints have the special form as in (3.9).

Proposition 3.3.4. *If the side constraints are of the form (3.9), the inverse SDP problem*

(3.19) is equivalent to the SDP

$$\min_{Y,\pi} \operatorname{Tr}(C\hat{X}) - \sum_{i\in I} \operatorname{Tr}(Y_i\hat{X}) \tag{3.22a}$$

$$\sum_{i\in I} Y_i \preceq C \tag{3.22b}$$

$$\pi \geq 0 \tag{3.22c}$$

$$F_0^i\pi_i + \sum_{j=1}^{n}\sum_{k=1}^{n} y_{jk}^i F_{jk}^i \succeq 0, \quad \forall i \in I \tag{3.22d}$$

$$l^i\pi_i J_n \leq Y_i \leq u^i\pi_i J_n, \quad \forall i \in I. \tag{3.22e}$$

The values of A_i^* are recovered using the optimal solution of this problem either as $\frac{1}{\pi_i^*}Y_i^*$ if $\pi_i^* > 0$, or by solving a feasibility problem consisting of constraints (3.9) for i.

Proof. The proof is similar to the proof of Proposition 3.3.1. □

Note that after finding optimal values of matrices A_i^* using Proposition 3.3.4, values of b_i^* can be recovered as $b_i^* = \operatorname{Tr}(A_i^*\hat{X})$, for all $i \in I$.

3.4 Inverse SDP problem given $\hat{\pi}$

In this subsection, we study three variants of the inverse SDP problem when a solution $\hat{\pi} \in \mathbb{R}_+^m$ to the dual SDP problem (3.2) is given.

3.4.1 Known Parameters C and b

When both parameters C and b are known, given a solution $\hat{\pi} \in \mathbb{R}^m_+$ to the dual problem, the inverse SDP problem is formulated as

$$\min_{A,X} \operatorname{Tr}(CX) - \hat{\pi}^T b \tag{3.23a}$$

$$\sum_{i\in I} \hat{\pi}_i A_i \preceq C, \tag{3.23b}$$

$$\operatorname{Tr}(A_i X) \geq b_i, \quad \forall i \in I \tag{3.23c}$$

$$X \succeq 0, \tag{3.23d}$$

$$\sum_{i\in I} [F_0^i + \sum_{j=1}^{n}\sum_{k=1}^{n} a_{jk}^i F_{jk}^i] \succeq 0. \tag{3.23e}$$

Observe that this is a nonconvex program with bilinear constraints (3.23c). Note that the term $\hat{\pi}^T b$ is the objective function is a constant.

To apply CCP to the inverse SDP problem (3.23), the nonconvex constraints (3.23c) are written in the DC form

$$\operatorname{Tr}\left((A_i + X)^2\right) - \operatorname{Tr}\left((A_i - X)^2\right) \geq 4b_i, \quad \forall i \in I. \tag{3.24}$$

At iteration k of CCP, this is convexified by approximating the nonconvex terms around point (A^k, X^k) as

$$2\operatorname{Tr}\left((A_i + X)(A_i^k + X^k)\right) - \operatorname{Tr}\left((A_i^k + X^k)^2\right) - \operatorname{Tr}\left((A_i - X)^2\right) \geq 4b_i, \quad \forall i \in I \tag{3.25}$$

Similarly, to implement the SSP procedure, at each iteration k, constraint (3.23c) is replaced by its first-order approximation around (A^k, X^k) as

$$\operatorname{Tr}(A_i^k X) + \operatorname{Tr}(A_i X^k) - \operatorname{Tr}(A_i^k X^k) \geq b_i, \quad \forall i \in I. \tag{3.26}$$

Similar to subsection 3.3.1, because the variables A_i do not appear in the objective function, the ACS procedure is not applicable to this variant.

3.4.2 Unknown Parameter C, Known Parameter b

Given a solution $\hat{\pi}$, when parameter C is also unknown, the inverse SDP problem is defined over the set of variables A, X, C as

$$\min_{A,X,C} \operatorname{Tr}(CX) - \hat{\pi}^T b \tag{3.27a}$$

$$\sum_{i\in I} \hat{\pi}_i A_i \preceq C \tag{3.27b}$$

$$\operatorname{Tr}(A_i X) \geq b_i, \quad \forall i \in I \tag{3.27c}$$

$$X \succeq 0 \tag{3.27d}$$

$$\sum_{i\in I} [F_0^i + \sum_{j=1}^{n}\sum_{k=1}^{n} a_{jk}^i F_{jk}^i] \succeq 0. \tag{3.27e}$$

Observe that this is a nonconvex program because of the bilinear objective function (3.27a) and the bilinear constraints (3.27c).

Lemma 3.4.1. *If there are no side constraints on the values of A_i, the inverse SDP problem (3.27) can be trivially solved.*

Proof. For any arbitrary $\hat{X} \succeq 0$ let $\hat{A}_i = \frac{b_i}{\operatorname{Tr}(\hat{X})}\mathbb{I}_n$, $\hat{C} = \sum_{i\in I} \hat{\pi}_i \hat{A}_i$. $(\hat{A}, \hat{X}, \hat{C})$ is a feasible solution to the inverse SDP problem (3.28) with objective function value of 0. □

The following proposition states a new reformulation of the inverse SDP problem (3.27) over variables (A, X) and by removing variable C.

Proposition 3.4.1. *Inverse SDP problem (3.27) is equivalent to the following problem*

$$\min_{A,X} \sum_{i\in I} \hat{\pi}_i\big(\operatorname{Tr}(A_i X) - b_i\big) \tag{3.28a}$$

$$\operatorname{Tr}(A_i X) \geq b_i, \quad \forall i \in I \tag{3.28b}$$

$$X \succeq 0 \tag{3.28c}$$

$$\sum_{i\in I}[F_0^i + \sum_{j=1}^{n}\sum_{k=1}^{n} a_{jk}^i F_{jk}^i] \succeq 0. \tag{3.28d}$$

Proof. The proof can be established similar to proof of Proposition 3.3.2 by introducing a slack variable and rewriting constraint (3.27b) in equality form, and replacing it in the objective function. □

Observe that both the objective function (3.28a) and constraints (3.28b) of the inverse SDP problem are bilinear in variables A_i and X. We showed earlier in (3.24) and (3.26) how to convexify constraints (3.28b) to be used in the CCP and SSP algorithms. Similarly, to convexify the objective function for the CCP heuristic, it is first written in the DC form

$$\frac{1}{4}\sum_{i\in I} \hat{\pi}_i\Big[\operatorname{Tr}\big((A_i + X)^2\big) - \operatorname{Tr}\big((A_i - X)^2\big)\Big] - \hat{\pi}^T b.$$

It is then approximated with the following convex form at iteration k of the algorithm

$$\frac{1}{4}\sum_{i\in I} \hat{\pi}_i\Big[\operatorname{Tr}\big((A_i + X)^2\big) - 2\operatorname{Tr}\big((A_i - X)(A_i^k - X^k)\big) + \operatorname{Tr}\big((A_i^k - X^k)^2\big)\Big] - \hat{\pi}^T b. \tag{3.29}$$

The first-order approximation of the objective function for using in the SSP algorithm is defined as

$$\sum_{i\in I} \hat{\pi}_i\big[\operatorname{Tr}(A_i^k X) + \operatorname{Tr}(A_i X^k) - \operatorname{Tr}(A_i^k X^k)\big] - \hat{\pi}^T b. \tag{3.30}$$

Finally, to apply the ACS procedure to the nonconvex inverse SDP problem (3.28), the two

subproblems are defined by partitioning the variables set into A and X.

3.4.3 Known Parameter C, Unknown Parameter b

When the parameters b are also unknown, given a solution $\hat{\pi}$, the inverse SDP problem can be formulated over the set of variables A, X, and b as

$$\min_{A,X,b} \operatorname{Tr}(CX) - \hat{\pi}^T b \tag{3.31a}$$

$$\sum_{i \in I} \hat{\pi}_i A_i \preceq C \tag{3.31b}$$

$$\operatorname{Tr}(A_i X) \geq b_i, \quad \forall i \in I \tag{3.31c}$$

$$X \succeq 0, \tag{3.31d}$$

$$\sum_{i \in I} [F_0^i + \sum_{j=1}^{n} \sum_{k=1}^{n} a_{jk}^i F_{jk}^i] \succeq 0, \tag{3.31e}$$

$$\operatorname{Tr}(X) = 1. \tag{3.31f}$$

The normalizing constraint (3.31f) is added to ensure $(X, b) = (0_n, 0_{m,1})$ with objective value of 0 is not feasible to the inverse problem. Note than the normalizing constraint is added without loss of optimality, given that the variable b is unconstrained. In general, this can be either if b is unconstrained or needs to satisfy side constraints of the homogeneous form.

The following proposition states a reformulation of this inverse problem by eliminating variable b.

Proposition 3.4.2. *Inverse SDP problem (3.31) is equivalent to the following problem*

$$\min_{A,X} \operatorname{Tr}\left((C - \sum_{i\in I} \hat{\pi}_i A_i)X\right) \tag{3.32a}$$

$$\sum_{i\in I} \hat{\pi}_i A_i \preceq C \tag{3.32b}$$

$$X \succeq 0, \tag{3.32c}$$

$$\sum_{i\in I}[F_0^i + \sum_{j=1}^{n}\sum_{k=1}^{n} a_{jk}^i F_{jk}^i] \succeq 0, \tag{3.32d}$$

$$\operatorname{Tr}(X) = 1. \tag{3.32e}$$

Proof. The proof can be established similar ot the proof of Proposition 3.3.3. □

Observe that problem (3.32) is also a nonconvex program with bilinear objective function and linear constraints as well as linear matrix inequalities.

Lemma 3.4.2. *If there are no side constraints on the values of A_i, or if the set $\left\{A | \sum_{i\in I} \hat{\pi}_i A_i = C, \sum_{i\in I}[F_0^i + \sum_{j=1}^{n}\sum_{k=1}^{n} a_{jk}^i F_{jk}^i] \succeq 0\right\}$ is nonempty, the inverse SDP problem (3.32) can be trivially solved.*

Proof. If there exist a feasible $\hat{A}$ with respect to the side constraints which satisfies constraint (3.32b) as equality, for any arbitrary $\hat{X} \succeq 0$ satisfying (3.32e), $(\hat{A}, \hat{X})$ would be a trivial solution to the inverse problem with objective value of 0. If there are no side constraints on A, for all $i \in I$, define $\hat{A}_i = \begin{cases} \frac{C}{m'\hat{\pi}_i}, & \text{if } \hat{\pi}_i > 0 \\ \text{An arbitrary matrix} \in S^n, & \text{otherwise} \end{cases}$ where $m' = \sum_{i\in I} \mathbb{1}_{\{\hat{\pi}_i > 0\}}$. For any arbitrary $\hat{X}$ satisfying constraints (3.32c) and (3.32e), the solution $(\hat{A}, \hat{X})$ is a trivially feasible solution to the inverse SDP problem with objective function value of 0. □

To apply CCP and SSP algorithms to the inverse SDP problem (3.32) we only need to replace the objective function with a convex function. For CCP algorithm, we first rewrite

the objective in a DC form as

$$\begin{aligned}\operatorname{Tr}\big((C-\sum_{i\in I}\hat{\pi}_i A_i)X\big) &= \operatorname{Tr}(CX)-\sum_{i\in I}\hat{\pi}_i\operatorname{Tr}(A_iX)\\ &= \operatorname{Tr}(CX)-\frac{1}{4}\sum_{i\in I}\hat{\pi}_i\Big[\operatorname{Tr}\big((A_i+X)^2\big)-\operatorname{Tr}\big((A_i-X)^2\big)\Big].\end{aligned}$$

Then replace the nonconvex terms with their first-order approximations around (A^k, X^k) as

$$\operatorname{Tr}(CX)-\frac{1}{4}\sum_{i\in I}\hat{\pi}_i\Big[2\operatorname{Tr}\big((A_i+X)(A_i^k+X^k)\big)-\operatorname{Tr}\big((A_i^k+X^k)^2\big)-\operatorname{Tr}\big((A_i-X)^2\big)\Big]. \tag{3.33}$$

The first-order approximation of the objective function around (A^k, X^k) for the SSP algorithm is

$$\operatorname{Tr}(CX)-\sum_{i\in I}\hat{\pi}_i\big[\operatorname{Tr}(A_i^kX)+\operatorname{Tr}(A_iX^k)-\operatorname{Tr}(A_i^kX^k)\big]. \tag{3.34}$$

Lastly, to apply the ACS procedure, we define two convex SDP subproblems, one with fixed A values and the other with fixed X values.

The table below summarizes our methodological findings about the six variants.

	Given $\hat{X}$	Given $\hat{\pi}$
Known C,b	Nonconvex program with BMIs/SDP if side constraints on A_i's are of form (3.3)	Nonconvex program with bilinear constraints
Unknown C, known b	m convex SDPs	Nonconvex program with bilinear objective and bilinear constraints
Known C, unknown b	Nonconvex program with bilinear objective and BMIs/SDP if side constraints on A_i's are of form (3.3)	Nonconvex program with bilinear objective

Table 3.1: Variants of the inverse SDP problem with unknown constraint parameters

Note that one can define the inverse SDP problem with all parameters A, b, C unknown. Although we did not include this variant, all heuristics employed in this chapter can be applied using the same convexification and approximation techniques.

3.5 Numerical Results

In the following subsections, we present computational results of the discussed heuristics for solving the inverse SDP variants with nonconvex formulation. We study four problem sets, with varying values of parameters (m, n). Recall that parameter $b \in \mathbb{R}^m$, and the coefficient matrices $A_1, \cdots, A_m, C \in S^n$. For each problem set, we generate 30 random instances. The maximum runtime for the algorithms is set as n minutes. Similar to chapter 2, we will stop the algorithms if they obtain the same objective value for 10 consecutive iterations. We will use the same metric defined in (2.21) to measure the percentage of

improvement in the objective function value obtained by each of the heuristics. For the matter of simplicity, we call this metric $DGIP$ (Duality Gap Improvement Percentage) in the rest of this section. Since the total runtime of the algorithms might depend on the defined stopping criteria, we will use the time at which each heuristic finds its best objective value as another metric to compare performance of the solution algorithms. We denote this time by t^*. All mathematical programs were solved using the cvxpy package on Python 3.8.

Throughout this section, we assume that the coefficient matrices A need to satisfy side constraints of the following form

$$A_i \succeq 0, \quad \forall i \in I \tag{3.35a}$$

$$\sum_{i \in I} w_i A_i \succeq 0 \tag{3.35b}$$

$$l_i J_n \leq A_i \leq u_i J_n, \quad \forall i \in I \tag{3.35c}$$

where parameter $w \in \mathbb{R}^m$ is sampled from a discrete uniform distribution on [-1,1]. If all the generated values of w_i are non-positive, constraint (3.35b) will have a conflict with constraint (3.35a), resulting in an infeasible set of values for A. In such cases, we randomly select one of the negative values and replace it with its absolute value. We set values of l_i and u_i to -100 and 100, respectively. Recall that all the heuristics require initial feasible values for the coefficient matrices. We generate the initial matrices A_i^0 by solving a feasibility problem with constraints (3.35). We will discuss how parameters b, C and the SDP solution ($\hat{X}$ or $\hat{\pi}$) are generated separately in each subsection. In cases where parameters C or b are unknown, we assume they are unconstrained variables in the inverse problem.

3.5.1 Given $\hat{X}$

In this subsection, we provide computational results for the two variants discussed in 3.3.1 and 3.3.3 where the inverse SDP problem is a nonconvex bilinear problem with joint constraints on values of A_i's. To generate a PSD initial solution $\hat{X}$, we first sample an $n \times n$

matrix denoted by E from a uniform $[-10, 10]$ distribution. The primal solution $\hat{X}$ is then generated as $\hat{X} = EE^T$. For the inverse problems discussed in 3.3.1 where it is assumed that the right hand side parameter is known, we generate it as $b_i = \text{Tr}(A_i^0 \hat{X}) - s_i$ for all $i \in I$, where the slack variable s_i is sampled from a $[0, 10]$ uniform distribution. This ensures the generated initial coefficient matrices A_i^0 are feasible with respect to the inverse SDP constraint (3.4b). Similarly, to generate the cost parameter C, we need to ensure that the generated initial A_i^0's are feasible with respect to constraint (3.4c), for some feasible value of the dual variable π. To do this, we set value of C to $\sum_{i \in I} \bar{\pi}_i A_i^0 + S$ where $\bar{\pi} \in \mathbb{R}^m$ is a feasible solution to the dual SDP problem sampled from a uniform $[0, 10]$ distribution. The slack matrix S is generated by sampling an $n \times n$ matrix G from a uniform $[-10, 10]$ distribution and setting value of S to $S = GG^T$.

As discussed earlier, we add trust region constraints to the approximated inverse SDP problems solved by the SSP algorithm. In iteration k of the SSP algorithm, we use trust region constraints of the following form

$$\|A_i - A_i^{k-1}\|_1 \leq \rho_A, \quad \forall i \in I$$
$$\|\pi - \pi^{k-1}\|_1 \leq \rho_\pi$$

In our experiments, we define the trust region radii as $\rho_A = n^2$, and $\rho_\pi = m$.

Results of our simulations for the variants with known values of b and C studied in subsection 3.3.1 are presented in Table 3.2. The table shows that both the CCP and SSP methods reached the optimal objective value of 0 for each problem instance. However, the values of t^* suggest that the CCP method is significantly faster than SSP in solving inverse SDP problems of this variant.

	CCP		SSP	
(m,n)	$DGIP$	$t^*(sec.)$	$DGIP$	$t^*(sec.)$
(2,5)	100	0.22	100	5.28
(3,8)	100	0.5	100	39.71
(4,10)	100	0.86	100	116.67
(5,13)	100	1.96	100	155.92

Table 3.2: Performance of CCP and SSP on 4 sets of inverse SDP problems with given primal SDP solution $\hat{X}$, unknown constraint parameters A, and known b and C parameters. Each set included 30 instances.

Similarly, Table 3.3 presents computational results for the inverse SDP variant discussed in subsection 3.3.3, where the constraints' right hand side parameters are also unknown. Recall that the alternate convex search (ACS) method is applicable to this inverse SDP problem variant.

	CCP		SSP		ACS	
(m,n)	$DGIP$	$t^*(sec.)$	$DGIP$	$t^*(sec.)$	$DGIP$	$t^*(sec.)$
(2,5)	100	0.18	100	0.15	100	0.06
(3,8)	100	48.89	100	0.42	100	0.1
(4,10)	100	106	100	2.3	100	0.18
(5,13)	100	134.26	100	126.16	100	17.35

Table 3.3: Performance of CCP, SSP, and ACS on 4 sets of inverse SDP problems with given primal SDP solution $\hat{X}$, unknown constraint parameters A and b, and known C parameters. Each set included 30 instances.

The results show that in all instances, all three methods reached the optimal objective value of 0. Unlike the previous variant where the CCP method was the fastest, in these problem instances CCP is the slowest method while the ACS methods appears to be the

fastest method to converge to the optimal value. One possible explanation for the better performance of simpler methods such as SSP and ACS versus the more complicated method of CCP in problem instances of this variant could be the following; Due to the larger degrees of freedom in the problem caused by having unknown right hand side parameters, the problem is more likely to have multiple optimal solutions making it easier for the solution algorithms to find an optimal solution.

3.5.2 Given $\hat{\pi}$

In this subsection, we will present computational results for the three variants of the inverse SDP problems discussed in 3.4.1, 3.4.2, and 3.4.3. The non-negative initial solution $\hat{\pi}$ is sampled from a uniform distribution over [0,10]. For the inverse problems discussed in 3.4.1 and 3.4.3 with known cost coefficient parameter C, we generate it similar to the previous subsection as $C = \sum_{i \in I} \hat{\pi}_i A_i^0 + S$, where S is a PSD slack variable generated in the same way. Again, we generate the right hand side parameter for variants requiring known value of b similar to the previous subsection as $b_i = \text{Tr}(A_i^0 \bar{X}) - s_i$ for all $i \in I$. Here, $\bar{X} \succeq 0$ is a PSD feasible solution for the primal SDP problem. It is generated by sampling an $n \times n$ matrix E from the uniform [-10,10] distribution, and setting $\bar{X}$ to EE^T. We define the trust region constraints for the SSP algorithm as

$$\|A_i - A_i^{k-1}\|_1 \leq \rho_A, \quad \forall i \in I$$
$$\|X - X^{k-1}\|_1 \leq \rho_X,$$

where we define the trust region radii as $\rho_A = \rho_X = n^2$.

Table 3.4 shows the computational results of the SSP and CCP algorithms on inverse SDP instances studied in 3.4.1. The results show that CCP converges to the optimal objective value of 0 in all instances, while SSP does not reach the optimal solution, on average. Furthermore, CCP outperforms the SSP method in terms of the values of t^*, making it the strictly better algorithm in solving inverse SDP problems of this variant.

	CCP		SSP	
(m,n)	$DGIP$	$t^*(sec.)$	$DGIP$	$t^*(sec.)$
(2,5)	100	0.74	84.46	39.32
(3,8)	100	1.19	94.81	24.93
(4,10)	100	0.68	97.82	18.93
(5,13)	100	1.43	97.47	36.39

Table 3.4: Performance of CCP and SSP on 4 sets of inverse SDP problems with given primal SDP solution $\hat{\pi}$, unknown constraint parameters A, and known b and C parameters. Each set included 30 instances.

Table 3.5 shows the results of the three discussed heuristics on inverse problem variants of subsection 3.4.2. The table shows that while all methods converge to the optimal solution, ACS outperforms the other two methods in terms of runtime.

	CCP		SSP		ACS	
(m,n)	$DGIP$	$t^*(sec.)$	$DGIP$	$t^*(sec.)$	$DGIP$	$t^*(sec.)$
(2,5)	100	0.2	99.99	39.42	100	0.06
(3,8)	100	0.3	100	49.94	100	0.1
(4,10)	100	0.36	100	32.69	100	0.16
(5,13)	100	0.51	100	59.69	100	0.31

Table 3.5: Performance of CCP, SSP, and ACS on 4 sets of inverse SDP problems with given primal SDP solution $\hat{\pi}$, unknown constraint parameters A and C, and known b parameters. Each set included 30 instances.

Finally, Table 3.6 shows the results of numerical experiments on the inverse problems of variant 3.4.3. Again, while all methods obtain the optimal solution in all instances, ACS appears as the winning solution algorithm in terms of convergence time.

	CCP		SSP		ACS	
(m,n)	$DGIP$	$t^*(sec.)$	$DGIP$	$t^*(sec.)$	$DGIP$	$t^*(sec.)$
(2,5)	100	0.16	100	0.16	100	0.06
(3,8)	100	0.2	100	0.19	100	0.1
(4,10)	100	0.25	100	0.42	100	0.15
(5,13)	100	0.42	100	0.72	100	0.28

Table 3.6: Performance of CCP, SSP, and ACS on 4 sets of inverse SDP problems with given primal SDP solution $\hat{\pi}$, unknown constraint parameters A and b, and known C parameters. Each set included 30 instances.

3.6 Conclusions

This chapter developed an inverse optimization framework for recovering constraint parameters of an SDP. We studied six variants of this problem. We showed that when a solution to the primal SDP problem is given, and the cost coefficients are also unknown, the inverse problem reduces to a polynomial number of convex SDP problems. When the cost coefficients are known and the side constraints have a special form, we showed that the inverse SDP problem reduces to solving an SDP using a variable transformation technique. In the rest of the cases, the inverse SDP cannot be rewritten in a tractable form. We thus applied CCP, SSP, and ACS heuristics to obtain approximate solutions. We compared their performance via numerical experiments. Our computational experiments showed that the CCP algorithm outperforms SSP in inverse SDP variants with fixed value of b and C discussed in 3.3.1 and 3.4.1. However, in the other 3 variants where the ACS method is applicable (variants 3.3.3, 3.3.2, and 3.4.3), while all methods converge to the optimal solution in all experimented problem instances, ACS has the best performance in terms of convergence time.

The objective in the inverse problems in this chapter was to minimize the duality gap associated with a given solution. If the parameter values discovered by any algorithm satisfy strong duality, there in fact may be multiple values of those parameters that render the

given solution optimal. In such cases, it might be interesting in the future to minimize some measure of distance from "nominal" parameter values, subject to strong duality. It would also be interesting to apply the methodology developed here to inverse versions of specific SDPs such as the ones that arise in maximum eigenvalue minimization problems or in minimum volume covering ellipsoid problems [79].

BIBLIOGRAPHY

[1] D Adelman and A J Mersereau. Relaxations of weakly coupled stochastic dynamic programs. *Operations Research*, 56(3):712–727, 2008.

[2] R K Ahuja and J B Orlin. Inverse optimization. *Operations Research*, 49(5):771–783, 2001.

[3] F A Al-Khayyal. Generalized bilinear programming: Part I. models, applications and linear programming relaxation. *European Journal of Operational Research*, 60(3):306–314, 1992.

[4] F Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5(1):13–51, 1995.

[5] F Alizadeh and D Goldfarb. Second-order cone programming. *Mathematical Programming, Ser. B*, 95:3–51, 2003.

[6] D Astaraky and J Patrick. A simulation based approximate dynamic programming approach to multi-class, multi-resource surgical scheduling. *European Journal of Operational Research*, 245(1):309–319, 2015.

[7] T Ayer. Inverse optimization for assessing emerging technologies in breast cancer screening. *Annals of Operations Research*, 230(1):57–85, 2015.

[8] D R Beil and L M Wein. An inverse optimization based mechanism to support a multiattribute rfq process. *Management Science*, 49(11):1529–1545, 2003.

[9] A Ben-Tal, G Eiger, and V Gershovitz. Global minimization by reducing the duality gap. *Mathematical Programming*, 63(1-3):193–212, 1994.

[10] D P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[11] D P Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, Belmont, MA, USA, 2000.

[12] D Bertsimas, V Gupta, and I C Paschalidis. Inverse optimization: a new perspective on the Black-Litterman model. *Operations Research*, 60(6):1389–1403, 2012.

[13] D Bertsimas, V Gupta, and I C Paschalidis. Data-driven estimation in equilibrium using inverse optimization. *Mathematical Programming*, 153(2):595–633, 2015.

[14] D Bertsimas and J Niño-Mora. Conservation laws, extended polymatroids and multi-armed bandit problems; a polyhedral approach to indexable systems. *Mathematics of Operations Research*, 21(2):257–306, 1996.

[15] J R Birge, A Hortaçsu, and J M Pavlin. Inverse optimization for the recovery of market structure from market outcomes: An application to the miso electricity market. *Operations Research*, 65(4):837–855, 2017.

[16] R Boucherie and N M van Dijk. *Markov Decision Processes in Practice*. Springer, New York, NY, USA, 2017.

[17] D Bouneffouf and I Rish. A survey on practical applications of multi-armed and contextual bandits. https://arxiv.org/abs/1904.10040, 2019.

[18] S Boyd and L Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 1st edition, 2004.

[19] P Brucker and N V Shakhlevich. Inverse scheduling with maximum lateness objective. *Journal of Scheduling*, 12(5):475–488, 2009.

[20] D Burton and P L Toint. On an instance of inverse shortest path problems. *Mathematical Programming*, 53(1-3):45–61, 1992.

[21] D Burton and P L Toint. On the inverse shortest path algorithm for recovering linearly correlated costs. *Mathematical Programming*, 63(1-3):1–22, 1994.

[22] F Caro and A D Gupta. Robust control of the multi-armed bandit problem. *Annals of Operations Research*, pages 1–20, 2015.

[23] S Carr and W Lovejoy. The inverse newsvendor problem: Choosing an optimal demand portfolio for capacitated resources. *Management Science*, 46(7):912–927, 2000.

[24] T C Y Chan, T Craig, T Lee, and M B Sharpe. Generalized inverse multiobjective optimization with application to cancer therapy. *Operations Research*, 62(3):680–695, 2014.

[25] T C Y Chan and N Kaw. Inverse optimization for the recovery of constraint parameters. *European Journal of Operational Research*, 282(2):415–427, 2020.

[26] T C Y Chan and T Lee. Trade-off preservation in inverse multi-objective convex optimization. *European Journal of Operational Research*, 270(1):1, 2018.

[27] T C Y Chan, T Lee, T Craig, and M B Sharpe. Determining objective function weights in prostate IMRT using inverse optimization. *Medical Physics*, 38(6):3687, 2011.

[28] T C Y Chan, T Lee, and D Terekhov. Inverse optimization: Closed-form solutions, geometry, and goodness of fit. *Management Science*, 65(3):1115–1135, 2019.

[29] A Charnes and W W Cooper. Chance constrained programming. *Management Science*, 6(1):73–79, 1959.

[30] P C Chu and J E Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4(1):63–86, 1998.

[31] E Delage and S Mannor. Percentile optimization for Markov decision processes with parameter uncertainty. *Operations Research*, 58(1):203–213, 2010.

[32] R Dial. Minimal-revenue congestion pricing part I: A fast algorithm for the single-origin case. *Transportation Research Part B: Methodological*, 33(3):189–202, 1999.

[33] R Dial. Minimal revenue congestion pricing. Part II: an efficient algorithm for the general case. *Transportation Research Part B: Methodological*, 34(8):645–665, 2000.

[34] D A Dolgov and E H Durfee. Optimal resource allocation and policy formulation in loosely-coupled Markov decision processes. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 315–324, Whistler, BC, Canada, 2004.

[35] Z Erkin, M D Bailey, L M Maillart, A J Schaefer, and M S Roberts. Eliciting patients' revealed preferences: An inverse Markov decision process approach. *Decision Analysis*, 7(4):358–365, 2010.

[36] C A Floudas and V Visweswaran. Primal-relaxed dual global optimization approach. *Journal of Optimization Theory and Applications*, 78(2):187–225, 1993.

[37] B Gebken and S Peitz. Inverse multiobjective optimization: Inferring decision criteria from data. *Journal of Global Optimization*, 80:3–29, January 2021.

[38] A M Geoffrion. Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260, 1972.

[39] A Ghate. Inverse optimization in countably infinite linear programs. *Operations Research Letters*, 43(3):231–235, 2015.

[40] A Ghate. Imputing radiobiological parameters of the linear-quadratic dose-response model from a radiotherapy fractionation plan. *Physics in Medicine & Biology*, 65(22):225009, 2020.

[41] A Ghate. Inverse optimization in semi-infinite linear programs. *Operations Research Letters*, 48(3):278–285, 2020.

[42] K Ghobadi and H Mahmoudzadeh. Multi-point inverse optimization of constraint parameters. *arXiv preprint arXiv:2001.00143*, 2020.

[43] J C Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(2):148–164, 1979.

[44] Y Gocgun. *Approximate dynamic programming for dynamic stochastic resource allocation with applications to healthcare*. PhD thesis, University of Washington, Seattle, WA, USA, 2010.

[45] Y Gocgun and A Ghate. A Lagrangian approach to dynamic resource allocation. In B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yucesan, editors, *Proceedings of the Winter Simulation Conference*, pages 3330–3338, Baltimore, 2010.

[46] Y Gocgun and A Ghate. Lagrangian relaxation and constraint generation for allocation and advance scheduling. *Computers & Operations Research*, 39(10):2323–2336, 2012.

[47] M X Goemans. Semidefinite programming in combinatorial optimization. *Mathematical Programming*, 79(1):143–161, 1997.

[48] J Gorski, F Pfeuffer, and K Klamroth. Biconvex sets and optimization with biconvex functions: a survey and extensions. *Mathematical Methods of Operations Research*, 66(3):373–407, 2007.

[49] R E Griffith and R A Stewart. A nonlinear programming technique for the optimization of continuous processing systems. *Management Science*, 7(4):379–392, 1961.

[50] J T Hawkins. *A Langrangian decomposition approach to weakly coupled dynamic optimization problems and its applications*. PhD thesis, Massachusetts Institute of Technology, 2003.

[51] C Helmberg. Semidefinite programming. *European Journal of Operational Research*, 137(3):461–482, 2002.

[52] C Heuberger. Inverse combinatorial optimization: A survey on problems, methods, and results. *Journal of Combinatorial Optimization*, 8(3):329–361, 2004.

[53] G Iyengar and W Kang. Inverse conic programming with applications. *Operations Research Letters*, 33(3):319–330, 2005.

[54] G N Iyengar. Robust dynamic programming. *Mathematics of Operations Research*, 30(2):257–280, 2005.

[55] J B Lasserre. Inverse polynomial optimization. *Mathematics of Operations Research*, 38(3):418–436, 2013.

[56] T Lipp and S Boyd. Variations and extension of the convex–concave procedure. *Optimization and Engineering*, 17(2):263–287, 2016.

[57] M S Lobo, L Vandenberghe, S Boyd, and H Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284(1-3):193–228, 1998.

[58] A Mahajan and D Teneketzis. Multi-armed bandit problems. In A P Hero, D A Castañón, D A Cochran, and K Kastella, editors, *Foundations and Applications of Sensor Management*, chapter 6, pages 121–151. Springer, Boston, MA, USA, 2008.

[59] S Mannor, D Simester, P Sun, and J N Tsitsiklis. Bias and variance approximation in value function estimates. *Management Science*, 53(2):308–322, 2007.

[60] M Naghavi, A A Faroughi, and M Zarepisheh. Inverse optimization for multi-objective linear programming. *Optimization Letters*, 13(2):281–294, 2019.

[61] Y Nesterov and A Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Studies in Applied and Numerical Mathematics. SIAM, 1994.

[62] G Neumann-Denzau and J Behrens. Inversion of seismic data using tomographical reconstruction techniques for investigations of laterally inhomogeneous media. *Gephysical Journal of Royal Astronomical Society*, 79(1):305–315, 1984.

[63] A Nilim and L El Ghaoui. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

[64] G Nolet. *Seismic Tomography*. Reidel, Dordrecht, 1987.

[65] S Nourollahi and A Ghate. Inverse optimization in minimum cost flow problems on countably infinite networks. *Networks*, 73(3):292–305, 2019.

[66] M Salemi Parizi and A Ghate. Multi-class, multi-resource advance scheduling with no-shows, cancellations and overbooking. *Computers & Operations Research*, 67:90–101, 2016.

[67] M Salemi Parizi, Y Gocgun, and A Ghate. Approximate policy iteration for dynamic resource-constrained project scheduling. *Operations Research Letters*, 45:442–447, 2017.

[68] J Patrick, M L Puterman, and M Queyranne. Dynamic multi-priority patient scheduling for a diagnostic resource. *Operations Research*, 56(6):1507–1525, 2008.

[69] A Prekopa. *Stochastic Programming*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1995.

[70] M L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

[71] J Roland, Y D Smet, and J R Figueira. Inverse multi-objective combinatorial optimization. *Discrete Applied Mathematics*, 161(16-17):2764–2771, 2013.

[72] A J Schaefer. Inverse integer programming. *Optimization Letters*, 3(4):483–489, 2009.

[73] B K Sriperumbudur and G RG Lanckriet. On the convergence of the concave-convex procedure. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, pages 1759–1767, 2009.

[74] Y Tan, A Delong, and D Terekhov. Deep inverse optimization. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 540–556. Springer, 2019.

[75] A Tarantola. *Inverse problem theory: methods for data fitting and model parameter estimation*. Elsevier, Amsterdam, The Netherlands, 1987.

[76] O Tavaslioglu, T Lee, S Valeva, and A J Schaefer. On the structure of the inverse-feasible region of a linear program. *Operations Research Letters*, 46(1):147–152, 2018.

[77] M D Troutt, W Pang, and S Hou. Behavioral estimation of mathematical programming objective function coefficients. *Management Science*, 52(3):422–434, 2006.

[78] J N Tsitsiklis. A short proof of the Gittins index theorem. *Annals of Applied Probability*, 4(1):194–199, 1994.

[79] L Vandenberghe and S Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996.

[80] L Vandenberghe and S Boyd. Applications of semidefinite programming. *Applied Numerical Mathematics*, 29(3):283–299, 1999.

[81] L Wang. Cutting plane algorithms for the inverse mixed integer linear programming problem. *Operations Research Letters*, 37(2):114–116, 2009.

[82] R Weber. On the Gittins index for multiarmed bandits. *Annals of Applied Probability*, 2(4):1024–1033, 1992.

[83] R E Wendell and A P Hurter Jr. Minimization of a non-separable objective function subject to disjoint constraints. *Operations Research*, 24(4):643–657, 1976.

[84] P Whittle. Multi-armed bandits and the Gittins index. *Journal of the Royal Statistical Society: Series B (Methodological)*, 42(2):143–149, 1980.

[85] P Whittle. Restless bandits: Activity allocation in a changing world. *Journal of Applied Probability*, 25(A):287–298, 1988.

[86] J H Woodhouse and A M Dziewonski. Mapping the upper mantle: three dimensional modeling of earth structure by inversion of seismic waveforms. *Journal of Geophysical Research*, 89(B7):5953–5986, 1989.

[87] S Xu and J Zhang. An inverse problem of the weighted shortest path problem. *Japan Journal of Industrial and Applied Mathematics*, 12:47–59, 1995.

[88] C Yang, J Zhang, and Z Ma. Inverse maximum flow and minimum cut problem. *Optimization*, 40(2):147–170, 1997.

[89] J Zhang and C Xu. Inverse optimization for linearly constrained convex separable programming problems. *European Journal of Operational Research*, 200(3):671–679, 2010.