Efficient Algorithms for Budget-Constrained Markov Decision Processes

Constantine Caramanis[1], Nedialko B. Dimitrov[2], and David P. Morton[3]

[1]Department of Electrical and Computer Engineering, The University of Texas at Austin
[2]Operations Research Department, Naval Postgraduate School
[3]Graduate Program in Operations Research, The University of Texas at Austin

***Abstract*—Discounted, discrete-time, discrete state-space, discrete action-space Markov decision processes (MDPs) form a classical topic in control, game theory, and learning, and as a result are widely applied, increasingly, in very large-scale applications. Many algorithms have been developed to solve large-scale MDPs. Algorithms based on value iteration are particularly popular, as they are more efficient than the generic linear programming approach, by an order of magnitude in the number of states of the MDP. Yet in the case of budget constrained MDPs, no more efficient algorithm than linear programming is known. The theoretically slower running times of linear programming may limit the scalability of constrained MDPs piratically; while, theoretically, it invites the question of whether the increase is somehow intrinsic. In this paper we show that it is not, and provide two algorithms for budget-constrained MDPs that are as efficient as value iteration. Denoting the running time of value iteration by VI, and the magnitude of the input by U, for an MDP with m expected budget constraints our first algorithm runs in time $O(\mathrm{poly}(m, \log U) \cdot \mathrm{VI})$. Given a pre-specified degree of precision, η, for satisfying the budget constraints, our second algorithm runs in time $O(\log m \cdot \mathrm{poly}(\log U) \cdot \frac{1}{\eta^2} \cdot \mathrm{VI})$, but may produce solutions that overutilize each of the m budgets by a multiplicative factor of $1 + \eta$. In fact, one can substitute value iteration with any algorithm, possibly specially designed for a specific MDP, that solves the MDP quickly to achieve similar theoretical guarantees. Both algorithms restrict attention to constrained infinite-horizon MDPs under discounted costs.**

I. Introduction

In a standard Markov decision process (MDP), the goal is to find a policy mapping states to actions that maximizes the expected reward over the lifetime of the process. In a constrained MDP, the goal is the same except that we associate a cost with each action and we can optimize only over policies whose expected total cost is within a prespecified limit. Constrained MDPs have seen application in a range of areas (see [1], [5], [28]). Applications of MDPs with expected budget constraints, such as the ones considered here, include computing policies for hospital admissions scheduling [22], maintenance [17], [30], and more recently wireless carrier communication [29], [34].

The primary method for solving constrained MDPs is based on linear programming (see, e.g., the classical reference [13], and the more recent work [1], [33]). To solve a constrained MDP with n states, k actions per state, and m expected-value budget constraints using current linear programming interior point methods requires a running time that exceeds $\Omega((n+m)^2 \cdot n \cdot k)$, ignoring factors dependent on the size of the input, U (see Ye [32, Section 5.1] and Anstreicher [2]), where U denotes the size of the input required to specify the constrained MDP. In contrast, the running time of algorithms for unconstrained MDPs is a factor of n more efficient; for example, value iteration can solve discounted MDPs in time $O(n^2 \cdot k)$, ignoring factors of U. This significant difference in efficiency limits the scalability of constrained MDPs to applications with a modest number of states [1], [24], [28].

This paper describes two algorithms for constrained MDPs, which achieve running times comparable to value iteration for unconstrained MDPs. Our first algorithm is based on column generation, using the ellipsoid method and value iteration as a separation oracle, and achieves a running time of $O(\mathrm{poly}(m) \cdot n^2 \cdot k)$, ignoring factors of U (Theorem III.1). In fact, one can substitute a more practical column generation technique for the ellipsoid method, and any special-purpose algorithm for value iteration to provide a pathway for practical running times. By a special-purpose algorithm, we mean any algorithm, possibly specially designed for a specific MDP, that solves the MDP quickly. Value iteration is one such algorithm, but others exist.

Our second algorithm is based on the generalized experts framework and, given a pre-specified degree of precision η for satisfying the budget constraints, achieves running time $O(\log m \cdot \frac{1}{\eta^2} \cdot n^2 \cdot k)$, ignoring factors of U (Theorem IV.2), reducing the dependence on m to logarithmic but allowing each of the m budget constraints to overuse the budget by a multiplicative factor of $1 + \eta$. Similarly as for our first algorithm, any special-purpose unconstrained MDP algorithm can substitute for value iteration.

Bertsimas and Orlin [6] use the ellipsoid method to derive efficient algorithms for problems with side constraints, including the traveling salesman problem, a vehicle routing problem, and a Steiner tree problem. The idea in this algorithmic approach is to: 1) reformulate the optimization problem in a column-generation manner, with an exponential number of columns; 2) use the ellipsoid method to generate columns; and, 3) use a convex combination of the generated columns to solve the original problem. A key step in this approach is to exploit the ellipsoid method to solve an exponentially-sized model in polynomial running time [19].

Our second algorithm uses the idea of exploiting a single-side-constraint version of a problem to build an algorithm for the problem with multiple side constraints. Plotkin, Shmoys, and Tardos introduce this approach in the context of fractional packing problems [27]; see also, [18]. It has since been used in boosting and in the generalized experts framework in machine learning [16], [23], [3]. And, this approach has been used to create an efficient algorithm for low-rank matrix approximation with guarantees [25]. To the best of our knowledge, our paper is the first to exploit these ideas to improve the running time for any MDP algorithms.

The remainder of this paper is structured as follows. In Section II we define basic notation. In Section III we present our efficient, exact algorithm, based on the ellipsoid method. In Section IV we present our efficient, approximately feasible algorithm based on the generalized experts framework. Section V summarizes.

II. Basics of MDPs

We assume the reader is familiar with the value iteration (e.g., [28, p.160]) and linear programming algorithms for solving MDPs (e.g., [9], [28]). We also assume familiarity with expected discounted-cost constrained MDPs [1]. Hence, we only briefly review the basics of MDPs.

Consider a discounted, discrete-time, discrete-state, discrete action-space MDP. Let $\mathcal{S}$ denote the finite set of states and A_s denote the finite set of actions for state s. We use n to denote the number of states, and, for simplicity, we assume that each state has the same number of actions available, k. We use $r_{s,a}$ to denote the reward received by performing action a in state s; $\bar{p}(s' \mid s, a)$ to denote the transition probability to state s' given that the MDP is in state s and action a is performed; and, w_s to denote the initial probability for the MDP to be in state s.

We focus on infinite-horizon MDPs with discount factor $\alpha \in (0, 1)$. We simplify notation by absorbing α into the transition probabilities. Dropping the "bar" notation, the transition probabilities with the discount factor absorbed are $p(s' \mid s, a)$, allowing us to express models without α appearing explicitly. That said, we emphasize that we assume the discount factor α is fixed. Because we focus on discounted MDPs, when we use *expected reward* or *expected cost*, we refer to a time-discounted expected reward or cost. We let U be larger than the largest magnitude number in the input, and larger than $\frac{1}{1-\alpha}$. We also assume that $\text{poly}(\log n, \log k) = O(nk) = O(\text{VI})$, in other words that the special-purpose algorithm requires at least nk operations.

Value iteration requires $O(U \log U \cdot n^2 \cdot k)$ time to solve an MDP, and linear programming requires $\Omega(\log U \cdot n^3 \cdot k)$. Other algorithms such as policy iteration or a hybrid algorithm are also frequently used [4], [20], [28]. However, it is arguably this result on running times that makes value iteration the basis of most practical algorithms for large-scale MDPs with many states.

To specify an expected budget constraint, we let $c_{s,a,i}$ be the cost of performing action a in state s, and b_i be an upper bound on the expected cost of the policy, where $i = 1, \ldots, m$ indexes a total of m such constraints. Expected budget constraints connect decisions across states and time, breaking the classic Bellman recursion. By expanding the domain of the value function to incorporate constraint information, it is possible to develop dynamic programming equations, and employ value iteration, for constrained MDPs [10], [26]. That said, handling expected budget constraints algorithmically is easily done using linear programming (again, see, e.g., [1]) and this is the path we follow.

III. An Ellipsoid Algorithm Approach

This section presents an efficient algorithm for an MDP with multiple expected budget constraints. The algorithm finds both the optimal value and an optimal policy via a column generation approach using the ellipsoid algorithm [6]. The key idea is to reformulate the problem with data defined by the extreme points of the polytope of the underlying linear programming formulation of the MDP. This results in a reformulation with exponentially many decision variables, an idea that began with Dantzig and Wolfe [11], [12]. Then, using the ellipsoid method as a form of column generation, we can find both the optimal value and an optimal policy for an MDP with m expected budget constraints in time $O(\text{poly}(m, \log U) \cdot \text{VI})$ (see Theorem III.1).

We begin with the linear program for an MDP with m expected budget constraints

$$\begin{aligned} z^* = \max_{\mathbf{x}} \quad & \mathbf{r}^\intercal \cdot \mathbf{x} \\ \text{s.t.} \quad & \mathbf{x} \in \mathbb{X} \\ & \mathbf{C} \cdot \mathbf{x} \leq \mathbf{b}. \end{aligned} \tag{1}$$

Here, $\mathbf{r} \in \mathbb{R}^{nk}$ is the vector of rewards, $r_{s,a}$; $\mathbf{C} \in \mathbb{R}^{m \times nk}$ is the matrix of costs, $c_{s,a,i}$, for the m budget constraints; $\mathbf{b} \in \mathbb{R}^m$ has components b_i; and,

$$\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^{nk} : \mathbf{x} \geq 0,$$
$$\sum_{a \in A_s} x_{s,a} - \sum_{s' \in \mathcal{S}} \sum_{a \in A_{s'}} p(s \mid s', a) x_{s',a} = w_s, s \in \mathcal{S}\},$$

using the same notation as the previous section. The decision variables $\mathbf{x}$, with individual coordinates $x_{s,a}$, represent the expected number of times action a is performed in state s. These decision variables produce a randomized policy for the MDP [9], [28]. Throughout we assume model (1) is feasible.

We reformulate model (1) by writing $\mathbf{x}$ as a convex combination of the extreme points of $\mathbb{X}$, which we denote $\mathrm{ext}(\mathbb{X})$. We define a new variable, $\theta_{\hat{\mathbf{x}}}$, for each $\hat{\mathbf{x}} \in \mathrm{ext}(\mathbb{X})$, and let $\boldsymbol{\theta}$ be the vector of all such variables. We reformulate model (1) with an exponential number of variables, but only $m+1$ constraints:

$$\begin{aligned} \max_{\boldsymbol{\theta}} \quad & \sum_{\hat{\mathbf{x}} \in \mathrm{ext}(\mathbb{X})} (\mathbf{r}^\intercal \cdot \hat{\mathbf{x}}) \theta_{\hat{\mathbf{x}}} \\ \text{s.t.} \quad & \sum_{\hat{\mathbf{x}} \in \mathrm{ext}(\mathbb{X})} \theta_{\hat{\mathbf{x}}} = 1 \\ & \sum_{\hat{\mathbf{x}} \in \mathrm{ext}(\mathbb{X})} (\mathbf{C} \cdot \hat{\mathbf{x}}) \theta_{\hat{\mathbf{x}}} \leq \mathbf{b} \\ & \boldsymbol{\theta} \geq 0, \end{aligned} \tag{2}$$

where the first and third constraints force $\boldsymbol{\theta}$ to specify a convex combination, and we have substituted $\mathbf{x}$ with that convex combination in the objective function and the budget constraints.

Consider the dual of model (2), with $m+1$ variables and exponentially many constraints. Using $\boldsymbol{\lambda} \in \mathbb{R}^m$ as the dual variables corresponding to the m budget constraints, and π as the dual variable corresponding to the simplex constraint, we have

$$\begin{aligned} \min_{\boldsymbol{\lambda}, \pi} \quad & \mathbf{b}^\intercal \cdot \boldsymbol{\lambda} + \pi \\ \text{s.t.} \quad & \boldsymbol{\lambda}^\intercal \cdot (\mathbf{C} \cdot \hat{\mathbf{x}}) + \pi \geq \mathbf{r}^\intercal \cdot \hat{\mathbf{x}}, \quad \forall \hat{\mathbf{x}} \in \mathrm{ext}(\mathbb{X}) \\ & \boldsymbol{\lambda} \geq 0. \end{aligned} \tag{3}$$

We can solve model (3) using the ellipsoid algorithm, if we can find an efficient separation oracle for its many constraints. Ignoring the nonnegativity constraints, because they can be easily checked, we search for the most violated inequality given specific values of the variables $\bar{\boldsymbol{\lambda}}$ and $\bar{\pi}$. Finding the most violated inequality amounts to solving the optimization problem:

$$\begin{aligned} \max_{\mathbf{x}} \quad & \mathbf{r}^\intercal \cdot \mathbf{x} - \bar{\boldsymbol{\lambda}}^\intercal \cdot (\mathbf{C} \cdot \mathbf{x}) - \bar{\pi} \\ \text{s.t.} \quad & \mathbf{x} \in \mathbb{X}. \end{aligned} \tag{4}$$

If the optimal value of model (4) is nonpositive then all constraints in model (3) are satisfied. Otherwise, model (4)'s solution yields a violated inequality. A sequence of values for $(\bar{\boldsymbol{\lambda}}, \bar{\pi})$ can be specified by the ellipsoid algorithm [19] so as to require at most $\mathrm{poly}(m, \log U)$ calls to the separation oracle. More precisely, the running time of solving the program is bounded by $O(\mathrm{poly}(m, \log U))$, where U is larger than the magnitude in the largest entry in $\mathbf{C}, \mathbf{b}$, and $\mathbf{r}$, and is larger than $\frac{1}{1-\alpha}$ [7, pp.373-383]. Practically, the sequence of $(\bar{\boldsymbol{\lambda}}, \bar{\pi})$ could also be specified using any row-generation algorithm for model (3).

The key in this approach is that the separation oracle, model (4), is a standard unconstrained MDP that we can solve in time VI using value iteration. In model (4), we write the optimization over $\mathbb{X}$ instead of $\mathrm{ext}(\mathbb{X})$. This is possible because under its linear objective function there exists an optimal solution, which is an extreme point of $\mathbb{X}$. Value iteration produces such an extreme point because it finds a policy from the set of nonrandomized policies, which have a one-to-one correspondence with the extreme points of $\mathbb{X}$; see, e.g., Wagner [31]. Other special-purpose algorithms can substitute for value iteration when solving model (4). If they do not produce an element of $\mathrm{ext}(\mathbb{X})$, the argument in this section can be repeated by substituting $\mathbb{X}$ for $\mathrm{ext}(\mathbb{X})$ and repeating the arguments about Models (2) and (3), which would have infinitely many variables and constraints, respectively.

Because model (3) is a convex optimization problem with $m+1$ variables, we need at most $O(\mathrm{poly}(m, \log U))$ calls to the separation oracle and a total time of $O(\mathrm{poly}(m, \log U) \cdot \mathrm{VI})$. Finding a solution to model (3) gives us the optimal value of model (1), but we still need to reconstruct an optimal solution.

To reconstruct an optimal solution to model (1), we index the constraints generated by the separation oracle by $\mathbb{G} \subset \mathrm{ext}(\mathbb{X})$ and obtain:

$$\begin{aligned} \min_{\boldsymbol{\lambda}, \pi} \quad & \mathbf{b}^\intercal \cdot \boldsymbol{\lambda} + \pi \\ \text{s.t.} \quad & \boldsymbol{\lambda}^\intercal \cdot (\mathbf{C} \cdot \hat{\mathbf{x}}) + \pi \geq \mathbf{r}^\intercal \cdot \hat{\mathbf{x}}, \quad \forall \hat{\mathbf{x}} \in \mathbb{G} \\ & \boldsymbol{\lambda} \geq 0, \end{aligned} \tag{5}$$

a linear program with $m+1$ variables and only $O(\mathrm{poly}(m, \log U))$ constraints; see, e.g., [6] and [19, pp.183-185]. The dual of model (5) yields a restricted version of model (2), with only $|\mathbb{G}| = O(\mathrm{poly}(m, \log U))$ variables and $m+1$ constraints, given by

$$\begin{aligned} \max_{\boldsymbol{\theta}} \quad & \sum_{\hat{\mathbf{x}} \in \mathbb{G}} (\mathbf{r}^\intercal \cdot \hat{\mathbf{x}}) \theta_{\hat{\mathbf{x}}} \\ \text{s.t.} \quad & \sum_{\hat{\mathbf{x}} \in \mathbb{G}} \theta_{\hat{\mathbf{x}}} = 1 \\ & \sum_{\hat{\mathbf{x}} \in \mathbb{G}} (\mathbf{C} \cdot \hat{\mathbf{x}}) \theta_{\hat{\mathbf{x}}} \leq \mathbf{b} \\ & \boldsymbol{\theta} \geq 0. \end{aligned} \tag{6}$$

Any solution to model (6) is feasible to model (2) because $\mathbb{G}$ is a subset of $\text{ext}(\mathbb{X})$. The optimal values of the primal-dual pair (5)-(6) are, of course, equal. As we indicate above, this value is in turn equal to z^*, the optimal value of the primal-dual pair (2)-(3). Thus, the solution of model (6) is not only feasible, but also optimal to model (2). The only remaining step is to solve model (6), which has $O(\text{poly}(m, \log U))$ variables and $m+1$ constraints. The run time of solving model (6) depends on the magnitude of the numbers in the problem. Each coordinate of the generated $\mathbf{x}$ is less than $\frac{1}{1-\alpha} \leq U$. The magnitude of the numbers in model (6) are therefore bounded by nkU^2, giving the run time of $O(\text{poly}(m, \log U, \log(nk)))$ to solve the problem once it is constructed [8], [7], [21]. In addition, it takes time $O(nk)$ to compute each of the coefficients in the problem. So, the overall run time of solving model (6) is $O(\text{poly}(m, \log U)nk)$

The procedure outlined above constructs randomized solutions to the constrained MDP. In particular, each of the generated solutions in $\mathbb{G}$ are deterministic because they come from a special-purpose algorithm like value iteration. However, solving model (6) constructs a convex combination of those deterministic solutions, giving the randomized policy for the constrained MDP.

Combining both the ellipsoid method step and the final reconstruction step, we are able to find both the optimal value and an optimal solution to model (1) in time $O(\text{poly}(m, \log U) \cdot \text{VI})$. We summarize this discussion in the form of an algorithm and an accompanying theorem:

Algorithm 1: Exact algorithm for model (1), an MDP with m expected budget constraints.

1) Using a special-purpose algorithm for an unconstrained MDP as a separation oracle, solve model (3) using the ellipsoid method, or any row generation scheme.
2) Let $\mathbb{G}$ index the set of generated constraints.
3) Form and solve model (6) to obtain the optimal value and optimal solution to model (1).

Theorem III.1. *Assume model (1) is feasible, and let* VI *be the running time of a special-purpose algorithm for solving an unconstrained MDP. Algorithm 1, when employing the ellipsoid method in step 1, solves an MDP with* m *expected budget constraints, i.e., model (1), obtaining both an objective function value and an optimal solution in time* $O(\text{poly}(m, \log U) \cdot \text{VI})$.

IV. An Online Learning Approach

This section creates another efficient algorithm for an MDP with multiple expected budget constraints. Instead of using the ellipsoid algorithm, we use the randomized weighted majority algorithm to prove that if we allow a slight violation of the m budget constraints, then it is possible to reduce the running time dependence on m, from a polynomial factor to a logarithmic factor. Intuitively, a super-optimal solution is not feasible but achieves an objective function value that is better than the optimal. We find a policy that is super-optimal and violates each expected budget constraint by at most a multiplicative factor of $1 + \eta$ in time $O(\log m \cdot \text{poly}(\log U) \cdot \frac{1}{\eta^2} \cdot \text{VI})$ (see Theorem IV.2).

Generalized Experts Framework. Consider a process in which we face an adversary and are tasked with combining the advice of m experts. The process proceeds in rounds. In each round, we choose an action advocated by one expert to face an event chosen by the adversary. If the adversary chooses event $\hat{\mathbf{x}}$, then the action advocated by expert i incurs penalty $M(i, \hat{\mathbf{x}})$. Thus, if we select an action using a distribution $\mathbf{p} = (p_1, \ldots, p_m)$ over the experts, we incur an expected penalty of $\sum_{i=1}^m p_i M(i, \hat{\mathbf{x}})$ on event $\hat{\mathbf{x}}$. In each round, t, we must choose a distribution, $\mathbf{p}^t$, *before* seeing the event $\hat{\mathbf{x}}^t$ chosen by the adversary. The randomized weighted majority algorithm follows with an accompanying theorem:

Randomized Weighted Majority Algorithm: Regret minimization in generalized experts framework.

1) To define the probability distribution over experts, we keep a vector of weights. We initialize the weight vector with equal weight on each expert ($w_i^0 = 1, \forall i$).
2) In round t:
   a) Select an action using $p_i^t = \frac{w_i^t}{\sum_{k=1}^m w_k^t}$.
   b) When the adversary's event, $\hat{\mathbf{x}}^t$, is revealed, update the weights as follows
   $$w_i^{t+1} = \begin{cases} w_i^t(1-\delta)^{M(i,\hat{\mathbf{x}}^t)/\rho} & \text{if } M(i, \hat{\mathbf{x}}^t) \geq 0 \\ w_i^t(1+\delta)^{-M(i,\hat{\mathbf{x}}^t)/\rho} & \text{if } M(i, \hat{\mathbf{x}}^t) < 0 \end{cases},$$
   where ρ and δ are parameters defined in Theorem IV.1.

Theorem IV.1 (Arora et al. [3]). *Fix* $\varepsilon > 0$ *to be an error parameter, and let* $\rho > 0$ *be a uniform bound on the penalty incurred by any expert under any event; i.e.,* $M(i, \mathbf{x}) \in [-\rho, \rho]$, $\forall i$ *and* $\forall \mathbf{x}$. *Fixing the parameter* δ *in the randomized weighted majority algorithm to* $\min\{\frac{\varepsilon}{4\rho}, \frac{1}{2}\}$, *for any sequence of events of length* $T \geq \frac{16\rho^2 \ln m}{\varepsilon^2}$, *the average loss incurred by the algorithm is within* ε *of the average loss incurred by the best expert:* $\frac{\sum_{t=1}^T \sum_{i=1}^m p_i^t M(i, \hat{\mathbf{x}}^t)}{T} \leq \varepsilon + \min_i \frac{\sum_{t=1}^T M(i, \hat{\mathbf{x}}^t)}{T}$.

Application to a Constrained MDP. The key observation is that the theorem guarantees the bound on loss for *any* sequence of events. We apply Theorem IV.1 to our constrained MDP by appropriately defining experts, events, penalties, and the adversary's event sequence.

Each of the m budget constraints corresponds to an expert. The "adversary" chooses events $\hat{\mathbf{x}}$ from the space of potential solutions to the unconstrained MDP, which are possibly infeasible for the constrained MDP in model (1). The penalty for expert i under event $\hat{\mathbf{x}}$, is the magnitude by which constraint i is *satisfied* by $\hat{\mathbf{x}}$: $M(i, \hat{\mathbf{x}}) = b_i - \mathbf{c}_i \cdot \hat{\mathbf{x}}$, where $\mathbf{c}_i$ is the i-th row of matrix $\mathbf{C}$ in model (1).

Therefore, the best expert in this definition corresponds to the *most unsatisfied* constraint. In the context of the randomized weighted majority algorithm, our definition of the penalties increases the weights associated with unsatisfied budget constraints. Now the key to applying Theorem IV.1 is the fact that the regret bounds hold *for any sequence of the adversary's choices*. In particular, we can define this sequence ourselves. We do so as follows. At each round, t, we use the decision maker's distribution to create a convex combination of the m constraints,

$$\left(\sum_{i=1}^{m} p_i^t \mathbf{c}_i\right) \cdot \mathbf{x} \leq \sum_{i=1}^{m} p_i^t b_i. \tag{7}$$

Let $\hat{\mathbf{x}}^t$ be the solution of the constrained MDP, under this single aggregated constraint (7). That is, we obtain $\hat{\mathbf{x}}^t$ by solving MDP model (1) except that its m budget constraints are replaced by (7) and we do so using an algorithm for a single-constraint [15], [14]. We treat this as the adversary's action. The basic result of this section is that running the randomized weighted majority algorithm under these definitions produces a solution $\bar{\mathbf{x}} = \frac{1}{T}\sum_{t=1}^{T} \hat{\mathbf{x}}^t$ that is both super-optimal and approximately feasible to model (1).

First, feasibility of model (1) implies feasibility of its relaxation under the aggregated budget constraint (7). Second, if each $\hat{\mathbf{x}}^t$ is super-optimal then so is the convex combination $\frac{1}{T}\sum_{t=1}^{T} \hat{\mathbf{x}}^t$. The super-optimality condition, i.e., $\mathbf{r} \cdot \mathbf{x} \geq z^*$, holds for each $\hat{\mathbf{x}}^t$ because we obtain it by solving a relaxation of the m-constraint model (1) under constraint (7).

Third, we show approximate feasibility of $\bar{\mathbf{x}}$ by applying Theorem IV.1. Intuitively, the randomized weighted majority algorithm increases weights on unsatisfied constraints and decreases weights on satisfied constraints. More specifically, Theorem IV.1 states that $\frac{1}{T}(\sum_{t=1}^{T}\sum_{i=1}^{m} p_i^t(b_i - \mathbf{c}_i \cdot \hat{\mathbf{x}}^t) - \sum_{t=1}^{T}(b_k - \mathbf{c}_k \cdot \hat{\mathbf{x}}^t)) \leq \varepsilon,\ \forall k = 1, \ldots, m$. We fix values of ε, δ, ρ, and T shortly, and continue proving approximate feasibility of $\bar{\mathbf{x}}$. Rearranging terms, we obtain: $\mathbf{c}_k \cdot \bar{\mathbf{x}} \leq b_k + \varepsilon - \frac{1}{T}\left(\sum_{t=1}^{T}\sum_{i=1}^{m} p_i^t(b_i - \mathbf{c}_i \cdot \hat{\mathbf{x}}^t)\right)$, $\forall k$, implying by constraint (7) the intermediate approximate feasibility result: $\mathbf{c}_k \cdot \bar{\mathbf{x}} \leq b_k + \varepsilon,\ \forall k$. Intuitively, ε is set to a value smaller than ηb_k for all k, giving the final, multiplicative approximate feasibility result. Concluding this step of the argument, the solution $\bar{\mathbf{x}}$ has an objective function value that is super-optimal with respect to the original problem, and $\bar{\mathbf{x}}$ violates each budget constraint by no more than ε.

Finally, we analyze the time we require to compute $\bar{\mathbf{x}}$. To construct the convex combination of constraints in each round, requires $O(m \cdot n \cdot k)$ time, where $n \cdot k$ is the length of a cost vector for a single budget constraint. Finding a solution for the resulting single constraint MDP using Lagrangian relaxation and a search for the multiplier (see [15] and references therein) requires $O(\mathrm{VI} \cdot \mathrm{poly}(\log U))$ time.

It only remains to analyze the total number of rounds required. Theorem IV.1 states that $T \geq \frac{16\rho^2 \ln m}{\varepsilon^2}$ rounds are required. The theorem requires ρ to bound the maximum penalty achievable in any round, which is the maximum amount by which the MDP can overuse any budget constraint. If the maximum cost on any action is $c_{\max}$, then the maximum overuse of any expected budget constraint is $c_{\max} \cdot \frac{1}{1-\alpha}$. Let $B = c_{\max} \cdot \frac{1}{1-\alpha}$, and set $\rho = B$. Setting $\varepsilon = \hat{\eta} B$ gives $T = \frac{16 \ln m}{\hat{\eta}^2}$ rounds, and an approximate feasibility result of $c_k \cdot \bar{\mathbf{x}} \leq (1 + \frac{\hat{\eta} B}{b_k}) b_k,\ \forall k$. We can now set $\hat{\eta} = \eta \frac{b_{\min}}{B}$, where $b_{\min} = \min_i b_i$, to obtain $T = \frac{16 B^2 \ln m}{b_{\min}^2 \eta^2} = O(\frac{\log m}{\eta^2})$ rounds, and an approximate feasibility result of $c_k \cdot \bar{\mathbf{x}} \leq (1 + \eta) b_k,\ \forall k$. We summarize the above discussion, into an algorithm and accompanying theorem as follows:

Algorithm 2: Given a parameter η, an algorithm for finding an approximately feasible, super-optimal solution to model (1), an MDP with m expected budget constraints.

1) Given $\eta > 0$, set $\hat{\eta} = \eta \frac{b_{\min}}{B}$, $\rho = B$, $\varepsilon = \hat{\eta} B$, $T \geq \frac{16\rho^2 \ln m}{\varepsilon^2}$, and $\delta = \min\{\frac{\varepsilon}{4\rho}, \frac{1}{2}\}$.
2) Run the randomized weighted majority algorithm for T rounds, using the definitions of experts, penalties, and events described in this section, producing the sequence of events $\hat{\mathbf{x}}^1, \ldots, \hat{\mathbf{x}}^T$ over the T rounds.
3) Form the vector $\bar{\mathbf{x}} = \frac{1}{T}\sum_{t=1}^{T} \hat{\mathbf{x}}^t$, which is both approximately feasible and super-optimal to model (1).

Theorem IV.2. *Assume model (1) is feasible, and let* VI *be the running time of a special-purpose algorithm for solving an unconstrained MDP. Algorithm 2 produces a super-optimal solution,* $\bar{\mathbf{x}}$*, to model (1) that violates each expected budget constraint by at most a multiplicative factor of* $1 + \eta$ *in the budget, for* $\eta > 0$*, and runs in time* $O(\log m \cdot \mathrm{poly}(\log U) \cdot \frac{1}{\eta^2} \cdot \mathrm{VI})$.

V. Summary

We show two algorithms for discounted constrained MDPs. The first algorithm uses a column-generation and the ellipsoid method, while the second uses a generalized experts framework. The algorithms reduce the complexity required to solve a constrained MDP by an order of magnitude in the number of states, over the standard linear programming approach. The price we pay in the first algorithm is a polynomial factor in the number of constraints, m. In the second algorithm we allow overuse of the budget constraints by a multiplicative factor of $1+\eta$, and we pay a $\log m \cdot \frac{1}{\eta^2}$ factor in running time.

The main contribution of this work is intended to be analytical. A practical implementation of the proposed schemes, and then a detailed study through extensive computational examples is an important step that we have not pursued here. Such a demonstration requires a suite of realistic constrained MDP problems of various sizes to test the algorithm performance. In addition, the algorithm based on a generalized experts framework does not produce solutions that satisfy the budget constraints exactly. Whether solutions that approximately satisfy the constraints are acceptable depends on the specific application. This work informs practitioners by providing a sketch of two approaches for efficient algorithms in large-scale constrained MDPs.

Acknowledgements. This work has been supported by the NSF through grants CMMI-0653916 and CMMI-0800676, the DTRA through grant HDTRA1-08-1-0029, and the US DHS grant number 2008-DN-077-ARI021-05. We thank Csaba Szepesvári and three referees whose comments improved the paper.

References

[1] E. Altman. *Constrained Markov Decision Processes*. Chapman and Hall/CRC, Boca Raton, Florida, 1999.

[2] K. M. Anstreicher. Linear programming in $O(n^3 L/\ln n)$ operations. *SIAM Journal on Optimization*, 9(4):803–812, 1999.

[3] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8:121–164, 2012.

[4] D. P. Bertsekas. A new value iteration method for the average cost dynamic programming problem. *SIAM Journal on Control and Optimization*, 36(2):742–759, 1998.

[5] D.P. Bertsekas. *Dynamic Programming and Optimal Control, Volume 1 and 2*. Athena Scientific, Belmont, Massachusetts, 1995.

[6] D. Bertsimas and J. B. Orlin. A technique for speeding up the solution of the Lagrangean dual. *Mathematical Programming*, 63(1):23–45, 1994.

[7] D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, Belmont, Massachusetts, 1997.

[8] R. G. Bland, D. Goldfarb, and M. J. Todd. The ellipsoid method: A survey. *Operations Research*, 29(6):1039–1091, 1981.

[9] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, 2004.

[10] R. C. Chen and E. A. Feinberg. Non-randomized policies for constrained Markov decision processes. *Mathematical Methods of Operations Research*, 66(1):165–179, 2007.

[11] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

[12] G. B. Dantzig and P. Wolfe. The decomposition algorithm for linear programs. *Econometrica*, 29:767–778, 1961.

[13] C. Derman and A. F. Veinott. Constrained Markov decision chains. *Management Science*, 19(4):389–390, 1972.

[14] D. D. DeWolfe, J. G. Stevens, and R. K. Wood. Setting military reenlistment bonuses. *Naval Research Logistics*, 40:143–160, 1993.

[15] B. L. Fox and D. M. Landi. Searching for the multiplier in one-constraint optimization problems. *Operations Research*, 18:253–262, 1970.

[16] Y. Freund and R. E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

[17] K. Golabi, R. B. Kulkarni, and G. B. Way. A statewide pavement management system. *Interfaces*, 12(6):5–21, 1982.

[18] M. D. Grigoriadis and L. G. Khachiyan. Fast approximation schemes for convex programs with many blocks and coupling constraints. *SIAM Journal on Optimization*, 4(1):86–107, 1994.

[19] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, New York, 1993.

[20] R. A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, Massachusetts, 1960.

[21] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, STOC '84, pages 302–311, New York, NY, USA, 1984. ACM.

[22] P. Kolesar. A Markovian model for hospital admission scheduling. *Management Science*, 16(6):B384–B396, 1970.

[23] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 256–261, 1989.

[24] M. L. Littman, T. L. Dean, and L. P. Kaelbling. On the complexity of solving Markov decision problems. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI–95)*, pages 394–402, Montreal, Québec, Canada, 1995.

[25] R. Meka, P. Jain, C. Caramanis, and I. S. Dhillon. Rank minimization via online learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 656–663, 2008.

[26] A. B. Piunovskiy and X. Mao. Constrained Markovian decision processes: the dynamic programming approach. *Operations Research Letters*, 27:119–126, 2000.

[27] S. A. Plotkin, D. B. Shmoys, and E. Tardos. Fast approximation algorithms for fractional packing and covering problems. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, pages 495–504, 1991.

[28] M. L. Puterman. *Markov Decision Processes: Discrete Dynamic Programming*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2005.

[29] C. Sun, E. Stevens-Navarro, and V. W. S. Wong. A constrained MDP-based vertical handoff decision algorithm for 4G wireless networks. In *IEEE International Conference on Communications. ICC '08*, pages 2169–2174. IEEE, 2008.

[30] C. van Winden and R. Dekker. Rationalisation of building maintenance by Markov decision models: A pilot case study. *The Journal of the Operational Research Society*, 49(9):928+, 1998.

[31] H. M. Wagner. On the optimality of pure strategies. *Management Science*, 6:268–269, 1960.

[32] Y. Ye. *Interior Point Algorithms: Theory and Analysis*. Wiley-Interscience, New York, New York, 1997.

[33] A. Zadorojniy, G. Even, and A. Shwartz. A strongly polynomial algorithm for controlled queues. *Mathematics of Operations Research*, 34(4):992–1007, 2009.

[34] Q. Zhao, S. Geirhofer, L. Tong, and B. M. Sadler. Optimal dynamic spectrum access via periodic channel sensing. In *Wireless Communications and Networking Conference*, pages 33–37. IEEE, 2007.