A Gradient-Aware Search Algorithm for Constrained Markov Decision Processes

Sami Khairy®, Member, IEEE, Prasanna Balaprapaksh, and Lin X. Cai®, Senior Member, IEEE

Abstract—The canonical solution methodology for finite constrained Markov decision processes (CMDPs), where the objective is to maximize the expected infinite-horizon discounted rewards subject to the expected infinite-horizon discounted costs’ constraints, is based on convex linear programming (LP). In this brief, we first prove that the optimization objective in the dual linear program of a finite CMDP is a piecewise linear convex (PWLC) function with respect to the Lagrange penalty multipliers. Next, we propose a novel, provably optimal, two-level gradient-aware search (GAS) algorithm which exploits the PWLC structure to find the optimal state-value function and Lagrange penalty multipliers of a finite CMDP. The proposed algorithm is applied in two stochastic control problems with constraints for performance comparison with binary search (BS), Lagrangian primal–dual optimization (PDO), and LP. Compared with the benchmark algorithms, it is shown that the proposed GAS algorithm converges to the optimal solution quickly without any hyperparameter tuning. In addition, the convergence speed of the proposed algorithm is not sensitive to the initialization of the Lagrange multipliers.

Index Terms—Constrained Markov decision process (CMDP), gradient aware search (GAS), Lagrangian primal–dual optimization (PDO), piecewise linear convex (PWLC).

I. INTRODUCTION

Markov decision processes (MDPs) are the classical formalization of sequential decision-making in discrete-time stochastic control processes [1]. In MDPs, the outcomes of actions are uncertain and influence not only immediate rewards but also future rewards through next states. Policies, which are strategies for action selection, should therefore strike a tradeoff between immediate rewards and delayed rewards and be optimal in some sense. MDPs have gained recognition in diverse fields such as operations research, economics, engineering, wireless networks, artificial intelligence, vehicle dispatching [2], and learning systems [3], [4], [5]. Moreover, an MDP is a mathematically idealized form of the reinforcement learning problem, which is an active research field within the machine learning community. In many situations however, finding optimal policies with respect to a single reward function does not suffice to fully describe sequential decision-making in problems with multiple conflicting objectives [6], [7]. The framework of constrained MDPs (CMDPs) is the natural approach for handling multiobjective decision-making under uncertainty [8]. For instance, the problem of finding a feasible and optimal communication protocol that satisfies a limited bandwidth constraint can be formulated as a CMDP [9].

Algorithmic methods for solving CMDPs have been extensively studied when the underlying transition probability function is known [8], [10], [11], [12], [13], [14], [15] and unknown [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30]. In the case of finite state-action spaces with known transition probability function, the solution for a CMDP can be obtained by solving finite linear programs [8], [10], [11], or by deriving a Bellman optimality equation with respect to an augmented MDP whose state vector consists of two parts: the first part is the state of the original MDP, while the second part keeps track of the cumulative constraints’ cost [12], [13], [14], [15]. Linear programming (LP) methods become computationally impractical at a much smaller number of states than dynamic programming, by an estimated factor of about 100 [1]. In practice, MDP-specific algorithms, which are dynamic-programming-based methods, hold more promise for efficient solution [31]. While MDP augmentation-based methods provide a theoretical framework to apply dynamic programming to constrained stochastic control, they introduce continuous variables to the state space which rules out practical tabular methods and make the design of a solution algorithm more challenging [32].

On the other hand, solution methods for the constrained reinforcement learning problem, i.e., CMDPs with unknown transition probability, are generally based on Lagrangian primal–dual-type optimization. In these methods, gradient ascent is performed on state values at a fast time scale to find the optimal value function for a given set of Lagrangian multipliers, while gradient descent is performed on the Lagrangian multipliers at a slower time scale. This process is repeated until convergence to a saddle point. The existing works have explored the primal–dual optimization (PDO) approach in the tabular setting, i.e., without function approximation [16], [17], [18], [19] and with function approximators such as deep neural networks [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30]. While this approach is appealing in its simplicity, convergence to the optimal solution depends on the exploration strategy during environment interaction, function approximation errors, and the learning rate sequence. Convergence speed is also sensitive to the initialization of the Lagrange multipliers [23], [25], [33], [34]. If the learning rate is small, the Lagrange multipliers will not update quickly to enforce the constraint, and if it is high, the algorithm may oscillate around the optimal solution. In practice, a sequence of decreasing learning rates should be adopted to guarantee convergence [18], yet we do not have an obvious method to select the optimal sequence, nor can we use the first-order optimality condition to check the optimality of a solution in the case when a nonsmooth objective is not differentiable at the optimal solution.

In this brief, we develop a new approach to solve finite CMDPs with discrete state–action space and known transition probability function. Unlike the Lagrangian PDO approach which updates the Lagrange penalty multipliers by taking small steps in the steepest descent direction, our proposed algorithm uses gradient information...
to adaptively narrow down the search space of the Lagrange penalty multiplier until the algorithm terminates at the optimal solution. Consequently, convergence of our proposed gradient-aware search (GAS) algorithm to the optimal solution is not dependent on the learning rate sequence as is the case with the Lagrangian PDO approach. Furthermore, our proposed GAS algorithm terminates at the optimal Lagrange multiplier even when the objective is non-differentiable at the optimal solution and the first-order optimality condition cannot be checked. In practice, the proposed algorithm can be applied to control problems where the system dynamics are available or can be estimated based on historical data. For instance, many queue management problems such as congestion control, data center resource allocation, and inventory management can be modeled as controlled Markov chains with a set of parameters that characterize the arrival and departure distributions and which can be inferred from past data [35], [36].

The main contributions of our work can be summarized as follows. First, we analyze the geometry of the optimization objective in the dual linear program of a finite CMDP and prove that the optimization objective is a piecewise linear convex function (PWLC) with respect to the Lagrange penalty multipliers. Second, we treat the dual linear program of a finite CMDP as a search problem over the Lagrange penalty multipliers and propose a novel two-level GAS algorithm that exploits the PWLC structure to find the optimal state-value function and optimal Lagrange penalty multipliers of a CMDP. Third, the convergence of the proposed GAS algorithm to the optimal solution is analyzed and theoretically proven. Finally, we empirically compare the convergence performance of the proposed GAS algorithm with binary search (BS), Lagrangian PDO, and LP, in two application domains: robot navigation in grid world and wireless network management. Compared with benchmark algorithms, the proposed GAS algorithm does not require any hyperparameter tuning, and its convergence speed is not sensitive to the initialization of the Lagrange penalty multiplier. Furthermore, it is shown that the proposed GAS algorithm converges to the optimal solution faster than BS, and in many cases, outperforms the Lagrangian PDO approach.

The remainder of this brief is organized as follows. A background of unconstrained and constrained MDPs is given in Section II. Our proposed GAS algorithm is proposed and its convergence is analyzed in Section III. Performance evaluation of GAS in two application domains is presented in Section IV, followed by our concluding remarks and future work in Section V.

II. BACKGROUND AND RELATED WORKS

A. Unconstrained MDPs

An infinite horizon MDP with discounted returns is defined as a tuple \((S, A, P, \beta, \gamma)\), where \(S\) and \(A\) are finite sets of states and actions, respectively, \(P : S \times A \times S \to [0, 1]\) is the model’s state-action-state transition probabilities, and \(\beta : S \to [0, 1]\) is the initial distribution over the states, \(\gamma : S \times A \to \mathbb{R}\) is the reward function which maps each state-action pair to the set of real numbers \(\mathbb{R}\), and \(\gamma\) is the discount factor. Denote the transition probability from state \(s_i = i\) to state \(s_{i+1} = j\) if action \(a_i = a\) is chosen by \(P_{ij}(a) := P(s_{i+1} = j | s_i = i, a = a)\). The transition probability from state \(i\) to state \(j\) is therefore \(P_{ij} = P(s_{i+1} = j | s_i = i) = \sum_j P_{ij}(a)\), where \(\pi(a|i)\) is the adopted policy. The state-value function of state \(i\) under policy \(\pi\), \(V_\pi(i)\), is the expected discounted return starting in state \(i\) and following \(\pi\) thereafter

\[
V_\pi(i) = \sum_{i=1}^{\infty} \sum_{j,a} \gamma^{i-1} P^\pi(s_i = j, a_i = a | s_0 = i) R(j, a) \quad \forall i \in S.
\]

Let \(V_\pi\) be the vector of state values, \(V_\pi(i), \forall i\). The solution of an MDP is a Markov stationary policy \(\pi^*\) which maximizes the inner product \(\langle V_\pi, \beta \rangle = \sum_i V_\pi(i) \beta(i), \text{i.e.,}\)

\[
\max \sum_{i=1}^{\infty} \sum_{j,a} \gamma^{i-1} P^\pi(s_i = j, a_i = a) R(j, a).
\]

There exist several methods to solve (2), including LP [3] and dynamic programming methods such as value iteration and policy iteration [1]. Based on the LP formulation, the optimal \(V_\pi\) can be obtained by solving the following primal linear program [3]:

\[
\begin{align*}
\min \sum_{i=1}^{\infty} \beta(i) V(i) \\
\text{s.t. } V(i) &\geq R(i, a) + \gamma \sum_j P_{ij}(a) V(j) \quad \forall (i, a) \in S \times A,
\end{align*}
\]

Linear program (3) has \(|S|\) variables and \(|S| \times |A|\) constraints, which becomes computationally impractical to solve for MDPs with large state-action spaces. On the contrary, dynamic programming is comparatively better suited to handling large state spaces than LP and is widely considered the only feasible way of solving (2) for large MDPs [1]. Of particular interest is the value iteration algorithm which generates an optimal policy in polynomial time for a fixed \(\gamma\) [31]. In this value iteration algorithm, the state-value function is iteratively updated based on Bellman’s principle of optimality

\[
V_{k+1}(i) = \max_{a \in A(i)} \left[ R(i, a) + \gamma \sum_{j \in S} P_{ij}(a) V_k(j) \right] \quad \forall i \in S.
\]

It has been shown that the nonlinear map \(V_k \to V_k\) in (4) is a monotone contraction mapping that admits a unique fixed point, which is the optimal value function \(V_\pi\) [3]. By obtaining the optimal value function, the optimal policy can be derived using one-step lookahead: the optimal action at state \(i\) is the one that attains the equality in (4), with ties broken arbitrarily.

B. Constrained Markov Decision Processes

In CMDPs, an additional immediate cost function \(C : S \times A \to \mathbb{R}\) is augmented, such that a CMDP is defined by the tuple \((S, A, P, \beta, \gamma, C, \gamma)\) [8]. The state-value function is defined as in (1). In addition, the infinite-horizon discounted cost of a state \(i\) under policy \(\pi\) is defined as

\[
C_\pi(i) = \sum_{i=1}^{\infty} \sum_{j,a} \gamma^{i-1} P^\pi(s_i = j, a_i = a | s_0 = i) C(j, a) \quad \forall i \in S.
\]

Let \(C_\pi\) be the vector of state costs, \(C_\pi(i), \forall i, \text{ and } E \in \mathbb{R}\) a given constant which represents the constraint upper bound. The solution of a CMDP is a Markov stationary policy \(\pi^*\) which maximizes \(\langle V_\pi, \beta \rangle\) subject to a constraint \((C_\pi, \beta) \leq E\)

\[
\begin{align*}
\max \sum_{i=1}^{\infty} \sum_{j,a} \gamma^{i-1} P^\pi(s_i = j, a_i = a) R(j, a) \\
\text{s.t. } \sum_{i=1}^{\infty} \sum_{j,a} \gamma^{i-1} P^\pi(s_i = j, a_i = a) C(j, a) \leq E.
\end{align*}
\]

The canonical solution methodology for (6) is based on convex LP [8]. Of interest is the following dual linear program\(^1\) which can be solved for the optimal state-value function and Lagrange penalty

\(^1\)The primal linear program is defined over a convex set of occupancy measures, and by the Lagrangian strong duality, (7) can be obtained. Interested readers can find more details about this formulation in Chapter 3 [8].
The objective is to maximize the expected infinite-horizon discounted rewards without constraints, in which case the optimal Lagrange multiplier is \( \mu^* = 0 \). Based on assumptions 1 and 2, an optimal positive Lagrange multiplier \( \mu^* > 0 \) exists which trades off rewards and costs.

Our proposed method works in an iterative two-level optimization scheme. For a given Lagrange penalty multiplier \( \mu \), an unconstrained MDP with a penalized reward function \( \mathcal{R}(i, a) = \mathcal{R}(i, a) - \mu \mathcal{C}(i, a) \), \( \forall i, a \), is specified, and we require the corresponding optimal value function \( V_\pi^* (i, \mu) \), \( \forall i \) to be found using dynamic programming (8). Denote the optimization objective in (7) as a function of \( \mu \) by \( \mathcal{O}(\mu) \), i.e., \( \mathcal{O}(\mu) = \sum_i \beta(i)V_\pi^*(i, \mu) + \mu E \). Thus, to evaluate \( \mathcal{O}(\mu) \), one has to solve for \( V_\pi^*(i, \mu) \), \( \forall i \) first. How to efficiently search for the optimal \( \mu^* \)? To answer this question, we first prove that \( \mathcal{O}(\mu) \) is a PWLC function with respect to \( \mu \) and design an efficient solution algorithm next.

**Definition 1:** Let \( a_1, \ldots, a_m \in \mathbb{R} \), and \( b_1, \ldots, b_n \in \mathbb{R} \) for some positive integer \( i \in \mathbb{Z}^+ \). A PWLC function is given by \( f(\mu) = \max_{k=1, \ldots, m} \{a_k \mu + b_k\}\). Without loss of generality, note that \( f(\mu) \) can also be written as \( f(\mu) = \max_{k, \mu} \{a_k \mu + b_k\} \), which is PWLC.

**Theorem 1:** Given that \( V_\pi^*(i, \mu) \), \( \forall i \) is the optimal value function of an unconstrained MDP with a penalized reward function \( \mathcal{R}(i, a) = \mathcal{R}(i, a) - \mu \mathcal{C}(i, a), \forall i, a \), \( \beta(i) \) is the probability that the initial state is \( i \), and \( E \) is a constant representing the cost constraint upper bound in the original CMDP, \( \mathcal{O}(\mu) = \sum_i \beta(i)V_\pi^*(i, \mu) + \mu E \) is a PWLC function with respect to \( \mu \).

**Proof:** For ease of exposition, we introduce the following three properties of PWLC functions and show how they can be used to construct the proof.

1. \( f(\mu) + f(\mu) \) is PWLC for any \( i, j \in \mathbb{Z}^+ \). Note that for pointwise maximum, \( \max_{i=1, \ldots, m} \{a_i \mu + b_i\} = \max_{i} \{a_i \mu + b_i\} \), which is PWLC.
2. \( \max_{i} \{a_i \mu + b_i + f(\mu)\} \) is PWLC for any \( i \), \( j \in \mathbb{Z}^+ \). Note that \( \max_{i} \{a_i \mu + b_i\} = \max_{i} \{a_i \mu + b_i\} \), which is PWLC.
3. \( f(\mu) + c \mu \) is PWLC. Note that \( f(\mu) + c \mu = \max_{i} \{a_i \mu + b_i\} + c \mu = \max_{i} \{a_i + c\} \mu + b_i \), which is PWLC.

Based on these properties, the proof proceeds as follows. Given \( V_\pi^*(i, \mu) \), the recursive formula in (8) is expanded by iteratively substituting for \( V_\pi^*(i, \mu) \) with the right-hand side of (8). This expansion allows us to deduce that \( V_\pi^*(i, \mu) \) has the following functional form:

\[
V_\pi^*(i, \mu) = \max_j \{a_j \mu + b_j + \gamma \max_k \{a_k^0 \mu + b_k^0 \} + \gamma \max_m \{a_m^0 \mu + b_m^0 + \cdots \}\}
\]

for some constants \( a_i, b_j, a_k^0, b_k^0, \ldots \in \mathbb{R} \) which come from discounted rewards, costs, and their product with the transition probability function. Based on successive application of properties 1 and 2, \( V_\pi^*(i, \mu) \) are PWLC functions with respect to \( \mu \). Also, \( \sum_i \beta(i)V_\pi^*(i, \mu) \) is PWLC with respect to \( \mu \) based on property 1. Finally, \( \mathcal{O}(\mu) = \sum_i \beta(i)V_\pi^*(i, \mu) + \mu E \) is a PWLC function with respect to \( \mu \) based on property 3.

To efficiently find \( \mu^* \) and solve (7), we propose a novel GAS algorithm which exploits the PWLC structure. Our proposed GAS algorithm, which is outlined in Algorithm 1 in the Supplementary Material, operates over two loops. For a given Lagrange multiplier \( \mu^* \) (line 4) at an outer loop iteration, the inner loop finds the optimal
state-value function, \( V_\pi(i, \mu^*) \), \( \forall i \) using value iteration, as well as the gradient of \( O(\mu) \) with respect to \( \mu^* \) (lines 5–14). Based on this information, the outer loop exploits the PWLC structure to calculate the next \( \mu^* \) (line 15, lines 18–23). This process continues until a termination criterion is met (lines 16 and 17). In what follows, the proposed GAS algorithm is outlined in detail.

**Inner loop:** Let \( Q_i(a, \mu) \), \( \forall i, a \), denote the state–action value function which are the expected infinite-horizon discounted penalized state–action value function (line 6), and the state–action value function at each step \( k \) can be represented as a linear function of \( \mu^* \)

\[
Q_i(a, \mu^*) = o_i^0(a) - o_i^\mu(a, \mu^*) \tag{9}
\]

In this representation, \( o_i^0(a) \) are the expected infinite-horizon discounted rewards, and \( o_i^\mu(a) \) are the expected infinite-horizon discounted costs (line 10). Given \( o_i^0(a), o_i^\mu(a), \forall a, \text{ and } \mu^* \), the value function at state \( i \) can be obtained by \( V_i(\mu^*, \mu) = \max_a Q_i(a, \mu^*) \) (equation 8), where \( a \) denotes the action which attains the maximum state–action value among the set of actions at state \( i \), \( a(i) = \text{argmax}_a Q_i(a, \mu^*) \) (line 9). Note that in the inner loop, two functions are estimated using value iterations, the penalized state–action value function (line 6), and the state–action cost function (line 10). Dynamic programming steps in the inner loop are performed until the mean relative absolute error in the estimation of the state value function is less than a preset accuracy threshold \( \epsilon \) (lines 12–14). When the inner loop concludes, the optimal \( V_\pi(i, \mu^*) \), \( \forall i \), are returned to the outer loop, where optimality is guaranteed because the Bellman operator is monotone contraction mapping that admits a unique fixed point [1].

**Outer loop:** Given \( \mu^* \), and \( V_\pi(i, \mu^*) \), \( o_i^\mu(a, \mu^*) \), \( \forall i \), from the inner loop, \( O(\mu^*) \) and the gradient of \( O(\mu^*) \) with respect to \( \mu^* \) can be obtained based on (lines 15)

\[
O(\mu^*) = \sum_i V_\pi(i, \mu^*) \beta(i) + \mu^* E \tag{10}
\]

and

\[
\frac{\partial O}{\partial \mu^*} = \frac{\partial}{\partial \mu^*} \sum_i V_\pi(i, \mu^*) \beta(i) + \mu^* E
= \sum_i \frac{\partial}{\partial \mu^*} \max_a Q_\pi(a, i, \mu^*) \beta(i) + \mu^* E
= \sum_i \frac{\partial}{\partial \mu^*} (o^0_i(a) - o^\mu_i(a, \mu^*) \beta(i) + \mu^* E
= - \sum_i o^\mu_i(a, \mu^*) \beta(i) + E . \tag{11}
\]

Note that there is a one-to-one correspondence between the sign of the gradient in (11) and constraint satisfaction in (6). When \( (\partial O/\partial \mu^*) \) is negative, i.e., \( \sum_i o^\mu_i(a, \mu^*) \beta(i) > E \), the current policy \( \pi^*(\mu^*) \) is infeasible, which means \( \mu^* \) should be increased. On the other hand, when \( (\partial O/\partial \mu^*) \) is nonnegative, i.e., \( \sum_i o^\mu_i(a, \mu^*) \beta(i) \leq E \), the current policy is feasible, yet we can potentially obtain another feasible policy with larger returns by decreasing \( \mu^* \). Note that if \( \mu^* \) is a cusp which lies at the intersection point of two linear segments of the PWLC function \( O(\mu) \), \( (\partial O/\partial \mu^*) \) does not exist in theory, although the two one-sided derivatives (left and right) are well-defined. The nonexistence of a derivative at cusps does not pose a challenge to our proposed algorithm, because computations are accurate up to the machine’s floating point precision. To further clarify, suppose \( \mu^* \) is a cusp on the curve of \( O(\mu) \). In practice, we can only compute the gradient (11) at \( (\mu^* \pm \hat{\epsilon}) \), where \( \hat{\epsilon} \) is a very small precision loss due to rounding in floating point arithmetic, which is at the order of \( 10^{-16} \) for double precision as defined by the IEEE 754-2008 standard. This means that computing the gradient at a point is essentially computing a one-sided derivative, which always exists. Furthermore, gradient computation is numerically stable based on the result of the following lemma which states that \( O(\mu) \) is always finite for a CMDP with bounded costs.

**Lemma 1:** \( O(\mu) \) has a finite Lipschitz constant \( L_c \) such that \( \sup_{\mu}(\partial O/\partial \mu) = L_c < \infty \).

**Proof:**

\[
\left| \frac{\partial O}{\partial \mu} \right| = \left| - \sum_i o^\mu_i(i, \tilde{\alpha}) \beta(i) + E \right| \leq \left| - \sum_i o^\mu_i(i, \tilde{\alpha}) \beta(i) \right| + |E| \leq \sum_i |o^\mu_i(i, \tilde{\alpha}) \beta(i) + E| = L_c \tag{12}
\]

where the expected long-term discounted cost for any policy \( \pi^* \) is \( o^\mu_i(i, \tilde{\alpha}) < \infty, \forall i \in S \), whenever \( C(i, \mu) < \infty, \forall(i, a) \in S \times A \). \( \blacksquare \)

It is worth to mention that our proposed algorithm does not perform gradient optimization but rather exploits the structure of the PWLC objective to find the optimal \( \mu^* \). The algorithm always retains two values for \( \mu \), \( [\mu^*, \mu^-] \), where \( (\partial O/\partial \mu^+) < 0 \) and \( (\partial O/\partial \mu^-) \geq 0 \). Note that if \( \mu^- = \mu^+ \) along with \( (\partial O/\partial \mu^-) \) define a line \( L^- \), while \( (\partial O/\partial \mu^-) \) define another line \( L^+ \). \( L^- \) and \( L^+ \) intersect at a point \( \mu^* \), which is passed to the inner loop. There are two possible outcomes when running the inner loop with \( \mu^* \).

1. \( 0 \leq (\partial O/\partial \mu^+ \leq (\partial O/\partial \mu^-), \) in this case \( O(\mu^*) < O(\mu^+) \) because \( O(\mu) \) is PWLC with respect to \( \mu \), unless \( \mu^* = \mu^+ \) in which case the algorithm terminates (lines 16 and 17). The algorithm retains \( \mu^* \) and \( (\partial O/\partial \mu^+) \) by setting, \( \mu^- = \mu^+ \) and \( (\partial O/\partial \mu^-) = (\partial O/\partial \mu^+) \) (lines 20 and 21).
2. \( (\partial O/\partial \mu^+) \leq (\partial O/\partial \mu^-) < 0, \) in this case \( O(\mu^*) < O(\mu^+) \) because \( O(\mu) \) is PWLC with respect to \( \mu \), unless \( \mu^* = \mu^- \) in which case the algorithm terminates (lines 16 and 17). The algorithm retains \( \mu^- \) and \( (\partial O/\partial \mu^-) \) by setting, \( \mu^* = \mu^- \) and \( (\partial O/\partial \mu^-) = (\partial O/\partial \mu^+) \) (lines 22 and 23).

The algorithm terminates when the absolute error between \( O_{\text{min}} \) which is the minimum objective value found so far, and \( O(\mu^*) \), is less than an arbitrarily small positive number \( e \) (lines 16 and 17). Note that the initial \( \mu^* \) is set to \( M \), an arbitrarily large number which attains \( (\partial O/\partial \mu)^* \geq 0 \). Based on assumption 1, such \( M \) exists. If assumption 1 does not hold and there exists no feasible policy, then such an \( M \) does not exist. If that is the case, \( O(\mu) \) is not lower bounded and can be made arbitrarily small \( O(\mu) \rightarrow -\infty \) by taking \( \mu \rightarrow \infty \). Furthermore, \( \mu^- = 0 \) always attains a negative gradient based on assumption 2. If assumption 2 does not hold, the CMDP reduces to an MDP with \( \mu^* = 0 \) and \( (\partial O/\partial \mu^-) \geq 0 \). Fig. 1 shows a visual illustration of how Algorithm 1 in the Supplementary Material works in practice. It is worth to mention that Algorithm 1 in the Supplementary Material only evaluates \( O(\mu) \) at the points indicated by a green circle. Finally, the convergence of the proposed GAS algorithm to the optimal \( \mu^* \) is studied in Theorem 2.

**Theorem 2:** The proposed GAS algorithm converges to the optimal \( \mu^* \) in finitely many outer loop iterations.

**Proof:** Recall that \( \mu^* \) is the intersection point of the two lines \( L^+ \), \( L^- \). If \( L^+ \) has a zero gradient, that is, \( (\partial O/\partial \mu^+) = 0 \), then \( O(\mu) \) has infinitely many local minima including \( \mu^* \), which lie on \( L^+ \) and attain the same value of \( O(\mu^*) \). A similar argument can be made for \( L^- \).
the step size parameter. In our experiments, \( M = 2/(1 - \gamma) \), \( \delta = 0.05 \), \( \gamma = 0.99 \), and 30 obstacles are deployed. We follow [32] in the choice of parameters, which trades off high penalty for obstacle collisions and computational complexity. \( \Omega(\mu) \) is plotted in Fig. 2(a) over \( \mu \in [0, 200] \). It can be seen from Fig. 2(a) that \( \Omega(\mu) \) is a PWLC function as given by Theorem 1. It can also be seen that the global minima is a nondifferentiable cusp around 90 with unequal one-sided derivatives.

In Fig. 2(b), we compare the convergence performance of the proposed GAS with BS, by plotting the number of outer loop iterations versus an accuracy level \( \epsilon' \). The initial search window for \( \mu \in [0, M]^3 \) and \( M \) is either \( 10^5 \) or \( 10^3 \). Given a value for \( \mu \), the inner loop evaluates (10) and its gradient (11). The outer loop then determines the next \( \mu \) based on either the proposed GAS or BS. This iterative procedures continue until the convergence criterion is met. Compared with BS, the proposed GAS algorithm requires a smaller number of outer loop iterations for all levels of accuracy thresholds \( \epsilon' \). This is because GAS adaptively shrinks the search window by evaluating points at the intersection of line segments with nonnegative and negative gradients, whereas BS blindly shrinks the window size by (1/2) every iteration. These results demonstrate that \( \epsilon' \) can be set arbitrarily small and \( M \) can be set arbitrarily large without substantially impacting the convergence time.

In Fig. 2(c), we compare the convergence performance of the proposed GAS with the Lagrangian PDO, by plotting the total number of value iterations, which is the sum of the number of value iterations over all the outer loop iterations, versus the learning rate decay parameter in a log–log scale. In the Lagrangian PDO, gradient descent on \( \mu \) is performed along side dynamic programming iterations on \( V(i), V_i \), which motivates us to compare the convergence performance in terms of total number of value iterations. In Lagrangian PDO, the update rule for \( \mu \) is \( \mu_{k+1} = \mu_k - k_t (\partial \Omega/\partial \mu_k) \), where \( k_t \) is the step size parameter. In our experiments, \( k_t \) is initially set to 1 to speed up descending toward the minima and is decayed according to \( k_{t+1} = k_t e^{-\xi T} \), where \( \xi \) is the learning rate decay parameter, and \( T \) is

V. Case Studies

In this section, we evaluate the efficacy of the proposed GAS algorithm across two distinct domains: grid world robot navigation and solar-powered UAV-based wireless network management. To ensure a fair evaluation, we compare the performance of GAS with BS, Lagrangian PDO, and LP, as these methods require the complete knowledge of the system dynamics and directly compute the optimal policy based on this information. In all the experiments, both \( \epsilon \) and \( \epsilon' \) are set to \( 10^{-10} \), unless stated otherwise.

A. Grid World Robot Navigation

In the grid world robot navigation problem, the agent controls the movement of a robot in a rectangular grid world, where states represent grid points on a 2-D terrain map. The robot starts in a safe region in the bottom right corner (blue square) and is required to travel to the goal located in the top right corner (green square). At each time step, the agent can move in either of the four directions to one of the neighboring states. However, the movement direction of the agent is stochastic and partially depends on the chosen direction. Specifically, with probability \( 1 - \delta \), the robot will move in the chosen direction, and uniformly randomly otherwise, i.e., with probability \( \delta/4 \) the robot will move to one of the four neighboring states. At each time step, the agent receives a reward of \(-1\) to account for fuel usage. Upon reaching the goal, the agent receives a positive reward of \( M \gg 1 \). In between the starting and destination states, there are a number of obstacles (red squares) that the agent should avoid. Hitting an obstacle costs the agent \( M \).

It is important to note that the 2-D terrain map is built such that a shorter path induces higher risk of hitting obstacles. By maximizing expected infinite-horizon discounted rewards subject to expected infinite-horizon discounted costs, the agent finds the shortest path from the starting state to the destination state such that the toll for hitting obstacles does not exceed an upper bound. This problem is inspired by classic grid world problems in the literature [22], [32].

1) Experimental Results: We choose a grid of 20 \( \times \) 20 with a total of 400 states. The start state is (2, 18), the destination is (19, 18), \( M = 2/(1 - \gamma) \), \( \delta = 0.05 \), \( \gamma = 0.99 \), and 30 obstacles are deployed. We follow [32] in the choice of parameters, which trades off high penalty for obstacle collisions and computational complexity. \( \Omega(\mu) \) is plotted in Fig. 2(a) over \( \mu \in [0, 200] \). It can be seen from Fig. 2(a) that \( \Omega(\mu) \) is a PWLC function as given by Theorem 1. It can also be seen that the global minima is a nondifferentiable cusp around 90 with unequal one-sided derivatives.

In Fig. 2(b), we compare the convergence performance of the proposed GAS with BS, by plotting the number of outer loop iterations versus an accuracy level \( \epsilon' \). The initial search window for \( \mu \in [0, M]^3 \) and \( M \) is either \( 10^5 \) or \( 10^3 \). Given a value for \( \mu \), the inner loop evaluates (10) and its gradient (11). The outer loop then determines the next \( \mu \) based on either the proposed GAS or BS. This iterative procedures continue until the convergence criterion is met. Compared with BS, the proposed GAS algorithm requires a smaller number of outer loop iterations for all levels of accuracy thresholds \( \epsilon' \). This is because GAS adaptively shrinks the search window by evaluating points at the intersection of line segments with nonnegative and negative gradients, whereas BS blindly shrinks the window size by (1/2) every iteration. These results demonstrate that \( \epsilon' \) can be set arbitrarily small and \( M \) can be set arbitrarily large without substantially impacting the convergence time.

In Fig. 2(c), we compare the convergence performance of the proposed GAS with the Lagrangian PDO, by plotting the total number of value iterations, which is the sum of the number of value iterations over all the outer loop iterations, versus the learning rate decay parameter in a log–log scale. In the Lagrangian PDO, gradient descent on \( \mu \) is performed along side dynamic programming iterations on \( V(i), V_i \), which motivates us to compare the convergence performance in terms of total number of value iterations. In Lagrangian PDO, the update rule for \( \mu \) is \( \mu_{k+1} = \mu_k - k_t (\partial \Omega/\partial \mu_k) \), where \( k_t \) is the step size parameter. In our experiments, \( k_t \) is initially set to 1 to speed up descending toward the minima and is decayed according to \( k_{t+1} = k_t e^{-\xi T} \), where \( \xi \) is the learning rate decay parameter, and \( T \) is

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Fig. 2. Performance comparison on the grid world robot navigation problem. (a) PWLC function \( O(\mu) \). (b) Comparison with BS. (c) Comparison with Lagrangian PDO.

Fig. 3. Filled contour maps of the value function and the learned control policy for the grid world robot navigation problem. With a lower upper bound on the cost constraint \( E \), the policy is more risk-averse and achieves a higher probability of success for reaching the goal \( P_s^{\pi^*} \). The shown path is one realization based on the learned policy. (a) \( P_s^{\pi^*} = 76.85\% \), \( E = 160 \). (b) \( P_s^{\pi^*} = 79.65\% \), \( E = 40 \). (c) \( P_s^{\pi^*} = 93.55\% \), \( E = 20 \). (d) \( P_s^{\pi^*} = 99.90\% \), \( E = 5 \).

In Table I, we compare the results obtained by the proposed GAS algorithm at convergence with those obtained by solving (7) using LP. We use the LP solver provided by Gurobi, which is arguably the most powerful mathematical optimization solver in the market. The optimal \( \mu^*, O(\mu^*), \min_{i} [BE(i)], E_{i} [BE(i)] \), and \( \max_{i} [BE(i)] \) are reported, where \( BE(i) \) is the Bellman error at state \( i \) given by \( BE(i) = V_{\pi^*}(i, \mu^*) - \max_{a \in A_0} [R(i, a) - \mu^* C(i, a) + \gamma \sum_{j \in S} P_{ij}(a) V_{\pi^*}(j, \mu^*)], \forall i \in S \). Note that if the state-value function corresponds to an optimal policy \( \pi^* \), then the Bellman error will be zero for all the states. This motivates us to look at the min, max and expected absolute Bellman error. It can be observed that Gurobi’s LP solver converges to a suboptimal solution with a higher value compared with the proposed GAS algorithm (recall that (7) is a minimization problem, hence the lower \( O(\mu) \) the better). This demonstrates that generic LP solvers may struggle to solve CMDPs, which has motivated the research community to develop CMDP-specific algorithms.

In Fig. 3, filled contour maps for the value function are plotted. Regions which have the same value have the same color shade. The value function is plotted for four different values of the cost constraint upper bound, along with the start state (blue square), destination (green square), and obstacles (red squares). With a tighter cost constraint \( (E = 5) \), a safer policy is learned in which the agent takes a long path to go around the obstacles as in Fig. 3(d), successfully reaching the destination \( P_s^{\pi^*} = 99.90\% \) of the times. When the cost constraint is loose \( (E = 160) \), a riskier policy is learned in which the agent takes a shorter path by navigating between the obstacles as in Fig. 3(a), successfully reaching the destination \( P_s^{\pi^*} = 76.85\% \) of the times, based on 2000 roll outs.

### Table I

| Grid World Robot Navigation | Solar-Powered UAV-Based Wireless Network |
|------------------------------|------------------------------------------|
| \( \mu^* \)                  | \( 68.33333 \)                            |
| \( \min_{i} [BE(i)] \)       | \( 0.0 \)                                 |
| \( E_{i} [BE(i)] \)          | \( 9.89e-10 \)                            |
| \( \max_{i} [BE(i)] \)       | \( 1.13e-07 \)                            |
| \( O(\mu^*) \)               | \( 15216.83 \)                            |

Fig. 4. Performance comparison on the solar-powered UAV-based wireless network management problem. Comparison with (a) BS and (b) Lagrangian approach.

**B. Solar-Powered UAV-Based Wireless Networks**

As a second application domain, we study a wireless network management problem, which is of interest to the wireless networking research community. We consider a solar-powered UAV-based wireless network consisting of one UAV which receives data from...
remote network servers through satellite backhaul links and relays the received data to $N$ wireless devices on the ground. To manage the network optimally, the agent controls the altitude of the UAV and transmission parameters of the wireless transceiver mounted on the UAV, to maximize the wireless coverage probability while ensuring energy sustainability of the UAV.

Specifically, wireless devices are independently and uniformly deployed within a circular geographical area $\Lambda$ of radius $R_c$. The UAV is deployed at the center of $\Lambda$ and is equipped with solar panels that harvest solar energy to replenish the on-board battery. Because solar light is attenuated through the cloud cover, solar energy output from the solar panels is highest above the cloud cover and decays exponentially through the clouds. System time is slotted into fixed-length discrete time units indexed by $t$. During a time slot $t$, the solar-powered UAV provides downlink wireless coverage to wireless devices on the ground. The UAV is equipped with an intelligent controller which at each time slot $t$ controls its altitude $z_t$, its directional antenna half power beamwidth angle $\theta_g$, and its transmission power $P_{TX}^t$ in dBm, based on its battery energy and altitude state. The objective of the controller is to maximize the wireless coverage probability for the worst case edge user, while ensuring energy sustainability of the solar-powered UAV. Note that the worst case edge user is the user which is farthest from the UAV. Due to the physics of the wireless channel, the received signal at the edge user from the UAV is attenuated the most, resulting in a low signal-to-noise ratio (SNR) and decreased coverage probability. By maximizing the coverage probability of the edge user, the communication performance is improved for every user in the network, albeit not proportionally.

This wireless communication system exhibits a tradeoff between the altitude of the UAV and coverage probability. When the UAV hovers at a higher altitude, it can harvest more solar energy to replenish its on-board battery storage. However, at higher altitudes, the wireless coverage probability is worse due to distance-dependent free-space path loss of RF signal propagation. By adaptively controlling $z_t$, $P_{TX}^t$, and $\theta_g$, coverage probability for the edge user can be maximized subject to energy sustainability constraints that ensure the UAV’s battery is not depleted. The detail problem modeling, CMDP formulation, and simulation parameters are delegated to the Supplementary Material due to page limitation.

1) Experimental Results: The CMDP for this problem has $|S| = 3025$ states and $|A| = 12$. In Fig. 4(b) and (c), we compare the convergence performance of the proposed GAS with BS and the Lagrangian PDO approach, respectively. As previously noted from Fig. 2(b) and (c), it can be see that the proposed algorithm compares favorably to BS and Lagrangian PDO, despite the increased problem size. From Table I, we can observe that both the proposed GAS and Gurobi’s LP solver converge to the same $\mu^{*}$, although GAS achieves a lower Bellman error in the estimation of the value function.

The learned policy by our proposed GAS algorithm is shown in Fig. 5. It can be seen from Fig. 5(a) that the agent learns an adaptive policy in which the UAV climbs up to recharge its on-board battery, and then climbs down when the battery is full to improve the coverage probability for the worst case edge user, as can be seen from Fig. 5(c) and (d). In addition, Fig. 5(b) shows the learned control policy for the transmission power and half power beamwidth angle. When the UAV is up to charge its battery, the lower transmission power and smaller beamwidth angle are selected. On the other hand, when the UAV is down to improve the coverage probability, the higher transmission power and larger beamwidth angle are selected. This is because with a smaller beamwidth angle, the antenna gain is higher, and a lower transmission power is required to counter the effects of large-scale fading.

V. CONCLUSION

In this brief, we have proved that the optimization objective in the dual LP formulation of a CMDP is a PWLC function with respect to the Lagrange penalty multiplier. Based on this result, a novel, provably optimal two-level GAS algorithm which exploits the PWLC structure has been proposed to find the optimal state-value function and Lagrange penalty multiplier of a CMDP. Performance comparison with benchmark algorithms on two stochastic control problems with constraints from different domains has shown the merits of the proposed algorithm. Specifically, the proposed GAS algorithm converges to the optimal solution quickly without any hyperparameter tuning, and its convergence speed is not sensitive to the initialization of the Lagrange penalty multiplier. In our future work, we will leverage the theoretical contributions of this work to design an efficient constrained model-free reinforcement learning algorithm.

ACKNOWLEDGMENT

The authors gratefully acknowledge the computing resources provided on Bebop, a high-performance computing cluster operated by the Laboratory Computing Resource Center, Argonne National Laboratory, Lemont, IL, USA.

REFERENCES

[1] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction. Cambridge, MA, USA: MIT Press, 2018.
[2] E. Liang, K. Wen, W. H. Lam, A. Sumalee, and R. Zhong, “An integrated reinforcement learning and centralized programming approach for online taxi dispatching,” IEEE Trans. Neural Netw. Learn. Syst., vol. 33, no. 9, pp. 4742–4756, Sep. 2021.
[3] A. Ray, J. Achiam, and D. Amodei, “Benchmarking safe exploration in deep reinforcement learning,” vol. 7, no. 1, p. 2, 2019, arXiv:1910.01708.
[4] T. Alpcan and I. Shames, “An information-based learning approach to dual control,” IEEE Trans. Neural Netw. Learn. Syst., vol. 26, no. 11, pp. 2736–2748, Nov. 2015.
[5] S. Xie, W. Zhong, K. Xie, R. Yu, and Y. Zhang, “Fair energy scheduling for vehicle-to-grid networks using adaptive dynamic programming,” IEEE Trans. Neural Netw. Learn. Syst., vol. 27, no. 8, pp. 1697–1707, Aug. 2016.
[6] S. Mannor and N. Shimkin, “A geometric approach to multi-criterion reinforcement learning,” J. Mach. Learn. Res., vol. 5, pp. 325–360, Dec. 2004.
[7] K. Van Moofaert and A. Nowé, “Multi-objective reinforcement learning using sets of Pareto dominating policies,” J. Mach. Learn. Res., vol. 15, no. 1, pp. 3483–3512, 2014.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
[8] E. Altman, *Constrained Markov Decision Processes*, vol. 7. Boca Raton, FL, USA: CRC Press, 1999.
[9] G. Hu, Y. Zha, D. Zhao, M. Zhao, and J. Hao, “Event-triggered communication network with limited-bandwidth constraint for multi-agent reinforcement learning,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 8, pp. 3966–3978, Aug. 2023.
[10] D. Dolgov and E. Durfee, “Stationary deterministic policies for constrained MDPs with multiple rewards, costs, and discount factors,” in *Proc. Int. Joint Conf. Artif. Intell.*, vol. 19, 2005, p. 1326.
[11] A. Zadorojniy and A. Shwartz, “Robustness of policies in constrained Markov decision processes,” *IEEE Trans. Autom. Control*, vol. 51, no. 4, pp. 635–638, Apr. 2006.
[12] A. B. Piunovskiy and X. Mao, “Constrained Markovian decision processes: The dynamic programming approach,” *Operations Res. Lett.*, vol. 27, no. 3, pp. 119–126, Oct. 2000.
[13] R. C. Chen and G. L. Blankenship, “Dynamic programming equations for discounted constrained stochastic control,” *IEEE Trans. Autom. Control*, vol. 49, no. 5, pp. 699–709, May 2004.
[14] A. B. Piunovskiy, “Dynamic programming in constrained Markov decision processes,” *Control Cybern.*, vol. 35, no. 3, p. 645, 2006.
[15] R. C. Chen and E. A. Feinberg, “Non-randomized policies for constrained Markov decision processes,” *Math. Methods Operations Res.*, vol. 66, no. 1, pp. 165–179, Jul. 2007.
[16] P. Geibel and F. Wysotzki, “Risk-sensitive reinforcement learning applied to control under constraints,” *J. Artif. Intell. Res.*, vol. 24, pp. 81–108, Jul. 2005.
[17] P. Geibel and F. Wysotzki, “Learning algorithms for discounted MDPs with constraints,” *Int. J. Math., Game Theory, Algebra*, vol. 21, nos. 2–3, p. 241, 2012.
[18] V. S. Borkar, “An actor-critic algorithm for constrained Markov decision processes,” *Syst. Control Lett.*, vol. 54, no. 3, pp. 207–213, Mar. 2005.
[19] F. V. Abad, V. Krishnamurthy, I. Baltcheva, and K. Martin, “Self learning control of constrained Markov decision processes—A gradient approach,” in *Proc. IEEE Conf. Decis. Control*, Dec. 2002, pp. 1940–1945.
[20] S. Bhatnagar, “An actor-critic algorithm with function approximation for discounted cost constrained Markov decision processes,” *Syst. Control Lett.*, vol. 59, no. 12, pp. 760–766, Dec. 2010.
[21] S. Bhatnagar and K. Lakshmanan, “An online actor–critic algorithm with function approximation for constrained Markov decision processes,” *J. Optim. Theory Appl.*, vol. 153, no. 3, pp. 688–708, Jun. 2012.
[22] C. Tessler, D. J. Mankowitz, and S. Mannor, “Reward constrained policy optimization,” 2018, arXiv:1805.11074.
[23] Q. Liang, F. Que, and E. Modiano, “Accelerated primal–dual policy optimization for safe reinforcement learning,” 2018, arXiv:1802.06480.
[24] M. Fu et al., “Risk-sensitive reinforcement learning: A constrained optimization viewpoint,” 2018, arXiv:1810.09126.
[25] J. Achiam, D. Held, A. Tamar, and P. Abbeel, “Constrained policy optimization,” in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 22–31.
[26] L. Prashanth and M. Ghavamzadeh, “Actor-critic algorithms for risk-sensitive reinforcement learning,” 2014, arXiv:1403.6530.
[27] S. Paternain, M. Calvo-Fullana, L. F. Chamon, and A. Ribeiro, “Safe policies for reinforcement learning via primal-dual methods,” 2019, arXiv:1911.09101.
[28] D. Ding, X. Wei, Z. Yang, Z. Wang, and M. R. Jovanović, “Provably efficient safe exploration via primal-dual policy optimization,” 2020, arXiv:2003.00534.
[29] A. Ray, J. Achiam, and D. Amodei, *Benchmarking Safe Exploration in Deep Reinforcement Learning*.
[30] S. Khairy, P. Balaprakash, L. X. Cai, and Y. Cheng, “Constrained deep reinforcement learning for energy sustainable multi- UAV based random access IoT networks with NOMA,” *IEEE J. Sel. Areas Commun.*, vol. 39, no. 4, pp. 1101–1115, 2020.
[31] M. L. Littman, T. L. Dean, and L. P. Kaelbling, “On the complexity of solving Markov decision problems,” 2013, arXiv:1302.4971.
[32] Y. Chow, A. Tamar, S. Mannor, and M. Pavone, “Risk-sensitive and robust decision-making: A CVAR optimization approach,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1522–1530.
[33] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone, “Risk-constrained reinforcement learning with percentile risk criteria,” *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6070–6120, 2017.
[34] Y. Liu, J. Ding, and X. Liu, “IPO: Interior-point policy optimization under constraints,” 2019, arXiv:1910.09615.
[35] X. Yu, S. Gao, X. Hu, and H. Park, “A Markov decision process approach to vacant taxi routing with e-hailing,” *Transp. Res. B, Methodol.*, vol. 121, pp. 114–134, Mar. 2019.
[36] X. Zhou et al., “Optimizing taxi driver profit efficiency: A spatial network-based Markov decision process approach,” *IEEE Trans. Big Data*, vol. 6, no. 1, pp. 145–158, Mar. 2018.
[37] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, “Efficient deployment of multiple unmanned aerial vehicles for optimal wireless coverage,” *IEEE Commun. Lett.*, vol. 20, no. 8, pp. 1647–1650, Aug. 2016.
[38] Y. Sun, D. Xu, D. W. K. Ng, L. Dai, and R. Schober, “Optimal 3D-trajectory design and resource allocation for solar-powered UAV communication systems,” *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4281–4298, Jun. 2019.
[39] J.-M. Garcia, O. Brun, and D. Gauchard, “Transient analytical solution of M/D/1/N queues,” *J. Appl. Probab.*, vol. 39, no. 4, pp. 853–864, Dec. 2002.
[40] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017, arXiv:1707.06347.