Scalable Gradient Ascent for Controllers in Constrained POMDPs

Kyle Hollins Wray and Kenneth Czuprynski

***Abstract*— This paper presents a novel gradient ascent algorithm and nonlinear programming algorithm for finite state controller policies in constrained partially observable Markov decision processes (CPOMDPs). A key component of the gradient ascent algorithm is a constraint projection to ensure constraints are satisfied. Both an optimal and an approximate projection are formally defined. A theoretical analysis of the algorithm and its projections is presented, formally proving aspects of projection correctness and algorithm convergence. Experiments evaluate the baseline and novel algorithms, as well as both constraint projections, on seven CPOMDP benchmark domains. The proposed novel algorithm is demonstrated on an actual robot performing a navigation task in a real household environment.**

I. Introduction

The constrained partially observable Markov decision process (CPOMDP) is a single agent decision-making model [1] that generalizes the POMDP [2] to include multiple objectives [3], [4], [5]. Specifically, the CPOMDP's objective is to maximize expected reward for a primary task objective, subject to the constraints that multiple additional objectives have expected costs less than or equal to their allotted budgets. Multi-objective decision-making in robotics has also enjoyed recent growing interest, with applications ranging from planetary rovers [6] to autonomous vehicles [7]. In these domains, the primary navigation task is subject to operation within a measure of expected safety, among other potential objectives [6], [8]. The growing number of these real world domains result in even larger CPOMDPs. Unfortunately, the current state-of-the-art algorithms struggle to scale, primarily due to the optimization complexities to ensure these constraints are satisfied, especially in belief point-based policy representations. In this paper, we propose a novel optimization formulation for finite state controller policy representations, and solve it using a mathematically-grounded scalable projected gradient ascent algorithm.

Finite state controllers represent the policy by a collection of abstract nodes [9]. Each node has a different action selection distribution function. After performing an action, the next node is selected by a successor selection distribution function, and can be unique based on the current node and observation made. This differs from a belief point-based representation, which uses a large collection sampled beliefs, each with a corresponding value hyperplane called α-vectors [10]. At a current belief, actions are chosen by finding the best α-vector's associated action. After performing an action, a belief update mechanism is used.

Controller-based algorithms for POMDPs benefit from a formal mathematical nonlinear programming (NLP) problem statement, which can be solved given the desired fixed number of nodes as a parameter [11]. Compared to point-based policies, the controller policy representation itself is not dependent on the state. It also does not require sampling beliefs, which is beneficial given that beliefs tend to grow exponentially with the number of states. This also tends to produce more compact policies. This NLP approach has been successfully extended to multiagent decentralized POMDPs and partially observable Markov games [11], [12], but to the best of our knowledge has not been explored in CPOMDPs. The NLP has also been solved directly using POMDP gradient ascent [13]. While prior CPOMDP work has mentioned the desire to use gradient ascent [1], it did not provide a formal optimization statement or an algorithm, instead proposing a genetic algorithm with limited success. One reason is that it is not obvious how best to enforce the CPOMDP constraints without sacrificing scalability.

Approaches have instead favored a belief point-based policy representation, combining clever heuristic methods to approximately enforce the constraints. Constrained point-base value iteration (CPBVI) for CPOMDPs [14] extends PBVI [10] by repeatedly solving a minimax quadratically constrained program to prune α-vectors and ensure constraints are satisfied. Risk-bounded AO* (RAO*), a heuristic search algorithm, was proposed for a CPOMDP-related model called a chance-constrained POMDP (CC-POMDP) [6]. Constrained approximate linear programming (CALP) [15] instead iteratively solves an approximate LP over a smaller subset of repeatedly sampled beliefs. These methods, however, suffer from the drawbacks of belief-point-based policies and can lack a formal connection to the underlying optimization problem.

We present projected gradient ascent (PGA) for finite state controller policies in CPOMDPs. PGA iteratively performs line search along the gradient then projects back to the space of constraint-satisfying policies. PGA proposes using an efficient golden section search for CPOMDPs. A key contribution in PGA is the novel projection step that integrates the constraint. We propose both the optimal as well as a highly scalable and effective approximate projection that obviates non-linearities in the constraining objectives.

Our contributions are: (1) a formal statement of the optimization problem (Section III); (2) a novel gradient ascent algorithm with its optimal and approximate projection formalizations (Section IV); (3) formal proofs of projection correctness and properties of convergence (Section V); and (4) experiments in seven benchmark domains including a complete implementation of a navigate and search domain on a real robot acting in a real home (Section VI).

II. Background

A *constrained partially observable Markov decision process (CPOMDP)* [1] is defined by the tuple $\langle S, A, \Omega, T, O, R, \mathbf{C}, \beta \rangle$. S, A, and Ω are finite sets of states, actions, and observations, respectively. $T : S \times A \times S \to [0,1]$ is a state transition denoting the probability $T(s,a,s') = Pr(s' \mid s,a)$ of transitioning to state s' given action a was performed in state s. $O : S \times \Omega \to [0,1]$ is an observation function denoting the probability $O(s',\omega) = Pr(\omega \mid s')$ of observing ω after transitioning to state s'. $R : S \times A \to \mathbb{R}$ denotes the primary reward $R(s,a)$ for performing action a in state s. $\mathbf{C} = [C_1, \ldots, C_m]^T$ denotes a vector of m constraint costs. Each $C_i : S \times A \to \mathbb{R}^+$ denotes the cost $C_i(s,a)$ for performing action a in state s. This can be written in a vector form $\mathbf{C}(s,a) = [C_1(s,a), \ldots, C_m(s,a)]^T$ as well. $\beta = [\beta_1, \ldots, \beta_m]^T$ denotes a vector of budgets. Each $\beta_i \geq 0$ denotes the maximum expected cost allowed for cost i.

The agent maintains a belief over the true state $b \in \triangle^{|S|}$ starting from an initial belief b_0. $\triangle^n$ refers to the $n-1$ simplex. For a belief b, performing an action a, and observing ω, the next belief b' over each state s' is computed by:

$$b'(s') \propto O(s',\omega) \sum_{s \in S} T(s,a,s') b(s). \tag{1}$$

An agent's *policy* π is used to select actions based on an internally maintained state. There are two leading policy representations. *Belief point-based policies* [10] directly map each belief to an action $\pi : \triangle^{|S|} \to A$. These policies require the maintained internal belief state during execution.

Finite state controller policies [9] maintain an abstract internal node state. Formally, a controller policy is defined by the tuple $\pi = \langle X, \psi, \eta \rangle$. X is a set of abstract nodes. $\psi : X \times A \to [0,1]$ is an action selection function denoting the probability $\psi(x,a) = Pr(a \mid x)$ of selecting action a in node x. $\eta : X \times \Omega \times X \to [0,1]$ is a node transition function denoting the probability $\eta(x,\omega,x') = Pr(x' \mid x,\omega)$ of transitioning to node x' after making observation ω from node x.

The expected *values* of the reward $\mathbf{v}_r^\pi$ and the costs $\mathbf{v}_i^\pi$ are computed by the Bellman equation [1], [13]:

$$\mathbf{v}_r^\pi = \mathbf{r}^\pi + \gamma \mathbf{M}^\pi \mathbf{v}_r^\pi \qquad \text{and} \qquad \mathbf{v}_i^\pi = \mathbf{c}_i^\pi + \gamma \mathbf{M}^\pi \mathbf{v}_i^\pi \tag{2}$$

where each $\mathbf{v}_r^\pi, \mathbf{r}^\pi, \mathbf{v}_i^\pi, \mathbf{c}_i^\pi \in \mathbb{R}^{|X \times S|}$ denotes the values, reward, or costs of each of node-state pair. The vector of rewards and costs are denoted by $\mathbf{r}^\pi(\langle x,s \rangle) = \sum_a \psi(x,a) R(s,a)$ and $\mathbf{c}_i^\pi(\langle x,s \rangle) = \sum_a \psi(x,a) C_i(s,a)$. The matrix $\mathbf{M}^\pi \in \mathbb{R}^{|X \times S| \times |X \times S|}$ denotes the transitions from each node-state pair $\langle x,s \rangle$ to $\langle x',s' \rangle$ [13]:

$$\mathbf{M}^\pi(\langle x,s \rangle, \langle x',s' \rangle) = \sum_{a \in A} \psi(x,a) T(s,a,s') \sum_{\omega \in \Omega} O(s',\omega) \eta(x,\omega,x'). \tag{3}$$

After solving for the node-state values in Equation 2, the final values of the policy at the initial node x_0 and belief b_0 can be computed by $\delta_0^\intercal \mathbf{v}_r^\pi$ and $\delta_0^\intercal \mathbf{v}_i^\pi$ with $\delta_0(\langle x,s \rangle) = b_0(s)[x = x_0]$ with Iverson bracket $[\cdot]$. The objective in CPOMDPs is to find a policy π maximize $\delta_0^\intercal \mathbf{v}_r^\pi$ subject to each $\delta_0^\intercal \mathbf{v}_i^\pi \leq \beta_i$.

III. Optimization Formulation

Formally, the constrained optimization can be written as:

$$\begin{aligned} \underset{\pi, \mathbf{v}_r^\pi, \mathbf{v}_1^\pi, \ldots, \mathbf{v}_k^\pi}{\text{maximize}} \quad & \delta_0^\intercal \mathbf{v}_r^\pi & & (4) \\ \text{subject to} \quad & \psi(x,\cdot) \in \triangle^{|A|}, & \text{for } x \in X, & \\ & \eta(x,\omega,\cdot) \in \triangle^{|X|}, & \text{for } x \in X, \omega \in \Omega, & \\ & \mathbf{v}_r^\pi = r^\pi + \gamma \mathbf{M}^\pi \mathbf{v}_r^\pi, & & \\ & \mathbf{v}_i^\pi = c_i^\pi + \gamma \mathbf{M}^\pi \mathbf{v}_i^\pi, & \text{for } i \in \{1, \ldots, m\}, & \\ & \delta_0^\intercal \mathbf{v}_i^\pi \leq \beta_i, & \text{for } i \in \{1, \ldots, m\}. & \end{aligned}$$

The equality constraints enforce satisfaction of the Bellman equations. The inequality constraints enforce the expected costs of the constraining value functions. This novel formulation to solve CPOMDPs can be solved using off-the-shelf nonlinear solvers, as was successfully done in standard POMDPs [11]. This approach is explored in Section VI.

An alternative means of expressing Equation 4 is to eliminate $\mathbf{v}^\pi$ entirely by considering optimization purely with respect to policy parameters. This can be done because value can be viewed as a dependent variable through the Bellman equations. Rewriting Equation 2 yields:

$$\mathbf{v}_r^\pi = (\mathbf{I} - \gamma \mathbf{M}^\pi)^{-1} \mathbf{r}^\pi \qquad \text{and} \qquad \mathbf{v}_i^\pi = (\mathbf{I} - \gamma \mathbf{M}^\pi)^{-1} \mathbf{c}_i^\pi. \tag{5}$$

This equivalent constrained optimization is:

$$\begin{aligned} \underset{\pi}{\text{maximize}} \quad & \delta_0^\intercal (\mathbf{I} - \gamma \mathbf{M}^\pi)^{-1} \mathbf{r}^\pi & & (6) \\ \text{subject to} \quad & \psi(x,\cdot) \in \triangle^{|A|}, & \text{for } x \in X, & \\ & \eta(x,\omega,\cdot) \in \triangle^{|X|}, & \text{for } x \in X, \omega \in \Omega, & \\ & \delta_0^\intercal (\mathbf{I} - \gamma \mathbf{M}^\pi)^{-1} c_i^\pi \leq \beta_i, & \text{for } i \in \{1, \ldots, m\}. & \end{aligned}$$

The Bellman equations are still guaranteed to be satisfied as it is used in the formulation of the objective function and constraints. This form benefits from far fewer constraints compared to Equation 4. The drawback is that this requires computing an inverse, and introduces non-convexities with respect to the policy. Thus the projection onto the constraint space does not have a clear simplification in the context of Projected Gradient Ascent.

We propose a novel gradient ascent algorithm for solving Equation 6 with an efficient approximate projection method.

IV. Controller Gradient Ascent for CPOMDPs

We begin by rewriting the optimization in Equation 6 with more compact notation to simplify presentation. Let

$$f(\pi) = \delta_0^\intercal (\mathbf{I} - \gamma \mathbf{M}^\pi)^{-1} \mathbf{r}^\pi, \tag{7}$$

denote the objective and $\mathbf{h}(\pi) = [h_1(\pi), \ldots, h_m(\pi)]^\intercal$ with

$$h_i(\pi) = \delta_0^\intercal (\mathbf{I} - \gamma \mathbf{M}^\pi)^{-1} \mathbf{c}_i^\pi, \quad \text{for } i \in \{1, \ldots, m\} \tag{8}$$

be used to denote the inequality constraints with π referring to the controller's parameters $\psi(x,a)$ and $\eta(x,\omega,x')$. Let $n = |X||A| + |X \times \Omega||X|$ be the total number of unknown policy parameters. Let $\mathbf{J} \in \mathbb{R}^{(|X| + |X \times \Omega|) \times n}$ be defined using

two vectors of ones $\mathbf{1}_\psi \in \mathbb{R}^{|A|}$ and $\mathbf{1}_\eta \in \mathbb{R}^{|X|}$ to form $\mathbf{J}_\psi = \text{blockdiag}(\mathbf{1}_\psi^\intercal, \ldots, \mathbf{1}_\psi^\intercal)$ and $\mathbf{J}_\eta = \text{blockdiag}(\mathbf{1}_\eta^\intercal, \ldots, \mathbf{1}_\eta^\intercal)$:

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_\psi & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_\eta \end{bmatrix}.$$

Let us represent the policy as a vector $\pi \in \mathbb{R}^n$, stacking ψ and η in order. The product $\mathbf{J}\pi$ can now be used to enforce probability summation to unity.

The optimization problem defined in Equation 6 is then:

$$\begin{aligned} \underset{\pi}{\text{maximize}} \quad & f(\pi) \\ \text{subject to} \quad & \mathbf{J}\pi = \mathbf{1}, \\ & \pi \geq \mathbf{0}, \\ & \mathbf{h}(\pi) \leq \beta. \end{aligned} \tag{9}$$

The first two constraints restrict π to valid probabilities:

$$\mathcal{J} = \{\pi \,|\, \mathbf{J}\pi = \mathbf{1} \text{ and } \pi \geq \mathbf{0}\}.$$

The last inequalities reflect budget constraint satisfaction:

$$\mathcal{H} = \{\pi \,|\, \mathbf{h}(\pi) \leq \beta\}. \tag{10}$$

We can then express the above problem compactly as

$$\underset{\pi \in \mathcal{C}}{\text{maximize}} \quad f(\pi) \tag{11}$$

where $\mathcal{C} = \mathcal{H} \cap \mathcal{J}$.

We solve the formulation given in Equation 11 via Projected Gradient Ascent (PGA) with line search. Let $\mathbf{Z} = \mathbf{I} - \gamma \mathbf{M}^\pi$ for notational convenience. The gradient $\nabla f(\pi)$ is defined by the partial derivatives [13]:

$$\frac{\partial f(\pi)}{\partial \pi_i} = \delta_0^\intercal \left(\mathbf{Z}^{-1} \left(\frac{\partial \mathbf{r}^\pi}{\partial \pi_i} + \frac{\partial \mathbf{Z}}{\partial \pi_i} \mathbf{Z}^{-1} \mathbf{r}^\pi \right) \right) \tag{12}$$

for each parameter π_i. The *gradient step* at iteration k is:

$$\xi^{(k)} = \pi^{(k)} + \alpha^{(k)} \nabla f(\pi^{(k)})$$

where $\alpha^{(k)}$ is determined via golden section line search, a novel approach for CPOMDPs, based on its success in POMDPs [16] (Algorithm 2). This intermediate update is not necessarily feasible. The *projection step*, $P_\mathcal{C}(\xi)$, projects ξ onto the set $\mathcal{C}$. Thus the iteration over k is:

$$\pi^{(k+1)} = P_\mathcal{C}\left(\pi^{(k)} + \alpha^{(k)} \nabla f(\pi^{(k)}) \right). \tag{13}$$

The set $\mathcal{C}$ is generally non-convex, making the computation of the projection $P_\mathcal{C}$ challenging to perform.

A. Policy Evaluation and Constraints

For a given policy π, evaluation of the objective function $f(\pi)$ and all constraints $h_i(\pi)$ requires $m+1$ policy evaluations following Equation 7 and 8. In all equations, the inversion $\mathbf{Z}^{-1} = (\mathbf{I} - \gamma \mathbf{M}^\pi)^{-1}$ is by far the most expensive operation. Importantly, however, it is independent of the specific value ($\mathbf{v}_r$ or $\mathbf{v}_i$) being computed. As a result, for any policy π, the LU factorization of $\mathbf{Z}$ can be computed once and reused for all m remaining policy evaluations. The subsequent policy evaluations can then be performed on the order of a matrix vector multiplication. Recognizing this feature and employing it provides significant performance improvements in practice.

Algorithm 1 CPOMDP projected gradient ascent (PGA).

Require: $|X|$: The number of nodes.
Require: ϵ: The convergence criterion.
$\xi^{(0)}$, $\mathbf{v}^{(0)} \leftarrow$ RAND($|X|$), $\mathbf{0}$
$\pi^{(1)} \leftarrow$ PROJECT($\xi^{(0)}$) ▷ Eq. 14 or 16
$k \leftarrow 1$
do
  $\nabla f(\pi^{(k)}) \leftarrow$ COMPUTEGRADIENT($\pi^{(k)}$) ▷ Eq. 12
  $\xi^{(k)}, \mathbf{v}^{(k)} \leftarrow$ LINESEARCH($\pi^{(k)}$, $\nabla f(\pi^{(k)})$) ▷ Alg. 2
  $\mathbf{Z}^{-1} \leftarrow$ COMPUTEINVERSE($\xi^{(k)}$)
  if $\delta_0^\intercal \mathbf{v}^{(k)} \geq \delta_0^\intercal \mathbf{v}^{(k)}$ **and** $\delta_0^\intercal \mathbf{Z}^{-1} \mathbf{c}_i^{(k)} \geq \delta_0^\intercal \mathbf{Z}^{-1} \mathbf{c}_i^{(k)}$ $\forall i$ **then**
    $\pi^{(k+1)} \leftarrow \xi^{(k)}$
  else
    $\pi^{(k+1)} \leftarrow$ PROJECT($\xi^{(k)}$) ▷ Eq. 14 or 16
    $\mathbf{Z}^{-1} \leftarrow$ COMPUTEINVERSE($\pi^{(k+1)}$)
    $\mathbf{v}^{(k+1)} \leftarrow \mathbf{Z}^{-1} \mathbf{r}^{(k+1)}$
  $k \leftarrow k+1$
while RELATIVEERROR($\mathbf{v}^{(k)}$, $\mathbf{v}^{(k-1)}$) $> \epsilon$
return $\pi^{(k)}, f(\pi^{(k)})$

Algorithm 2 LINESEARCH: Custom golden section search.

Require: π: The current policy.
Require: $\nabla f(\pi)$: The gradient line to search along.
Require: ϵ: The convergence criterion.
$\alpha_1, \alpha_2, \alpha_3, \alpha_4 \leftarrow 0, 1 - \frac{1}{\phi}, \frac{1}{\phi}, 1$
π_2, $\pi_3 \leftarrow \pi + \alpha_2 \nabla f(\pi)$, $\pi + \alpha_3 \nabla f(\pi)$
$\mathbf{Z}_2^{-1} \leftarrow$ COMPUTEINVERSE(π_2)
$\mathbf{Z}_3^{-1} \leftarrow$ COMPUTEINVERSE(π_3)
while $|\alpha_4 - \alpha_1| < \epsilon(|\alpha_2| + |\alpha_3|)$ **do**
  $\mathbf{v}^{\pi_2}$, $\mathbf{v}^{\pi_3} \leftarrow \mathbf{Z}_2^{-1} \mathbf{r}^{\pi_2}$, $\mathbf{Z}_3^{-1} \mathbf{r}^{\pi_3}$
  if $\delta_0^\intercal \mathbf{v}^{\pi_3} \geq \delta_0^\intercal \mathbf{v}^{\pi_2}$ **then** ▷ Option $\delta_0^\intercal \mathbf{Z}_3^{-1} \mathbf{c}_i^{\pi_3} \geq \delta_0^\intercal \mathbf{Z}_3^{-1} \mathbf{c}_i^{\pi_2}$
    $\alpha_1, \alpha_2, \mathbf{Z}_2^{-1}, \alpha_3 \leftarrow \alpha_2, \alpha_3, \mathbf{Z}_3^{-1}, \alpha_2 + \frac{1}{\phi}(\alpha_4 - \alpha_2)$
    π_3, $\pi_2 \leftarrow \pi + \alpha_3 \nabla f(\pi)$, π_3
    $\mathbf{Z}_3^{-1} \leftarrow$ COMPUTEINVERSE(π_3)
  else
    $\alpha_4, \alpha_3, \mathbf{Z}_3^{-1}, \alpha_2 \leftarrow \alpha_3, \alpha_2, \mathbf{Z}_2^{-1}, \alpha_3 - \frac{1}{\phi}(\alpha_3 - \alpha_1)$
    π_2, $\pi_3 \leftarrow \pi + \alpha_2 \nabla f(\pi)$, π_2
    $\mathbf{Z}_2^{-1} \leftarrow$ COMPUTEINVERSE(π_2)
return $\frac{1}{2}(\pi_2 + \pi_3)$, $\frac{1}{2}(\mathbf{v}^{\pi_2} + \mathbf{v}^{\pi_3})$

B. Optimal Constraint Projections

The one-step projection $P_\mathcal{C}$ in Equation 13 requires solving

$$\begin{aligned} \underset{\pi}{\text{minimize}} \quad & \frac{1}{2} \left\| \pi^{(k)} + \alpha_k \nabla f(\pi^{(k)}) - \pi \right\|_2^2 \\ \text{subject to} \quad & \mathbf{J}\pi = \mathbf{1}, \\ & \pi \geq \mathbf{0}, \\ & \mathbf{h}(\pi) \leq \beta. \end{aligned} \tag{14}$$

This has a convex quadratic objective function but contains non-convex inequality constraints. As a result, the projection is expensive to perform; further, it needs to be computed each iteration. To make the projection more feasible, an approximation is introduced in the next section.

C. Approximate Constraint Projections

The primary difficulty in solving Equation 14 is related to non-convexity of the functions $h_i(\pi)$. Thus, we approximate each $h_i(\pi)$ by its linearization around the current policy iterate $\pi^{(k)}$ via its truncated Taylor expansion:

$$h_i(\pi) \approx h_i(\pi^{(k)}) + \nabla h_i(\pi^{(k)})^{\mathrm{T}}(\pi - \pi^{(k)}). \tag{15}$$

To apply this approximation, let the Jacobian $\nabla \mathbf{h}(\pi^{(k)})$ be:

$$\nabla \mathbf{h}(\pi^{(k)}) = \begin{bmatrix} \nabla h_1(\pi^{(k)})^{\mathrm{T}} \\ \vdots \\ \nabla h_m(\pi^{(k)})^{\mathrm{T}} \end{bmatrix}$$

and let $\mathbf{b} \in \mathbb{R}^{n+m}$ be:

$$\mathbf{b} = \begin{bmatrix} \mathbf{0}_n \\ \beta - \mathbf{h}(\pi^{(k)}) + \nabla \mathbf{h}(\pi^{(k)}) \pi^{(k)} \end{bmatrix}$$

where $\mathbf{0}_n \in \mathbb{R}^n$ denotes the vector with n zeros and $\mathbf{A}$ be:

$$\mathbf{A} = \begin{bmatrix} -\mathbf{I}_n \\ \nabla \mathbf{h}(\pi^{(k)}) \end{bmatrix}$$

where $\mathbf{I}_n \in \mathbb{R}^{n \times n}$ is the identity matrix. The optimization problem in Equation 14 then becomes:

$$\begin{aligned} \underset{\pi}{\text{minimize}} \quad & \frac{1}{2} \left\| \pi^{(k)} + \alpha^{(k)} \nabla f(\pi^{(k)}) - \pi \right\|_2^2 \\ \text{subject to} \quad & \mathbf{J}\pi = \mathbf{1}, \\ & \mathbf{A}\pi \leq \mathbf{b}. \end{aligned} \tag{16}$$

This optimization problem is convex and, more specifically, is a linearly-constrained quadratic program. This approximation is much easier to solve than Equation 14 and can be done using off-the-shelf solvers. As a result, we use this formulation for the projection $P_{\mathcal{C}}$.

In the context of gradient ascent, this can be viewed as defining an approximation of the set $\mathcal{C} = \mathcal{H} \cap \mathcal{J}$ at each iterate by linearizing the constraints in $\mathcal{H}$ defined in Equation (10). This results in the update at iteration k:

$$\begin{aligned} \mathcal{H}^{(k)} &\leftarrow \left\{ \pi \,|\, \mathbf{h}(\pi^{(k)}) + \nabla \mathbf{h}(\pi^{(k)})(\pi - \pi^{(k)}) \leq \beta \right\}, \\ \mathcal{C}^{(k)} &\leftarrow \mathcal{H}^{(k)} \cap \mathcal{J}, \\ \pi^{(k+1)} &\leftarrow P_{\mathcal{C}^{(k)}} \left(\pi^{(k)} + \alpha^{(k)} \nabla f(\pi^{(k)}) \right) \end{aligned} \tag{17}$$

which replaces the iteration defined in Equation 13. In summary, we linearize each $h_i(\cdot)$ around the current policy $\pi^{(k)}$ to construct an approximation of the constraint near $\pi^{(k)}$ which we denote by $\mathcal{C}^{(k)}$. We then project onto $\mathcal{C}^{(k)}$ by solving Equation 16.

As this is an approximation, it is of course possible that linearization results in inconsistent constraints. In this case, the approximate problem (Equation 16) would be ill-posed. In order to ensure this is avoided, we can evaluate feasibility of the linearzation during line search. If the proposed step results in an infeasible approximate sub-problem, we reduce the proposed step-size until a feasible approximate projection is possible. If not found, we can perform the more expensive projection (Equation 13) to obtain the next iterate.

V. Theoretical Analysis

In this section we will establish relevant theoretical details of the presented algorithm. In all of the subsequent propositions we assume the existence of feasible points for all optimization formulations considered. We first show the existence of solutions for the primary optimization formulation.

Proposition 1: Let U^* denote the optimal set for Equation 9. Then U^* is non-empty.

Proof: We first show that $\mathcal{C} = \mathcal{H} \cap \mathcal{J}$ is compact. The pre-image of each inequality constraint is given by

$$h_i^{-1}\left((-\infty, c_i]\right) = \{\pi \,|\, h_i(\pi) \in (-\infty, c_i]\}$$

for $i \in \{1, \ldots, m\}$. Because $(-\infty, c_i]$ is closed and each $h_i(\cdot)$ is continuous we have $h_i^{-1}\left((-\infty, c_i]\right)$ is closed. Therefore,

$$\mathcal{H} = \bigcap_{i=1}^{m} h_i^{-1}\left((-\infty, c_i]\right)$$

is closed. The simplex $\mathcal{J}$ is a closed bounded subset of $\mathbb{R}^n$ which means it is compact. Then $\mathcal{C} = \mathcal{H} \cap \mathcal{J}$ is the intersection of a closed and compact set which implies $\mathcal{C}$ is compact. Since $f(\pi)$ is continuous it follows that the optimal value f^* is finite and the optimal set U^* is non-empty. ■

With existence of a maximizer established, we next present some properties of the projection sub-problems. This will also serve to motivate our approximation.

Proposition 2: Assume that $\mathcal{H}$ defined in Equation 10 is convex. Then the projection $P_{\mathcal{C}} : \mathbb{R}^n \to \mathcal{C}$ is unique.

Proof: The objective function in Equation 14 is strictly convex. If $\mathcal{H}$ convex, it follows that the set $\mathcal{C} = \mathcal{H} \cap \mathcal{J}$ is compact and convex. Then Equation 14 is the minimization of a strictly convex function over a compact convex set. The result follows from standard convex theory. ■

The following corollary results from $\mathcal{H}^{(k)}$ being a convex.

Corollary 1: Each projection $P_{\mathcal{C}^{(k)}} : \mathbb{R}^n \to \mathcal{C}^{(k)}$ is unique.

Because $\mathcal{H}$ is generally not convex, Corollary 1 highlights the approximation's usefulness in addition to its reduction in computational cost.

Next we show that under mild regularity assumptions on the objective function $f : \mathcal{C}^{(k)} \to \mathbb{R}$, our iteration produces policies which increase the expected reward $f(\pi)$ while respecting the constraint approximation $\mathcal{C}^{(k)}$. Following this, we will show that for a convergent sequence of policies the limit is feasible with respect to the original constraint set.

Proposition 3: Assume $f : \mathcal{C}^{(k)} \to \mathbb{R}$ is L-smooth and let $\pi^{(k)}$ be defined through the iteration from Equation 17. Then

$$F(\pi^{(k+1)}) - F(\pi^{(k)}) \geq \left(\frac{1}{\alpha^{(k)}} - \frac{L}{2} \right) \left\| \pi^{(k+1)} - \pi^{(k)} \right\|_2^2$$

where $F(\pi) = f(\pi) - g^{(k)}(\pi)$ with $g^{(k)}(\pi)$ the indicator function on $\mathcal{C}^{(k)}$ and $\alpha^{(k)} \in (0, \frac{2}{L})$.

Proof: The update defined in Equation 17 can be written as the proximal operator associated with the indicator function $g^{(k)}(\pi)$. We have

$$\begin{aligned} \pi^{(k+1)} &= P_{\mathcal{C}^{(k)}} \left(\pi^{(k)} + \alpha^{(k)} \nabla f(\pi^{(k)}) \right) \\ &= \underset{\pi}{\operatorname{argmin}} \left\{ \alpha^{(k)} g^{(k)}(\pi) + \frac{1}{2} \left\| \pi - \left(\pi^{(k)} + \alpha^{(k)} \nabla f(\pi^{(k)}) \right) \right\|_2^2 \right\} \end{aligned}$$

which by definition of the proximal operator yields

$$\pi^{(k+1)} = \underset{\alpha^{(k)} g^{(k)}}{\text{prox}} \left(\pi^{(k)} + \alpha^{(k)} \nabla f(\pi^{(k)})\right).$$

Therefore, $\pi^{(k+1)}$ satisfies the differential inclusion

$$0 \in \partial\left(\alpha^{(k)} g^{(k)}(\pi^{(k+1)})\right) + \pi^{(k+1)} - \left(\pi^{(k)} + \alpha^{(k)} \nabla f(\pi^{(k)})\right)$$

and rearranging shows that the difference is a subgradient,

$$\left(\pi^{(k)} + \alpha^{(k)} \nabla f(\pi^{(k)})\right) - \pi^{(k+1)} \in \partial\left(\alpha^{(k)} g^{(k)}(\pi^{(k+1)})\right).$$

From the definition of subgradient, we have

$$\begin{aligned}\langle \pi^{(k)} + \alpha^{(k)} \nabla f(\pi^{(k)}) - \pi^{(k+1)}, \pi^{(k)} - \pi^{(k+1)} \rangle \\ \leq \alpha^{(k)} \left(g^{(k)}(\pi^{(k)}) - g^{(k)}(\pi^{(k+1)})\right).\end{aligned}$$

Multiplying through by a negative and rearranging yields

$$\begin{aligned}&\langle \nabla f(\pi^{(k)}), \pi^{(k+1)} - \pi^{(k)} \rangle \\ &\geq \frac{1}{\alpha^{(k)}} \left\| \pi^{(k)} - \pi^{(k+1)} \right\|_2^2 + g^{(k)}(\pi^{(k+1)}) - g^{(k)}(\pi^{(k)}). \quad (18)\end{aligned}$$

Because f is L-smooth we

$$\begin{aligned}f(\pi^{(k+1)}) \geq f(\pi^{(k)}) + \nabla f(\pi^{(k)})^{\mathrm{T}} (\pi^{(k+1)} - \pi^{(k)}) \\ - \frac{L}{2} \left\| \pi^{(k+1)} - \pi^{(k)} \right\|_2^2.\end{aligned}$$

Subsituting in Inequality 18 we have

$$\begin{aligned}F(\pi^{(k+1)}) \geq F(\pi^{(k)}) + \frac{1}{\alpha^{(k)}} \left\| \pi^{(k)} - \pi^{(k+1)} \right\|_2^2 \\ - \frac{L}{2} \left\| \pi^{(k+1)} - \pi^{(k)} \right\|_2^2\end{aligned}$$

where $F(\pi) = f(\pi) - g^{(k)}(\pi)$, from which we obtain

$$F(\pi^{(k+1)}) - F(\pi^{(k)}) \geq \left(\frac{1}{\alpha^{(k)}} - \frac{L}{2}\right) \left\| \pi^{(k+1)} - \pi^{(k)} \right\|_2^2.$$

■

Proposition 4: Let $\{\pi^{(k)}\}$ be a sequence generated by Algorithm 1. Further, assume $\pi^{(k)} \to \pi^*$ and that each h_i in Equation 8 is smooth in a neighborhood of π^*. Then π^* is a feasible point of the optimization given in Equation 9.

Proof: Consider the Taylor series approximation of h_i around policy iterate $\pi^{(k)}$ evaluated at π^*. We have

$$\begin{aligned}h_i(\pi^*) = h_i(\pi^{(k)}) + \nabla h_i(\pi^{(k)})^{\mathrm{T}} (\pi^* - \pi^{(k)}) \\ + (\pi^* - \pi^{(k)})^{\mathrm{T}} h_i \nabla^2(\xi) (\pi^* - \pi^{(k)})\end{aligned}$$

for some ξ in between $\pi^{(k)}$ and π^*. The projection $P_{\mathcal{C}^{(k)}}$ then ensures

$$\begin{aligned}h_i(\pi^*) - (\pi^* - \pi^{(k)})^{\mathrm{T}} \nabla^2 h_i(\xi) (\pi^* - \pi^{(k)}) \\ = h_i(\pi^{(k)}) + \nabla h_i(\pi^{(k)})^T (\pi^* - \pi^{(k)}) \leq c_i.\end{aligned}$$

Taking the limit of both sides and using the smoothness of h_i we have

$$h_i(\pi^*) = \lim_{k \to \infty} \left(h_i(\pi^{(k)}) + \nabla h_i(\pi^{(k)})^{\mathrm{T}} (\pi^* - \pi^{(k)})\right) \leq c_i.$$

■

VI. Experiments

For direct comparison of controller-based algorithms, the proposed PGA algorithm, with its optimal and approximate projections is compared with an scalable optimal constrained NLP (CNLP) baseline on seven standard benchmark domains. PGA is demonstrated on a real robot implementing the home healthcare domain in the real world.

A. Experimental Setting

The standard metrics are used [10], [15]: average discounted reward/cost (ADR/ADC) and time to solve (in seconds). The proposed PGA algorithm was implemented in Julia 1.5.2. The constraint nonlinear program (CNLP) baseline algorithm solves the optimal formulation in Equation 4 with an off-the-shelf NLP solver [11]. It was solved using Ipopt in Julia's JuMP package. The experiments were done on an Intel Core i7-6700HQ CPU with 4 cores at 2.60GHz and 16GB of RAM. Results are averaged over 10 trials for each combination of algorithm and domain. Each combination of algorithm, domain, and controller size was allotted 2 hours to converge and return a solution.

B. Domains

The domains are constrained versions of standard POMDP benchmarks. The *toy* and *qcd* domains are from the original CPOMDP paper [1]. The *milos-aaai97*, *query.s3*, *query.s4*, and *tiger-grid* domains are CPOMDPs created from augmented POMDPs that include a cost constraint [15].

The *home-healthcare* domain is a POMDP domain [16] that is augmented to include a cost constraint. It has $S = S_r \times S_t$ denoting the regions of the robot and the target, respectively. A denotes neighboring region selection and Ω denotes observing the target or not. The goal is to navigate in this regional topological map to find the static target based on refining the belief about where it is. The reward is 1 for finding the target, 0 otherwise. The CPOMDP version includes a cost of 1 for traversing certain rooms. The robot experiments implement this domain in the real world.

C. Results and Discussion

Table I presents the results from the experiments. Overall, we observe that the proposed PGA algorithm using an approximate projection solves the problem much faster in comparison to the constrained NLP (CNLP) baseline. CNLP uses a generic off-the-shelf solver, whereas PGA is able to exploit the CPOMDP problem structure to more quickly solve the CPOMDP and compute similar final policies.

Increasing the controller size $|X|$ is shown to reliably improve the solution quality while increasing the time to solve. As the domains increase in size ($|S|$, $|A|$, and $|\Omega|$), and/or the number of controller nodes ($|X|$), other algorithms begin to fail to solve the problems at all. PGA is able to scale to solve these much larger CPOMDPs, enabling us to even solve a real robot planning problem.

Comparing CNLP with PGA Optimal, we see that the optimal projection inside of gradient ascent is far more limited. It is only able to solve the smallest of problems

Domain	$\|S\|$	$\|A\|$	$\|\Omega\|$	β	$\|X\|$	Constrained NLP Baseline [11]			PGA Optimal			PGA Approximate		
						ADR	ADC	T	ADR	ADC	T	ADR	ADC	T
toy	3	2	1	0.95	5	**0.08**	**0.08**	0.65	**0.08**	0.09	2.0	**0.07**	**0.07**	**0.21**
toy	3	2	1	0.95	10	**0.08**	**0.08**	0.39	**0.08**	0.09	48.7	**0.08**	**0.08**	**2.0**
qcd	3	2	3	0.2	5	**−0.04**	0.8	0.47	−0.31	**0.0**	2.3	−0.22	0.67	**0.29**
qcd	3	2	3	0.2	10	−0.33	**0.15**	2.3	−0.26	0.45	29.6	**−0.17**	0.91	**2.1**
milos-aaai97	20	6	8	10	5	**23.2**	**10.0**	740.5	—	—	—	12.7	**10.0**	**227.2**
milos-aaai97	20	6	8	10	10	**20.0**	**10.0**	3124.3	—	—	—	16.9	**10.0**	**563.6**
query.s3	27	3	3	53	5	261.2	14.6	**2.5**	—	—	—	**335.1**	**9.2**	64.3
query.s3	27	3	3	53	10	—	—	—	—	—	—	**344.1**	**8.7**	**166.0**
tiger-grid	36	5	17	18	5	**−0.25**	**19.4**	5684.8	—	—	—	−1.16	**19.8**	**287.1**
tiger-grid	36	5	17	18	10	—	—	—	—	—	—	**−0.76**	**19.9**	**734.9**
query.s4	81	4	3	60	5	—	—	—	—	—	—	**342.4**	**8.8**	**335.6**
query.s4	81	4	3	60	10	—	—	—	—	—	—	**339.4**	**8.9**	1607.7
home-healthcare	169	4	2	3	5	—	—	—	—	—	—	**7.7**	**6.5**	**580.6**
home-healthcare	169	4	2	3	10	—	—	—	—	—	—	**8.2**	**7.5**	**2777.8**

TABLE I

RESULTS FROM SIMULATION. ALGORITHMS: THE PROPOSED PROJECTED GRADIENT ASCENT (PGA) (OPTIMAL AND APPROXIMATE PROJECTIONS) AND THE CONSTRAINED NLP (CNLP) BASELINE [11]. DOMAINS: SEVEN BENCHMARKS, EACH VARYING NUMBER OF NODES ($|X|$). METRICS: AVERAGE DISCOUNTED REWARD/COST (ADR & ADC), AND TIME IN SECONDS (T).

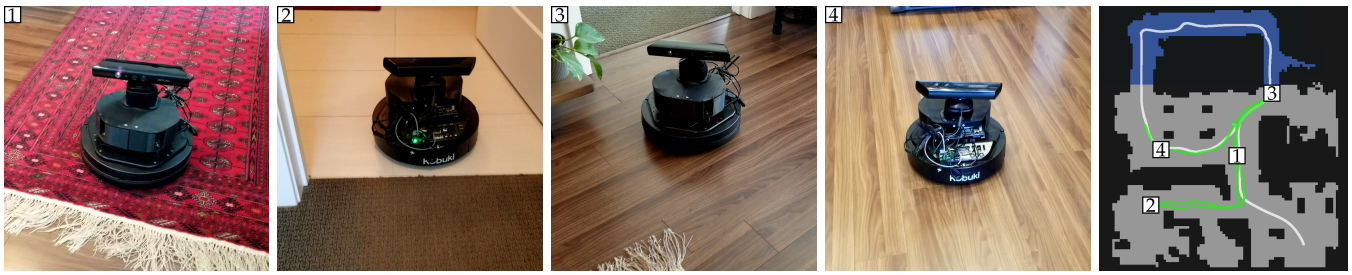

Fig. 1. Experiments that implement *home-healthcare* on a real robot in an actual household environment. The unconstrained POMDP's path (white) freely traverses the home [16]. The CPOMDP's path (green) is constrained to avoid northern rooms (blue), demonstrating PGA's successful constraint satisfaction.

(e.g., $|S|=3$) and at a high cost (up to 48 seconds). The CNLP, using a generic optimization formulation, is able to compute solutions for problems up to $|S|=36$ here, albeit with $|X|=5$ nodes, and at a very high time to compute of 5684 seconds, which is over 1.5 hours. The primary reason for PGA Optimal's inability to scale is the optimal projection continually iterated inside of gradient ascent.

Addressing this exact point, PGA Approximate significantly improves the speed of this iterated projection inside gradient ascent. Comparing PGA Approximate to CNLP, we see that it is able to scale to solve much larger problems. For example, CNLP took 5684 seconds to solve a domain of size $|S|=36$, $|A|=5$ and $|\Omega|=17$. PGA Approximate was able to solve this exact problem in 287 seconds. The trend continues, as the larger *home-healthcare* domain with $|S|=169$ states using $|X|=10$ nodes took only 2777 seconds. CNLP was unable to solve *home-healthcare* after 2+ hours of computation. Comparing values, such as in *milos-aaai97* and *query.s3*, we see that CNLP and PGA Approximate are able to both compute reasonable policies that are within the budget, in spite of the approximations. Depending on the problem structure, either CNLP's or PGA Approximate's projections, which are approximate, might compute a better valued policies. However, the overall trend in scalability and quality remains clear. The other controller-based CPOMDP algorithms are unable to solve these large problems, whereas PGA Approximate can solve them, and is able to retain good reward and cost values.

Figure 1 demonstrates the *home-healthcare* domain implemented on a real robot. PGA Approximate is the only controller-based algorithm able to solve this large CPOMDP domain; its policy was deployed in this home environment and shown in green in Figure 1. For comparison, we also solved the unconstrained POMDP to illustrate the effect of the constraint on PGA Approximate's computation. Unconstrained by the room traversal constraint, the robot moves in a circular loop around the home searching for its target goal (white). When constrained to avoid the northern rooms (blue), PGA Approximate is demonstrated to successfully produce constraint-satisfying search behavior (green).

VII. CONCLUSION

This paper presents two novel algorithms for controllers: projected gradient ascent (PGA)—with an optimal and approximate projection—and a variant nonlinear programming (NLP) method. PGA performs a customized golden section search along the policy gradient and projects the space of policies satisfying constraint objectives. The approximate projection converts the optimal nonlinear projection with a matrix inverse into an efficient quadratic program. Theoretical analysis of PGA show the uniqueness of the approximate projection and constraint satisfaction for convergent sequences of policies. The approach is experimentally validated in standard benchmarks as well as on a real robot, demonstrating PGA's success in increasing scalability towards the goal of improving real world autonomous robots.

REFERENCES

[1] J. D. Isom, S. P. Meyn, and R. D. Braatz, "Piecewise linear dynamic programming for constrained POMDPs," in *Proc. of the 23nd AAAI Conf. on Artificial Intelligence*, 2008, pp. 291–296.

[2] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1, pp. 99–134, 1998.

[3] H. Soh and Y. Demiris, "Evolving policies for multi-reward partially observable Markov decision processes (MR-POMDPs)," in *Proc. of the 13th Annual Conf. on Genetic and Evolutionary Computation*, 2011, pp. 713–720.

[4] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *Journal of Artificial Intelligence Research*, vol. 48, pp. 67–113, 2013.

[5] K. H. Wray and S. Zilberstein, "Multi-objective POMDPs with lexicographic reward preferences," in *Proc. of the 24th Int'l. Joint Conf. on Artificial Intelligence*, 2015, pp. 1719–1725.

[6] P. Santana, S. Thiébaux, and B. Williams, "RAO*: An algorithm for chance-constrained POMDPs," in *Proc. of the 30th AAAI Conf. on Artificial Intelligence*, 2016, pp. 3308–3314.

[7] K. H. Wray, S. J. Witwicki, and S. Zilberstein, "Online decision-making for scalable autonomous systems," in *Proc. of the 26th Int'l. Joint Conf. on Artificial Intelligence*, 2017, pp. 4768–4774.

[8] K. H. Wray, B. Lange, A. Jamgochian, S. J. Witwicki, A. Kobashi, S. Hagaribommanahalli, and D. Ilstrup, "POMDPs for safe visibility reasoning in autonomous vehicles," in *2021 IEEE Int'l. Conf. on Intelligence and Safety for Robotics*, 2021.

[9] E. A. Hansen, "Solving POMDPs by searching in policy space," in *Proc. of the 14th Conf. on Uncertainty in Artificial Intelligence*, 1998, pp. 211–219.

[10] J. Pineau, G. Gordon, and S. Thrun, "Anytime point-based approximations for large POMDPs," *Journal of Artificial Intelligence Research*, vol. 27, pp. 335–380, 2006.

[11] C. Amato, D. S. Bernstein, and S. Zilberstein, "Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs," *Autonomous Agents and Multi-Agent Systems*, vol. 21, no. 3, pp. 293–320, 2010.

[12] K. H. Wray, A. Kumar, and S. Zilberstein, "Integrated cooperation and competition in multi-agent decision-making," in *Proc. of the 32nd AAAI Conf. on Artificial Intelligence*, 2018, pp. 4751–4758.

[13] N. Meuleau, K.-E. Kim, L. P. Kaelbling, and A. R. Cassandra, "Solving POMDPs by searching the space of finite policies," in *Proc. of the 15th Conf. on Uncertainty in Artificial Intelligence*, 1999, p. 417–426.

[14] D. Kim, J. Lee, K.-E. Kim, and P. Poupart, "Point-based value iteration for constrained POMDPs," in *Proc. of the 22nd Int'l. Joint Conf. on Artificial Intelligence*, 2011, pp. 1968–1974.

[15] P. Poupart, A. Malhotra, P. Pei, K.-E. Kim, B. Goh, and M. Bowling, "Approximate linear programming for constrained partially observable Markov decision processes," in *Proc. of the 29th AAAI Conf. on Artificial Intelligence*, 2015, pp. 3342–3348.

[16] K. H. Wray and K. Czuprynski, "Scalable POMDP decision-making using circulant controllers," in *2021 IEEE International Conference on Robotics and Automation*, 2021.