Scalable POMDP Decision-Making Using Circulant Controllers

Kyle Hollins Wray and Kenneth Czuprynski

Abstract— **This paper presents a novel policy representation for partially observable Markov decision processes (POMDPs) called circulant controllers and a provably efficient gradient-based algorithm for them. A formal mathematical description is provided that leverages circulant matrices for the controller's stochastic node transitions. This structure is particularly effective for capturing decision-making patterns found in real-world domains with repeated periodic behaviors that adapt their cycles based on observation. This includes domains such as bipedal walking over varied terrain, pick-and-place tasks in warehouses, and home healthcare monitoring and medicine delivery in household environments. A performant gradient-based algorithm is presented with a detailed theoretical analysis, formally proving the algorithm's improved performance, as well as circulant controllers' structural properties. Experiments on these domains demonstrate that the proposed controller algorithm outperforms other state-of-the-art POMDP controller algorithms. The proposed novel controller approach is demonstrated on an actual robot performing a navigation task in a real household environment.**

I. Introduction

The partially observable Markov decision process (POMDP) is a general single agent decision-making model [1]. The model has benefited from an increasingly growing interest with the development of more scalable algorithms [2], [3], [4], [5], [6], [7]. Consequently POMDPs have been applied to many robotic domains, growing to include self-driving car decision-making [8] and aircraft collision avoidance systems [9]. However, POMDP policy representations and algorithms are still challenging to scale as POMDPs are still PSPACE-complete. One reason is that many algorithms are general-purpose, devised to solve any POMDP. They cannot readily benefit from known realistic domain-specific knowledge of the underlying problem structure which if leveraged could significantly improve performance. Towards this goal, we classify a common structure found in many robotic applications: cyclic patterns of actions that adapt from observation. We demonstrate that a class of POMDP finite state controllers, called circulant controllers, captures this structure and can be leveraged to improve convergence in gradient-based algorithms.

Cyclic behaviors are found in many robotic applications. In bipedal and quadrupedal walking robots, repeated cyclic patterns of actuation provide the major motion that drives the system forward [10], [11]. For each configuration of limbs, the cyclic patterns and overall gait is distinct, with an additional emphasis put on important foot placement decisions and balance adjustments augmenting the actuation cycle. In manufacturing and warehouse scenarios, pick and place robots must repeatedly pick up requested objects and collect or otherwise organize them [12], [13]. The major arm motions are cyclic, as each object is repeatedly picked up, moved to a specific location, and placed. In navigation and search domains, such as in home healthcare robots for eldercare monitoring and medicine delivery, cyclic patterns of home traversal arise when searching for moving patients [14], [15], [16]. In order to model partial observability using POMDPs in these kinds of domains, we should devise algorithms that explicitly leverage this cyclic behavioral property. These domains serve as inspiration for this paper's circulant controller formulation as it strives towards this goal.

POMDP algorithms are primarily categorized into belief point-based algorithms and finite state controller-based algorithms. Point-based algorithms can approximate by applying the update equation at chosen belief points [3], efficiently apply this update [17], and intelligently choose beliefs [18]. However, these methods stuffer from curses in both dimensionality and history, as the number of required beliefs grows exponentially in the states and horizon. Controller-based algorithms benefit from a memory-efficient policy form [19] that can be scalably expanded [20]. An optimal nonlinear programming (NLP) method for fixed-sized controllers [4] benefits from simplicity but suffers from poor performance using off-the-shelf NLP solvers. Early work on periodic finite state controllers [21] assigned a hand-coded layer structure, in the spirit of neural networks, but lacked the formalized circulant constraint, projection, and provable computational benefits presented here. Gradient-based optimization of controllers [2] benefits from a direct optimization formulation, amenable to customization. Its primary drawback is the inversion of a controller-related matrix.

This paper proposes circulant controllers, which ensure controller transitions are built from multiple circulant matrices. Circulant matrices enable a provably efficient inverse, enabling a highly efficient gradient computation. Line search for circulant controllers is also presented, leveraging the new efficient inverse in its policy evaluation steps. Circulant controllers capture the behavior found in many domains: periodic behavior that changes based on observation. By utilizing the domain structure, we can quickly compute highly performant policies.

Our contributions are: (1) a formal statement of the circulant controller (Section III); (2) a gradient-based algorithm that leverages circulant controllers in policy evaluation and line search (Section IV); (3) a detailed theoretical analysis of circulant controllers and the gradient-based algorithm (Section V); and (4) experiments in three prevalent robotic domains including a fully implemented circulant controller for a navigation and search task on a real robot acting in a real household environment (Section VI).

II. Background

A partially observable Markov decision process (POMDP) [22] is defined by the tuple $\langle S, A, \Omega, T, O, R\rangle$. S is a finite set of states. A is a finite set of actions. Ω is a finite set of observations. $T: S \times A \times S \to [0,1]$ is a state transition such that $T(s,a,s') = Pr(s' \mid s,a)$ denotes the probability of transitioning to state s' given action a was performed in state s. $O: S \times \Omega \to [0,1]$ is an observation function such that $O(s',\omega) = Pr(\omega \mid s')$ denotes the probability of observing ω after transitioning to state s'. $R: S \times A \to \mathbb{R}$ is a reward function such that $R(s,a)$ denotes the reward for performing action a in state s.

In POMDPs, the true state of the system in not observed directly. The agent represents uncertainty over the state by beliefs, denoted as $b \in B \subseteq \triangle^{|S|}$ for any belief set B from the standard n-1 simplex $\triangle^{|S|}$. An initial belief b^0 is known.

There are two primary ways to maintain an internal state and choose actions: *belief point-based policies* [3] or *finite state controller policies* [19].

A. Belief Point-Based Policies

In belief point-based policies, the agent explicitly maintains a belief b, updating it after performing action a and observing ω. The resulting successor belief b' updated via

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

A belief point-based policy $\pi: B \to A$ maps each belief to an action. The objective is to maximize expected reward given an initial belief $b^0 \in B$ and a discount factor $\gamma \in [0,1)$. The value function is $V^\pi: B \to \mathbb{R}$ is this expected value at each belief. It can be computed by the Bellman optimality equation

$$V^*(b) = \max_{a \in A} \Big[\sum_{s \in S} b(s) R(s,a) + \gamma \sum_{\omega \in \Omega} Pr(\omega|b,a) V^*(b') \Big].$$

B. Finite State Controller Policies

A finite state controller's policy is defined as $\pi = \langle X, \psi, \eta \rangle$. X is a finite set of controller nodes. $\psi: X \times A \to [0,1]$ is a stochastic action selection function such that $\psi(x,a) = Pr(a \mid x)$ denotes the probability of selecting action a in node x. $\eta: X \times \Omega \times X \to [0,1]$ is a stochastic successor selection function such that $\eta(x,\omega,x') = Pr(x' \mid x,\omega)$ denotes the probability of choosing successor node x' given observation ω was made after performing node x's action.

Controller policies have a value function derived from the cross-product MDP formed by the controller node transitions and the underlying MDP [19]. Formally, $V^\pi: X \times S \to \mathbb{R}$ is

$$V^\pi(x,s) = \sum_{a \in A} \psi(x,a) \Big[R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \sum_{\omega \in \Omega} O(s',\omega) \sum_{x' \in X} \eta(x,\omega,x') V^\pi(x',s') \Big]. \quad (1)$$

Solving this system of linear equations is called *policy evaluation*. Given an initial controller node $x^0 \in X$ and initial belief $b^0 \in B$, the initial controller's value is $V^\pi(x^0,b^0) = \sum_{s \in S} b(s) V^\pi(x^0,s)$.

C. Controller Policy Gradient Ascent

To compute the optimal controller policy, we can derive a policy gradient and perform gradient ascent [2]. First, let $\mathbf{M}^\pi \in \mathbb{R}^{|X \times S| \times |X \times S|}$ denote the cross-product MDP state transition, defined by:

$$\mathbf{M}^\pi(\langle x,s\rangle, \langle x',s'\rangle) = \sum_{a \in A} \psi(x,a) T(s,a,s') \sum_{\omega \in \Omega} O(s',\omega) \eta(x,\omega,x'). \quad (2)$$

Importantly, all vectors and matrices in $X \times S$ are in *state-major ordering*. Let $\mathbf{r}^\pi \in \mathbb{R}^{|X \times S|}$ with $\mathbf{r}^\pi(\langle x,s\rangle) = \sum_a \psi(x,a) R(s,a)$. Now Equation 1 can be rewritten as a value vector $\mathbf{v}^\pi \in \mathbb{R}^{|X \times S|}$ for the policy π:

$$\mathbf{v}^\pi = \mathbf{r}^\pi + \gamma \mathbf{M}^\pi \mathbf{v}^\pi \quad (3)$$

resulting in

$$\mathbf{v}^\pi = (\mathbf{I} - \gamma \mathbf{M}^\pi)^{-1} \mathbf{r}^\pi. \quad (4)$$

Let $\mathbf{Z} = \mathbf{I} - \gamma \mathbf{M}^\pi$ for notational convenience. The partial derivative of $\mathbf{v}^\pi$ with respect to a policy parameter π:

$$\frac{\partial \mathbf{v}^\pi}{\partial \pi} = \mathbf{Z}^{-1} \Big(\frac{\partial \mathbf{r}^\pi}{\partial \pi} + \frac{\partial \mathbf{Z}}{\partial \pi} \mathbf{Z}^{-1} \mathbf{r}^\pi \Big). \quad (5)$$

Also for notational convenience, π refers to any controller policy parameter $\psi(x,a)$ or $\eta(x,\omega,x')$.

Gradient ascent strives to maximize the value of the initial node and belief vector $\beta^0 \in \mathbb{R}^{|X \times S|}$ with $\beta^0(\langle x,s\rangle) = b^0(s)[x = x^0]$ using Iverson bracket $[\cdot]$. With Equation 5, gradient ascent updates the policy at iteration k via:

$$\psi^{(k+1)}(x,a) = \psi^{(k)}(x,a) + \alpha \beta^0 \cdot \frac{\partial \mathbf{v}^{\pi^{(k)}}}{\partial \psi^{(k)}(x,a)}$$
$$\eta^{(k+1)}(x,\omega,x') = \eta^{(k)}(x,\omega,x') + \alpha \beta^0 \cdot \frac{\partial \mathbf{v}^{\pi^{(k)}}}{\partial \eta^{(k)}(x,\omega,x')} \quad (6)$$

with a step size of $\alpha > 0$. There are two key aspects of gradient ascent that limit its performance: (1) computing $\mathbf{Z}^{-1}$ and (2) determining α. The remainder of this paper defines and analyzes a novel policy structure we call circulant controllers, that provably addresses these aspects.

III. Circulant Controllers

This section formally introduces the circulant controller and how general controllers can be projected onto this space.

A. Circulant Matrices

Circulant controllers, as defined herein, are closely related to the notion of a circulant matrix. As a result, a brief overview of the necessary properties is given followed by the formal definition of a circulant controller.

Any circulant matrix is uniquely defined by it first row. For any vector $\mathbf{c} \in \mathbb{R}^n$ we have an associated circulant matrix $\mathbf{C} = circ(\mathbf{c})$ defined by applying the circular shift operator $\mathbf{Q}$ to each successive row, resulting in matrices of the form

$$\mathbf{C} = \begin{pmatrix} c_1 & c_2 & \dots & c_n \\ c_n & c_1 & \dots & c_{n-1} \\ \vdots & & \ddots & \vdots \\ c_2 & c_3 & \dots & c_1 \end{pmatrix}.$$

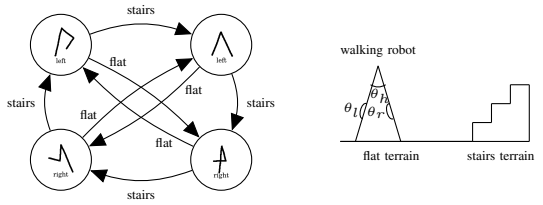

Fig. 1. Abstracted example: circulant controller (left) for walking domain (right) (Section VI). Actions are shown in nodes. Repeating actions denotes careful shifting weight between legs. Edges branch on observed terrain.

These matrices have several well known properties which are stated here for convenience. Let $\mathcal{C}_n$ be the space of real valued $n \times n$ circulant matrices.

Proposition 1: The space of circulant matrices $\mathcal{C}_n$ is isomorphic to $\mathbb{R}^n$ and is closed under addition & multiplication.

Corollary 1: The space of circulant matrices $\mathcal{C}_n$ is convex.

By definition every circulant matrix is uniquely defined by its first row and, therefore, a bijection is readily available. Closure under addition and scalar multiplication is also easily verified. Next, one of the most celebrated properties of circulant matrices is related to its eigenvalue decomposition and is given in the following proposition.

Proposition 2: Any circulant matrix $\mathbf{C} = circ(\mathbf{c})$ has eigenvalue decomposition $\mathbf{C} = \mathbf{F}\mathbf{\Lambda}\mathbf{F}^{-1}$ where $\mathbf{F}$ is the Fourier matrix and $\mathbf{\Lambda} = diag(\mathbf{F}\mathbf{c})$.

That is, circulant matrices are diagonalizable via the Fourier matrix with eigenvalues given by the Fourier transform of the first row. When coupled with the Fast Fourier Transform (FFT), significant computational gains can be achieved. Matrix-vector multiplications can be done using the FFT, reducing computations from $O(n^2)$ to $O(n \log n)$.

B. The Space of Circulant Controllers

Equipped with the definition of a circulant matrix, the formal definition of a circulant controller is as follows.

Definition 1: A controller $\langle X, \psi, \eta \rangle$ is circulant if the stochastic successor selection mapping $\eta(\cdot, \omega, \cdot)$ is circulant for every fixed $\omega \in \Omega$.

This results in structurally constrained policy representations (Figure 1) that inherit the rich properties of circulant matrices. It also propagates into the general optimization problem which will work to promote computational structure in our gradient updates. We can exploit this for significant computation gains as detailed in the following sections.

C. Circulant Projection

Applying gradient ascent in policy space will result in updates from $\eta^{(k)}$ to $\eta^{(k+1)}$ following Equation 6. However, the space of circulant controllers is not closed under this mapping. As a result, it will be necessary to project each update onto the convex set of circulant matrices.

An efficient orthogonal projection onto this space can be obtained by noting that the circular shift operator $\mathbf{Q}$ can be used to construct a basis for the space of circulant matrices. Viewing an arbitrary $n \times n$ matrix as an element of $\mathbb{R}^{n^2}$, the projection reduces to the standard projection onto a subspace of $\mathbb{R}^{n^2}$ for which we have a basis. The details of this construction are given in the proof of Proposition 4.

IV. Circulant Gradient Ascent with Line Search

The algorithm can be broken into three primary operations: policy evaluation, projection onto the set of constraints (circulant and simplex), and line search along the gradient.

A. Policy Evaluation

This subsection details how fast policy evaluation can be achieved by utilizing the computational structure induced by a circulant controller.

Policy evaluation is given by (4). Without any structure, the computation of $(\mathbf{I} - \gamma \mathbf{M}^\pi)^{-1}$ is $O(|X \times S|^3)$, which rapidly becomes prohibitive for large problems. However, when restricted to the space of circulant controllers, the matrix $\mathbf{M}^\pi$ can be factored into the product of a block diagonal matrix $\mathbf{D}$ and a block circulant matrix $\mathbf{C}$ (cf. Proposition 3 in Section V.). Concretely, we have

$$\mathbf{D} = BlockDiag\left(\mathbf{D}_1, \ldots, \mathbf{D}_{|X|}\right)$$

where each $\mathbf{D}_k \in \mathbb{R}^{|S| \times |S|}$ and

$$\mathbf{C} = BlockCirc\left(\mathbf{C}_1, \ldots, \mathbf{C}_{|X|}\right)$$

where each $\mathbf{C}_k \in \mathbb{R}^{|S| \times |S|}$ is diagonal. Importantly, although $\mathbf{M}^\pi$ is structured, the matrix $\mathbf{Z} = (\mathbf{I} - \gamma \mathbf{M}^\pi)$ does not share this structure. As a result, in order to exploit the factorization of $\mathbf{M}^\pi$ in the computation of $\mathbf{v}^\pi = (\mathbf{I} - \gamma \mathbf{M}^\pi)^{-1} \mathbf{r}^\pi$, the system is solved iteratively. We do this by proposing a matrix splitting from which we obtain an efficient stable iteration.

We begin by noting that the matrix $\mathbf{M}^\pi$ can be expressed in terms of the above block matrices as

$$\mathbf{M}^\pi = \begin{pmatrix} \mathbf{D}_1\mathbf{C}_1 & \mathbf{D}_1\mathbf{C}_2 & \ldots & \mathbf{D}_1\mathbf{C}_{|X|} \\ \mathbf{D}_2\mathbf{C}_{|X|} & \mathbf{D}_2\mathbf{C}_1 & \ldots & \mathbf{D}_2\mathbf{C}_{|X|-1} \\ \vdots & & \ddots & \vdots \\ \mathbf{D}_{|X|}\mathbf{C}_2 & \mathbf{D}_{|X|}\mathbf{C}_3 & \ldots & \mathbf{D}_{|X|}\mathbf{C}_1 \end{pmatrix}. \quad (7)$$

Then letting $\mathbf{M}_D^\pi$ denote the blocks along the diagonal in Equation 7 and letting $\mathbf{C}^0 = BlockCirc\left(\mathbf{0}, \mathbf{C}_2, \ldots, \mathbf{C}_{|X|}\right)$:

$$\mathbf{M}^\pi = \mathbf{M}_D^\pi + \mathbf{D}\mathbf{C}^0. \quad (8)$$

Substituting Equation 8 into Equation 4 and rearranging

$$\begin{aligned} \left[\mathbf{I} - \gamma\left(\mathbf{M}_D^\pi + \mathbf{D}\mathbf{C}^0\right)\right]\mathbf{v}^\pi &= \mathbf{r}^\pi \\ \left(\mathbf{I} - \gamma\mathbf{M}_D^\pi\right)\mathbf{v}^\pi &= \mathbf{r}^\pi + \gamma\mathbf{D}\mathbf{C}^0\mathbf{v}^\pi. \end{aligned}$$

Now we can invert the block diagonal and define the iteration

$$\mathbf{v}_{k+1}^\pi = \left(\mathbf{I} - \gamma\mathbf{M}_D^\pi\right)^{-1}\left[\mathbf{r}^\pi + \gamma\mathbf{D}\mathbf{C}^0\mathbf{v}_k^\pi\right], \quad k = 1, 2, \ldots \quad (9)$$

By using the proposed splitting, the iteration defined in Equation 9 maintains efficient block structure in all updates. The block diagonal structure of $(\mathbf{I} - \gamma\mathbf{M}_D^\pi)$ allows the inverse to be computed efficiently and parallelized simply. Further, because $\mathbf{M}^\pi$ is a stochastic matrix, each block is invertible. Using the structure in both $\mathbf{D}$ and $\mathbf{C}^0$, the multiplications can be done quickly using FFTs and exploiting the parallel block structure. A detailed complexity analysis is given in Proposition 5 in Section V. PolicyEvaluation performs the iteration in circulant space from (9).

Algorithm 1 CGA+LS: Circulant gradient ascent line search.

Require: ℓ: The number of nodes.
Require: ϵ: The convergence criterion.
Require: P: The circulant projection matrix.
$\pi^{(0)} \leftarrow \langle X{=}\{1,\dots,\ell\}, \psi^{(0)}{=}\textsc{Rand}(\,), \eta^{(0)}{=}\textsc{Rand}(\,)\rangle$
$\mathbf{v}^{(0)} \leftarrow \textsc{PolicyEvaluation}(\pi^{(0)})$
$k \leftarrow 0$
do
  $\frac{\partial \mathbf{v}^{\pi^{(k)}}}{\partial \pi^{(k)}} \leftarrow \textsc{ComputeGradient}(\pi^{(k)})$
  $\pi^{(k+1)}, \mathbf{v}^{(k+1)} \leftarrow \textsc{LineSearch}(\pi^{(k)}, \mathbf{v}^{(k)}, \frac{\partial \mathbf{v}^{\pi^{(k)}}}{\partial \pi^{(k)}})$
  $\pi^{(k+1)} \leftarrow \textsc{ProjectToSimplex}(\pi^{(k+1)})$
  $\eta^{(k+1)} \leftarrow \textsc{ProjectToCirculant}(P, \eta^{(k+1)})$
  $k \leftarrow k+1$
while $\textsc{RelativeError}(\mathbf{v}^{\pi^{(k)}}, \mathbf{v}^{\pi^{(k-1)}}) > \epsilon$
return $\textsc{ProjectToSimplex}(\pi^{(k)})$

Algorithm 2 LINESEARCH: Custom golden section search.

Require: π: The current policy.
Require: $\frac{\partial \mathbf{v}^{\pi}}{\partial \pi}$: The gradient line to search along.
$\alpha_1, \alpha_2, \alpha_3, \alpha_4 \leftarrow 0, 1{-}\frac{1}{\phi}, \frac{1}{\phi}, 1$
$\mathbf{v}^{\alpha_2} \leftarrow \textsc{PolicyEvaluation}(\pi + \alpha_2 \frac{\partial \mathbf{v}^{\pi}}{\partial \pi})$
$\mathbf{v}^{\alpha_3} \leftarrow \textsc{PolicyEvaluation}(\pi + \alpha_3 \frac{\partial \mathbf{v}^{\pi}}{\partial \pi})$
while $|\alpha_4 - \alpha_1| < \epsilon(|\alpha_2| + |\alpha_3|)$ **do**
  if $\beta^0 \mathbf{v}^{\alpha_3} \geq \beta^0 \mathbf{v}^{\alpha_2}$ **then**
    $\alpha_1, \alpha_2, \mathbf{v}^{\alpha_2}, \alpha_3 \leftarrow \alpha_2, \alpha_3, \mathbf{v}^{\alpha_3}, \alpha_2 + \frac{1}{\phi}(\alpha_4 - \alpha_2)$
    $\mathbf{v}^{\alpha_3} \leftarrow \textsc{PolicyEvaluation}(\pi + \alpha_3 \frac{\partial \mathbf{v}^{\pi}}{\partial \pi})$
  else
    $\alpha_4, \alpha_3, \mathbf{v}^{\alpha_3}, \alpha_2 \leftarrow \alpha_3, \alpha_2, \mathbf{v}^{\alpha_2}, \alpha_3 - \frac{1}{\phi}(\alpha_3 - \alpha_1)$
    $\mathbf{v}^{\alpha_2} \leftarrow \textsc{PolicyEvaluation}(\pi + \alpha_2 \frac{\partial \mathbf{v}^{\pi}}{\partial \pi})$
return $\pi + \frac{1}{2}(\alpha_2 + \alpha_3)\frac{\partial \mathbf{v}^{\pi}}{\partial \pi}, \frac{1}{2}(\mathbf{v}^{\alpha_2} + \mathbf{v}^{\alpha_3})$

B. Line Search Along the Gradient

Gradient ascent requires a step size α. *Line search* algorithms intelligently search along the gradient line for an α that results in a high utility [23]. Specific to POMDPs, just evaluating the ascending function is a relatively costly computation (policy evaluation). Luckily, circulant controllers have an efficient POLICYEVALUATION step (see the previous section). Also, we consider a specialized line search called *golden section search* [24]. It minimally evaluates the policy by leveraging the golden ratio ϕ. Algorithm 2 provides pseudocode that combines these for our LINESEARCH.

C. Algorithm Summary

Algorithm 1 is pseudocode that combines our policy evaluation and line search to compute a circulant controller. RAND initializes a random circulant controller parameters. COMPUTEGRADIENT computes Equation 6. PROJECTTOSIMPLEX projects each vector $\psi(x,\cdot)$ and $\eta(x,\omega,\cdot)$ to the simplex with a standard method [25]. PROJECTTOCIRCULANT projects each matrix $\eta(\cdot,\omega,\cdot)$ to the space of circulant matrices $\mathcal{C}_{|X\times S|}$, as stated in Section III-C, with P derived from Propositions 3 and 4. RELATIVEERROR computes the relative sum of differences to check for convergence.

V. THEORETICAL ANALYSIS

In this section algorithmically relevant properties of the proposed algorithm are proved and analyzed. The first proposition provides the base factorization which allows for fast policy evaluation.

Proposition 3: Given a circulant controller, Equation 2 has the factorization $\mathbf{M}^{\pi} = \mathbf{DC}$, where $\mathbf{D}$ is block diagonal and $\mathbf{C}$ is block circulant.

Proof: Let $\mathbf{M}_{x_i,x_j}(s,s') := \mathbf{M}^{\pi}(\langle x_i, s\rangle, \langle x_j, s'\rangle)$ denote the $|S| \times |S|$ matrix obtained by fixing an arbitrary $x_i, x_j \in X$ in Equation 2. We then have

$$\mathbf{M}^{\pi} = \begin{pmatrix} \mathbf{M}_{x_1,x_1} & \mathbf{M}_{x_1,x_2} & \dots & \mathbf{M}_{x_1,x_{|X|}} \\ \mathbf{M}_{x_2,x_1} & \mathbf{M}_{x_2,x_2} & \dots & \mathbf{M}_{x_2,x_{|X|}} \\ & \vdots & & \\ \mathbf{M}_{x_{|X|},x_1} & \mathbf{M}_{x_{|X|},x_2} & \dots & \mathbf{M}_{x_{|X|},x_{|X|}} \end{pmatrix}. \quad (10)$$

Defining the $|S| \times |S|$ matrices

$$\mathbf{D}_{x_i}(s,s') = \sum_{a \in A} \psi(a, x_i) T(s,a,s')$$

and

$$\mathbf{C}_{x_i,x_j}(s,s') = \begin{cases} \sum_{\omega \in \Omega} O(s',\omega)\eta(x_i,\omega,x_j) & \text{if } s = s' \\ 0 & \text{if } s \neq s' \end{cases}$$

we obtain a factorization for each block $\mathbf{M}_{x_i,x_j} = \mathbf{D}_{x_i}\mathbf{C}_{x_i,x_j}$. Substituting the block definitions into Equation 10 the matrix can be factored as

$$\mathbf{M}^{\pi} = \begin{pmatrix} D_{x_1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & D_{x_{|X|}} \end{pmatrix} \begin{pmatrix} C_{x_1,x_1} & \dots & C_{x_1,x_{|X|}} \\ \vdots & \ddots & \vdots \\ C_{x_{|X|},x_1} & \dots & C_{x_{|X|},x_{|X|}} \end{pmatrix}.$$

To see that $\mathbf{C}$ is block circulant (that is circulant in the nodal indices), follows directly from the circulant structure of $\eta(\cdot,\omega,\cdot)$. Using the appropriate modular arithmetic for the indices, we have

$$\begin{aligned} \mathbf{C}_{x_i,x_j} &= diag\left(\sum_{\omega \in \Omega} O(\cdot,\omega)\eta(x_i,\omega,x_j)\right) \\ &= diag\left(\sum_{\omega \in \Omega} O(\cdot,\omega)\eta(x_{i+1},\omega,x_{j+1})\right) \\ &= \mathbf{C}_{x_{i+1},x_{j+1}}. \end{aligned}$$

■

The next proposition provides the existence of a projection onto the space of circulant controllers via explicit construction. This is the operator used in Algorithm 1.

Proposition 4: There is an orthogonal projection from the space of FSCs onto the space of circulant controllers.

Proof: Let $\mathbf{C}$ be a $n \times n$ circulant matrix and $\mathbf{Q}$ be the $n \times n$ circular shift matrix. The circular shift matrix can be used to define a basis for $\mathbf{C}$, yielding

$$\mathbf{C} = \sum_{k=1}^{n} \alpha_k \mathbf{Q}^k.$$

Then considering the vectorization $\mathbf{c}=vec(\mathbf{C})$ and $\mathbf{q}_k=\frac{1}{n^2}vec\left(\mathbf{Q}^k\right)$ we have

$$\mathbf{c}=\sum_{k=1}^{n}\alpha_k\mathbf{q}_k$$

where the circulant matrix is now represented as an element of $\mathbb{R}^{n^2}$. Further, the set of vectors $\mathbf{q}_1,\ldots,\mathbf{q}_n$ form an orthonormal basis for the space of circulant matrices. As a result, the projection of an arbitrary matrix $\mathbf{a}=vec(\mathbf{A})\in\mathbb{R}^{n^2}$ is now a standard Euclidean projection from $\mathbb{R}^{n^2}$ onto a subspace of $\mathbb{R}^{n^2}$ for which we have an orthonormal basis. The projection is then given by

$$P=\sum_{k=1}^{n}\mathbf{q}_k\mathbf{q}_k^T.$$

■

Proposition 5: Policy evaluation in the space of circulant controllers is $O(|S|^3|X|+|S||X|\log|X|)$.

Proof: The complexity of policy evaluation reduces to determining the complexity of the iteration given in Equation 9. The matrix $(\mathbf{I}-\gamma\mathbf{M}_D^\pi)$ is block diagonal with $|X|$ blocks of size $|S|\times|S|$. The inverse is obtained by inverting the individual blocks, resulting in $O(|S|^3|X|)$. Matrix-vector multiplications are then $O(|S|^2|X|)$ for $(\mathbf{I}-\gamma\mathbf{M}_D^\pi)$ and $\mathbf{D}$.

Lastly, we need to determine multiplication with $\mathbf{C}^0$. The extension of Proposition 2 to the block circulant case is straight forward. The circulant structure of $\mathbf{C}^0$ is with respect to the nodal indices. Therefore, letting $\mathbf{F}_{|X|}$ denote the Fourier matrix of size $|X|$, the block Fourier matrix is given by $\mathbf{F}_b=\mathbf{F}_{|X|}\otimes\mathbf{I}_{|S|}$ where $\mathbf{I}_{|S|}$ is the identify matrix of size $|S|$ and $\otimes$ denotes the Kronecker product. Then

$$\mathbf{C}^0=\mathbf{F}_b\mathbf{\Lambda}_b\mathbf{F}_b^{-1}$$

where $\mathbf{\Lambda}_b$ is a block diagonal matrix whose blocks are obtained by applying $\mathbf{F}_b$ to the first block row of $\mathbf{C}^0$. In our case, because each block of $\mathbf{C}^0$ is diagonal, $\mathbf{\Lambda}_b$ is a diagonal matrix. By grouping indices appropriately, a matrix-vector product with $\mathbf{F}_b$ can be computed via $|S|$ FFTs of length $|X|$ yielding $O(|S||X|\log|X|)$. The same follows for $\mathbf{F}_b^{-1}$ using inverse FFTs. As a result, multiplication with $\mathbf{C}^0$ is $O(|S||X|\log|X|)$. Combining the asymptotically significant pieces of the above yields the result.

■

This not only provides a reduction in the complexity of the inversion but the factorization reduces the memory requirements significantly by only storing the blocks needed to represent $\mathbf{C}$ and $\mathbf{D}$. Therefore, storage of $\mathbf{M}^\pi$ goes from $|X|^2|S|^2$ to $|S|^2|X|+|S||X|$.

Proposition 6: A $k-step$ line search algorithm in the space of circulant FSCs is $O\left(k\left[|S|^3|X|+|S||X|\log|X|\right]\right)$.

Proof: In the context of Algorithm 1 line search performs policy evaluations along the direction of the projected gradient. By Proposition 1 circulant matrices are closed under addition which implies all policies along the projected gradient are circulant. The result then follows from Proposition 5. ■

VI. Experiments

We evaluate the proposed circulant gradient ascent with line search in four robot POMDP scenarios. It is compared against two baseline algorithms in three distinct metrics. The resulting circulant policies are demonstrated in a home healthcare domain on a real robot in an actual home.

A. Experimental Setting

The standard three metrics are used [3], [4], [7]. The average sum of discounted rewards (ADR) is an unbiased measure of policy performance computed by sampled policy evaluation [3]. The time (T) to compute the policy (in seconds). The size of the final policy ($|\pi|$) is the total required number of parameters (floating point numbers).

All algorithms were implemented in Julia 1.5.2 on Ubuntu 18.04.5. Experiments were run on an Intel Core i7-6700HQ CPU at 2.60GHz with 16 GB of RAM. Each domain and algorithm configuration averaged over 10 trials for a horizon of 100, with a maximum of 900 seconds allotted per trial.

B. Baseline Algorithms

The nonlinear programming (NLP) formulation is a standard state-of-the-art POMDP solver for controllers [4], [7]. The algorithm solves for the optimal controller of fixed-sized ℓ. Here we use Julia's JuMP [26] NLP solver. We also compare our approach to the original vanilla gradient ascent (GA) [2] with a default $\alpha=0.01$ step size.

To evaluate the effect of both circulant controllers and line search, we evaluate both the proposed circulant gradient ascent (CGA) algorithm and CGA with line search (CGA+LA).

C. Domains

The proposed circulant controllers and algorithm are designed specifically to solve POMDP robotic scenarios which have different periodic modes of behavior: a walking robot, a pick-and-place robot, and a home healthcare robot. The complete source code and descriptions will be provided.

The *walking* domain is a simplified configuration-space ($\mathcal{C}$-space) planning scenario based on prior (PO)MDP-based models [10], [11]. A bipedal robot must move legs in a periodic pattern between different angles $\theta\in[\theta_{min},\theta_{max}]$ to move forward that is discretized for planning. However, different terrain features $\mathcal{T}=\{\text{flat},\text{stairs},\text{wall}\}$ are detected by a noisy sensor and each requires the speed of motion to vary lest the robot fall. $S=\theta_c\times\theta_p\times S_t$ denotes the current leg parameters, previous leg parameters, and terrain type. $A=\theta$ are the desired next leg parameters. $\Omega=\mathcal{T}$ are the noisy observations of the terrain. The agent is rewarded with 1 for moving forward, with discount $\gamma=0.9$, and 0 otherwise.

The *pick-and-place* domain requires both individual object-picking planning in $\mathcal{C}$-space [12] and planning how many among $N_o=\{1,\ldots,r\}$ multiple objects to bin together for delivery, given an order request of a certain number of objects $n_o\in N_o$ [13]. We assume a 3-DOF arm with a base pivot, an elbow joint, and a hand is parameterized by θ_b, θ_j, and θ_h, respectively. There are object locations $L_{o_i}=\{\text{shelf},\text{held},\text{box}\}$ and a box

Domain	$\|S\|$	$\|A\|$	$\|\Omega\|$	ℓ	NLP [4]			GA [2]			CGA			CGA+LS		
					ADR	T	$\|\pi\|$	ADR	T	$\|\pi\|$	ADR	T	$\|\pi\|$	ADR	T	$\|\pi\|$
Walking	12	2	3	2	**3.9**	1.3	8	1.9	0.6	8	1.9	0.4	**5**	**3.9**	**0.1**	**5**
Walking	12	2	3	4	**6.6**	2.3	40	1.9	2.4	40	1.7	1.4	**13**	3.7	**0.7**	**13**
Pick & Place	72	7	1	5	**9.9**	63.1	50	8.8	93.1	50	8.0	65.9	**34**	8.0	**10.7**	**34**
Pick & Place	72	7	1	10	7.6	190.5	150	**9.5**	395.6	150	8.3	218.8	**69**	**9.0**	**31.4**	**69**
Home Healthcare (H)	144	5	2	5	8.4	372.8	60	**9.3**	256.4	60	**9.2**	158.7	**28**	8.2	**71.0**	**28**
Home Healthcare (H)	144	5	2	10	—	—	—	—	—	—	**9.4**	740.3	**58**	8.3	**568.3**	**58**
Home Healthcare (A)	169	4	2	4	7.1	625.9	36	**8.5**	168.5	36	**8.6**	145.3	**18**	7.1	**107.6**	**18**
Home Healthcare (A)	169	4	2	8	—	—	—	—	—	—	**8.2**	467.9	**38**	7.6	**267.5**	**38**

TABLE I

RESULTS FROM SIMULATION COMPARING THE PERFORMANCE OF TWO CONTROLLER BASELINES NLP AND VANILLA GRADIENT ASCENT (GA) VERSUS THE PROPOSED CIRCULANT GA (CGA) AND CGA WITH LINE SEARCH (CGA+LS). FOUR DOMAINS ARE CONSIDERED, EACH VARYING THE NUMBER OF NODES (ℓ). METRICS INCLUDE: AVERAGE DISCOUNTED REWARD (ADR), TIME IN SECONDS (T), AND THE SIZE OF THE POLICY ($|\pi|$).

Fig. 2. Experiments on a real robot in an actual household environment. This implements the Home Healthcare (A) domain on an actual robot.

location $L_b = \{\text{table}, \text{held}, \text{conveyor belt}\}$. $S = \theta_b \times \theta_j \times \theta_h \times N_o \times L_b \times_i L_{o_i}$ describes all configurations of these factors. $A = \theta_b \cup \theta_j \cup \theta_h$ describes the movement actions of the arm. $\Omega = N_o$ describes the number of requested objects. The agent is only rewarded with a 1 when the correct requested objects are placed in the box and the box is placed on the conveyor belt, with a discount of $\gamma = 0.9$, and 0 otherwise.

The *home healthcare* domains consider a robot navigating in a household environment, searching for a human to either monitor or deliver medicine [14]. These tasks are among the most desired ones for a home healthcare robot [15] and is based on prior POMDP models [16], [27]. They are two different real LIDAR-mapped homes denoted *house (H)* and *apartment (A)*. Figure 2 shows the map for the apartment home. Its topological map is represented by regions R_m and edges E_m. Let $N(r)$ be the neighbors of region $r \in M_r$, with a maximum degree (number of neighbors) of $d = \max_r N(r)$. $S = S_r \times S_t$ denotes the possible configurations of robot and target human locations, with $S_r = S_t = M_r$. $A = \{1, \ldots, d\}$ denotes which neighbor region (via its neighbor index) is selected next. $\Omega = \{\text{no}, \text{yes}\}$ denotes observing the human person to find or not. The agent is rewarded 1 for finding the human, with discount of $\gamma = 0.9$, and 0 otherwise.

D. Results and Discussion

Table I presents the results from simulations across the four domains and four controller algorithms varying the number of controller nodes ℓ. Figure 3 also visualizes the value over iteration for all four of the algorithms for the home healthcare (A) domain. We observe that while the NLP baseline does solve for the optimal fixed-sized controller, it relies on generic off-the-shelf NLP algorithms and is consequently quite slow. Vanilla gradient ascent (GA) is also very slow. GA's number of iterations is high because of its non-adaptive step size. It also has an expensive $\mathbf{Z}^{-1}$ computation per iteration. Both baselines struggle to scale well to the larger POMDPs. The proposed circulant gradient ascent (CGA), however, are demonstrated to rapidly converge to good-performing circulant policies. Moreover, the proposed CGA with line search (CGA+LS) significantly improves convergence with its adaptive step size. Interestingly, the proposed CGA+LS also significantly saves on memory $|\pi|$, given the circulant structure of $\eta(\cdot, \omega, \cdot)$. Lastly, Figure 2 shows the execution of a circulant controller produced by CGA+LS on a real robot navigating in an actual household, demonstrating their effectiveness in practice.

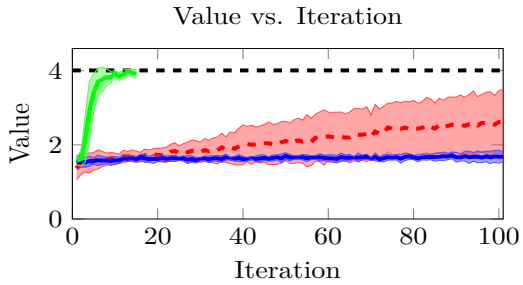

Fig. 3. Value for 100 iterations of the baseline GA (dotted red), CGA (solid blue), and CGA+LS (solid green) in the walking domain. The optimal value is shown (black dotted). Here, CGA+LS converges quickly and optimally.

VII. CONCLUSION

Circulant controllers are POMDP policy forms that can capture periodic action behaviors found in robotic domains. Circulant gradient ascent (CGA) and CGA with line search (CGA+LS) are rapidly-converging and memory-efficient algorithms for computing a circulant controller. Results in simulation and on a robot demonstrate their effectiveness.

REFERENCES

[1] 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.

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

[3] 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.

[4] 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.

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

[6] K. H. Wray and S. Zilberstein, "Approximating reachable belief points in POMDPs," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 117–122.

[7] K. H. Wray and S. Zilberstein, "Generalized controllers in POMDP decision-making," in *2019 IEEE International Conference on Robotics and Automation*, 2019, pp. 7166–7172.

[8] K. H. Wray, S. J. Witwicki, and S. Zilberstein, "Online decision-making for scalable autonomous systems," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017, pp. 4768–4774.

[9] M. J. Kochenderfer, *Decision Making Under Uncertainty: Theory and Application*. MIT Press, 2015.

[10] E. Schuitema, M. Wisse, T. Ramakers, and P. Jonker, "The design of LEO: A 2D bipedal walking robot for online autonomous reinforcement learning," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 3238–3243.

[11] R. Tedrake, T. W. Zhang, and H. S. Seung, "Learning to walk in 20 minutes," in *Proceedings of the 14th Yale Workshop on Adaptive and Learning Systems*, 2005, pp. 1939–1412.

[12] M. Gualtieri, A. ten Pas, and R. Platt, "Pick and place without geometric object models," in *2018 IEEE International Conference on Robotics and Automation*, 2018, pp. 7433–7440.

[13] C. Hernandez, M. Bharatheesha, W. Ko, H. Gaiser, J. Tan, K. van Deurzen, M. de Vries, B. Van Mil, J. van Egmond, R. Burger, *et al.*, "Team delft's robot winner of the amazon picking challenge 2016," in *Robot World Cup*. Springer, 2016, pp. 613–624.

[14] H. Robinson, B. MacDonald, and E. Broadbent, "The role of healthcare robots for older people at home: A review," *International Journal of Social Robotics*, vol. 6, no. 4, pp. 575–591, November 2014.

[15] E. Broadbent, R. Tamagawa, N. Kerse, B. Knock, A. Patience, and B. MacDonald, "Retirement home staff and residents' preferences for healthcare robots," in *Proceedings of the 18th IEEE International Symposium on Robot and Human Interactive Communication*, 2009, pp. 645–650.

[16] K. H. Wray and S. Zilberstein, "Policy networks: A framework for scalable integration of multiple decision-making models (extended abstract)," in *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*, 2019, pp. 2270–2272.

[17] M. Spaan and N. Vlassis, "Perseus: Randomized point-based value iteration for POMDPs," *Journal of Artificial Intelligence Research*, vol. 24, pp. 195–220, 2005.

[18] H. Kurniawati, D. Hsu, and W. S. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces." in *Robotics: Science and systems*, 2008.

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

[20] P. Poupart and C. Boutilier, "Bounded finite state controllers," in *Proceedings of Advances in Neural Information Processing Systems 16*, 2004, pp. 823–830.

[21] J. K. Pajarinen and J. Peltonen, "Periodic finite state controllers for efficient POMDP and DEC-POMDP planning," in *Advances in Neural Information Processing Systems*, 2011, pp. 2636–2644.

[22] E. J. Sondik, "The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs," *Operations Research*, vol. 26, no. 2, pp. 282–304, 1978.

[23] J. Nocedal and S. Wright, *Numerical Optimization*. Springer Science & Business Media, 2006.

[24] J. Kiefer, "Sequential minimax search for a maximum," *Proceedings of the American Mathematical Society*, vol. 4, no. 3, pp. 502–506, 1953.

[25] W. Wang and M. A. Carreira-Perpinán, "Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application," *arXiv preprint arXiv:1309.1541*, 2013.

[26] I. Dunning, J. Huchette, and M. Lubin, "JuMP: A modeling language for mathematical optimization," *SIAM Review*, vol. 59, no. 2, pp. 295–320, 2017.

[27] K. H. Wray, "Abstractions in reasoning for long-term autonomy," Ph.D. dissertation, University of Massachusetts, Amherst, MA, 2019.