Abstract—A variety of queries about stochastic systems boil down to study of Markov chains and their properties. If the Markov chain is large, as is typically true for discretized continuous spaces, such analysis may be computationally intractable. Nevertheless, in many scenarios, Markov chains have underlying structural properties that allow them to admit a low-dimensional representation. For instance, the transition matrix associated with the model may be low-rank and hence, representable in a lower-dimensional space. We consider the problem of learning low-dimensional representations for large-scale Markov chains. To that end, we formulate the task of representation learning as that of mapping the state space of the model to a low-dimensional state space, referred to as the kernel space. The kernel space contains a set of meta states which are desired to be representative of only a small subset of original states. To promote this structural property, we constrain the number of nonzero entries of the mappings between the state space and the kernel space. By imposing the desired characteristics of the structured representation, we cast the problem as the task of nonnegative matrix factorization. To compute the solution, we propose an efficient block coordinate gradient descent algorithm and theoretically analyze its convergence properties. Our extensive simulation results demonstrate the efficacy of the proposed algorithm in terms of the quality of the low-dimensional representation as well as its computational cost.

I. INTRODUCTION

Markov chains are a well-established framework to represent the evolution of stochastic systems. They have been widely used as a modeling tool in different applications including control [1], machine learning [2], and computational biology [3]. Moreover, they create the foundation for more complex probabilistic graphical models including hidden Markov models and Markov decision processes [4], [5].

In many practical settings, a system modeled as a Markov chain, has a large state space. For instance, fine discretization of a zero-input dynamical model with continuous space leads to a Markov chain with a huge discrete state space. The fact that analyzing such large-scale models may be intractable has motivated significant research on model reduction algorithms. These algorithms attempt to create compressed abstractions of a large Markov chain to enable efficient downstream analysis without compromising the accuracy of the inference tasks.

A key enabling factor in abstracting Markov chains is the existence of certain structural properties of the characterizing transition probabilities. For instance, the transition probabilities, captured by a stochastic matrix, may be low-rank or sparse. Therefore, one can exploit these structural properties to construct abstractions that accurately approximate the original model. The present work is motivated by the state aggregation framework for reducing the complexity of reinforcement learning and control systems [6]. State aggregation schemes attempt to group similar states into a small number of meta states, which are typically handpicked based on domain-specific knowledge [7], [8], or based on a given similarity metric or feature function [9].

In this paper, we propose an algorithm to find a surrogate representation that approximates the original Markov chain while having a low-dimensional state space. The proposed framework aims to learn a bidirectional mapping between the original high-dimensional state space and the low-dimensional state space, referred to as the kernel space. More specifically, we search for a representation of the probability transition matrix of the Markov chain in a low-dimensional state space. We model the task of learning the mappings and the kernel space as a combinatorial optimization problem. Then, we relax this formulation and establish a sparsity-promoting constrained nonnegative matrix factorization problem with the goal of factorizing the the transition matrix into three factors that determine the bidirectional mapping as well as the structure of the low-dimensional state space.

In order to solve this factorization, we propose an efficient block coordinate gradient descent algorithm that starting from an initial guess, learns the bidirectional mapping and the kernel space in an iterative fashion. We further analyze convergence properties of the the proposed iterative algorithm and demonstrate that under certain conditions on the step sizes of the gradient steps, the algorithm converges to a stationery point of the proposed optimization problem. One advantage of the proposed abstraction is that it is independent of the downstream analysis. To the best of our knowledge, this is the first matrix factorization approach toward abstraction of Markov chains. We complement our methodology with extensive simulation results where we demonstrate efficacy of our method in learning meaningful and representative low-dimensional structures of Markov chains.

The rest of the paper is organized as follows. We present the related work in the remainder of this section. Section II overviews the preliminary concepts and describes the problem statement. Section III focuses on the proposed nonnegative matrix factorization formulation and outlines
the solution approach. In Section IV, we demonstrate the performance of the proposed method in different settings. Finally, Section V states the concluding remarks and points to future directions.

A. Related Work

In the analysis of dynamical systems, different model-reduction techniques have been designed, such as approximating the transfer operators [10], dynamic mode decomposition [11], and data-driven approximations [12]. In control theory and reinforcement learning, state abstraction has been widely studied as a way of reducing the computational complexity of computing the optimal controller or optimal value function [8], [13]. Additionally, a related direction of research, called representation learning, tries to construct basis functions for representing high-dimensional value functions. Different Laplacian-based methods have been proposed to generate a surrogate for the exact transition operator [14]–[16].

Matrix factorization is an optimization framework that decomposes a matrix into a product of two or more matrices [17]. In contrast to spectral-based decomposition, in matrix factorization, one can impose desired structural properties. Common structural properties are those of being low-rank or sparse that can be promoted by nuclear norm regularization and $\ell_1$-norm regularization, respectively. Furthermore, matrix factorization is typically amenable to efficient gradient descent solutions.

Recovery of a low-rank probability transition matrix has been considered in [18]–[21]. The majority of the proposed methods are based on spectral decomposition framework. In contrast, we propose a matrix factorization formulation which can easily induce different desired structural properties by imposing additional constraints.

The problem of decomposing a matrix into a product of factors arises in different applications such as learning Markov models [22] and bioinformatics [23]. What makes matrix factorization methods appealing is the fact that their solution complexity depends on the rank of the factors which is typically much smaller than the input matrix. Nevertheless, matrix factorization has not been previously used for state abstraction over Markov chains.

II. Problem Formulation

In this section, we provide the outline of the related concepts and definitions, and formally state the problem of learning representations for Markov chains.

A. Preliminaries

In this paper, we focus on time-homogeneous discrete-time Markov chains with finite state space, as formally defined below.

Definition 1 (Markov Chain). A Markov chain is a random process such that its evolution is characterized by a tuple $\mathcal{MC} = (S, \mu_{\text{init}}, P)$, where

- $S$ is a finite set of states,
- $\mu_{\text{init}}$ is an initial distribution over the states,
- $P : S \times S \rightarrow [0, 1] \subseteq \mathbb{R}$ is a probabilistic transition function such that $\forall s \in S : \sum_{s' \in S} P(s, s') = 1$.

A finite path in $\mathcal{MC}$ is a realization of a finite-length sequence $X_0, X_1, \ldots$, of states, denoted by $\sigma = x_0 x_1 x_2 \ldots \in S^*$, such that $x_0$ is in the support of $\mu_{\text{init}}$ and $\forall i \in \mathbb{Z} : P(x_i, x_{i+1}) > 0$. Using the Markovian property, the probability of going from state $s_i$ at time step $t$ to state $s_j$, in $m$ time steps, is

\[
Pr(X_{t+m} = j|X_t = i) = P^{(m)}_{ij},
\]

where $P^{(m)}_{ij} = [P^m]_{ij}$.

Runnegburg [24] introduced the notion of Markov chains with small rank as a type of dependence that is close to independence. Hoekstra [25] has further analyzed properties of Markov chains with small rank. Next, we provide the formal definition of the nonnegative rank of a Markov chain that will later motivate the proposed transition matrix decomposition.

Definition 2 (Nonnegative Rank of Markov Chain). Let $P$ denote the transition matrix of a Markov chain $\mathcal{MC}$. The nonnegative rank of $\mathcal{MC}$ is the smallest $k \in \mathbb{N}$ for which the following decomposition exists:

\[
Pr(X_{t+1}|X_t) = \sum_{i=1}^{k} f_i(X_t) g_i(X_{t+1}),
\]

where $f_1, f_2, \ldots, f_k$ and $g_1, g_2, \ldots, g_k$ are real-valued functions mapping $S$ to $\mathbb{R}_+$.

In particular, $f_1, f_2, \ldots, f_k$ denote the left Markov features and $g_1, g_2, \ldots, g_k$ denote the right Markov features. Without loss of generality, one can assume that the left and right Markov features are probability mass functions. Notice that if the nonnegative rank of $\mathcal{MC}$ is $k$, it holds that [25]:

$$\text{rank}(P) \leq k.$$ 

In other words, the nonnegative rank of a Markov chain upperbounds the rank of its transition matrix. The next proposition states a critical property of nonnegative rank of Markov chains that we exploit in the proposed decomposition formulation.

Proposition 1 (Decomposition into Stochastic Matrices [21]). The nonnegative rank of a Markov chain is $k$ if and only if there exists $U \in \mathbb{R}_+^{k \times k}$, $P \in \mathbb{R}_+^{k \times k}$, and $V \in \mathbb{R}_+^{k \times n}$ such that

$$P = U \tilde{P} V,$$

where $U$, $P$, and $V$ are stochastic matrices, i.e., $U 1 = 1$, $P 1 = 1$, and $V 1 = 1$.

Proposition 1 establishes that $\tilde{P}$ resembles a surrogate lower-dimensional model for the transition matrix $P$ if $P$...
admits a low nonnegative rank. We refer to the state space and the transitions in this abstract model as kernel space and kernel transitions, respectively. In the decomposition model in Proposition 1, \( U \) maps the state of the original Markov chain into a smaller set of states in the kernel space, \( P \) indicates a transition between the states in the kernel space, and \( V \) maps the kernel states back to original states. In control and reinforcement learning, rows of \( U \) correspond to aggregation distributions while rows of \( V \) correspond to disaggregation distributions [26]. Low-rank decomposition of Markov chains have led to various reduced-order representations, such as membership model which tries to identify a small number of latent variables [21].

The strength of the model abstraction setting introduced in Proposition 1 is that it is independent of the downstream analysis to be performed on the Markov chain. Therefore, once such decomposition is found, the abstracted model can be used for accelerating different types of analyses. For instance, in the next proposition, we show how the \( m \)-step transition matrix in (1) can be computed more efficiently by using the factorized model.

**Proposition 2** (Efficient \( m \)-Step Transition). Given a Markov chain \( MC = (S, \mu_{init}, P) \), assume that a perfect low-rank decomposition of the transition matrix exists such that \( P = U \tilde{P} V \), \( \tilde{P} \in \mathbb{R}^{k \times k} \). The \( m \)-step transition matrix of \( MC \) can be computed by

\[
Pr(X_{t+m}|X_t) = \sum_{i_1}^{k} \sum_{i_2}^{k} U_{i_1,i} (\tilde{P} K^{m-1})_{i_1,i_2} V_{i_2,i_{t+m}},
\]

where \( K = UV \).

Hence, one can reduce the complexity of computing the \( m \)-step transition matrix from \( \mathcal{O}(mn^2) \) to \( \mathcal{O}(mk^2) \). The decomposition of transition matrix has inspired different approaches including spectral-based methods. Nevertheless, use of nonnegative matrix factorization has been less explored. Generally, nonnegative matrix factorization seeks to minimize a divergence metric between the input matrix and the product of its factors. But, more importantly, it allows one to impose additional desired constraints such as sparsity on the factors [17].

**B. Problem Statement**

In many applications, often the Markov chain model of a system has underlying structural properties, including possession of a low-rank or sparse transition function. Motivated by this fact, we look for an abstraction of a Markov chain in a low-dimensional kernel space. To that end, we need to find the mapping from the original state space to the state space of the abstracted (surrogate) model as well as the inverse mapping from the state space of the surrogate model to the original state space. Figure 1 demonstrates a pictorial overview of the mapping between the spaces. Essentially, a transition in the original model can be represented through three steps: step 1 maps the state in the original model to a meta state in the surrogate model; step 2 is a transition inside the surrogate model; and step 3 is a mapping from a meta state back to the original states. Additionally, we would like the meta states to be representative of a small subset of states. This requirement means that each meta state should be connected to as few states as possible. Therefore, we impose this property by looking for sparse mappings between the spaces.

**Problem 1.** Given a Markov chain \( MC = (S, \mu_{init}, P) \), we aim to find a kernel space and kernel transition, denoted by \( (\tilde{S}, \tilde{P}) \), with lower dimensionality, i.e., \( |\tilde{S}| \ll |S| \). Further, we look for a sparse bidirectional mapping \( (U, V) \) along the bidirectional mapping \( (U, V) \) must be such that the following decomposition property holds:

\[
P = U \tilde{P} V.
\]

**III. Approach**

Let \( n = |S| \) to be the size of the high-dimensional state space and \( k \) denote the nonnegative rank of the \( MC \). Let \( \mathcal{D} : \mathbb{R}^{n \times n} \times \mathbb{R}^{n \times n} \rightarrow \mathbb{R}_+ \) denote a metric on the space of \( n \times n \) matrices. As we discussed in Section III, we seek to promote sparsity patterns in the rows of the bidirectional mapping \( (U, V) \). Therefore, in order to find the factorization in Problem 1, we propose the following optimization task

\[
\min_{U, \tilde{P}, V} \mathcal{D}(P, U \tilde{P} V)
\]

\[
\text{s.t. } \sum_{j=1}^{k} U_{ij} = 1, \quad \|u_i\|_0 \leq s_i^{(u)}, \forall i \in [n],
\]

\[
\sum_{j=1}^{k} \tilde{P}_{ij} = 1, \quad \forall \ell \in [k],
\]

\[
\sum_{j=1}^{n} V_{ij} = 1, \quad \|v_\ell\|_0 \leq s_\ell^{(v)}, \forall \ell \in [k],
\]

Fig. 1: Mapping between high- and low-dimensional spaces. A transition between two states in the original Markov chain \( (P_{ij}) \) is equivalent to concatenation of the following sequence: a mapping from high-dimensional space to low-dimensional space \( (U_{ij}) \), a transition in the low-dimensional space \( (\tilde{P}_{ij}) \), and a mapping from low-dimensional space back to the high-dimensional space \( (V_{ij}) \).
Algorithm 1 Block Coordinate Gradient Descent (BCGD)

1: **Input:** Probability transition matrix $P$, number of low-dimensional states $k$, step sizes $\alpha$, $\beta$, and $\gamma$, regularization parameters $\lambda_u$ and $\lambda_v$, maximum number of iterations $T$
2: **Output:** Factor matrices $U$, $\tilde{P}$, and $V$
3: **Initialization:** Initialize $U_0$ at random
4: for $t = 0, 1, 2, \ldots, T-1$
5: Update rule for $U_{t+1}$
   - $\nabla f(U_t) = -(P - U_t \tilde{P} V_t) V_t^\top \tilde{P}^\top$
   - $U_{t+1} = U_t - \alpha_t \nabla f(U_t)$
   - $U_{t+1} = \Pi_{\Delta U} \left( T_{2\lambda_u} (U_{t+1}) \right)$ (Algorithm 2)
6: Update rule for $\tilde{P}_{t+1}$
   - $\nabla f(\tilde{P}_t) = -U_{t+1}^\top (P - U_{t+1} \tilde{P}_t V_t) V_t^\top$
   - $\tilde{P}_{t+1} = \tilde{P}_t - \beta_t \nabla f(\tilde{P}_t)$
   - $\tilde{P}_{t+1} = \Pi_{\Delta \tilde{P}} (\tilde{P}_{t+1})$ (Algorithm 2)
7: Update rule for $V_{t+1}$
   - $\nabla f(V_t) = -\tilde{P}_{t+1}^\top U_{t+1}^\top (P - U_{t+1} \tilde{P}_{t+1} V_t)$
   - $V_{t+1} = V_t - \gamma_t \nabla f(V_t)$
   - $V_{t+1} = \Pi_{\Delta V} \left( T_{2\lambda_v} (V_{t+1}) \right)$ (Algorithm 2)
8: end for

where $\| \cdot \|_0$ is the so-called $\ell_0$-norm and returns the number of nonzero entries of its argument, and $\{s_i^{(u)}\}$ and $\{s_i^{(v)}\}$ are positive integers that determine the extent of the desired sparsity structure in the rows of $U$ and $V$. Notice that because of the $\ell_0$-norm constraints, (5) is a combinatorial optimization problem and generally NP-hard to solve. Therefore, we propose to relax these constraints by using the $\ell_1$-norm which is the convex envelope of the $\ell_0$-norm and is known to promote sparsity in the solution of an optimization problem. Following this idea and by specifying $D(X,Y) = \frac{1}{2} \| X - Y \|_F^2$ as the metric, we consider the relaxed and regularized problem

$$
\min_{U,P,V} \frac{1}{2} \| P - U \tilde{P} V \|_F^2 + \lambda_u \| U \|_1 + \lambda_v \| V \|_1
$$

s.t. $U1 = 1$, $\tilde{P}1 = 1$, $V1 = 1$.

Here, $\lambda_u > 0$ and $\lambda_v > 0$ are the regularization parameters that determine the sparsity level of the rows of the bidirectional mapping matrices $U$ and $V$. Recall that sparsity of $U$ means that each state is mapped to one (or few) meta state(s). On the other hand, sparsity of $V$ asks for mapping of a meta state to a few states. This sparsity promoting terms ensure that the meta states are a good representative of their corresponding states. Additionally, the constraints of the optimization make sure that each of matrices are a stochastic matrix, i.e., can be interpreted as a transition matrix.

The objective function in (5) consists of a convex function and hence is convex in each of the matrices when the other matrices are fixed. However, due to the fact that the first term in the objective function contains a product of the unknowns, (5) is generally a nonconvex program. Notice that even if $P$ is perfectly decomposable into the corresponding factors, any permutation of the low-rank abstraction $\tilde{P}$ is also a solution. Therefore, in general, Problem (5) has at least $k!$ global optima.

To facilitate a computationally efficient search for the solution of (5), we rely on a modified gradient search algorithm which exploits the special structures of $U$, $\tilde{P}$, and $V$. The algorithm (summarized as Algorithm 1) is essentially a block coordinate gradient descent (BCGD) method that alternatively updates matrices $U$, $\tilde{P}$, and $V$ in an iterative fashion starting from an initial point $(U_0, \tilde{P}_0, V_0)$. That is, in the $(t + 1)$th iteration ($t = 0, \ldots, T - 1$ where $T$ is the total number of iterations), given $(U_t, \tilde{P}_t, V_t)$ we optimize with respect to $U$ to find $U_{t+1}$. Similarly we find update $\tilde{P}_{t+1}$ and $V_{t+1}$ by using the values $(U_{t+1}, \tilde{P}_t, V_t)$ and $(U_{t+1}, \tilde{P}_{t+1}, V_t)$, respectively.

In Algorithm 1, $T_{\eta}(x) = ([x] - \eta)_+ \text{sgn}(x)$ is the so-called shrinkage-thresholding operator that acts on each element of the given matrix, and $\Pi_{\Delta \lambda} (\cdot)$ denotes the projection operator that projects each row of its argument onto the probability simplex in $\mathbb{R}^d$. This projection can be efficiently computed by the method of [27, Algorithm 1] that we summarize in Algorithm 2 for completeness.

Algorithm 2 Projection onto $\Delta_d$

1: **Input:** $y = [y_1, \ldots, y_d]^T \in \mathbb{R}^d$
2: **Output:** projection $\Pi_{\Delta_d}(y)$
3: Sort $y$ in the ascending order as $y(1) \leq \cdots \leq y(d)$
4: for $i = d-1, d-2, \ldots, 1$
5: $b_i = \sum_{j=i+1}^d y(j) - 1$
6: if $b_i \geq y(i)$ then
7: $\bar{b} = b_i$
8: return $\Pi_{\Delta_d}(y) = (y - \bar{b})_+$
9: end if
10: end for
11: $\bar{b} = \sum_{j=1}^d y(j) - 1$
12: return $\Pi_{\Delta_d}(y) = (y - \bar{b})_+$

A. Convergence Analysis of BCGD

In this section, we analyze the convergence properties of BCGD. Specifically, in Theorem 1 we establish that given judicious choices of the step sizes, the value of the objective function in (5) decreases as one alternates between updating the factor matrices which in turn implies that the BCGD algorithm converges to a stationary point of the nonconvex optimization task in (5).

**Theorem 1.** Assume the step sizes of Algorithm 1 satisfy

$$
\alpha_t = \frac{C_1 \| \nabla f(U_t) \|_F^2}{\| \nabla f(U_t) \|_F^2},
$$

$$
\beta_t = \frac{C_2 \| \nabla f(V_t) \|_F^2}{\| U_{t+1} \|_F^2},
$$
where $C_1, C_2, C_3 \in (0, 2)$. Then, the solution $(U^*, \tilde{P}^*, V^*)$ found by the BCGD scheme is a stationary point of $f(U, \tilde{P}, V)$.

**Proof.** Let

$$f(U, \tilde{P}, V) = \frac{1}{2} \|P - U\tilde{P}V\|_F^2.$$  

For the proposed algorithm to converge, it must hold that

$$f(U_{t+1}, \tilde{P}_{t+1}, V_{t+1}) \leq f(U_t, \tilde{P}_t, V_t),$$  

for all $t$. First note that since $T_\eta(x)$ and $\Pi_{\triangle_d}$ are projections onto convex sets of constraints (the former being the projection operator onto the $\ell_1$ ball), following a similar analysis as those in the proofs of the projected gradient descent and the iterative shrinkage-thresholding algorithms (ISTA) [28], [29],

$$f(U_{t+1}, \tilde{P}_t, V_t) \leq f(U_{t+\frac{1}{2}}, \tilde{P}_t, V_t),$$

$$f(U_{t+\frac{1}{2}}, \tilde{P}_{t+1}, V_t) \leq f(U_{t+\frac{1}{2}}, \tilde{P}_{t+\frac{1}{2}}, V_t),$$

$$f(U_{t+\frac{1}{2}}, \tilde{P}_{t+\frac{1}{2}}, V_{t+\frac{1}{2}}) \leq f(U_{t+\frac{1}{2}}, \tilde{P}_{t+1}, V_{t+\frac{1}{2}}).$$

Thus, it suffices to show

$$f(U_{t+\frac{1}{2}}, \tilde{P}_t, V_t) \leq f(U_t, \tilde{P}_t, V_t),$$  

$$f(U_{t+\frac{1}{2}}, \tilde{P}_{t+\frac{1}{2}}, V_t) \leq f(U_{t+\frac{1}{2}}, \tilde{P}_{t+\frac{1}{2}}, V_t),$$

$$f(U_{t+\frac{1}{2}}, \tilde{P}_{t+\frac{1}{2}}, V_{t+\frac{1}{2}}) \leq f(U_{t+\frac{1}{2}}, \tilde{P}_{t+1}, V_{t+\frac{1}{2}}).$$

We now show that under [6], the sufficient condition [10] holds. Proof of the parts that [11] and [12] hold under [7] and [5], respectively, follows a similar argument.

**Note that**

$$f(U_{t+\frac{1}{2}}, \tilde{P}_t, V_t) - f(U_t, \tilde{P}_t, V_t)$$

$$= \frac{1}{2} \|P - U_t\tilde{P}_tV_t + \alpha_t \nabla f(U_t)\tilde{P}_tV_t\|_F^2$$

$$- \frac{1}{2} \|P - U_t\tilde{P}_tV_t\|_F^2$$

$$= \alpha_t \text{Tr} \left( (P - U_t\tilde{P}_tV_t)^\top \nabla f(U_t)\tilde{P}_tV_t \right)$$

$$+ \frac{\alpha_t^2}{2} \|\nabla f(U_t)\tilde{P}_tV_t\|_F^2. \quad (13)$$

**Now, consider the first term in the last line of (13).** Following straightforward linear algebra, we obtain

$$\text{Tr} \left( (P - U_t\tilde{P}_tV_t)^\top \nabla f(U_t)\tilde{P}_tV_t \right)$$

$$= -\text{Tr} \left( (P - U_t\tilde{P}_tV_t)(P - U_t\tilde{P}_tV_t)\tilde{P}_t^\top \tilde{P}_tV_t \right)$$

$$= -\text{Tr} \left( \tilde{P}_tV_t(P - U_t\tilde{P}_tV_t)V_t^\top \tilde{P}_t^\top \tilde{P}_tV_t \right)$$

$$= -\| (P - U_t\tilde{P}_tV_t)V_t^\top \tilde{P}_t^\top \|_F^2 = -\| \nabla f(U_t)\|_F^2.$$  

Therefore,

$$f(U_{t+\frac{1}{2}}, \tilde{P}_t, V_t) - f(U_t, \tilde{P}_t, V_t)$$

$$= \frac{\alpha_t^2}{2} \|\nabla f(U_t)\tilde{P}_tV_t\|_F^2 - \alpha_t \|\nabla f(U_t)\|_F^2$$

$$= \left( \frac{C_t^2}{2} - C_t \right) \frac{\|\nabla f(U_t)\|_F^2}{\|\nabla f(U_t)\tilde{P}_tV_t\|_F^2}, \quad (15)$$

where the last equality follows according to the definition of $\alpha_t$ in [6]. It is now clear that if $C_1 \in (0, 2)$ it must be the case that [10] holds, which in turn implies convergence of Algorithm 1. $\blacksquare$

**B. Computational Complexity of BCGD**

The computational complexity of the proposed BCGD algorithm is analyzed next. Note that the determining factor for cost per iteration of Algorithm [4] is computation of the gradients. Finding $\nabla f(U_t)$ incurs $O(nk)$ as it contains matrix products between $k \times k$ and $k \times n$ matrices. Similarly, $\nabla f(\tilde{P}_t)$ and $\nabla f(V_t)$ require $O(nk)$ computational costs. Thus, Algorithm [4] incurs a linear complexity of $O(nkT)$.
IV. SIMULATION RESULTS

We implemented the proposed BCGD algorithm for non-negative matrix factorization in Python. We evaluated the performance of the proposed abstraction solution for a variety of parameters. In particular, we investigated the effect of sparsity-promoting term on the approximation quality of the factorized transition matrix. We also ran the algorithm for different values of step size and compared the convergence of BCGD. For these simulations, we generated a transition matrix $P$ of size $100 \times 100$ for the original Markov chain. The transition matrix has a rank of 25 and is constructed by multiplying three stochastic matrices. Each stochastic matrix is generated by independently sampling its rows by a uniform sampling from the simplex of proper size. We set the number of iterations of BCGD to 1000. If the difference between two consecutive instances of a factor, i.e., $U_t$, $P_t$, or $V_t$ falls below a threshold $10^{-8}$ of their magnitude, the algorithm terminates, where the magnitudes are measured by Frobenius norm. Similarly, if the difference between the objective function falls below $10^{-8}$, the algorithm terminates. We run the simulation with each set of parameters for 10 independent instances and report the average values along the standard deviations. All simulations were run on a machine with 2.0 GHz Intel Core i7-4510U CPU and with 8.00 GB RAM.

A. Effect of Regularization Parameter on Performance

One of the key differences of the proposed abstraction formulation is the integration of a $\ell_1$-norm regularization term in the optimization objective. This term promotes sparsity for the bidirectional mapping between the original and the kernel space. Intuitively, this sparsity ensures that the meta states in the kernel space are representative of a small number of original states. The sparsity level of the mappings depends on regularization parameters $\lambda_u$ and $\lambda_v$.

In Figure 2, we demonstrate the effect of these parameters on the quality of the solutions. We compare four different values, more specifically, $\lambda = \lambda_u = \lambda_v \in \{0, 0.001, 0.005, 0.01\}$. The results in Figure 2(a) show that a careful selection of $\lambda$, while increasing sparsity, does not affect the quality of the decomposition in terms of the approximation error, computed by $\|P - U_t P V_t\|^2_F$. However, a large $\lambda$ leads to high approximation error. Therefore, one has to find the right trade-off between lower approximation error and higher sparsity of the mappings. Furthermore, we can see that error significantly reduces once the size of the kernel transition is set to values higher than 20. Above that value, the approximation error only slightly changes. Therefore, the proposed algorithm is successful in identifying a low-rank representation.

Figure 2(b) depicts the running time of the BCGD algorithm. As derived in Section IV-B, the running time is linear with respect to the kernel size and the simulations reflect that. Furthermore, the addition of $\ell_1$-norm regularization term has negligible effect on the running time. Note that for $\lambda = 0.01$, the algorithm terminates early as it cannot improve the objective value sufficiently. Furthermore, for this choice of $\lambda$, we can observe high variance in the running time.

B. Effect of Step Size on Convergence

In Section IV-A, we derived necessary conditions on the step sizes $\alpha$, $\beta$, and $\gamma$ for the convergence of the BCGD algorithm. In this section, we show the sensitivity of the convergence to different values of the step size. To that end, we ran the algorithm for different values of step size that we kept constant throughout the run. In particular, we ran the algorithm for $\alpha = \beta = \gamma \in \{0.002, 0.02, 0.2\}$. Figure 3 depicts the evolution of error over the course of 500 iterations of BCGD. While the algorithm converges for all three values, the smaller step sizes achieve lower approximation error at the end. We also observed that the algorithm would often diverge for step sizes over 0.2.

V. CONCLUSION

We studied the problem of approximating a large-scale Markov chain by a surrogate model in a low-dimensional state space, called kernel space. In order to find the surrogate model, we exploited the low-rank representation of the original transition matrix. The proposed low-rank representation decomposes the transition matrix into three factors, indicating the forward and backward mappings between the original and the kernel space as well as the kernel transition. We proposed a nonnegative matrix factorization formulation that learns the low-rank representation as well as the sought mappings while promoting a sparse connection between the high and low-dimensional states. We showed that the formulated optimization is amenable to an iterative solution that sequentially updates the desired factors and converges to a stationary solution under a judicious schedule of step sizes. Finally, in a variety of examples, we showed the quality of the approximate model, the computational complexity, as well as the convergence of the algorithm. As part of future work, we aim to extend the proposed matrix factorization formulation to model reduction of Markov decision processes. Furthermore, we would like to evaluate

---

1The code is available at https://github.com/MahsaGhasemi/state-abstraction
the abstracted Markov decision process in different analyses, including model checking and value function approximation.

REFERENCES

[1] H. Kushner, “Introduction to stochastic control,” tech. rep., Brown University Providence Division of Applied Mathematics, 1971.
[2] C. M. Bishop, Pattern recognition and machine learning, springer, 2006.
[3] S. R. Eddy, “Hidden Markov models,” Current opinion in structural biology, vol. 6, no. 3, pp. 361–365, 1996.
[4] D. Koller and N. Friedman, Probabilistic graphical models: Principles and techniques. MIT press, 2009.
[5] M. Ghasemi and U. Topcu, “Perception-aware point-based value iteration for partially observable Markov decision processes,” in Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2019.
[6] D. Abel, D. Arumugam, L. Lehnert, and M. Littman, “State abstractions for lifelong reinforcement learning,” in International Conference on Machine Learning, pp. 10–19, 2018.
[7] D. F. Rogers, R. D. Plante, R. T. Wong, and J. R. Evans, “Aggregation and disaggregation techniques and methodology in optimization,” Operations Research, vol. 39, no. 4, pp. 553–582, 1991.
[8] D. P. Bertsekas and J. N. Tsitsiklis, Neuro-dynamic programming, vol. 5. Athena Scientific Belmont, MA, 1996.
[9] J. N. Tsitsiklis and B. Van Roy, “Feature-based methods for large scale dynamic programming,” Machine Learning, vol. 22, no. 1-3, pp. 59–94, 1996.
[10] L. Molgedey and H. G. Schuster, “Separation of a mixture of independent signals using time delayed correlations,” Physical review letters, vol. 72, no. 23, p. 3634, 1994.
[11] P. J. Schmid, “Dynamic mode decomposition of numerical and experimental data,” Journal of fluid mechanics, vol. 656, pp. 5–28, 2010.
[12] S. Klus, F. Nüske, P. Kolthai, H. Wu, I. Kevrekidis, C. Schütte, and F. Noć, “Data-driven model reduction and transfer operator approximation,” Journal of Nonlinear Science, vol. 28, no. 3, pp. 985–1010, 2018.
[13] Z. Ren and B. H. Krogh, “State aggregation in Markov decision processes,” in Proceedings of the 41st IEEE Conference on Decision and Control, 2002., vol. 4, pp. 3819–3824, IEEE, 2002.
[14] J. Johns and S. Mahadevan, “Constructing basis functions from directed graphs for value function approximation,” in Proceedings of the 24th international conference on machine learning, pp. 385–392, ACM, 2007.
[15] R. Parr, C. Painter-Wakefield, L. Li, and M. Littman, “Analyzing feature generation for value-function approximation,” in Proceedings of the 24th international conference on Machine learning, pp. 737–744, ACM, 2007.
[16] M. Petrik, “An analysis of Laplacian methods for value function approximation in MDPs,” in IJCAI, pp. 2574–2579, 2007.
[17] D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” in Advances in neural information processing systems, pp. 556–562, 2001.
[18] D. Hsu, S. M. Kakade, and T. Zhang, “A spectral algorithm for learning hidden Markov models,” Journal of Computer and System Sciences, vol. 78, no. 5, pp. 1460–1480, 2012.
[19] Q. Huang, S. M. Kakade, W. Kong, and G. Valiant, “Recovering structured probability matrices,” arXiv preprint arXiv:1602.06586, 2016.
[20] X. Li, M. Wang, and A. Zhang, “Estimation of Markov chain via rank-constrained likelihood,” in 35th International Conference on Machine Learning, ICML 2018, pp. 4729–4744, International Machine Learning Society (IMLS), 2018.
[21] A. Zhang and M. Wang, “Spectral state compression of Markov processes,” arXiv preprint arXiv:1802.02920, 2018.
[22] G. Cybenko and V. Crespi, “Learning hidden Markov models using nonnegative matrix factorization,” IEEE Transactions on Information Theory, vol. 57, no. 6, pp. 3963–3970, 2011.
[23] A. Hashemi, B. Zhu, and H. Vikalo, “Sparse tensor decomposition for haplotype assembly of diploids and polyploids,” BMC genomics, vol. 19, no. 4, p. 191, 2018.
[24] J. T. Runnenburg, Markov processes in waiting-time and renewal theory. PhD thesis, Thesis, Pootppers, Amsterdam, 1966.
[25] Æ. H. Hoekstra, “On Markov chains of finite rank,” 1983.
[26] D. P. Bertsekas, Dynamic programming and optimal control, vol. 1. Athena scientific Belmont, MA, 1995.
[27] Y. Chen and X. Ye, “Projection onto a simplex,” arXiv preprint arXiv:1101.6081, 2011.
[28] A. Beck and M. Teboulle, “A fast iterative shrinkage-thresholding algorithm for linear inverse problems,” SIAM journal on imaging sciences, vol. 2, no. 1, pp. 183–202, 2009.
[29] S. Bubeck et al., “Convex optimization: Algorithms and complexity,” Foundations and Trends® in Machine Learning, vol. 8, no. 3-4, pp. 231–357, 2015.