Markov Chain Monte Carlo for Koopman-based Optimal Control

João Hespanha[1]

***Abstract*— We propose a Markov Chain Monte Carlo (MCMC) sampling algorithm based on Gibbs sampling with parallel tempering to solve nonlinear optimal control problems. The algorithm is applicable to nonlinear systems with dynamics that can be approximately represented by a finite dimensional Koopman model, potentially with high dimension. Linearity of the Koopman representation enables significant computational saving in sample generation. We use a video-game to illustrate the use of the method and compare it with a recent alternative approach, based on dynamic programming.**

I. INTRODUCTION

This paper address the optimal control of nonlinear systems that have reasonably accurate finite-dimensional representations in terms of their Koopman operator [15]. While Koopman's pioneering work is almost a century old, its use as a practical tool to model complex dynamics is much more recent and only became practical when computational tools became available for the analysis of systems with hundreds–thousands of dimensions. The use of Koopman models for control is even more recent, but has attracted significant attention in the last few years [3, 5, 16, 17, 21, 22].

The linear structure of the Koopman representation provides tremendous opportunities for control and permits very efficient solutions for optimal control when the Koopman dynamics are linear on the control input, as in [5, 16, 22]. However, linearity of the Koopman representation in the control input severely limits the class of applicable nonlinear dynamics. While bilinear representations are more widely applicable [3, 17, 21], they are also much harder to control.

When the set of admissible values for the control input is finite, the Koopman representation can be viewed as a switched linear system, where the optimal control problem becomes the optimal selection, at each time step, of one out of several admissible dynamics [4]. While the optimal control of switched system it typically computationally difficult [2, 18, 24], when the optimization criterion has a linear representation in the lifted state, the dynamic programming cost-to-go is concave and piecewise linear, with a simple representation in terms of the minimum over a finite set of linear functions. This observation enabled the design of efficient algorithms that combine dynamic programming with dynamic pruning [4].

This paper exploits the structure of the Koopman representation to develop very efficient Markov Chain Monte Carlo (MCMC) sampling method for optimal control. Following the pioneering work of [6, 10, 14], we draw samples from a Boltzmann distribution with energy proportional to the criterion to minimize. The use of Gibbs sampling [9, 10] is especially attractive because a full variable sweep of the basic Gibbs algorithm can exploit the linearity of the Koopman representation to minimize computation.

The original use of MCMC methods for combinatorial optimization relied on a gradual decrease in temperature, now commonly known as *simulated annealing*, to prevent the chain from getting trapped into states that do not minimize energy. An alternative approach relies on the use of multiple replicas of a Markov chain, each generating samples for a different temperature. The introduction of multiple replicas of a Markov chain to improve the mixing time can be traced back to [23]. The more recent form of *parallel tempering* (also known as *Metropolis–coupled MCMC*, or *exchange Monte Carlo*) is due to [7, 13].

The key contribution of this paper is an MCMC sampling algorithm that combines Gibbs sampling with parallel tempering to solve the switched linear optimizations that arise from the Koopman representation of nonlinear optimal control problems. This algorithm is computationally very efficient due to the combination of two factors: (i) the linear structure of the cost function enables the full variable sweep need for Gibbs sampling to be performed with the computation that scales linearly with the horizon length and (ii) parallel tempering can be fully parallelized across computation cores, as noted in [8]. While here we only explore parallelization across CPU cores, it is worth noting that in the last few year, hardware parallelization using GPUs and FPGAs has been used to achieve many orders of magnitude improvements on the number of samples that can be generated with MCMC sampling [1, 20].

The remaining of this paper is organized as follows: Section II shows how a nonlinear control problem can be converted into a switching linear systems optimization, using the Koopman operator. Section III provides basic background on MCMC optimization, Gibbs sampling, and parallel tempering. Our optimization algorithm is described in Section IV and its use is illustrated in Section V in the context of a video game. While this paper is self-contained, details of some of the algebraic derivations are omitted, but can be found in the technical report [12].

II. OPTIMAL CONTROL OF KOOPMAN MODELS

Given a discrete-time nonlinear system of the form

$$x_{t+1} = f(x_t, u_t), \quad \forall t \in \mathcal{T},\ x_t \in \mathcal{X},\ u_t \in \mathcal{U}_t, \qquad (1)$$

*This material is based upon work supported by the U.S. Office of Naval Research MURI grant No. N00014-23-1-2708 and by the National Science Foundation grant No. 2229876.

[1]Univ. of California, Santa Barbara, USA; hespanha@ucsb.edu

with the time t taking values over $\mathcal{T} \coloneqq \{1, \dots, T\}$, our goal is to solve a final-state optimal control problem of the form

$$J^* \coloneqq \min_{u \in \mathcal{U}} J(u), \quad J(u) \coloneqq g(x_T), \tag{2}$$

where $u \coloneqq (u_1, \dots, u_T) \in \mathcal{U} \coloneqq \mathcal{U}_1 \times \cdots \times \mathcal{U}_T$ denotes the control sequence to be optimized, which we assume finite but potentially with a large number of elements.

For each input $u \in \mathcal{U}_t$, $t \in \mathcal{T}$, the Koopman operator K_u for the system (1) operates on the linear space of functions $\mathcal{F}$ from $\mathcal{X}$ to $\mathbb{R}$ and is defined by

$$\varphi(\cdot) \in \mathcal{F} \mapsto \varphi(f(\cdot, u)) \in \mathcal{F}.$$

Assuming there is a finite dimensional linear subspace $\mathcal{F}_{\text{inv}}$ of $\mathcal{F}$ that is invariant for every Koopman operator in the family $\{K_u : \forall u \in \mathcal{U}_t,\ t \in \mathcal{T}\}$, there is an associated family of matrices $\{A_u \in \mathbb{R}^{n_\psi \times n_\psi} : \forall u \in \mathcal{U}_t,\ t \in \mathcal{T}\}$ such that

$$\psi_{t+1} = A(u_t)\psi_t, \quad \forall t \in \mathcal{T},\ \psi_t \in \mathbb{R}^{n_\psi},\ u_t \in \mathcal{U}_t, \tag{3}$$

where $\psi_t \coloneqq \begin{bmatrix} \varphi_1(x_t) & \cdots & \varphi_{n_\psi}(x_t) \end{bmatrix}' \in \mathbb{R}^{n_\psi}$ and the functions $\{\varphi_1(\cdot), \dots, \varphi_{n_\psi}(\cdot)\}$ form a basis for $\mathcal{F}_{\text{inv}}$ [4, 11]. If in addition, the function $g(\cdot)$ in (2) also belongs to $\mathcal{F}_{\text{inv}}$, there also exists a row vector $c \in \mathbb{R}^{1 \times n_\psi}$ such that

$$g(x_T) = c\psi_T, \tag{4}$$

which enables re-writing the optimization criterion (2) as

$$J(u) \coloneqq c\psi_T = cA(u_T) \cdots A(u_1)x_1. \tag{5}$$

We can thus view the original optimal control problem for the nonlinear system (1) as a switched linear control problem [4].

In practice, it is generally not possible to find a finite-dimensional subspace $\mathcal{F}_{\text{inv}}$ that contains $g(\cdot)$ and is invariant for the family of Koopman operators $\{K_u : u \in \mathcal{U}_t,\ t \in \mathcal{T}\}$. Instead, we typically work with a finite dimensional space that is "almost" invariant, meaning that the equalities in (3)–(4) hold up to a small error. However, to make this error small, one typically needs to work with high-dimensional subspaces, i.e., large values of n_ψ.

III. MCMC METHODS FOR OPTIMIZATION

To solve a combinatorial minimization of the form

$$J^* \coloneqq \min_{u \in \mathcal{U}} J(u). \tag{6}$$

over a finite set $\mathcal{U}$, it is convenient to consider the *Boltzmann distribution* with energies $J(u)$, $u \in \mathcal{U}$:

$$p(u; \beta) \coloneqq \frac{e^{-\beta J(u)}}{Q(\beta)},\ \forall u \in \mathcal{U}, \quad Q(\beta) \coloneqq \sum_{\bar{u} \in \mathcal{U}} e^{-\beta J(\bar{u})}, \tag{7}$$

for some constant $\beta \geqslant 0$. For consistency with statistical mechanics, we say that values of β close to zero correspond to high temperatures, whereas large values of β correspond to low temperatures. The normalization function $Q(\beta)$ is called the *canonical partition function*.

For $\beta = 0$ (infinite temperature), the Boltzmann distribution becomes uniform and all $u \in \mathcal{U}$ are equally probable. However, as we increase β (lower the temperature) all probability mass is concentrated on the subset of $\mathcal{U}$ that minimizes the energy $J(u)$. This motivates a procedure to solve (6): draw samples from a random variable $\boldsymbol{u}_\beta$ with Boltzmann distribution given by (7) with β sufficiently high (temperature sufficiently low) so that all samples correspond to states with the minimum energy and cost.

A. Markov Chain Monte Carlo Sampling

Consider a *discrete-time (time-homogeneous) Markov chain* $\{\boldsymbol{u}[1], \boldsymbol{u}[2], \dots\}$ on a finite set $\mathcal{U}$, with *transition probability* $p : \mathcal{U} \times \mathcal{U} \to [0, 1]$:

$$p(u'|u) = \mathrm{P}\left(\boldsymbol{u}[k+1] = u' \,|\, \boldsymbol{u}[k] = u\right),\ \forall u', u \in \mathcal{U},\ k \geqslant 1.$$

Combining the probabilities of all possible realization of $\boldsymbol{u}[k]$ in a row vector $p[k]$ and organizing the values of the transition probability $p(u'|u)$ as a *transition matrix* P with one $u' \in \mathcal{U}$ per column and one $u \in \mathcal{U}$ per row, enable us to express the evolution of the $p[k]$ as

$$p[k+K] = p[k]P^K, \quad \forall k, K \geqslant 1.$$

A Markov chain is called *regular* is that there exists an integer N such that all entries of P^N are strictly positive. In essence, this means that any $\bar{u} \in \mathcal{U}$ can be reached in N steps from any other $\tilde{u} \in \mathcal{U}$ through a sequence of transitions with positive probability $p(u'|u) > 0$, $u, u' \in \mathcal{U}$.

Theorem 1 (Fundamental Theorem of Markov Chains): For every regular Markov chain on a finite set $\mathcal{U}$ and transition matrix P, there exists a vector π with the following two properties:

1) There exists constants $c > 0, \lambda \in [0, 1)$ such that
$$\|p[k]P^K - \pi\| \leqslant c\lambda^K, \quad \forall k, K \geqslant 0. \tag{8}$$
2) The vector π is called the *invariant distribution* and is the unique solution to the *global balance equation*:
$$\pi P = \pi, \qquad \pi \mathbf{1} = 1, \qquad \pi \geqslant 0. \tag{9}$$

To use an MCMC method to draw samples from a desired distribution $p(u; \beta)$, we need to construct a discrete-time Markov Chain with two property:

1) Its (unique) invariant distribution π should match a desired sampling distribution $p(u; \beta)$ (balance); and
2) the chain should be regular.

This chain is used to "solve" our sampling problem by computing a sequence of samples $\boldsymbol{u}[1], \boldsymbol{u}[2], \dots$ but only accepting a subsequence of samples $\boldsymbol{u}[K+1], \boldsymbol{u}[2K+1], \dots$ where K is a "sufficiently large" integer so that the distribution $p[K+1] = p[1]P^K$ of the sample $\boldsymbol{u}[K+1]$ satisfies

$$\|p[1]P^K - \pi\| \leqslant \epsilon \quad \Leftarrow \quad K \geqslant \frac{\log c + \log \epsilon^{-1}}{\log \lambda^{-1}}. \tag{10}$$

for some "sufficiently small" value of ϵ. This selection of K guarantees that the samples $\boldsymbol{u}[K], \boldsymbol{u}[2K], \dots$ may not quite have the desired distribution, but are away from it by no more than ϵ. Since ϵ appears in (10) "inside" a logarithm, we can get ϵ extremely small without having to increase K very much. However, the dependence of K on λ can be more problematic because $\lambda \in [0, 1)$ can be very close to

1. The multiplication factor $1/\log\lambda^{-1}$ is often called the *mixing time* of the Markov chain and is a quantity that we would like to be small to minimize the number of "wasted samples."

To make sure that the chain's invariant distribution π matches a desired sampling distribution $p(u;\beta)$, $u \in \mathcal{U}$, we need the latter to satisfy the *global balance equation* (9), which can be re-written in non-matrix form as

$$p(u';\beta) = \sum_{u\in\mathcal{U}} p(u'|u)p(u;\beta), \quad \forall u' \in \mathcal{U}. \tag{11}$$

A sufficient (but not necessary) condition for global balance is *detailed balance*, which instead asks for

$$p(u|u')p(u';\beta) = p(u'|u)p(u;\beta), \quad \forall u', u \in \mathcal{U}. \tag{12}$$

B. Gibbs Sampling

Gibbs sampling starts with a function $p(u;\beta)$, $u \in \mathcal{U}$ that defines a desired multi-variable joint distribution up to a normalization constant. We use the subscript notation u_t to refer to the variable $t \in \{1,\dots,T\}$ in u and u_{-t} to refer to the remaining variables. The algorithm operates as follows:

Algorithm 1 (Gibbs sampling):

1) Pick arbitrary $\boldsymbol{u}[1] \in \mathcal{U}$.
2) Set $k = 1$ and repeat:
   - *Variable sweep*: For each $t \in \{1,\dots,T\}$:
     - Sample $\boldsymbol{u}_t[k+1]$ with distribution
     $$\frac{p(u_t, \boldsymbol{u}_{-t}[k];\beta)}{\sum_{\bar{u}_t} p(\bar{u}_t, \boldsymbol{u}_{-t}[k];\beta)}, \tag{13}$$
     which is the desired conditional distribution of $\boldsymbol{u}_t$, given $\boldsymbol{u}_{-t}[k]$.
     - Set $\boldsymbol{u}_{-t}[k+1] = \boldsymbol{u}_{-t}[k]$ and increment k.
3) go back to 2 until enough samples have been collected. □

1) Balance: It is straightforward to show that the Gibbs transition probability for a sample corresponding to the update of each variable $\boldsymbol{u}_t[k]$, $t \in \{1,\dots,T\}$ satisfies the detailed balance equation (12) for the desired distribution $p(u;\beta)$, which means that we also have global balance. Detailed balance is not preserved through the full variable sweep, but global balance is, which means that the state transition matrix associated with the samples $\boldsymbol{u}[1], \boldsymbol{u}[K+1], \dots$ has the desired invariant distribution.

2) Regularity and Convergence: It is also straightforward to show that the condition

$$p(u;\beta) > 0, \quad \forall u \in \mathcal{U}, \tag{14}$$

suffices to guarantee that the Markov chain generated by Gibbs sampling is regular *over the n steps of a variable sweep.* This is a consequence of the fact that, for any two possible states of the Markov chain, there is a path of nonzero probability over a full variable sweep.

C. Parallel tempering

Tempering aims at decreasing the mixing time of a Markov chain by creating high-probability "shortcuts" between states. It is applicable whenever we can embed the desired distribution into a family of distributions parameterized by a parameter $\beta \in [\beta_{\min}, \beta_{\max}]$, with the property that we may have slow mixing for the desired distribution, which corresponds to $\beta = \beta_{\max}$, but we can have fast mixing for the distribution corresponding to $\beta = \beta_{\min}$; which is typically the case of the Boltzmann distribution (7).

The key idea behind tempering is to select M values

$$\mathcal{B} \coloneqq \{\beta_1 \coloneqq \beta_{\min} < \beta_2 < \beta_3 < \cdots < \beta_M \coloneqq \beta_{\max}\}$$

and then use MCMC to generate samples for the joint distribution of an independent set of M random variable, one for each parameter value. We group these random variables as an M-tuple and denote the joint Markov chain by

$$\boldsymbol{u}[k] \coloneqq (\boldsymbol{u}^\beta[k] : \beta \in \mathcal{B}).$$

Eventually, from each M-tuple we only use the samples $\boldsymbol{u}^\beta[k]$, $\beta = \beta_{\max}$ that correspond to the desired distribution.

1) General algorithm: Tempering can be applied to any MCMC method with transition probabilities $p(u'|u;\beta)$, $\beta \in \mathcal{B}$ that satisfies:

1) for each $\beta \in \mathcal{B}$, the chain generated by $p(u'|u;\beta)$ is regular with a strictly positive transition matrix P_β;
2) for each $\beta \in \mathcal{B}$, $p(u'|u;\beta)$ satisfies the global balance equation for the desired distribution $p(u;\beta)$, $u \in \mathcal{U}$.

These properties suffice to guarantee that the tempered Markov chain defined below is regular and its transition probability satisfies the global balance equation for the joint distribution

$$p(u^\beta : \beta \in \mathcal{B}) \coloneqq \prod_{\beta\in\mathcal{B}} p(u^\beta;\beta), \tag{15}$$

which corresponds to independent distributions for the random variables $\boldsymbol{u}^\beta$ associated with different values of $\beta \in \mathcal{B}$.

The tempering algorithm uses a *flip function* $f_{\text{flip}} : \mathcal{U} \times \mathcal{U} \to [0,1]$ defined by

$$f_{\text{flip}}(u^{\beta_j}, u^{\beta_{j+1}}) = \min\left\{\frac{p(u^{\beta_{j+1}};\beta_j)}{p(u^{\beta_{j+1}};\beta_{j+1})}\frac{p(u^{\beta_j};\beta_{j+1})}{p(u^{\beta_j};\beta_j)}, 1\right\}, \tag{16}$$

and operates as follows:

Algorithm 2 (Tempering):

1) Pick arbitrary $\boldsymbol{u}[1] = (\boldsymbol{u}^\beta[1] \in \mathcal{U} : \beta \in \mathcal{B})$.
2) Set $k = 1$ and repeat:
   a) For each $\beta \in \mathcal{B}$, sample $\boldsymbol{u}^\beta[k+1]$ with probability $p(u'|\boldsymbol{u}^\beta[k];\beta)$ and increment k.
   b) *Tempering sweep*: For each $j \in \{1,\dots,M-1\}$:
      - Compute the flip probability
      $$p_{\text{flip}} = f_{\text{flip}}(\boldsymbol{u}^{\beta_j}[k], \boldsymbol{u}^{\beta_{j+1}}[k]) \tag{17}$$
      and set
      $$\boldsymbol{u}[k+1] = \begin{cases} \tilde{\boldsymbol{u}}[k] & \text{with prob. } p_{\text{flip}}, \\ \boldsymbol{u}[k] & \text{with prob. } 1-p_{\text{flip}}, \end{cases}$$
      where $\tilde{\boldsymbol{u}}[k]$ is a version of $\boldsymbol{u}[k]$ with the entries $\boldsymbol{u}^{\beta_j}[k]$ and $\boldsymbol{u}^{\beta_{j+1}}[k]$ flipped; and increment k.
3) go back to 2 until enough samples have been collected. □

Step 2a essentially corresponds to one step of a base MCMC algorithm (e.g., Gibbs sampling), for each value of

$\beta \in \mathcal{B}$. For the Boltzmann distribution (7), the flip function is given by

$$f_{\text{flip}}(u^{\beta_j}, u^{\beta_{j+1}}) = \min\left\{e^{-(\beta_j-\beta_{j+1})(J(u^{\beta_{j+1}})-J(u^{\beta_j}))}, 1\right\},$$

which means that the variable $\boldsymbol{u}^{\beta_j}[k]$ and $\boldsymbol{u}^{\beta_{j+1}}[k]$ will be flipped with probability one whenever $J(\boldsymbol{u}^{\beta_j}) < J(\boldsymbol{u}^{\beta_{j+1}})$. Intuitively, the main goal of the tempering sweep in step 2b is to quickly bring to $\boldsymbol{u}^{\beta_{\max}}[k]$ low-energy/low-cost samples that may have been "discovered" by other $\boldsymbol{u}^{\beta}[k]$, $\beta < \beta_{\max}$ with better mixing.

2) Balance: Since the sample extractions in step 2a are independent, the transition probability corresponding this step is given by

$$p\big((u'^{\beta} : \beta \in \mathcal{B}) \,|\, (u^{\beta} : \beta \in \mathcal{B})\big) = \prod_{\beta\in\mathcal{B}} p(u'^{\beta}|u^{\beta};\beta)$$

and it is straightforward to verify that this transition probability satisfies the global balance equation for the joint distribution in (15). Picking some $j \in \{1,\dots,M-1\}$, the transition probability corresponding to step 2b is given by

$$p\Big((u'^{\beta} : \beta \in \mathcal{B}) \,|\, (u^{\beta} : \beta \in \mathcal{B})\Big) =$$
$$\begin{cases} f_{\text{flip}}(u^{\beta_j}, u^{\beta_{j+1}}) & u'^{\beta_j} = u^{\beta_{j+1}},\, u^{\beta_j} = u'^{\beta_{j+1}},\, u'^{\beta} = u^{\beta},\, \beta \notin \{\beta_j, \beta_{j+1}\} \\ 0 & \text{otherwise.} \end{cases}$$

The flip function in (16) guarantees detailed balance for all $M-1$ steps in 2b and therefore global balance for all the combined steps in 2a and 2b within a full tempering sweep.

3) Regularity and Convergence: Assuming that for each $\beta \in \mathcal{B}$, the Markov chain generated by $p(u'|u;\beta)$ is regular with a strictly positive transition matrix P_β, any possible combination of states $\boldsymbol{u}[k] = (u^{\beta}[k] : \beta \in \mathcal{B})$ at time k can transition in one time step to any possible combination of states $\boldsymbol{u}[k+1] = (u^{\beta}[k] : \beta \in \mathcal{B})$ at time $k+1$ at step 2a. This means that this step corresponds to a transition matrix P with strictly positive entries. In contrast, each flip step corresponds to a transition matrix $P_{\text{flip}j}$ that is non-negative and right-stochastic, but typically has many zero entries. However, each matrix $P_{\text{flip}j}$ cannot have any row that is identically zero (because rows must add up to 1). This suffices to conclude that any product of the form

$$Q = P_{\text{flip}1} P_{\text{flip}2} \cdots P_{\text{flip}M-1} P$$

must have all entries strictly positive. This transition matrix corresponds to the transition from the start of step 2b in one "tempering sweep" to the start of the same step at the next sweep and defines a regular Markov chain, which suffices to prove convergence to the invariant distribution.

IV. MCMC FOR OPTIMAL CONTROL

As discussed in Section II, our goal is to optimize a criterion of the form

$$J(u) = cA(u_T)\cdots A(u_1)x_1, \qquad u \in \mathcal{U}. \tag{18}$$

In the context of MCMC sampling from the Boltzmann distribution (7), this corresponds to a Gibbs update for the control $\boldsymbol{u}_t^{\beta}[k+1]$ with distribution (13) given by

$$\frac{e^{-\beta J(u_t, \boldsymbol{u}^{\beta}_{-t}[k])}}{\sum_{\bar{u}_t} e^{-\beta J(\bar{u}_t, \boldsymbol{u}^{\beta}_{-t}[k])}} = \frac{e^{-\beta \boldsymbol{c}_t^{\beta}[k] A(u_t) \boldsymbol{x}_t^{\beta}[k]}}{\sum_{\bar{u}_t} e^{-\beta e^{-\beta \boldsymbol{c}_t^{\beta}[k] A(\bar{u}_t) \boldsymbol{x}_t^{\beta}[k]}}},$$

and a tempering flip probability in (17) given by

$$p_{\text{flip}} = \min\left\{e^{-(\beta_j-\beta_{j+1}) c(\boldsymbol{x}_T^{\beta_{j+1}}[k]) - \boldsymbol{x}_T^{\beta_j}[k]))}, 1\right\},$$

where

$$\boldsymbol{c}_t^{\beta}[k] \coloneqq cA(\boldsymbol{u}_T^{\beta}[k]) \cdots A(\boldsymbol{u}_{t+1}^{\beta}[k]),$$
$$\boldsymbol{x}_t^{\beta}[k] \coloneqq A(\boldsymbol{u}_{t-1}^{\beta}[k]) \cdots A(\boldsymbol{u}_1^{\beta}[k]) x_1.$$

The following algorithm thus implements MCMC Gibbs sampling with tempering:

Algorithm 3 (Tempering for Koopman optimal control):

1) Pick arbitrary $\boldsymbol{u}[1] = (\boldsymbol{u}_t^{\beta}[1] \in \mathcal{U} : t \in \mathcal{T}, \beta \in \mathcal{B})$.
2) Set $k = 1$ and repeat:
   a) *Gibbs sampling*: For each $\beta \in \mathcal{B}$:
      - Set $\boldsymbol{x}_1^{\beta}[k] = x_1$ and, $\forall t \in \{1,\dots,T\}$,
      $$\boldsymbol{c}_t^{\beta}[k] = cA(\boldsymbol{u}_T^{\beta}[k]) \cdots A(\boldsymbol{u}_t^{\beta}[k]). \tag{19}$$
      - *Variable sweep*: For each $t \in \{1,\dots,T\}$
        - Sample $\boldsymbol{u}_t^{\beta}[k+1]$ with distribution
        $$\frac{e^{-\beta \boldsymbol{c}_t^{\beta}[k] A(u_t) \boldsymbol{x}_t^{\beta}[k]}}{\sum_{\bar{u}_t} e^{-\beta e^{-\beta \boldsymbol{c}_t^{\beta}[k] A(\bar{u}_t) \boldsymbol{x}_t^{\beta}[k]}}}, \tag{20}$$
        - Set $\boldsymbol{u}^{\beta}_{-t}[k+1] = \boldsymbol{u}^{\beta}_{-t}[k]$, update
        $$\boldsymbol{x}_{t+1}^{\beta}[k+1] = A(\boldsymbol{u}_t^{\beta}[k+1]) \boldsymbol{x}_t^{\beta}[k], \tag{21}$$
        and increment k.
   b) *Tempering sweep*: For each $j \in \{1,\dots,M-1\}$
      - Compute the flip probability
      $$p_{\text{flip}} = \min\left\{e^{-(\beta_j-\beta_{j+1}) c(\boldsymbol{x}_T^{\beta_{j+1}}[k]) - \boldsymbol{x}_T^{\beta_j}[k]))}, 1\right\},$$
      and set
      $$\boldsymbol{u}[k+1] = \begin{cases} \tilde{\boldsymbol{u}}[k] & \text{with prob. } p_{\text{flip}} \\ \boldsymbol{u}[k] & \text{with prob. } 1 - p_{\text{flip}}, \end{cases}$$
      where $\tilde{\boldsymbol{u}}[k]$ is a version of $\boldsymbol{u}[k]$ with the entries $\boldsymbol{u}^{\beta_j}[k]$ and $\boldsymbol{u}^{\beta_{j+1}}[k]$ flipped; and increment k.
3) go back to 2 until enough samples have been collected. □

Computation complexity and parallelization: The bulk of the computation required by Algorithm 3 lies in the computation of the matrix-vector products that appear in (19), (20), and (21), each of these products requiring $O(n_\psi^2)$ floating-point operation for a total of $O(MT(2+|\mathcal{U}|)n_\psi^2)$ operations. The tempering step, does not use any additional vector-matrix multiplications. In contrast, a naif implementation of Gibbs sampling with tempering would require $MT|\mathcal{U}|$ evaluations of the cost function (18), each with computational complexity $O(Tn_\psi^2)$. The linearity of (18), enable us to re-use matrix-vector products across the variable sweep to reduce computation by a factor of $T|\mathcal{U}|/(2+|\mathcal{U}|)$, which is especially significant when the time horizon is large.

The computations of the matrix-vector products mentioned above are independent across different values of $\beta \in \mathcal{B}$ and can be performed in parallel. In essence, this means that tempering across M temperature parameters can be computationally very cheap if enough computational cores are available. In contrast, parallelization across a Gibbs variable sweep is not so easily parallelizable because, within one variable sweep, the value of each sample $\boldsymbol{u}_t^\beta[k+1]$, typically depends on the values of previous samples $\boldsymbol{u}_t^\beta[k]$ (for the same value of β). Tempering thus both decreases the mixing time of the Markov chain and opens the door for a high degree of parallelization.

V. Numerical Examples

We tested the algorithm proposed in this paper on a Koopman model for the Atari 2600 *Assault* video game. The goal of the game is to protect earth from small attack vessels deployed by an alien mothership. The mother ship and attack vessels shot at the player's ship and the player uses a joystick to dodge the incoming fire and fire back at the alien ships. The player's ship is destroyed either if it is hit by enemy fire or if it "overheats" by shooting at the aliens. The player earns points by destroying enemy ships.

We used a Koopman model with $|\mathcal{U}| = 4$ control action, which correspond to "move left", "move right", "shoot up", and do nothing. The state of the model is built directly from screen pixel information. Specifically, the pixels are segmented into 5 categories corresponding to the player's own ship, the player's horizontal fire, the player's vertical fire, the attacker's ships and fire, and the temperature bar. For the player's own ship and the attacker's ships/fire, we consider pixel information from the current and last screen-shot, so that we have "velocity" information. The pixels of each category are used to construct "spatial densities" using the entity-based approach described in [4]. The densities associated with the 5 groups of pixels listed above are represented by 50, 50, 100, 200, 16 points, respectively. All this pixel information, together with the cumulative points earned by the player, are used to construct the state ψ_t of the Koopman model with dimension $n_\psi = 667$. The optimization minimizes the following cost

$$J(u) := \begin{cases} -\frac{1}{T}\sum_{t=1}^{T} r_t & \text{if player's ship not destroyed} \\ +10 - \frac{1}{T}\sum_{t=1}^{T} r_t & \text{if player's ship destroyed} \end{cases}$$

where r_t denotes the points earned by the player at time t, as provided by the *Assault* simulator. This cost balances the tradeoff between taking some risk to collect points and to make the term $-\frac{1}{T}\sum_{t=1}^{T} r_t$ as negative as possible, while not getting destroyed and incurring the $+10$ penalty. For game play, this optimization is solved at every time step with a receding horizon starting at the current time t and ending at time $t+T$, with only the first control action executed. However, because the focus of this paper is on the solution to (1)–(2), we present results for a single-shot optimization starting from a typical initial condition.

Figure 1 depicts a typical run of Algorithm 3, showing constant flipping of solutions across adjacent temperatures, with a flip occurring in 40-60% of the tempering sweeps for most temperatures.

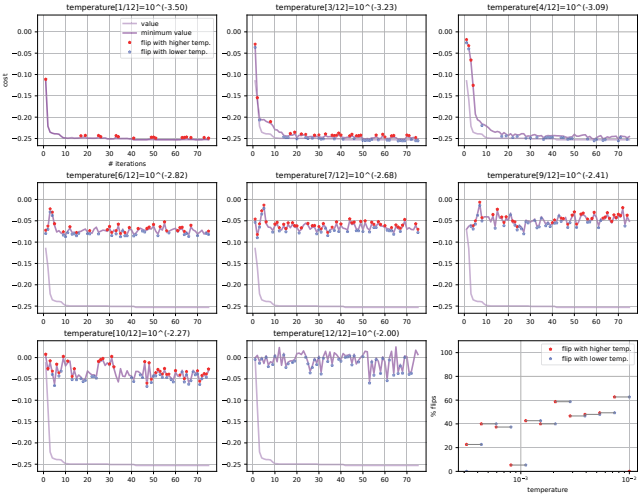

Fig. 1. Typical run of Algorithm 3 with horizon $T = 40$: Each of the first 8 plots shows the value of the cost (in the y-axis) at the end of each iteration (in the x-axis), as well as the minimum cost found so far. For this run, 12 logarithmic scaled temperatures were used, but only 8 are shown here. In each plot, red and blue dots mark iterations where flips occurred during the tempering sweep. The bottom-right plot shows the total number of flips across "adjacent" temperatures (in the x-axis), as a percentage of the total number of iterations (in the y-axis).

Figure 2 shows a comparison between Algorithm 3 and the dynamic programming algorithms in [4]. The latter algorithm makes use of the piecewise-linear structure of the cost-to-go to efficiently represent and evaluate the value function and also to dynamically prune the search tree. Due to the need for exploration, this algorithm "protects" from pruning a random fraction of tree-branches (see [4] for details). Run times refer to Julia implementations of both algorithms running on a 2018 MacBook Pro with a 2.6GHz 6-Core Intel Core i7 CPU. Both algorithms were allowed to use 6 CPU cores. For Algorithm 3, each core executed one sweep of the tempering algorithm and for the algorithm in [4], the 6 cores were used by BLAS to speedup matrix multiplication.

A time horizon $T = 40$ was used for the results in Figure 2, which would correspond to a total number of control options $|\mathcal{U}|^T \approx 1.2 \times 10^{24}$. Both algorithms were executed multiple times and the plots show the costs obtained as a function of run time. For Algorithm 3, the run time is directly controller by the number of samples drawn. For the algorithms in [4], the run time is controller by the number of vectors used to represent the (pruned) value function. The key observation from the plots in Figure 2 is that both algorithms eventually discover comparable "optimal" solutions, but Algorithm 3 can typically achieve a lower cost much faster. The fastest solutions are obtained by Algorithm 3 with 6 temperatures, which makes an almost optimal use of the 6 cores available, by devoting to each core a single temperature.

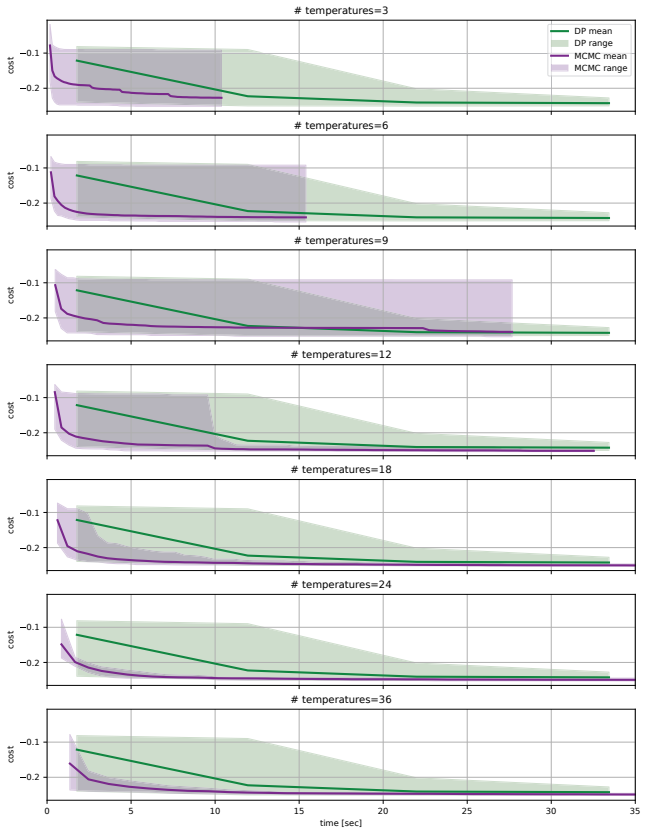

Fig. 2. Run-time comparison between Algorithm 3 (in purple) and the dynamic programming algorithms in [4] (green) with horizon $T = 40$. The x-axis denotes run-time and the y-axis the cost achieved, with the solid lines showing the average cost across 15 different runs and the shaded areas the whole range of costs obtained over those runs. The different plots correspond to different values for the number M of temperatures.

REFERENCES

[1] N. Aadit, A. Grimaldi, M. Carpentieri, L. Theogarajan, J. Martinis, G. Finocchio, and K. Y. Camsari. Massively parallel probabilistic computing with sparse Ising machines. *Nature Electronics*, 5(7):460–468, 2022.

[2] S. C. Bengea and R. A. DeCarlo. Optimal control of switching systems. *Automatica*, 41(1):11–27, 2005.

[3] P. Bevanda, S. Sosnowski, and S. Hirche. Koopman operator dynamical models: Learning, analysis and control. *Annual Reviews in Control*, 52:197–212, 2021.

[4] M. Blischke and J. P. Hespanha. Learning switched Koopman models for control of entity-based systems. In *Proc. of the 62th IEEE Conf. on Decision and Contr.*, Dec. 2023.

[5] S. Brunton, B. Brunton, J. Proctor, and J. N. Kutz. Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control. *PloS one*, 11(2), 2016.

[6] V. Černỳ. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *J. Opt. Theory and Applications*, 45:41–51, 1985.

[7] G. CJ. Markov chain Monte Carlo maximum likelihood. In *Computing Science and Statistics: Proc. of the 23rd symp. on the Interface*, pages 156–163, 1991.

[8] D. J. Earl and M. W. Deem. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910–3916, 2005.

[9] A. E. Gelfand and A. F. Smith. Sampling-based approaches to calculating marginal densities. *J. of the American Statistical Assoc.*, 85(410):398–409, 1990.

[10] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. on Pattern Anal. and Machine Intell.*, PAMI-6(6):721–741, 1984.

[11] M. Haseli and J. Cortés. Modeling nonlinear control systems via Koopman control family: Universal forms and subspace invariance proximity, 2023. arXiv: 2307.15368.

[12] J. P. Hespanha. Markov chain Monte Carlo for Koopman-based optimal control: Technical report. Technical report, University of California, Santa Barbara, Mar. 2024.

[13] K. Hukushima and K. Nemoto. Exchange Monte Carlo method and application to spin glass simulations. *J. of the Physical Society of Japan*, 65(6):1604–1608, 1996.

[14] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi. Optimization by simulated annealing. *science*, 220 (4598):671–680, 1983.

[15] B. O. Koopman. Hamiltonian systems and transformation in Hilbert space. *Proc. of the National Academy of Sciences U.S.A*, 17(5):315–318, 1931.

[16] M. Korda and I. Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, 2018.

[17] A. Mauroy, Y. Susuki, and I. Mezić. *Koopman operator in systems and control*. Springer, 2020.

[18] T. R. Mehta and M. Egerstedt. An optimal control approach to mode generation in hybrid systems. *Nonlinear Analysis: Theory, Methods & Applications*, 65 (5):963–983, 2006.

[19] I. Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41:309–325, 2005.

[20] N. Mohseni, P. L. McMahon, and T. Byrnes. Ising machines as hardware solvers of combinatorial optimization problems. *Nature Reviews Physics*, 4(6):363–379, 2022.

[21] S. E. Otto and C. W. Rowley. Koopman operators for estimation and control of dynamical systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:59–87, 2021.

[22] J. Proctor, S. Brunton, and J. N. Kutz. Generalizing Koopman theory to allow for inputs and control. *SIAM J. on Applied Dynamical Syst.*, 17(1):909–930, 2018.

[23] R. H. Swendsen and J.-S. Wang. Replica Monte Carlo simulation of spin-glasses. *Physical Review Lett.*, 57 (21):2607, 1986.

[24] X. Xu and P. Antsaklis. Optimal control of switched systems based on parameterization of the switching instants. *IEEE Trans. on Automat. Contr.*, 49(1):2–16, 2004.