2021 60th IEEE Conference on Decision and Control (CDC)
December 13-15, 2021. Austin, Texas

Data-driven Rollout for Deterministic Optimal Control

Yuchao Li, Karl H. Johansson, Jonas Mårtensson, Dimitri P. Bertsekas

2021 60th IEEE Conference on Decision and Control (CDC) | 978-1-6654-3659-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CDC45484.2021.9683499

***Abstract*— We consider deterministic infinite horizon optimal control problems with nonnegative stage costs. We draw inspiration from learning model predictive control scheme designed for continuous dynamics and iterative tasks, and propose a rollout algorithm that relies on sampled data generated by some base policy. The proposed algorithm is based on value and policy iteration ideas, and applies to deterministic problems with arbitrary state and control spaces, and arbitrary dynamics. It admits extensions to problems with trajectory constraints, and a multiagent structure.**

I. INTRODUCTION

Many practical challenges lend themselves to be formulated as deterministic infinite horizon optimal control problems. Those problems can be addressed in principle by dynamic programming (DP) methodology via value iteration (VI), and policy iteration (PI) algorithms. However, VI and PI are often limited by large problem size. Instead, one needs to resort to various approximate, sample-based schemes that are based on VI and PI ideas. Among them, rollout has stood out due to its simplicity and reliability. For problems with continuous components, model predictive control (MPC) is another major option. It relies on constrained optimization and has been developed independently of DP. In learning MPC (LMPC) one uses sampled trajectories for obtaining solutions, in a way that is similar to rollout. In this paper we explore these connections further, we develop a rollout algorithm that extends LMPC to arbitrary state and control spaces, system dynamics, and we discuss applications to multiagent problems.

We consider problems involving stationary dynamics

$$x_{k+1} = f(x_k, u_k), \quad k = 0, 1, \ldots, \tag{1}$$

where x_k and u_k are state and control at stage k, which belong to state and control spaces X and U, respectively, and f maps $X \times U$ to X. The control u_k must be chosen from a nonempty constraint set $U(x_k) \subset U$ that may depend on x_k. The cost of applying u_k at state x_k is denoted by $g(x_k, u_k)$, and is assumed to be nonnegative:

$$0 \leq g(x_k, u_k) \leq \infty, \quad x_k \in X,\, u_k \in U(x_k). \tag{2}$$

By allowing an infinite value of $g(x, u)$ we can implicitly introduce state constraints: a pair (x, u) is infeasible if $g(x, u) = \infty$. We consider feedback policies of the form $\{\mu, \mu, \ldots\}$, with μ being a function mapping X to U and satisfying $\mu(x) \in U(x)$ for all x. This type of polices are called *stationary*. When no confusion arises, with a slight abuse of notation, we also denote $\{\mu, \mu, \ldots\}$ as μ. There is no restriction on the state space X, and the control space U, so the problems addressed here are very general.

The *cost function* of a policy μ, denoted by J_μ, maps X to $[0, \infty]$, and is defined at any initial state $x_0 \in X$, as

$$J_\mu(x_0) = \sum_{k=0}^{\infty} g(x_k, \mu(x_k)), \tag{3}$$

where $x_{k+1} = f(x_k, \mu(x_k))$, $k = 0, 1, \ldots$. The optimal cost function J^* is defined pointwise as

$$J^*(x_0) = \inf_{\substack{u_k \in U(x_k),\ k=0,1,\ldots \\ x_{k+1}=f(x_k,u_k),\ k=0,1,\ldots}} \sum_{k=0}^{\infty} g(x_k, u_k). \tag{4}$$

A stationary policy μ^* is called optimal if

$$J_{\mu^*}(x) = J^*(x), \quad \forall x \in X.$$

In the context of DP, one hopes to obtain a stationary optimal policy μ^*. Two exact DP algorithms for addressing the problem are VI and PI. However, these algorithms may be intractable, due to the *curse of dimensionality*, which refers to the explosion of the computation as the cardinality of X increases. Thus, various approximation schemes have been proposed, where one aims to obtain some suboptimal policy $\tilde{\mu}$ such that $J_{\tilde{\mu}} \approx J^*$.

One such scheme is MPC, which is widely applied to problems with continuous state and control spaces, stage-decoupled state and control constraints, and continuous dynamics, without requiring continuous state and control space discretization. For a given state constraint set, denoted by C, and at any $x \in C$, the MPC algorithm solves the constrained optimization problem

$$\min_{\{u_k\}_{k=0}^{\ell-1}} \quad \sum_{k=0}^{\ell-1} g(x_k, u_k) + G(x_\ell) \tag{5a}$$

$$\text{s. t.} \quad x_{k+1} = f(x_k, u_k),\ k = 0, ..., \ell - 1, \tag{5b}$$

$$x_k \in C,\ u_k \in U(x_k),\ k = 0, ..., \ell - 1, \tag{5c}$$

$$x_0 = x. \tag{5d}$$

The function G is some real-valued function, which is often designed as an approximation to the cost function of a certain policy, as given in (3), while fulfilling additional requirements that we will discuss later. With the optimal attained at $(\tilde{u}_0, \tilde{u}_1, \ldots, \tilde{u}_{\ell-1})$, the policy $\tilde{\mu}$ applied to the

This work was supported by the Swedish Foundation for Strategic Research, the Swedish Research Council, and the Knut and Alice Wallenberg Foundation.

Y. Li, K. H. Johansson, and J. Mårtensson are with the division of Decision and Control Systems, KTH Royal Institute of Technology, Sweden, yuchao,jonas1,kallej@kth.se

D. P. Bertsekas is Fulton Professor of Computational Decision Making, ASU, Tempe, AZ, and McAfee Professor of Engineering, MIT, Cambridge, MA, dbertsek@asu.edu, dimitrib@mit.edu

system is given by setting $\tilde{\mu}(x) = \tilde{u}_0$. The above procedure is repeated to select the control at each stage.

There are two challenges in the MPC scheme introduced above. First, investigating conditions under which the MPC problem (5) is feasible. Second, designing the function G so that the resulting system is stable. Those two challenges are often addressed independently, with independent lines of analysis for feasibility and stability. With abundant capability to generate data, there have been many proposals of MPC variants that provide guarantee of both feasibility and stability while taking advantage of data. A notable approach, relevant to our study, is a scheme known as LMPC introduced in [1]. It is designed for problems with continuous state and control spaces, continuous dynamics, and problems where the same initial x_0 is repeatedly visited. Its analysis follows the classic lines of arguments, where the above challenges are investigated independently.

When going beyond the scope addressed by MPC, a reliable and simple methodology is *rollout*, which is couched on rigorous VI and PI based analysis. It requires availability of a policy μ^0, referred to as a *base policy*, and at a given state x_0, it computes the values of J_{μ^0} for all subsequent states of interests. At any state, it solves an optimization problem similar to (5), with J_{μ^0} in place of G (see Section III for a precise definition). This is referred to as rollout with ℓ-step lookahead. The policy given by the computation is referred to as the *rollout policy*. When $\ell = 1$, rollout is one step of the PI algorithm applied to x_0. Rollout and MPC have clear similarities, and it is not surprising that they are closely related. Still, the analysis of rollout revolves around a concept known as *sequential improvement condition*, introduced in [2] and [3],

$$\inf_{u \in U(x)} \big\{ g(x,u) + J_\mu\big(f(x,u)\big) \big\} \leq J_\mu(x), \tag{6}$$

and the constraints considered may be different in nature to the ones considered in MPC, such as constraints imposed on entire trajectories. While rollout is a less ambitious method than the VI and PI algorithms, the rollout policy $\tilde{\mu}$ is guaranteed to outperform the base policy μ^0. In fact, rollout can be viewed as one step of Newton's method for solving a fixed point equation regarding J^*, as explained in [4]. Since Newton's method has a superlinear convergence property, the performance improvement achieved by the policy $\tilde{\mu}$ is often dramatic and sufficient for practical purposes.

In this paper, we draw inspiration from LMPC and propose a new variant of rollout. It uses sampled trajectories, and requires access to $J_{\mu^0}(x_\ell)$ for only some x_ℓ reached from x_0 in ℓ steps, instead of all such x_ℓ as is required by traditional rollout with ℓ step lookahead. The states x for which the values $J_{\mu^0}(x)$ are available form a subset $S^0 \subset X$. This set is assumed to have the following invariance property under the base policy μ^0:

$$f\big(x, \mu^0(x)\big) \in S^0, \quad \forall x \in S^0. \tag{7}$$

With such a set S^0, we will show that the cost improvement property of the rollout algorithm remains intact. One prime case where such a set S^0 can be constructed is for the problem with a nonempty stopping set $X_s \subset X$, which consists of cost-free and forward invariant states in the sense that

$$g(x,u) = 0, \quad f(x,u) \in X_s, \quad \forall x \in X_s,\, u \in U(x).$$

If we can use the base policy μ^0 to generate a trajectory $\{x_0, \mu^0(x_0), x_1, \mu^0(x_1), \ldots, x_j\}$ that starts at some initial state x_0 and ends at a stopping state $x_j \in X_s$, the values $J_{\mu^0}(x)$ for all $x = x_0, \ldots, x_j$ can be computed.

By introducing the rollout viewpoint, we extend the LMPC scheme to problems with arbitrary state and control spaces, arbitrary system dynamics and nonnegative costs. We show that the method admits extensions to problems where there is a trajectory constraint. When the base policy μ^0 is available for all $x \in S^0$, it allows distributed computation, which is useful for multiagent problems. For yet another extension, if multiple such subsets S^i for multiple different policies μ^i are available, they can be combined together to form a set S and the performance may be further improved. The contributions of our proposed scheme, are as follows:

(a) A rollout algorithm that provides cost improvement over a base policy, for arbitrary state and control spaces, and system dynamics with nonnegative stage costs.
(b) An extension to trajectory constraints and the sequential improvement theoretical framework that goes with it.
(c) Validation of the methodology using examples ranging from hybrid systems, trajectory constrained problems, multiagent problems, and discrete and combinatorial optimization.

The paper is organized as follows. In Section II we provide background and references, which place in context our results in relation to the literature. In Section III, we describe our rollout algorithm and its variants, and we provide analysis. In Section IV, we demonstrate the validity of our algorithm through examples.

II. Background

The study of nonnegative cost optimal control with an infinite horizon dates back to the thesis and paper [5], following the seminal earlier research of [6], [7]. Owing to its broad applicability, the followup research has been voluminous and comprehensive, with extensive results provided in the book [8]; see [9] for a more accessible discussion.

Among those results, if the minima in the relevant optimization are assumed to be attained, our analysis here relies on only two classical results, which hold well beyond the scope of our study. This is one key contributing factor to the wide applicability of rollout in general and our algorithm in particular. These two results are parts (a) and (b) of the following proposition.

Proposition 2.1: Let the nonnegativity condition (2) hold.

(a) For all stationary policies μ we have

$$J_\mu(x) = g\big(x, \mu(x)\big) + J_\mu\big(f\big(x, \mu(x)\big)\big), \, \forall x \in X. \tag{8}$$

(b) For all stationary policies μ, if a nonnegative function $\bar{J}: X \to [0,\infty]$ satisfies

$$g\big(x,\mu(x)\big) + \bar{J}\big(f\big(x,\mu(x)\big)\big) \leq \bar{J}(x),\ \forall x \in X,$$

then $\bar{J}$ is an upper bound of J_μ, e.g., $J_\mu(x) \leq \bar{J}(x)$ for all $x \in X$.

Prop. 2.1(a) can be found in [9, Prop. 4.1.2]. Prop. 2.1(b) can be found in [9, Prop. 4.1.4(a)], and is established through Prop. 2.1(a) and the following monotonicity property: For all nonnegative functions J and J' that map X to $[0,\infty]$, if $J(x) \leq J'(x)$ for all $x \in X$, then for all $x \in X$, $u \in U(x)$, we have

$$g(x,u) + J\big(f(x,u)\big) \leq g(x,u) + J'\big(f(x,u)\big). \quad (9)$$

It has long been recognized that the application of the exact forms of VI and PI algorithms are computationally prohibitive. This has motivated suboptimal but less computationally intensive schemes, as noted by Bellman in his classical book [10]. In this regard, approximation schemes patterned after the VI and PI algorithms have had great success, and the rollout methodology has stood out thanks to its simple form and reliable performance.

Proposed first in [11] for addressing backgammon, rollout has since been modified and extended for combinatorial optimization [2], stochastic scheduling [12], vehicle routing [13], and more recently, Bayesian optimization [14], [15]. It admits variants that are suitable for problems with multiagent structure (suggested first in [16, Section 6.1.4] and investigated thoroughly in [17]), and problems with trajectory constraints [18]. It underlies the success of AlphaZero and related high-profile successes in the context of games, when combined with Monte-Carlo simulation and neural networks [19], [11].

Even with all those extensions, the existing forms of rollout do not address the cases where the values J_{μ^0} are available only in some selected states of X, not all of them. Situations of this type arise when obtaining μ^0 is a challenge itself and can only be stored for some selective states, as can be the case for the trajectory constrained problems that we consider in this paper. The interest in this type of settings is justified in view of the abundant availability of data, which could be historical trajectories collected and stored off-line. In this work, we address such problems and show that as long as the values of J_{μ^0} are available over a subset that has an invariance property like (7), a nearly identical analysis to the existing rollout variants is possible and the cost improvement condition remains valid.

MPC is a major methodology that primarily applies to optimal control problems with state and control consisting mainly of continuous components and stage-decoupled constraints. Developed independently from DP [20], MPC was recognized as a moving horizon approximation scheme for solving infinite horizon problems in some early works, e.g., [21]. MPC is also closely connected to rollout, as was pointed out in [3]: it can be viewed as rollout with one step lookahead with the base policy μ^0 involving $(\ell-1)$-step optimization (lookahead steps in our discussion is referred as predicting horizon in MPC literature). The central issues in MPC when applied to deterministic problems are the feasibility of the on-line numerical implementation, and the stability of the resulting closed-loop system. The stability analysis often revolves around a Lyapunov function that is bounded in certain form by a continuous function [22, Theorem 7.2], [23, Theorem B.13] when the dynamics is not continuous, and is also related to the sequential improvement condition (6), which is central to the cost improvement property of rollout. The feasibility issues are addressed separately, and revolve around the concept of reachability [24]. In particular, one goal of such analysis is to establish a property known as *recursive feasibility*. It means that the feasibility of the MPC problem at current state implies its feasibility at the subsequent state, driven by the current MPC control. As opposed to these approaches, we will show that a line of analysis centered around the fixed point equation (8) and performance improvement of PI can serve as an effective substitute for feasibility and stability analysis. For this to hold, we require that the cost function values $J_{\mu^0}(x)$ are known for a subset of states x that has the property (7), but place no continuity-type restrictions.

Our MPC variant that involves final cost and constraints, and sampled trajectories, coincides with the exact form of LMPC [1] when applied to problems with continuous dynamics and time-decoupled state and control constraints. Still, our work generalizes substantially LMPC. It can deal with problems with trajectory constraints, and discontinuous dynamics (like the ones involving hybrid systems), and admits multiagent variants, as will be discussed shortly.

III. Main results

In this section, we will provide details on the data-driven rollout methods and their analysis. We will first introduce the basic form of the rollout algorithm that deals with unconstrained-trajectory problems. It requires a base policy μ^0 and its corresponding cost function over a subset of the state space. We will then extend the method to handle a general set constraint on the state-control trajectories. The basic form will also be extended to incorporate multiple base policies. A variant of the basic form will be introduced in the end where the minimization steps are simplified. This variant is well-suited for multiagent applications.

Throughout our discussion of the first three variants, we make the following assumption. For the last variant, we make a slightly different assumption.

Assumption 3.1: The cost nonnegativity condition (2) holds. Moreover, for all functions J that map X to $[0,\infty]$, the minimum in the following optimization

$$\inf_{u \in U(x)} \big\{ g(x,u) + J\big(f(x,u)\big) \big\}$$

is attained for all $x \in X$.

A. Basic form of data-driven rollout

Consider the optimal control problem with dynamics (1) and stage cost (2). We assume that we have computed off-line a subset S^0 of the state space, and the cost function

values $J_\mu^0(x)$ for all $x \in S^0$ of a stationary policy μ^0. The subset and the stationary policy have the property

$$f\big(x, \mu^0(x)\big) \in S^0, \quad \forall x \in S^0. \tag{10}$$

We define $\bar{J}_{S^0} : X \to [0, \infty]$ as

$$\bar{J}_{S^0}(x) = J_{\mu^0}(x) + \delta_{S^0}(x),$$

where $\delta_{S^0} : X \to \{0, \infty\}$ is the indicator function of set S^0 defined as 0 if $x \in S^0$ and ∞ otherwise. For a given state x, the rollout algorithm solves the following problem

$$\min_{\{u_k\}_{k=0}^{\ell-1}} \quad \sum_{k=0}^{\ell-1} g(x_k, u_k) + \bar{J}_{S^0}(x_\ell) \tag{11a}$$

$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k),\ k = 0, ..., \ell - 1, \tag{11b}$$

$$u_k \in U(x_k),\ k = 0, ..., \ell - 1, \tag{11c}$$

$$x_0 = x. \tag{11d}$$

The optimal value of the problem is denoted as $\tilde{J}_{S^0}(x)$ (with subscript indicating the dependence on the set S^0), and the minimizing sequence is denoted as $(\tilde{u}_0, \tilde{u}_1, \ldots, \tilde{u}_{\ell-1})$. The rollout policy $\tilde{\mu}$ is defined by setting $\tilde{\mu}(x) = \tilde{u}_0$.

When the state and control spaces are continuous and under time-decoupled constraints, and the set S^0 contains m sample points, the solution of the problem (11) requires solving m continuous optimization problems, with each of the m sample points used as a fixed terminal state. The infinite stage costs $g(x, u) = \infty$ imply constraints imposed on state and control pairs. In this case, our rollout algorithm coincides with the LMPC given in [1], where more discussions on computational issues can be found. On the other hand, when trajectory constraints are present, the set S^0 may introduce additional constraints; cf. Example 4.2.

The preceding rollout algorithm may be viewed as a generalization of the originally proposed MPC algorithm, whereby we first drive the state to 0 after ℓ steps ($\{0\}$ is the set S^0 in our algorithm), and then use the policy that keeps the state at 0 (this is the policy μ^0 in our algorithm). For our rollout policy, we have the following property hold.

Proposition 3.1: Let Assumption 3.1 hold. For the rollout policy $\tilde{\mu}$ defined by (11) with set S^0 satisfying the invariance property (10) under the base policy μ^0, we have

$$J_{\tilde{\mu}}(x) \le \tilde{J}_{S^0}(x) \le \bar{J}_{S^0}(x), \quad \forall x \in X. \tag{12}$$

Proof: Using a principle of optimality argument (see, e.g., [25, Section 1.3]), the optimization problem (11) can be viewed as performing ℓ VI iterations with $J_0 = \bar{J}_{S^0}$, generating $\{J_k\}_{k=0}^{\ell}$ according to

$$J_{k+1}(x) = \min_{u \in U(x)} \big\{ g(x, u) + J_k\big(f(x, u)\big) \big\}, \tag{13}$$

and with the rollout policy at x given by

$$\tilde{\mu}(x) \in \arg\min_{u \in U(x)} \big\{ g(x, u) + J_{\ell-1}\big(f(x, u)\big) \big\}, \tag{14}$$

with $\tilde{J}_{S^0}(x) = J_\ell(x)$. With such a view, we will first show that the sequential improvement condition holds, based upon which the finite sequence $\{J_k\}_{k=0}^{\ell}$ will be shown to be monotonically decreasing. Finally, by applying Prop. 2.1(b), the monotonicity property (9) and Prop. 2.1(a) in this order, we can prove the desired inequality (12).

By regarding rollout as ℓ-step VI, we first note that for all $x \notin S^0$, $J_1(x) \le J_0(x) = \bar{J}_{S^0}(x) = \infty$. For $x \in S^0$, we have $J_0(x) = \bar{J}_{S^0}(x) = J_{\mu^0}(x)$ and $\bar{J}_{S^0}\big(f\big(x, \mu(x)\big)\big) = J_{\mu^0}\big(f\big(x, \mu(x)\big)\big)$, as per (10). Then we have that

$$\begin{aligned} J_1(x) = &\min_{u \in U(x)} \big\{ g(x, u) + \bar{J}_{S^0}\big(f(x, u)\big) \big\} \\ \le &g\big(x, \mu^0(x)\big) + \bar{J}_{S^0}\big(f\big(x, \mu^0(x)\big)\big) \\ = &g\big(x, \mu^0(x)\big) + J_{\mu^0}\big(f\big(x, \mu^0(x)\big)\big) = J_0(x), \end{aligned} \tag{15}$$

where the first equality is per (13), the second equality is due to $\bar{J}_{S^0}\big(f\big(x, \mu^0(x)\big)\big) = J_{\mu^0}\big(f\big(x, \mu^0(x)\big)\big)$ for $x \in S^0$, and the last equality is due to that the fixed point equation holds for μ^0, as per Prop. 2.1(a), and $J_0(x) = J_{\mu^0}(x)$ for $x \in S^0$.

With $J_1(x) \le J_0(x)$, we apply induction and assume that $J_k(x) \le J_{k-1}(x)$ for all x. Then we have

$$\begin{aligned} J_{k+1}(x) = &\min_{u \in U(x)} \big\{ g(x, u) + J_k\big(f(x, u)\big) \big\} \\ \le &\min_{u \in U(x)} \big\{ g(x, u) + J_{k-1}\big(f(x, u)\big) \big\} = J_k(x). \end{aligned}$$

Therefore, we have that $\tilde{J}_{S^0}(x) = J_\ell(x) \le J_{\ell-1}(x) \le J_0(x) = \bar{J}_{S^0}(x)$. Since the rollout policy $\tilde{\mu}(x)$ is defined by (14), we then have

$$g\big(x, \tilde{\mu}(x)\big) + J_{\ell-1}\big(f\big(x, \tilde{\mu}(x)\big)\big) = J_\ell(x) \le J_{\ell-1}(x) \tag{16}$$

for all $x \in X$. Applying Prop. 2.1(b) with $\bar{J}$ as $J_{\ell-1}$ here, we have $J_{\tilde{\mu}}(x) \le J_{\ell-1}(x)$. By the monotonicity property (9), we have

$$\begin{aligned} J_{\tilde{\mu}}(x) = &g\big(x, \tilde{\mu}(x)\big) + J_{\tilde{\mu}}\big(f\big(x, \tilde{\mu}(x)\big)\big) \\ \le &g\big(x, \tilde{\mu}(x)\big) + J_{\ell-1}\big(f\big(x, \tilde{\mu}(x)\big)\big) = J_\ell(x), \end{aligned}$$

where the first equality is due to Prop. 2.1(a), the last equality is due to the definition of $\tilde{\mu}$. The proof is thus complete. ■

Remark 3.1: The property (10) for the set S^0 is called forward invariance (or positive invariance) in the MPC literature. However, such a set is easy to obtain by recording a trajectory, as opposed to conventional forward invariant set where major computation may be needed. Still, our analysis applies to these settings, where S^0 may be an ellipsoid or polyhedron, provided that the values of $J_{\mu^0}(x)$ are known for all $x \in S^0$.

Remark 3.2: The recursive feasibility of the optimization (11) is implied by sequential improvement (15). To see this, for a given x, we denote by x' its subsequent state under rollout, e.g., $x' = f\big(x, \tilde{\mu}(x)\big)$. The condition $\tilde{J}_{S^0}(x) = g\big(x, \tilde{\mu}(x)\big) + J_{\ell-1}(x') < \infty$ implies $J_{\ell-1}(x') < \infty$. Since sequential improvement implies (16), we have $\tilde{J}_{S^0}(x') = J_\ell(x') \le J_{\ell-1}(x') < \infty$. In other words, due to the construction of our algorithm and nonnegativity of stage costs, the function $\tilde{J}_{S^0}$ is monotonically decreasing along the trajectory under the rollout policy $\tilde{\mu}$.

Remark 3.3: The stability of the closed-loop system obtained by the rollout algorithm should be addressed independently of the cost improvement property. Note that the cost function of $\tilde{\mu}$ is bounded by $\tilde{J}_{S^0}$. Thus, starting with some x such that $\tilde{J}_{S^0}(x) < \infty$, the resulting trajectory is guaranteed to have a finite cost, which essentially guarantees stability.

Remark 3.4: We note that the inequality (12) holds for all $x \in X$, apart from the elements in S^0. This means the rollout method has inherited a robustness property in the MPC context, in the sense that if the subsequent state $\tilde{x}'$ is different from the anticipated next state x', possibly due to external disturbance, one only needs to ensure $\tilde{J}_{S^0}(\tilde{x}) < \infty$ (initial feasibility) in order to achieve finite trajectory cost.

An important aspect in our rollout algorithm is the choice of the set S^0. It is natural to speculate that with increased size of S^0, better performance may be obtained. Our analysis partially confirms this intuition, in the sense that a performance bound, rather than the policy cost function, is improved, as we will show shortly.

Let S^0 and $\bar{S}^0$ denote two sets satisfying the invariance property (10) with respect to a common base policy μ^0, and S^0 is a subset of $\bar{S}^0$. Let $\tilde{J}_{S^0}(x)$ and $\tilde{J}_{\bar{S}^0}(x)$ denote the optimal values of the problem (11) when $\bar{J}_{S^0}$ and $\bar{J}_{\bar{S}^0}$ are applied as the final costs respectively. Since $S^0 \subset \bar{S}^0$, we have $\delta_{\bar{S}^0} \leq \delta_{S^0}$, which implies that $\bar{J}_{\bar{S}^0} \leq \bar{J}_{S^0}$. By applying the monotonicity property (9), one can show that the VI sequence generated by starting with $\bar{J}_{\bar{S}^0}$ is always upper bounded by the one starting with $\bar{J}_{S^0}$, which leads to $\tilde{J}_{\bar{S}^0} \leq \tilde{J}_{S^0}$. We state the result just described as the following proposition.

Proposition 3.2: Let Assumption 3.1 hold, S^0 and $\bar{S}^0$ be two sets satisfying the invariance property (10) under the common base policy μ^0, and $S^0 \subset \bar{S}_0$. Denote by $\tilde{J}_{S^0}(x)$ and $\tilde{J}_{\bar{S}^0}(x)$ the optimal values of the problem (11) when $\bar{J}_{S^0}$ and $\bar{J}_{\bar{S}^0}$ are applied as the final costs respectively. Then we have $\tilde{J}_{\bar{S}^0}(x) \leq \tilde{J}_{S^0}(x)$, $\forall x \in X$.

B. Trajectory constrained rollout

We now provide a variant of our rollout algorithm that handles the constraint imposed on entire trajectories. In particular, a complete state-control trajectory $\{(x_k, u_k)\}_{k=0}^{\infty}$ is feasible if it belongs to a set C_∞, e.g., $\{(x_k, u_k)\}_{k=0}^{\infty} \in C_\infty$. Since our rollout algorithm applies to problems with arbitrary state and control spaces, we can transform the trajectory-constrained problem to an unconstrained one by state augmentation.

To this end, we introduce an information state e_k from some set E. Given a partial trajectory $\{(x_i, u_i)\}_{i=0}^{k}$, its information state can be specified through some suitably defined function $I(\cdot)$ as $e_k = I\big(\{(x_i, u_i)\}_{i=0}^{k}\big)$. The information state subsumes the information contained in the partial trajectory that is relevant to the trajectory constraint C_∞ in the sense that for every k, $\{(x_i, u_i)\}_{i=0}^{\infty}$ is feasible if and only if

$$\{e_k\} \cup \{(x_i, u_i)\}_{i=k+1}^{\infty} \in C_\infty.$$

Such an information state always exists. In the extreme case, it could be simply $e_k = \{(x_i, u_i)\}_{i=0}^{k}$. Accordingly, the dynamics of e_k can be defined through some function $h : E \times X \times U \to E$. Thus, via state augmentation, we obtain a trajectory-unconstrained problem. Its state is (x_k, e_k), while its control is still u_k. The constraints imposed on the state (x_k, e_k) is specified implicitly through C_∞. More details on state augmentation and dealing with trajectory constraints can be found in, e.g., [25, Sec. 1.4], [26, Sec. 3.3].

Given a feasible sample trajectory $\{(x_k, u_k)\}_{k=0}^{\infty}$ obtained under some policy μ^0, we can construct the sample set S^0 for the corresponding trajectory-unconstrained problem as

$$S^0 = \cup_{k=0}^{\infty}\big\{(x, e) \,\big|\, x = x_k,\, \{e\} \cup \{(x_i, u_i)\}_{i=k+1}^{\infty} \in C_\infty\big\}.$$

Thus, the algorithm of the basic form applies and its results hold for this transformed problems as well.

C. Rollout with multiple policies

We now consider the case where there are multiple subsets S^i that have the property (10) with respect to the corresponding policies μ^i. Let $\mathcal{I}$ denote the collection of indices of these sets. We define $\bar{J}_S : X \to [0, \infty]$ as

$$\bar{J}_S(x) = \inf_{i \in \mathcal{I}_x} \bar{J}_{S^i}(x), \tag{17}$$

where $\mathcal{I}_x = \{i \in \mathcal{I} \,|\, x \in S^i\}$, with the understanding that $\inf \emptyset = \infty$, so that $S = \cup_{i \in \mathcal{I}} S^i$. In addition, if $x \in S$, we denote as $i_x^* \in \arg\min_{\mathcal{I}_x} J_{\mu^i}(x)$, and we have that

$$\begin{aligned}
&\min_{u \in U(x)} \big\{g(x, u) + \bar{J}_S\big(f(x, u)\big)\big\} \\
\leq &g\big(x, \mu^{i_x^*}(x)\big) + \bar{J}_S\big(f\big(x, \mu^{i_x^*}(x)\big)\big) \\
\leq &g\big(x, \mu^{i_x^*}(x)\big) + J_{\mu^{i_x^*}}\big(f\big(x, \mu^{i_x^*}(x)\big)\big) = \bar{J}_S(x),
\end{aligned} \tag{18}$$

where the second inequality is due to (17). To see this, for $x \in S$ with $i_x^* \in \arg\min_{\mathcal{I}_x} J_{\mu^i}(x)$, we have $f\big(x, \mu^{i_x^*}(x)\big) \in S^{i_x^*}$ as is required by the property (10). Then by applying the basic form of rollout (12) with $\bar{J}_{S^0}$ replaced by $\bar{J}_S$, we have the following result hold. The proof is similar of the previous one by replacing (15) with (18), and is thus neglected.

Proposition 3.3: Let Assumption 3.1 hold. For the rollout policy $\tilde{\mu}$ defined by (11) with $\bar{J}_{S^0}$ replaced by $\bar{J}_S$ given in (17), sets S^i, $i \in \mathcal{I}$, satisfying the property (10) with respect to μ^i, and optimal value of the problem (11) denoted as $\tilde{J}_S$, we have $J_{\tilde{\mu}}(x) \leq \tilde{J}_S(x) \leq \bar{J}_S(x)$, $\forall x \in X$.

Remark 3.5: If the set S is replaced by some set $\bar{S} = \cup_{i \in \mathcal{I}} \bar{S}^i$, where the sets $\bar{S}^i$, $i \in \mathcal{I}$, have the invariance property (10) under μ^i, and $S^i \subset \bar{S}^i$, a result on performance bound improvement, similar to Prop. 3.2, can be established.

D. Simplified rollout

For yet another variant, we consider again the same problem as in the basic form case. This time, however, given a set S^0 satisfying the property (10), we assume that we have constructed nonempty control constraints $\bar{U}(x) \subset U(x)$ such that $\mu^0(x) \in \bar{U}(x)$ and the minimum is attained in the relevant minimization. We formalize the scheme by first introducing the following assumption.

Assumption 3.2: The cost nonnegativity condition (2) holds. The set S^0 fulfills the property (10) under the base

policy μ^0. Moreover, for all functions J that map X to $[0,\infty]$, the minimum of the following optimization

$$\inf_{u\in\bar{U}(x)} \big\{g(x,u)+J\big(f(x,u)\big)\big\}$$

is attained for all $x\in X$, where $\bar{U}(x)\subset U(x)$ are nonempty subsets such that $\mu^0(x)\in\bar{U}(x)$ for all $x\in S^0$.

With Assumption 3.2, we introduce a variant of the basic rollout form, which we call simplified rollout. The simplification is achieved by replacing the constraint $u_k\in U(x_k)$ in (11c) with $u_k\in\bar{U}(x_k)$, $k=0,...,\ell-1$, while keeping the other part of the algorithm unchanged. This method is relevant in a multiagent context, where for example the control u is composed of two parts $u=(v,w)$ that are under the disposal of two agents. At a given state x, the base policy μ^0 assigns the values v^0 and w^0 to those agents. Then the minimization

$$\min_{(v,w)\in U(x)} \big\{g\big(x,(v,w)\big)+J\big(f\big(x,(v,w)\big)\big)\big\}$$

may be replaced by first solving

$$\min_{(v,w)\in U(x)} \big\{g\big(x,(v,w)\big)+J\big(f\big(x,(v,w)\big)\big)\big\},\ \text{s.t.}\ w=w^0,$$

with minimum attained at $(\tilde{v},w^0)$. This step is followed by

$$\min_{(v,w)\in U(x)} \big\{g\big(x,(v,w)\big)+J\big(f\big(x,(v,w)\big)\big)\big\},\ \text{s.t.}\ v=\tilde{v}.$$

In doing so, a constraint set $\bar{U}(x)$ is implicitly constructed, cf. Fig. 1. This scheme can be extended to the case of multiple agents, and with further modifications, can be made suitable for distributed computation (see [17], [26]). We state without proof the performance guarantee that is similar to the previous basic form.

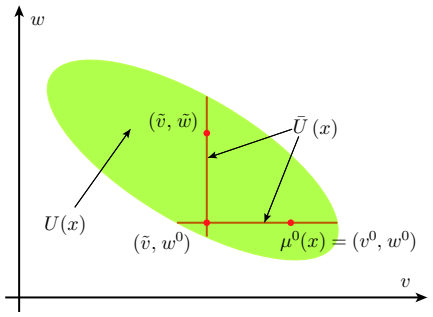

Fig. 1. Construction of the set $\bar{U}(x)$. The original control constraint set $U(x)$ is the green ellipsoid. Starting from the horizontal line segment contained in the set $U(x)$ with $w=w^0$, the first agent optimizes its control and attains the optimum at $(\tilde{v},w^0)$. The second agent then searches on the line segment contained in the set $U(x)$ with $v=\tilde{v}$. Thus, the implicitly constructed set $\bar{U}(x)$ is the union of those two line segments.

Proposition 3.4: Let Assumption 3.2 hold. For the rollout policy $\tilde{\mu}$ defined by (11) with the constraint $u_k\in U(x_k)$ in (11c) replaced by $u_k\in\bar{U}(x_k)$ so that $\mu^0(x_k)\in\bar{U}(x_k)$, $k=0,...,\ell-1$, we have $J_{\tilde{\mu}}(x)\le\tilde{J}_{S^0}(x)\le\bar{J}_{S^0}(x)$, $\forall x\in X$.

IV. Illustrating Examples

In this section, we illustrate through examples the validity of our main results. Our first example is from [27] and involves discontinuous dynamics. In the second example, we illustrate a state augmentation technique, as applied to a trajectory-constrained problem modified from [1, Example A]. The third example involves a collision avoidance situation, where simplified rollout is applied. In the last example, we consider a variant of a traveling salesman example, adapted from [26, Example 1.3.1]. It demonstrates the conceptual equivalence between the rollout with multiple heuristics and LMPC [1]. The numerical examples 4.1, 4.2 and 4.3 are solved in MATLAB with toolbox YALMIP [28] and solver `fmincon`.

With our examples, we will aim to elucidate the following points:

(1) The performance improvement of rollout is valid regardless of the nature of the state and control spaces.
(2) The proposed algorithm can be used for problems with discontinuous dynamics, consistent with the theory (cf. Example 4.1).
(3) The rollout algorithms are applicable to trajectory-constrained problems once suitable state augmentation is carried out (cf. Example 4.2).
(4) The simplified rollout enables distributed computation and applies to multiagent problems (cf. Example 4.3).
(5) Further performance improvement may be obtained via applying multiple heuristics and/or enlarging the sample sets (cf. Example 4.4).

Example 4.1: (Discontinuous dynamics) In this example, we apply the basic form of rollout to a problem involving a hybrid system. For this type of problems, conventional analytic approaches for MPC require substantial modifications, while the rollout viewpoint we take applies to such problems without customization. The dynamics is given as

$$f(x,u)=0.8\begin{bmatrix}\cos(\beta(x)) & -\sin(\beta(x))\\ \sin(\beta(x)) & \cos(\beta(x))\end{bmatrix}x+\begin{bmatrix}0\\1\end{bmatrix}u,$$

where $X=[-10,10]^2\subset\Re^2$, $U=[-1,1]\subset\Re$, where $\Re^n$ denotes the n-dimensional Euclidean space. The function $\beta(\cdot)$ is defined as $\frac{\pi}{3}$ if $[1\ 0]x\ge 0$ and $-\frac{\pi}{3}$ otherwise. The stage cost is defined as $g(x,u)=x^{\mathrm{T}}x$. The discountinous dynamics is modelled by using binary variables, as in [27, Example 4.1]. Since the system is stable when the control is set to zero, we use as initial feasible solution $\mu^0(x)=0$ for all x. The final stage cost $\bar{J}_{S^0}$ of rollout is computed accordingly. We compute the trajectory costs of the base policy μ^0, the rollout policy $\tilde{\mu}$ with $\ell=5$, and a classical MPC policy with horizon $\ell=10$, starting from initial states $(1,1)$ and $(8,-9)$ respectively. The corresponding costs are listed in Table I. It can be seen the cost improvement property indeed holds.

Example 4.2: (Trajectory-constrained problem) We consider a trajectory-constrained problem, and demonstrate the use of state augmentation. The system dynamics is given as

TABLE I

TRAJECTORY COSTS UNDER DIFFERENT POLICIES IN EXAMPLE 4.1.

Initial state	J_{μ^0}	$J_{\tilde{\mu}}$	MPC cost
$[1\,1]^{\mathrm{T}}$	5.5555	5.0162	7.2811
$[8\ -9]^{\mathrm{T}}$	402.7778	318.9486	355.7865

$f(x,u) = Ax + Bu$, where

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix},\ B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

The stage cost is given as $g(x,u) = x^{\mathrm{T}}x + u^2 + \delta_C(x)$ with constraint set $C = [-4,4]^2$, $U(x) = U = [-1,1]$. We use the initial state $x^0 = [-3.95\ -0.05]^{\mathrm{T}}$. The lookahead length is $\ell = 4$. All of above settings are the same as those given in [1, Example A]. In addition to the box constraints, we impose constraints on entire trajectories. In particular, we require that a complete trajectory driven under some control sequence $\{u_k\}_{k=0}^{\infty}$ shall consume energy no more than 0.5, e.g., $\sum_{k=0}^{\infty} u_k^2 \leq 0.5$. This type of constraint goes beyond what is accounted by MPC methodology that is couched on Lyapunov-based analysis. However, the DP framework is flexible and admits modifications that can handle the problem. To this end, we apply state augmentation and define as our new state $y = (x, e)$ where e stands for energy budget that takes values in the interval $[0, 0.5]$, and its dynamics is $e_{k+1} = e_k - u_k^2$.

Suppose we have one trajectory $\{x_0, x_1, \dots\}$ under policy μ^0 that fulfills the state, control constraints and trajectory constraint. Then the sample set S^0 is given as

$$S^0 = \cup_{i=0}^{\infty}\left\{(x,e)\,\middle|\, x = x_i,\ e + \sum_{j=i}^{\infty}(\mu^0(x_j))^2 \leq 0.5\right\}.$$

Thus, even if the trajectory $\{x_0, x_1, \dots\}$ has a finite length, it 'generates' infinite samples in our augmented state space. So from the perspective of the augmented state y, the sample set S^0 contains infinitely many trajectories and the basic form of rollout can be applied here.

The obtained solution, in comparison with the initial feasible solution, and the trajectory-unconstrained suboptimal solutions are shown in Fig. 2. The initial feasible solution has a total cost 74.7492, with energy consumption 0.1853, thus feasible; the suboptimal solution without trajectory constraint has a total cost 49.9164. However, it consumes total energy 1.8616, and is thus infeasible. The suboptimal solution based upon the reformulated state results in a trajectory with total cost 59.4915 and energy consumption 0.5.

Example 4.3: (Simplified rollout scheme) To illustrate the simplified rollout when applied to multiagent system, we describe briefly an example of two vehicle path planning. Both vehicles are governed by the kinematic bicycle model discretized with some sampling time Δ_t and given as

$$\begin{aligned} y_{k+1} &= y_k + \Delta_t v_k \cos(\psi_k), & z_{k+1} &= z_k + \Delta_t v_k \sin(\psi_k), \\ v_{k+1} &= v_k + \Delta_t a_k, & \psi_{k+1} &= \psi_k + \Delta_t w_k, \end{aligned}$$

where a_k and w_k are controls that are subject to box constraints. The states at stage k are denoted collectively

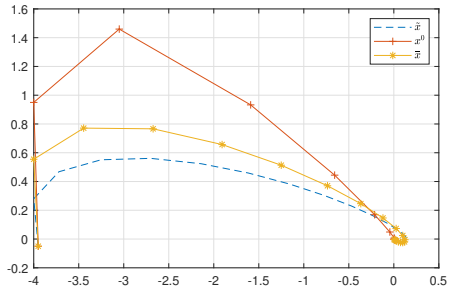

Fig. 2. The initial feasible trajectory, marked with dashed blue line and labelled $\tilde{x}$, has total cost 74.7492 and energy consumption 0.1853. The solution given by trajectory-unconstrained problem is in red and labelled with x^0. It has total cost 49.9164, but consumes 1.8616 energy, thus infeasible. The solution based upon reformulated state is in yellow and it has total cost 59.4915 and energy consumption 0.5.

by $x_k = (x_k^1, x_k^2) \in \Re^8$, where x_k^i, $i = 1, 2$, is the state associated with vehicle i. Similarly, the controls are collectively denoted by $u_k = (u_k^1, u_k^2) \in \Re^4$. The vehicles are tasked to plan shortest paths to their respective targets while remaining safe distances between each other.

At state x_k, we assume that a feasible trajectory $S^0 = \{x_k, \bar{x}_{k+1}, \bar{x}_{k+2}, \dots\}$ has been obtained under policy $\mu^0 = (\mu_1^0, \mu_2^0)$, where μ_i^0, $i = 1, 2$, prescribes controls to vehicle i. When simplified rollout is applied to this problem, vehicle 1 optimizes over $\{u_{k+i}^1\}_{i=0}^{\ell-1}$ while vehicle 2 is assumed to apply μ_2^0 (or equivalently, to follow the trajectory $\{\bar{x}_{k+i}^2\}_{i=1}^{\ell}$). To be consistent with the rest of the solution, vehicle 1 also needs to arrive at $\bar{x}_{k+\ell}^1$ at the end of the trajectory. Once the optimal is attained at $\{\tilde{u}_{k+i}^1\}_{i=0}^{\ell-1}$, similar procedure is repeated by vehicle 2, except that vehicle 1 uses the newly computed $\{\tilde{u}_{k+i}^1\}_{i=0}^{\ell-1}$. This procedure is repeated with best solution S^0 updated accordingly. The obtained solution is guaranteed to outperform initial solution through sequentially optimizing controls of two vehicles.

One important feature of this scheme is that it allows distributed computation, with relevant control signals (or equivalently, planned tentative paths) communicated between each other. A similar scheme in MPC literature is known as cooperative distributed MPC [29]. What is novel here is the construction of final state constraint and final cost, which is based upon the collected feasible paths.

Example 4.4: (A discrete optimization problem) We consider a variant of the classical travelling salesman problem. The explicit illustration of solutions elucidates the mechanism for the performance improvement from the use of multiple heuristics and/or enlarging the sample set. Starting from a city A, the task is to traverse three cities B, C, D, and go back to the city A. All city-wise travels have positive costs, while staying in the same city before completing the trip has an infinite cost. The exact values of costs are immaterial to our purpose, and one can find the corresponding numerical example in [26, Example 1.3.1]. All trips that end with city A and include all four cities belong to a stopping set X_s, e.g., ABCDA belongs to X_s while ABABA does not. Since cities can be repeatedly visited (e.g., ABABA), some policies may have infinite costs. Still, our problem is equivalent to the classical problem in the sense that they admit the same

optimal solution.

Suppose that the optimal schedule is ABDCA. Let μ^0 and μ^1 be two policies that bring states to X_s, where starting from A, μ^0 results in the complete trajectory $S^0 = \{\text{A}, \text{AC}, \text{ACD}, \text{ACDB}, \text{ACDBA}\}$, and μ^1 follows alphabetic order and gives $S^1 = \{\text{A}, \text{AB}, \text{ABC}, \text{ABCD}, \text{ABCDA}\}$. The policy μ^0 is inferior to μ^1, and neither is optimal. When the basic form with lookahead steps $\ell = 2$ and S^0 is used, one can see that $\tilde{\mu}$ coincides with μ^0 as any other choice would have an infinite cost. If rollout with multiple heuristics is applied, with the set S^0 replaced by $S = S^0 \cup S^1$, the rollout policy would recover the policy μ^1, as shown in Fig. 3. The performance improvement is due to the use of multiple heuristics. In either case, the rollout solutions do not contain cycles like ABABA. In addition, this example shows that when the data-driven rollout converges, it may converge at a suboptimal schedule, namely ABCDA. On the other

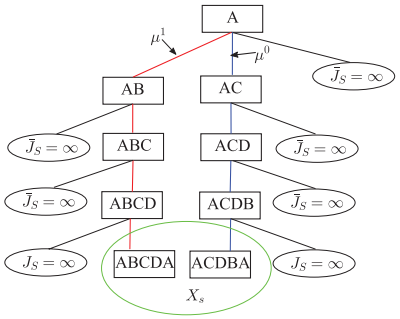

Fig. 3. The travelling salesman example with data-driven rollout method. The sampled trajectory under μ^0 is highlighted in blue, and the one under μ^1 is in red. The ellipsoids with black boundaries are subsets of X where $\tilde{J}_S$ is infinity. The ellipsoid with green boundary belongs to X_s.

hand, if a trajectory starting from ABD and under policy μ^1 is recorded and added to S, then our method obtains the optimal schedule. This indicates the value of exploration, and is consistent with the Remark 3.5.

V. CONCLUSIONS

We considered infinite horizon deterministic optimal control problems with nonnegative stage costs. By drawing inspiration from LMPC, we proposed a rollout algorithm variant that applies to problems with arbitrary state and control spaces, discontinuous dynamics, admits extensions for trajectory constrained problems, and applies to multiagent problems. We illustrated the validity of our assertions with various examples from combinatorial optimization, trajectory constrained problems, hybrid system optimization, and multiagent problems. Our method relies upon knowing a perfect model. The robustness issues that arise in the face of model mismatches require further research and numerical experimentation.

REFERENCES

[1] U. Rosolia and F. Borrelli, "Learning model predictive control for iterative tasks. A data-driven control framework," *IEEE Transactions on Automatic Control*, vol. 63, no. 7, pp. 1883–1896, 2017.

[2] D. P. Bertsekas, J. N. Tsitsiklis, and C. Wu, "Rollout algorithms for combinatorial optimization," *Journal of Heuristics*, vol. 3, no. 3, pp. 245–262, 1997.

[3] D. P. Bertsekas, "Dynamic programming and suboptimal control: A survey from ADP to MPC," *European Journal of Control*, vol. 11, no. 4-5, pp. 310–334, 2005.

[4] ——, "Lessons from AlphaZero for optimal, model predictive, and adaptive control," *arXiv preprint arXiv:2108.10315*, 2021.

[5] R. E. Strauch, "Negative dynamic programming," *The Annals of Mathematical Statistics*, vol. 37, no. 4, pp. 871–890, 1966.

[6] D. Blackwell, "Positive dynamic programming," in *Proceedings of the 5th Berkeley symposium on Mathematical Statistics and Probability*, vol. 1. University of California Press, 1967, pp. 415–418.

[7] ——, "Discounted dynamic programming," *The Annals of Mathematical Statistics*, vol. 36, no. 1, pp. 226–235, 1965.

[8] D. P. Bertsekas and S. Shreve, *Stochastic optimal control: the discrete-time case*. Academic Press, 1978.

[9] D. P. Bertsekas, *Dynamic programming and optimal control*, 4th ed. Athena Scientific, 2012, vol. 2.

[10] R. E. Bellman, *Dynamic programming*. Princeton University, 1957.

[11] G. Tesauro and G. R. Galperin, "On-line policy improvement using Monte-Carlo search," in *Proceedings of the 9th International Conference on Neural Information Processing Systems*, 1996, pp. 1068–1074.

[12] D. P. Bertsekas and D. A. Castanon, "Rollout algorithms for stochastic scheduling problems," *Journal of Heuristics*, vol. 5, no. 1, pp. 89–108, 1999.

[13] N. Secomandi, "A rollout policy for the vehicle routing problem with stochastic demands," *Operations Research*, vol. 49, no. 5, pp. 796–802, 2001.

[14] R. R. Lam, K. E. Willcox, and D. H. Wolpert, "Bayesian optimization with a finite budget: An approximate dynamic programming approach," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 883–891.

[15] X. Yue and R. A. Kontar, "Why non-myopic bayesian optimization is promising and how far should we look-ahead? A study via rollout," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2808–2818.

[16] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-dynamic programming*. Athena Scientific Belmont, MA, 1996, vol. 5.

[17] D. Bertsekas, "Multiagent reinforcement learning: Rollout and policy iteration," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 2, pp. 249–272, 2021.

[18] ——, "Rollout algorithms for constrained dynamic programming," *Lab. for Information and Decision Systems Report*, vol. 2646, 2005.

[19] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[20] J. Testud, J. Richalet, A. Rault, and J. Papon, "Model predictive heuristic control: Applications to industial processes," *Automatica*, vol. 14, no. 5, pp. 413–428, 1978.

[21] S. A. Keerthi and E. G. Gilbert, "Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: Stability and moving-horizon approximations," *Journal of optimization theory and applications*, vol. 57, no. 2, pp. 265–293, 1988.

[22] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[23] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*. Nob Hill Publishing Madison, WI, 2017, vol. 2.

[24] D. P. Bertsekas, "Control of uncertain systems with a set-membership description of the uncertainty." Ph.D. dissertation, Massachusetts Institute of Technology, 1971.

[25] ——, *Dynamic programming and optimal control*, 4th ed. Athena Scientific, 2017, vol. 1.

[26] ——, *Rollout, policy iteration, and distributed reinforcement learning*. Athena Scientific Belmont, MA, 2020.

[27] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[28] J. Lofberg, "Yalmip: A toolbox for modeling and optimization in matlab," in *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*. IEEE, 2004, pp. 284–289.

[29] B. T. Stewart, S. J. Wright, and J. B. Rawlings, "Cooperative distributed model predictive control for nonlinear systems," *Journal of Process Control*, vol. 21, no. 5, pp. 698–704, 2011.