Parallel Model Predictive Control for Deterministic Systems

Yuchao Li, Aren Karapetyan, Niklas Schmid,
John Lygeros, Karl H. Johansson, Jonas Mårtensson

Abstract— In this note, we consider infinite horizon optimal control problems with deterministic systems. Since exact solutions to these problems are often intractable, we propose a parallel model predictive control (MPC) method that provides an approximate solution. Our method computes multiple lookahead minimization problems at each time, where each minimization may involve a different number of lookahead steps, and terminal cost and constraint. The policy computed via parallel MPC applies the first control of the lookahead minimization with the lowest cost. We show that the proposed method can harnesses the power of multiple computing units. Moreover, we prove that the policy computed via parallel MPC has better performance guarantee than that computed via the single lookahead minimization involved in parallel MPC.

Index Terms— Model predictive control, deterministic systems, optimal control

I. INTRODUCTION

In this work, we consider discrete-time optimal control problems involving deterministic systems

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

where x_k and u_k are state and control at stage k belonging 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:

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

By allowing an infinite value of $g(x, u)$ we can implicitly introduce state and control constraints: a pair (x, u) is infeasible if $g(x, u) = \infty$. We consider *stationary* feedback policies μ, which are functions mapping X to U that satisfy $\mu(x) \in U(x)$ for all x.

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

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

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

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

A stationary policy μ^* is called optimal if

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

Optimal policies satisfy

$$\mu^*(x) \in \arg\min_{u \in U(x)} \{g(x, u) + J^*(f(x, u))\}, \quad \forall x \in X, \quad (3)$$

assuming the minimum in (3) can be attained for all x.

In the context of dynamic programming (DP), the goal is to obtain an optimal policy μ^* and optimal cost function J^*. However, computing μ^* and J^* is often intractable. To obtain an approximate solution, one may use some known function J in place of J^* and compute a policy $\tilde{\mu}$ via a minimization calculation similar to (3) at each time. If J resembles J^* in some way, we may expect the obtained policy $\tilde{\mu}$ close to optimal, in the sense that $J_{\tilde{\mu}} \approx J^*$. This idea constitutes the essence of a broad class of suboptimal schemes.

A popular method of this kind is model predictive control (MPC), which defines a policy through online computation similar to (3).[1] At state x_k, the MPC method considers the state sequence $x_k \ldots, x_{k+\ell}$ generated under the control sequence $u_k, \ldots, u_{k+\ell-1}$, where the number ℓ is referred to as (the length of) the *lookahead steps*. The desirability of different sequences are compared using the corresponding sums of the stage costs, plus a function defined on $x_{k+\ell}$. In addition, the state $x_{k+\ell}$ may also be restricted within a set. The function and the constraint associated with $x_{k+\ell}$ are called *terminal cost* and *terminal constraint*, respectively. Collectively, they aim to approximate the optimal cost J^*. The resulting optimization problem is simpler than computing J^*, and may be possible to solve online. Suppose the minimum of the problem is attained at $\tilde{u}_k, \ldots, \tilde{u}_{k+\ell-1}$, MPC policy $\tilde{\mu}$ is defined by setting $\tilde{\mu}(x_k) = \tilde{u}_k$. When ℓ is sufficiently large, and the terminal cost and constraint have certain properties, the resulting MPC policy can be close to optimal.

In contrast to the MPC method that involves a single minimization problem, a different suboptimal scheme, known as parallel rollout, solves simultaneously multiple copies of (3). Each copy of the minimization problem in parallel rollout uses a different cost function J_μ in place of J^*, corresponding to a different policy μ. The method then uses the control corresponding to the problem with the lowest cost. Parallel rollout was first proposed to address finite horizon problems, and has been popular in contexts involving discrete state and/or control elements.

Taking inspiration from parallel rollout, we propose a new MPC scheme, which we call *parallel MPC*. It requires solving multiple MPC lookahead minimizations at each time, where each minimization can involve different lookahead steps, and terminal cost and/or constraint. Compared with the MPC scheme noted earlier, the policy $\tilde{\mu}$ defined by parallel MPC has better performance guarantees. Let us consider the following scalar example to illustrate the scheme.

Example 1.1: (Parallel MPC for a Scalar Problem) Let $X = \Re$, $U(x) = U = [-1, 1]$ for all x, and $f(x, u) = 2x + u$. Assume that $g(x, u) = x^2 + u^2$ if $|x| \leq 2$, and $g(x, u) = \infty$ otherwise. For this problem, an example of parallel MPC can be described as follows:

---

at state x_k, solve the following two optimization problems:

$$\min_{u_k, u_{k+1} \in U} g(x_k, u_k) + g(x_{k+1}, u_{k+1}) \ \text{ s. t. } x_{k+2} = 0,$$

$$\min_{u_k \in U} g(x_k, u_k) + (13/3)x_{k+1}^2 \ \text{ s. t. } |x_{k+1}| \le 2/3,$$

with additional constraints $x_{k+1} = 2x_k + u_k$, and $x_{k+2} = 2x_{k+1} + u_{k+1}$; cf. Eq. (1). The procedure through which the terminal costs and constraints are computed will be explained in Example 3.2. Then the optimal values and the controls $\tilde{u}_k$ that attain the minimum in the two problems are computed, and the control that corresponds to the minimum of the two optimal values is applied by parallel MPC.

The contributions of this work are as follows:

(a) We introduce a parallel MPC method that involves solving multiple minimization problems with different lookahead horizons, and terminal costs and/or constraints.

(b) We show that parallel MPC can harnesses the power of multiple computing units.

(c) We prove that the policy defined by parallel MPC has a better performance guarantee than that defined by the single lookahead minimization involved in parallel MPC.

Note that our method and analysis also apply to the problems with an additional discount factor $\alpha \in (0, 1)$ on future stage costs. We restrict our attention to the problems without a discount factor as this is the primary context where MPC methods are used.

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 MPC algorithm and its variants, and we provide analysis. In Section IV, we demonstrate our method through computational examples.

II. BACKGROUND

The study of infinite horizon optimal control with deterministic systems and nonnegative cost has a long and rich history. Exact solutions have been derived for some special types of problems, see, e.g., [2], [21], [24], [27]. For a rigorous analysis that applies generally, see [6], where the properties of J^*, as well as algorithmic behaviors for exact solution methods, are addressed.

Despite these extensive research efforts, many deterministic optimal control problems remain intractable. As a result, one often resorts to approximate solution methods. For problems with continuous state and control spaces under constraints, MPC has had abundant applications with wide success. The central themes of MPC analysis are maintaining the *feasibility* of the ℓ-step lookahead minimization problems upon reaching new states, and the *stability* of the resulting closed-loop system under the obtained policy. The former question is typically addressed by introducing a terminal constraint with an invariance property [3]. The terminal cost is often designed to fulfill a condition, known as the *Lyapunov inequality*, to provide the stability guarantees [30]. Since the early work [22], there have been many MPC variants with these desired properties. Among them, examples involving multiple terminal constraints and/or costs include [16], [25], [29]. Yet these methods lack generality or flexibility compared with parallel MPC introduced here. Our method is also different from the MPC schemes reported in [18], [19], which focus on solving efficiently a single lookahead minimization via parallel computation. In contrast, our methods involves multiple lookahead minimization problems solved in parallel, and our goal is to improve the performance of the MPC policy measured by its cost function.

As for the problems with discrete state and/or control elements, parallel rollout has been shown to be effective. The method was first introduced in [14] (under the name *rollout with multiple heuristics*) to address combinatorial problems. Since then, parallel rollout and its

variants have been extended to address a variety of other problems, see, e.g., [1], [13], [15]. For a comprehensive discussion, see the recent monograph [8] and the references quoted therein. The analysis associated with parallel rollout focuses on ensuring a performance improvement property, which implies that the cost function of the policy $\tilde{\mu}$ computed via parallel rollout is no more than all the cost functions J_μ used to approximate J^*. This property of parallel rollout can be proven by using the result [7, Prop. 6.1.1] that holds more generally for functions other than the cost functions J_μ. Parallel MPC leverages this generality by allowing more flexible choices of the function to approximate J^* and the numbers of lookahead steps.

Although MPC and parallel rollout are proposed in different contexts, these methods can be unified through a conceptual framework that is based on DP; see the monograph [10] and the surveys [11], [12]. In particular, it is shown that both MPC and parallel rollout are one step of the Newton's method applied to compute the optimal cost function J^*, which is responsible for the wide success of these methods. Moreover, the analysis of MPC and rollout can be unified by using a condition that connects to the Lyapunov inequality, as is done in [7, Sections 6.1 and 6.5]. We will use this framework to introduce and to analyze the parallel MPC method.

Technical Preliminaries

We apply the abstract DP model given in [9] in order to concisely state our method and to streamline our proofs. In particular, we denote as $\mathcal{E}^+(X)$ the set of all functions $J : X \mapsto [0, \infty]$. A mapping that plays a key role in our development is the *Bellman operator* $T : \mathcal{E}^+(X) \mapsto \mathcal{E}^+(X)$, defined pointwise as

$$(TJ)(x) = \inf_{u \in U(x)} \big\{ g(x, u) + J\big(f(x, u)\big) \big\}. \tag{4}$$

This operator is well-posed in view of the nonnegativity (2) of the stage cost. We denote as T^ℓ the ℓ-fold composition of T, with the convention that $T^0 J = J$. From (4), it can be seen that if $J(x) \le J'(x)$ for all $x \in X$, we have that $(TJ)(x) \le (TJ')(x)$ for all $x \in X$; we refer to this as the *monotonicity property* of T. We are particularly interested in a subset of $\mathcal{E}^+(X)$, which we call the *L-region* (L stands for Lyapunov), defined as

$$\mathcal{L}(X) = \{ J \in \mathcal{E}^+(X) \,|\, (TJ)(x) \le J(x), \forall x \in X \}. \tag{5}$$

Since the function $J \in \mathcal{E}^+(X)$ can take the value infinity, it can be used to encode an invariance property, as we will show in Section III.

Throughout our analysis, we make frequent reference to the following monotonicity property: For all nonnegative functions J and J' that map X to $[0, \infty]$, if $J(x) \le J'(x)$ for all $x \in X$, then for all $x \in X$, $u \in U(x)$, we have

$$g(x, u) + J\big(f(x, u)\big) \le g(x, u) + J'\big(f(x, u)\big). \tag{6}$$

In addition, we also use the following classical result, which holds well beyond the scope of our study, and can be found in [5, Prop. 4.1.2, Prop. 4.1.4(a)].

Proposition 2.1: Under the nonnegativity condition (2), the following hold.

(a) For all stationary policies μ we have

$$J_\mu(x) = g(x, \mu(x)) + J_\mu\big(f(x, \mu(x))\big), \quad \forall x \in X.$$

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

$$g(x, \mu(x)) + J\big(f(x, \mu(x))\big) \le J(x), \quad \forall x \in X,$$

then $J_\mu(x) \le J(x)$ for all $x \in X$.

III. MAIN RESULTS

Let us formally state our parallel MPC algorithm and analyze its properties. We impose the following assumption, which will allow us to replace inf by min thereafter. *The assumption remains valid throughout this work and thus is omitted in the subsequent theoretical statements.*

Assumption 3.1: The cost nonnegativity condition (2) holds. Moreover, for all functions $J \in \mathcal{E}^+(X)$ that we consider, 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 $x \in X$.

Note that our method and analysis is valid under Assumption 3.1, regardless of the nature of the state and control spaces. On the other hand, different types of state and control spaces, being continuous or discrete, may impact the validity of the assumption, and can determine the ways in which we solve the lookahead minimization problems involved in our method.

A. Description of Parallel MPC

Consider the optimal control problem with state equation (1) and stage cost (2). We assume that we have access to p computational units as well as a central unit. Given the current state x_k, computational unit $i = 1, \ldots, p$ solves the problem

$$\min_{\{u_{k+j}\}_{j=0}^{\ell_i-1}} \quad J_i(x_{k+\ell_i}) + \sum_{j=0}^{\ell_i-1} g(x_{k+j}, u_{k+j}) \tag{7a}$$

$$\text{s.t.} \quad x_{k+j+1} = f(x_{k+j}, u_{k+j}), \ j = 0, \ldots, \ell_i - 1, \tag{7b}$$

$$u_{k+j} \in U(x_{k+j}), \ j = 0, \ldots, \ell_i - 1, \tag{7c}$$

where ℓ_i is some positive integer, which can vary with i, and $J_i \in \mathcal{L}(X)$ is a function *computed off-line in explicit form.*[2] The optimal value of the problem obtained by the ith unit is denoted by $\tilde{J}_i(x_k)$, and the corresponding minimizing sequence is denoted by $(\tilde{u}_k^i, \tilde{u}_{k+1}^i, \ldots, \tilde{u}_{k+\ell_i-1}^i)$. The values $\tilde{J}_i(x_k)$ and $\tilde{u}_k^i$ are then sent to the central unit that computes

$$\tilde{i} \in \arg \min_{i=1,2,\ldots,p} \tilde{J}_i(x_k). \tag{8}$$

The parallel MPC policy is then defined by setting $\tilde{\mu}(x_k) = \tilde{u}_k^{\tilde{i}}$. If we employ the Bellman operator (4), the algorithm can be stated succinctly as computing

$$\tilde{J}(x_k) = \min_{i=1,2,\ldots,p} (T^{\ell_i} J_i)(x_k), \tag{9}$$

by the central unit, where the values $(T^{\ell_i} J_i)(x_k)$ are supplied by the p computational units. Let us now connect the parallel MPC stated in Example 1.1 with its generic form given in (7).

Example 3.1: In the parallel MPC discussed in Example 1.1, there are 2 computing units. The lookahead minimization problems solved in the two units are:

$$\text{unit 1:} \quad \min_{u_k, u_{k+1} \in U} g(x_k, u_k) + g(x_{k+1}, u_{k+1}) \ \text{ s.t. } x_{k+2} = 0,$$

$$\text{unit 2:} \quad \min_{u_k \in U} g(x_k, u_k) + (13/3)x_{k+1}^2 \ \text{ s.t. } |x_{k+1}| \le 2/3, \tag{10}$$

[2] We have restricted our attention to the cases where the functions J_i are known in explicit forms. Regarding the situation where the functions J_i can only be evaluated online through simulation, as in parallel rollout given in [8, Section 5.2], our method and analysis to be introduced shortly applies in principle. However, in such a case, the lookahead minimization involved in our method can be solved only if the control space is discrete.

with additional constraints $x_{k+1} = 2x_k + u_k$, and $x_{k+2} = 2x_{k+1} + u_{k+1}$. Here $\ell_1 = 2$, and $\ell_2 = 1$. The function $J_1(x)$ takes the value 0 if $x = 0$ and infinity otherwise. Similarly, $J_2(x) = (13/3)x^2$ if $|x| \le 2/3$ and infinity otherwise.

Although we have described the algorithm by assigning computational tasks to several distributed units, it should be clear that the scheme can also be used if there is one single computing unit available. In this case, the scheme is executed by obtaining $(T^{\ell_i} J_i)(x)$ in sequential computation.

In what follows, we will show the equivalence between parallel MPC and an MPC scheme involving a single lookahead minimization with a suitably defined terminal cost. We will also analyze the properties of the corresponding policy $\tilde{\mu}$, and provide a variant of the scheme. For parallel MPC defined by (7) and (8), we denote by ℓ the minimum value of the lookahead steps, i.e., $\ell = \min_i\{\ell_i\}$. In addition, we define $\bar{J}_i = T^{\ell_i - \ell} J_i$. As a result, the minimization (7) involving different lookahead steps can be transformed to the ones that have *a common lookahead step* ℓ with terminal costs $\bar{J}_i$.

B. Justification of Parallel MPC

In what follows, we show that parallel MPC is equivalent to a single lookahead minimization with some function that approximates J^*. For this purpose, let us introduce some concepts. For every function $J \in \mathcal{E}^+(X)$ and state $x \in X$, we define the set $\tilde{U}(J, x)$ as

$$\tilde{U}(J, x) = \arg \min_{u \in U(x)} \big\{ g(x, u) + J\big(f(x, u)\big) \big\}. \tag{11}$$

Assumption 3.1 asserts that these sets are always nonempty. In addition let $\{\bar{J}_i\}_{i=1}^p \subset \mathcal{E}^+(X)$ be a finite collection of nonnegative functions, and $\bar{J}$ be their pointwise minimum

$$\bar{J}(x) = \min_{i=1,2,\ldots,p} \bar{J}_i(x), \ \forall x \in X. \tag{12}$$

For every nonnegative integer ℓ and state x, we define the nonempty set of the indices of the computing units

$$\tilde{I}(\ell, x) = \arg \min_{i=1,2,\ldots,p} (T^\ell \bar{J}_i)(x). \tag{13}$$

Aided by those concepts, we provide our first result, which states that parallel MPC involving $\bar{J}_i$ is equivalent to solve a single lookahead minimization with $\bar{J}$ defined in (12). This equivalence is shown both in terms of minimum value [part (a)], and also the control that attains the minimum [part (b)].

Proposition 3.1: Let $\{\bar{J}_i\}_{i=1}^p \subset \mathcal{E}^+(X)$ where p is a positive integer, $\bar{J}$ be given by (12), and ℓ be a positive integer.

(a) The equality

$$\min_{u \in U(x)} \big\{ g(x, u) + (T^{\ell-1} \bar{J})\big(f(x, u)\big) \big\}$$

$$= \min_{i=1,2,\ldots,p} \left[\min_{u \in U(x)} \big\{ g(x, u) + (T^{\ell-1} \bar{J}_i)\big(f(x, u)\big) \big\} \right]$$

holds for all $x \in X$, or equivalently,

$$(T^\ell \bar{J})(x) = \min_{i=1,2,\ldots,p} \big[(T^\ell \bar{J}_i)(x) \big], \ \forall x \in X. \tag{14}$$

(b) The equality

$$\tilde{U}(T^{\ell-1} \bar{J}, x) = \cup_{i \in \tilde{I}(\ell, x)} \tilde{U}(T^{\ell-1} \bar{J}_i, x) \tag{15}$$

holds for all $x \in X$, where $\tilde{I}(\ell, x)$ is defined as in (13).

Proof: We will first show that the proposition hold for $\ell = 1$. The cases where $\ell > 1$ are shown by induction.

(a) Suppose $\ell = 1$. For every fixed i, by (12) we have that $\bar{J}(x) \leq \bar{J}_i(x)$ for all x. By the monotonicity property (6), using $\bar{J}$ and $\bar{J}_i$ in place of J and J', and taking minimum on both sides, we have

$$\min_{u \in U(x)} \left\{ g(x,u) + \bar{J}\big(f(x,u)\big) \right\}$$
$$\leq \min_{u \in U(x)} \left\{ g(x,u) + \bar{J}_i\big(f(x,u)\big) \right\}, \; \forall x \in X.$$

By taking minimum over i on the right-hand side, we have

$$\min_{u \in U(x)} \left\{ g(x,u) + \bar{J}\big(f(x,u)\big) \right\}$$
$$\leq \min_{i=1,2,\ldots,p} \left[\min_{u \in U(x)} \left\{ g(x,u) + \bar{J}_i\big(f(x,u)\big) \right\} \right].$$

For the converse, we note that for every x, $\tilde{U}(\bar{J}, x)$ defined in Eq. (11) is nonempty by Assumption 3.1. Let $\tilde{u}$ be an arbitrary element of $\tilde{U}(\bar{J}, x)$. Then

$$\min_{u \in U(x)} \left\{ g(x,u) + \bar{J}\big(f(x,u)\big) \right\} = g(x,\tilde{u}) + \bar{J}\big(f(x,\tilde{u})\big).$$

In addition, if $\tilde{i} \in \tilde{I}\big(0, f(x,\tilde{u})\big)$ given by (13),

$$g(x,\tilde{u}) + \bar{J}\big(f(x,\tilde{u})\big) = g(x,\tilde{u}) + \bar{J}_{\tilde{i}}\big(f(x,\tilde{u})\big).$$

However, we also have that

$$\min_{i=1,2,\ldots,p} \left[\min_{u \in U(x)} \left\{ g(x,u) + \bar{J}_i\big(f(x,u)\big) \right\} \right]$$
$$\leq g(x,\tilde{u}) + \bar{J}_{\tilde{i}}\big(f(x,\tilde{u})\big),$$

which concludes the proof for $\ell = 1$. In particular, we obtain the equality

$$\min_{u \in U(x)} \left\{ g(x,u) + \bar{J}\big(f(x,u)\big) \right\}$$
$$= \min_{i=1,2,\ldots,p} \left[\min_{u \in U(x)} \left\{ g(x,u) + \bar{J}_i\big(f(x,u)\big) \right\} \right] \quad (16)$$

Let us assume that (14) hold for $\ell - 1$ with $\ell > 1$, then

$$\min_{u \in U(x)} \left\{ g(x,u) + (T^{\ell-1}\bar{J})\big(f(x,u)\big) \right\}$$
$$= \min_{u \in U(x)} \left\{ g(x,u) + \min_{i=1,2,\ldots,p} \left[(T^{\ell-1}\bar{J}_i)\big(f(x,u)\big) \right] \right\}.$$

Applying the equality (16) with $T^{\ell-1}\bar{J}_i$ in place of $\bar{J}_i$, and $\min_{i=1,2,\ldots,p}(T^{\ell-1}\bar{J}_i)(x)$ in place of $\bar{J}(x)$, yields that

$$\min_{u \in U(x)} \left\{ g(x,u) + \min_{i=1,2,\ldots,p} \left[(T^{\ell-1}\bar{J}_i)\big(f(x,u)\big) \right] \right\}$$
$$= \min_{i=1,2,\ldots,p} \left[\min_{u \in U(x)} \left\{ g(x,u) + (T^{\ell-1}\bar{J}_i)\big(f(x,u)\big) \right\} \right].$$

Combining two equations gives the desired result.

(b) We show that the result holds for $\ell = 1$. The induction part is similar to that of the proof for part (a) and is thus omitted. Given an x, let $\tilde{u} \in \tilde{U}(\bar{J}, x)$, then we have that

$$(T\bar{J})(x) = g(x,\tilde{u}) + \bar{J}\big(f(x,\tilde{u})\big).$$

Also, if $\tilde{i} \in \tilde{I}\big(0, f(x,\tilde{u})\big)$, where $\tilde{I}(\cdot, \cdot)$ is given as in (13), then it yields that

$$g(x,\tilde{u}) + \bar{J}\big(f(x,\tilde{u})\big) = g(x,\tilde{u}) + \bar{J}_{\tilde{i}}\big(f(x,\tilde{u})\big).$$

By part (a), we have that

$$g(x,\tilde{u}) + \bar{J}_{\tilde{i}}\big(f(x,\tilde{u})\big) = \min_{i=1,2,\ldots,p} \left[(T\bar{J}_i)(x) \right].$$

Therefore, $\tilde{i} \in \tilde{I}(1, x)$ and $\tilde{u} \in \tilde{U}(\bar{J}_{\tilde{i}}, x)$.

Conversely, let $\tilde{i} \in \tilde{I}(1, x)$ and $\tilde{u} \in \tilde{U}(\bar{J}_{\tilde{i}}, x)$. We will show that $(T\bar{J})(x) = g(x,\tilde{u}) + \bar{J}\big(f(x,\tilde{u})\big)$. By the definition of $\tilde{I}(1, x)$ and $\tilde{U}(\bar{J}_{\tilde{i}}, x)$, we have that $(T\bar{J})(x) = g(x,\tilde{u}) + \bar{J}_{\tilde{i}}\big(f(x,\tilde{u})\big)$. On the other hand, we also have

$$(T\bar{J})(x) \leq g(x,\tilde{u}) + \bar{J}\big(f(x,\tilde{u})\big) \leq g(x,\tilde{u}) + \bar{J}_{\tilde{i}}\big(f(x,\tilde{u})\big),$$

which gives the desired result. ∎

Remark 3.1: When $\bar{J}_i$ are the cost functions of certain policies, this result has been established in [14] for combinatorial problems. It has also been shown for the Markovian decision problem in [8, Prop. 5.2.2], the stochastic shortest path problem under certain conditions, as well as deterministic problems [8, p. 378]. We generalize these results by allowing $\bar{J}_i$ to be arbitrary nonnegative functions.

In view of Prop. 3.1(a), we can interpret the proposed scheme (9) as solving $(T^\ell \bar{J})(x_k)$ for some integer ℓ and some function $\bar{J}$. To see this, define $\ell = \min_i\{\ell_i\}$, and $\bar{J}_i = T^{\ell_i - \ell}J_i$. Moreover, Prop. 3.1(b) establishes that the unit $\tilde{i}$ selected by the central unit with the smallest value $\tilde{J}_i(x_k)$ must be an element of $\tilde{I}(\ell, x_k)$. In addition, the first control $\tilde{u}_k^{\tilde{i}}$ must be an element of $\tilde{U}(T^{\ell-1}\bar{J}, x_k)$. Therefore, the proposed algorithm solves the problem $(T^\ell \bar{J})(x_k)$ with ℓ and $\bar{J}$ defined accordingly. As a result of Prop. 3.1, analysis on the parallel MPC can be transformed to studying $T^\ell \bar{J}$. Indeed, the left side of Eq. (14) serves as an analytical device for our investigation on the properties of parallel MPC, while the right side of Eq. (14) represents the actual computational approach we employ. This is because the optimization problem defined by $(T^\ell \bar{J})(x)$ is often more challenging to solve compared with that defined by the right side of Eq. (14); see Examples 3.2 and 4.1. Yet, it is easier to analyze the properties of the policy defined by $(T^\ell \bar{J})(x)$.

In what follows, we will show that $\bar{J}$ inherits the desired properties of J_i, thus resulting in the desired behavior of the policy $\tilde{\mu}$.

Proposition 3.2: Let $\{J_i\}_{i=1}^p \subset \mathcal{L}(X)$ where p is a positive integer, and $\{\ell_i\}_{i=1}^p$ be a collection of p positive integers. In addition, let $\ell = \min_i\{\ell_i\}$, $\bar{J}_i = T^{\ell_i - \ell}J_i$, and $\bar{J}$ be defined (12). Then $\bar{J} \in \mathcal{L}(X)$.

Proof: Note that if $J \in \mathcal{L}(X)$, then due to the monotonicity of T, we have that $T^j J \in \mathcal{L}(X)$ for any nonnegative integer j. Therefore, $\bar{J}_i = T^{\ell_i - \ell}J_i \in \mathcal{L}(X)$. This yields that

$$(T\bar{J}_i)(x) \leq \bar{J}_i(x), \text{ for all } x \text{ and } i.$$

Taking minimum over i, the left side becomes $(T\bar{J})(x)$ due to (14) in Prop. 3.1(a) for $\ell = 1$, and the right side equals $\bar{J}(x)$ by (12). ∎

Finally, we show that the minimum value computed in (9) provides an upper bound for the cost function of the parallel MPC policy.

Proposition 3.3: The cost function $J_{\tilde{\mu}}(x)$ of the parallel MPC policy $\tilde{\mu}$ defined through (7) and (8), is upper bounded by the value $\tilde{J}(x)$ obtained via minimization (9) by the central unit.

Proof: Our preceding discussion has shown that the parallel MPC scheme is equivalent to computing $(T^\ell \bar{J})(x)$. In addition, we have that for $i = 1, \ldots, p$,

$$g\big(x, \tilde{\mu}(x)\big) + \big(T^{\ell-1}\bar{J}\big)\big(f\big(x, \tilde{\mu}(x)\big)\big)$$
$$= (T^\ell \bar{J})(x) \leq (T^{\ell-1}\bar{J})(x),$$

where the last inequality follows as $\bar{J} \in \mathcal{L}(X)$, by Prop. 3.2. Applying Prop. 2.1(b) with $T^{\ell-1}\bar{J}$ as J, we have $J_{\tilde{\mu}}(x) \leq T^{\ell-1}\bar{J}(x)$. By the monotonicity property (6), we have

$$J_{\tilde{\mu}}(x) = g\big(x, \tilde{\mu}(x)\big) + J_{\tilde{\mu}}\big(f\big(x, \tilde{\mu}(x)\big)\big)$$
$$\leq g\big(x, \tilde{\mu}(x)\big) + \big(T^{\ell-1}\bar{J}\big)\big(f\big(x, \tilde{\mu}(x)\big)\big) = (T^\ell \bar{J})(x),$$

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 3.2: Let us denote by $\tilde{\mu}_i$ the MPC policy defined by $T^{\ell_i} J_i$; i.e., the MPC policy computed by the ith unit. Classical MPC theory states that $J_{\tilde{\mu}_i}(x) \leq \tilde{J}_i(x) \leq \infty$ for all $x \in X$. The proposition states that the cost function of the parallel MPC policy $\tilde{\mu}$ is upper bounded by the pointwise minimum of the functions $\tilde{J}_i$ computed via individual lookahead. Therefore, $J_{\tilde{\mu}}(x) < \infty$ as long as $\tilde{J}_i(x) < \infty$ for some i. This implies that the set of the states x where $J_{\tilde{\mu}}(x) < \infty$ is the union of the sets $\{x \in X \mid J_{\tilde{\mu}_i}(x) < \infty\}$, $i = 1, 2, \ldots, p$.

Remark 3.3: Under certain conditions, we can show that the cost function $J_{\tilde{\mu}}$ satisfies $J_{\tilde{\mu}} \leq J_{\tilde{\mu}_i}$ for some i. In particular, if $J_j = J_{\tilde{\mu}_i}$ for some $j \neq i$, then $\tilde{J}_j = T^{\ell_j} J_j \leq J_j = J_{\tilde{\mu}_i}$. Such a case arises when the function J_j is constructed via collecting state-and-control trajectory data under $\tilde{\mu}_i$, as is proposed in [31]. Our computational study, reported in Section IV, also provides empirical evidence that the policy $\tilde{\mu}$ often outperforms the MPC policies defined by the individual lookahead minimization even without such conditions.

Remark 3.4: If $\ell_i = \ell$ for all i, and the functions J_i are cost functions of some policies μ_i, then the result here recovers the conclusion stated in [8, p. 378].

We demonstrate these propositions using our running example.

Example 3.2: Consider the parallel MPC applied to the problem studied in Examples 1.1 and 3.1. Prop. 3.1 states that the minimum of the optimal values computed by the two units, as shown in Example 3.1, equals to the minimum $\tilde{J}(x_k)$ of the following problem:

$$\min_{u_k \in U} \quad g(x_k, u_k) + \bar{J}(x_{k+1})$$
$$\text{s.t.} \quad \bar{J}(x_{k+1}) = \min\{\bar{J}_1(x_{k+1}), \bar{J}_2(x_{k+1})\},$$
$$\bar{J}_1(x_{k+1}) = \min_{\substack{u_{k+1} \in U \\ 2x_{k+1} + u_{k+1} = 0}} g(x_{k+1}, u_{k+1}), \quad (17)$$
$$\bar{J}_2(x_{k+1}) = \begin{cases} 13/3x_{k+1}^2, & \text{if } |x_{k+1}| \leq 2/3, \\ \infty, & \text{otherwise}; \end{cases}$$

cf. Eq. (14). Moreover, the control $\tilde{u}_k$ that attains the minimum in problem (17) also attains the minimum of the problem computed by the unit with smaller minimum values in problems (10); cf. Eq. (15). Clearly, problems (10) are easier to solve than problem (17).

Taking as given that $\bar{J}_1, \bar{J}_2 \in \mathcal{L}(X)$ (we will show this later), Props. 3.2 and 3.3 state that the policy $\tilde{\mu}$ defined by $\tilde{\mu}(x_k) = \tilde{u}_k$ satisfies $J_{\tilde{\mu}}(x_k) \leq \tilde{J}(x_k)$.

C. Properties of the L-Region

We now address the question on how to obtain J_i that belongs to the L-region defined in (5). Prop. 2.1(a) suggests that any J_μ constitutes a valid option. The following result from [25] (included here for completeness) shows that J_i can be used to encode simultaneously the Lyapunov inequality of the terminal cost and the invariance property of the terminal constraint. The cost and the constraint can be computed via optimization, as in [20], [26], as well as via sampled data [25], [31]. To this end, we denote by $\delta_C : X \mapsto \{0, \infty\}$ the function taking value 0 if $x \in C$ and infinity otherwise.

Proposition 3.4: Let $S \subset X$, $V \in \mathcal{E}^+(X)$, and μ be a policy such that

$$x \in S \implies f(x, \mu(x)) \in S, \quad (18)$$

and

$$g(x, \mu(x)) + V(f(x, \mu(x))) \leq V(x), \quad \forall x \in S. \quad (19)$$

Then $J \in \mathcal{E}^+(X)$ defined as $J(x) = V(x) + \delta_S(x)$ belongs to the L-region.

Remark 3.5: The condition stated in (18) is often referred to as the invariance property of the set S, also known as S being

strong reachable, introduced first in [3]. When the function V takes nonnegative real numbers, the inequality (19) is equivalent to the Lyapunov inequality. Both of these concepts are extensively studied in the MPC literature, and Prop. 3.4 shows that they can be encoded as the property defining the L-region. This is convenient for our algorithmic design and analysis. However, we do not investigate the computation of the set S and the function V in the present work.

We show next that the functions $\bar{J}_1$ and $\bar{J}_2$ in Example 3.2 belong to the L-region using Prop. 3.4.

Example 3.3: Let us consider $\bar{J}_1$ and $\bar{J}_2$ defined in the problem (17). We define $J_1 = V_1 + \delta_{S_1}$, where $V_1 \equiv 0$ and $S_1 = \{0\}$. Let $\mu_1(0) = 0$. Then one can verify that conditions (18) and (19) are satisfied with S_1, V_1, and μ_1 in place of S, V, and μ. Therefore, J_1 belongs to the L-region. Due to monotonicity of T, we have $\bar{J}_1 = TJ_1$ belong to the L-region as well.

As for $\bar{J}_2$, we consider $\mu_2(x) = -3/2x$. In view of the control constraint $U = [-1, 1]$, $\mu_2(x) = -3/2x$ is defined for $S_2 = \{x \mid |x| \leq 2/3\}$. One can verify that $x_2 \in S_2$ implies $f(x, \mu_2(x)) \in S_2$. Moreover, for every $x \in S_2$, $J_{\mu_2}(x) = 13/3x^2$. As a result, we set $V_2 = J_{\mu_2}$ and $g(x, -3/2x) + V_2(1/2x) = V_2(x)$; cf. condition (19). Since $\bar{J}_2 = V_2 + \delta_{S_2}$, $\bar{J}_2$ belongs to the L-region.

The following result show that for an element $J \in \mathcal{L}(X)$, the values of $(T^\ell J)(x_k)$ are decreasing as k increases, provided that the state sequence $\{x_k\}$ is generated under the policy $\tilde{\mu}$ defined by

$$\tilde{\mu}(x) \in \arg \min_{u \in U(x)} \left\{ g(x, u) + (T^{\ell-1} J)(f(x, u)) \right\}.$$

Proposition 3.5: Let $J \in \mathcal{L}(X)$, ℓ be a positive integer, and for a given x, let $\tilde{u}$ be an element in $\tilde{U}(T^{\ell-1} J, x)$. Then we have

$$(T^\ell J)(f(x, \tilde{u})) \leq (T^\ell J)(x).$$

Proof: We first show the result hold for $\ell = 1$. We will then establish the inequality for $\ell > 1$.

By the definition of $\tilde{U}(J, x)$, we have that $(TJ)(x) = g(x, \tilde{u}) + J(f(x, \tilde{u}))$. The function J belongs to the L-region so that $g(x, \tilde{u}) + (TJ)(f(x, \tilde{u})) \leq g(x, \tilde{u}) + J(f(x, \tilde{u}))$. Given that $g(x, \tilde{u}) \in [0, \infty]$, we obtain the equality

$$(TJ)(f(x, \tilde{u})) \leq (TJ)(x). \quad (20)$$

For $\ell > 1$, since $J \in \mathcal{L}(X)$, then in view of the monotonicity of T, we have $T^\ell J \in \mathcal{L}(X)$. Applying the inequality (20) with $T^\ell J$ replacing J, we get the desired inequality. ∎

D. A Simplified Variant of Parallel MPC

For some problems, computing $(TJ)(x)$ could be challenging due to the size of the set $U(x)$. In such cases, it may be convenient to restrict our attention to some subsets of $U(x)$. Given $\bar{U}(x) \subset U(x)$, we introduce the *simplified Bellman operator* $\bar{T} : \mathcal{E}^+(X) \mapsto \mathcal{E}^+(X)$, which is defined pointwise as

$$(\bar{T}J)(x) = \inf_{u \in \bar{U}(x)} \left\{ g(x, u) + J(f(x, u)) \right\}. \quad (21)$$

Similar to the Bellman operator, we denote as $\bar{T}^\ell$ the ℓ-fold composition of $\bar{T}$ with itself ℓ times, with the convention that $\bar{T}^0 J = J$. It is straightforward to verify that $\bar{T}$ also has the monotonicity property: $J(x) \leq J'(x)$ for all $x \in X$ implies that $(\bar{T}J)(x) \leq (\bar{T}J')(x)$ for all $x \in X$. In addition, for every $J \in \mathcal{E}^+(X)$, we have that $(TJ)(x) \leq (\bar{T}J)(x)$ for all $x \in X$. For functions defined through the operator $\bar{T}$, we have the following results.

Proposition 3.6: Let $J \in \mathcal{L}(X)$ and $\bar{U}(x) \subset U(x)$ such that

$$(\bar{T}J)(x) \leq J(x), \quad \forall x \in X,$$

where $\bar{T}$ is given in (21). Then we have that $\bar{T}^\ell J \in \mathcal{L}(X)$ where ℓ is a positive integer.

Proof: Let us first show the inequality for $\ell = 1$. Since $(\bar{T}J)(x) \leq J(x)$ for all x, in view of the monotonicity property of $\bar{T}$, we have that

$$(\bar{T}^2 J)(x) = \big(\bar{T}(\bar{T}J)\big)(x) \leq (\bar{T}J)(x), \quad \forall x \in X.$$

On the other hand,

$$\big(T(\bar{T}J)\big)(x) \leq \big(\bar{T}(\bar{T}J)\big)(x), \quad \forall x \in X.$$

Combining the above inequalities, we have that

$$\big(T(\bar{T}J)\big)(x) \leq (\bar{T}J)(x), \quad \forall x \in X,$$

which means that $\bar{T}J \in \mathcal{L}(X)$.

By recursively applying above arguments with $\bar{T}^{\ell-1}J$, $\ell = 1, 2, \ldots$, in place of J, we obtain the desired result. ∎

In view of the above result, we can introduce a simplified parallel MPC given as follows: instead of solving for each unit $(T^{\ell_i}J_i)(x)$, we may instead solve $\big(T(\bar{T}_i^{\ell_i-1}J_i)\big)(x)$, where $\bar{T}_i$ are simplified Bellman operators that can vary with i. Conceptually, we can view $\bar{T}^{\ell_i-1}J_i$ as a function computed to approximate J^* while each unit solves a one-step lookahead problem.

IV. ILLUSTRATING EXAMPLES

In this section, we illustrate through examples the validity of the parallel MPC applied to problems with deterministic dynamics. We aim to elucidate the following points:

(1) Parallel MPC can be applied to a wide range of optimal control problems with deterministic dynamics.
(2) Parallel implementation improves computational efficiency at no cost of optimality (Example 4.1); see Prop. 3.1.
(3) Parallel MPC provides better performance guarantees than that of the individual lookahead minimization involved in the method (Example 4.2); see Prop. 3.3.
(4) The simplified variant is well-suited for problems with discrete components (Example 4.3); see Prop. 3.6.

We use MATLAB with toolbox MPT [17] to solve Examples 4.1, 4.2, and 4.3, and we also use toolbox YALMIP [28] in Example 4.3. The code in MATLAB to reproduce all the examples and figures in this section, as well as additional examples, can be found in https://github.com/akarapet/pr-mpc. Computational details of J_i used in the examples are peripheral to the parallel MPC ideas and are thus deferred towards the appendices.

Example 4.1: (Linear Quadratic Problem with Constraints) In this example, we consider a two-dimensional linear quadratic (LQ) problem with constraint. We aim to show that parallel MPC can harness the parallel computing units at no cost of optimality. Here the state space is $X = \Re^2$, where $\Re^n$ denotes the n-dimensional Euclidean space, and the control constraints are $U(x) = U = \Re$. The system dynamics are given by $f(x, u) = Ax + Bu$, where

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}.$$

The stage cost is given by $g(x, u) = x'Qx + Ru^2 + \delta_C(x, u)$, where prime denotes transposition, Q is an identity matrix, and $R = 1$. The set C is $\{(x, u) \mid \|x\|_\infty \leq 5, |u| \leq 1\}$ ($\|x\|_\infty$ denotes the infinity norm of the vector x). We assume that four computational units are available, where $\ell_i = 2$ for $i = 1, 2$ and $\ell_i = 3$ for $i = 3, 4$. The functions $J_i(x)$ are computed in the form of $x'K_i x + \delta_{S_i}(x)$ so that $J_i \in \mathcal{L}(X)$ by Prop. 3.4; see Appendix A.

As discussed after Prop. 3.1, parallel MPC defined through Eqs. (7)-(9) can be described succinctly as the right-hand side of Eq. (14). Prop. 3.1 shows that parallel MPC is equivalent to solve directly the problem defined by left-hand side of Eq. (14), which

we refer to as alternative approach from now on. Compared with the parallel MPC, this alternative approach uses one computing unit and solves a single lookahead minimization problem involving binary variables. Our computational studies show that the parallel MPC significantly improves the computational efficiency compared with the alternative approach. For example, Fig. 1 shows the computational time required for parallel MPC and alternative approach for the states $\{x_k\}$ of trajectories starting from $x_0 = [-5 \ 2.7]'$ and $[2.3 \ -0.6]'$ and generated under the parallel MPC policy. We repeated the test with 100 trials and averaged over the trials. We note that on average in this experiment the time required for parallel MPC is about 15% of that needed for the alternative approach.
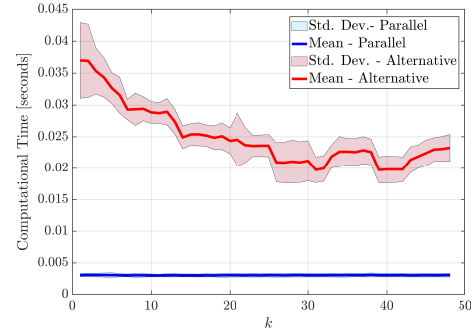

Fig. 1. The computational time of parallel MPC (labeled as Parallel in the figure) is compared to that of the alternative approach with binary variables. The horizontal axis represents the index of the state along the state trajectories $\{x_k\}$, which are generated under the parallel MPC policy $\tilde{\mu}$. The computational time is averaged over **100** trials.

Example 4.2: (Discontinuous Dynamics) In this example we demonstrate that the cost $J_{\tilde{\mu}}(x)$ of the policy $\tilde{\mu}$ computed via parallel MPC is upper bounded by the value $(T^\ell \bar{J})(x)$. This bound is no worse than the bounds $(T^{\ell_i}J_i)(x)$, which are computed via the individual lookahead minimization. For this purpose, we consider the problem studied in [23, Section VI], where the dynamics are

$$x_{k+1} = \begin{cases} A_1 x_k + B u_k, & \text{if } [1 \ 0]x_k \geq 0, \\ A_2 x_k + B u_k, & \text{otherwise.} \end{cases}$$

The parameters of the system dynamics are

$$A_1 = \begin{bmatrix} 0.35 & -0.61 \\ 0.61 & 0.35 \end{bmatrix}, \ A_2 = \begin{bmatrix} 0.35 & 0.61 \\ -0.61 & 0.35 \end{bmatrix}, \ B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

The stage cost is $g(x, u) = x'Qx + Ru^2 + \delta_C(x, u)$, where $R = 0.4$, and the matrix Q and the set C are the same as in Example 4.1. We assume four computational units and set $\ell_i = 2$ for all i, and introduce for each unit a function $J_i \in \mathcal{L}(X)$; see Appendix B for the computational details. Table I lists the results where parallel MPC is applied to some states x_0. These include the bounds $\tilde{J}_i$ computed by the individual units, the unit $\tilde{i}$ selected by the central unit, the actual performance bound $(T^\ell \bar{J})(x_0)$, which bounds the cost function values $J_{\tilde{\mu}}(x_0)$; see Prop. 3.3. Empirically, we also found that the cost function $J_{\tilde{\mu}}$ of parallel MPC policy is lower than that of the policies defined by the individual lookahead minimization involved in the scheme. In other words, parallel MPC outperforms the MPC policies defined by the individual lookahead minimization.

Example 4.3: (Switching System) In this example we show that the simplified variant of parallel MPC is well-suited for problems involving switching systems, where the control has both discrete and continuous elements. We consider a problem modified from [32, Section IV.A], where the dynamics are $x_{k+1} = A_{d_k}x_k + B_{d_k}v_k$,

TABLE I
OPTIMAL VALUES AND COSTS IN EXAMPLE 4.2.

x_0	$\tilde{J}_1$	$\tilde{J}_2$	$\tilde{J}_3$	$\tilde{J}_4$	$\tilde{i}$	$T^\ell \bar{J}$	$J_{\tilde{\mu}}$
$[-4\,4.6]'$	53.4	∞	53.4	54.3	3	53.4	52.2
$[5\,3.4]'$	53.1	∞	∞	54.7	1	53.1	52.0
$[-4.5\,2.7]'$	39.2	39.2	39.1	40.4	3	39.1	38.4
$[-5\,-5]'$	∞	∞	91.1	85.9	4	85.9	81.4

where $u = (v, d)$ with $v \in \Re$ and $d \in \{1, 2\}$, and the control constraint set is $U(x) = \Re \times \{1, 2\}$ for all x. In addition,

$$A_1 = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}, \ B_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \ A_2 = \begin{bmatrix} 2 & 1 \\ 0 & 0.5 \end{bmatrix}, \ B_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

The stage cost is $g(x, u) = x'Qx + Rv^2 + \delta_C(x, v)$, where the parameters Q, R, and C are the same as those in Example 4.1. Compared with the problem studied in [32, Section IV.A], we impose state and control constraints; in particular the term $\delta_C(x, v)$ is added to the stage cost. We use two computing units and we set $\ell_i = 5$ for both units. We also introduce control constraint subsets $\bar{U}_i(x) = \Re \times \{i\}$, $i = 1, 2$. The details of J_i are given in Appendix C. The results for simplified parallel MPC, computed for some states x_0, are summarized in Table II. The cost function values are compared against a lower bound of J^*, which is given by $T^8 \bar{J}_0$, where $\bar{J}_0(x) \equiv 0$ for all x. The difference between $J_{\tilde{\mu}}$ and $T^8 \bar{J}_0$ defines an optimality gap of $J_{\tilde{\mu}}$. The values of the gap indicate that the policy $\tilde{\mu}$ is near optimal. We note that the parallel MPC scheme solves 4 quadratic programs per time step, whereas the lower bound solves 2^8 quadratic programs of a similar kind but larger size. For the computation of the lower bound, see Appendix D. Moreover, we also verified in these tests that the cost function values of parallel MPC policy are lower than that of the MPC policies defined by the individual lookahead.

TABLE II
OPTIMAL VALUES AND COSTS IN EXAMPLE 4.3.

x_0	$T^\ell \bar{J}$	$J_{\tilde{\mu}}$	$\frac{J_{\tilde{\mu}} - T^8 \bar{J}_0}{J_{\tilde{\mu}}}$
$[-4\,4.6]'$	65.8	65.8	$< 10^{-5}$
$[1.2\,1.5]'$	89.2	86.6	$< 10^{-3}$
$[-3.5\,2.0]'$	123.3	113.5	$< 10^{-4}$
$[-1.5\,-0.5]'$	34.6	34.6	$< 10^{-4}$

V. CONCLUSIONS

We considered infinite horizon optimal control problems with deterministic dynamics. Drawing inspiration from parallel rollout, we proposed a parallel MPC method that computes approximate solutions. The proposed method solves multiple lookahead minimization problems with different lookahead steps, and terminal costs and terminal constraints. We showed that the proposed method can harness the power of multiple computing units. In addition, we proved that parallel MPC provides better performance bounds than that given by the individual lookahead minimization involved in the method. Our computational examples suggest that in addition to better bounds, parallel MPC often provides better performance than each of the MPC schemes that correspond to the individual lookahead minimizations. Preliminary tests show that the computational savings carry to the larger scale problems.

APPENDIX

In the appendix, we provide computational details of the functions J_i used in Examples 4.1, 4.2, and 4.3. Using Prop. 3.4, all functions J_i of the examples are constructed in the form of $V + \delta_S$. All these functions are computed offline before the realtime operation starts.

We also provide a lower bound of J^* obtained via value iteration. This bound is used in Example 4.3 to demonstrate the suboptimality of the proposed scheme. Note that alternative approaches for computing J_i introduced in [26] can also be applied here.

A. Off-Line Computation for Example 4.1

In this example, we compute the functions J_i such that for some policy μ_i, $J_i(x) = J_{\mu_i}(x)$ for all x in some set S_i. For $i = 1$, we compute $L_1 = -(B'K_1B + R)^{-1}B'K_1A$, where K_1 is the solution of the matrix equation

$$K = A'(K - KB(B'KB + R)^{-1}B'K)A + Q.$$

The set S_1 is computed by the command `invariantSet()` in the MPT toolbox for the system $A + BL_1$ with the presence of the constraint set C. The function J_1 is defined as $x'K_1x + \delta_{S_1}(x)$. As a result, we have $J_1(x) = J_{\mu_1}(x)$ and $\mu_1(x) = L_1x$ for all $x \in S_1$. Similar to $i = 1$, we compute $L_2 = -[0.1\ 1.2]$, $L_3 = -[0.2\ 0.7]$ and $L_4 = -[0.3\ 0.8]$ for the units $i = 2, 3, 4$. The values of L_i for these units are obtained via trial and error. The J_i for $i = 2, 3, 4$ are defined as $x'K_ix + \delta_{S_i}(x)$, where K_i is the solution to the equation

$$K = (A + BL_i)'K(A + BL_i) + Q + L_i'RL_i, \quad i = 2, 3, 4.$$

The sets S_2 and S_3 are computed in the same way as is for S_1, while the set S_4 is defined as $S_4 = \{x \in \Re^2 \mid x'K_4x \leq \alpha^*\}$. The value α^* used for defining S_4 is the optimal value of the optimization problem

$$\min_{\alpha \in \Re} \ \alpha, \ \ \text{s.t.} \ x'K_4x \leq \alpha, \ \forall x \text{ so that } (x, L_4x) \in C.$$

This approach of constructing the set dates back to the thesis [3]. The constraint in the optimization can be transformed as linear constraints imposed on α via standard multiplier method, cf. [4].

B. Off-Line Computation for Example 4.2

In this example, we compute the functions J_i such that for some policy μ_i, we have that $g(x, \mu_i(x)) + J_i(f(x, \mu_i(x))) \leq J_i(x)$ for all x in some set S_i. The policies μ_i for $i = 1, 2, 3$ take the form $\mu_i(x) = L_i^1 x$ if $[1\ 0]x_k \geq 0$ and $L_i^2 x$ otherwise. The policy μ_1 is the same as the one given in [23, Section VI]. The function J_1 takes the form $x'K_1x + \delta_{S_1}(x)$, with K_1 and S_1 given by [23, Section VI]. For the second unit, we set $L_2^i = -(B'K_iB + R)^{-1}B'K_iA_i$, where K_i is the solution of

$$K = A_i'(K - KB(B'KB + R)^{-1}B'K)A_i + Q, \quad i = 1, 2.$$

We follow the procedure outlined in [23, Sections IV, V] and obtain

$$K_2 = \begin{bmatrix} 1.4786 & 0 \\ 0 & 1.9605 \end{bmatrix},$$

as well as the set $S_2 = \{x \in \Re^2 \mid P_2x \leq e^4\}$, where $e^n \in \Re^n$ is an n-dimensional vector with all elements equal to 1, and P_2 is

$$P_2' = \begin{bmatrix} -0.4452 & 0.4452 & 0.4452 & -0.4452 \\ -0.3354 & 0.3354 & -0.3354 & 0.3354 \end{bmatrix}.$$

For the third unit, the parameters are synthesized by solving a linear matrix inequality given in [23, Sections IV]. The obtained parameters are $L_3^1 = -[0.5112\ 0.2963]$ and $L_3^2 = [0.5112\ -0.2963]$. Similarly, the function J_3 takes the form $x'K_3x + \delta_{S_3}(x)$ with

$$K_3 = \begin{bmatrix} 1.7847 & 0 \\ 0 & 2.1375 \end{bmatrix},$$

and the set $S_3 = \{x \in \Re^2 \mid P_3x \leq e^8\}$, where

$$P_3' = \begin{bmatrix} 0.15 & -0.15 & -0.15 & 0.15 & -0.51 & 0.51 & 0.51 & -0.51 \\ -0.33 & 0.33 & -0.33 & 0.33 & -0.30 & 0.30 & -0.30 & 0.30 \end{bmatrix}.$$

Given that matrices A_1 and A_2 are both stable, we set $\mu_4(x) = 0$ for all x. The function J_4 is given by $x'K_4x + \delta_{S_4}(x)$, where $S_4 = \{x \in \Re^2 \mid P_4x \le e_4\}$, and

$$K_4 = \begin{bmatrix} 1.9631 & 0 \\ 0 & 1.9631 \end{bmatrix}, \ P_4' = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}.$$

C. Off-Line Computation for Example 4.3

We define the functions J_i such that for some policy μ_i, $J_i(x) = J_{\mu_i}(x)$ for all x in some set S_i. To this end, we set $L_i = -(B_i'K_iB_i + R)^{-1}B_i'K_iA_i$, $i = 1, 2$, where K_i is the solution of the matrix equation

$$K = A_i'(K - KB_i(B_i'KB_i + R)^{-1}B_i'K)A_i + Q.$$

The set S_i is computed by the command `invariantSet()` in the MPT toolbox for the system $A_i + B_iL_i$ with the presence of the constraint set C. The function J_i is defined as $x'K_ix + \delta_{S_i}(x)$. This is the cost function of the policy that sets $d = i$, and $v = L_ix$ for $x \in S_i$. When applying the simplified variant at x_k with $\ell_i = 5$ and J_i, the computation on the ith unit takes the form $(T(\bar{T}_i^4 J_i))(x_k)$, where $(\bar{T}_iJ)(x) = \inf_{u \in \bar{U}_i(x)} \{g(x, u) + J(f(x, u))\}$, with $\bar{U}_i(x) = \Re \times \{i\}$, $i = 1, 2$; cf. (21). This is equivalent to the following optimization problem:

$$
\begin{aligned}
\min_{\{v_{k+j}\}_{j=0}^4, d} \quad & x'_{k+5}K_ix_{k+5} + \sum_{j=0}^4 x'_{k+j}Qx_{k+j} + Rv_{k+j}^2 \\
\text{s.t.} \quad & x_{k+1} = A_dx_k + B_dv_k, \ d \in \{1, 2\}, \\
& x_{k+j+1} = A_ix_{k+j} + B_iv_{k+j}, \ j = 1, ..., 4, \\
& \|x_{k+j}\|_\infty \le 5, \ j = 0, ..., 5, \\
& |v_{k+j}| \le 1, \ j = 0, ..., 4, \ x_{k+5} \in S_i.
\end{aligned}
$$

D. A Lower Bound of Optimal Cost

In Example 4.3, the lower bound of $J^*(x_k)$ is $(T^8\bar{J}_0)(x_k)$, which is the optimal value of the following optimization problem:

$$
\begin{aligned}
\min_{\{(v_{k+j}, d_{k+j})\}_{j=0}^7} \quad & \sum_{j=0}^7 x'_{k+j}Qx_{k+j} + Rv_{k+j}^2 \\
\text{s.t.} \quad & x_{k+j+1} = A_{d_k}x_{k+j} + B_{d_k}v_{k+j}, \ j = 0, ..., 7, \\
& \|x_{k+j}\|_\infty \le 5, \ j = 0, ..., 7, \\
& |v_{k+j}| \le 1, \ d_{k+j} \in \{1, 2\} \ j = 0, ..., 7.
\end{aligned}
$$

ACKNOWLEDGEMENT

REFERENCES

[1] Duarte Antunes and W. P. Maurice H. Heemels. Rollout event-triggered control: Beyond periodic control performance. *IEEE Transactions on Automatic Control*, 59(12):3296–3311, 2014.

[2] Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

[3] Dimitri P. Bertsekas. *Control of Uncertain Systems with a Set-Membership Description of the Uncertainty*. PhD thesis, Massachusetts Institute of Technology, 1971.

[4] Dimitri P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic press, 1982.

[5] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, 4 edition, 2012.

[6] Dimitri P. Bertsekas. Value and policy iterations in optimal control and adaptive dynamic programming. *IEEE Transactions on Neural Networks and Learning Systems*, 28(3):500–509, 2015.

[7] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, 4 edition, 2017.

[8] Dimitri P. Bertsekas. *Rollout, Policy Iteration, and Distributed Reinforcement Learning*. Athena Scientific Belmont, MA, 2020.

[9] Dimitri P. Bertsekas. *Abstract Dynamic Programming*. Athena Scientific, 3rd edition, 2022.

[10] Dimitri P. Bertsekas. *Lessons from AlphaZero for Optimal, Model Predictive, and Adaptive Control*. Athena Scientific, 2022.

[11] Dimitri P. Bertsekas. Newton's method for reinforcement learning and model predictive control. *Results in Control and Optimization*, 7:100121, 2022.

[12] Dimitri P. Bertsekas. Model predictive control and reinforcement learning: A unified framework based on dynamic programming. *IFAC-PapersOnLine*, 58(18):363–383, 2024.

[13] Dimitri P. Bertsekas and David A. Castanon. Rollout algorithms for stochastic scheduling problems. *Journal of Heuristics*, 5:89–108, 1999.

[14] Dimitri P. Bertsekas, John N. Tsitsiklis, and Cynara Wu. Rollout algorithms for combinatorial optimization. *Journal of Heuristics*, 3(3):245–262, 1997.

[15] Hyeong Soo Chang, Robert Givan, and Edwin K.P. Chong. Parallel rollout for online solution of partially observable Markov decision processes. *Discrete Event Dynamic Systems*, 14(3):309–341, 2004.

[16] Ismi Rosyiana Fitri and Jung-Su Kim. A nonlinear model predictive control with enlarged region of attraction via the union of invariant sets. *Mathematics*, 8(11):2087, 2020.

[17] Martin Herceg, Michal Kvasnica, Colin N. Jones, and Manfred Morari. Multi-Parametric Toolbox 3.0. In *Proc. of the European Control Conference*, pages 502–510, Zürich, Switzerland, July 17–19 2013. http://control.ee.ethz.ch/~mpt.

[18] Juan L. Jerez, George A. Constantinides, Eric C. Kerrigan, and Keck-Voon Ling. Parallel MPC for real-time fpga-based implementation. *IFAC Proceedings Volumes*, 44(1):1338–1343, 2011.

[19] Yuning Jiang, Juraj Oravec, Boris Houska, and Michal Kvasnica. Parallel MPC for linear systems with input constraints. *IEEE Transactions on Automatic Control*, 66(7):3401–3408, 2020.

[20] Mikael Johansson and Hamed Taghavian. Stable MPC with maximal terminal sets and quadratic terminal costs. *arXiv preprint arXiv:2406.02760*, 2024.

[21] Rudolf Emil Kalman. Contributions to the theory of optimal control. *Boletinde la Sociedad Matematica Mexicana*, 5(2):102–119, 1960.

[22] SS A Keerthi and Elmer 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*, 57(2):265–293, 1988.

[23] Mircea Lazar, W.P. Maurice H. Heemels, Siep Weiland, and Alberto Bemporad. Stabilization conditions for model predictive control of constrained PWA systems. In *2004 43rd IEEE Conference on Decision and Control (CDC)*, volume 5, pages 4595–4600. IEEE, 2004.

[24] Yuchao Li and Dimitri Bertsekas. Semilinear dynamic programming: Analysis, algorithms, and certainty equivalence properties. *arXiv preprint arXiv:2501.04668*, 2025.

[25] Yuchao Li, Karl H. Johansson, Jonas Mårtensson, and Dimitri P. Bertsekas. Data-driven rollout for deterministic optimal control. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 2169–2176. IEEE, 2021.

[26] Yuchao Li, Aren Karapetyan, John Lygeros, Karl H Johansson, and Jonas Mårtensson. Performance bounds of model predictive control for unconstrained and constrained linear quadratic problems and beyond. *IFAC-PapersOnLine*, 56(2):8464–8469, 2023.

[27] Yuchao Li and Anders Rantzer. Exact dynamic programming for positive systems with linear optimal cost. *IEEE Transactions on Automatic Control*, 2024.

[28] Johan Lofberg. YALMIP: A toolbox for modeling and optimization in MATLAB. In *2004 IEEE International Conference on Robotics and Automation*, pages 284–289. IEEE, 2004.

[29] D. Limón Marruedo, Teodoro Alamo, and Eduardo F. Camacho. Enlarging the domain of attraction of MPC controller using invariant sets. *IFAC Proceedings Volumes*, 35(1):47–52, 2002.

[30] David Q. Mayne, James B. Rawlings, Christopher V. Rao, and Pierre O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[31] Ugo Rosolia and Francesco Borrelli. Learning model predictive control for iterative tasks. A data-driven control framework. *IEEE Transactions on Automatic Control*, 63(7):1883–1896, 2017.

[32] Wei Zhang, Jianghai Hu, and Alessandro Abate. Infinite-horizon switched LQR problems in discrete time: A suboptimal algorithm with performance analysis. *IEEE Transactions on Automatic Control*, 57(7):1815–1821, 2011.