Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Fast Non-Parametric Learning to Accelerate Mixed-Integer Programming for Hybrid Model Predictive Control ⋆

Jia-Jie Zhu * **Georg Martius** **

* *Empirical Inference Department,*
Max Planck Institute for Intelligent Systems, Tübingen, Germany.
(e-mail: jzhu@tuebingen.mpg.de)
** *Autonomous Learning Group,*
Max Planck Institute for Intelligent Systems, Tübingen, Germany.
(e-mail: georg.martius@tuebingen.mpg.de)

Abstract: Today's fast linear algebra and numerical optimization tools have pushed the frontier of model predictive control (MPC) forward, to the efficient control of highly nonlinear and hybrid systems. The field of hybrid MPC has demonstrated that exact optimal control law can be computed, e. g., by mixed-integer programming (MIP) under piecewise-affine (PWA) system models. Despite the elegant theory, online solving hybrid MPC is still out of reach for many applications. We aim to speed up MIP by combining geometric insights from hybrid MPC, a simple-yet-effective learning algorithm, and MIP warm start techniques. Following a line of work in approximate explicit MPC, the proposed learning-control algorithm, LNMS, gains computational advantage over MIP at little cost and is straightforward for practitioners to implement.

Keywords: Model Predictive Control of Hybrid System, Machine Learning, Learning for Control, Nonparametric learning, Mixed-Integer Programming

1. INTRODUCTION

The presence of hybrid dynamical systems, defined as systems whose state evolution is governed by both continuous dynamics (flow) and discrete events (jump), is ubiquitous in physical systems. Consider a legged robot that dynamically navigates the environment. It must not only coordinate the motion of its joints but also decide when to make and break contact at its end-effectors. This inherent coupling of *discrete events* and *continuous decision-making* challenges optimization-based control designs such as model predictive control (MPC). One of the greatest difficulties is perhaps the combinatorial growth of computational complexity caused by mode switches. This paper aims to answer an intuitive question: what information can be *learned* when solving hybrid MPC problems over and over again? Our *main insight* is: The geometric structure of hybrid MPC solutions — the mode sequences — can be learned by a nonparametric learning algorithm from previously visited states.

Instead of storing exact solution structures, e.g., polyhedral partitions, we store samples of visited states to greatly speed up hybrid MPC at little extra computational effort.

As an illustrative example, consider the dynamics of a cart in front of a wall, as shown in Figure 1 (adapted from Marcucci and Tedrake (2019)). It can be thought of as an actuated version of a bouncing ball, whose dynamics is continuous, except when it interacts with the wall.

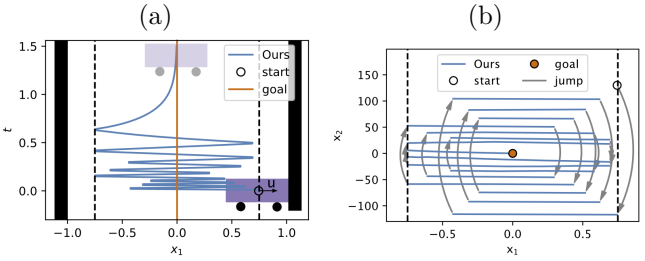

Fig. 1. Illustrative example of Cart with two walls. (a) Cart starts at high velocity and bounces (discrete timesteps), note the vertical time axis. (b) same as in (a) but in state space (x_1:position and x_2: velocity). The gray lines with arrow denote jumps across switching modes. See Sec. 4 for details.

During continuous dynamics periods, tools from continuous optimization such as sensitivity analysis, that give rise to efficient *online* methods Ferreau et al. (2008) or sequential quadratic programming (SQP)-based methods Diehl et al. (2002), can be used to control the system in a receding horizon fashion. However, when contact happens, the structured solution of the sensitivity analysis loses validity. In such cases, online hybrid MPC using mixed-integer programming (MIP) can be an effective way of making optimal discrete decisions (e.g. whether or not to willingly impact the wall). However, solving MIP online is computationally heavy.

⋆ Jia-Jie Zhu is supported by funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 798321.

We hope to contribute in the following aspects:

- We propose a simple-yet-effective fast learning algorithm to warm-start mixed-integer programs for MPC in PWA hybrid systems. As learning progresses, it greatly reduces the computational cost.
- The properties of the proposed algorithm are exploited to allow i) post-processing of solutions improving towards optimality and early-termination of MIP; ii) straightforward implementation for practitioners.

Notation In this paper, $x \in R^d$ denotes a column vector of dimension d and $x^\top$ its transpose. N denotes the horizon of an optimal control problem (OCP) and n the number of samples of the learning algorithm. x_p is a parameter (e. g. current state estimation) of an optimization problem. e_i denotes the one-hot vector of suitable dimensions where the i-th element is 1 and rest are zeros. A sequence of modes of a discrete dynamical system is denoted as $\mathcal{M} = \{m_t\}_{t=1}^N$ where each $m_t = (\mu_t^1, \mu_t^2, \dots, \mu_t^{n_M})$ is a system mode at time t and μ_t^i a binary variable. A feasible solution of an optimization problem is one at which all constraints are satisfied.

2. PRELIMINARIES

2.1 Model predictive control for piecewise affine hybrid systems

In this paper, we consider a discrete-time optimal control problem (OCP). The goal is to compute a sequence of control actions $\{u_t\}_{t=1}^N$ to steer the system state x to the origin.

$$\begin{aligned} \underset{u_t, t=0,\dots,N-1}{\text{minimize}} \quad & \sum_{t=0}^{N-1} \left(x_t^\top Q x_t + u_t^\top R u_t\right) + x_N^\top P x_N, \\ \text{subject to} \quad & x_{t+1} = f(x_t, u_t),\ t = 0, \dots, N-1, \\ & h_t(x_t, u_t) \le 0,\ t = 0, \dots, N-1, \\ & x_0 = x_p, \end{aligned} \tag{1}$$

where x_p is the current state, $x_t^\top Q x_t + u_t^\top R u_t$ is the stage cost and $x_N^\top P x_N$ the terminal cost. f is the dynamics that describes the evolution of the system over time and h_t's are the constraints (e. g., bounds on control input u_t).

Model predictive control (MPC) Richalet et al. (1978); Rawlings and Mayne (2009) solves OCP (1) at every sampling time and implement the first control input u_0.

Consider the running example of the simple cart and wall in Fig. 1 again. In this case, there is no single dynamics function f that governs on the whole state-space — this is a hybrid system. One approach to solve hybrid MPC is to formulate the dynamics constraints using piecewise affine (PWA) formulations (Sontag (1981)). PWA models can describe general nonlinear dynamics while enjoying the advantage of having a wide range of linear control tools,

$$x_{t+1} = A^i x_t + B^i u_t, \text{ for } x \in \mathcal{C}_i, i = 1 \dots, n_M. \tag{2}$$

Mathematically, solving OCP (1) under PWA dynamics (2) is typically formulated as a mixed-integer program (MIP), due to the presence of both continuous and discrete variables. We consider the big-M formulation of MIP:

$$\begin{aligned} |x_{t+1} - A^i x_t - B^i u_t| &\le (1 - \mu_t^i) M \\ h(x_t, u_t) &\le (1 - \mu_t^i) M \\ \sum_i \mu_t^i = 1,\ \mu_t^i &\in \{0, 1\}, \quad \forall i, t, \end{aligned} \tag{3}$$

where $\mu_t^i, i = 1, 2, \dots n_M$ are the auxiliary integer variables. At time t, it is easy to see that the system is in mode i: $m_t = e_i$ if and only if $\mu_t^i = 1$. A mode sequence $\mathcal{M} = \{m_t\}_{t=1}^N$ thus corresponds to a set of integer decision variables of the hybrid OCP (1),(3).

We use $\mathcal{X}^\mathcal{M}$ to denote the feasible region (not to be confused with the critical region) of an OCP given $\mathcal{M}$: the set of parameters x_p where a mode sequence $\mathcal{M}$ is feasible (i. e., OCP (1)(3) has a solution with this fixed $\mathcal{M}$). It is easy to see that one mode sequence may not be feasible for the whole state space in hybrid systems. For example, consider a PWA control law of the cart example in Fig. 2 (a), the mode sequence associated with the region (colored red) in the upper right corner is not feasible once we cross the boundary of this region. This corresponds to a different set of integer solutions to OCP (1)(3). The geometric property of feasible regions can be summarized as follows: i) Feasible regions are convex polyhedra. ii) Each feasible region corresponds to an integer solution to the hybrid OCP(1)(3) (and hence a mode sequence). iii) They may or may not overlap.

One insight commonly exploited is that the hybrid OCP (1)(3) becomes a continuous (convex) optimization problem if the integer variables in (3) are fixed to a *feasible mode sequence* $\mathcal{M}$. Thus, the problem can be solved by extremely efficient algorithms. Hence, if we store the feasible region $\mathcal{X}^\mathcal{M}$ and associated $\mathcal{M}$ offline, during the online runtime of MPC, we only need to *look up which* $\mathcal{X}^\mathcal{M}$ *the current state* x_p *belongs to and fix the mode sequence to associated* $\mathcal{M}$.

2.2 Voronoi tessellation and nearest neighbor classification

One of the simplest yet effective machine learning algorithms is nearest neighbor (NN) classification [Cover et al. (1967)]. It stores all data in terms of input-output pairs. For a new data point, the nearest neighbor is retrieved and its associated output is returned. Let D be the set of data points with entries $x_i \in \mathbb{R}^d$. The runtime complexity depends on the underlying storage structure: with tree structures [Omohundro (1989)] the retrieval is $O(d \log(|D|))$ (while building the indexing structure requires $O(d|D| \log(|D|))$ operations.). Interestingly, the explicit MPC approach in Jones et al. (2006) uses Voronoi diagram as a way to reduce the complexity of online execution. Their online look-up complexity is logarithmic in number of regions whereas our setting scales logarithmically with the number of samples. This gives us a flexible trade-off: we can choose to store fewer samples with faster look-up time but more likely to need to solve MIP, or to store more samples for the opposite, or somewhere in between.

Fig. 2 (b,c,d) is an intuitive example of NN classification resulting in the Voronoi diagram approximating the true region partition. Such approximation is the *key motivation* to our method: we directly store sampled points rather than keeping track of polyhedral regions in Fig. 2 (a).

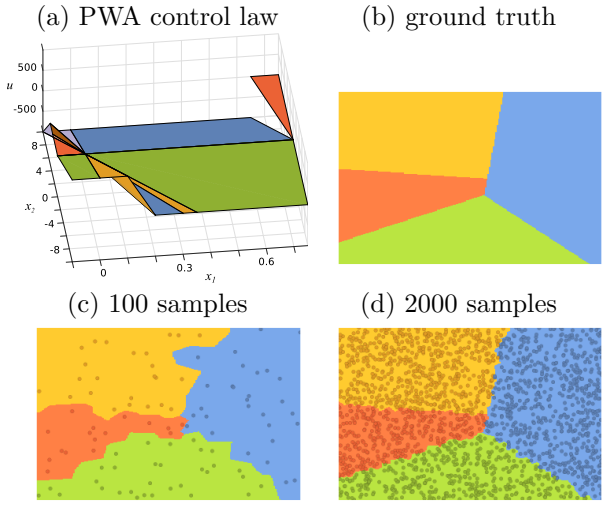

Fig. 2. (a) piecewise affine control law in hybrid systems (cart example). The z-axis is the control input while x,y-axes are states. Notice the control law is discontinuous. (b) Ground truth NN partition of regions. (c,d) Evolution of Voronoi regions induced by NN with increasing numbers of sampling points.

Sampled points offer a straightforward alternative to solving multi-parametric (quadratic) programs or performing polyhedron operations.

Remark One key feature of NN is that it belongs to the class of "*lazy learning*" algorithms; it does not require an additional training process such as neural networks do. This important aspect facilitates the *fast-learning* capability of the learning-controller we shall propose next.

3. METHOD

3.1 Approximate hybrid MPC by learning from nearest neighbor mode sequences

Drawing form geometric insights discussed in Sec. 2, we propose a concise approximation approach based on the nearest neighbor (NN) rule. Our *main idea* is simple: use Voronoi tessellation induced by the NN classifier to approximate feasible regions for the hybrid MPC control law. The data for the NN classifier is given by the mode sequence $\mathcal{M}$ based on the *sampled* states.

The learning-control algorithm works as follows. We query the NN classier for the mode sequence $\mathcal{M}$ for a given state x_p based on which feasible region it falls in (a simple NN look-up). We then use $\mathcal{M}$'s corresponding integer solution to warm-start the MI(Q)P (1),(3). If the modes are feasible, the computation is reduced to a quadratic program that can be solved with state-of-the-art solvers in the order of microseconds, e. g. Houska et al. (2011). The algorithm is explained in Alg. 1. For the rest of the paper, we refer to the proposed method as *approximate MPC by learning from nearest-neighbor mode sequences* (LNMS).

3.2 Theoretical properties

We present the theoretical analysis for Alg. 1. Typical MPC analysis concerns feasibility (whether the OCP has a solution) and stability (whether the closed-loop system can be bounded around a set-point or within a set). For (exact) hybrid MPC, many properties are inherited from nominal MPC, i. e. *recursive feasibility (and therefore asymptotic stability) is guaranteed* by choosing the appropriate terminal cost (Lyapunov function) and terminal constraint (control invariant set). We refer interested readers to Borrelli et al. (2017), Section 17.8.1 for detailed theoretical analysis regarding HMPC stability. As Alg. 1 executes the regular hybrid MPC as a warm-start improvement in the case of predicted modes being infeasible, it inherits the feasibility (hence stability) properties from regular hybrid MPC. We will focus on the computational aspect, i.e., the speed-up over exact hybrid MPC.

Algorithm 1 LNMS: Online Approximate MPC with Nearest Neighbor Learning

1: Given: sample initial state $x(0) \sim P_0$ where P_0 is the initial state distribution. Optional: an initial dataset $\mathcal{D} = \mathcal{D}_0$ of sampled points.
2: **loop**
3: Get current state estimation x_p .
4: Query the nearest neighbor classifier (with dataset $\mathcal{D}$) for the mode sequence $\mathcal{M} = \{m_i\}_{i=1}^N$
5: Solve hybrid OCP (1)(3) with integer variables $\{m_i\}_{i=1}^N$ as warm-start solution. Terminate once a feasible solution is found.
6: Add the $(x_p, \mathcal{M}^*)$-pair to the dataset $\mathcal{D}$, where $\mathcal{M}^*$ is the integer solution obtained last step.
7: Apply the first control solution u_0^* to the system.
8: **end loop**

Proposition 3.1. Given dataset $\mathcal{D}$ of size n obtained by Alg. 1, let P_n^{MIP} denote the probability of hybrid MPC, whose initial state $x(0) \sim P_0$, having to execute MIP solver in Step 5 of Alg. 1. Then,

$$P_n^{\text{MIP}} \to 0 \text{ as } n \to \infty.$$

Proof For every feasible state, we consider the MIQP solver to be an oracle. It achieves the minimum possible classification error rate $P^* = 0$ as we assume the solver is always able to return feasible mode sequences for feasible states. Let $P_n(\text{error})$ denote the probability that the NN classifier in Algorithm 1 predicts an infeasible mode sequence (classification error). Use the NN-convergence bound $P^* \leq \lim_{n\to\infty} P_n(\text{error}) \leq P^*(2 - \frac{c}{c-1}P^*) = 0$, where c is the number of classes (c.f. Cover et al. (1967)). Since we only execute MIP solver if the mode sequence is infeasible, $P_n^{\text{MIP}} = P_n(\text{error})$. The conclusion follows. □

This proposition implies that, with an increasing amount of samples, we get a faster and faster controller. This is empirically validated in the experiment section.

3.3 Improving optimality of sampled mode sequences by warm-starting MIP

As an approximate MPC algorithm, LNMS only aims to produce feasible instead of optimal mode sequence prediction. In practice, one may also choose to early terminate the MIP at a feasible solution since the most costly computation is to produce a tight dual bound to certify optimality — a good solution may be available much sooner [cf. Bertsimas et al. (2016)]. Both those

two sources contribute to suboptimal mode sequences for sampled points.

However, using the instance-based nature of the NN classifier, we can improve on the resulting controller by simply "relabeling" the stored samples via warm-starting techniques of MIP.

Intuitively, this process picks a point from the stored samples and feed its mode sequence as warm-start to the MIP solver, see Alg. 2). As it is already feasible, the solver will always return a mode sequence that has the same or lower cost. Hence we have the intuitive results in Proposition 3.2 (omitting the straightforward proof). We provide examples of this process in Sec. 4 (Fig. 3(d), Fig. 4(c)).

Algorithm 2 LNMS: Solution optimality improvement

1: **loop**
2: Pick a state-label $(x_0, \mathcal{M})$-pair from dataset $\mathcal{D}$
3: Solve OCP (1) with integer variables $\mathcal{M} = \{m_i\}_{i=1}^N$ as a warm-start incumbent solution. Optionally, early terminate after the computational budget reached.
4: Relabel this sample and add the new $(x_0, \mathcal{M}*)$-pair to $\mathcal{D}$, where $\mathcal{M}^*$ is the integer solution from step 3..,
5: **end loop**

Proposition 3.2. If the re-labeling using Algorithm 2 is done to full MIP optimality for all stored data points, then Algorithm 1, with dataset $\mathcal{D}$ and initial state $x(0) \sim P_0$, recovers the exact optimal solution of hybrid MPC as the number of samples $|\mathcal{D}| \to \infty$.

4. NUMERICAL EXPERIMENTS

Experiment setup We implement the MPC controller in Python with Gurobi as the optimization solver. In NN learning, we use a simple weighted Euclidean distance. We consider two examples using a cart with one and two walls, see Fig. 1(a) (in the one wall case the left wall is missing), and a pendulum with an elastic wall, see Fig. 6(a).

4.1 Example 1. Cart-wall contact dynamics

The dynamics equation for the cart-wall example in Fig. 1 is described by the following PWA formulation:

$$\begin{cases} x_1^+ = x_1 + x_2\ \Delta t, \quad x_2^+ = x_2 + \dfrac{u}{m}\ \Delta t, & \text{if } x \in \mathcal{C}_1 \\ x_1^+ = x_1, \quad x_2^+ = -\epsilon x_2, & \text{if } x \in \mathcal{C}_2 \end{cases} \tag{4}$$

where m is the cart mass (set to 1.0) and ϵ is the coefficient of restitution (set to 0.9). It could be thought of an actuated version of a bouncing ball — a classic hybrid system. $\mathcal{C}_1 = \{x_1 + x_2\ \Delta t < x_{\text{wall}}\}$ denotes the state space where the dynamics is the double integrator and $\mathcal{C}_2 = \{x_1 + x_2\ \Delta t \geq x_{\text{wall}}\}$ denotes the state space where the contact with the wall happens. In our case $x_{\text{wall}} = 0.75$ and the discretization is set to $\Delta t = 0.01$ for all our experiments. The PWA dynamics for the cart with two walls is a straightforward extension.

We synthesize MPC with horizon $N = 10$ (no early termination of the MIP). The cost weights in Eqn. (1) are $Q = I_2, R = 0.001$. $S = \beta \cdot K$ where K is the solution to the algebraic Riccati equation for the system in mode $\mathcal{C}_1$, we choose $\beta = 1000$ for faster convergence behavior to the attractor but this is not a crucial choice. In the first experiment, we use terminal costs instead of terminal sets following common practice in MPC applications (cf. Rawlings and Mayne (2009)).

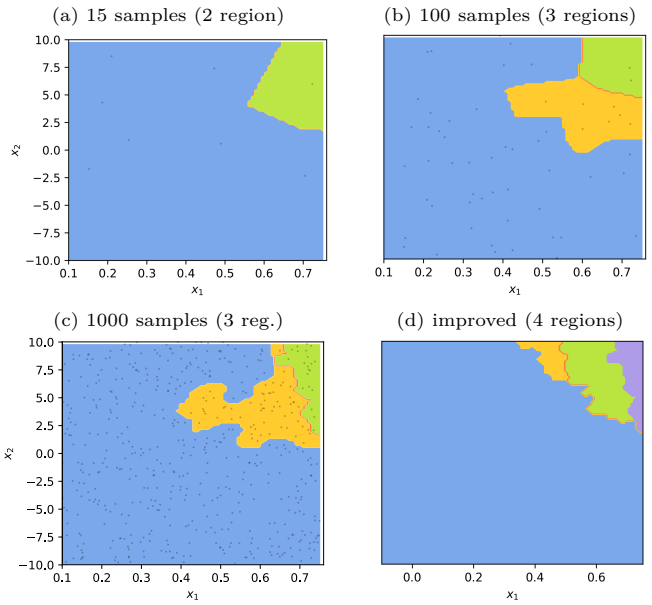

Fig. 3. PWA control law resulting from LNMS in Alg. 1 associated with different sample sizes shown in (a-c). Panel (d) displays the optimal control law after applying the improvement scheme in Sec. 3.3.

In this experiment, we start from initial states $x(0)$ randomly sampled within the region $[0.1, 0.75] \times [-10, 10]$. For each initial state, the hybrid OCP (1)(3) is solved by LNMS in Alg. 1. As a result of the non-parametric learning Alg. 1, we obtained a set of samples $\mathcal{D}$. Those samples store the solution structure (feasible mode sequences) of LNMS and are the key to speeding up MIP. They are displayed in Fig. 3 (a)-(c) (gray dots) with different sample size.

We visualize the resulting PWA control law in Fig. 3 (a)-(c). Each color patch indicates a region with different affine control laws. Notably, Fig. 3 (a)-(c) shows the evolution of the PWA control law as the learning-controller gathers more samples. We refer to our earlier discussion on Fig. 2 for how to interpret the visualization. We plot the resulting closed-loop state trajectories in Fig. 5 (with different initial states). The trajectories obtained by LNMS are identical to the exact hybrid MPC control law.

Scaling up to complex control law structure It is known that constrained control law with numerous mode switches has a complex structure. To demonstrate that LNMS scales to such cases, we consider the cart example with two walls, higher initial velocities, and an MPC prediction horizon of $N = 25$. Furthermore, we impose the input constraint ($|u| \leq 10$). The resulting control law is significantly more complex than the previous example due to a large number of mode sequences and constraints.

A direct consequence is that the MIP is challenging to solve to full optimality. Therefore, early termination is necessary — we use a 5 sec early-termination threshold.

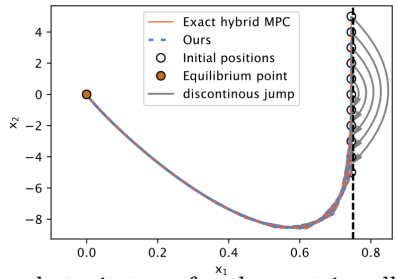

Fig. 5. Example trajectory for the cart-1-wall.

The HMPC control law is shown in Fig. 4(a,b). We observe a more complex structure of the control law — 493 regions of different mode sequences correspond to different PWA control laws.

Improving solution optimality with Alg. 2 As discussed, Alg. 1 seeks a feasible solution to speed up and warm-start MIP. The resulting solution might be a sub-optimal control law. However, due to the non-parametric nature of our learning algorithm, we can simply post-process the stored sample points to improve the optimality of the learning-controller. We apply the optimality-improvement scheme in Alg. 2 to both the one-wall and two-wall cases. The resulting *optimal* control law is shown in Fig. 3(d) for one-wall and Fig. 4(c) for two-walls. Interestingly, we observe that the control law in Fig. 4(c) is simplified due to merging via comparing optimal cost-to-go of different PWA control laws. This is similar to the procedures in merging piecewise partitions in explicit MPC (cf. Borrelli et al. (2017)).

4.2 Example 2. Elastic wall with variable duration contact

The elastic pendulum environment, see Fig. 6(a), is adapted from Marcucci et al. (2017). The dynamics is given by

$$\dot{x} = \begin{cases} A_1 x + B_1 u, & \text{if} \quad (x,u) \in \mathcal{D}_1, \\ A_2 x + B_2 u + c_2, & \text{if} \quad (x,u) \in \mathcal{D}_2, \end{cases} \tag{5}$$

with

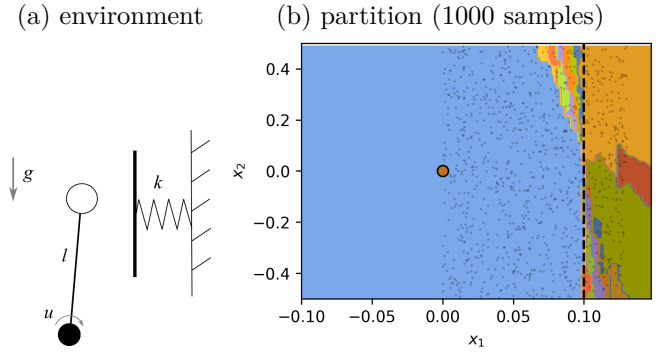

Fig. 6. Pendulum with an elastic wall. (a) Illustration of the environment. (b) Partitioning from Alg. 1 resulting in 18 regions.

$$A_1 = \begin{bmatrix} 0 & 1 \\ g/l & 0 \end{bmatrix}, \quad B_1 = \begin{bmatrix} 0 \\ 1/ml^2 \end{bmatrix},$$
$$A_2 = \begin{bmatrix} 0 & 1 \\ g/l - k/m & 0 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 0 \\ 1/ml^2 \end{bmatrix}, \quad c_2 = \begin{bmatrix} 0 \\ kd/ml \end{bmatrix},$$
$$\mathcal{D}_1 = \{(x,u) \mid x_1 \leq d/l, x_{\min} \leq x \leq x_{\max}, u_{\min} \leq u \leq u_{\max}\},$$
$$\mathcal{D}_2 = \{(x,u) \mid x_1 > d/l, x_{\min} \leq x \leq x_{\max}, u_{\min} \leq u \leq u_{\max}\}$$

The key difference between the two examples is that the cart-wall system makes and breaks contact instantaneously while the pendulum with an elastic wall allows variable contact duration, thus resulting in different mode sequences. Due to this difference, we use a control invariant set as terminal set in this experiment 2, avoiding the terminal mode "gets stuck in the wall".

Similar to the previous experiment, we execute Alg.1 to obtain the LNMS learning-controller and collected data samples. They are plotted in Fig. 6. An example closed-loop trajectory in Fig. 7. We can see that, compared with the previous example, the LNMS trajectories differ from the optimal solution obtained by running MIP to full optimality (no time limit). This is caused by the fact that variable-during contact results in suboptimal mode sequences being reused by Alg. 1. Again, this may be either improved by Alg. 2 or a shift-started mode sequence in HMPC.

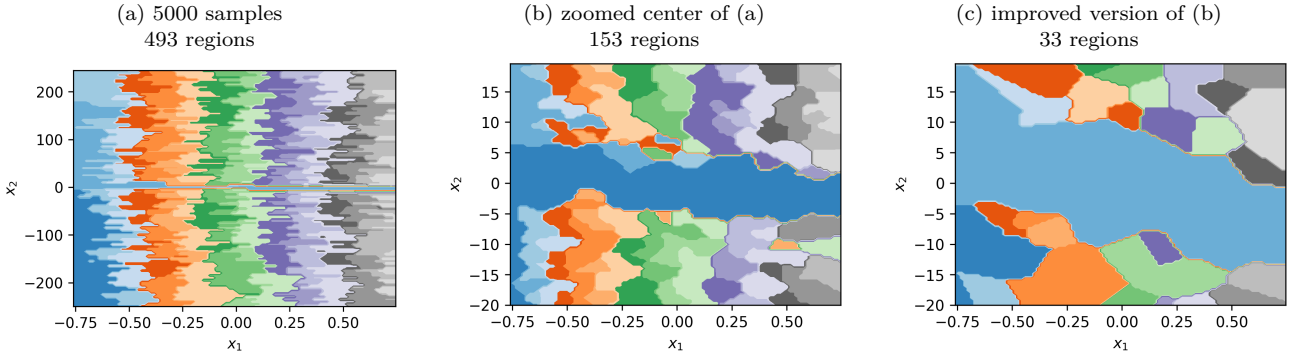

Fig. 4. Control law of LNMS in the cart with two walls. (a) PWA control law of Alg. 1 with early termination of 5.0s. Different color patches denote different affine control policies. (b) zoomed view on the center area of (a). (c) after applying the warm-start relabeling method using 5s improvement proposed in Sec. 3.3.

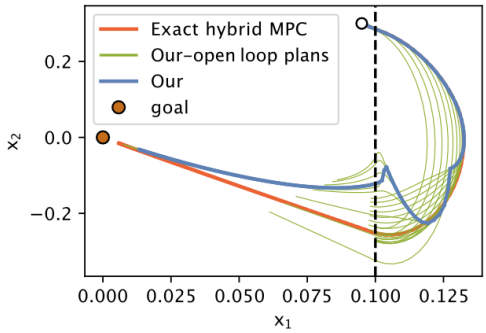

Fig. 7. Example trajectory with MPC open-loop plans for the pendulum with an elastic wall.

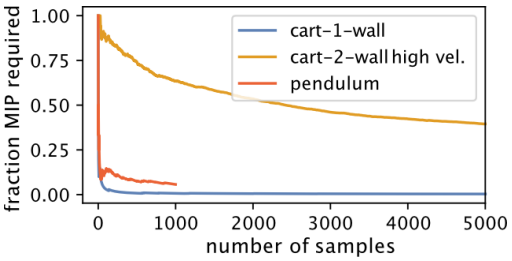

Fig. 8. Percentage of MIP runs during the execution of LNMS in different environments. This empirically validates Proposition 3.2. A preliminary wall-clock comparison is given in the appendix.

4.3 Computational complexity reduction

We now discuss how LNMS in Alg. 1 can greatly speed up the HMPC solution method. The results here empirically validate the theory in Sec. 3.2. We compare the solution time of LNMS in Alg. 1 with that of the exact MIQP solution of hybrid MPC (1)(3) and report the relative run-time (time of Alg. 1)/(time of exact MIQP)%). This comparison is shown in Fig. 8 for all experiments.

We observe that, as Alg. 1 collects more data samples, only a small fraction of mixed-integer optimization needs to be solved by branch&bound. As there is a big gap in computation time between branch&bound and continuous programs, the proposed scheme drastically improve computational efficiency. The actual wall clock time may vary greatly depending on the specific learning and MPC code implementation. Nonetheless, we include a wall clock-time comparison in Appendix A. This experiment demonstrates the answer to our question posed in Section. 1, of what can be *learned* from HMPC runs in the same environment.

5. RELATED WORK

Explicit MPC [Bemporad et al. (2000, 2002); TøNdel et al. (2003)] seeks to offload online computation to offline. Its insight is that 1) the optimal control law of an OCP with quadratic objective linear constraints is piecewise affine state-feedback, i. e. $u(x) = F^i x + G^i, x \in \mathcal{R}_i$, where $\mathcal{R}_i$ are convex polyhedra (referred to as critical regions) and 2) there may be exponentially many such regions need to be computed and stored. In the context of hybrid systems, different regions correspond to different (switching) mode sequences, e. g. in-contact or not-in-contact. It is then possible to carry out the mode sequence enumeration offline and store the state-feedback-affine policy. Despite its elegant theory, explicit MPC may incur considerable complexity in 1) both online look-up and offline computation and 2) storage of the optimal partitions. Many approximate MPC algorithms, e. g. Zeilinger et al. (2011); Domahidi et al. (2011), trade off optimality for fast run-time and feasibility. The goal is to avoid exhaustively computing, storing, and comparing regions.

Motivated by this challenge, many approaches have been proposed in the literature to solve such hybrid control problems. For example, Mordatch et al. (2012) suggest the use of soft contact and dynamics models to smooth the dynamics, thus allowing the use of gradient-based numerical optimization. The work of Posa and Tedrake (2013) exploits homotopy methods to compute solutions that gradually converge from relaxed to accurate ones.

Closely related to this work, the authors of Marcucci et al. (2017) use insights from MPC stability proof to simplify explicit MPC from critical regions to inner approximations of feasible regions, resulting in significant computational saving. Our approach further simplifies the region computation and storage by the non-parametric learning algorithm, sidestepping polyhedron operations completely.

Very recently, a few works show promises in applying machine learning to hybrid control. Deits et al. (2018) experimented with value function and policy learning in hybrid systems. Our method in this work can also serve as an efficient oracle for policy training. This work shares similar features with Hogan et al. (2018) in reducing computational cost by learning mode sequences. However, we exploit the non-forgetting property of non-parametric methods to perform online learning, as well as the geometric structure of hybrid MPC solutions to avoid exhaustive offline MIP runs.

There is a thread of works studying the topic of "learning to branch and bound." [He et al. (2014); Khalil et al. (2016); Rachelson et al. (2010)]. We share the common thread of using *function approximation*. However, those learning methods are mostly *black-box* approaches. In contrast, our insight is rooted in the understanding of HMPC solution structure.

6. DISCUSSION

This paper presents a new approach that learns from prior experiences to accelerate MIP for hybrid MPC. Our choice of non-parametric learning algorithms is motivated by the theoretical understanding of (H)MPC solutions. For example, neural network training is parametric and cannot easily deal with wrong points near the hybrid mode boundaries. It also looses *stability guarantees* of (H)MPC. In contrast, our method does not need training and preserves the stability of the (H)MPC. It achieves *speed-up virtually for free* by using adaptive samples while is easy-to-implement.

It is known that if absolute optimality is the goal instead of feasibility which was the aim of this paper, the boundaries of *optimal mode sequence partition* may be quadratic rather than affine (cf. Borrelli et al. (2017)).

While LNMS still works, an interesting extension is to generalize the notion of (weighted) Euclidean distance, possibly via kernelized nearest neighbor.

One potential application of LNMS is to serve as an efficient supervised learning oracle for training an imitation learning policy $\pi : x \mapsto u$, such as in Deits et al. (2018); Hogan et al. (2018). Their computational cost of training may be greatly reduced by adopting LNMS.

ACKNOWLEDGEMENTS

We would like to thank Majid Khadiv and Brahayam Ponton for their helpful discussions, Marin Vlastelica Pogancic for the help in producing Fig. 5, Vincent Berenz and Lidia Pavel for their help in equipment and facility support.

REFERENCES

Bemporad, A., Borrelli, F., and Morari, M. (2000). Piecewise linear optimal controllers for hybrid systems. In *Proceedings of the 2000 American Control Conference. ACC*, volume 2, 1190–1194. IEEE.

Bemporad, A., Morari, M., Dua, V., and Pistikopoulos, E.N. (2002). The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1), 3–20.

Bertsimas, D., King, A., Mazumder, R., et al. (2016). Best subset selection via a modern optimization lens. *The annals of statistics*, 44(2), 813–852.

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

Cover, T.M., Hart, P., et al. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1), 21–27.

Deits, R., Koolen, T., and Tedrake, R. (2018). LVIS: Learning from value function intervals for contact-aware robot controllers. *arXiv preprint arXiv:1809.05802*.

Diehl, M., Bock, H.G., Schlöder, J.P., Findeisen, R., Nagy, Z., and Allgöwer, F. (2002). Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4), 577–585.

Domahidi, A., Zeilinger, M.N., Morari, M., and Jones, C.N. (2011). Learning a Feasible and Stabilizing Explicit Model Predictive Control Law by Robust Optimization. *Conference on Decision and Control*, 513–519.

Ferreau, H.J., Bock, H.G., and Diehl, M. (2008). An online active set strategy to overcome the limitations of explicit mpc. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 18(8), 816–830.

He, H., Daume III, H., and Eisner, J.M. (2014). Learning to search in branch and bound algorithms. In *Advances in neural information processing systems*, 3293–3301.

Hogan, F.R., Grau, E.R., and Rodriguez, A. (2018). Reactive planar manipulation with convex hybrid mpc. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 247–253. IEEE.

Houska, B., Ferreau, H.J., and Diehl, M. (2011). ACADO toolkit–an open-source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, 32(3), 298–312.

Jones, C.N., Grieder, P., and Raković, S.V. (2006). A logarithmic-time solution to the point location problem for parametric linear programming. *Automatica*, 42(12), 2215–2218.

Khalil, E.B., Le Bodic, P., Song, L., Nemhauser, G., and Dilkina, B. (2016). Learning to branch in mixed integer programming. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Marcucci, T., Deits, R., Gabiccini, M., Bicchi, A., and Tedrake, R. (2017). Approximate hybrid model predictive control for multi-contact push recovery in complex environments. In *IEEE-RAS International Conference on Humanoid Robotics*, (Humanoids), 31–38. IEEE.

Marcucci, T. and Tedrake, R. (2019). Mixed-integer formulations for optimal control of piecewise-affine systems. In *Proceedings of the Intl. Conference on Hybrid Systems: Computation and Control*, 230–239. ACM.

Mordatch, I., Todorov, E., and Popović, Z. (2012). Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics*, 31(4).

Omohundro, S.M. (1989). Five balltree construction algorithms. Technical report, UC Berkeley.

Posa, M. and Tedrake, R. (2013). Direct trajectory optimization of rigid body dynamical systems through contact. In *Algorithmic Foundations of Robotics X*, 527–542. Springer.

Rachelson, E., Abbes, A.B., and Diemer, S. (2010). Combining mixed integer programming and supervised learning for fast re-planning. In *International Conference on Tools with Artificial Intelligence*, 63–70. IEEE.

Rawlings, J.B. and Mayne, D.Q. (2009). *Model predictive control: Theory and design*. Nob Hill Pub., Wisconsin.

Richalet, J., Rault, A., Testud, J., and Papon, J. (1978). Model predictive heuristic control. *Automatica (Journal of IFAC)*, 14(5), 413–428.

Sontag, E. (1981). Nonlinear regulation: The piecewise linear approach. *IEEE Transactions on Automatic Control*, 26(2), 346–358.

TøNdel, P., Johansen, T.A., and Bemporad, A. (2003). An algorithm for multi-parametric quadratic programming and explicit mpc solutions. *Automatica*, 39(3), 489–497.

Zeilinger, M.N., Jones, C.N., and Morari, M. (2011). Real-time suboptimal model predictive control using a combination of explicit MPC and online optimization. *IEEE Trans. on Automatic Control*, 56(7), 1524–1534.

Appendix A. WALL-CLOCK COMPARISON

This section corresponds to the computation time benchmark in Fig. 8. We show the results using a straightforward implementation of LNMS in Python Sklearn library.

Number of OCPs	LNMS runtime (s)	Regular MIP
10	**4.07**	69.02
100	**69.28**	156.25
500	**249.53**	1186.90

We generate OCPs (with different initial states). There is a big gap in solving speed between MIP and continuous programs. Our method is significantly faster ($\sim 2 - 17X$ speed-up). Speed-ups are expected values across all runs. Note, that this is done without code optimization on our side while MIP is solved using Gurobi – further putting LNMS in a disadvantage.