Dynamic Programming using Tensor Approximation for Contact-rich Manipulation

Teng Xue*[1,2], Suhan Shetty*[1,2], and Sylvain Calinon[1,2]

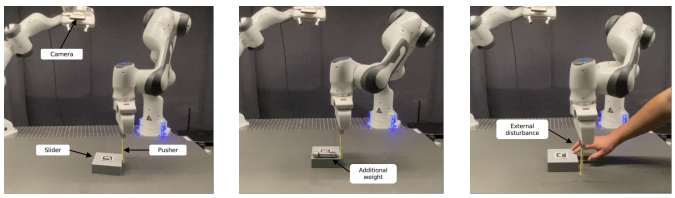
***Abstract*—Contact-rich manipulation is a challenging task for robot planning and control, due to the varying contact modes involved and the resultant uncertainties that arise from the contact points. Approximate Dynamic programming (ADP) is a promising approach for such problems as it can handle hybrid systems. Most existing methods in ADP that use neural networks for function approximation and gradient-based optimization procedures for policy retrieval struggle to handle the hybrid nature of contacts. In this work, we present a gradient-free, low-rank tensor approximation approach using Tensor Train (TT) to approximate the value function. The associated numerical optimization techniques for functions in TT format further allow performing optimization over both continuous and discrete variables, hence allowing handling hybrid systems. We demonstrate the effectiveness of our approach on a non-prehensile manipulation task with hybrid states and actions in both simulation and the real world. A video describing our work can be found at: https://youtu.be/ZrShvUQAyIs.**

***Index Terms*—approximate dynamic programming, tensor-train, contact-rich manipulation.**

I. Introduction

Motion planning and control of systems involving contact with the environment is a challenging task for model-based control techniques. This study focuses on the planar pushing task, which requires joint logistic and geometric planning over diverse interaction modes related to different types of contact and contact interaction [6]. For example, to push an object, a prerequisite is to decide how much force should be applied, and which point/surface to push. Moreover, in some cases such as pushing an object with a small distance to the target but a large orientation error, relying on a single fixed face is not feasible. Therefore, a sequence of face switching is required, as well as the corresponding contact mode schedule. Thus this problem has become a benchmark problem for the planning and control of hybrid systems.

The objective is to push a block with the option to switch both the contact mode and the face of the object to be used as contact. To tackle this, previous methods such as mixed integer programming [2] and hybrid Differential Dynamic Programming [1] have been proposed, but both still struggle to cope with the computational complexity involved in solving this problem. The state-triggered contact-implicit approach in [5] demonstrated promising performance by heuristically designing constraints for the hybrid system, but selecting a suitable set of constraints that allows for dynamic responsiveness to the environment can be difficult, and the method is not generalizable.



(a) Reaching (b) with add. weight (c) with disturbance

Fig. 1. Pusher-slider system where the robot pushes an object by contact switching. Three experiments were performed: a) Reaching: The block is pushed by the end-effector from initial state $\boldsymbol{q}_{s_0}$ to the origin. b): Reaching with additional weight: The block is pushed by the end-effector from $\boldsymbol{q}_{s_0}$ to the origin, but with an additional weight on the block, leading to nonuniform friction distribution. c) Reaching with external disturbance: The block is pushed by the end-effector from $\boldsymbol{q}_{s_0}$ to the origin, disturbed by moving around $\boldsymbol{q}_{\text{dist}} = [0.1\text{m } 0.03\text{m } 90°]^\top$.

Dynamic Programming (DP) is a powerful tool for motion planning and control that can potentially deal with hybrid systems. However, as the complexity of the problem increases, classical DP algorithms become less efficient. To overcome this challenge, Approximate DP (ADP) was introduced, which uses function approximation techniques to reduce the computational burden of DP. Neural Networks (NNs) have been a popular choice for function approximation, however, they lose convergence guarantees, resulting in heuristic approaches requiring laborious work in training them. Moreover, to the best of our knowledge, ADP algorithms using NN have not been able to tackle problems beyond classical problems of pendulum swing-up and cart-pole problems often involving only one-dimensional continuous action. To overcome this issue, Reinforcement Learning (RL) using NN has been a popular alternative to ADP. However, RL comes with other challenges to cope with generalizability and robustness. In this work, we focus on ADP for hybrid systems, and we propose using low-rank tensor approximation, specifically the Tensor Train (TT) method, to handle the hybrid nature of contacts.

TT is a versatile function approximator that can handle both continuous and discrete variables and is particularly efficient when the function being modeled is smooth and/or when there is a correlation among the variables. The approach models the state-value and action-value functions as a sum of products of univariate functions, and TT-Cross [3], a powerful gradient-free approximation method, allows us to achieve TT approximation of a given function with a desired accuracy.

*Authors contributed equally.
[1]Robot Learning & Interaction Group, Idiap Research Institute, Martigny, Switzerland (e-mail: firstname.lastname@idiap.ch)
[2]École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

Moreover, a recent technique called TTGO [4] enables the direct finding of optima of functions represented in TT format. This allows us to efficiently obtain the best action for a given state by modeling the action-value or advantage function in TT format. We will demonstrate that our proposed algorithm can achieve the planar pushing task robustly in both simulation and the real world.

II. Problem and Proposed Method

The state of the system is denoted as $[\boldsymbol{q}_s^\top \ \boldsymbol{q}_p^\top \ c_c]^\top$, where $\boldsymbol{q}_s = [s_x \ \ s_y \ \ s_\theta]^\top$ is the position and orientation of the block, $\boldsymbol{q}_p = [p_x \ \ p_y]^\top$ is the position of end-effector, and $c_c \in \{0, 1, 2, 3\}$ is the current contact face. The action is expressed as $[\boldsymbol{v}^\top \ c_n]^\top$, where $\boldsymbol{v} = [v_n \ v_t]^\top$ is the velocity of the end-effector, and $c_n \in \{0, 1, 2, 3\}$ is the next contact face. The system, therefore, has $n = 6$ states and $m = 3$ control variables in total, including both continuous and discrete variables.

We use Value Iteration (VI) to find the optimal value function. At any iteration k, the $(k+1)$-th value function approximation is computed as:

$$\begin{aligned} V^{k+1} &= \text{TT-Cross}(\mathcal{B}^{\pi_k} V^k, \epsilon), \\ \mathcal{B}^{\pi_k} V^k(s) &= R(s, \pi^k(s)) + \gamma V^k(f(s, \pi^k(s))) \\ \pi^k(s) &= \operatorname*{argmax}_a A^k(s, a), \\ A^k(s, a) &= R(s, a) + \gamma V^k(f(s, a)) - V^k(s), \forall s, \end{aligned} \tag{1}$$

where $A^k(s, a)$ is the advantage function.

The value update step involves computing $\pi^k(s) = \arg\max_a A(s, a)$ numerous times across several iterations. To compute V^{k+1} in TT-format, the function $\mathcal{B}^{\pi_k} V^k$ is queried iteratively using TT-Cross$(\mathcal{B}^{\pi_k} V^k, \epsilon)$, with batches of states (usually ranging from 1000 to 100,000 in practice). This requires computing the policy π^k for each of these states, making a fast computation of $\arg\max_a A^k(s, a)$ in batch form crucial.

To resolve the bottleneck, the advantage function A^k is computed in TT format using TT-Cross. This is efficient as the calculation only requires evaluating V^k and R, which are cheap to compute. This enables use of TTGO [4], a numerical optimization technique for a function in TT format that can handle optimization over a mix of continuous and discrete variables. As a result, solutions for $\pi^k(s) = \operatorname{argmax}_a A^k(s, a)$ over batches of states can be obtained quickly. This results in an efficient way of policy retrieval for problems involving hybrid actions. The pseudocode is shown in Algorithm 1.

III. Experiments

We first trained the control policy in simulation based on the motion equation shown in [6], but removing the acceleration part and explicitly representing contact face as a state and input variable for ease of implementation on a real-world system. The continuous variables in state and action space are discretized into 200 and 50 points separately, while the discrete variables are kept as they are. The domain is set in the range from $[-0.5m, -0.5m, -\pi]$ to $[0.5m, 0.5m, \pi]$, with maximum velocity defined as 0.1 m/s. The accuracy of function approximation in TT-cross is defined as 10^{-3}. The reward function is defined as:

$$R(s, a) = -\|\boldsymbol{q}_s\| \times (1 + \gamma_1 \times (1 - \delta(c_c - c_n)) + \gamma_2 \times \|\boldsymbol{v}\|), \tag{2}$$

Algorithm 1 ADP using Tensor Approximation

Input:

n_v : Number of value update steps,
ϵ : Accuracy of TT representation,
$\delta_{\max}$: Tolerance specified for convergence of DP,
α: learning rate ($0 \leq \alpha \leq 1$, $\alpha = 1$ if no learning rate),
$R(s, a)$: Reward Function,
$f(s, a)$: Forward dynamics model (or a simulator),
Discretization for each state and action (implicitly assumed in TT-Cross)

Output: Policy π^*

Initialize:
Value model in TT-format: $V^0 = 0$,
Advantage model in TT-format:
$A^0 = \text{TT-Cross}(R(s, a), \epsilon)$,
(alternatively, initialize them arbitrarily in TT-format)
while $\delta \leq \delta_{\max}$ **do**
  $k = k + 1$
  $\pi^k(s) = \operatorname{argmax}_a A^{k-1}(s, a)$ (definition)
  $V_0^k = V^{k-1}$
  for $j \leftarrow 1$ to n_v **do**
    $V_j^k(s) = \text{TT-Cross}(\mathcal{B}^{\pi^k} V_{j-1}^k, \epsilon)$
  end for
  $V^k = V_{n_v}^k$
  $\hat{A}^k(s, a) = R(s, a) + \gamma V^k(f(s, a)) - V^k(s)$
  $A^k = \text{TT-Cross}(\hat{A}^k, \epsilon)$
  $A^k = \alpha A^k + (1 - \alpha) A^{k-1}$
  $\delta = \frac{\|V^k - V^{k-1}\|_2}{\|V^{k-1}\|_2}$
end while

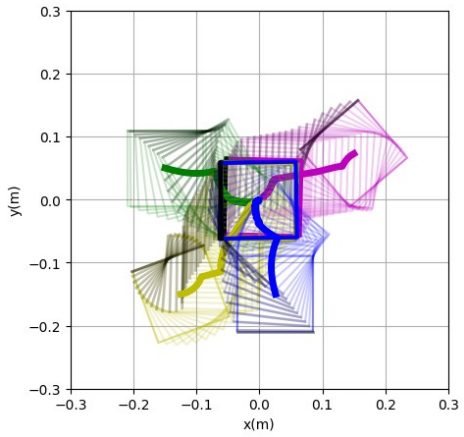
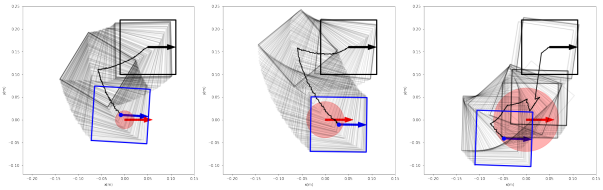


Fig. 2. Simulation of a block motion under a policy from four different initial states. The colored trajectories represent the motion of the block to the target $\boldsymbol{q}_s = [0 \ 0 \ 0]^\top$, by means of contact mode and face switching.



(a) Reaching (b) W/ add. weight (c) W/ disturbance

Fig. 3. Block trajectories of the 3 experiments conducted in the real world, namely reaching, reaching with additional weight and reaching with external disturbance. The black square and arrow represents the initialization. The blue square and arrow denote the final pose of the block. The red arrow is the target orientation, while the red circle represents the position errors. The numerical results are shown in Table I.

TABLE I
PERFORMANCE OF THREE REAL-WORLD EXPERIMENTS

Experiments	$\boldsymbol{x}_{\text{err}}$/cm	$\boldsymbol{y}_{\text{err}}$/cm	$\boldsymbol{\theta}_{\text{err}}$/rad
Reaching	-0.83	1.07	-0.06
Reaching, w. weight	2.89	-1.04	-0.01
Reaching, w. disturbance	-4.78	-4.10	-0.04

where $\boldsymbol{q}_s$ represent the block pose, $\delta(c_c - c_n)$ returns 1 if $c_c = c_n$, otherwise, 0. γ_1 and γ_2 are set as 0.1 and 0.001, respectively. Note that the flexibility offered by our method allows us to utilize such reward functions.

Each iteration of the VI procedure took about 10 seconds on average. To test the generalization capability of the policy, we randomly selected 1000 initializations in the domain. The success rate is 100%. Fig. 2 shows part of the simulation results. We reached a robust policy generating a 100% success rate in about 40 iterations while the convergence of the whole algorithm took about 500 iterations.

We then tested the trained policy on the real robot setup (Fig. 1), using a 7-axis Franka Emika robot and a RealSense D435 camera. The slider ($r_s = 6$ cm) is a 3D-printed prismatic object with PLA, lying on a flat plywood surface, with an Aruco Marker on the top face. A wooden pusher ($r_p = 0.5$cm) is attached to the robot to move the object. The motion of the object is tracked by the camera at 30 HZ, and the policy is called at 100 HZ, with a low-level Cartesian velocity controller (1000 HZ) actuating the robot.

Three experiments were conducted to assess the robustness of our policy: **a) Reaching task:** The robot pushes the slider from $\boldsymbol{q}_{s_0} = [0.05\text{m}\ \ 0.16\text{m}\ \ 0]^\top$ to origin (Fig. 1a); **b) Reaching with additional weight:** The robot pushes the block from the same initialization as before, but with an additional weight, 3 times heavier than the block (Fig. 1b); **c) Reaching with external disturbance:** The same initialization like before, but with a significant external disturbance of $\boldsymbol{q}_{\text{dist}} = [0.1\text{m}\ 0.03\text{m}\ \frac{\pi}{2}]^\top$ exerted by a human (Fig. 1c).

The results of these experiments are displayed in Fig. 3 and Table I. The results show that in all experiments, the policy successfully reaches the final target in terms of both position and orientation. The error increases with the disturbance, while orientation errors remaining less than 4° and position errors staying less than 5cm even under a significant disturbance. Experiment 3 demonstrates that the policy is able to dynamically select the contact face based on the current state, as evidenced by the change in contact face after a 90-degree rotation. This highlights the global optimality of our method and its ability to handle both continuous and discrete variables in hybrid systems.

REFERENCES

[1] Neel Doshi, Francois R Hogan, and Alberto Rodriguez. Hybrid differential dynamic programming for planar manipulation primitives. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 6759–6765, 2020.

[2] François Robert Hogan and Alberto Rodriguez. Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics. In *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*, pages 800–815. Springer, 2020.

[3] Ivan Oseledets and Eugene Tyrtyshnikov. TT-cross approximation for multidimensional arrays. *Linear Algebra and its Applications*, 432(1):70–88, 2010.

[4] S. Shetty, T. Lembono, T. Löw, and S. Calinon. Tensor trains for global optimization problems in robotics. *arXiv:2206.05077*, 2022.

[5] Maozhen Wang, Aykut Özgün Önol, Philip Long, and Taşkın Padır. Contact-implicit planning and control for non-prehensile manipulation using state-triggered constraints. In *Robotics Research*, pages 189–204. Springer, 2023.

[6] Teng Xue, Hakan Girgin, Teguh Santoso Lembono, and Sylvain Calinon. Demonstration-guided optimal control for long-term non-prehensile planar manipulation. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2023.