Logic-Geometric Planning and Control Using Graph of Tensor Networks

Teng Xue[1,2], Amirreza Razmjoo[1,2], Suhan Shetty[1,2], Sylvain Calinon[1,2]
[1]Idiap Research Institute [2]École Polytechnique Fédérale de Lausanne (EPFL)

***Abstract*—Robot manipulation usually involves multiple contact modes and primitives, leading to a hybrid structure. This requires joint logic-geometric programming for robot planning and control, posing significant challenges to existing gradient-based and sampling-based techniques. To address this issue, we propose Graph of Tensor Networks (GTN), where the problem is formulated as a shortest-path problem in this graph. The path includes a sequence of vertices corresponding to the contact modes and the continuous state in the vertex set. Our main contribution is to introduce the first-order logic into graph to alleviate the combinatorial complexity and to use Tensor Networks to approximate the value function within each vertex set, representing the distance between any two vertex states. The low-rank structure further facilitates finding near-globally optimal solutions. We validate the effectiveness of the proposed idea in a pusher-slider system and sequential manipulation tasks.**

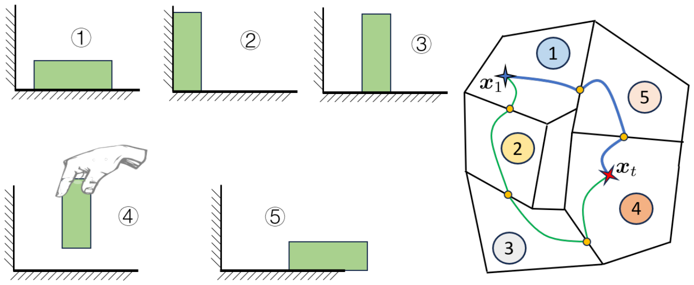


Fig. 1: An example of GTN for sequential manipulation. Left: The vertex states of a rectangular object that can only be grasped from the side. Right: The graph structure with the found optimal logic-geometric paths that connect $\boldsymbol{x}_1$ and $\boldsymbol{x}_t$. Both kinematic and dynamic constraints are considered to be satisfied.

I. Introduction

Physically interacting with the surroundings is crucial for robots to operate effectively in our daily lives. One of the key challenges to achieving this is the hybrid structure inherent in manipulation tasks: The under-actuated dynamics leads to frequent contact breaking and establishment; Coulomb friction results in different contact modes, such as sticking and sliding. Moreover, multiple different manipulation primitives, such as pushing and pivoting, are involved for long-horizon manipulation, each with distinct motion equations. This hybrid structure requires joint logic-geometric reasoning, posing significant challenges to current planning and control techniques. Gradient-based techniques excel at geometric motion planning due to the locally linear model constructed with gradients. However, the non-smooth nature of these tasks quickly disrupts this because the Taylor approximation no longer holds at the intersection of different contact modes [10]. Sampling-based techniques are available to address gradient vanishing issues; however, the combination of logic and geometric decision variables results in combinatorial complexity. The resulting high-dimensional solution space makes it tough for sampling methods to find the correct solutions. In general, manipulation tasks require joint reasoning over both logic and geometric decision variables, encountering the limitations of both gradient-based and sampling-based methods. This makes manipulation tasks particularly challenging and necessitates advanced techniques to effectively address this complexity.

To address this issue, mathematical program using complementary constraints (MPCC) is proposed in [11, 6]. By uncovering the internal structures of different contact modes, this method eliminates the need for integer variables in problem formulation, allowing the use of gradient-based techniques. However, adding complementary constraints introduces more discontinuities and non-differentiabilities in the feasible region of the problem, increasing the likelihood of getting trapped in local optima. Recently, a new technique called Graph of Convex Sets (GCS) [5] has emerged. It decomposes a complex non-convex problem into multiple simpler convex problems. By combining the global perspective of graph search with the local efficiency of convex optimization, this method can effectively provide the global optimal solution. It performs well once the non-convex problems can be convexified [4]. However, convexifying manipulation tasks involving physical contact is typically challenging due to the complicated contact dynamics.

Inspired by GCS, we propose Graph of Tensor Networks (GTN) in this study. GTN utilizes graph structures to express the combinatorial complexity arising from discrete variables. However, instead of employing convex optimization for geometric aspects, we propose leveraging tensor networks to approximate the value function of each vertex set. This method eliminates the need for convexifying the dynamics model and cost/reward function while still potentially providing a near-global solution. We applied this method to the pusher-slider system and several sequential manipulation tasks, showcasing reactive control under real-world uncertainty and fast planning for long-horizon manipulation tasks.

II. Background

A. Tensors as Discrete Analogue of a Function

A multivariate function $P(x_1, \dots, x_d)$ defined over a rectangular domain made up of the Cartesian product of intervals (or discrete sets) $I_1 \times \cdots \times I_d$ can be discretized by evaluating it at points in the set $\mathcal{X} = \{(x_1^{i_1}, \ldots, x_d^{i_d}) : x_k^{i_k} \in I_k, i_k \in \{1, \ldots, n_k\}\}$. This gives us a tensor $\boldsymbol{\mathcal{P}}$, a discrete version of P, where $\boldsymbol{\mathcal{P}}_{(i_1,\ldots,i_d)} = P(x_1^{i_1}, \ldots, x_d^{i_d}), \forall (i_1, \ldots, i_d) \in \mathcal{I}_{\mathcal{X}}$, and $\mathcal{I}_{\mathcal{X}} = \{(i_1, \ldots, i_d) : i_k \in \{1, \ldots, n_k\}, k \in \{1, \ldots, d\}\}$. The value of P at any point in the domain can then be approximated by interpolating between the elements of the tensor $\boldsymbol{\mathcal{P}}$.

B. Tensor Networks and Tensor Train Decomposition

However, naively approximating a high-dimensional function using a tensor is intractable due to the combinatorial and storage complexities of the tensor $(\mathcal{O}(n^d))$. Tensor networks mitigate the storage issue by decomposing the tensor into factors with fewer elements, akin to using Singular Value Decomposition (SVD) to represent a large matrix. In this paper, we explore the use of Tensor Train, a type of Tensor Network that represents a high-dimensional tensor using several third-order tensors called *cores*.

We can access the element $(i_1, \ldots, i_d)$ of the tensor in this format simply given by multiplying matrix slices from the cores:

$$\boldsymbol{\mathcal{P}}_{(i_1,\ldots,i_d)} = \boldsymbol{\mathcal{P}}^1_{:,i_1,:} \boldsymbol{\mathcal{P}}^2_{:,i_2,:} \cdots \boldsymbol{\mathcal{P}}^d_{:,i_d,:}, \tag{1}$$

where $\boldsymbol{\mathcal{P}}^k_{:,i_k,:} \in \mathbb{R}^{r_{k-1} \times r_k}$ represents the i_k-th frontal slice (a matrix) of the third-order tensor $\boldsymbol{\mathcal{P}}^k$. For any given tensor, there always exists a TT decomposition [8]. This low-rank structure further facilitates sampling and optimization for robot planning and control.

There are several ways to acquire a TT model, including TT-SVD [9] and TT-Cross [7, 13]. TT-SVD extends the SVD decomposition from matrix level to a high-dimensional tensor level. However, it needs to store the full tensor first, which is impractical to very high-dimensional functions. TT-Cross solves this issue by selectively evaluating function P on a subset of elements, avoiding the need to store the entire tensor.

C. Function approximation using Tensor Train

Given the discrete analogue tensor $\boldsymbol{\mathcal{P}}$ of a function P, we obtain the continuous approximation by spline-based interpolation of the TT cores corresponding to the continuous variables only. For example, we can use linear interpolation for the cores (i.e., between the matrix slices of the core) and define a matrix-valued function corresponding to each core $k \in \{1, \ldots, d\}$,

$$\boldsymbol{P}^k(x_k) = \frac{x_k - x_k^{i_k}}{x_k^{i_k+1} - x_k^{i_k}} \boldsymbol{\mathcal{P}}^k_{:,i_k+1,:} + \frac{x_k^{i_k+1} - x_k}{x_k^{i_k+1} - x_k^{i_k}} \boldsymbol{\mathcal{P}}^k_{:,i_k,:}, \tag{2}$$

where $x_k^{i_k} \leq x_k \leq x_k^{i_k+1}$ and $\boldsymbol{P}^k : I_k \subset \mathbb{R} \to \mathbb{R}^{r_{k-1} \times r_k}$ with $r_0 = r_d = 1$. This induces a continuous approximation of P given by

$$P(x_1, \ldots, x_d) \approx \boldsymbol{P}^1(x_1) \cdots \boldsymbol{P}^d(x_d). \tag{3}$$

This allows us to selectively do the interpolation only for the cores corresponding to continuous variables, and hence we can represent functions in TT format whose variables could be a mix of continuous and discrete elements.

D. Global Optimization using Tensor Train

An arbitrary function can be transformed into a nonnegative function in TT format, which can be interpreted as a probability density function. The efficient sampling techniques for density functions in TT format allow to pick samples of only high-density regions which in turn correspond to the optima. In practice, the chosen number of prioritized samples $N \geq 1$ and the sample(s) with the highest density (or least cost) is used to represent the optima. This leads to the near-global solutions, which can be further refined using local optimization techniques such as Newton-type optimization for continuous variables. Such process is gradient-free, and it can handle a mix of continuous and discrete variables. For more details, readers are referred to [14].

III. Method

A. Graph of Tensor Networks

We define a *Graph of Tensor Networks (GTN)* as a directed graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the vertex set and $\mathcal{E}$ is the edge set. For each vertex $v \in \mathcal{V}$, it is associated with a set $\mathcal{X}_v$ and a point $\boldsymbol{x}_v$ is contained in it. The edge length $e(u, v) \in \mathcal{E}$ is defined by an edge cost function $l_e(\boldsymbol{x}_u, \boldsymbol{x}_v)$. Moreover, the edges of the graph satisfy the transition constraints $(\boldsymbol{x}_u, \boldsymbol{x}_v) \in \mathcal{X}_e$, which define the vertex switching conditions. Note that in this formulation, we do not assume the convex structures of the vertex set, cost functions, and constraints, which are required in GCS.

Given the initial vertex state $\boldsymbol{x}_1$ and target state $\boldsymbol{x}_t$, the full logic-geometric path p in the graph G is a sequence of distinct vertices $(v_1, \cdots, v_K)$ and the continuous states $\boldsymbol{x}_{1:K}$. We define the family of all paths that connect $\boldsymbol{x}_1$ and $\boldsymbol{x}_t$ as $\mathcal{P}$ and the set of edges traversed by the path p as $\mathcal{E}_p$. In this work, In this work, we eliminate the need for an explicit target goal by using an evaluation function l_T over the final vertex state, with explicit target state definition being a special case. Therefore, the Logic-Geometric Programming problem [16] in this graph G can be defined as:

$$\begin{aligned} \text{minimize} \quad & \sum_{(u,v,t) \in p} l_e(\boldsymbol{x}_u, \boldsymbol{x}_v) + l_T(\boldsymbol{x}_K), \\ \text{s.t.} \quad & p \in \mathcal{P}, \\ & \boldsymbol{x}_v \in \mathcal{X}_v, \boldsymbol{x}_u \in \mathcal{X}_u, \boldsymbol{x}_K \in \mathcal{X}_K, \\ & (\boldsymbol{x}_u, \boldsymbol{x}_v) \in \mathcal{X}_e. \end{aligned} \tag{4}$$

The above problem is typically a Mixed-Integer Program (MIP), which is NP-hard and difficult to solve. To address this issue, we introduce more structures to this problem: 1) Edge constraints are described by first-order logic action operators. 2) The cost functions and path constraints within each vertex (e.g., dynamics of a specific contact mode) are locally smooth. These assumptions are commonly used in the Task and Motion Planning literature [17]. Given that smooth functions can be well-approximated by a few basis functions, the discretized tensor can be effectively approximated by a lower-dimensional subspace, leading to a low-rank structure. Therefore, we propose employing Tensor Networks to approximate the cost functions and edge constraints in (4).

B. Approximating Value Functions using Tensor Train

The cost function $l_e(\boldsymbol{x}_v, \boldsymbol{x}_u)$ can be easily defined as the Euclidean distance for kinematic obstacle avoidance tasks in Cartesian space. However, this metric is ineffective for manipulation tasks with dynamics constraints. From a control perspective, the value function is a general metric that measures the distance between two states, considering either kinematic or dynamics constraints. However, approximating such value functions is challenging due to the curse of dimensionality of the state space. In this work, we propose to address this issue by using Tensor Train to explore the low-rank structure. The value function is initialized as zero and is refined iteratively through value iteration. At iteration i, the $(i+1)$-th value function approximation using TT-Cross is computed as follows:

$$\begin{aligned} A^i(\boldsymbol{x},\boldsymbol{u}) =& R(\boldsymbol{x},\boldsymbol{u}) + \gamma(V^i(f(\boldsymbol{x},\boldsymbol{u})) - V^i(\boldsymbol{x})), \forall \boldsymbol{x}, \\ \pi(\boldsymbol{x}) =& \arg\max_{\boldsymbol{u}} A^i(\boldsymbol{x},\boldsymbol{u}), \\ \mathcal{B}^\pi V^i(\boldsymbol{x}) =& R(\boldsymbol{x},\pi(\boldsymbol{x})) + \gamma V^i(f(\boldsymbol{x},\pi(\boldsymbol{x}))) \\ V^{i+1} =& \text{TT-Cross}(\mathcal{B}^\pi V^i, \epsilon), \end{aligned} \tag{5}$$

where $A^i(\boldsymbol{x},\boldsymbol{u})$ is the advantage function, $f(\boldsymbol{x},\boldsymbol{u})$ is the dynamics model and $R(\boldsymbol{x},\boldsymbol{u})$ is the reward function, ϵ is the accuracy threshold of TT-cross approximation. To compute V^{i+1} in TT format, the function $\mathcal{B}^\pi V^i$ is queried iteratively using TT-Cross($\mathcal{B}^\pi V^i, \epsilon$), until convergence. This algorithm is called Tensor Train for Generalized Policy Iteration (TTPI). The reader is referred to [15] for more details.

C. Logic-Geometric Programming using GTN

We can therefore associate each vertex set in the graph G with a value function that indicates the distance between any two states in the vertex set. To solve (4), we propose an approach that alternates between symbolic vertex search and value optimization.

1) Level 1: Symbolic Search: We use Planning Domain Definition Language (PDDL) [1] to describe the edge constraints $\mathcal{X}_e$ and then utilize Monte Carlo Tree Search (MCTS) to search for the appropriate vertex sequence $v_{1:K}$. The preconditions and effects of each vertex are denoted as s_{k-1} and s_k, respectively, forming a rule-based representation of symbolic transitions $s_k = succ(s_{k-1}, v_k)$. In each iteration, MCTS relies on the Upper Confidence Bound (UCB1) [18] to balance exploration and exploitation strategies.

2) Level 2: Value Optimization: Conditioning on the vertex skeleton, we can compute the vertex states $\boldsymbol{x}_{1:K}$ by optimizing (4). The resulting path should satisfy both the vertex constraints and the edge constraints. The edge constraints dictate that $\boldsymbol{x}_k$ must lie within the intersection space of two adjacent vertex sets, $\mathcal{X}_k$ and $\mathcal{X}_{k+1}$, ensuring configuration consistency. The vertex constraints are addressed by the local controller, e,.g., primitive policy, within each vertex set.

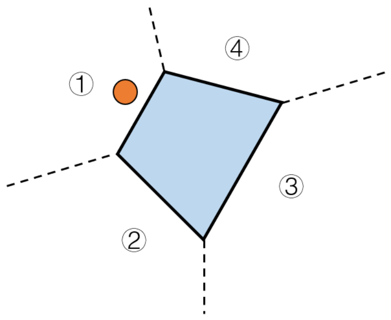
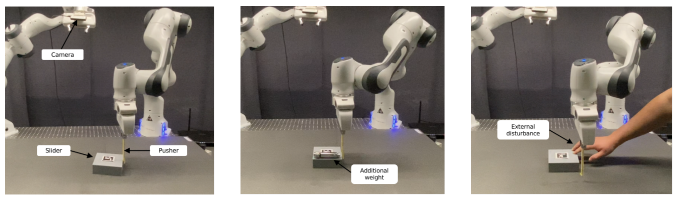


Fig. 2: An example of GTN for a planar pushing task. The pusher and slider are represented in orange and blue colors, respectively. Each face can be seen as a vertex in the graph.



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

Fig. 3: Pusher-slider system where the robot pushes an object by contact switching.

Since the objective function can be in any form, either convex or non-convex, depending on the value functions of selected skills, we employ the Cross-Entropy Method (CEM) [12, 3] as the optimization technique. The distributions are iteratively updated towards the fraction of the population with higher objective scores until converging to the best solution. For more details, readers are referred to [19].

IV. Experiments

We validate the proposed idea on two applications: face-switching pusher-slider system and long-horizon sequential manipulation. The accompanying video can be found at: https://youtu.be/RQ_OQUmzSdk.

A. Face-switching pusher-slider system

The pusher-slider system is widely used for testing algorithms in robot planning and control due to the challenges posed by its under-actuated dynamics and multiple contact modes. In this study, we introduce a face-switching mechanism to broaden the feasible target space. However, it further complicates the problem by introducing additional logic states.

The planar slider geometry is assumed to be a polygon with N_F faces. Each face can be considered as a vertex in the graph, associated with a value function, as shown in Fig. 2. For each contact face, the slider satisfies the same motion equations. Therefore, we can treat the discrete vertex selection together with the continuous variables and solve (4) directly using dynamic programming, leading to an optimal control policy with a mix of discrete and continuous outputs. We then tested the policy on the real robot setup (Fig. 3), using a 7-axis Franka Emika robot and a RealSense D435 camera. Three

Table I: Comparison of computation time and solution quality for sequential skill planning

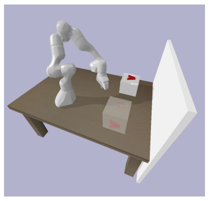
	Computation Time (s) [w/ sym. goal]		Computation Time (s) [w/o sym. goal]		Cumulative Reward		Sequence Length
	GTN	STAP	GTN	STAP	GTN	STAP	
NPM	0.14 ± 0.23	0.06 ± 0.03	0.27 ± 0.15	NA	3.0 ± 0	2.4 ± 1.57	3.0 ± 0
PPM	0.17 ± 0.1	0.08 ± 0.02	0.22 ± 0.13	NA	2.9 ± 0.7	1.88 ± 1.01	2.9 ± 0.7
PM	0.25 ± 0.15	0.21 ± 0.02	0.41 ± 0.02	NA	5.0 ± 0	3.28 ± 0.62	5.0 ± 0

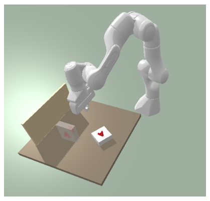
Table II: Performance of real-world experiments for planar pushing

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

experiments were conducted to assess the robustness of the obtained policy, as depicted in Fig. 3. The results of these experiments are summarized in Table II. Across all experiments, the policy successfully reached the final target with precise positioning and orientation, even amidst significant disruption. Moreover, Experiment 3 showcased 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° rotation. This highlights the ability of our method to handle both logic and geometric variables in hybrid systems.

B. Long-horizon sequential manipulation


(a) Non-Prehensile Manipulation


(b) Partly-Prehensile Manipulation

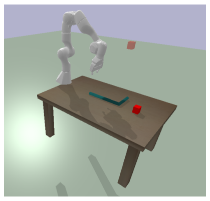

(c) Prehensile Manipulation

Fig. 4: Three sequential manipulation tasks, including both prehensile and non-prehensile manipulation primitives. The transparent object represents the final target configuration in each domain.

We further expand the approximated value function library by incorporating more primitives, including `push`, `pivot`, `pull`, `pick` and `place`. The approximation accuracy of the value functions using different approaches is shown in Table III. The metric is whether the approximated value function can yield the same result as the cumulative reward given any two states. The results indicate that value functions obtained through TTPI offer better accuracy under less computation time, thanks to the exploration of the low-rank structure.

Three long-horizon manipulation tasks requiring a sequence of primitives are used to validate the proposed method, as shown in Fig. 4. Given the same initial state and target configuration, GTN actively discovers multiple solutions, including both the vertex skeleton and the optimal vertex states within it. The results are shown in the accompanying video.

We then compare our method with another state-of-the-art sampling-based sequential manipulation planning method

Table III: Comparison of value function accuracy and computation time (minute).

	TTPI		SAC		PPO	
	accuracy	time	accuracy	time	accuracy	time
pushing	0.85	5.6	0.63	36.3	0.61	44.8
pivoting	0.94	0.9	0.51	16.0	0.68	8.7
pulling	0.97	1.1	0.84	16.5	0.72	10.7
pick/place	0.93	1.67	0.64	33.6	0.66	24.6

called STAP [2]. STAP focuses on constraints satisfaction. It requires an explicit symbolic vertex goal for discrete vertex planning, followed by feasible solutions sampling. To ensure a fair comparison, we use MCTS with an explicit symbolic goal as the task planner in STAP and then employ CEM for feasibility checking given the vertex skeleton. Table I illustrates the time required to find one solution and the solution quality between GTN and STAP. We can observe that GTN does not require the symbolic vertex goal, whereas STAP relies on it. STAP can find the solution faster than GTN. The reason is that STAP focuses only on finding the feasible solution, while GTN aims to provide (global) optimality over the full logic-geometric path. This aligns with the comparison of cumulative rewards. We can observe that the trajectory found by GTN leads to a higher cumulative reward compared with STAP, indicating better optimality.

V. Conclusion

We propose Graph of Tensor Networks (GTN) to address the combinatorial complexity and gradient vanishing in logic-geometric programming. First-order logic is introduced into the graph to define the edge constraints, while Tensor Train is utilized to explore the low-rank structure of the value function associated with each vertex, showcasing better accuracy. We demonstrate the effectiveness of the proposed idea on a pusher-slider system and sequential manipulation tasks with contact uncertainty and external disturbances.

In this work, Tensor Train is employed for value function approximation, followed by the application of the Cross-Entropy Method for full optimization. However, we believe that such low-rank structure expressed by Tensor Train can also be leveraged for full path optimization. This will be investigated in the future.

In the sequential manipulation task, value functions for different manipulation primitives are separately approximated with distinct scales. Further consideration can be given to use a common metric for all value functions. This would enable the use of more efficient graph search techniques, expanding beyond solely Monte Carlo Tree Search.

Acknowledgments

References

[1] Constructions Aeronautiques, Adele Howe, Craig Knoblock, ISI Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins SRI, Anthony Barrett, Dave Christianson, et al. Pddl— the planning domain definition language. *Technical Report, Tech. Rep.*, 1998.

[2] Christopher Agia, Toki Migimatsu, Jiajun Wu, and Jeannette Bohg. Stap: Sequencing task-agnostic policies. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7951–7958. IEEE, 2023.

[3] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134:19–67, 2005.

[4] Graesdal, Bernhard P and Chia, Shao YC and Marcucci, Tobia and Morozov, Savva and Amice, Alexandre and Parrilo, Pablo A and Tedrake, Russ. Towards tight convex relaxations for contact-rich manipulation. 2024.

[5] Tobia Marcucci, Mark Petersen, David von Wrangel, and Russ Tedrake. Motion planning around obstacles with convex optimization. *Science robotics*, 8(84):eadf7843, 2023.

[6] João Moura, Theodoros Stouraitis, and Sethu Vijayakumar. Non-prehensile planar manipulation via trajectory optimization with complementarity constraints. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 970–976. IEEE, 2022.

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

[8] Ivan V. Oseledets. Tensor-Train Decomposition. *SIAM Journal on Scientific Computing*, 33:2295–2317, 2011.

[9] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[10] Tao Pang, HJ Terry Suh, Lujie Yang, and Russ Tedrake. Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models. *IEEE Transactions on Robotics*, 2023.

[11] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *International Journal of Robotics Research (IJRR)*, 33(1):69–81, 2014.

[12] Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1:127–190, 1999.

[13] Dmitry V. Savostyanov and Ivan V. Oseledets. Fast adaptive interpolation of multi-dimensional arrays in tensor train format. *The 2011 International Workshop on Multidimensional (nD) Systems*, pages 1–8, 2011.

[14] S. Shetty, T. Lembono, T. Löw, and S. Calinon. Tensor Train for Global Optimization Problems in Robotics. *International Journal of Robotics Research (IJRR)*, 43 (6):811–839, 2024. doi: 10.1177/02783649231217527.

[15] S. Shetty, T. Xue, and S. Calinon. Generalized Policy Iteration using Tensor Approximation for Hybrid Control. In *Proc. Intl Conf. on Learning Representations (ICLR)*, 2024.

[16] Marc Toussaint. Logic-geometric programming: an optimization-based approach to combined task and motion planning. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 1930–1936, 2015.

[17] Marc Toussaint, Kelsey R Allen, Kevin A Smith, and Josh B Tenenbaum. Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning. In *Proc. of Robotics: Science and Systems (R:SS)*, 2018. *Best Paper Award.*

[18] Brian C Williams. Cognitive Robotics: Monte Carlo Tree Search. Cambridge MA, 2016. MIT OpenCourseWare.

[19] T. Xue, A. Razmjoo, S. Shetty, and S. Calinon. Logic-Skill Programming: An Optimization-based Approach to Sequential Skill Planning. In *Proc. Robotics: Science and Systems (R:SS)*, 2024.