Journal Title
XX(X):1–21
©The Author(s) 2025
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Monte Carlo Tree Search with Tensor Factorization for Robot Optimization

Teng Xue[1, 2], Yan Zhang[1, 2*], Amirreza Razmjoo[1, 2*], and Sylvain Calinon [1, 2]

Abstract

Many robotic tasks, such as inverse kinematics, motion planning, and optimal control, can be formulated as optimization problems. Solving these problems involves addressing nonlinear kinematics, complex contact dynamics, long-horizon correlation, and multi-modal landscapes, each posing distinct challenges for state-of-the-art optimization methods. Monte Carlo Tree Search is a powerful approach that can strategically explore the solution space and can be applied to a wide range of tasks across varying scenarios. However, it typically suffers from combinatorial complexity when applied to robotics, resulting in slow convergence and high memory demands. To address this limitation, we propose *Tensor Train Tree Search* (TTTS), which leverages tensor factorization to exploit correlations among decision variables arising from common kinematic structures, dynamic constraints, and environmental interactions in robot decision-making. This yields a compact, linear-complexity representation that significantly reduces both computation time and storage requirements. We prove that TTTS can efficiently reach the bounded global optimum within a finite time. Experimental results across inverse kinematics, motion planning around obstacles, legged robot manipulation, multi-stage motion planning, and bimanual whole-body manipulation demonstrate the efficiency of TTTS on a diverse set of robotic tasks. Project website: https://sites.google.com/view/tt-ts.

Keywords

Monte Carlo Tree Search, Multi-modal Robot Optimization, Contact-rich Manipulation, Tensor Factorization

1 Introduction

Optimization plays a critical role in robotics and serves as the foundation for a wide range of tasks, including inverse kinematics (Goldenberg et al. 2003), obstacle avoidance (Marcucci et al. 2023), multi-stage motion planning (Toussaint 2015), and contact-rich manipulation (Mason 1986, 1999), see Figure 1. Solving these problems requires addressing inherent nonlinear system dynamics, non-convex constraints, joint reasoning over discrete contact modes and continuous motion trajectories, as well as complex interactions with the environment—each of which presents substantial challenges for state-of-the-art optimization methods. Specifically, the nonlinearity intrinsic to robotics renders many of these problems non-convex, making gradient-based methods prone to getting trapped in poor local optima (Lembono et al. 2020). Similarly, the need to jointly optimize over discrete modes and continuous motion introduces significant combinatorial complexity, substantially increasing computational costs and slowing convergence for both sampling-based (LaValle 2006; Kavraki et al. 1996) and search-based (Hart et al. 1968) approaches. Another key challenge is multi-modal solution discovery, where multiple distinct feasible solutions may exist due to task redundancies or environmental symmetries. Identifying and reasoning over such diverse solutions is essential for both robustness and global optimization. To cope with these challenges, general formulations are often discarded to instead focus on a small set of problems by exploiting the specific structures. For example, trajectory optimization is often addressed through convex optimization with local linearization (Wang and Grant 2017; Malyuta et al. 2022). Obstacle avoidance is typically tackled using sampling-based methods, assuming that the environment contains a sufficiently large free space (Lin and Saripalli 2017). Contact-rich manipulation—a significantly more challenging task involving hybrid dynamics—is often tackled through local smoothing (Pang et al. 2023) or the use of complementarity constraints (Moura et al. 2022). Different optimization techniques can also be combined hierarchically to solve more complex tasks such as aggressive quadrotor flight (Natarajan et al. 2021), global optimization around obstacles (Marcucci et al. 2023), and fast continuous-time motion planning (Mukadam et al. 2018).

Although these approaches are effective for individual tasks, they lack generalizability across domains. This limitation necessitates substantial human effort in problem design and imposes a heavy burden on problem decomposition in order to coordinate multiple specialized solvers for multi-task autonomy. Moreover, certain problems, such as multi-stage motion planning (Toussaint 2015; Xue et al. 2024a)

[1]Idiap Research Institute, Martigny, Switzerland
[2]École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
[*] Contributed equally to this work.

Corresponding author:
Teng Xue, Idiap Research Institute, Rue Marconi 19, 1920 Martigny, Switzerland.
Email: teng.xue@idiap.ch

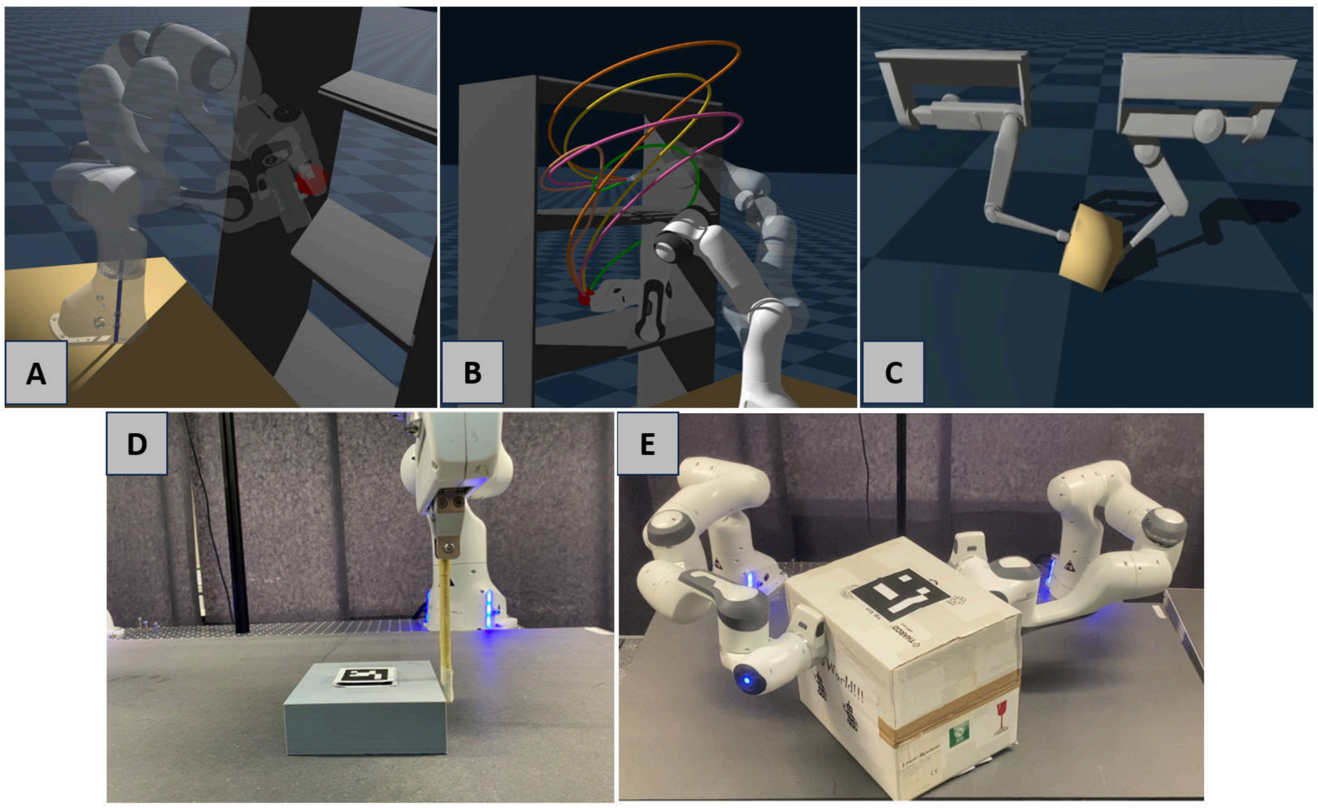


Figure 1. Overview of diverse applicable domains. We demonstrate that TTTS is widely applicable in many tasks, such as Inverse Kinematics (A), Motion Planning (B), Legged Robot Manipulation (C), Multi-stage Motion Planning (D) and Bimanual Whole-body Manipulation (E).

(also referred to as task and motion planning (Garrett et al. 2021)), are inherently coupled and cannot be decomposed into independent subproblems that can be solved in isolation. In such cases, discrete mode sequencing (typically addressed with search-based methods) and continuous motion planning (typically addressed with gradient-based or sampling-based methods) must be considered jointly.

These challenges highlight the need for a general formulation for robot decision-making. Monte Carlo Tree Search (MCTS) (Coulom 2006) offers a compelling framework: it eliminates reliance on gradient information, thereby avoiding poor local optima and making it well-suited for handling nonconvex constraints, nonlinear kinematics and dynamics. Moreover, by strategically exploring multiple branches of the search tree, MCTS is naturally capable of multi-modal solution discovery, identifying diverse feasible strategies across different modes. Furthermore, the continuous domain can be discretized into tree nodes, enabling joint modeling of both discrete and continuous decision variables. This discretization, however, often results in excessively large trees and correspondingly slow convergence. To alleviate this issue, the spectrum of locally linearized controllability Gramians has been used in (Rivière et al. 2024) to efficiently decompose a continuous dynamical system into a few discrete sets. While effective in purely continuous state–action spaces, this approach remains limited in addressing hybrid discrete–continuous decision-making and the combinatorial complexity inherent in multi-stage robotic problems. Moreover, in high-dimensional nonlinear systems such as robotic manipulators, local linearization can still lead to a combinatorial explosion. Neural network heuristics have also been explored for guiding MCTS (Silver et al. 2016), but they typically require extensive training data and often generalize poorly beyond the training domain.

Considering that in robotics the decision trees resulting from domain discretization are typically not arbitrary, different branches often exhibit substantial redundancy because they share correlations induced by common robot kinematics, dynamics, and environmental constraints, as discussed in (Roy et al. 2021). Therefore, in this work, we propose to address the curse of dimensionality using tensor factorization, which provides a principled way to capture and exploit this redundancy, analogous to its use in quantum physics for capturing localized correlations (Eisert et al. 2010). Specifically, we introduce *Tensor Train Tree Search* (TTTS), which encodes the tree in Tensor Train (TT) format (Oseledets 2011)—a decomposition that represents a high-dimensional tensor as a sequence of third-order *cores* with significantly fewer elements, each associated with a tree layer. This compact representation enables efficient search and optimization in otherwise exponentially large decision spaces, while preserving the branching logic required for tree-based decision making. Unlike conventional node- and table-based representations, the TT representation allows full parallelization of MCTS, reducing search time with linear scaling. Moreover, its separable structure effectively transforms the combinatorial growth of the original tree into linear complexity, leading to exponential reductions in both search time and memory. By combining tensor factorization with MCTS-based search, our algorithm can achieve efficient convergence to a bounded global optimum (within a parameterized trajectory class) in finite time.

Our main contributions are as follows:

1. We propose a general formulation for decision making in robotics that can handle nonlinear dynamical systems, non-convex constraints, hybrid state-action spaces, and multi-modal solution discovery.
2. We introduce the use of Tensor Train (TT) to exploit the redundancy of decision trees in robotics, where its separable structure enables fully parallel tree search and achieves linear time and space complexity for efficient decision-making.
3. We provide theoretical guarantees, proving bounded global convergence for robot decision making.
4. We evaluate our approach against state-of-the-art methods and demonstrate its effectiveness on inverse kinematics, motion planning around obstacles, multi-stage motion planning, legged manipulation, and real-world bimanual whole-body manipulation tasks.

2 Related Work

2.1 Gradient-based Methods

Gradient-based and Newton-based methods (i.e., first and second-order optimization) are the most popular approaches in robot optimization, characterized by rapid convergence when gradients of the objective and constraints are available. Standard methods include Differential Dynamic Programming (DDP) (Mayne 1966; Tassa et al. 2012), iterative Linear Quadratic Regulators (iLQR) (Li and Todorov 2004), and trajectory optimization frameworks such as CHOMP (Ratliff et al. 2009) and TrajOpt (Schulman et al. 2014). The main advantage of gradient-based and Newton-based optimization is its efficiency in smooth and convex problem formulations, enabling practical deployment in real-time robot control scenarios. In these applications, gradients and Hessians provide useful local guidance, significantly speeding up convergence.

However, the critical limitation of gradient-based and Newton-based methods arises in scenarios involving highly nonlinear, non-convex cost landscapes or discontinuous constraints, such as hybrid modes, contacts, and collision avoidance conditions (Xue et al. 2023; Posa et al. 2014). These scenarios are often too sensitive to initialization and often get trapped in poor local minima, demanding accurate initial guesses or sophisticated initialization schemes.

In contrast to these approaches, TTTS does not rely on convex structures and initial guesses. It can address highly nonlinear, non-convex and multi-modal landscapes, as well as mixed-integer decision variables. For example, the planar pushing task with face switching mechanism in (Xue et al. 2023) requires optimization on both discrete and continuous decision variables, which is not solvable with typical gradient-based and Newton-based methods. TTTS tackles this problem through TT approximation and strategic tree search, where gradient-based approaches can still be integrated within the framework as a refinement step for continuous variables, thereby combining the global exploration capability of tree search with the local optimization efficiency of gradient methods.

2.2 Sampling-based Methods

Sampling-based methods, including Rapidly-exploring Random Trees (RRT) (LaValle 2006) and Probabilistic Roadmaps (PRM) (Kavraki et al. 1996), complement gradient-based approaches by leveraging random exploration of the search space. Their primary advantage lies in their ability to handle complex constraints, discontinuities, and high-dimensional state spaces without explicit gradient information. Variants like RRT* (Karaman and Frazzoli 2011), BIT* (Gammell et al. 2015) and CMA-ES (Hansen et al. 2003) further improve path optimality and convergence rates, demonstrating remarkable flexibility and scalability in challenging environments.

Nevertheless, sampling-based methods typically suffer from slow convergence when dealing with narrow passages or high-dimensional solution space, as their success heavily depends on the probabilistic coverage of critical regions in the search space (Karaman and Frazzoli 2011; Salzman and Halperin 2016). Consequently, they often require extensive computational resources or advanced heuristic-guided sampling strategies to achieve satisfactory performance. Moreover, sampling-based methods can only ensure probabilistic completeness, lacking a strategical sampling behavior for global convergence, thus making solution finding not guaranteed in finite time.

In contrast, TTTS leverages TT approximation to quickly explore the full tree, playing as informative priors. It also possesses strategical search for solution finding, leading to global convergence in finite time. In our framework, CMA-ES is integrated in TTTS for refinement of the TT-Tree search solutions to bridge the gap between continuous decision domain and discretized nodes.

2.3 Search-based Methods

Search-based planners, such as A* (Hart et al. 1968) and Monte Carlo Tree Search (MCTS) (Coulom 2006), formulate the optimization problem explicitly as a discrete search through structured state or state-action spaces (Hart et al. 1968; Browne et al. 2012). A major advantage of these methods is their strong theoretical guarantees of completeness and optimality (given suitable heuristics), particularly valuable in environments with discrete or easily discretizable action sets. Furthermore, methods like MCTS effectively balance exploration and exploitation, significantly improving planning efficiency in complex domains (Silver et al. 2017). Recent studies have also applied MCTS to contact-rich hybrid planning problems in robotic manipulation (Zhu et al. 2023; Cheng et al. 2023), demonstrating its potential to handle challenging contact dynamics and improve dexterity through hierarchical exploration.

While these works highlight the promise of MCTS in such domains, they also expose its limited scalability, largely due to the need to discretize continuous spaces. This discretization leads to exponential growth in computational complexity as dimensionality increases (i.e., the curse of dimensionality). Moreover, the discretization itself introduces approximation errors and scaling issues, rendering these approaches less effective or computationally prohibitive in high-dimensional continuous domains without carefully designed schemes or heuristics (Choset et al. 2005). Rivière *et al.* addresses this issue by decomposing continuous dynamical systems through the spectrum of the locally linearized controllability Gramian (Rivière et al. 2024), but the discretization is achieved through local linearization, which can still leads to node explosion for highly nonlinear dynamical system, such as robot manipulator.

TTTS leverages tensor train (TT) to reduce combinatorial complexity to linear complexity in both storage and computation, where TT also provides effective guidance and enables parallel rollouts for faster convergence. The strategic search from MCTS is preserved to guarantee global convergence.

2.4 *Hybrid Methods*

Hybrid methods have also been developed that integrate multiple planning paradigms to overcome the limitations of purely gradient-based, sampling-based, or search-based methods. For example, combining trajectory optimization with sampling-based planners (such as RRT followed by trajectory optimization (Deits and Tedrake 2015)) enables more effective exploration in high-dimensional spaces. In the context of mixed-integer programming, approaches like Logic-Geometric Programming (LGP) (Toussaint 2015) alternate between discrete symbolic search and continuous geometric optimization. More recent hybrid methods combine global search strategies (e.g., MCTS or heuristic-based planners) with local refinement techniques or learning-based value estimation (Marcucci et al. 2024; Xue et al. 2024c). These approaches have proven especially effective for solving complex planning problems that involve both discrete and continuous variables (Anthony et al. 2017; Kim et al. 2020). By leveraging the strengths of both local optimization and global search, these methods achieve practical efficiency across a wide range of applications, including obstacle avoidance, contact-rich manipulation and legged locomotion.

Despite these benefits, hybrid methods often require careful engineering and tuning of parameters to balance computational resources effectively. Additionally, the integration of different planning paradigms introduces additional algorithmic complexity and implementation overhead, complicating both theoretical analysis and practical deployment.

TTTS can be seen as a hybrid approach that combines global search with local refinement, but different from the vanilla combination, TT provides a novel representation of decision tree with separable structure, enabling more efficient and parallelizable tree search. Moreover, different from the hierarchical framework that alternates between high-level discrete search and low-level continuous optimization, TTTS addresses the mixed-integer optimization jointly, by considering a joint distribution.

3 Background

3.1 *Tensor Train Function Approximation*

A multivariate function $F(x_1, \ldots, x_d)$ defined on a Cartesian product domain $I_1 \times \cdots \times I_d$ can be discretized into a tensor $\boldsymbol{\mathcal{F}}$ by sampling it on a grid, where each entry is given by

$$\boldsymbol{\mathcal{F}}_{i_1,\ldots,i_d} = F(x_1^{i_1}, \ldots, x_d^{i_d}), \quad i_k \in \{1, \ldots, n_k\}.$$

The continuous function F can then be approximated by interpolating the entries of $\boldsymbol{\mathcal{F}}$. However, direct storage and computation of high-dimensional tensors is infeasible due to their $\mathcal{O}(n^d)$ complexity. Analogous to matrix factorizations, tensor networks provide compact representations through factorization. In particular, the *Tensor Train* (TT) decomposition expresses a d-dimensional tensor as a sequence of low-rank three-dimensional tensors, called *cores*. In TT format, a tensor entry is represented as

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

where $\boldsymbol{\mathcal{F}}^k_{:,i_k,:} \in \mathbb{R}^{r_{k-1} \times r_k}$ denotes the i_k-th slice of the k-th core. TT decompositions are guaranteed to exist and can significantly reduce computational complexity (Oseledets 2011).

Algorithms such as TT-SVD (Oseledets 2011) and TT-Cross (Oseledets and Tyrtyshnikov 2010; Savostyanov and Oseledets 2011) provide efficient frameworks for computing and storing TT decompositions. TT-SVD relies on successive matricizations of the tensor and truncated singular value decompositions (SVDs), yielding quasi-optimal low-rank approximations but requiring access to the full tensor, which can be impractical in very high dimensions. In contrast, TT-Cross avoids the need for full tensor evaluation by constructing the decomposition from a relatively small, adaptively chosen set of tensor entries. The method is based on the principle of cross approximation, where one iteratively selects “skeleton” rows and columns in appropriate unfolding matrices. By exploiting the maximal-volume submatrices (*maximum volume* principle), TT-Cross identifies the most informative entries of the tensor and uses them to interpolate the remaining values. This process is carried out sequentially across tensor modes, updating ranks adaptively and ensuring numerical stability. As a result, TT-Cross can efficiently obtain the TT representation while querying and storing only a fraction of the entries, which makes it particularly well-suited for high-dimensional functions. For more details, please refer to (Oseledets and Tyrtyshnikov 2010; Savostyanov and Oseledets 2011).

Given a discretized TT representation, it can be extended to approximate continuous functions by interpolating across the tensor cores. For instance, by using linear interpolation between core slices, each core defines a matrix-valued interpolation:

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

valid for $x_k^{i_k} \leq x_k \leq x_k^{i_k+1}$. The resulting continuous approximation then takes the form

$$F(x_1, \ldots, x_d) \approx \boldsymbol{F}^1(x_1) \cdots \boldsymbol{F}^d(x_d),$$

allowing efficient representation and approximation of functions defined on mixed discrete–continuous domains.

3.2 Global Optimization via Tensor Train (TTGO)

The goal of global optimization is to identify decision variables $\boldsymbol{x}$ that maximize a target function $f(\boldsymbol{x})$. TTGO (Shetty et al. 2024) addresses this task by mapping $f(\boldsymbol{x})$ into an unnormalized density function $F(\boldsymbol{x})$ through a monotone transformation that preserves the ordering of optima. The function $F(\boldsymbol{x})$ is then approximated in Tensor Train (TT) format using the TT-Cross algorithm, resulting in a compact, structured representation:

$$F(x_1, \ldots, x_d) \approx \sum_{\gamma_1=1}^{r_1} \sum_{\gamma_2=1}^{r_2} \cdots \sum_{\gamma_{d-1}=1}^{r_{d-1}} \boldsymbol{\mathcal{F}}^1(1, x_1, \gamma_1)\, \boldsymbol{\mathcal{F}}^2(\gamma_1, x_2, \gamma_2) \cdots \boldsymbol{\mathcal{F}}^d(\gamma_{d-1}, x_d, 1), \tag{1}$$

where each TT core $\boldsymbol{\mathcal{F}}^k$ is a tensor of size $r_{k-1} \times N_k \times r_k$, with N_k denoting the discretization resolution of the k-th variable and r_k the associated TT ranks.

This TT representation yields a low-rank surrogate that enables efficient optimization. Instead of exhaustive grid search, which suffers from exponential complexity $\mathcal{O}(N^d)$, TTGO leverages the tensor structure to perform coordinated dimension-wise search, reducing the computational cost to $\mathcal{O}(Ndr^2)$, where N is the typical discretization size per dimension and r the maximal TT rank. However, TTGO relies heavily on accurate TT approximation, which may not hold under limited storage capacity.

3.3 Monte Carlo Tree Search

Given a decision tree, MCTS begins at the root and selects a promising node at each layer by balancing exploitation and exploration, typically via an upper confidence bound (UCB):

$$i_{[j]} \leftarrow \arg\max_{i_{[j]} \in \mathcal{I}_{[j]}} \left(\frac{w_{i_{[j]}}}{v_{i_{[j]}}} + c\sqrt{\frac{\log v_{i_{[j-1]}}}{v_{i_{[j]}}}} \right), \tag{2}$$

where j denotes the current depth in the tree, and $i_{[j]} = (i_1, i_2, \cdots, i_j)$ represents the complete sequence from the root to the selected node. $\mathcal{I}$ denotes the index set of all complete node sequences corresponding to branches in the decision tree, and $\mathcal{I}_{[j]}$ denotes the subset of index sequences truncated at depth j, i.e., all partial paths from the root to depth j. Here, $w_{i_{[j]}}$ and $v_{i_{[j]}}$ denote the cumulative reward and visit count of the terminal node in path $i_{[j]}$, while $v_{i_{[j-1]}}$ refers to the visit count of its parent. The constant c regulates the trade-off between exploration and exploitation.

After selection, MCTS expands the chosen node by generating a new child, followed by a simulation stage in which a rollout policy estimates the outcome from that child. The simulation results are then propagated back through the visited nodes, updating their value estimates and visit counts to guide subsequent searches. Through strategic exploitation and exploration, MCTS can efficiently explore large search spaces and asymptotically converge to the global optimum.

To support strategic search behavior, it is essential to store and update the value and visit count for each node. This leads to a combinatorial complexity of $\mathcal{O}(N^d)$ in both computation time and memory usage, where N is the number of nodes per layer and d is the depth of the tree. Such complexity severely limits the applicability of MCTS in many tasks.

4 Problem Formulation

In this section, we first present a general formulation that encompasses a wide range of robotic tasks. We then introduce the proposed algorithm, which leverages Tensor Train (TT) factorization for efficient Monte Carlo Tree Search (MCTS). Finally, we provide a theoretical analysis demonstrating convergence to the global optimum.

4.1 Problem formulation

We consider a general robot optimization formulation that can address diverse problems, such as inverse kinematics, motion planning, multi-stage motion planning with mode switching, and model predictive control, which corresponds to a large sets of mathematical programs, including nonlinear programming (NLP), large-scale (aka. high-dim) NLP and mixed-integer nonlinear programming (MINLP). In particular, we leverage basis functions to reduce the dimensionality in large-scale NLP formulations, while the choice of basis functions remains flexible.

We describe here the notation and variables:

- K the number of discrete modes (a.k.a. stages).
- $m_k \in \mathcal{M}$ the discrete *mode* at stage k, chosen from a finite (or countable) set $\mathcal{M}$.
- $a_k \in \mathcal{A}$ the discrete decision variable (action) at stage k, chosen from the finite action set $\mathcal{A}$.
- $\boldsymbol{x}_k^t \in \Omega_{\boldsymbol{x}} \subseteq \mathbb{R}^n$ a continuous state (or configuration) at stage k at time t.
- $\boldsymbol{u}_k^t \in \Omega_{\boldsymbol{u}} \subseteq \mathbb{R}^p$ a continuous control input at stage k at time t.
- T trajectory length for each stage.
- $B = B_1 + \cdots + B_K$, the total number of basis functions, where B_k denotes the number of basis functions at stage k.
- $\boldsymbol{\Psi}_k \in \mathbb{R}^{T \times B_k}$ a chosen set of basis functions for stage k, which is used to reconstruct the continuous decision variable from weights.
- $\boldsymbol{w}_k \in \Omega_{\boldsymbol{w}} \subseteq \mathbb{R}_k^B$ the vector of basis *weights* at stage k. Hence we let $\boldsymbol{u}_k^{[T]} = \boldsymbol{\Psi}_k\, \boldsymbol{w}_k$, which encodes the continuous variables in terms of basis representations.
- $c(m_k, a_k, x_k^t, u_k^t)$ stage cost (e.g., energy, distance, penalty) at time t in stage k.
- $c_{\text{terminal}}(x_K^T)$ terminal cost, e.g. capturing the final configuration error.
- $\phi(\cdot) \leq 0, \quad \psi(\cdot) = 0$ general inequality/equality constraints that can represent kinematic limits, collision avoidance, robot dynamics, or boundary conditions.

Note that we use a bracket subscript to indicate a sequence, e.g., $\boldsymbol{x}_k^{[T]}$ represents the full trajectory at stage k, and $\boldsymbol{u}_{[K]}^{[T]}$ represents the control variables including the complete long-horizon trajectory. The unified mathematical formulation for our robot optimization problems is:

$$\min_{a_{[K]}, \boldsymbol{u}_{[K]}^{[T]}} \sum_{k=0}^{K} \int_0^T c\big(m_k,\, a_k,\, \boldsymbol{x}_k^t,\, \boldsymbol{u}_k^t\big) dt \;+\; c_{\text{terminal}}\big(\boldsymbol{x}_K^T\big) \tag{3}$$

$$\text{s.t.} \quad \boldsymbol{u}_k^{[T]} = \sum_{b=0}^{B} \boldsymbol{\Psi}_k^b\, \boldsymbol{w}_k^b, \tag{4}$$

$$\forall_{k=0}^{K} \quad m_{k+1} \;\in\; \text{succ}\big(m_k,\, a_k,\, \boldsymbol{x}_k^{[T]},\, \boldsymbol{u}_k^{[T]}\big), \tag{5}$$

$$\forall_{k=0}^{K} \quad \phi\big(m_k,\, a_k,\, \boldsymbol{x}_k^{[T]},\, \boldsymbol{u}_k^{[T]}\big) \;\le\; 0, \tag{6}$$

$$\forall_{k=0}^{K} \quad \psi\big(m_k,\, a_k,\, \boldsymbol{x}_k^{[T]},\, \boldsymbol{u}_k^{[T]}\big) \;=\; 0, \tag{7}$$

$$(m_0,\, \boldsymbol{x}_0^0) = (m_{\text{init}},\, \boldsymbol{x}_{\text{init}}), \tag{8}$$

where:

- (4) encodes the full trajectory of decision variables using basis functions.
- (5) enforces the allowed transition to m_{k+1} in the discrete mode set $\mathcal{M}$, given the current mode m_k, discrete action a_k, and the continuous trajectories $\boldsymbol{x}_k^{[T]}$, $\boldsymbol{u}_k^{[T]}$.
- (6) and (7) represent the system dynamics, consistency of different modes, and other physical constraints.
- m_{init} and x_{init} denote the initial mode and state.

This formulation unifies many problems in robotics, including:

Inverse Kinematics (IK) For a basic IK problem, we can set $T = 1$ and have only a single mode $K = 1$. The joint configuration $\boldsymbol{u}_1 = \boldsymbol{\Psi}_1\, \boldsymbol{w}_1$ describes the joint configuration, namely

$$\min_{\boldsymbol{w}_1} \; c\big(\boldsymbol{\Psi}_1\, \boldsymbol{w}_1\big) \quad \text{s.t.} \quad \phi(\boldsymbol{\Psi}_1\, \boldsymbol{w}_1) \le 0, \quad \psi(\boldsymbol{\Psi}_1\, \boldsymbol{w}_1) = 0,$$

where c could measure the end-effector pose error relative to a target, and ϕ, ψ encode joint limits, collision avoidance, etc.

Motion Planning (MP) For a typical MP problem, we can set $K = 1$ to ensure there is only a single mode. The trajectory length is T, and the decision variables are the weights $\boldsymbol{w}$ for the basis function encoding.

Multi-stage Motion Planning (MsMP) For a multi-stage planning problem with multiple discrete contact modes (e.g. foot placements or manipulation primitives) and continuous joint trajectories, we use K to represent the number of overall stages. At each stage k:

$$a_k \;\in \mathcal{A}, \quad m_k \;\in \mathcal{M}, \quad \boldsymbol{u}_k^{[T]} \;=\; \boldsymbol{\Psi}_k\, \boldsymbol{w}_k.$$

The constraints ensure collision-free motion, piecewise dynamics, and valid contact transitions among modes. The task is to find both the discrete action sequence $a_{[K]}$ and the continuous motion trajectory $\boldsymbol{u}_{[K]}^{[T]}$.

5 Tensor Train Tree Search

Equation (3) provides a general formulation that captures many optimization problems in robotics. This formulation belongs to the class of Mixed-Integer Nonlinear Programming (MINLP) problems, a class that is known to be NP-hard (Liberti 2019). MCTS has emerged as a powerful tool for addressing such problems through strategic exploration of the solution space, but it suffers from combinatorial complexity. In this section, we introduce TT as an efficient representation for decision trees that can significantly reduce complexity from combinatorial to linear by exploiting correlations among branches.

Consider a problem with K stages, m elements in the discrete action set $\mathcal{A}$, B basis functions, and discretization granularity set to N. To solve such an optimization problem, we can construct a decision tree of depth $d = K + B$, where the first K layers each have m nodes and the remaining B layers each have N nodes. This multi-layer decision tree can be represented as a high-dimensional tensor $\mathbf{T}$ of size $m^K N^B$, where each complete branch corresponds to a particular element of $\mathbf{T}$, and the element indices represent the sequence of node indices along that branch. As illustrated in Figure 2 (A), the two representations are equivalent. For convenience, we denote the size of the tensor $\mathbf{T}$ as N^d throughout this work. This notational simplification enables a unified representation and highlights the exponential dependence on the total depth $d = K + B$, which constitutes the dominant factor in the subsequent complexity analysis. If the optimal solution in the tree is indexed by $(i_1, i_2, \ldots, i_d)$, where i_j denotes the node index at the j-th layer, then the corresponding optimal tensor entry has the same index.

To reduce this combinatorial complexity, our objective is to express the tree with linear complexity $\mathcal{O}(\lambda N d)$, where λ is a scaling factor. This is equivalent to representing a high-dimensional tensor of size $\mathcal{O}(N^d)$ using a compact, low-rank decomposition, which is a well-studied problem in numerical mathematics (Cichocki et al. 2016). Specifically, we employ TT to exploit the separable structure and reduce inter-layer dependencies. As a result, the original high-dimensional tensor can be efficiently represented using significantly fewer parameters through a sequence of third-order cores, namely:

$$\mathbf{T}_{(i_1,\ldots,i_d)} \approx \boldsymbol{\mathcal{T}}^1_{:,i_1,:} \boldsymbol{\mathcal{T}}^2_{:,i_2,:} \cdots \boldsymbol{\mathcal{T}}^d_{:,i_d,:}, \tag{9}$$

where $\boldsymbol{\mathcal{T}}^j$ is the core corresponding to the j-th dimension (i.e., the j-th layer of the tree). Figure 2 (B) presents an illustrative example of a three-dimensional tensor.

Algorithm 1 presents the pseudocode of the proposed method. As indicated in (2), a key component of MCTS is its ability to perform strategic search by leveraging the value and visit count information stored at each node in the tree, denoted by $\boldsymbol{Q}$ and $\boldsymbol{V}$, respectively. However, $\boldsymbol{Q}$ and $\boldsymbol{V}$ are high-dimensional tensors, which are impractical to store due to the combinatorial complexity. In this work, we utilize tensor factorization to approximate them in TT format, denoted as $\boldsymbol{\mathcal{Q}}$ and $\boldsymbol{\mathcal{V}}$.

To initialize the search, TT-Cross is first employed to approximate the decision tree using a low-rank TT model, augmented with conditional variables (e.g., task parameters or initial states), resulting in a conditional model $\widetilde{\boldsymbol{\mathcal{Q}}}$. Given a set of condition variables $\boldsymbol{z}$, the value tensor is instantiated as $\boldsymbol{\mathcal{Q}} = \widetilde{\boldsymbol{\mathcal{Q}}}(\boldsymbol{z})$. Owing to the *maximum volume* principle used in TT-Cross, the resulting TT model typically captures representative features of the decision tree, enabling the

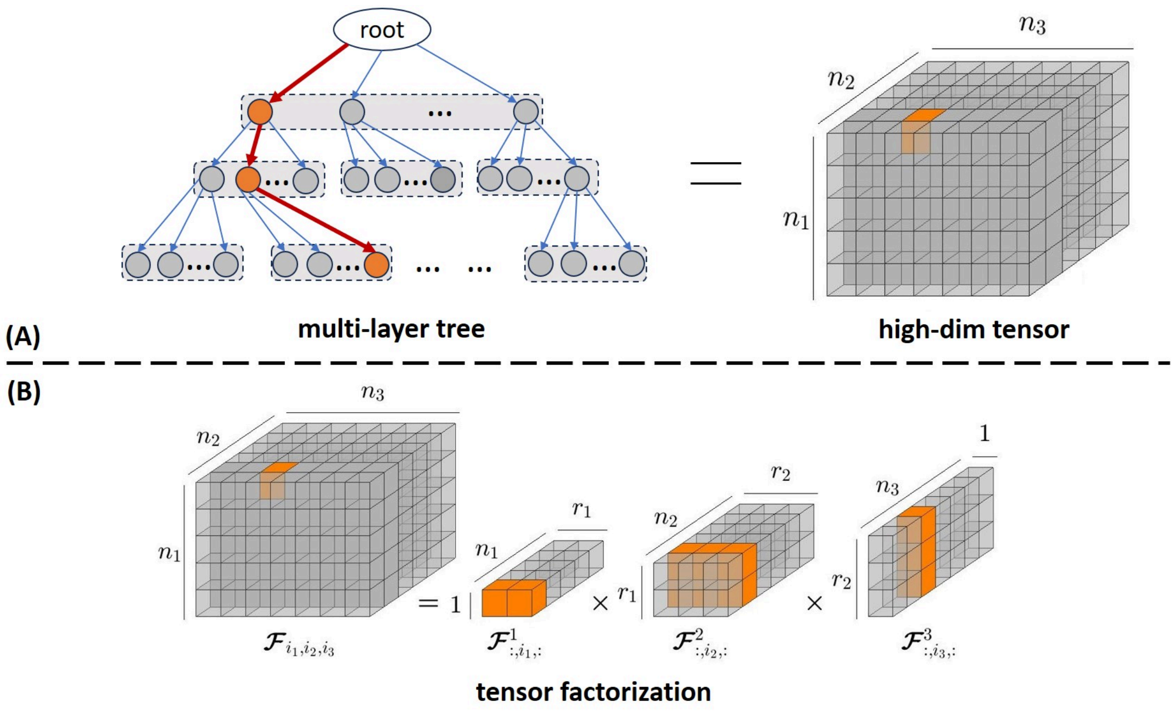

root
multi-layer tree
high-dim tensor
(A)
(B)
tensor factorization
n_1
n_2
n_3
r_1
r_2
$\mathcal{F}_{i_1,i_2,i_3}$
$= 1$
$\mathcal{F}^1_{:,i_1,:}$
$\mathcal{F}^2_{:,i_2,:}$
$\mathcal{F}^3_{:,i_3,:}$

Figure 2. Tree-Tensor-TT transformation. (A) A multi-layer decision tree can be equivalently represented as a high-dimensional tensor, where each tensor element corresponds to the value at the terminal node of a branch. **(B)** Tensor decomposition in TT format. A 3-dimensional tensor can be represented using three third-order TT cores.

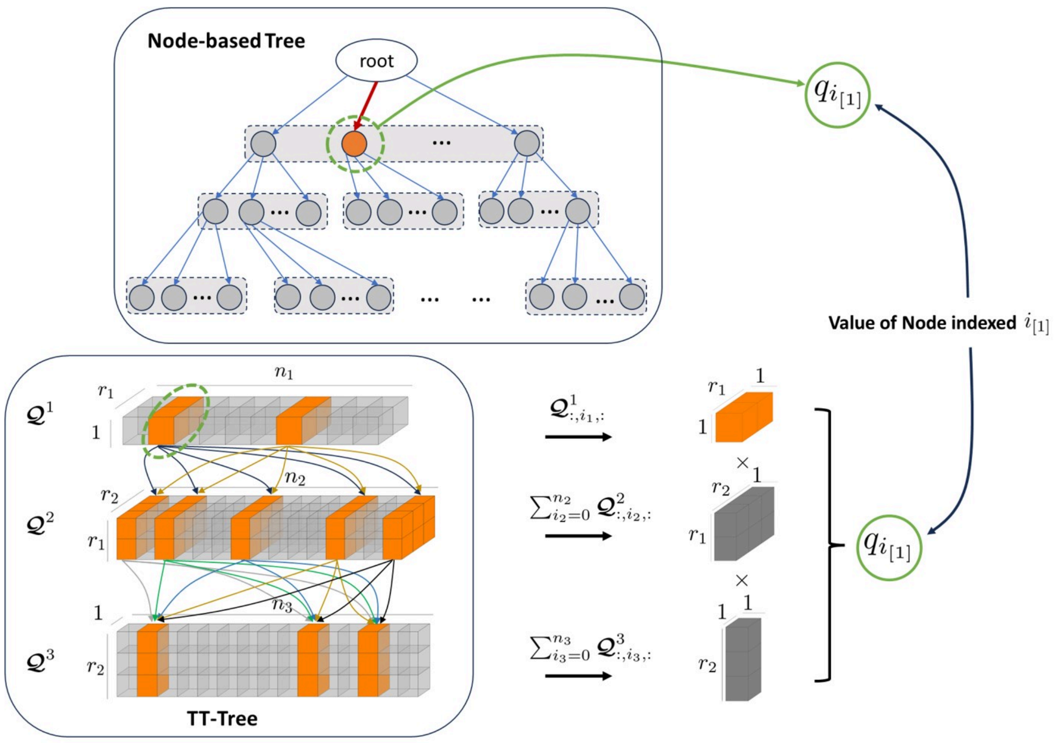

Node-based Tree
root
$q_{i_{[1]}}$
Value of Node indexed $i_{[1]}$
$\mathcal{Q}^1$
$\mathcal{Q}^2$
$\mathcal{Q}^3$
n_1
n_2
n_3
r_1
r_2
TT-Tree
$\mathcal{Q}^1_{:,i_1,:}$
$\sum_{i_2=0}^{n_2} \mathcal{Q}^2_{:,i_2,:}$
$\sum_{i_3=0}^{n_3} \mathcal{Q}^3_{:,i_3,:}$

Figure 3. Node value computation given a tree in TT format. This example illustrates how the value of a node in the first layer is computed using TT cores.

identification of promising branches from the very first iteration. The visit count tensor $\boldsymbol{\mathcal{V}}$ is initialized to zero.

The algorithm then follows the standard MCTS pipeline: selection, expansion, simulation, and backpropagation. Due to dependence on the parent nodes, the value and visitation statistics of a node are path-dependent, introducing non-Markovian behavior. Querying the value of node $n_{i_{[j]}}$ requires summing over all completions of the branch from layer $j+1$ to depth d. Given the full tree approximated in TT format, the value of a node at level j, denoted as $q_{i_{[j]}}$, can be efficiently computed by

$$q_{i_{[j]}} = \sum_{i_{j+1}=0}^{N_{j+1}} \cdots \sum_{i_d=0}^{N_d} \boldsymbol{Q}_{(i_1,\ldots,i_j,i_{j+1},\ldots,i_d)}$$
$$\approx \boldsymbol{\mathcal{Q}}^1_{:,i_1,:} \cdots \boldsymbol{\mathcal{Q}}^j_{:,i_j,:} \sum_{i_{j+1}=0}^{N_{j+1}} \cdots \sum_{i_d=0}^{N_d} \boldsymbol{\mathcal{Q}}^{j+1}_{:,i_{j+1},:} \cdots \boldsymbol{\mathcal{Q}}^d_{:,i_d,:}, \tag{10}$$

where each TT core $\boldsymbol{\mathcal{Q}}^l$ encodes the factorized structure of the tree at layer l. To compute the node value at depth j, we extract the corresponding slices from the first j TT cores (representing the visited layers), and perform summation over indices in the remaining cores from $j+1$ to d. The final node value is obtained by multiplying the resulting matrices across all layers. Figure 3 illustrates how to compute the value of a node in the first layer of a three-layer tree using TT cores. The multiple arrows between TT cores indicate the ability to perform parallel selection in the TT-Tree.

Similarly, the visit count $v_{i_{[j]}}$ is computed as:

$$v_{i_{[j]}} = \sum_{i_{j+1}=0}^{N_{j+1}} \cdots \sum_{i_d=0}^{N_d} \boldsymbol{V}_{(i_1,\ldots,i_j,i_{j+1},\ldots,i_d)}$$
$$\approx \boldsymbol{\mathcal{V}}^1_{:,i_1,:} \cdots \boldsymbol{\mathcal{V}}^j_{:,i_j,:} \sum_{i_{j+1}=0}^{N_{j+1}} \cdots \sum_{i_d=0}^{N_d} \boldsymbol{\mathcal{V}}^{j+1}_{:,i_{j+1},:} \cdots \boldsymbol{\mathcal{V}}^d_{:,i_d,:}. \tag{11}$$

Given $q_{i_{[j]}}$, $v_{i_{[j]}}$, and $v_{i_{[j-1]}}$, node expansion follows the UCB rule from (2). The TT-based representation enables parallel selection, which is typically non-trivial in traditional node-based or table-based implementations (Chaslot et al. 2008). If no further expansion is possible, the current branch terminates. Otherwise, the branch is extended by one layer and followed by a simulation to the leaf. Rather than performing a random rollout, we leverage the TT value model $\boldsymbol{\mathcal{Q}}$ as a global approximation of the decision tree to guide the simulation strategy via stochastic sampling, treating TT values as unnormalized probabilities. This enables parallel simulation and yields a top-τ set of candidate solutions. After the simulation phase, optimal branches and their corresponding indices are identified, followed by backpropagation to update $\boldsymbol{\mathcal{Q}}$ and $\boldsymbol{\mathcal{V}}$ in preparation for the next iteration. During backpropagation, the visit count model $\boldsymbol{\mathcal{V}}$ is updated using a residual strategy. We define $\Delta \boldsymbol{V}^\ell_\gamma$ as the tensor that records which nodes are visited at layer γ of iteration ℓ. It contains only binary values: 1 indicates that a node is visited, while 0 indicates it is unvisited. Since the newly visited indices are known, we adapt the standard TT-Cross algorithm into a guided version, which can obtain the TT representation of $\Delta \boldsymbol{V}^\ell_\gamma$ using only one iteration by directly inputting the indices corresponding to 1. The value model $\boldsymbol{\mathcal{Q}}$ can also be updated in a similar manner to enhance approximation accuracy, but this step can be omitted in practice to reduced runtime. In each iteration, the discrete action solution set $\mathcal{S}_a$ and the basis weight solution set $\mathcal{S}_w$ are refreshed to maintain the top-τ candidates. Finally, the candidates obtained from TT-Tree Search are further refined using CMA-ES to correct for discretization errors in the continuous domain.

Algorithm 1 Tensor Train Tree Search (TTTS)

```
1:  function TensorTrainTreeSearch($\boldsymbol{x}_0$, $\boldsymbol{z}$, $\widetilde{\mathbf{Q}}$, $J$, $L$, $\mathcal{A}$, $\Omega_w$, $\mathcal{I}$)
2:      Input: Initial state $\boldsymbol{x}_0$, Condition var. $\boldsymbol{z}$, Maximum iter. $L$,
3:        Augmented decision tree $\widetilde{\mathbf{Q}}$, Objective function $J$,
4:        Action domain $\mathcal{A} = \{(a_1^{i_1}, \ldots, a_K^{i_K}) : i_k \in \{1, \ldots, N_k\}\}$,
5:        Weight domain $\Omega_w = \{(w_1^{i_1}, \ldots, w_B^{i_B}) : i_b \in \{1, \ldots, N_b\}\}$,
6:        Index set $\mathcal{I} \subseteq \{1, \ldots, N_1\} \times \cdots \times \{1, \ldots, N_d\}, d = K + B$
7:      Hyperparameters: Number of solutions $\tau$,
8:        Exploration param. $c$ # default: $\tau = 10, c = 3$
9:      Output: top-$\tau$ solutions: discrete action set $\mathcal{S}_a$, basis weight set $\mathcal{S}_w$
10:     // ===== TT-Tree Initialization =====
11:     $\widetilde{\boldsymbol{\mathcal{Q}}}$ = TT-Cross($\widetilde{\mathbf{Q}}$), $\boldsymbol{\mathcal{Q}} \leftarrow \widetilde{\boldsymbol{\mathcal{Q}}}(\boldsymbol{z})$
12:     // ===== TT-Tree Search =====
13:     $i_0 \leftarrow$ Node($\boldsymbol{x}_0$), $\boldsymbol{\mathcal{V}}_0 \leftarrow \mathbf{0}$, $\mathcal{S}_a = [\,]$, $\mathcal{S}_w = [\,]$
14:     for $\ell = 1, 2, \ldots$, L do
15:         for $j = 1, \ldots, d$ do
16:             $q_{i_{[j]}} \leftarrow$ Eq. (10), $v_{i_{[j]}}, v_{i_{[j-1]}} \leftarrow$ Eq. (11)
17:             $i_{[j]} \leftarrow \arg\max^\tau_{i_{[j]} \in \mathcal{I}_{[j]}} \left( \frac{q_{i_{[j]}}}{v_{i_{[j]}}} + c \cdot \sqrt{\frac{\log v_{i_{[j-1]}}}{v_{i_{[j]}}}} \right)$
18:             if $i_j$ is not expanded then break
19:         end for
20:         for $s = j+1, \ldots, d$ do
21:             $q_{i_{[s]}} \leftarrow$ Eq. (10)
22:             $i_{[s]} \leftarrow \arg\max^\tau_{i \in \mathcal{I}_{[s]}} q_{i_{[s]}}$
23:         end for
24:         $a^\star, \boldsymbol{w}^\star \leftarrow \arg\max^\tau_{(a,\boldsymbol{w}) \in (\mathcal{A},\Omega_w)(i_{[d]})} J(a, \boldsymbol{w})$
25:         for $\gamma = 1, \ldots, j$ do
26:             $\boldsymbol{\mathcal{V}}^\ell_\gamma = \boldsymbol{\mathcal{V}}^{\ell-1}_\gamma$ + Guided-Cross($\Delta \boldsymbol{V}^\ell_\gamma, i_{[\gamma]}$)
27:         end for
28:         $\mathcal{S}_a^{aug} \leftarrow$ append($\mathcal{S}_a, a^\star$), $\mathcal{S}_w^{aug} \leftarrow$ append($\mathcal{S}_w, \boldsymbol{w}^\star$)
29:         $\mathcal{S}_a, \mathcal{S}_w \leftarrow \arg\max^\tau_{a \in \mathcal{S}_a^{aug}, \boldsymbol{w} \in \mathcal{S}_w^{aug}} J(a, \boldsymbol{w})$
30:     end for
31:     // ===== TT-Tree Refinement =====
32:     $\mathcal{S}_w \leftarrow$ CMA-ES($\mathcal{S}_w$)
33:     return $\mathcal{S}_a, \mathcal{S}_w$
34: end function
```

5.1 Theoretical analysis

A tensor is considered low-rank if it can be well approximated using a decomposition format such as CP (Harshman 1970), Tucker (Tucker 1963), or TT (Oseledets 2011), where the ranks are significantly smaller than the original tensor dimensions. For example, in the TT format, a tensor is low-rank if the TT ranks of the third-order cores satisfy $r_j \ll N_j$ for all j. Since a decision tree can be equivalently represented as a tensor, we define a decision tree as *low-rank* if its corresponding tensor is low-rank.

Here, we establish three theoretical results demonstrating that TTTS converges efficiently to a bounded-error solution of the optimization problem in (3). First, we show that the global optimum of a TT model can be retrieved efficiently with linear complexity. Next, we prove that the discrepancy between the TT solution and the global optimum is bounded by the TT approximation error. Finally, we establish the global convergence of TTTS, enabled by the exploration–exploitation mechanism of MCTS.

Proposition 1. *Given a TT model $\boldsymbol{\mathcal{T}}$, the global optimum can be efficiently retrieved with linear time complexity.*

Proof. Given a TT representation, the value of any specific element can be accessed as:

$$\mathcal{T}(x_1,\dots,x_d) = \sum_{\gamma_1=1}^{r_1}\sum_{\gamma_2=1}^{r_2}\cdots\sum_{\gamma_{d-1}=1}^{r_{d-1}} \mathcal{T}^1(1,\,x_1,\,\gamma_1)\mathcal{T}^2(\gamma_1,\,x_2,\,\gamma_2)\cdots\mathcal{T}^d(\gamma_{d-1},\,x_d,\,1), \tag{12}$$

where each $\mathcal{T}^k$ is a TT core of size $r_{k-1}\times N_k\times r_k$, with N_k denoting the number of discretization points for x_k, and r_k representing the TT ranks, which are small for low-rank trees.

This shows that each element of $\mathcal{T}$ can be expressed as a finite sum of products of separable TT cores. As a result, the global optimum of $\mathcal{T}$ can be found through a dimension-wise optimization procedure that does not require convexity or differentiability. This process is analogous to traversing a fully evaluated tree: by selecting the node with the highest value at each layer, the optimal branch can be identified.

Specifically, we carry out this process in TT format. As illustrated in Figure 2, a decision tree can be equivalently represented as a tensor, which can then be expressed in TT format for efficient computation. Based on (10), we compute the node values for each layer and choose the one with the highest value. By recursively traversing from the root to the final layer, the global optimum of the TT model is obtained, with a linear time complexity $\mathcal{O}(Ndr^2)$.

Proposition 2. *Consider a high-dimension tensor* $\mathbf{T}$ *that is approximated by a low-rank TT model* $\mathcal{T}$*, with approximation error* ϵ*, the error of the found solution compared with the maximum value of* $\mathbf{T}$ *is bounded with* 2ϵ*, namely*

$$\|\max\mathbf{T}-\mathbf{T}(\arg\max\mathcal{T})\|\le 2\epsilon \tag{13}$$

Proof. Consider a discrete search space Ω, with $|\Omega|<\infty$. We aim to find

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}\in\Omega}\mathbf{T}(\boldsymbol{x}),\qquad \mathbf{T}^* = \max_{x\in\Omega}\mathbf{T}(x).$$

Suppose we first approximate $\mathbf{T}$ by a low-rank TT model $\mathcal{T}$, satisfying

$$\|\mathcal{T}-\mathbf{T}\|_\infty \;=\; \max_{x\in\Omega}|\mathcal{T}(\boldsymbol{x})-\mathbf{T}(\boldsymbol{x})| \;\le\; \epsilon.$$

Then we can prove that

$$\begin{aligned}&\|\max\mathbf{T}-\mathbf{T}(\arg\max\mathcal{T})\|\\ \le\;&\|\max\mathbf{T}-\max\mathcal{T}\|+\|\max\mathcal{T}-\mathbf{T}(\arg\max\mathcal{T})\|\\ \le\;&2\epsilon\end{aligned} \tag{14}$$

Proposition 3. *TTTS retains the property of asymptotic global convergence of MCTS.*

Proof. In a MCTS framework (with a finite-depth tree corresponding to the coordinates of $\boldsymbol{x}\in\Omega_x$), we initialize each leaf node $\boldsymbol{x}$ with $Q_0(\boldsymbol{x})=\mathcal{Q}(\boldsymbol{x})$. A standard UCB-based selection rule then balances exploitation (nodes with high Q) and exploration (nodes that are under-visited).

Because Ω_x is finite, a classical result states that if every node is explored infinitely often (i.e., the algorithm does not permanently abandon any branch), MCTS converges almost surely to the global maximum:

$$\lim_{l\to\infty}\;\max_{\boldsymbol{x}\in\Omega_x}\left|\widehat{Q}_l(\boldsymbol{x})-\mathbf{T}(\boldsymbol{x})\right| \;=\; 0,$$

hence $\arg\max_{\boldsymbol{x}}\widehat{Q}_l(\boldsymbol{x}) \to \arg\max_{\boldsymbol{x}}\mathbf{T}(\boldsymbol{x})$, where $\widehat{Q}_l$ is the node value estimate after l iterations. Because TTTS does not alter the UCB exploration mechanism, this guarantee is preserved.

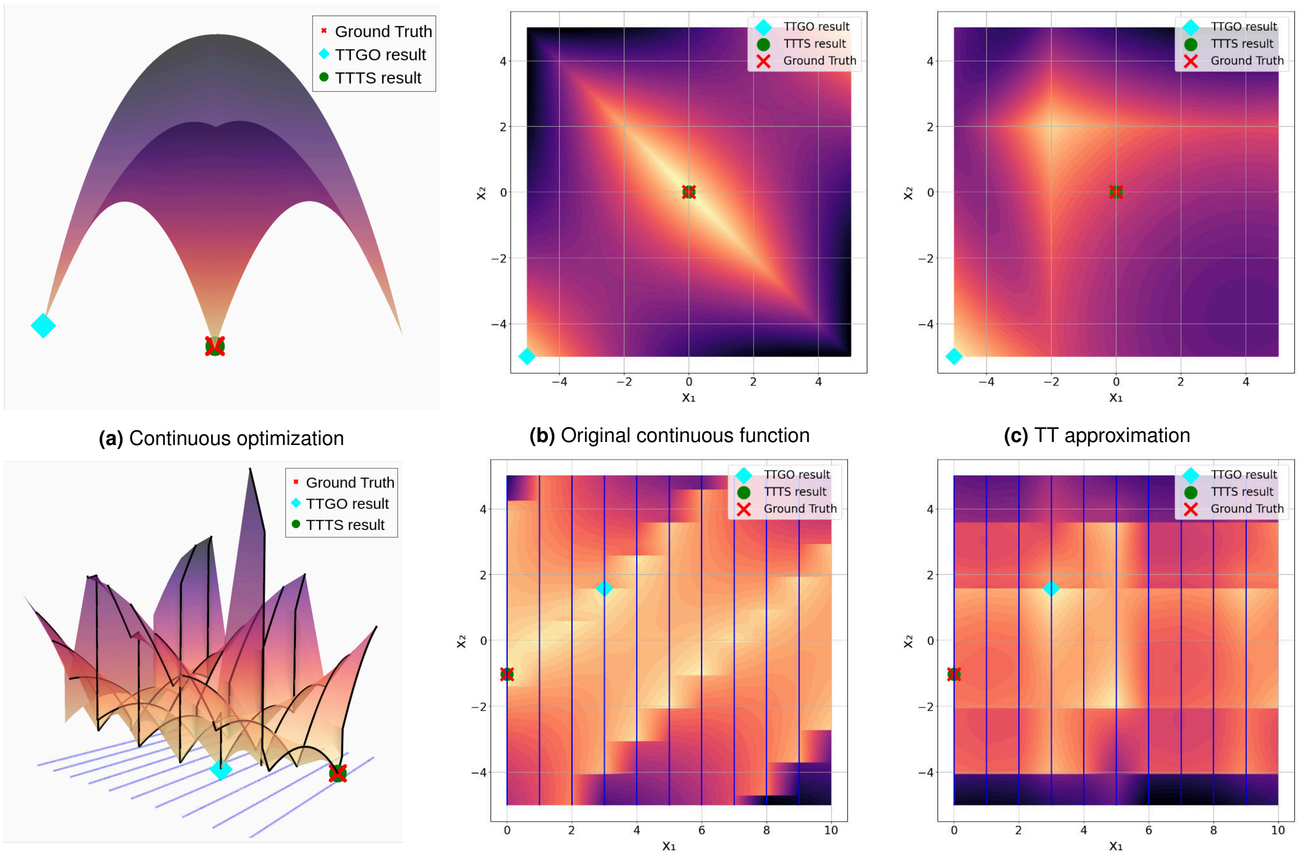
6 Experimental Results

We first applied our approach to simple continuous optimization and mixed-integer programming problems to motivate our work and provide intuition for the reader. We then demonstrate the effectiveness of the proposed mathematical formulation and solver on a diverse set of robotic tasks, including inverse kinematics, motion planning around obstacles, legged robot manipulation, multi-stage motion planning, and bimanual whole-body manipulation. The breadth of these experiments highlights the general applicability of TTTS, with a wide range of robot optimization problems. We also conduct ablation studies to evaluate the contribution of each component, along with numerical comparisons against state-of-the-art baselines to assess overall performance. The hyperparameters and cost functions used for each task are detailed in the appendix.

6.1 Toy examples

6.1.1 Continuous non-convex optimization. Many optimization problems in robotics are non-convex due to obstacles and nonlinear system dynamics. To demonstrate the capability of TTTS in solving such problems, we begin by optimizing a simple non-convex continuous function (15) to build intuition. As shown in Figure 4a, the function exhibits multiple local optima, highlighting strong multi-modality. Such multi-modality often causes gradient-based methods and single-modal evolutionary algorithms (e.g., CMA-ES) to become trapped in local optima. In contrast, the Tensor Train (TT) approach reformulates the optimization problem as density estimation, enabling it to capture multi-modal structures. The resulting factorized representation reveals a separable structure across dimensions, which facilitates the fast discovery of globally optimal solutions. However, existing TT-based optimization methods (Shetty et al. 2024; Sozykin et al. 2022) typically require the function to be low-rank, but they lack completeness once the rank exceeds the storage limit. For instance, Figure 4a can be viewed as a high-rank cost function, and its discrete matrix analogue, shown in Figure 4b, clearly exhibits full-rank characteristics. To illustrate both the strengths and limitations of low-rank approximation (i.e., approximating a high-rank function using a tensor with lower ranks), we set the TT rank to $r_{max}=2$ and construct TT approximation using TT-Cross.

The results are depicted in Figure 4c, which shows that, despite using a much lower rank, TT-Cross effectively captures the structure of the function and enables rapid localization of regions likely to contain optimal solutions. However, TT-Cross relies on cross approximation through



(a) Continuous optimization **(b)** Original continuous function **(c)** TT approximation

(d) Mixed-integer optimization **(e)** Original mixed-integer function **(f)** TT approximation

Figure 4. Toy examples of function optimization with low-rank tensor-train (TT) approximations. (a–c) Continuous non-convex optimization: (a) landscape of a multi-modal non-convex function $f_1(x_1, x_2)$; (b) its contour map; (c) TT-Cross reconstruction with maximal TT rank $r_{\max} = 2$, which captures the dominant structure but smooths out fine details. (d–f) Mixed-integer non-convex optimization: (d) landscape of $f_2(x_1, x_2)$ with both discrete (x_1) and continuous (x_2) variables; (e) its grid-aligned contour plot; (f) TT approximation, which highlights promising regions but may miss the true global optimum—motivating a TTTS strategy that couples TT approximation with tree search. Markers indicate solutions obtained by TTGO and TTTS compared with the ground-truth optimum.

submatrices, but selecting these submatrices is an NP-hard problem. And the original function may have a significantly higher rank than the maximum TT-rank permitted by storage constraints, which limits the accuracy of TT approximation. Due to these factors, TT-Cross alone can sometimes yield imprecise function approximations, adversely impacting the solution-finding process. TTGO relies on TT-Cross to obtain the TT model, where the quality of the optimization solution depends heavily on the accuracy of this approximation. This limitation motivates the proposed TTTS method, which combines the redundancy-reduction ability of TT approximation with the strategic search capabilities of MCTS. This synergy enables TTTS to efficiently converge to the global optimum, as illustrated in Figure 4a.

6.1.2 Mixed-integer programming. We further demonstrate the effectiveness of our approach in tackling a more challenging problem: mixed-integer nonlinear programming (MINLP). Such problems include both non-convexity due to nonlinear constraints and the combinatorial complexity imposed by integer variables. This setting is particularly difficult because integer variables are discrete and lack gradient information, rendering standard nonlinear programming (NLP) solvers ineffective. An alternative is to discretize the continuous variables and solve the problem using discrete search methods (e.g., A* or MCTS), but these approaches typically suffers from the curse of dimensionality and are computationally inefficient.

Figure 4d illustrates a simple MINLP example, where x_1 is an integer variable ranging from 0 to 10, and x_2 is a continuous variable. The cost function (16) is nonlinear and exhibits multiple local optima. Figure 4e presents the discrete matrix analogue of the continuous cost function. We approximate the function in TT format with rank $r_{\max} = 2$, as shown in Figure 4f. The results indicate that even with a low TT rank, TT-Cross can identify high-quality local optima, demonstrating the strong modeling and optimization capabilities of TT. However, the results obtained by TTGO do not correspond to the global optimum, highlighting that a low-rank TT approximation may fail to fully capture the complexity of the original function. In such cases, TTTS leverages strategic search to efficiently converge to the global optimal solution within finite time.

6.2 Inverse kinematics

We consider a standard inverse kinematics (IK) task, where the objective is to compute a collision-free configuration of a robot manipulator such that the end-effector reaches a desired target point. This corresponds to a single-stage, one-step optimization problem (i.e., $T = 1$ and $K = 1$ in (3)) with continuous decision variables $\boldsymbol{u}$, subject to kinematic and collision constraints. The objective function minimizes task-space error, which penalizes deviations between the end-effector position and the desired target. Additional constraints include joint limits, collision avoidance, and reachability. Despite its simple definition, this task serves as a meaningful testbed featuring nonlinear kinematics and nonconvex constraints, making it well-suited to evaluate the effectiveness of TT in enabling efficient tree search.

We randomly generated five targets in the ablation study to analyze the necessity of each component of TTTS, which consists of three main components: TT-Tree Initialization, TT-Tree Search, and TT-Tree Refinement. TT-Tree Initialization typically requires some computation time due to TT-Cross approximation (particularly for high-dimensional systems), but this process is performed offline. At this step, the state space is augmented with task variables, enabling rapid task-conditioned retrieval for TT-Tree Search. In our experiments, we set $r_{\max} = 21$ for both the 3-joint and 7-joint manipulators (which we refer to as *IK1* and *IK2* in Figure 8), resulting in a coarse approximation of the full decision tree. Notably, obstacle avoidance tends to introduce high-rank behavior, making low-rank TT approximations less accurate. As shown in Figure 5 (A), TT-Tree Initialization alone does not yield the global optimum, highlighting the limitations of using TT approximation by itself. However, after performing TT-Tree Search and Refinement, the final task-space error is zero, indicating that the globally optimal solution was successfully found.

We further compared our proposed approach with state-of-the-art baselines, including TTGO, MCTS, and CMA-ES. Both TTGO and MCTS provide valuable insights that inspire our method, and each can be considered a special case of TTTS. TTGO samples solutions directly from the TT approximation, which can result in repeated sampling of the same solutions. In contrast, our approach incorporates the strategic search capability of MCTS, which ensures global convergence. CMA-ES is a well-known sampling-based optimization technique that does not rely on gradient information, allowing it to avoid poor local optima. This makes it suitable for many robotic optimization problems involving nonlinear dynamics or obstacle avoidance. Figure 8 shows the comparison between our approach and the baselines. Except for TTGO, all other methods found the global optimal solutions. This is because TT-based approximation reduces tree redundancy but lacks high precision. While it can highlight promising regions, it cannot reliably identify the exact solution. Moreover, the sampling-based refinement in TTTS is better suited to address the non-convexity commonly present in robotic optimization problems, outperforming the gradient-based refinement used in TTGO. Among the three approaches that reach the global optimum, TTTS requires the least time, highlighting the effectiveness of TT approximation for efficient tree search.

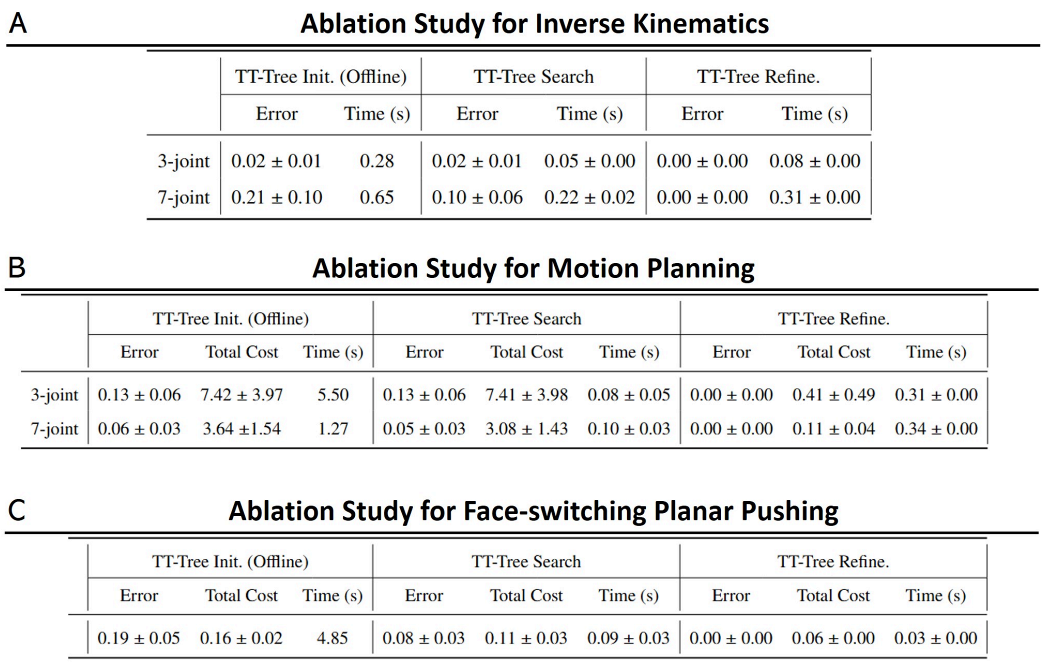
6.3 Motion planning around obstacles

We further applied our framework to a motion planning problem in which a robot must generate a smooth, collision-free trajectory from a given start to a goal configuration. The problem is formulated over T time steps, where the continuous trajectory is represented using basis functions as $\boldsymbol{u}^{[T]} = \sum_{b=1}^{B} \boldsymbol{\Psi}^b \boldsymbol{w}^b$, and the optimization variables are the corresponding weights $\{\boldsymbol{w}^b\}_{b=1}^{B}$. The objective is to minimize a cost function that encourages smooth motion (e.g., penalizing velocity or acceleration), while ensuring accurate goal reaching. The constraints include joint limits and collision avoidance with static obstacles.

Figure 6 (A) presents our first test scenario: a 3-joint manipulator reaching task, which we refer to as *MP1*. This task is particularly challenging because the target lies on the opposite side of the robot, with two circular obstacles obstructing the direct path. The trajectories found by our approach exhibit multiple solution modalities under the same initial configuration and target. Notably, the first two trajectories shown in the figure require the manipulator to initially move away from the target and then pass through a narrow passage between the obstacles, which is not easy to find and requires long-horizon anticipation. This setting involves numerous local optima and a narrow feasible solution space, necessitating long-horizon planning rather than short-horizon control. The results illustrate our framework's ability to overcome local optima and support long-term decision-making. We further applied our approach to a 7-joint manipulator reaching task, which we refer to as *MP2*. The robot arm must move its end-effector from one level of the shelf to another while avoiding obstacles, such as the shelf frame, across its entire body surface. This setup is representative of daily tasks such as pick-and-place or bookshelf arrangement. Figure 1 (B) shows the resulting trajectories, where end-effector paths are visualized as curves. Different colors indicate distinct solutions discovered by the algorithm.

Figure 5 (B) presents the ablation study of our approach applied to motion planning (MP) problems. As the three stages of TTTS are applied, both the final state error and the total trajectory cost consistently decrease, demonstrating the effectiveness of each component. The final errors for both tasks are zero, indicating that a collision-free joint trajectory was successfully found to reach the target. An interesting observation is that TT-Tree Initialization takes longer for task *MP1* than for *MP2*, as *MP1* is a more challenging problem with more local optima. Accordingly, we set $r_{\max} = 41$ for *MP1* and $r_{\max} = 21$ for *MP2*. The TT approximation in *MP1* achieves high accuracy, and further iterations of TT-MCTS provide limited improvement. This suggests that, with sufficient storage capacity, we can use higher TT ranks for more accurate tree approximation, accelerating online inference. In contrast, the results of *MP2* highlight the effectiveness of TTTS under limited storage conditions, where low-rank approximations still capture informative representations of complex decision trees. By combining low-rank approximation with the strong exploration capability of MCTS, TTTS ensures convergence to the bounded global optimum.

Figure 8 compares TTTS with other optimization methods in terms of *Final Error*, *Total Cost*, and *Runtime*. *Final*



A **Ablation Study for Inverse Kinematics**

	TT-Tree Init. (Offline)		TT-Tree Search		TT-Tree Refine.	
	Error	Time (s)	Error	Time (s)	Error	Time (s)
3-joint	0.02 ± 0.01	0.28	0.02 ± 0.01	0.05 ± 0.00	0.00 ± 0.00	0.08 ± 0.00
7-joint	0.21 ± 0.10	0.65	0.10 ± 0.06	0.22 ± 0.02	0.00 ± 0.00	0.31 ± 0.00

B **Ablation Study for Motion Planning**

	TT-Tree Init. (Offline)			TT-Tree Search			TT-Tree Refine.		
	Error	Total Cost	Time (s)	Error	Total Cost	Time (s)	Error	Total Cost	Time (s)
3-joint	0.13 ± 0.06	7.42 ± 3.97	5.50	0.13 ± 0.06	7.41 ± 3.98	0.08 ± 0.05	0.00 ± 0.00	0.41 ± 0.49	0.31 ± 0.00
7-joint	0.06 ± 0.03	3.64 ±1.54	1.27	0.05 ± 0.03	3.08 ± 1.43	0.10 ± 0.03	0.00 ± 0.00	0.11 ± 0.04	0.34 ± 0.00

C **Ablation Study for Face-switching Planar Pushing**

TT-Tree Init. (Offline)			TT-Tree Search			TT-Tree Refine.		
Error	Total Cost	Time (s)	Error	Total Cost	Time (s)	Error	Total Cost	Time (s)
0.19 ± 0.05	0.16 ± 0.02	4.85	0.08 ± 0.03	0.11 ± 0.03	0.09 ± 0.03	0.00 ± 0.00	0.06 ± 0.00	0.03 ± 0.00

Figure 5. Ablation studies for diverse robotics tasks. *Error* denotes the ℓ_2 norm between the final and target configurations. *Total Cost* refers to the value of the cost function for the obtained solution. *Time* indicates the runtime required to find the solution.

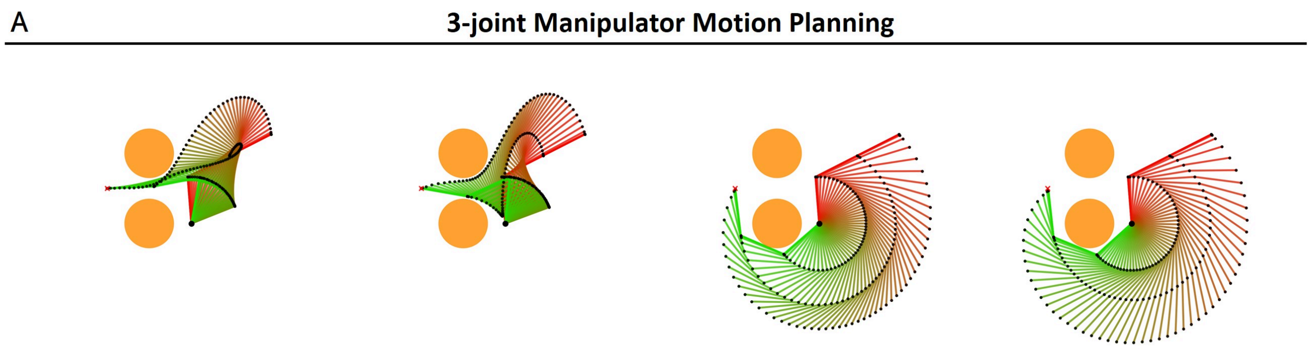
Error denotes the ℓ_2 norm between the system's final and target configurations. *Total Cost* refers to the value of the cost functions evaluated at the obtained solution, with the cost functions described in Section A.2. *Runtime* indicates the time required to find the solution. From the *Final Error* and *Total Cost*, we observe that TTTS achieves results comparable to MCTS when the latter is given sufficient time, highlighting TTTS's ability to reach similar global solutions. The *Runtime* further shows that TTTS requires significantly less computation time. TTGO performs well for *MP2*, but it struggles in *MP1* due to the complex cost landscape, which shows TT approximation alone is insufficient to capture all necessary features. In contrast, TTTS leverages strategic tree search to explore the decision space more effectively, achieving global solutions in finite time. CMA-ES also fails in *MP1* because the narrow passage between obstacles challenges its single-modality evolutionary strategy, causing it to be trapped in local optima. These experiments highlight both the computational efficiency of TTTS—enabled by the TT approximation of the decision tree—and its global solution-finding capability, enabled by the compact representation of TT and the strategic search adopted from MCTS.

In addition, we compared TTTS with two widely used specific approaches for obstacle-avoidance motion planning: VP-STO (Jankowski et al. 2023) and the Probabilistic Roadmap combined with trajectory optimization (PRM+TO) (Gasparetto et al. 2015). To evaluate performance, we randomly generated five targets and applied the approaches to compute the joint trajectories required to reach them. The comparison results, including *reaching error*, *total cost*, and *computation time*, are reported in Figure 7. We also visualize the manipulator trajectories for one of the targets, where TTTS and PRM+TO both find a solution, while VP-STO struggles with the narrow passage. From the table, we observe that TTTS achieves performance comparable to PRM+TO while requiring less computation time, thereby demonstrating its computational efficiency. In contrast, VP-STO results in higher error due to its reliance on a good initial guess and its inability to handle multi-modal solution spaces.

6.4 Legged Robot Manipulation

In addition to motion planning around obstacles, we also evaluated our approach on a contact-rich manipulation task using a legged robot (Zhu et al. 2023). As illustrated in Figure 1(C), a solo robot manipulates a cube of size 10×10 cm under a joint impedance controller in the Genesis simulator (Zhou et al. 2024). The cube, weighing 0.05kg, is required to pivot about the y-axis by a certain angle. This task is challenging because the robot must coordinate contact interactions with the object while maintaining stability, handle the hybrid dynamics arising from intermittent contacts, and plan a smooth motion under impedance control. Small errors in trajectory generation can cause the cube to slip, fail to pivot, or destabilize the supporting leg. To solve this task, we define the cost function as a weighted sum of the terminal pose error and the average cube velocity, with details provided in the appendix. During planning, we optimize the Cartesian-space position trajectory for each leg tip. Both legs are then controlled via impedance control with default parameters of $K_p = 100$ N/m and $K_v = 10$ N·s/m. To enable stable pivoting against gravity, we increase the stiffness of the left leg to $K_p = 2000$ N/m along the y- and z-axes.

Figure 8 reports the comparison of TTTS with other methods on this task. TTTS and MCTS achieve similar performance in terms of reaching error and total cost, highlighting TTTS's global convergence capability comparable to MCTS, while TTTS requires significantly less computation time owing to the TT representation of the decision tree. In contrast, TTGO and CMA-ES perform worse. This further



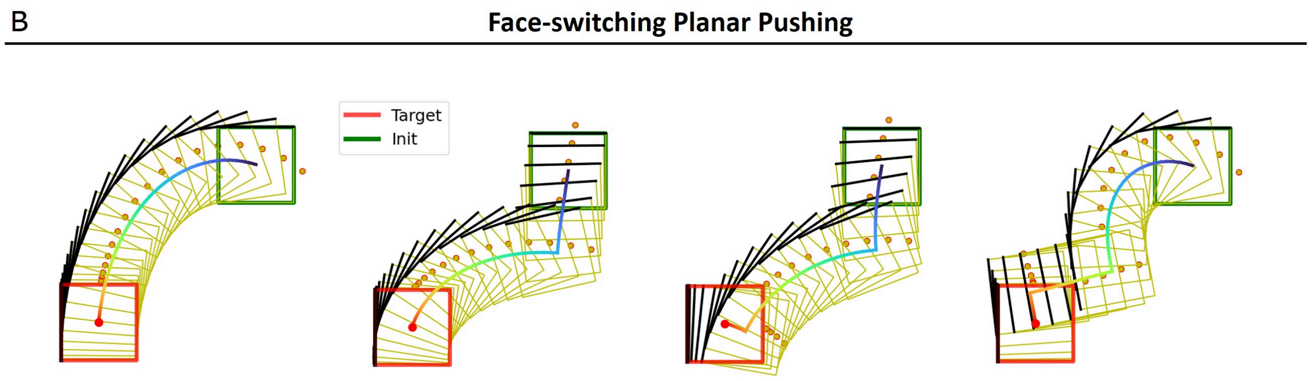


Figure 6. Multi-modal solutions for motion planning around obstacles and face-switching planar pushing. (A) 3-joint manipulator motion planning with multi-modal solutions. The trajectory is visualized using a red-to-green color spectrum to indicate temporal evolution. (B) Face-switching planar pushing task with multi-modal solutions.Given the same initial configuration $[0.25, 0.25, -\frac{\pi}{2}]$ and target $[0.25, 0.25, -\frac{\pi}{2}]$, our algorithm can find diverse solutions to accomplish the task, by jointly optimizing over discrete contact faces and continuous motion variables. The black edge of the rectangle indicates the cube's orientation. The number of face switches varies from 0 to 2.

underscores the importance of TT factorization and strategic search for effectively solving this task.

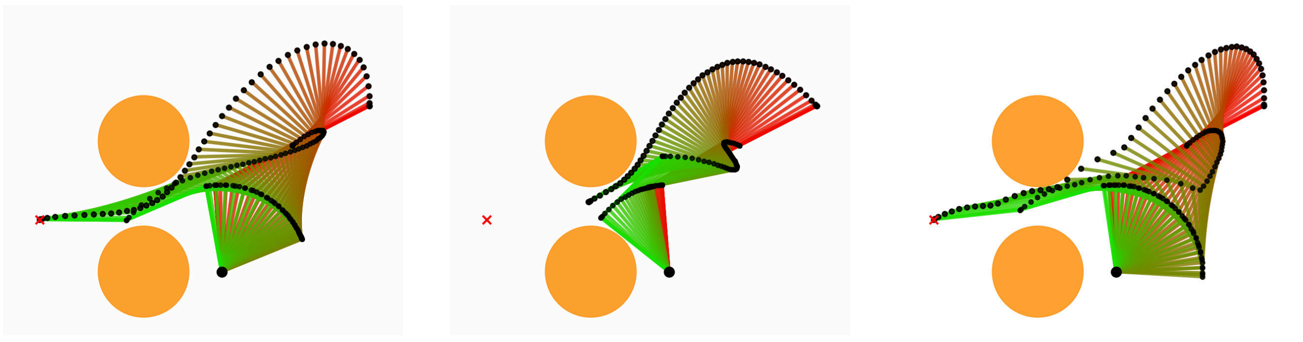
6.5 Multi-stage motion planning

Multi-stage motion planning encompasses a broad class of realistic robotic problems, such as multi-stage forceful manipulation (Holladay et al. 2024) and multi-primitive sequencing (Xue et al. 2024c). The objective is to generate a trajectory that enables a robot to interact intelligently with its environment, typically involving contact mode switches (e.g., sticking, sliding, or transitioning between different manipulation primitives). In this article, the trajectory is parameterized by a sequence of basis weights $\boldsymbol{w}_k$, such that the continuous state at each stage is given by $\boldsymbol{u}_k = \boldsymbol{\Psi}_k \boldsymbol{w}_k$. Discrete modes m_k represent different contact or task-specific phases. The optimization aims to minimize a total cost consisting of smoothness penalties, control effort, and task-specific objectives, while satisfying constraints such as collision avoidance, dynamics, and mode transitions.

We use face-switching planar pushing (Doshi et al. 2020) as a representative task, in which a robot must push a cube from an initial configuration to a target configuration, while accounting for underactuated dynamics and face-switching mechanisms. The robot must determine the end-effector velocity (continuous) and which face to establish contact (discrete). This task is particularly challenging due to the nonlinear dynamics and the hybrid nature of the decision variables, which present difficulties for both gradient-based and sampling-based optimization methods. The goal of this experiment is to demonstrate the ability of our approach to efficiently handle such hybrid decision-making problems, coupled with nonlinear dynamics. Figure 6 (B) illustrates the found cube trajectories given the same initial and target configuration, showcasing the multi-modal nature of the solutions. The number of face switches varies from 0 to 2, along with diverse, smooth continuous trajectories.

Figure 5 (C) presents the ablation study for this task. The TT approximation of the decision tree is obtained with a rank setting of $r_{\text{max}} = 41$. The results show that the TT approximation alone is not sufficiently accurate to capture the full landscape of the objective function. However, the subsequent tree search significantly improves the solution quality, and the final refinement step enables convergence to the global optimum. An interesting observation is that, compared with the policy learning formulation (i.e., infinite-horizon dynamic programming) proposed in (Xue et al. 2024b, 2025), this finite-horizon planning formulation requires a significantly higher TT rank to accurately represent the objective function. This is because trajectory-level planning uses decision variables (i.e., basis weights) that influence the entire trajectory, where small changes can induce large, structured variations, necessitating higher representational capacity to capture complex temporal dependencies. This further motivates the necessity of combining TT approximation with tree search, rather than relying on TT alone.

Figure 8 compares TTTS with other approaches. CMA-ES struggles with this problem (indicated by diagonal



(a) TTTS **(b)** VP-STO **(c)** PRM+TO

TTTS			VP-STO			PRM+TO		
Error	Total Cost	Comp. Time (s)	Error	Total Cost	Comp. Time (s)	Error	Total Cost	Comp. Time (s)
0.00 ± 0.00	0.56 ± 0.74	1.00 ± 0.03	0.07 ± 0.13	3.67 ± 6.85	2.52 ± 2.43	0.00 ± 0.00	0.41 ± 0.51	1.87 ± 0.34

Figure 7. Comparison of TTTS, VP-STO, and PRM+TO. The red cross indicates the reaching target. TTTS and PRM+TO successfully generate optimal manipulator trajectories, while VP-STO fails in the narrow passage. The table reports reaching error, control cost, and computation time, highlighting TTTS's superior efficiency.

hatching in the bar chart) because it involves both discrete and continuous decision variables. Both TTTS and MCTS successfully reach the final target, but TTTS requires less computation time owing to the TT factorization of the decision tree. TTGO is highly efficient in terms of runtime, but its solutions are less accurate due to the absence of the strategic search incorporated in TTTS. Overall, these experiments highlight both the computational efficiency and the global solution-finding capability of TTTS.

6.6 *Model Predictive Control for bimanual whole-body manipulation*

We evaluate our approach using a Model Predictive Control (MPC) formulation applied to a bimanual whole-body manipulation task. This task is particularly challenging due to complex contact dynamics between objects, the whole-body geometry of the robot, and interactions with the environment, all of which make accurate modeling difficult. Physical simulators such as MuJoCo (Todorov et al. 2012) and IsaacGym (Liang et al. 2018) can help address these challenges. However, the resulting sim-to-real gap necessitates the use of real-time MPC. Given the simulator as a black-box forward dynamics model, sampling-based MPC becomes a promising approach, as it does not rely on explicit gradient information. Nevertheless, such methods often suffer from high sample complexity and slow convergence, limiting their practicality for real-world deployment. In this experiment, we aim to demonstrate that TTTS can quickly find high-quality solutions to support real-time, sampling-based MPC. Specifically, we use Genesis (Zhou et al. 2024) as the simulator due to its parallel simulation capabilities, and the number of environments is set to 500.

Figure 9 illustrates the performance differences between TTTS, TTGO, MCTS, and CMA-ES. The computation time is limited to 1 second. A task is considered successful if the angular error at the final timestep satisfies $|\theta - \theta^*| < 3°$, where θ is the object's final z-axis orientation and θ^* is the desired target angle. We evaluate five randomly generated configurations and report success rate, final state error, and total trajectory cost. TTGO achieves a decent success rate with a low TT rank ($r_{\max} = 10$), but its limited accuracy leads to occasional failures. MCTS offers theoretical global convergence but requires more time, making it impractical for real-time MPC. CMA-ES also performs poorly due to slow convergence. In contrast, TTTS first approximates the decision tree in TT format, which accelerates convergence toward promising regions. It then performs a TT-based tree search, enabling efficient exploitation and strategic exploration. The results, including final error and total cost, confirm the effectiveness of TTTS in supporting real-time MPC for contact-rich manipulation.

6.7 *Real-world experiments*

We validated our approach in the real world through the whole-body bimanual manipulation task. Two 7-DoF Franka robots and a RealSense D435 camera were used. The manipulated object was a large box with the size of 36cm $\times$ 26cm $\times$ 34cm, which is representative of objects commonly found in warehouse applications. The task was to rotate the box to a target orientation and then lift it. It involved complex contact interactions among the robots, the object, and the table, as well as the full-body surface geometry of the two robot arms, making the system particularly difficult to model.

To address this, we used Genesis to predict the box trajectory given a sequence of control commands, eliminating the need to manually model the complex system dynamics. We first applied TT-Cross to obtain a low-rank TT approximation, augmented with the object pose. During execution, we conditioned on the current object pose and performed tree search for 3 iterations, which typically produced effective results thanks to the guidance provided by the TT approximation.

At each time step (every 1 second), we generated a 9-second trajectory in a model predictive control (MPC) manner to bridge the sim-to-real gap. Figure 10 presents keyframes of one task of rotating the box for 90°. We can observe that the robots geometry is actively utilized to establish contacts with the object, enabling whole-body

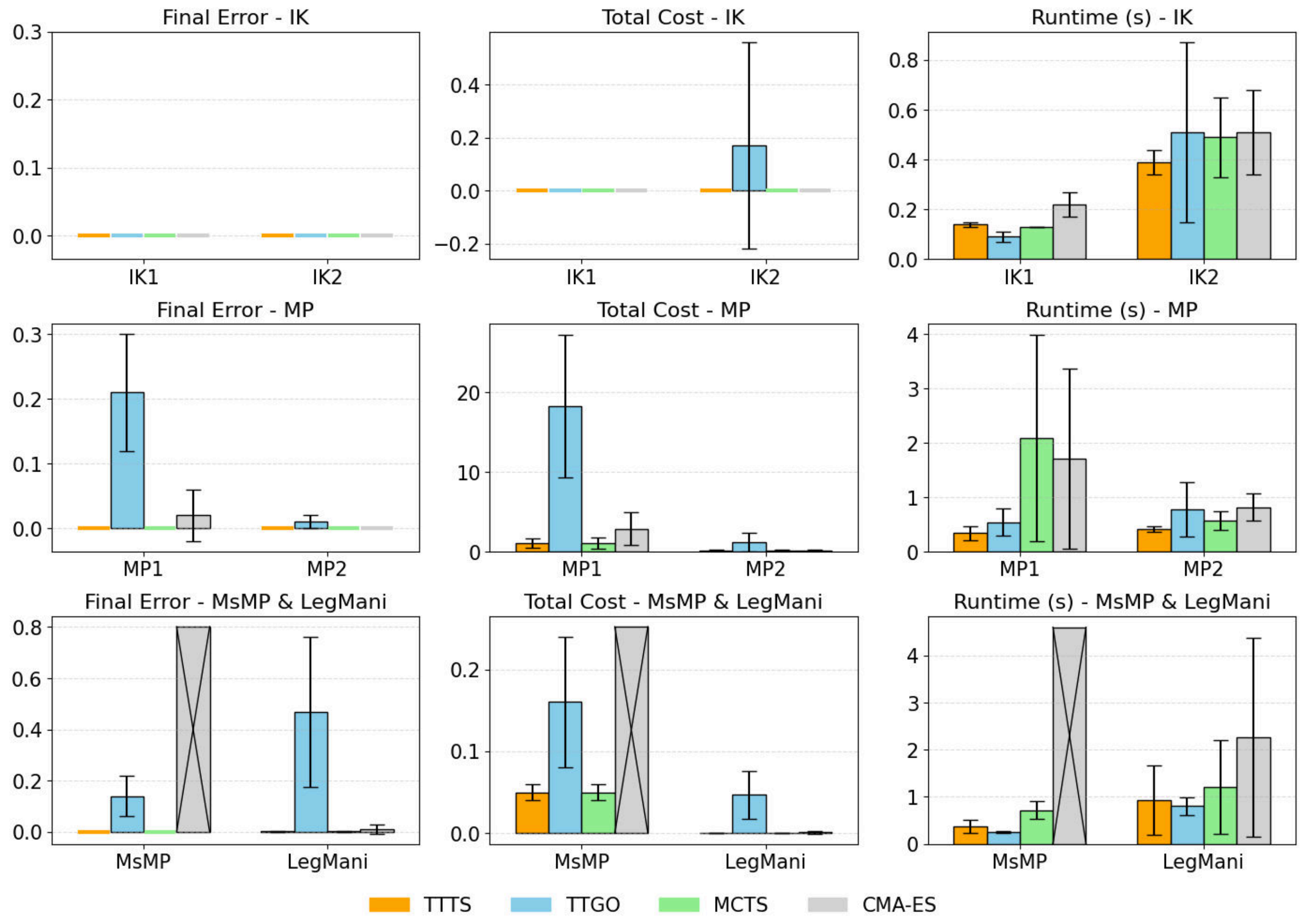


Figure 8. Comparison of TTTS, TTGO, CMA-ES, and MCTS with respect to final reaching error, total control cost, and runtime. *IK1* and *IK2* represent the 3-joint and 7-joint inverse kinematics tasks, respectively. *MP1* and *MP2* correspond to the 3-joint and 7-joint manipulator reaching tasks with obstacle avoidance. *MsMP* denotes the planar pushing task with a face-switching mechanism, and *LegMani* refers to the legged robot manipulation (cube pivoting) task. Diagonal patterns in the bar charts indicate that CMA-ES is incompatible with *MsMP* task.

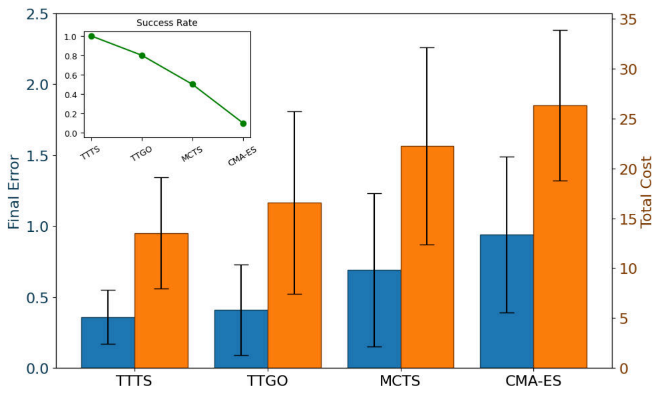


Figure 9. Comparison for bimanual whole-body manipulation. The blue bars represent the final state error achieved by different methods, while the orange bars correspond to the total cost.

bimanual manipulation. Furthermore, we compared TTTS and CMA-ES on this task by running five trials each. The results are shown in Figure 11a and Figure 11b. With a tolerance of $10°$, TTTS achieves a 100% success rate, whereas CMA-ES achieves only 40% with similar computation time. This demonstrates that TTTS offers better sampling efficiency, owing to the tensor factorization.

To evaluate the robustness of the proposed method, we further tested it under several variations, including changes in the initial position, adding weight to the box (1.2 kg), combining both variations, and altering the initial orientation. The corresponding box trajectories are presented in Figure 11c. We observe that changes in the initial position and orientation have little effect on TTTS performance, thanks to the feedback mechanism embedded in MPC. However, adding weight results in slightly worse performance because the sim-to-real gap arises when the Genesis simulator is used as the model for generating receding-horizon plans in MPC, and this simulator does not account for the additional weight. Nevertheless, even with this gap, the box can still be reoriented by more than $70°$, which demonstrates the robustness of our approach under model uncertainty. More tasks and comparisons are presented in the accompanying video.

7 Conclusion and Future Work

In this work, we present an approach called *Tensor Train Tree Search* (TTTS), a method that leverages tensor factorization to exploit the redundancy in decision trees

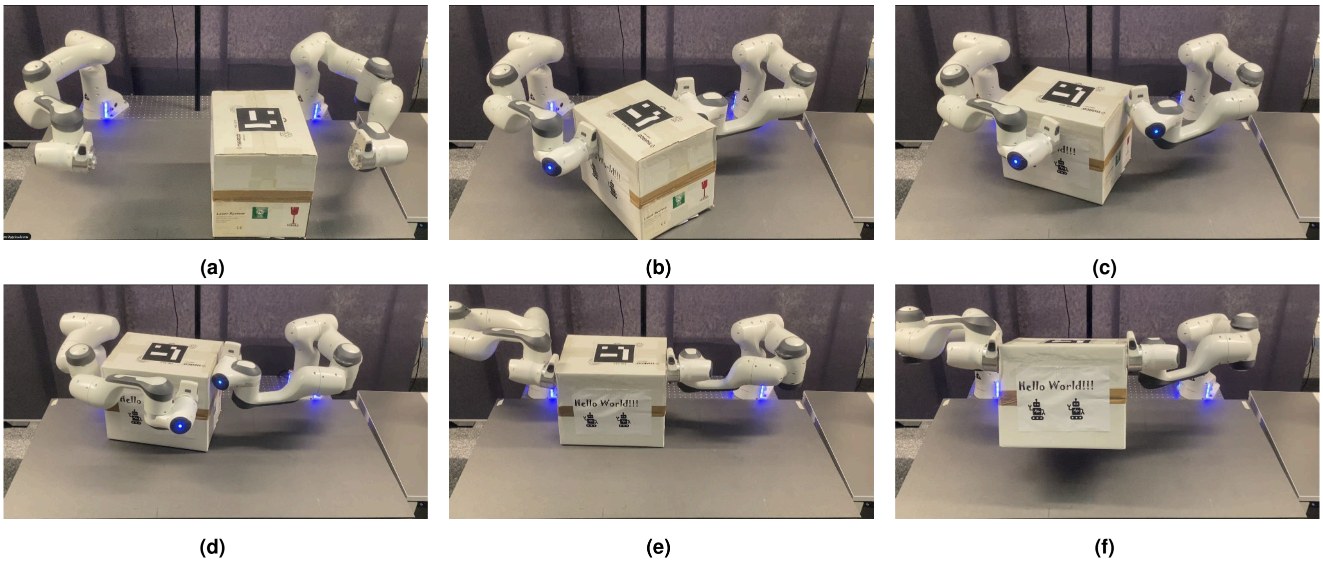
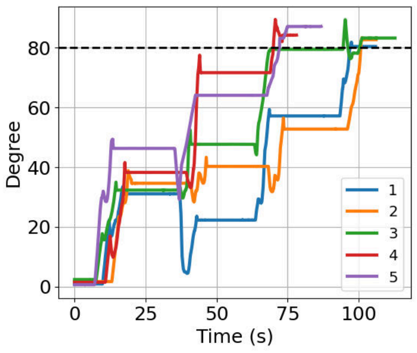
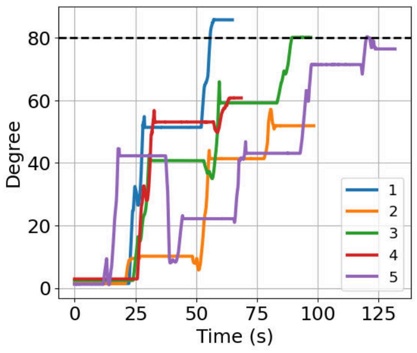


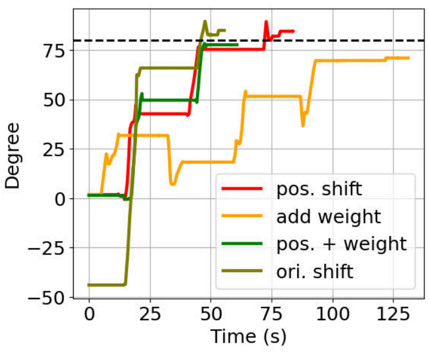
Figure 10. Keyframes of bimanual whole-body manipulation. The system is initialized as (a), and the objective is to rotate the box 90° and then lift it as (f). We can observe that the robots can actively exploit whole-body geometry to make and break contacts with the object, completing the overall task successfully.



(a) TTTS Results



(b) CMA-ES Results



(c) TTTS Robustness Evaluation

Figure 11. Statistical analysis of real-world bimanual whole-body manipulation. The objective is to rotate the box by 90°. A trial is considered successful if the box is rotated beyond 80° (indicated by the dashed line in the plots). (a) and (b) present the box orientation trajectories produced by TTTS and CMA-ES, respectively, showing that TTTS achieves a significantly higher success rate. To further assess TTTS robustness, additional experiments were conducted by varying initial position, adding weight on the box, combining both variations, and altering the initial orientation, as illustrated in (c).

for robot optimization. The key idea is to represent a multi-layer decision tree as a high-dimensional tensor and apply tensor factorization—specifically the TT-Cross approximation—to obtain its Tensor Train (TT) format, which then enables efficient Monte Carlo Tree Search. We demonstrate the effectiveness of the proposed approach across diverse domains in simulation, including inverse kinematics, motion planning around obstacles, planar pushing with face switching, and legged robot manipulation. Furthermore, a real-world experiment on bimanual whole-body manipulation highlights the practical efficiency of TTTS in enabling fast sampling-based model predictive control.

While these results establish the promise of TTTS, several open challenges remain. In particular, TT-Cross provides an efficient global approximation by actively querying function values, but it faces scalability issues in very high-dimensional settings (e.g., with visual observations). To address this limitation, we plan to integrate TT with neural networks in a data-driven manner (Dolgov et al. 2023; Steinlechner 2016; Novikov et al. 2021), where the key idea is to learn compact TT cores such that the reconstructed tensor accurately approximates the observed elements.

Beyond TT, we can also explore improvements to the tree representation itself. In this work we employ the standard TT format to approximate the decision tree and exploit its separable structure, which improves both search efficiency and memory usage. For future work, we aim to investigate Quantized Tensor Train (QTT) decomposition (Dolgov et al. 2012), which provides a multi-resolution embedding by reshaping each dimension into a sequence of binary (or small) modes. This hierarchical representation aligns naturally with the layered structure of decision trees, enabling coarse-to-fine reasoning and facilitating more efficient branch-and-bound procedures that can significantly reduce computation time for global optimization.

In our current implementation, Genesis is used as the simulation model for planning and control. While it offers general-purpose flexibility and reliable physics, its rollout speed remains a computational bottleneck, limiting scalability for tasks that require extensive sampling or rapid execution. A promising direction for future work is to integrate learned dynamics models as lightweight surrogates, enabling significantly faster rollouts. Such models could greatly improve computational efficiency and expand the applicability of our approach.

Finally, our current representation of discrete modes is limited to integer encoding, which could be extended to symbolic forms such as first-order logic. Since symbolic transitions often introduce geometric constraints, decision-making requires integrated logic-geometric programming (Toussaint 2015). TTTS naturally supports the joint modeling of logic and geometric variables, offering improved computational efficiency over traditional hierarchical frameworks.

Acknowledgements

This work was supported by the State Secretariat for Education, Research and Innovation in Switzerland for participation in the European Commission's Horizon Europe Program through the INTELLIMAN project (`https://intelliman-project.eu/`, HORIZON-CL4-Digital-Emerging Grant 101070136) and the SESTOSENSO project (`http://sestosenso.eu/`, HORIZON-CL4-Digital-Emerging Grant 101070310), and by the China Scholarship Council (grant No.202106230104).

References

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. *Advances in neural information processing systems*, 30, 2017.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Guillaume MJ B Chaslot, Mark HM Winands, and H Jaap van Den Herik. Parallel monte-carlo tree search. In *Computers and Games: 6th International Conference, CG 2008, Beijing, China, September 29-October 1, 2008. Proceedings 6*, pages 60–71. Springer, 2008.

Xianyi Cheng, Sarvesh Patil, Zeynep Temel, Oliver Kroemer, and Matthew T Mason. Enhancing dexterity in robotic manipulation via hierarchical contact exploration. *IEEE Robotics and Automation Letters (RA-L)*, 9(1):390–397, 2023.

Howie Choset, Kevin M Lynch, Seth Hutchinson, George A Kantor, and Wolfram Burgard. *Principles of robot motion: theory, algorithms, and implementations*. MIT press, 2005.

Andrzej Cichocki, Namgil Lee, Ivan Oseledets, Anh-Huy Phan, Qibin Zhao, and Danilo P. Mandic. Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *Foundations and Trends® in Machine Learning*, 9(4-5):249–429, 2016.

Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

Robin Deits and Russ Tedrake. Computing large convex regions of obstacle-free space through semidefinite programming. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pages 109–124. Springer, 2015.

Sergey Dolgov, Dante Kalise, and Luca Saluzzi. Data-driven tensor train gradient cross approximation for hamilton–jacobi–bellman equations. *SIAM Journal on Scientific Computing*, 45 (5):A2153–A2184, 2023.

Sergey V Dolgov, Boris N Khoromskij, and Ivan V Oseledets. Fast solution of parabolic problems in the tensor train/quantized tensor train format with initial application to the fokker–planck equation. *SIAM Journal on Scientific Computing*, 34 (6):A3016–A3038, 2012.

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.

Jens Eisert, Marcus Cramer, and Martin B Plenio. Colloquium: Area laws for the entanglement entropy. *Reviews of modern physics*, 82(1):277–306, 2010.

Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 3067–3074, 2015.

Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4(1):265–293, 2021.

Alessandro Gasparetto, Paolo Boscariol, Albano Lanzutti, and Renato Vidoni. Path planning and trajectory planning algorithms: A general overview. *Motion and operation planning of robotic systems: Background and practical approaches*, pages 3–27, 2015.

Andrew Goldenberg, Beno Benhabib, and Robert Fenton. A complete generalized solution to the inverse kinematics of robots. *IEEE Journal on Robotics and Automation*, 1(1):14–20, 2003.

Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18, 2003.

Richard A Harshman. Foundations of the parafac procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA working papers in phonetics*, 16(1):84, 1970.

Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

Rachel Holladay, Tomás Lozano-Pérez, and Alberto Rodriguez. Robust planning for multi-stage forceful manipulation. *International Journal of Robotics Research (IJRR)*, 43(3):330–353, 2024.

Julius Jankowski, Lara Brudermüller, Nick Hawes, and Sylvain Calinon. Vp-sto: Via-point-based stochastic trajectory optimization for reactive robot behavior. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 10125–10131, 2023.

Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research (IJRR)*, 30(7):846–894, 2011.

Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

Beomjoon Kim, Kyungjae Lee, Sungbin Lim, Leslie Kaelbling, and Tomás Lozano-Pérez. Monte carlo tree search in continuous spaces using voronoi optimistic optimization with regret bounds. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9916–9924, 2020.

Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

Teguh Santoso Lembono, Antonio Paolillo, Emmanuel Pignat, and Sylvain Calinon. Memory of motion for warm-starting trajectory optimization. *IEEE Robotics and Automation Letters (RA-L)*, 5(2):2594–2601, 2020.

Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *International Conference on Informatics in Control, Automation and Robotics*, volume 2, pages 222–229. SciTePress, 2004.

Jacky Liang, Viktor Makoviychuk, Ankur Handa, Nuttapong Chentanez, Miles Macklin, and Dieter Fox. Gpu-accelerated robotic simulation for distributed reinforcement learning. In *Conference on Robot Learning*, pages 270–282. PMLR, 2018.

Leo Liberti. Undecidability and hardness in mixed-integer nonlinear programming. *RAIRO-Operations Research*, 53(1): 81–109, 2019.

Yucong Lin and Srikanth Saripalli. Sampling-based path planning for uav collision avoidance. *IEEE Transactions on Intelligent Transportation Systems*, 18(11):3179–3192, 2017.

Danylo Malyuta, Taylor P Reynolds, Michael Szmuk, Thomas Lew, Riccardo Bonalli, Marco Pavone, and Behçet Açıkmeşe. Convex optimization for trajectory generation: A tutorial on generating dynamically feasible trajectories reliably and efficiently. *IEEE Control Systems Magazine*, 42(5):40–113, 2022.

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

Tobia Marcucci, Jack Umenberger, Pablo Parrilo, and Russ Tedrake. Shortest paths in graphs of convex sets. *SIAM Journal on Optimization*, 34(1):507–532, 2024.

Matthew T Mason. Mechanics and planning of manipulator pushing operations. *International Journal of Robotics Research (IJRR)*, 5(3):53–71, 1986.

Matthew T Mason. Progress in nonprehensile manipulation. *International Journal of Robotics Research (IJRR)*, 18(11): 1129–1141, 1999.

David Mayne. A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems. *International Journal of Control*, 3(1):85–95, 1966.

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

Mustafa Mukadam, Jing Dong, Xinyan Yan, Frank Dellaert, and Byron Boots. Continuous-time gaussian process motion planning via probabilistic inference. *International Journal of Robotics Research (IJRR)*, 37(11):1319–1340, 2018.

Ramkumar Natarajan, Howie Choset, and Maxim Likhachev. Interleaving graph search and trajectory optimization for aggressive quadrotor flight. *IEEE Robotics and Automation Letters*, 6(3):5357–5364, 2021.

Georgii S Novikov, Maxim E Panov, and Ivan V Oseledets. Tensor-train density estimation. In *Uncertainty in artificial intelligence*, pages 1321–1331. PMLR, 2021.

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

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

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.

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.

Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 489–494, 2009.

Benjamin Rivière, John Lathrop, and Soon-Jo Chung. Monte carlo tree search with spectral expansion for planning with dynamical systems. *Science Robotics*, 9(97):eado1010, 2024.

Nicholas Roy, Ingmar Posner, Tim Barfoot, Philippe Beaudoin, Yoshua Bengio, Jeannette Bohg, Oliver Brock, Isabelle Depatie, Dieter Fox, Dan Koditschek, et al. From machine learning to robotics: Challenges and opportunities for embodied intelligence. *arXiv preprint arXiv:2110.15245*, 2021.

Oren Salzman and Dan Halperin. Asymptotically near-optimal rrt for fast, high-quality motion planning. *IEEE Transactions on Robotics*, 32(3):473–483, 2016.

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.

John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *International Journal of Robotics Research (IJRR)*, 33(9):1251–1270, 2014.

Suhan Shetty, Teguh Lembono, Tobias Löw, and Sylvain Calinon. Tensor train for global optimization problems in robotics. *International Journal of Robotics Research (IJRR)*, 43(6):811–839, 2024. doi: 10.1177/02783649231217527.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Konstantin Sozykin, Andrei Chertkov, Roman Schutski, Anh-Huy Phan, Andrzej S Cichocki, and Ivan Oseledets. Ttopt: A maximum volume quantized tensor train-based optimization and its application to reinforcement learning. *Advances in neural information processing systems*, 35:26052–26065, 2022.

Michael Steinlechner. Riemannian optimization for high-dimensional tensor completion. *SIAM Journal on Scientific Computing*, 38(5):S461–S484, 2016.

Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE, 2012.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

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.

Ledyard R Tucker. Implications of factor analysis of three-way matrices for measurement of change. *Problems in measuring change*, 15(122-137):3, 1963.

Zhenbo Wang and Michael J Grant. Constrained trajectory optimization for planetary entry via sequential convex programming. *Journal of Guidance, Control, and Dynamics*, 40(10):2603–2615, 2017.

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)*, pages 4999–5005, 2023.

Teng Xue, Amirreza Razmjoo, and Sylvain Calinon. D-LGP: Dynamic logic-geometric program for reactive task and motion planning. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 14888–14894, 2024a.

Teng Xue, Amirreza Razmjoo, Suhan Shetty, and Sylvain Calinon. Robust manipulation primitive learning via domain contraction. In *Proc. Conference on Robot Learning (CoRL)*, 2024b.

Teng Xue, Amirreza Razmjoo, Suhan Shetty, and Sylvain Calinon. Logic-Skill Programming: An Optimization-based Approach to Sequential Skill Planning. In *Proc. Robotics: Science and Systems (RSS)*, 2024c.

Teng Xue, Amirreza Razmjoo, Suhan Shetty, and Sylvain Calinon. Robust contact-rich manipulation through implicit motor adaptation. *International Journal of Robotics Research (IJRR)*, 2025.

Xian Zhou, Yiling Qiao, Zhenjia Xu, Tsun-Hsuan Wang, Zhehuan Chen, Juntian Zheng, Ziyan Xiong, Yian Wang, Mingrui Zhang, Pingchuan Ma, Yufei Wang, Zhiyang Dou, Byungchul Kim, Yunsheng Tian, Yipu Chen, Xiaowen Qiu, Chunru Lin, Tairan He, Zilin Si, Yunchu Zhang, Zhanlue Yang, Tiantian Liu, Tianyu Li, Kashu Yamazaki, Hongxin Zhang, Huy Ha, Yu Zhang, Michael Liu, Shaokun Zheng, Zipeng Fu, Qi Wu, Yiran Geng, Feng Chen, Yuanming Hu, Guanya Shi, Lingjie Liu, Taku Komura, Zackory Erickson, David Held, Minchen Li, Linxi Fan (Jim), Yuke Zhu, Wojciech Matusik, Dan Gutfreund, Shuran Song, Daniela Rus, Ming Lin, Bo Zhu, Katerina Fragkiadaki, and Chuang Gan. Genesis: A generative and universal physics engine for robotics and beyond, December 2024. URL https://github.com/Genesis-Embodied-AI/Genesis.

Huaijiang Zhu, Avadesh Meduri, and Ludovic Righetti. Efficient object manipulation planning with monte carlo tree search. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pages 10628–10635, 2023.

A Appendix

A.1 Experimental hyperparameter

In our experiments, we utilized an NVIDIA GeForce RTX 3090 GPU with 24 GB of memory. The tolerance for the TT-Cross approximation was set to $\epsilon = 10^{-3}$. Table 1 summarizes the hyperparameters applied across different tasks. *Batch size* indicates the maximum number of parallel environments; *Num. MCTS* denotes the number of independent MCTS instances executed concurrently; and *Num. Sim.* specifies the number of parallel environments used during the simulation phase. *Num. Discret.* refers to the discretization granularity of the state and action spaces. The parameter $r_{\max}$ defines the maximum TT rank employed during TT-Tree initialization via TT-Cross. *Pop. Size* and *CMA-ES Iter.* are the population size and iteration count used in the CMA-ES refinement step, respectively.

A.2 Cost Function Used in Experiments

Continuous non-convex function. To illustrate the capability of TTTS in handling non-convex optimization problems and its advantage over TTGO, we construct the following full-rank two-dimensional function as a toy example. The performance of TTTS on this function is reported in Section 6.1.1.

$$\begin{aligned} f_1(\boldsymbol{x}) &= -\tfrac{1}{2}\left(0.8z - 2\,\mathrm{sign}(z)\right)^2 + 0.1\|\boldsymbol{x}\|_2^2 + 2, \\ \boldsymbol{x} &= (x_1, x_2)^\top, \quad z = \tfrac{x_1 + x_2}{\sqrt{2}} \end{aligned} \tag{15}$$

Mixed-integer non-convex function. MsMP problems can be generally formulated as mixed-integer non-convex programs. To demonstrate the effectiveness of TTTS in addressing such problems, we consider the following piecewise-defined mixed-integer non-convex function:

$$\tilde{\boldsymbol{x}} = \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \boldsymbol{x},$$

$$f_2(\boldsymbol{x}) = \tfrac{1}{2}\, g(\tilde{x}_1) + 0.1\, h(\tilde{x}_1, \tilde{x}_2), \tag{16}$$

where the piecewise components are defined as

$$g(\tilde{x}_1) = \begin{cases} |-(\tilde{x}_1 + 5)^2 + 8|, & -7 \le \tilde{x}_1 < -3, \\ |-(\tilde{x}_1 + 1)^2 + 3|, & -3 \le \tilde{x}_1 < 1, \\ |-(\tilde{x}_1 - 3)^2 + 4|, & 1 \le \tilde{x}_1 \le 5, \\ |-(\tilde{x}_1 - 7)^2 + 5|, & 5 < \tilde{x}_1 \le 9, \\ |-(\tilde{x}_1 - 11)^2 + 10|, & 9 < \tilde{x}_1 \le 13, \\ 0, & \text{otherwise}, \end{cases}$$

Table 1. Hyperparameters employed for different tasks.

Task	Batch Size	Num. MCTS	Num. Discret.	Num. Sim.	r_{max}	Pop. Size	CMA-ES Iter.
3-joint IK	1000	5	20	1000	21	25	20
7-joint IK	500	5	20	1000	21	25	20
3-joint MP	1000	5	20	1000	41	25	20
7-joint MP	500	5	20	1000	21	25	20
LegMani	500	5	20	1000	21	25	20
Multi-stage MP	1000	5	50	1000	41	25	20
Whole-body Manipulation	500	1	20	500	21	25	20

$$h(\tilde{x}_1, \tilde{x}_2) = \begin{cases} (\tilde{x}_1+5)^2 + (\tilde{x}_2+5)^2, & -7 \le \tilde{x}_1 < -3, \\ (\tilde{x}_1+1)^2 + (\tilde{x}_2+1)^2, & -3 \le \tilde{x}_1 < 1, \\ (\tilde{x}_1-3)^2 + (\tilde{x}_2-3)^2, & 1 \le \tilde{x}_1 \le 5, \\ (\tilde{x}_1-7)^2 + (\tilde{x}_2-7)^2, & 5 < \tilde{x}_1 \le 9, \\ (\tilde{x}_1-11)^2 + (\tilde{x}_2-11)^2, & 9 < \tilde{x}_1 \le 13, \\ 0, & \text{otherwise.} \end{cases}$$

subject to the constraints

$$\boldsymbol{x} = (x_1, x_2)^\top, \quad x_1 \in \{0, 1, \ldots, 10\}, \quad x_2 \in [-5, 5].$$

The performance of TTTS on this function is reported in Section 6.1.2.

Cost function for inverse kinematics. In the inverse kinematics (IK) setting, the goal is to find a joint configuration $\boldsymbol{q} \in \mathbb{R}^n$ such that the robot's end-effector reaches a desired target position while avoiding collisions and maintaining reasonable deviation from the current posture. The total cost function used is defined as:

$$c_{\text{total}} = 50\, c_{\text{goal}} + c_{\text{obst}}.$$

The term c_{goal} penalizes the distance between the forward kinematics output of the proposed joint configuration and the desired end-effector position:

$$c_{\text{goal}} = \|\text{eef}_{\text{pos}}(\boldsymbol{q}) - \boldsymbol{x}_{\text{goal}}\|_2 .$$

Here, $\boldsymbol{x}_{\text{goal}}$ is the target pose of the end-effector, and $\text{eef}_{\text{pos}}(\boldsymbol{q})$ denotes the pose of the end-effector computed using forward kinematics at joint configuration $\boldsymbol{q}$.

The term c_{obst} penalizes collisions by summing binary collision indicators for the final robot configuration:

$$c_{\text{obst}} = \sum_{i=1}^{N_{\text{links}}} \text{collides}_i(\boldsymbol{q}),$$

where $\text{collides}_i(\boldsymbol{q}) \in \{0, 1\}$ indicates whether the i-th link is in collision.

Cost function for motion planning. The total cost function used for trajectory optimization is composed of three terms: a goal-reaching cost, a collision avoidance cost, and a control smoothness cost. The overall cost is defined as:

$$c_{\text{total}} = 50\, c_{\text{goal}} + c_{\text{obst}} + 0.1\, c_{\text{control}}.$$

The first term, c_{goal}, encourages the robot to reach the desired target position. It is computed as the Euclidean distance between the current end-effector position and the goal:

$$c_{\text{goal}} = \|\text{eef}_{\text{pos}} - \boldsymbol{x}_{\text{goal}}\|_2 ,$$

where eef_{pos} denotes the Cartesian position of the robot's end-effector at the final time step, and $\boldsymbol{x}_{\text{goal}}$ is the desired target position.

The second term, c_{obst}, penalizes trajectories that result in collisions. It is computed by summing binary collision indicators over the trajectory:

$$c_{\text{obst}} = \sum_{t=1}^{T} \text{collisions}_t,$$

where $\text{collisions}_t \in \{0, 1\}$ is a binary variable indicating whether a collision occurs at time step t.

The third term, c_{control}, measures the smoothness of the joint trajectory. It compares the total length of the path in joint space with the direct distance between the start and end configurations:

$$c_{\text{control}} = \frac{\sum_{t=1}^{T} \|\boldsymbol{q}_t - \boldsymbol{q}_{t-1}\|_2}{\|\boldsymbol{q}_T - \boldsymbol{q}_0\|_2 + \varepsilon}.$$

Here, $\boldsymbol{q}_t$ represents the robot's joint configuration at time step t, and ε is a small positive constant added for numerical stability. We set $\varepsilon = 10^{-6}$ in our experiments.

Cost function for legged robot manipulation. In the legged robot manipulation setting, the objective is to find optimal robot trajectories which move the manipulated box toward the target pose. The total cost is defined as

$$c_{\text{total}} = c_{\text{orn}} + c_{\text{vel}} + 5\, c_{\text{pos}},$$

where the three components are given below.

The first term penalizes deviation between the final box orientation and the desired orientation:

$$c_{\text{orn}} = 0.1 \left\| \mathbf{o}_T^{\text{box}} - \mathbf{o}_{\text{target}} \right\|_2,$$

where o_T^{box} is the yaw angle of the final box orientation.

The second term encourages stable manipulation by penalizing abrupt orientation changes of the box during the trajectory:

$$c_{\text{vel}} = \frac{0.1}{T-1} \sum_{t=0}^{T-2} \left| o_{t+1}^{\text{box}} - o_t^{\text{box}} \right|,$$

where o_t^{box} is the yaw angle of the box at time t.

The third term penalizes final position error of the box in the horizontal plane:

$$c_{\text{pos}} = 0.1 \left\| \mathbf{p}_T^{\text{box}} - \mathbf{p}_{\text{target}} \right\|_2,$$

where $\mathbf{p}_T^{\text{box}}$ denotes the final box position projected onto the xy-plane.

The cost function therefore balances final pose accuracy (position and orientation) with the smoothness of the manipulated box trajectory, promoting stable and efficient robot–box interaction.

Cost function for multi-stage motion planning. In the planar pushing setting, the objective is to compute a sequence of discrete contact modes and continuous robot velocities that move an object from its initial state to a target pose. The total cost combines two components: a state reaching cost and a control effort cost. It is defined as:

$$c_{\text{total}} = c_{\text{state}} + 0.01\, c_{\text{action}}.$$

The state cost c_{state} evaluates how far the final state $\boldsymbol{x}_T \in \mathbb{R}^3$ (position and orientation of the object) is from the target pose $\boldsymbol{x}_{\text{target}} \in \mathbb{R}^3$. It is defined as:

$$c_{\text{state}} = \left\|\boldsymbol{x}_T^{\text{pos}} - \boldsymbol{x}_{\text{target}}^{\text{pos}}\right\|_2 + 0.1\, |\theta_T - \theta_{\text{target}}|\,,$$

where:

- $\boldsymbol{x}_T^{\text{pos}} = [x_{1_T}, x_{2_T}]^\top$ is the object's final position
- θ_T is the final orientation
- $\boldsymbol{x}_{\text{target}}^{\text{pos}}$ and θ_{target} are the desired position and orientation

The action cost c_{action} penalizes the total effort of the pushing trajectory, where each control input $\mathbf{u}_t \in \mathbb{R}^2$ represents a planar velocity vector. The total control cost is the sum of the norms of all pushing actions:

$$c_{\text{action}} = \sum_{k=1}^{K}\sum_{t=1}^{T} \|\mathbf{u}_t\|_2\,.$$

These actions are generated based on a discrete contact face selection and continuous parameters defining a trajectory. The discrete mode $i_t \in \{1,2,3,4\}$ specifies which face of the object is being pushed at each time.

The final cost balances reaching the target pose accurately with minimizing the pushing effort, with a small weight on the latter to avoid excessive motion without overconstraining the optimization.

Cost function for whole-body manipulation. In the bimanual whole-body manipulation setting, the objective is to control two arms collaboratively to manipulate an object (e.g., a box) toward a target pose while maintaining effective contact and avoiding awkward configurations. The total cost function is composed of five terms:

$$c_{\text{total}} = c_{\text{pos}} + 50\, c_{\text{orn}} + 0.1\, c_{\text{control}} - c_{\text{contact}} + 5\, \delta_{\text{eef}},$$

where c_{pos} penalizes deviation of the box's final position from the target:

$$c_{\text{pos}} = \left\|\mathbf{p}_T^{\text{box}} - \mathbf{p}_{\text{target}}\right\|_2\,.$$

$\mathbf{p}_T^{\text{box}} \in \mathbb{R}^3$ is the final box position and $\mathbf{p}_{\text{target}}$ is the desired position. c_{orn} measures orientation alignment between the final box orientation and the target orientation, namely

$$c_{\text{orn}} = \left|\theta_T^{\text{box}} - \theta_{\text{target}}\right|\,.$$

c_{control} is a regularization term that penalizes excessive joint movement across the trajectory:

$$c_{\text{control}} = \sum_{t=0}^{T} \|\boldsymbol{q}_{t+1} - \boldsymbol{q}_t\|_2\,,$$

where $\boldsymbol{q}_t \in \mathbb{R}^n$ is the full-body joint configuration at time step t.

c_{contact} rewards the maintenance of valid bimanual contact. Let $c_t^L, c_t^R \in \{0,1\}$ be binary contact flags for the left and right hands. Then:

$$c_{\text{contact}} = \sum_{t=1}^{T} \mathbb{I}\left[c_t^L = 1 \wedge c_t^R = 1\right]\,,$$

where $\mathbb{I}[\cdot]$ is the indicator function. This term is subtracted from the total cost to encourage simultaneous dual-arm contact.

δ_{eef} is a binary term that penalizes configurations where the two arms are too close to each other, in order to avoid self-collision:

$$\delta_{\text{eef}} = \mathbb{I}\left[\left\|\mathbf{x}^{\text{eef}_0} - \mathbf{x}^{\text{eef}_1}\right\|_2 < 0.3\right]\,,$$

where $\mathbf{x}^{\text{eef}_i}$ is the Cartesian position of the i-th end-effector.

The cost function encourages accurate placement and orientation of the manipulated object, smooth and efficient joint trajectories, persistent bimanual contact, and physically feasible arm configurations.