Robotics: Science and Systems 2025
Los Angeles, CA, USA, June 21–25, 2025

Global Contact-Rich Planning with Sparsity-Rich Semidefinite Relaxations

Shucheng Kang*[1], Guorui Liu*[2], Heng Yang[1]
[1]School of Engineering and Applied Sciences, Harvard University
[2]School of Mathematical Sciences, University of Science and Technology of China
*Equal contribution

https://computationalrobotics.seas.harvard.edu/project-spot/

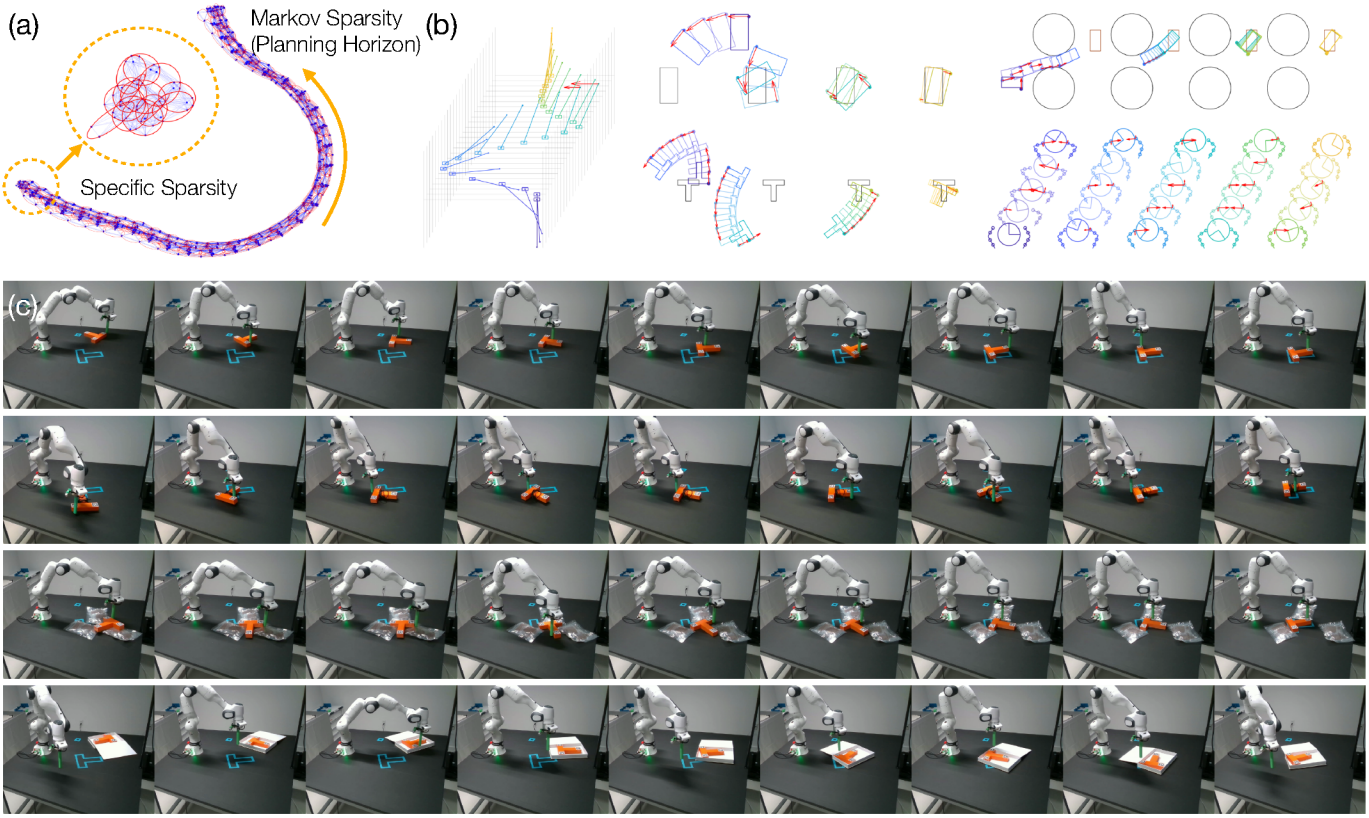


Fig. 1: Contact-rich planning is *sparsity-rich*. (a) Sparsity graph of planar hand manipulation showing two types of sparsity. (b) Sparsity enables high-order and tight, yet small-scale, semidefinite programming (SDP) relaxations solvable by off-the-shelf SDP solvers, computing certified near globally optimal trajectories for a suite of simulated problems. (c) Real-world validation on push-T. Planning powered by global optimization succeeds the task even under severe model mismatches and disturbances (from top to bottom: clean T; T tied up by a cable; T on top of a cluttered table; and T inside a "sliding" box).

***Abstract*—We show that contact-rich motion planning is also *sparsity-rich* when viewed as polynomial optimization (POP). We can exploit not only the *correlative* and *term* sparsity patterns that are general to all POPs, but also specialized sparsity patterns from the robot kinematic structure and the separability of contact modes. Such sparsity enables the design of high-order but sparse semidefinite programming (SDPs) relaxations—building upon Lasserre's moment and sums of squares hierarchy—that (*i*) can be solved in seconds by off-the-shelf SDP solvers, and (*ii*) compute near globally optimal solutions to the nonconvex contact-rich planning problems with small certified suboptimality. Through extensive experiments both in simulation (Push Bot, Push Box, Push Box with Obstacles, and Planar Hand) and real world (Push T), we demonstrate the power of using convex SDP relaxations to generate global contact-rich motion plans. As a contribution of independent interest, we release the Sparse Polynomial Optimization Toolbox (SPOT)—implemented in C++ with interfaces to both Python and Matlab—that automates sparsity exploitation for robotics and beyond.**

I. Introduction

Contact-rich planning plays a fundamental role in robotics tasks ranging from manipulation to locomotion [29, 9, 14]. At the heart of such planning problems lie two interrelated challenges: (a) *Contact mode selection*: determining when and where to establish or break contact is critical, yet the number of possible contact sequences grows exponentially with the number of contact modes and the planning horizon; (b) *Nonlinear dynamics and nonconvex geometric constraints*: the planned trajectory must satisfy the system's nonlinear dynamics and geometric constraints such as avoiding self and obstacle collisions. Together, these challenges exacerbate the problem's nonconvexity and computational complexity.

Problem statement. Let N represent the planning horizon with $[N] := \{1, 2, \ldots, N\}$. Define the state trajectory $\{x_k\}_{k=0}^{N} \subset \mathbb{R}^{n_x}$ and the control input trajectory $\{u_k\}_{k=0}^{N-1} \subset \mathbb{R}^{n_u}$. For contact-rich planning problems, we introduce "*contact variables*" $\{\lambda_k\}_{k=0}^{N-1} \subset \mathbb{R}^{n_\lambda}$, which can be interpreted either as a set of binary contact modes or as continuous contact forces (see examples in §IV). With these definitions, we focus on the following general contact-rich planning problem

$$\min_{\substack{\{x_k\}_{k=0}^{N}, \{u_k\}_{k=0}^{N-1} \\ \{\lambda_k\}_{k=0}^{N-1}}} \quad \ell_N(x_N) + \sum_{k=0}^{N-1} \ell_k(x_k, u_k, \lambda_k) \tag{1a}$$

$$\text{subject to} \quad x_0 = x_{\text{init}} \tag{1b}$$

$$F_k(x_{k-1}, u_{k-1}, \lambda_{k-1}, x_k) = 0, \ k \in [N] \tag{1c}$$

$$(u_{k-1}, \lambda_{k-1}, x_k) \in \mathcal{C}_k, \ k \in [N] \tag{1d}$$

where $\ell_k, k = 0, \ldots, N$ represents instantaneous loss and terminal loss functions. F_k represents the discretized system dynamics obtained from differential algebraic equations and multiple shooting, possibly involving explicit or implicit contact mode switching. $\mathcal{C}_k$ imposes various types of constraints on $u_{k-1}, \lambda_{k-1}, x_k$, including (a) control limits; (b) geometric constraints such as collision avoidance; (c) complementarity constraints related to contact. A well-known special case of (1) occurs when the dynamics are linear when fixing λ_k's. In such case, (1) can be modeled either as mixed-integer linear/quadratic programming [10, 28] or as linear complementarity problems [3, 51].

In this paper, we do not assume linearity or convexity but assume (a) ℓ_k and F_k are polynomial functions; (b) $\mathcal{C}_k$ is basic semi-algebraic (*i.e.,* described by polynomial constraints). Thus, (1) becomes a *polynomial optimization problem* (POP). Formulating robotics problems as polynomial optimization is now well established in the literature [23, 47, 16, 39], because 3D rotations and rigid body dynamics expressed in maximal coordinates [5] admit natural polynomial representations.

Previous methods. We briefly review five different methods for solving the contact-rich planning (1). (a) *Hybrid MPC*: These methods alternate between contact sequence generation using discrete search [7, 45, 8, 30] and continuous-state planning with a fixed sequence. (b) *Mixed-integer programming*: Contact modes are modeled as binary variables, leading to mixed-integer convex programming (with linear dynamics) [10, 28] or mixed-integer nonconvex programming (with nonlinear dynamics) [18]. These methods scale poorly with the planning horizon, as the worst-case computational complexity grows exponentially with the number of binary variables. (c) *Dynamics smoothing*: This approach approximates nonsmooth complementarity constraints with smooth surrogate functions, simplifying the problem into a smooth nonlinear programming formulation suitable for local solvers [6, 38, 31, 37]. Convex smoothing methods [34] also exist, at the cost of locally linearizing the dynamics. (d) *Contact-implicit methods*: These mainstream methods encode contact modes implicitly through contact forces and complementarity constraints. Numerous local solvers are based on this framework [2, 52, 35, 27, 49, 22]. However, it is well known that contact-implicit planning problems fail the common constraint qualifications that are crucial for the convergence of numerical solvers. (e) *Graph of convex sets (GCS)*: As a recently proposed powerful planning framework that explicitly models both discrete and continuous actions, GCS has been extended to contact-rich tasks [13, 48, 32]. These methods can be viewed as an extension of mixed-integer nonconvex optimization with two-level convex relaxations where level one is a semidefinite relaxation and level two involves inequality multiplication. However, in contact-rich motion planning, achieving both tight relaxations and fast solve times remains challenging, and the current GCS framework has yet to fully integrate nonlinear dynamics with geometric constraints.

Is it possible to solve the contact-rich planning problem (1) *to (near) global optimality efficiently?*

"Sparse" Moment-SOS hierarchy. Modeling contact-implicit planning as polynomial optimization (POP) in (1) brings both opportunities and challenges.

- **Opportunities.** Lasserre's hierarchy of moment and sums-of-squares (SOS) relaxations [19] provides a principled and powerful machinery for global optimization of POPs through convex relaxations. Particularly, the Moment-SOS hierarchy generates a series of convex *semidefinite programs* (SDPs) with growing sizes whose optimal values provide nondecreasing *lower bounds* that asymptotically converge to the global minimum of (1). Combined with a feasible (or locally optimal) solution of (1) that provides an *upper bound* to the global minimum, one can compute increasingly tight (small) *(sub)optimality certificates* by measuring the relative gap between the lower bound and the upper bound. To enhance scalability of the hierarchy, "*sparse*" Moment-SOS hierarchy has been proposed to exploit sparsity in the POPs, including correlative sparsity [21, 15] and term sparsity [43, 26] (see more details in §II). Notably, the Julia package TSSOS [25] supports automatic sparsity exploitation as long as the user provides a POP formulation. The recent work [40] in robotics applied TSSOS to several motion planning problems and demonstrated that sparse relaxations can deliver small suboptimality gaps. Furthermore, [16] has shown that all trajectory optimization problems exhibit a generic *chain-like* correlative sparsity pattern and designed a GPU-based ADMM SDP solver that achieves significant speedup than off-the-shelf SDP solvers. *Can we directly apply sparse Moment-SOS relaxations to the contact-implicit planning problem* (1)*?*
- **Challenges.** The answer is unfortunately NO, due to three challenges. First, multiple contact modes will make the chain-like correlative sparsity pattern introduced in [16] too large to be solved efficiently. Second, TSSOS allows exploiting more flexible sparsity patterns but its automatic sparsity exploitation operates like a black box—it does not visualize the sparsity patterns being exploited and it is unclear whether robotics-specific domain knowledge can

lead to customized sparsity. Third, as reported in [40], the suboptimality gaps when using TSSOS for many smooth planning problems are already large (above 20%), not to mention the extra nonsmoothness and combinatorial complexity brought by contact-implicit planning. As shown in [39], TSSOS can fail to extract feasible solutions for certain difficult instances of (1).

Contributions. In this paper, we tackle the aforementioned challenges and show that it is indeed possible to solve many instances of the contact-implicit planning problem (1) to near global optimality. The key strategy is to build "sparsity-rich" semidefinite relaxations from the ground up, for robotics.

We summarize our contributions as follows.

(I) **White-box sparsity exploitation.** We provide a tutorial-style review of the fundamental mathematical concepts underpinning correlative and term sparsity for POPs, and further ground our discussion in a concrete contact-implicit planning problem. We build a new C++ Sparse Polynomial Optimization Toolbox (SPOT), interfacing both Matlab and Python, that (a) is faster than TSSOS, (b) offers richer relaxation options, and (c) visualizes the automatically discovered sparsity patterns (see Fig. 1).

(II) **Robotics-specific sparsity.** Beyond automatic exploitation of the generic correlative and term sparsity, we show that it is possible and crucial to exploit robotics-specific sparsity patterns. Particularly, we investigate sparsities derived from robot kinematic chains and separable contact modes, and demonstrate that robotics-specific sparsity patterns achieve both tighter lower bounds and reduced computation times compared to automatically generated ones in large-scale problems such as Planar Hand.

(III) **Robust minimizer extraction.** An important but often overlooked problem in SDP relaxations is how to extract good solutions to the nonconvex optimization from optimal SDP solutions, especially when the relaxation is not tight (*i.e.,* the suboptimality gap is large). Inspired by the recent advances in Gelfand-Naimark-Segal (GNS) construction [17], we develop a new minimizer extraction routine for sparse Moment-SOS relaxations that demonstrates superior robustness over naive extraction methods previously implemented in [16, 25].

(IV) **Extensive case studies.** We test our sparse semidefinite relaxations on five contact-rich planning problems: Push Bot, Push Box, Push T, Push Box with Obstacles, and Planar Hand. Of independent interests, some of our polynomial modeling techniques also appear to be new in the planning literature. Thanks to rich sparsity, the generated small-to-medium scale SDP relaxations can be solved in *seconds* while achieving decent tightness and certified global optimality. Furthermore, we showcase robust push-T performance of our SDP relaxations using a real-world robotic manipulator. In fact, with global optimization, model predictive control is so robust that it succeeds the task even under severe environment disturbances that effectively make the "model wrong" (see Fig. 1).

Paper organization. We present correlative and term sparsity in §II, robotics-specific sparsity in §III. We give numerical and real-world experiments in §IV, and conclude in §V.

Notations. Let $\mathbf{x} = (x_1, \dots, x_n)$ be a tuple of variables and $\mathbb{R}[\mathbf{x}] = \mathbb{R}[x_1, \dots, x_n]$ be the set of polynomials in $\mathbf{x}$ with real coefficients. A monomial is defined as $\mathbf{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$. A polynomial in $\mathbf{x}$ can be written as $f(\mathbf{x}) = \sum_{\boldsymbol{\alpha} \in \mathbb{N}^n} f_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$ with coefficients $f_{\boldsymbol{\alpha}} \in \mathbb{R}$. We denote the set of all polynomials with degree less than or equal to d as $\mathbb{R}_d[\mathbf{x}]$. The support of f is defined by $\operatorname{supp}(f) = \{\boldsymbol{\alpha} \in \mathbb{N}^n \mid f_{\boldsymbol{\alpha}} \neq 0\}$, i.e., the set of exponents with nonzero coefficients. The set of all variables contained in f is defined by $\operatorname{var}(f)$. Let $\mathbf{x}^{\mathbb{N}^n_d}$ be the *standard monomial basis*, abbreviated as $[\mathbf{x}]_d$. Given an index set $I \subseteq [n]$, let $\mathbf{x}(I) = (x_i, i \in I)$ and $[\mathbf{x}(I)]_d$ denote the standard monomial basis of the subspace spanned by the variables $x_i, i \in I$. An undirected graph $G(V, E)$ consists of a vertex set $V = v_1, v_2, \dots, v_n$ and an edge set $E \subseteq \{(v_i, v_j) \mid v_i, v_j \in V, v_i \neq v_j\}$. Let $\mathbf{S}^n$ denote the space of $n \times n$ symmetric matrices, and $\mathbf{S}^n_+$ denote the cone of $n \times n$ symmetric positive semidefinite (PSD) matrices.

II. Correlative and Term Sparsity

In this section, we review a systematic mechanism to relax a general (nonconvex) polynomial optimization problem (POP) as a (convex) semidefinite program (SDP) while exploiting two levels of sparsity: (a) variable level—correlative sparsity (CS); and (b) term level—term sparsity (TS). Formally, we consider the following POP

$$\rho^\star = \min_{\mathbf{x} \in \mathbb{R}^n} \left\{ f(\mathbf{x}) \;\middle|\; \begin{array}{l} g_1(\mathbf{x}) \geq 0, \dots, g_{m_{\text{ineq}}}(\mathbf{x}) \geq 0, \\ h_1(\mathbf{x}) = 0, \dots, h_{m_{\text{eq}}}(\mathbf{x}) = 0 \end{array} \right\} \tag{2}$$

where the objective function f and the constraints g_i, h_i are all real polynomials. To ground our discussion in a concrete robotics example, let us consider the following simple contact-rich motion planning problem.

Example 1 (Double Integrator with Soft Wall)**.** *As shown in Fig. 2, consider a point mass m driven by a control force u that can bounce between two soft walls with spring coefficients k_1 and k_2. Denote the system state as (x, v) (x: position, v: velocity), the control input as u, and the two wall's forces as (λ_1, λ_2), we consider the following trajectory optimization (optimal control) problem*

$$\min \quad \sum_{k=0}^{N-1} u_k^2 + x_{k+1}^2 + v_{k+1}^2 \tag{3a}$$

$$\text{s.t.} \quad x_{k+1} - x_k = \Delta t \cdot v_k \tag{3b}$$

$$v_{k+1} - v_k = \frac{\Delta t}{m} \cdot (u_k + \lambda_{1,k} - \lambda_{2,k}) \tag{3c}$$

$$u_{\max}^2 - u_k^2 \geq 0 \tag{3d}$$

$$0 \leq \lambda_{1,k} \perp \frac{\lambda_{1,k}}{k_1} + d_1 + x_k \geq 0 \tag{3e}$$

$$0 \leq \lambda_{2,k} \perp \frac{\lambda_{2,k}}{k_2} + d_2 - x_k \geq 0 \tag{3f}$$

$$x_0 = x_{\text{init}} \quad \textit{and} \quad v_0 = v_{\text{init}} \tag{3g}$$

where Δt represents the time discretization, (3a) formulates a quadratic regulation loss function around $x = 0, v = 0$, (3b)-(3c) represent system dynamics, (3d) enforces control saturation, (3e)-(3f) specify the soft complementarity constraints between position and contact forces, and (3g) provides the initial condition. $\lambda_{1,k}$ denotes the contact force at step k.

Outline. We begin by introducing the fundamentals of chordal graphs, a key mathematical tool for analyzing sparsity patterns (§II-A). Next, we review correlative sparsity (CS) (§II-B) and term sparsity (TS) (§II-C) separately. Finally, we introduce our high-performance C++ sparse polynomial optimization toolbox, SPOT (§II-D).

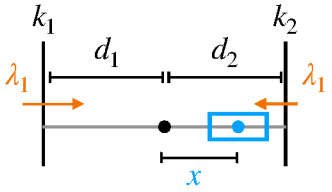


Fig. 2: Double integrator with soft wall.

A. Chordal Graph

Definition 2 (Chordal graph)**.** *A graph $G = (V, E)$ is* chordal *if every cycle of four or more vertices has a chord—an edge connecting two non-adjacent vertices in the cycle.*

For a quick example, the graph in Fig. 3(a) is not chordal but the graph in Fig. 3(b) is. Apparently, one can change a non-chordal graph to chordal by adding edges. This is the notion of a chordal extension.

Chordal extension. A graph $G'(V', E')$ is called a *chordal extension* of graph $G(V, E)$ if (a) G' is chordal, and (b) $V' = V$ and $E \subseteq E'$. The ideal objective of chordal extension is to add the *minimum* number of edges to G to make G' chordal. However, it is known that finding such a minimal chordal extension is NP-complete [50]. A common heuristic for approximating the minimal chordal extension is the *minimal degree* (MD) chordal extension [36]. An alternative heuristic is to select vertices based on the number of additional edges (MF) required to maintain chordality [50]. We summarize the MD chordal extension method in Algorithm 1 and MF chordal extension in Algorithm 2 in Appendix A. In our SPOT package, we implement both algorithms.

Maximal cliques. Once the chordal extension is constructed, the next step is to identify the *maximal cliques*.

Definition 3 (Clique)**.** *A* clique *in a graph $G(V, E)$ is a subset of vertices $C \subseteq V$ where every pair of vertices is connected by an edge. A clique is* maximal *if it is not properly contained within any other clique in G.*

The nice consequence of the chordal extension is that the maximal cliques of a chordal graph can be enumerated efficiently in linear time in terms of the number of nodes and edges [11, 4, 12]. Identifying all maximal cliques in a chordal graph plays a fundamental role in exploiting sparsity patterns.

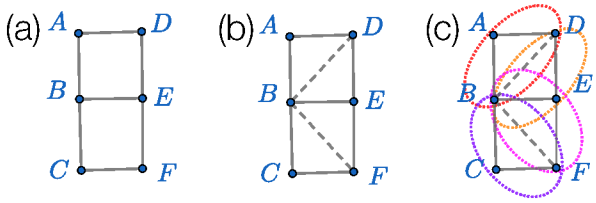


Fig. 3: An example of chordal extension and maximal cliques.

Example 4 (Chordal Extension and Maximal Cliques)**.** *Consider the graph in Fig. 3(a) that is non-chordal. The MD chordal extension Algorithm 1 adds two edges (B, D), (B, F), leading to the chordal graph in Fig. 3(b). The maximal cliques of the resulting chordal graph are $\{A, B, D\}$, $\{B, D, E\}$, $\{B, E, F\}$, and $\{B, C, F\}$, as shown in Fig. 3(c).*

With the notion of a chordal graph and maximal cliques, it is natural to use such a graph-theoretic tool to exploit sparsity.

B. Correlative Sparsity

As mentioned before, correlative sparsity (CS) seeks to exploit sparsity in the "variable" level. Roughly speaking, the intuition is that we can construct a graph that represents the connectivity in the POP (2), perform a chordal extension to that graph, and find its maximal cliques to group the (potentially very large number of) POP variables into many groups where each group only contains a few variables [41].

CS graph construction. The graph $G^{\text{csp}}(V, E)$ is the correlative sparsity pattern (CSP) graph of a POP with variables $\mathbf{x} \in \mathbb{R}^n$ if $V = [n]$ and $(i, j) \in E$ if at least one of the following three conditions holds:

1) $\exists \boldsymbol{\alpha} \in \text{supp}(f)$, s.t. $\alpha_i, \alpha_j > 0$,
2) $\exists k \in [m_{\text{ineq}}]$, s.t. $x_i, x_j \in \text{var}(g_k)$,
3) $\exists k \in [m_{\text{eq}}]$, s.t. $x_i, x_j \in \text{var}(h_k)$.

In words, the nodes of the CSP graph represent variables of the POP, and a pair of nodes are connected if and only if they simultaneously appear in the objective or the constraints.

Chordal extension and grouping. With the MD chordal extension Algorithm 1 (or the MF chordal extension Algorithm 2), we compute the chordal extension of G^{csp}— denoted $(G^{\text{csp}})'$—and its maximal cliques $\{I_l\}_{l=1}^p$, where each clique I_l contains a set of nodes (variables). We then partition the constraint polynomials $g_1, \ldots, g_{m_{\text{ineq}}}$ and $h_1, \ldots, h_{m_{\text{eq}}}$ into groups $\{g_j \mid j \in \mathcal{G}_l\}$ and $\{h_j \mid j \in \mathcal{H}_l\}$, where $\mathcal{G}_l$ and $\mathcal{H}_l$, $l \in [p]$ index the inequality and equality constraints involving the variables in I_l, respectively. Formally, this is

1) $\forall j \in \mathcal{G}_l, \text{var}(g_j) \subseteq I_l$,
2) $\forall j \in \mathcal{H}_l, \text{var}(h_j) \subseteq I_l$.

We make this concrete by recalling our robotics Example 1.

Example 5 (CSP Graph of Example 1)**.** *Fig. 4 shows the CSP graph of Example 1. It is already a chordal graph without chordal extension. Two maximal cliques are highlighted: $\{x_0, v_0, \lambda_{1,0}, \lambda_{2,0}\}$ and $\{x_0, x_1, v_0\}$. There is an edge between x_0 and v_0 since $x_1 - x_0 = \Delta t \cdot v_0$ shows up in (3b).*

After grouping the variables into cliques $\{I_l\}_{l=1}^p$ and the polynomial constraints into subsets $\mathcal{H}_l, \mathcal{G}_l$, we can design

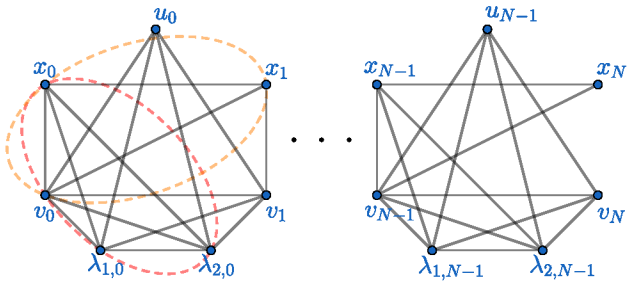


Fig. 4: CSP graph of the toy Example 1. Red circle: maximal clique $\{x_0, v_0, \lambda_{1,0}, \lambda_{2,0}\}$. Orange circle: maximal clique $\{x_0, x_1, v_0\}$. Only variables in the first and last planning steps are shown for simplicity.

a hierarchy of "sparse" moment and sums-of-squares (SOS) relaxations to globally optimize the POP (2).

Before we present the sparse Moment-SOS hierarchy, it is useful to understand the "dense" Moment-SOS hierarchy.

Dense Moment-SOS relaxations. Recall the POP (2). Let $\mathbf{y} = (y_{\boldsymbol{\alpha}})_{\boldsymbol{\alpha}}$ be a sequence of real numbers indexed by the standard monomial basis of $\mathbb{R}[\mathbf{x}]$. Define the Riesz linear functional $L_{\mathbf{y}}: \mathbb{R}[\mathbf{x}] \to \mathbb{R}$ as:

$$f = \sum_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}} \mapsto \sum_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}} y_{\boldsymbol{\alpha}}, \quad \forall f(\mathbf{x}) \in \mathbb{R}[\mathbf{x}] \tag{4}$$

In words, the Riesz linear functional $L_{\mathbf{y}}$ transforms a real polynomial f to a real number that is the inner product between $\mathbf{y}$ and the vector of coefficients of f. The notation of $L_{\mathbf{y}}$ can be naturally extended to polynomial vectors and matrices, as illustrated in the following example.

Example 6 (Riesz Linear Functional). *Let* $n = 3$, $I = \{1, 3\}$. *Then* $[\mathbf{x}]_1 = [1; x_1; x_2; x_3]$ *and* $[\mathbf{x}(I)]_2 = [1; x_1; x_3; x_1^2; x_1x_3; x_3^2]$. *Applying* $L_{\mathbf{y}}$, *we have*

$$L_{\mathbf{y}}((x_1 + x_3) \cdot [\mathbf{x}(I)]_2) = \begin{bmatrix} y_{1,0,0} + y_{0,0,1} \\ y_{2,0,0} + y_{1,0,1} \\ y_{1,0,1} + y_{0,0,2} \\ y_{3,0,0} + y_{2,0,1} \\ y_{2,0,1} + y_{1,0,2} \\ y_{1,0,2} + y_{0,0,3} \end{bmatrix}, \tag{5}$$

$$L_{\mathbf{y}}([\mathbf{x}]_1 [\mathbf{x}]_1^T) = \begin{bmatrix} y_{0,0,0} & y_{1,0,0} & y_{0,1,0} & y_{0,0,1} \\ y_{1,0,0} & y_{2,0,0} & y_{1,1,0} & y_{1,0,1} \\ y_{0,1,0} & y_{1,1,0} & y_{0,2,0} & y_{0,1,1} \\ y_{0,0,1} & y_{1,0,1} & y_{0,1,1} & y_{0,0,2} \end{bmatrix}, \tag{6}$$

where the number $y_{1,0,0}$ *is applying* $L_{\mathbf{y}}$ *to the monomial* $x_1^1 \cdot x_2^0 \cdot x_3^0 = x_1$.

With the Riesz linear functional, we can state the dense Moment-SOS hierarchy. Essentially, through the Riesz linear functional $L_{\mathbf{y}}$, the Moment-SOS hierarchy relaxes the original POP (2) as a convex optimization problem whose variable becomes the sequence $\mathbf{y}$. The reason why this is called a "hierarchy" is because one can make the sequence arbitrarily long, depending on how many monomials are included.

Proposition 7 (Dense Moment-SOS Hierarchy). *Consider the POP* (2). *Let* $d_j^g = \lceil deg(g_j)/2 \rceil$, $\forall j \in [m_{ineq}]$ *and* $d_j^h = deg(h_j)$, $\forall j \in [m_{eq}]$. *Define*

$$d_{\min} = \max\left\{ \lceil deg(f)/2 \rceil, \{d_j^g\}_{j \in [m_{ineq}]}, \{\lceil d_j^h/2 \rceil\}_{j \in [m_{eq}]} \right\}. \tag{7}$$

Given a positive integer $d \geq d_{\min}$, *the* d*-th order dense Moment-SOS hierarchy reads:*

$$\min_{\mathbf{y}} \quad L_{\mathbf{y}}(f) \tag{8a}$$

$$s.t. \quad L_{\mathbf{y}}\left([\mathbf{x}]_d [\mathbf{x}]_d^{\mathsf{T}}\right) \succeq 0 \tag{8b}$$

$$L_{\mathbf{y}}\left(g_j \cdot [\mathbf{x}]_{d-d_j^g} [\mathbf{x}]_{d-d_j^g}^{\mathsf{T}}\right) \succeq 0, \forall j \in \mathcal{G}_l, l \in [p] \tag{8c}$$

$$L_{\mathbf{y}}\left(h_j \cdot [\mathbf{x}]_{2d-d_j^h}\right) = 0, \forall j \in \mathcal{H}_l, l \in [p] \tag{8d}$$

$$\mathbf{y_0} = 1 \tag{8e}$$

The optimal value of the convex optimization (8) *converges to the optimal value of the nonconvex POP* (2) $\rho^\star$ *as* $d \to \infty$.

In (8), the matrix $L_{\mathbf{y}}([\mathbf{x}]_d[\mathbf{x}]_d^{\mathsf{T}})$ is usually called a *moment matrix* and it is enforced by a positive semidefinite (PSD) constraint. One can see that the dense Moment-SOS hierarchy can become expensive very quickly as the relaxation order d increases. This is because the "dense" moment matrix at order d has size $\binom{n+d}{d}$ which quickly makes the PSD constraint too large to be handled by off-the-shelf SDP solvers (recall that the length of the monomial basis indexing the moment matrix—$[\mathbf{x}]_d$—is $\binom{n+d}{d}$).

Moment-SOS relaxations with CS. On the other hand, with correlative sparsity and when the variables are divided into cliques $\{I_l\}_{l=1}^p$ where the size of clique I_l is n_l, then instead of generating a single moment matrix with size $\binom{n+d}{d}$, the sparse Moment-SOS hierarchy will generate p moment matrices where the size of each moment matrix is $\binom{n_l+d}{d}$. A different way to view this is that, correlative sparsity breaks a large PSD constraint into multiple smaller PSD constraints.

Let us formalize this.

Proposition 8 (Sparse Moment-SOS Hierarchy). *Consider the POP* (2) *and assume its variables are grouped into cliques* $\{I_l\}_{l=1}^p$ *and its constraints are grouped into* $\mathcal{G}_l, \mathcal{H}_l$ *where* $\mathcal{G}_l$ *and* $\mathcal{H}_l$ *include constraints only involving variables* $\mathbf{x}[I_l]$. *For any fixed integer* $d \geq d_{\min}$ ($d_{\min}$ *defined as in* (7)), *define the following polynomial matrices and vectors associated with each clique* I_l *as:*

$$M_d(I_l) = [\mathbf{x}(I_l)]_d [\mathbf{x}(I_l)]_d^{\mathsf{T}}, l \in [p] \tag{9a}$$

$$M_d(g_j, I_l) = g_j \cdot [\mathbf{x}(I_l)]_{d-d_j^g} [\mathbf{x}(I_l)]_{d-d_j^g}^{\mathsf{T}}, j \in \mathcal{G}_l, l \in [p] \tag{9b}$$

$$H_d(h_j, I_l) = h_j \cdot [\mathbf{x}(I_l)]_{2d-d_j^h}, j \in \mathcal{H}_l, l \in [p] \tag{9c}$$

Let $g_0 := 1$, *then* $M_d(g_0, I_l) = M_d(I_l)$. *The* d*-th order*

Moment-SOS hierarchy with correlative sparsity reads:

$$\rho_d := \min_{\mathbf{y}} \quad L_{\mathbf{y}}(f) \tag{10a}$$
$$s.t. \quad L_{\mathbf{y}}\left(M_d(g_j, I_l)\right) \succeq 0, \forall j \in \{0\} \cup \mathcal{G}_l, l \in [p] \tag{10b}$$
$$L_{\mathbf{y}}\left(H_d(h_j, I_l)\right) = 0, \forall j \in \mathcal{H}_l, l \in [p] \tag{10c}$$
$$\mathbf{y_0} = 1 \tag{10d}$$

Moreover, under appropriate compactness assumptions [20], the sequence $\rho_d \to \rho^\star$ as $d \to \infty$.

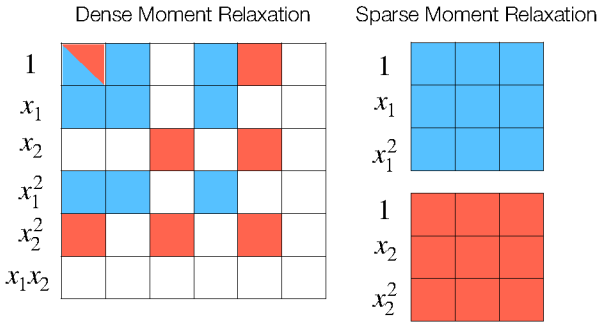


Fig. 5: Comparison of the moment matrices in dense and sparse Moment-SOS relaxations.

As an illustrative example, suppose the POP has two variables $\mathbf{x} = (x_1, x_2)$. Then the moment matrix corresponding to the dense Moment-SOS hierarchy at $d = 2$ is shown in Fig. 5 left, where its rows and columns are indexed by the standard monomial basis $[\mathbf{x}]_2$. However, suppose from the chordal extension of the CSP graph one can find two cliques $\{x_1\}$ and $\{x_2\}$ (i.e., they are not connected), then the sparse Moment-SOS hierarchy at $d = 2$ would generate two moment matrices shown in Fig. 5 right. From the color coding in Fig. 5, we can see that the two smaller moment matrices are principal submatrices of the big moment matrix. Hence, a 6×6 PSD constraint is broken into two 3×3 PSD constraints.

It is worth mentioning that (a) while both the dense and the sparse Moment-SOS hierarchy converge to $\rho^\star$, they may, and in general, converge at different speeds; (b) associated with both (8) and (10) are their dual sums-of-squares (SOS) convex relaxations (hence the name Moment-SOS hierarchy). Though we do not explicitly state the SOS relaxations (see [46] and references therein), our SPOT package implements them.

C. Term Sparsity

While correlative sparsity (CS) focuses on relationships between variables, term sparsity (TS) addresses relationships between monomials. Specifically, TS is designed to partition monomial bases into blocks according to the monomial connections in the POP [43]. Rather than analyzing general TS in isolation, we focus on the integration of CS and TS [44].

Similar to CS, exploiting TS is also related to constructing a graph called the term sparsity pattern (TSP) graph. This construction involves three steps: (a) initialization; (b) support extension; and (c) chordal extension.

We remark that the notion of TS and the construction of the TSP graph can be less intuitive than the CSP graph mentioned before, and the mathematical notations can get quite involved. However, it is safe for the reader to quickly glance the TSP construction just to understand its high-level idea, and revisit the math a couple more times later.

Initialization. Let I_l, $l \in [p]$ be the maximal cliques of $(G^{\text{csp}})'$, with $n_l := |I_l|$ the size of each clique. Let $\mathcal{G}_l$ and $\mathcal{H}_l$, $l \in [p]$ be defined in the CS grouping procedure and contain polynomial constraints related to clique I_l. The variables $\mathbf{x}$ are grouped into subsets $\mathbf{x}(I_1), \ldots, \mathbf{x}(I_l)$. Denote $\mathcal{A}$ as the set of all monomials appearing in the POP (2), union with all even-degree monomials:

$$\mathcal{A} := \text{supp}(f) \cup \bigcup_{j=1}^{m_{\text{ineq}}} \text{supp}(g_j) \cup \bigcup_{j=1}^{m_{\text{eq}}} \text{supp}(h_j) \cup (2\mathbb{N})^n \tag{11}$$

where $(2\mathbb{N})^n$ is defined as $\{2\boldsymbol{\alpha} \mid \boldsymbol{\alpha} \in \mathbb{N}^n\}$.

Support extension. For each $l \in [p]$ and $j \in \{0\} \cup \mathcal{G}_l$, construct constraint g_j's TSP graph $G^g_{d,l,j}$ with:

1) Nodes: $V_{d,l,j} := |[\mathbf{x}(I_l)]_{d-d_j^g}|$ (these relate to monomials)
2) Edges: $E_{d,l,j} = \{(\beta, \gamma) \mid \text{supp}(M_d(g_j, I_l)_{\beta,\gamma}) \cap \mathcal{A} \neq 0\}$, where

$$M_d(g_j, I_l)_{\beta,\gamma} = g_j \cdot [\mathbf{x}(I_l)]_{d-d_j^g}(\beta) \cdot [\mathbf{x}(I_l)]_{d-d_j^g}(\gamma). \tag{12}$$

For each each $l \in [p]$ and $j \in \mathcal{H}_l$, define the binary mask vector $B^h_{d,l,j}$ as:

$$B^h_{d,l,j}(\beta) = \begin{cases} 1, & \text{supp}(H_d(h_j, I_l)_\beta) \cap \mathcal{A} \neq \emptyset \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

where

$$H_d(h_j, I_l)_\beta = h_j \cdot [\mathbf{x}(I_l)](\beta). \tag{14}$$

Chordal extension. For each $l \in [p]$ and $j \in \{0\} \cup \mathcal{G}_l$, let $G'^g_{d,l,j}$ be the chordal extension of $G^g_{d,l,j}$. Define $B^g_{d,l,j}$ as its adjacency matrix, which is naturally a binary mask matrix.

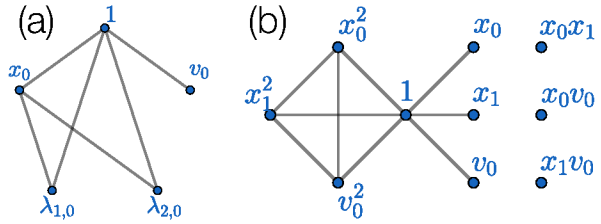


Fig. 6: (a) $G^g_{2,1,1}$: support extension for g_1 in clique I_1; (b) $G^g_{2,2,0}$: support extension for g_0 in clique I_2.

We shall illustrate this using our robotics example.

Example 9 (TSP Graph of Example 1)**.** *Consider the toy example's two cliques: $I_1 := \{x_0, v_0, \lambda_{1,0}, \lambda_{2,0}\}$ and $I_2 := \{x_0, x_1, v_0\}$ (illustrated in Fig. 4). Define $g_1 := \lambda_{1,0} + 1 + x_0$ and $g_0 := 1$. Then, $G^g_{2,1,1}$ is illustrated in Fig. 6(a).*

There exists an edge between x_0 and $\lambda_{1,0}$ since x_0 and $\lambda_{1,0}$ appear at the second and fifth positions of $\mathbf{x}(I_1)_{d-d_g} := [1, x_0, v_0, \lambda_{1,0}, \lambda_{2,0}]^\mathsf{T}$, respectively. According to (12),

$$\lambda_{2,0}x_0 \in M_2(g_1, I_1)_{2,5} = (\lambda_{1,0} + 1 + x_0) \cdot x_0 \cdot \lambda_{2,0} \tag{15}$$

and $\lambda_{2,0}x_0 \in \mathcal{A}$ *since* $\lambda_{2,0} \cdot (\frac{\lambda_{2,0}}{k_2} + d_2 - x_0) = 0$.

Similarly, $G^g_{2,2,0}$ *is illustrated in Fig. 6(b). There exists an edge between* x_0^2 *and* x_1^2 *since* $x_0^2 \cdot x_1^2$ *is of even degree. Both* $G^g_{2,1,1}$ *and* $G^g_{2,2,0}$ *are already chordal.*

Note that the crucial difference between CSP and TSP is that the nodes of a CSP graph are the variables while the nodes of a TSP graph are monomials. Therefore, while the goal of CSP is to break the entire variable $\mathbf{x}$ into smaller cliques of variables, the goal of TSP is to break the dense monomial basis $[\mathbf{x}(I_l)]_d$ into smaller cliques of monomials.

Combining CS and TS, we can further decompose the PSD constraints in the sparse Moment-SOS hierarchy (10) into smaller ones. Due to space constraints, we defer a formal presentation to Appendix B. The high-level intuition, however, is that the binary masks obtained from the TSP allow us to only focus on the entries of the matrix variables in (10) corresponding to nonzero entries in the masks.

D. Sparse Polynomial Optimization Toolbox (SPOT)

To automate the aforementioned sparsity exploitation, we develop a new C++ package SPOT. Compared with the Julia package TSSOS, SPOT is faster and also provides more options. (i) SPOT provides both general moment relaxation and SOS relaxation, which complements TSSOS (only provides SOS relaxation). (ii) For both CS and TS, SPOT provides four options: (a) maximal chordal extension (MAX); (b) minimal degree chordal extension (MD); (c) minimum fill chordal extension (MF); (d) user-defined. (iii) SPOT provides a special option "*partial term sparsity*", which enables TS only for the moment matrices and the localizing matrices (*i.e.,* inequality constraints), while applying CS to equality constraints. This heuristic approach is motivated by the observation that in SOS relaxation, tightening equality constraints merely introduces more free variables and does not significantly impact the solution time of an SDP solver. In many cases, partial TS yields tighter lower bounds while keeping the computation time nearly unchanged. (iv) Through our Matlab and Python interface, SPOT provides a way for the user to visualize the CSP and TSP graphs, as shown in Fig. 1. The automatic sparsity detection pipleline is illustrated in Fig. 7.

III. Robotics-Specific Sparsity

While automatic correlative and term sparsity (CS-TS) exploitation is powerful, it has two notable limitations. (a) It occasionally fails to capture time, spatial, and kino-dynamical sparsity inherent in contact-rich planning problems, such as the Markov property described in [16]. (b) The approach introduced in §II heavily depends on approximate minimal chordal extensions. Although theoretically rigorous, chordal extensions can substantially increase the size of variable or term cliques, resulting in scalability challenges.

To address these, we propose several robotics-specific sparsity patterns as auxiliary tools to complement the automatic sparsity. Similarly, we categorize robotics-specific sparsity patterns into two levels, (a) variable level and (b) term level. The semi-automatic robotics-specific sparsity pattern injection is shown in Fig. 7, marked in red.

A. Variable-level Robotics-Specific Sparsity

Kinematic chain. Numerous robotic systems exhibit chain-like (or tree-like) mechanical structures, commonly found in manipulation and locomotion. These structures lead to a natural separation of kinematic and dynamic variables across different links. For example, consider the following system:

$$x_{1,k+1} = x_{1,k} + u_{1,k}, \tag{16a}$$
$$x_{2,k+1} = x_{2,k} + u_{2,k}, \tag{16b}$$
$$x_{3,k+1} = x_{3,k} + u_{3,k}, \tag{16c}$$
$$(x_{1,k} - x_{2,k})^2 = r^2, \quad (x_{2,k} - x_{3,k})^2 = r^2 \tag{16d}$$

which can be viewed as a 1-D two-link chain, with three states (x_1, x_2, x_3) and two geometric constraints in (16d). If only exploring the Markov property in [16], a chain of cliques will be derived, with the k'th clique containing 9 variables:

$$\{x_{i,k}, x_{i,k+1}, u_{i,k}\}_{i=1}^3 . \tag{17}$$

However, since (x_1, x_2), (x_2, x_3) forms two links, and each $u_i, i \in \{1, 2, 3\}$ only has direct effect on x_i, we can further decompose the clique as three cliques of size 3:

$$\{x_{i,k}, x_{i,k+1}, u_{i,k}\}\,, \ i = 1, 2, 3 \tag{18}$$

and four cliques of size 2:

$$\{x_{i,m}, x_{i+1,m}\}\,, \ i = 1, 2, \ m = k, k+1, \tag{19}$$

as illustrated in Fig. 8(a). Note that this clique partition cannot be discovered by the general sparsity pattern introduced in §II, since the graph in Fig. 8(a) is not chordal.

Separation plane. Consider a general obstacle avoidance task: both the robot and the obstacles can be decomposed as a union of convex sets. We denote the decomposition of robot as $\{P_i\}_{i\in[m]}$ and the decomposition of obstacles as $\{Q_j\}_{j\in[n]}$. For each i and j, P_i has no collision with Q_j if and only if there exists a plane $H_{i,j}$ separating P_i and Q_j. Consider P_i and Q_j being both 2-D polygons:

$$A_{i,j}v_{r,x} + B_{i,j}v_{r,y} + C_{i,j} \geq 0, \ \forall v_r \in P_i's \text{ vertices} \tag{20}$$
$$A_{i,j}v_{o,x} + B_{i,j}v_{o,y} + C_{i,j} \leq 0, \ \forall v_o \in Q_j's \text{ vertices} \tag{21}$$

where $(A_{i,j}, B_{i,j}, C_{i,j})$ determines a 2-D separation plane (line) $H_{i,j}$. For constraints (20), $(A_{i,j}, B_{i,j}, C_{i,j})$ has no direct relationship between each other. Thus, in each time step, instead of defining a large variable clique:

$$\left\{\text{other variables}, \{(A_{i,j}, B_{i,j}, C_{i,j})\}_{i\in[m],j\in[n]}\right\} \tag{22}$$

we define mn smaller variable cliques:

$$\{\text{other variables}, (A_{i,j}, B_{i,j}, C_{i,j})\}_{i\in[m],j\in[n]} \tag{23}$$

The resulting clique size is invariant to m and n, as illustrated in Fig. 8(b). What's more, one can check that the decomposition (23) still satisfies the running intersection property (RIP) required in correlative sparsity pattern [21].

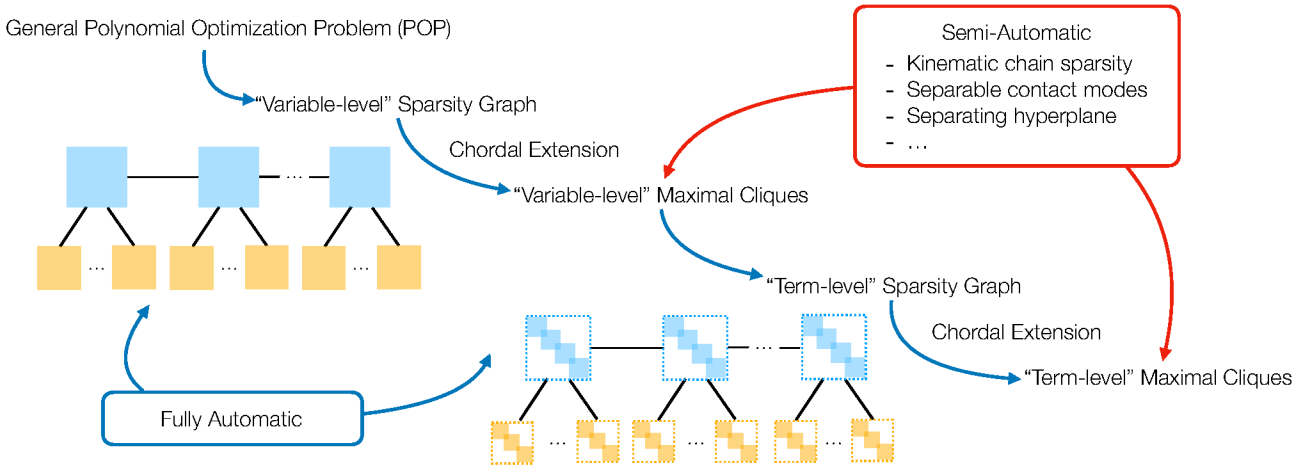
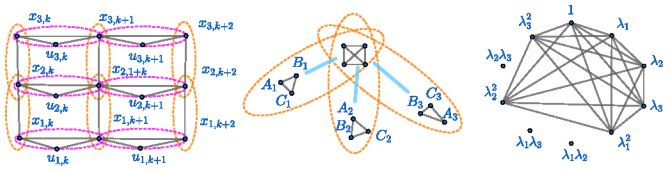


Fig. 7: Overall pipeline of the Sparse Polynomial Optimization Toolbox (SPOT). Blue curves: the automatic detection of correlative and term sparsity patterns. Red curves: semi-automatic injection of robotics-specific sparsity patterns.



(a) Kinematic chain (b) Separation plane (c) Separable contact

Fig. 8: Robotics-specific sparsity patterns.

B. Term-level Robotics-Specific Sparsity

Separable contact modes. Frequently in contact-rich planning, we will have to "select one out of a bunch of modes" (*cf.* §C2 and§C3). It can be modelled as polynomial equalities:

$$h_0 \triangleq \sum_{i\in[n]} \lambda_i^2 - 1 = 0 \tag{24}$$

$$h_i \triangleq \lambda_i \cdot (1-\lambda_i) = 0,\ \forall i \in [n], \tag{25}$$

where λ_i is a binary variable corresponding to whether the i-th contact mode is selected. In variable level, there is no sparsity in (24), since $\sum_{i\in[n]} \lambda_i^2 = 1$ groups all λ_i's together. However, we show that the sparsity is still rich in the term level. Consider the generation procedure of $\mathcal{A}$ in term sparsity (11), for the polynomial equality constraint system (24):

$$\mathcal{A} = \left\{1, \{\lambda_i\}_{i\in[n]}, \{\lambda_i^2\}_{i\in[n]}\right\}. \tag{26}$$

Consider the special case $n = 3$ in second-order relaxation $d = 2$, and there is only one variable clique (*i.e.,* $l = 1$). By definition of the polynomial multiplier $H_2(h_i, I_1), i \in \{0,1,2,3\}$ for equality constraints (9c):

$$H_2(h_i, I_1) = \left[1, \lambda_1, \lambda_2, \lambda_3, \lambda_1^2, \lambda_1\lambda_2, \lambda_1\lambda_3, \lambda_2^2, \lambda_2\lambda_3, \lambda_3^2\right]^\top. \tag{27}$$

If we proceed with the support extension procedure for equality constraints (13), then we have

$$B^h_{2,1,0} = [1,1,1,1,1,0,0,1,0,1] \tag{28}$$

$$B^h_{2,1,1} = [1,1,0,0,0,0,0,0,0,0] \tag{29}$$

$$B^h_{2,1,2} = [1,0,1,0,0,0,0,0,0,0] \tag{30}$$

$$B^h_{2,1,2} = [1,0,0,1,0,0,0,0,0,0] \tag{31}$$

In the second-order moment matrix, the unmasked terms are:

$$\begin{aligned} &1, \{\lambda_i\}_{i\in[3]}, \{\lambda_i^2\}_{i\in[3]}, \{\lambda_i^3\}_{i\in[3]}, \\ &\{\lambda_i^4\}_{i\in[3]}, \{\lambda_i\lambda_j^2\}_{i\in[3],j\in[3],i\neq j} \end{aligned} \tag{32}$$

leading to the moment matrix generated by a reduced basis:

$$\left\{1, \lambda_1, \lambda_2, \lambda_3, \lambda_1^2, \lambda_2^2, \lambda_3^2\right\} \tag{33}$$

as shown in Fig. 8(c). The key observation is that there is no $\lambda_i\lambda_j (i \neq j)$ in the new basis (i.e., the contact modes are separated). For a general n, the reduced basis is of size $2n+1$, which is much smaller than the standard monomial basis of size $\frac{(n+1)(n+2)}{2}$.

Separable contact forces. In contact-implicit formulation, each possible contact is modelled as a set of complementary constraints. Suppose there are n contact points, then a typical dynamics formulation is (*cf.* §C1 and §C5):

$$0 \leq \lambda_i \perp g_i(x) \geq 0,\ i \in [n] \tag{34}$$

$$\sum_{i\in[n]} f_i(\lambda_i, x, u) = 0 \tag{35}$$

where $x \in \mathbb{R}^{n_x}$ is the system state and $u \in \mathbb{R}^{n_u}$ is the control input. Similar to the contact mode case (24), $\sum_{i\in[n]} f_i(\lambda_i, x, u)$ groups all λ_i's together. Thus, there is no variable-level sparsity pattern. However, inspired by the reduced basis introduced in (33), we can directly write down an extended reduced basis containing x and u. For simplicity,

assume x and u are both of one dimension, f_i is of quadratic form and linear in λ_i, then the reduced basis is

$$\left\{ \begin{matrix} 1, x, u, x^2, xu, u^2, \\ \{\lambda_i\}_{i\in[n]}, \{\lambda_i^2\}_{i\in[n]}, \{x\lambda_i\}_{i\in[n]}, \{u\lambda_i\}_{i\in[n]} \end{matrix} \right\} \tag{36}$$

which grows linearly in n.

Although we introduce these robotics-specific sparsity patterns in the context of contact-rich motion planning, many—such as kinematic-chain structures and supporting-hyperplane constraints—are generic and apply equally well to tasks that do not involve contact.

IV. Experiments

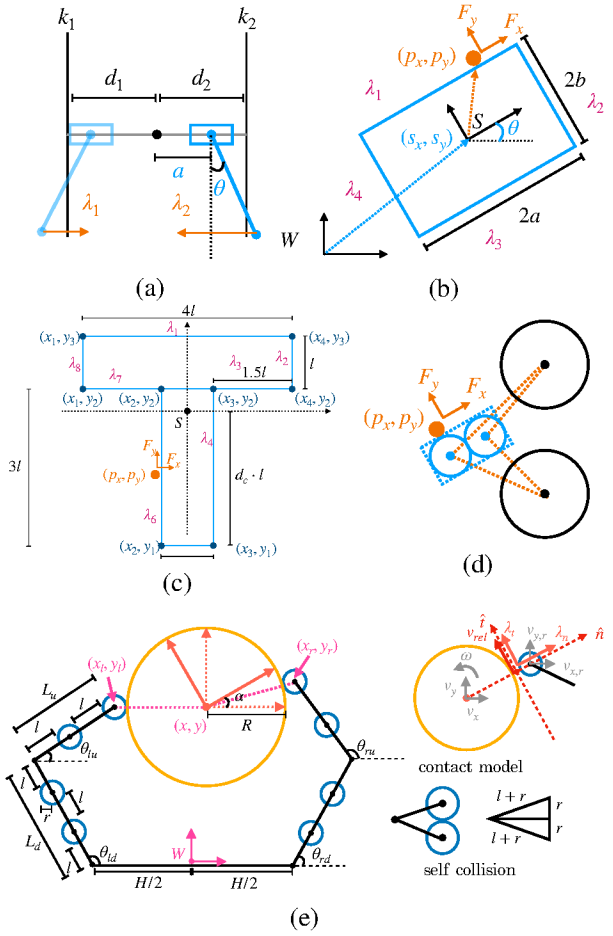


Fig. 9: illustrations of five contact-rich planning tasks. (a) Push Bot; (b) Push Box; (c) Push T-block; (d) Push Box with a Tunnel; (e) Planar Hand.

General setup. We consider five contact-rich problems.

1) **Push Bot:** A cart-pole system between two soft walls, shown in Fig. 9(a). The objective is to stabilize the cart-pole at $(a, \theta) = (0, \pi)$. For polynomial dynamics and other detailed settings, please refer to Appendix C1.
2) **Push Box:** A simple pusher-slider system, illustrated in Fig. 9(b). The goal is to push the box from an initial configuration (s_x, s_y, θ) to a target configuration. There are 4 possible contact modes at each time step. Detailed settings can be found in Appendix C2.
3) **Push T-Block:** Similar to Push Box, but the box is replaced with a T-block, resulting in 8 possible contact modes at each time step, as shown in Fig. 9(c). Please refer to Appendix C3 for detailed settings.
4) **Push Box with a Tunnel:** Similar to Push Box, except that two circular obstacles form a tunnel along the box's path to its goal, as shown in Fig. 9(d). Detailed settings are provided in Appendix C4.
5) **Planar Hand:** A two-fingered system rotating a 2D disk in a horizontal plane, as shown in Fig. 9(e). The goal is to rotate the disk by 360° using two fingertip contacts while minimizing the translation of the disk's center of mass. See Appendix C5 for further details.

Additionally, extensive real-world validations are conducted for the Push T task. Experiments were performed on a high-performance workstation equipped with a 2.7 GHz AMD 64-Core sWRX8 Processor and 1 TB of RAM, enabling MOSEK [1] to solve large-scale problems utilizing 64 threads.

Conversion speed. We compare SPOT and TSSOS in terms of conversion time across the five planning problems, considering different CS and TS options. In all cases, the planning horizon is set to $N = 30$. For the Push Box with a Tunnel problem, the CS relaxation order is set to $d = 3$ to obtain a tighter lower bound, while for the other examples, d is set to 2. The results are summarized in Table I. SPOT consistently outperforms TSSOS, achieving at least a $2\times$ speedup. For large-scale problems such as Push Box with a Tunnel, SPOT achieves approximately a $5\times$ speedup. For the Planar Hand problem, the automatic CS pattern generation produces clique sizes exceeding 20 through both MF and MD methods, resulting in a large-scale SDP with over 1 million constraints. Since TSSOS tightly integrates its conversion and SDP solving processes, it is difficult to isolate the conversion time. Therefore, we only report SPOT's conversion time for this case, which remains under 100 seconds despite the problem's scale.

Sparsity Examples	CS / TS	MF MAX	MF MF	MF MD	MD MAX	MD MF	MD MD
Push Bot	MOMENT	2.33	2.29	2.13	2.03	1.99	1.82
	SOS	2.39	2.43	2.26	2.09	2.13	1.97
	TSSOS	4.32	14.95	19.58	3.49	15.90	16.91
Push Box	MOMENT	1.68	1.58	1.52	1.39	1.31	1.24
	SOS	1.64	1.65	1.57	1.37	1.37	1.30
	TSSOS	4.56	8.83	10.12	3.94	9.60	8.92
Push T	MOMENT	3.67	3.58	3.29	4.01	3.95	3.48
	SOS	3.70	3.77	3.47	4.06	4.16	3.67
	TSSOS	15.54	42.86	43.41	13.13	43.22	45.02
Tunnel	MOMENT	21.19	18.03	16.19	19.78	19.79	17.69
	SOS	20.55	18.62	16.81	22.76	20.81	18.63
	TSSOS	90.37	109.25	112.20	83.47	129.17	125.63
Planar Hand	MOMENT	74.35	75.60	64.98	67.54	68.31	58.53
	SOS	88.60	91.35	80.70	80.70	83.14	70.75
	TSSOS	–	–	–	–	–	–

TABLE I: Conversion time comparison between SPOT and TSSOS across examples, with different CS-TS options.

Self-defined variable cliques. Since the automatic sparsity exploitation mechanism may fail to detect robotics-specific sparsity, we adopt the following clique generation procedure:

1) Generate a general sparsity pattern using SPOT.

2) Manually inspect the variable cliques to determine whether certain robotics-specific variable-level sparsity patterns can be incorporated.
3) Modify the generated cliques and resend them to SPOT using the "SELF" option.

For example, in the Planar Hand task, leveraging the kinematic chain pattern discussed in §III, we manually partition the variables at each time step into 14 smaller cliques, with sizes ranging from 6 to 14. Due to space constraints, these cliques are detailed in Appendix D. These manually defined cliques precisely correspond to the red circles in Fig. 1(a), illustrating the specific sparsity pattern.

Robust minimizer extraction. Due to the inherent complexity of contact-rich planning problems, it is not uncommon to encounter two types of relaxation "failure cases":

1) The sparse Moment-SOS Hierarchy is not tight.
2) The sparse Moment-SOS Hierarchy is tight but admits multiple solutions.

In both cases, the moment matrices fail to attain rank 1 (or other general tightness certificates). To extract minimizers, TSSOS and [16] both employ the same "naive" approach: obtain the degree-1 submatrix of each moment matrix, then average the normalized eigenvectors across different variable cliques. While straightforward to implement, this method is not robust in practice, often leading to infeasible local rounding and large suboptimality gaps. On the other hand, [17] demonstrates that the Gelfand-Naimark-Segal (GNS) construction provides a robust approach for minimizer extraction from a single moment matrix. Inspired by this, we propose the following heuristic algorithm for minimizer extraction in the presence of multiple moment matrices:

1) For each moment matrix, extract minimizers along with their associated weights using the GNS construction.
2) Select the minimizer with the highest weight and average it across different variable cliques.

This seemingly minor modification significantly improves robustness compared to the "naive" extraction method. We implemented the new minimizer extraction scheme in our SPOT package. A detailed discussion of GNS is beyond the scope of this paper; we refer interested readers to [17].

Numerical results. The results are presented in Table II. The planning horizon N is set to 30. For the Push Bot, Push Box, and Push T Block tasks, we evaluate 10 random initial states. For the Push Box with a Tunnel and Planar Hand tasks, we consider 5 random initial states. All reported statistics represent mean values. From semidefinite relaxation, we can get a lower bound f_{lower} of original nonconvex POP. After extracting solution, we use an in-house local solver to round a feasible solution with an upper bound f_{upper}. The suboptimality gap is defined as:

$$\eta_g := \frac{|f_{\text{lower}} - f_{\text{upper}}|}{1 + |f_{\text{lower}}| + |f_{\text{upper}}|} \tag{37}$$

We also report max KKT residual η_{kkt} and MOSEK solving time. Since SOS relaxation tends to generate much less constraint numbers compared to moment relaxation, and MOSEK is sensitive to constraint numbers, we only test SOS relaxations.

The numerical results reveal a clear trade-off between computational efficiency and relaxation tightness. When TS is not enabled, all average suboptimality gaps remain below 10%, except for the Planar Hand task. However, solving large-scale problems can take up to 5 minutes. Enabling TS with MF significantly improves computational efficiency. For the Push Bot, Push Box, and Push T Block tasks, we achieve near real-time solving speeds. However, this comes at the cost of significantly larger suboptimality gaps. In all of our case studies, the local solver recovered trajectories that succeed the tasks. Notably, there are two instances where MOSEK fails to solve the SDP to high accuracy: (a) Push Bot, SELF + MAX; (b) Planar Hand, SELF + MF. The underlying causes of these failures remain unknown. We also report the corresponding SDP constraint number, PSD cone number, and max PSD cone size in Table III. The demonstration of global optimal trajectories can be found in Fig. 10.

Real-world Push T-Block validations. We extensively validate our global optimal policy in the real world Push T-Block setting, as illustrated in Fig. 1(c). We use AprilTag [33] to get accurate pose estimation of the T-block. Due to the global optimality of our approach, a planning horizon of $N = 5$ is sufficient to generate high-quality planned trajectories. At each time step, we execute the pusher's first action and then re-plan. The average re-planning time is 3.7 seconds, with CS set to MF and TS set to NON. We evaluate our framework in 20 random trials, evenly split between two categories: clean push-T tasks, which involve standard push-T scenarios without disturbances, and "dirty" push-T tasks, where severe model mismatches and external disturbances are introduced to assess the planner's robustness. These disturbances include cases where the T-block is wrapped in a cable, put on some irregular surface, or contained within a small box. These three scenarios correspond to lines 2-4 in Fig. 1(c). Note that all these "dirty" push-T cases make our modeling (which is very simple) effectively "wrong". However, our planner demonstrates remarkable efficiency and robustness across all test cases, achieving a 100% success rate while naturally accommodating a wide range of initial states and challenging environments. For more demonstrations, please see Fig. 11 and our attached supplementary materials.

V. CONCLUSION

We introduced a new paradigm to contact-rich motion planning by exploiting both generic and robotics-specific sparsity patterns within semidefinite relaxations. Our method efficiently exploits correlative, term, and robotics-specific sparsity, enabling near-global optimization of complex robotic motion planning tasks. Through the Sparse Polynomial Optimization Toolbox (SPOT), we automated sparsity detection, significantly reducing computation time while maintaining solution quality. Extensive experiments, including various real-world push-T validations, demonstrated the robustness of our approach.

Examples	SELF + NON			SELF + MAX			SELF + MF		
	η_g (%)	$-\log_{10}(\eta_{\text{kkt}})$	time (s)	η_g (%)	$-\log_{10}(\eta_{\text{kkt}})$	time (s)	η_g (%)	$-\log_{10}(\eta_{\text{kkt}})$	time (s)
Push Bot	0.08	5.90	22.07	9.28	3.67	14.01	32.39	5.47	5.21
Push Box	0.15	5.65	8.90	0.20	5.70	5.76	13.77	6.37	1.67
Push T	7.83	5.30	46.20	17.10	5.67	29.35	35.98	5.50	4.99
Tunnel	4.89	5.52	342.31	5.22	5.35	266.17	8.20	5.04	33.31
Planar Hand	22.26	5.00	343.79	23.97	5.15	143.42	25.61	3.94	54.00

TABLE II: Comparison of suboptimality gap η_g, max KKT residual η_{kkt}, and MOSEK solving times for different tasks under SELF + NON, SELF + MAX, and SELF + MF settings. Throughout the table, we only consider SOS relaxation.

Examples	SELF + NON			SELF + MAX			SELF + MF		
	Constraint Number	PSD Cone Number	Max PSD Cone Size	Constraint Number	PSD Cone Number	Max PSD Cone Size	Constraint Number	PSD Cone Number	Max PSD Cone Size
Push Bot	45183	1429	66	32807	6310	32	11457	14489	11
Push Box	38869	1316	45	27631	6043	35	12212	11456	9
PushT	101080	1476	91	67656	8589	59	27777	18875	13
Tunnel2	279547	1260	165	222503	27549	12	53553	46322	13
Planar Hand	405740	4733	120	256654	25091	15	104097	57569	13

TABLE III: Comparison of constraint number, PSD cone number, and max PSD cone size for different tasks under SELF + NON, SELF + MAX, and SELF + MF settings. Throughout the table, only SOS relaxation are considered.

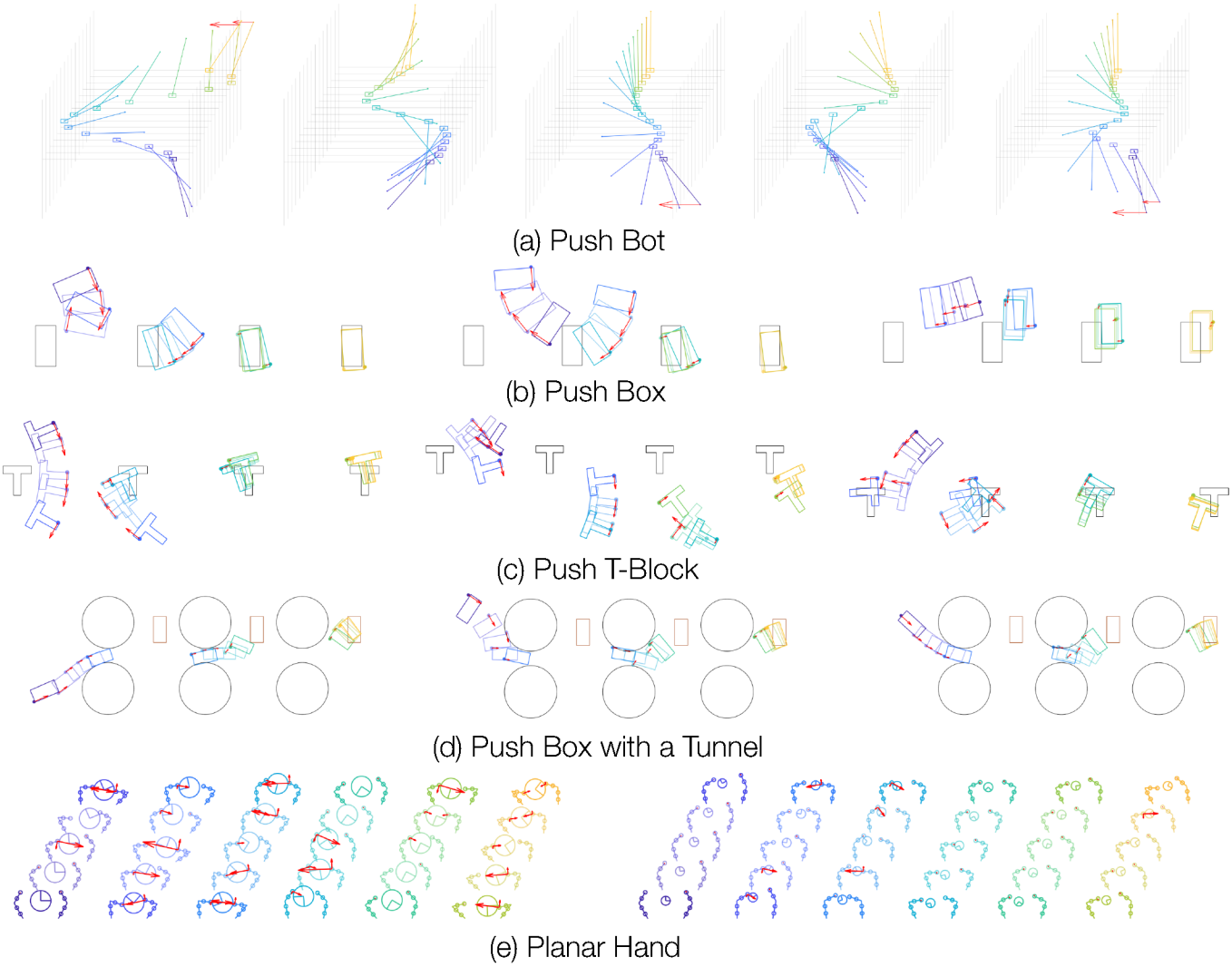


Fig. 10: More globally optimal trajectories from SPOT.

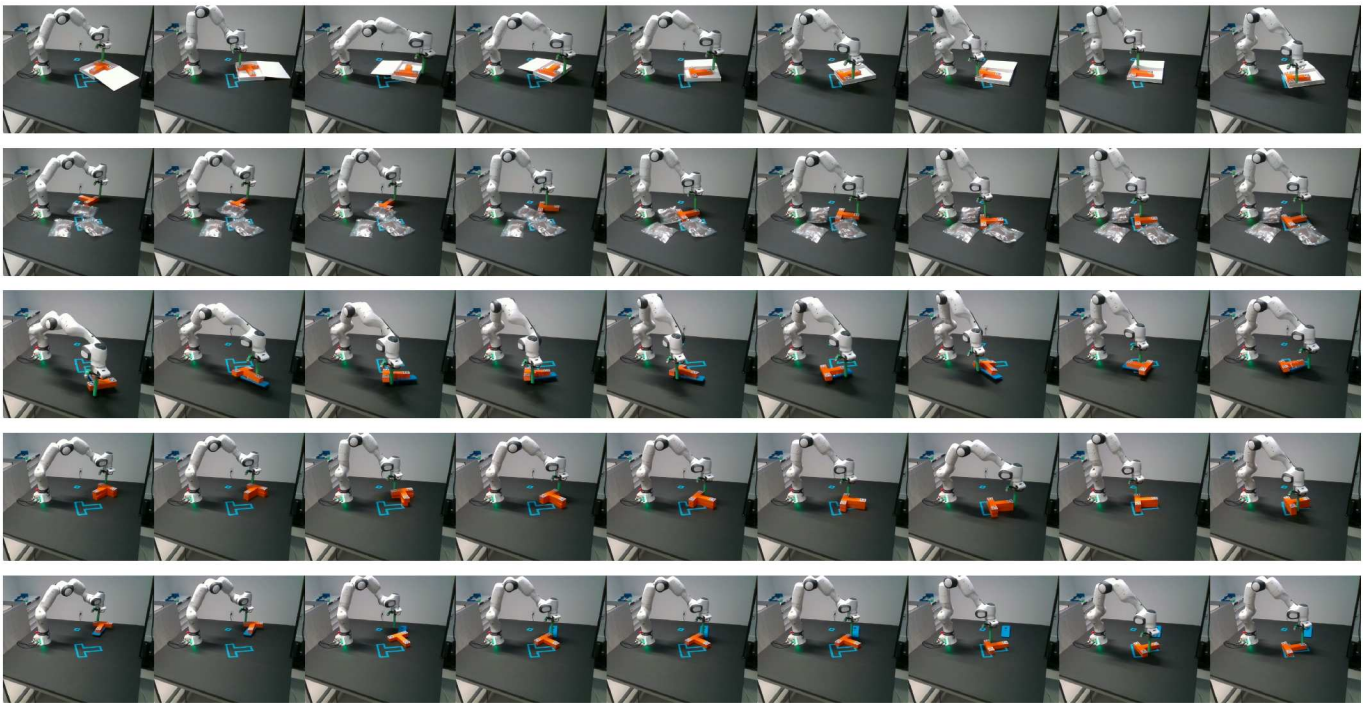


Fig. 11: More demonstrations from real-world "dirty" push-T tasks. From top to bottom: T-block (1) contained in a box; (2) sits on top of some irregular surfaces; (3) attached to some irregular objects; (4) stacked with another T-block; (5) put on some irregular objects.

Limitations and future work. Despite its efficiency, our approach faces challenges in scalability, suboptimality gaps, and real-time execution. Large-scale problems with many contact modes still pose computational bottlenecks, and automatic sparsity detection can lead to oversized relaxations. Future improvements can explore GPU-accelerated first-order SDP solvers [16], hybrid relaxations, and learning-based heuristics to further advance real-time and large-scale applications. Moreover, at present, sparsity patterns are crafted individually for each problem, but we anticipate that our open-source SPOT package—designed for rapid experimentation with alternative sparsity structures—will spur the discovery of many more.

REFERENCES

[1] Mosek ApS. Mosek optimization toolbox for matlab. *User's Guide and Reference Manual, Version*, 4(1), 2019. 9

[2] Alp Aydinoglu and Michael Posa. Real-time multi-contact model predictive control via admm. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3414–3421. IEEE, 2022. 2

[3] Alp Aydinoglu, Philip Sieg, Victor M Preciado, and Michael Posa. Stabilization of complementarity systems via contact-aware controllers. *IEEE Transactions on Robotics*, 38(3):1735–1754, 2021. 2

[4] Hans L. Bodlaender and Arie M.C.A. Koster. Treewidth computations i. upper bounds. *Information and Computation*, 208(3):259–275, 2010. ISSN 0890-5401. 4

[5] Jan Brüdigam, Stefan Sosnowski, Zachary Manchester, and Sandra Hirche. Variational integrators and graph-based solvers for multibody dynamics in maximal coordinates. *Multibody System Dynamics*, 61(3):381–414, 2024. 2

[6] Iordanis Chatzinikolaidis and Zhibin Li. Trajectory optimization of contact-rich motions using implicit differential dynamic programming. *IEEE Robotics and Automation Letters*, 6(2):2626–2633, 2021. 2

[7] Claire Chen, Preston Culbertson, Marion Lepert, Mac Schwager, and Jeannette Bohg. Trajectotree: Trajectory optimization meets tree search for planning multi-contact dexterous manipulation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8262–8268. IEEE, 2021. 2

[8] Xianyi Cheng, Eric Huang, Yifan Hou, and Matthew T Mason. Contact mode guided motion planning for quasidynamic dexterous manipulation in 3d. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2730–2736. IEEE, 2022. 2

[9] Jared Di Carlo, Patrick M Wensing, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1–9. IEEE, 2018. 1

[10] Yanran Ding, Chuanzheng Li, and Hae-Won Park. Kino-dynamic motion planning for multi-legged robot jumping

via mixed-integer convex program. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3998–4005. IEEE, 2020. 2

[11] Delbert Ray Fulkerson and Oliver Alfred Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15:835–855, 1965. 4

[12] Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*. Elsevier, 2004. 4

[13] Bernhard P Graesdal, Shao YC Chia, Tobia Marcucci, Savva Morozov, Alexandre Amice, Pablo A Parrilo, and Russ Tedrake. Towards tight convex relaxations for contact-rich manipulation. *arXiv preprint arXiv:2402.10312*, 2024. 2, 17

[14] Kazuo Hirai, Masato Hirose, Yuji Haikawa, and Toru Takenaka. The development of honda humanoid robot. In *Proceedings. 1998 IEEE international conference on robotics and automation (Cat. No. 98CH36146)*, volume 2, pages 1321–1326. IEEE, 1998. 1

[15] Lei Huang, Shucheng Kang, Jie Wang, and Heng Yang. Sparse polynomial optimization with unbounded sets. *arXiv preprint arXiv:2401.15837*, 2024. 2

[16] Shucheng Kang, Xiaoyang Xu, Jay Sarva, Ling Liang, and Heng Yang. Fast and certifiable trajectory optimization. *arXiv preprint arXiv:2406.05846*, 2024. 2, 3, 7, 10, 12

[17] Igor Klep, Janez Povh, and Jurij Volcic. Minimizer extraction in polynomial optimization is robust. *SIAM Journal on Optimization*, 28(4):3177–3207, 2018. 3, 10

[18] Frans Anton Koolen. *Balance control and locomotion planning for humanoid robots using nonlinear centroidal models*. PhD thesis, Massachusetts Institute of Technology, 2020. 2

[19] Jean B Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on optimization*, 11(3):796–817, 2001. 2

[20] Jean B Lasserre. Convergent sdp-relaxations in polynomial optimization with sparsity. *SIAM Journal on optimization*, 17(3):822–843, 2006. 6

[21] Jean B Lasserre. Convergent sdp-relaxations in polynomial optimization with sparsity. *SIAM Journal on optimization*, 17(3):822–843, 2006. 2, 7

[22] Simon Le Cleac'h, Taylor A Howell, Shuo Yang, Chi-Yen Lee, John Zhang, Arun Bishop, Mac Schwager, and Zachary Manchester. Fast contact-implicit model predictive control. *IEEE Transactions on Robotics*, 2024. 2

[23] Taeyoung Lee. *Computational geometric mechanics and control of rigid bodies*. PhD thesis, University of Michigan, 2008. 2, 16

[24] Kevin M Lynch, Hitoshi Maekawa, and Kazuo Tanie. Manipulation and active sensing by pushing using tactile feedback. In *IROS*, volume 1, pages 416–421, 1992. 18

[25] Victor Magron and Jie Wang. Tssos: a julia library to exploit sparsity for large-scale polynomial optimization. *arXiv preprint arXiv:2103.00915*, 2021. 2, 3

[26] Victor Magron and Jie Wang. *Sparse polynomial optimization: theory and practice*. World Scientific, 2023. 2

[27] Zachary Manchester and Scott Kuindersma. Variational contact-implicit trajectory optimization. In *Robotics Research: The 18th International Symposium ISRR*, pages 985–1000. Springer, 2020. 2

[28] Tobia Marcucci and Russ Tedrake. Warm start of mixed-integer programs for model predictive control of hybrid systems. *IEEE Transactions on Automatic Control*, 66 (6):2433–2448, 2020. 2

[29] Matthew T Mason. Mechanics and planning of manipulator pushing operations. *The International Journal of Robotics Research*, 5(3):53–71, 1986. 1, 17

[30] Carlos Mastalli, Rohan Budhiraja, Wolfgang Merkt, Guilhem Saurel, Bilal Hammoud, Maximilien Naveau, Justin Carpentier, Ludovic Righetti, Sethu Vijayakumar, and Nicolas Mansard. Crocoddyl: An efficient and versatile framework for multi-contact optimal control. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2536–2542. IEEE, 2020. 2

[31] Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (ToG)*, 31(4):1–8, 2012. 2

[32] Savva Morozov, Tobia Marcucci, Alexandre Amice, Bernhard Paus Graesdal, Rohan Bosworth, Pablo A Parrilo, and Russ Tedrake. Multi-query shortest-path problem in graphs of convex sets. *arXiv preprint arXiv:2409.19543*, 2024. 2

[33] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE international conference on robotics and automation*, pages 3400–3407. IEEE, 2011. 10

[34] 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. 2

[35] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014. 2, 20

[36] Donald J Rose, R Endre Tarjan, and George S Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on computing*, 5(2):266–283, 1976. 4

[37] 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. 2

[38] Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-limited differential dynamic programming. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1168–1175. IEEE, 2014. 2

[39] Sangli Teng, Ashkan Jasour, Ram Vasudevan, and Maani Ghaffari. Convex geometric motion planning on lie groups via moment relaxation. In *Robotics: Science and*

Systems, 2023. 2, 3, 16

[40] Sangli Teng, Ashkan Jasour, Ram Vasudevan, and Maani Ghaffari. Convex geometric motion planning of multi-body systems on lie groups via variational integrators and sparse moment relaxation. *The International Journal of Robotics Research*, page 02783649241296160, 2024. 2, 3

[41] Hayato Waki, Sunyoung Kim, Masakazu Kojima, and Masakazu Muramatsu. Sums of squares and semidefinite program relaxations for polynomial optimization problems with structured sparsity. *SIAM Journal on Optimization*, 17(1):218–242, 2006. 4

[42] Jie Wang. An introduction to polynomial optimization. 2023. 16

[43] Jie Wang, Victor Magron, and Jean-Bernard Lasserre. Tssos: A moment-sos hierarchy that exploits term sparsity. *SIAM Journal on optimization*, 31(1):30–58, 2021. 2, 6

[44] Jie Wang, Victor Magron, Jean B Lasserre, and Ngoc Hoang Anh Mai. Cs-tssos: Correlative and term sparsity for large-scale polynomial optimization. *ACM Transactions on Mathematical Software*, 48(4):1–26, 2022. 6, 16

[45] Albert Wu, Sadra Sadraddini, and Russ Tedrake. R3t: Rapidly-exploring random reachable set tree for optimal kinodynamic planning of nonlinear hybrid systems. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4245–4251. IEEE, 2020. 2

[46] Heng Yang. Semidefinite optimization and relaxation. Lecture notes: https://hankyang.seas.harvard.edu/Semidefinite/, 2024. 6

[47] Heng Yang and Luca Carlone. Certifiably optimal outlier-robust geometric perception: Semidefinite relaxations and scalable global optimization. *IEEE transactions on pattern analysis and machine intelligence*, 45(3):2816–2834, 2022. 2

[48] Lujie Yang, Tobia Marcucci, Pablo A Parrilo, and Russ Tedrake. A new semidefinite relaxation for linear and piecewise-affine optimal control with time scaling. 2

[49] William Yang and Michael Posa. Dynamic on-palm manipulation via controlled sliding. *arXiv preprint arXiv:2405.08731*, 2024. 2

[50] Mihalis Yannakakis. Computing the minimum fill-in is np-complete. *SIAM Journal on Algebraic Discrete Methods*, 2(1):77–79, 1981. 4

[51] K. Yunt. Optimal trajectory planning for structure-variant mechanical systems. In *International Workshop on Variable Structure Systems, 2006. VSS'06.*, pages 298–303, 2006. doi: 10.1109/VSS.2006.1644534. 2

[52] Kerim Yunt and Christoph Glocker. A combined continuation and penalty method for the determination of optimal hybrid mechanical trajectories. In *Iutam Symposium on Dynamics and Control of Nonlinear Systems with Uncertainty: Proceedings of the IUTAM Symposium held in Nanjing, China, September 18-22, 2006*, pages 187–196. Springer, 2007. 2

APPENDIX

A. MD and MF Chordal Extension

We present the MD chordal extension algorithm, which selects vertices based on minimum degree for elimination ordering.

Algorithm 1: MD Chordal Extension

Input: Graph $G(V, E)$ with n vertices
Output: Chordal graph G', elimination order π

$G' \leftarrow G$, $H \leftarrow G$, $R \leftarrow V$;
for $i = 1$ *to* n **do**
  Find vertex $v \in R$ with minimum degree in H;
  $\pi(v) \leftarrow i$;
  `// Unprocessed neighbors`;
  $N \leftarrow \{u \in R : (v, u) \in E(H)\}$;
  for *each pair* $(u, w) \in N \times N, u \neq w$ **do**
    if $(u, w) \notin E'$ **then**
      Add edge (u, w) to G' and H;
      Remove edges (v, u) in H, $\forall u \in N$;
    end
  end
  $R \leftarrow R \setminus \{v\}$;
end

We also present the MF chordal extension algorithm, which selects vertices causing minimum fill-in edges during the elimination process.

Algorithm 2: MF Chordal Extension

Input: Graph $G(V, E)$ with n vertices
Output: Chordal graph G', elimination order π

$G' \leftarrow G$, $H \leftarrow G$, $R \leftarrow V$;
for $i = 1$ *to* n **do**
  For each $v \in R$, compute fill-in cost: fill-in$(v) = |\{(u, w) : u, w \in N_H(v), u \neq w, (u, w) \notin E(H)\}|$;
  Find vertex $v \in R$ with minimum fill-in(v) in H;
  $\pi(v) \leftarrow i$;
  `// Unprocessed neighbors`;
  $N \leftarrow \{u \in R : (v, u) \in E(H)\}$;
  for *each pair* $(u, w) \in N \times N, u \neq w$ **do**
    if $(u, w) \notin E'$ **then**
      Add edge (u, w) to G' and H;
      Remove edges (v, u) in H, $\forall u \in N$;
    end
  end
  $R \leftarrow R \setminus \{v\}$;
end

B. Moment-SOS Hierarchy with CS-TS

Given a graph $G(V, E)$, define:

$$\mathbf{S}_G = \left\{ Q \in \mathbf{S}^{|V|} \;\middle|\; Q_{\beta,\gamma} = Q_{\gamma,\beta} = 0, \forall \beta \neq \gamma, (\beta, \gamma) \notin E \right\} \tag{A1}$$

where the rows and columns of $Q \in \mathbf{S}_G$ are indexed by V. Let Π_G be the projection form $\mathbf{S}^{|V|}$ to the subspace $\mathbf{S}_G$. Specifically, forall $Q \in \mathbf{S}^{|V|}$:

$$\Pi_G(Q) = \begin{cases} Q_{\beta,\gamma}, & \beta = \gamma \text{ or } (\beta, \gamma) \in E \\ 0, & \text{otherwise} \end{cases} \tag{A2}$$

If we further define $\Pi_G(\mathbf{S}_+^{|V|})$ as $\left\{\Pi_G(Q) \;\middle|\; Q \in \mathbf{S}_+^{|V|}\right\}$, the image of PSD cone with under projection $\Pi_G(\cdot)$, then the Moment-SOS Hierarchy with combined CS and TS can be concretely written as:

$$\begin{aligned}
\min \quad & L_y(f) && \text{(A3)}\\
\text{s.t.} \quad & L_y\left(M_d(g_j, I_l)\right) \circ B_{d,l,j}^g \in \Pi_{G'_{d,l,j}}(S_+^{|V_{d,l,j}|}), \\
& \forall j \in \{0\} \cup \mathcal{G}_l, l \in [p] && \text{(A4)}\\
& L_y\left(H_d(h_j, I_l)\right) \circ B_{d,l,j}^h = 0, \forall j \in \mathcal{H}_l, l \in [p] && \text{(A5)}\\
& y_{\mathbf{0}} = 1 && \text{(A6)}
\end{aligned}$$

The above procedure outlines the CS-TS Moment-SOS Hierarchy with a sparse order of $k = 1$. As shown in [44], one can further iteratively apply support extension and chordal extension within term sparsity. This process generates new sets $B_{d,l,j}^g$ and $B_{d,l,j}^h$, leading to a two-level hierarchy:

1) The outer level is governed by CS's relaxation order d.
2) The inner level is controlled by TS's sparse order k, which corresponds to the number of iterations used to generate new $B_{d,l,j}^g$ and $B_{d,l,j}^h$.

Define the optimal value of (A3) as ρ_d^k. The sequence $\{\rho_d^k\}_{k\geq 1}$ is monotonically non-decreasing and satisfies $\rho_d^k \leq \rho_d$ for all k. What's more:

Theorem A1 (Theorem 4.26 in [42]). *If we use maximal chordal extension* (i.e., *block closure) for term sparsity, $\rho_d^k \to \rho_d$ as $k \to \infty$.*

C. Polynomial Dynamics of Robotics Systems

1) Push Bot: Push bot is essentially cart-pole with soft wall. The configuration is shown in Figure 9 (a). a is cart's position, θ is pole's angle, k_1 and k_2 is soft wall's elastic modulus, λ_1 and λ_2 is two contact forces between two walls and pole's tip. The goal is to stabilize the cart-pole to $(a, \theta) = (0, \pi)$. From Newtonian mechanics:

$$(m_c + m_p)\frac{d^2}{dt^2}a + m_p\ell\frac{d^2}{dt^2}(\sin\theta) - (u + \lambda_1 - \lambda_2) = 0 \tag{A7}$$

$$\ell\frac{d^2}{dt^2}\theta + (\frac{d^2}{dt^2}a + \lambda_2 - \lambda_1)\cos\theta + g\sin\theta = 0 \tag{A8}$$

$$0 \leq \lambda_1 \perp \frac{\lambda_1}{k_1} + d_1 + (a + \ell\sin\theta) \geq 0 \tag{A9}$$

$$0 \leq \lambda_2 \perp \frac{\lambda_2}{k_2} + d_2 - (a + \ell\sin\theta) \geq 0 \tag{A10}$$

Use the same techniques introduced in [39], we discretize (A7) on the lie group [23] to yield polynomial dynamics:

$$\begin{aligned}
&(m_c + m_p)\cdot\frac{a_{k+1} - 2a_k + a_{k-1}}{\Delta t^2} \\
&+ (m_p\ell)\cdot\frac{r_{s,k+1} - 2r_{s,k} + r_{s,k-1}}{\Delta t^2} - (u_k + \lambda_{1,k} - \lambda_{2,k}) = 0 && \text{(A11)}\\
&\ell\cdot\frac{f_{s,k} - f_{s,k-1}}{\Delta t^2} \\
&+ \left(\frac{a_{k+1} - 2a_k + a_{k-1}}{\Delta t^2} + (\lambda_{2,k} - \lambda_{1,k})\right)\cdot r_{c,k} + g\cdot r_{s,k} = 0 && \text{(A12)}\\
&0 \leq \lambda_{1,k} \perp \left(\frac{\lambda_{1,k}}{k_1} + d_1 + a_k + \ell r_{s,k}\right) \geq 0 && \text{(A13)}\\
&0 \leq \lambda_{2,k} \perp \left(\frac{\lambda_{2,k}}{k_2} + d_2 - a_k - \ell r_{s,k}\right) \geq 0 && \text{(A14)}\\
&r_{c,k} = r_{c,k-1}f_{c,k-1} - r_{s,k-1}f_{s,k-1} && \text{(A15)}\\
&r_{s,k} = r_{s,k-1}f_{c,k-1} + r_{c,k-1}f_{s,k-1} && \text{(A16)}\\
&r_{c,k}^2 + r_{s,k}^2 = 1 && \text{(A17)}\\
&f_{c,k}^2 + f_{s,k}^2 = 1 && \text{(A18)}
\end{aligned}$$

The loss function is designed as:

$$\begin{aligned}\text{loss} = &\sum_{k=0}^{N-1} c_a \cdot a_k^2 + c_{a,f} \cdot a_N^2 \\ &+ \sum_{k=0}^{N-1} c_\theta \cdot \left\{(r_{c,k}+1)^2 + r_{s,k}^2\right\} + c_{\theta,f} \cdot \left\{(r_{c,N}+1)^2 + r_{s,N}^2\right\} \\ &+ \sum_{k=0}^{N-1} c_{\dot{\theta}} \cdot \left\{(f_{c,k}-1)^2 + f_{s,k}^2\right\} + c_{\dot{\theta},f} \cdot \left\{(f_{c,N}-1)^2 + f_{s,N}^2\right\}\end{aligned} \tag{A19}$$

2) Push Box: Consider a simple pusher-slider system illustrated in Figure 9 (b). Our goal is to push the box from one configuration ((s_x, s_y, θ)) to another. From [13], given (1) the pusher's position (p_x, p_y) and the contact force (F_x, F_y) in the slider frame; (2) the slider's position (s_x, s_y) and angle θ in the world frame, the quasi-static dynamics of the slider can be written as:

$$\frac{d}{dt}s_x = \frac{1}{\mu_1 mg} \cdot (\cos\theta F_x - \sin\theta F_y) \tag{A20}$$

$$\frac{d}{dt}s_y = \frac{1}{\mu_1 mg} \cdot (\sin\theta F_x + \cos\theta F_y) \tag{A21}$$

$$\frac{d}{dt}\theta = \frac{1}{cr \cdot \mu_1 mg} \cdot (-p_y F_x + p_x F_y) \tag{A22}$$

where μ_1 is the friction coefficient between the slider and table. $c \in (0, 1)$ is the integration constant that depends on the slider geometry. r is a characteristic distance, typically chosen as the max distance between a contact point and origin of slider frame [29]. Use the dimensionless trick:

$$F_x \leftarrow \frac{1}{\mu_1 mg} \cdot F_x, \;\; F_y \leftarrow \frac{1}{\mu_1 mg} \cdot F_y \tag{A23}$$

Discretize over the lie group:

$$s_{x,k} = s_{x,k-1} + \Delta t \cdot (r_{c,k-1}F_{x,k-1} - r_{s,k-1}F_{y,k-1}) \tag{A24}$$

$$s_{y,k} = s_{y,k-1} + \Delta t \cdot (r_{s,k-1}F_{x,k-1} + r_{c,k-1}F_{y,k-1}) \tag{A25}$$

$$f_{s,k-1} = \Delta t \cdot \frac{1}{cr} \cdot (-p_{y,k-1}F_{x,k-1} + p_{x,k-1}F_{y,k-1}) \tag{A26}$$

Here, the lie-group constraints (A15) - (A18) are omitted for simplicity. Since when pusher has no contact with the slider, slider remains still and pusher's planning task is trivial, we only fucus on the time steps when pusher and slider have contact. As illustrated in Figure 9 (b), we assign for modes λ_i's ($i = 1, 2, 3, 4$ for box's four sides):

$$\lambda_i(1 - \lambda_i) = 0, \;\; i = 1, 2, 3, 4 \tag{A27}$$

$$\sum_{i=1}^{4} \lambda_i^2 = 1 \tag{A28}$$

In each mode, the relationship between F_x, F_y, p_x, p_y is different. For example, in mode 1:

$$\lambda_1 \cdot (a^2 - p_x^2) \geq 0 \tag{A29}$$

$$\lambda_1 \cdot (p_y - b) = 0 \tag{A30}$$

$$\lambda_1 \cdot (-F_y) \geq 0 \tag{A31}$$

where for modelling simplicity, we assume the pushing direction will always be normal to the contact surface. Similar contact constraints can be assigned to mode 2 - 4. Simplify them:

$$(\lambda_1 + \lambda_3) \cdot (a^2 - p_x^2) + (\lambda_2 + \lambda_4) \cdot (b^2 - p_y^2) \geq 0 \tag{A32}$$

$$\lambda_1 \cdot (p_y - b) + \lambda_2 \cdot (p_x - a) + \lambda_3 \cdot (p_y + b) + \lambda_4 \cdot (p_x + a) = 0 \tag{A33}$$

$$(-\lambda_1 + \lambda_3) \cdot F_y + (-\lambda_2 + \lambda_4) \cdot F_x \geq 0 \tag{A34}$$

$$(\lambda_1 + \lambda_3) \cdot F_x + (\lambda_2 + \lambda_4) \cdot F_y = 0 \tag{A35}$$

The loss function is in the same spirit as (A19). We omit it here.

3) Push T-block: Now we consider a more complicated pushing task: push a T-block, as illustrated in Figure 9 (c). Unlike 4 modes in the box setting, now we have 8 modes to assign. From [24], when μ_1 is uniformly distributed between the slider and the table, the friction center coincides with the projection of the center of mass (CM) to the table. Thus, we set the origin of the Slider frame to T-block's CM for convenience. d_c from Figure 9 (c) can be derived as:

$$d_c = \frac{3 \times 1.5 + 4 \times 3.5}{3+4} = \frac{37}{14} \tag{A36}$$

There are eight key points in the T-block:

$$x_1 = -2l,\ x_2 = -0.5l,\ x_3 = 0.5l,\ x_4 = 2l \tag{A37}$$

$$y_1 = -d_c l,\ y_2 = (3-d_c)l,\ y_3 = (4-d_c)l \tag{A38}$$

Connect each mode with geometric and dynamical constraints:

$$\lambda_1 \Longrightarrow p_y - y_3 = 0, p_x - x_1 \geq 0, x_4 - p_x \geq 0, -F_y \geq 0, F_x = 0 \tag{A39}$$

$$\lambda_2 \Longrightarrow p_x - x_4 = 0, p_y - y_2 \geq 0, y_3 - p_y \geq 0, -F_x \geq 0, F_y = 0 \tag{A40}$$

$$\lambda_3 \Longrightarrow p_y - y_2 = 0, p_x - x_3 \geq 0, x_4 - p_x \geq 0, F_y \geq 0, F_x = 0 \tag{A41}$$

$$\lambda_4 \Longrightarrow p_x - x_3 = 0, p_y - y_1 \geq 0, y_2 - p_y \geq 0, -F_x \geq 0, F_y = 0 \tag{A42}$$

$$\lambda_5 \Longrightarrow p_y - y_1 = 0, p_x - x_2 \geq 0, x_3 - p_x \geq 0, F_y \geq 0, F_x = 0 \tag{A43}$$

$$\lambda_6 \Longrightarrow p_x - x_2 = 0, p_y - y_1 \geq 0, y_2 - p_y \geq 0, F_x \geq 0, F_y = 0 \tag{A44}$$

$$\lambda_7 \Longrightarrow p_y - y_2 = 0, p_x - x_1 \geq 0, x_2 - p_x \geq 0, F_y \geq 0, F_x = 0 \tag{A45}$$

$$\lambda_8 \Longrightarrow p_x - x_1 = 0, p_y - y_2 \geq 0, y_3 - p_y \geq 0, F_x \geq 0, F_y = 0 \tag{A46}$$

Other things are the same as the Push Box case.

4) Push Box with a Tunnel: Everything is the same as Push Box setting, except that the box needs to avoid two circle obstacles this time. To model the collision avoidance constraints, we approximate the box as a union of two circles, as shown in Figure 9 (d). For each obstacle-slider circle pair, the non-collision constraint is:

$$(x_o - x_s)^2 + (y_o - y_s)^2 \geq (r_o + r_s)^2 \tag{A47}$$

where (x_o, y_o, r_o) (resp. (x_s, y_s, r_s)) represents center and radius of obstacle's (resp. slider's) center.

5) Planar Hand: The geometric and mechanical information of the Planar Hand system is illustrated in Figure 9 (e). The goal is to rotate the circle disk 360° with planar hand's two finger tips, while minimize the translation of the disk's center of mass.

Kinematics of the fingers. For two fingers, we use position control. For example, for the right finger:

$$x_r = L_d \cdot \cos\theta_{rd} + L_u \cdot \cos\theta_{ru} + \frac{H}{2} \tag{A48}$$

$$y_r = L_d \cdot \sin\theta_{rd} + L_u \cdot \sin\theta_{ru} \tag{A49}$$

$$v_{x,r} = -L_d \cdot \sin\theta_{rd} \cdot \dot{\theta}_{rd} - L_u \cdot \sin\theta_{ru} \cdot \dot{\theta}_{ru} \tag{A50}$$

$$v_{y,r} = L_d \cdot \cos\theta_{rd} \cdot \dot{\theta}_{rd} + L_u \cdot \cos\theta_{ru} \cdot \dot{\theta}_{ru} \tag{A51}$$

where "r" and "l" represent "right" and "left" finger, while "u" and "d" represent "upper" and "down" link. Since

$$f_s = \sin(\dot{\theta} \cdot \Delta t) \Longrightarrow \dot{\theta} \approx \frac{f_s}{\Delta t} \tag{A52}$$

Then,

$$x_{r,k} = L_d \cdot r_{c,rd,k} + L_u \cdot r_{c,ru,k} + \frac{H}{2} \tag{A53}$$

$$y_{r,k} = L_d \cdot r_{s,rd,k} + L_u \cdot r_{s,ru,k} \tag{A54}$$

$$v_{x,r,k} = -\frac{L_d}{\Delta t} \cdot r_{s,rd,k} \cdot f_{s,rd,k} - \frac{L_u}{\Delta t} \cdot r_{s,ru,k} \cdot f_{s,ru,k} \tag{A55}$$

$$v_{y,r,k} = \frac{L_d}{\Delta t} \cdot r_{c,rd,k} \cdot f_{s,rd,k} + \frac{L_u}{\Delta t} \cdot r_{c,ru,k} \cdot f_{s,ru,k} \tag{A56}$$

$$r_{c,rd,k+1} = r_{c,rd,k} \cdot f_{c,rd,k} - r_{s,rd,k} \cdot f_{s,rd,k} \tag{A57}$$

$$r_{s,rd,k+1} = r_{c,rd,k} \cdot f_{s,rd,k} + r_{s,rd,k} \cdot f_{c,rd,k} \tag{A58}$$

$$r_{c,ru,k+1} = r_{c,ru,k} \cdot f_{c,ru,k} - r_{s,ru,k} \cdot f_{s,ru,k} \tag{A59}$$

$$r_{s,ru,k+1} = r_{c,ru,k} \cdot f_{s,ru,k} + r_{s,ru,k} \cdot f_{c,ru,k} \tag{A60}$$

Self collision avoidance. For each finger, there are two types of self collisions: (1) the first circle with the ground; (2) the second and the third circle. For the first type:

$$\theta_{ld} \geq \arcsin(\frac{r}{l+r}),\ \pi - \theta_{ld} \geq \arcsin(\frac{r}{l+r}) \tag{A61}$$

$$\theta_{rd} \geq \arcsin(\frac{r}{l+r}),\ \pi - \theta_{rd} \geq \arcsin(\frac{r}{l+r}) \tag{A62}$$

For the second type:

$$\pi - \theta_{ld} + \theta_{lu} \geq 2 \cdot \arcsin(\frac{r}{l+r}) \tag{A63}$$

$$2\pi - (\pi - \theta_{rd} + \theta_{ru}) \geq 2 \cdot \arcsin(\frac{r}{l+r}) \tag{A64}$$

Also, like a human finger, we assume the upper link won't "turn outward":

$$\theta_{ld} - \theta_{lu} \geq 0 \tag{A65}$$

$$\theta_{ru} - \theta_{rd} \geq 0 \tag{A66}$$

Now denote θ_0 as $\arcsin(\frac{r}{l+r})$. Writing the constraints as polynomials:

$$r_{s,ld,k} \geq \sin\theta_0 \tag{A67}$$

$$r_{s,rd,k} \geq \sin\theta_0 \tag{A68}$$

and

$$\sin(\theta_{ld,k} - \theta_{lu,k}) = r_{s,ld,k} \cdot r_{c,lu,k} - r_{c,ld,k} \cdot r_{s,lu,k} \geq 0 \tag{A69}$$

$$\cos(\theta_{ld,k} - \theta_{lu,k}) = r_{c,ld,k} \cdot r_{c,lu,k} + r_{s,ld,k} \cdot r_{s,lu,k} \geq -\cos(2\theta_0) \tag{A70}$$

$$\sin(\theta_{ru,k} - \theta_{rd,k}) = r_{s,ru,k} \cdot r_{c,rd,k} - r_{c,ru,k} \cdot r_{s,rd,k} \geq 0 \tag{A71}$$

$$\cos(\theta_{ru,k} - \theta_{rd,k}) = r_{c,ru,k} \cdot r_{c,rd,k} + r_{s,ru,k} \cdot r_{s,rd,k} \geq -\cos(2\theta_0) \tag{A72}$$

Contact model. Now we deal with the contact between the fingers and the disk. Without loss of generality, we consider the right finger. Denote d_r as:

$$d_r^2 = (x_r - x)^2 + (y_r - y)^2 \tag{A73}$$

$$d_r \geq R + r \tag{A74}$$

where (x, y) is the position of the disk's center. When contact happens, $d_r = R + r$. In this case, denote (v_x, v_y, w) as the translational and angular velocity of the disk, $(\lambda_{t,r}, \lambda_{n,r})$ as the tangential and normal force exerted on the disk by the tip of the finger, and $v_{rel,r}$ as the relative tangential velocity of finger's tip compared to the disk. The physical quantities are illustrated in Figure 9 (e)'s upper right position. Denote the angle η_r as:

$$\cos\eta_r = \frac{x_r - x}{d_r} \tag{A75}$$

$$\sin\eta_r = \frac{y_r - y}{d_r} \tag{A76}$$

Then,

$$\begin{aligned} v_{rel,r} \approx &- v_{x,r} \cdot \sin\eta_r + v_{y,r} \cdot \cos\eta_r \\ &- (-v_x \cdot \sin\eta_r + v_y \cdot \cos\eta_r + \omega R) \end{aligned} \tag{A77}$$

Here we do one approximation for finger tip's tangential velocity, by ignoring the tip's size. One thing we should notice is, λ_n will always point inside the disk:

$$-\lambda_{n,r} \geq 0 \tag{A78}$$

By the Coulomb's law:

$$\lambda_{t,r} \begin{cases} = \mu \cdot (-\lambda_{n,r}),\ v_{rel,r} > 0 \\ = -\mu \cdot (-\lambda_{n,r}),\ v_{rel,r} < 0 \\ \in [-\mu \cdot (-\lambda_{n,r}), \mu \cdot (-\lambda_{n,r})],\ v_{rel,r} = 0 \end{cases} \tag{A79}$$

Unlike [35] who introduced two auxiliary variables to app:pdress the above model as nine quadratic polynomials, we advocate for a more concise representation:

$$\mu^2 \cdot \lambda_{n,r}^2 - \lambda_{t,r}^2 \geq 0 \tag{A80}$$
$$v_{rel,r} \cdot (\mu^2 \cdot \lambda_{n,r}^2 - \lambda_{t,r}^2) = 0 \tag{A81}$$
$$v_{rel,r} \cdot \lambda_{t,r} \geq 0 \tag{A82}$$

Combine everything together:

$$d_{r,k}^2 = (x_{r,k} - x_k)^2 + (y_{r,k} - y_k)^2 \tag{A83}$$
$$d_{r,k} \geq R + r \tag{A84}$$
$$v_{rel,r,k} = -(v_{x,r,k} - v_{x,k}) \cdot \frac{y_{r,k} - y_k}{R + r} + (v_{y,r,k} - v_{y,k}) \cdot \frac{x_{r,k} - x_k}{R + r} - R \cdot \frac{f_{s,k}}{\Delta t} \tag{A85}$$
$$(d_{r,k} - R - r) \cdot \lambda_{n,r,k} = 0 \tag{A86}$$
$$-\lambda_{n,r,k} \geq 0 \tag{A87}$$
$$\mu^2 \cdot \lambda_{n,r,k}^2 - \lambda_{t,r,k}^2 \geq 0 \tag{A88}$$
$$v_{rel,r,k} \cdot (\mu^2 \cdot \lambda_{n,r,k}^2 - \lambda_{t,r,k}^2) = 0 \tag{A89}$$
$$v_{rel,r,k} \cdot \lambda_{t,r,k} \geq 0 \tag{A90}$$

Dynamics of disk. Consider the quasi-static dynamics similar to (A20):

$$\frac{d}{dt}x = \lambda_{n,r} \cdot \cos\eta_r - \lambda_{t,r} \cdot \sin\eta_r + \lambda_{n,l} \cdot \cos\eta_l - \lambda_{t,r} \cdot \sin\eta_l \tag{A91}$$
$$\frac{d}{dt}y = \lambda_{n,r} \cdot \sin\eta_r + \lambda_{t,r} \cdot \cos\eta_r + \lambda_{n,l} \cdot \sin\eta_l + \lambda_{t,r} \cdot \cos\eta_l \tag{A92}$$
$$\frac{d}{dt}\alpha = \frac{1}{c \cdot R} \cdot (\lambda_{t,r} + \lambda_{t,l}) \tag{A93}$$

Write them as polynomials:

$$\frac{1}{\Delta t}(x_{k+1} - x_k) = \lambda_{n,r,k} \cdot \frac{x_{r,k} - x_k}{R + r} - \lambda_{t,r,k} \cdot \frac{y_{r,k} - y_k}{R + r} + \lambda_{n,l,k} \cdot \frac{x_{l,k} - x_k}{R + r} - \lambda_{t,l,k} \cdot \frac{y_{l,k} - y_k}{R + r} \tag{A94}$$
$$\frac{1}{\Delta t}(y_{k+1} - y_k) = \lambda_{n,r,k} \cdot \frac{y_{r,k} - y_k}{R + r} + \lambda_{t,r,k} \cdot \frac{x_{r,k} - x_k}{R + r} + \lambda_{n,l,k} \cdot \frac{y_{l,k} - y_k}{R + r} + \lambda_{t,l,k} \cdot \frac{x_{l,k} - x_k}{R + r} \tag{A95}$$
$$f_{s,k} = \frac{\Delta t}{c \cdot R} \cdot (\lambda_{t,r,k} + \lambda_{t,l,k}) \tag{A96}$$
$$r_{c,k+1} = r_{c,k} \cdot f_{c,k} - r_{s,k} \cdot f_{s,k} \tag{A97}$$
$$r_{s,k+1} = r_{c,k} \cdot f_{s,k} + r_{s,k} \cdot f_{c,k} \tag{A98}$$
$$f_{s,k}^2 + f_{c,k}^2 = 1 \tag{A99}$$

Collision avoidance. Consider the right finger. Since the finger tip has already been considered in the contact model, we only need to consider collision avoidance between the object and the remaining three circles attached to the finger. The three circles' positions are:

$$\left((l + r)\cos\theta_{rd} + \frac{H}{2}, (l + r)\sin\theta_{rd}\right) \tag{A100}$$
$$\left((2l + 3r)\cos\theta_{rd} + \frac{H}{2}, (2l + 3r)\sin\theta_{rd}\right) \tag{A101}$$
$$\left(L_d\cos\theta_{rd} + (l + r)\cos\theta_{ru} + \frac{H}{2}, L_d\sin\theta_{rd} + (l + r)\sin\theta_{ru}\right) \tag{A102}$$

Therefore, the constraints are:

$$\left((l+r)\cdot r_{c,rd,k}-x_k+\frac{H}{2}\right)^2+\left((l+r)\cdot r_{s,rd,k}-y_k\right)^2\geq(R+r)^2 \tag{A103}$$

$$\left((2l+3r)\cdot r_{c,rd,k}-x_k+\frac{H}{2}\right)^2+\left((2l+3r)\cdot r_{s,rd,k}-y_k\right)^2\geq(R+r)^2 \tag{A104}$$

$$\left(L_d\cdot r_{c,rd,k}+(l+r)\cdot r_{c,ru,k}-x_k+\frac{H}{2}\right)^2+\left(L_d\cdot r_{s,rd,k}+(l+r)\cdot r_{s,ru,k}-y_k\right)^2\geq(R+r)^2 \tag{A105}$$

D. Self-Defined Variable Cliques for Planar hand

The self-defined variable cliques for PLanar Hand are defined as:

$$\{x_k,y_k,r_{c,k},r_{s,k},r_{c,ld,k},r_{s,ld,k},r_{c,lu,k},r_{s,lu,k},r_{c,rd,k},r_{s,rd,k},r_{c,ru,k},r_{s,ru,k}\} \tag{A106a}$$

$$\{x_k,y_k,r_{c,ld,k},r_{s,ld,k},r_{c,lu,k},r_{s,lu,k},x_{l,k},y_{l,k},d_{l,k}\} \tag{A106b}$$

$$\{r_{c,ld,k},r_{s,ld,k},r_{c,lu,k},r_{s,lu,k},x_{l,k},y_{l,k}\} \tag{A106c}$$

$$\{r_{c,ld,k},r_{s,ld,k},f_{c,ld,k},f_{s,ld,k},r_{c,lu,k},r_{s,lu,k},f_{c,lu,k},f_{s,lu,k},v_{x,l,k},v_{y,l,k}\} \tag{A106d}$$

$$\{x_k,y_k,r_{c,rd,k},r_{s,rd,k},r_{c,ru,k},r_{s,ru,k},x_{r,k},y_{r,k},d_{r,k}\} \tag{A106e}$$

$$\{r_{c,k},r_{c,k+1},r_{s,k},r_{s,k+1},f_{c,k},f_{s,k}\} \tag{A106f}$$

$$\{r_{c,rd,k},r_{s,rd,k},r_{c,ru,k},r_{s,ru,k},x_{r,k},y_{r,k}\} \tag{A106g}$$

$$\{r_{c,rd,k},r_{s,rd,k},f_{c,rd,k},f_{s,rd,k},r_{c,ru,k},r_{s,ru,k},f_{c,ru,k},f_{s,ru,k},v_{x,r,k},v_{y,r,k}\} \tag{A106h}$$

$$\{x_k,x_{k+1},y_k,y_{k+1},f_{c,k},f_{s,k},x_{l,k},y_{l,k},v_{x,l,k},v_{y,l,k},d_{l,k},v_{rel,l,k},\lambda_{n,l,k},\lambda_{t,l,k}\} \tag{A106i}$$

$$\{r_{c,ld,k},r_{c,ld,k+1},r_{s,ld,k},r_{s,ld,k+1},f_{c,ld,k},f_{s,ld,k},r_{c,lu,k},r_{c,lu,k+1},r_{s,lu,k},r_{s,lu,k+1},f_{c,lu,k},f_{s,lu,k}\} \tag{A106j}$$

$$\{x_k,x_{k+1},y_k,y_{k+1},f_{c,k},f_{s,k},x_{l,k},y_{l,k},x_{r,k},y_{r,k},\lambda_{n,l,k},\lambda_{t,l,k},\lambda_{n,r,k},\lambda_{t,r,k}\} \tag{A106k}$$

$$\{x_k,x_{k+1},y_k,y_{k+1},f_{c,k},f_{s,k},x_{r,k},y_{r,k},v_{x,r,k},v_{y,r,k},d_{r,k},v_{rel,r,k},\lambda_{n,r,k},\lambda_{t,r,k}\} \tag{A106l}$$

$$\{r_{c,rd,k},r_{c,rd,k+1},r_{s,rd,k},r_{s,rd,k+1},f_{c,rd,k},f_{s,rd,k},r_{c,ru,k},r_{c,ru,k+1},r_{s,ru,k},r_{s,ru,k+1},f_{c,ru,k},f_{s,ru,k}\} \tag{A106m}$$

$$\{x_{k+1},y_{k+1},r_{c,ld,k+1},r_{s,ld,k+1},r_{c,lu,k+1},r_{s,lu,k+1},r_{c,rd,k+1},r_{s,rd,k+1},r_{c,ru,k+1},r_{s,ru,k+1}\} \tag{A106n}$$