WHAT MAKES A GOOD SYMBOL FOR HYBRID DISCRETE–CONTINUOUS PLANNING?

Anonymous authors
Paper under double-blind review

ABSTRACT

The physical world has continuous state and action spaces, yet robot planning often benefits from hybrid discrete-continuous formulations that explicitly expose combinatorial structure, such as contact modes or discrete primitive actions. Existing methods typically rely on human-defined symbol sets or symbols derived from language supervision, without asking what makes a symbol computationally *useful*. In this paper, we take an inference-algorithm–centric view to define *usefulness*: a good symbol should expose the discrete commitments that simplify solving the original planning problem. We first introduce the concept of *solver-sufficient symbolization* by studying finite-horizon piecewise-affine systems, for which solver sufficiency can be precisely characterized, and develop a counterexample-guided algorithm for symbol invention. We then extend this principle to invent symbols for optimization objectives parameterized by general neural networks as energy functions, showing that solver-sufficient symbols can nevertheless be induced from their continuous optimization landscapes. Experiments on simulated and real-robot tasks show that the resulting symbols reduce solving cost while preserving solution quality, while also improving interpretability and steerability. Furthermore, our theoretical and algorithmic insights help explain why certain human-defined symbols are computationally useful.

1 INTRODUCTION

The physical world has continuous state and action spaces, yet planning often benefits from hybrid discrete–continuous formulations that explicitly expose combinatorial structure, such as contact modes in manipulation (e.g., *fixed and sliding contacts*) (Posa et al., 2016; Cheng et al., 2022) and symbolic relational states (e.g., *holding*, *in*) (Garrett et al., 2021). From a resource-rational perspective in cognitive science, a useful symbol can expose key commitments and abstract away details to support efficient planning under bounded computation (Lieder & Griffiths, 2020; Ho et al., 2022): When a person wants to go to the office with a bag, the motion of the person and the bag must initially be considered jointly. Once "the bag is being carried" is established as a symbolic commitment, the bag's motion can be abstracted away when planning the person's path.

While humans can invent symbols to reduce the complexity of decision-making, deep learning–based solutions to decision-making often choose to encode the full decision landscape within a single continuous-space model (Chi et al., 2025; Janner et al., 2022). They remove the need to declare discrete modes and dependencies *a priori*, but fundamentally cannot remove the underlying combinatorial structure of the problem. For example, for neural-network–parameterized dynamics models, basins and barriers appear in the optimization landscape. This makes planning or optimization hard as tasks and horizons grow (Florence et al., 2022). To this end, researchers have considered exposing symbol commitments explicitly as state variables, such as the object relation of person-carrying-bag, to break much of the global coupling in planning. The symbol invention problem studies inducing these symbols from data, but typically from predefined grammars and labels (Silver et al., 2023; Athalye et al., 2025). Such symbols are often interpretable and descriptive, yet their construction typically relies on heuristics about the problem structure, such as sparsity and locality. As a result, they may hinder rather than simplify planning. In this paper, we address this fundamental question:

What makes a symbol **useful** *in hybrid discrete-continuous planning?*

Our definition of a useful symbol comes from the perspective of inference-time solving. We consider a general family of hybrid discrete-continuous planning algorithms based on *search-then-optimize*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

(Geoffrion, 1972; Dechter, 1990): symbols partition the entire decision space into conditioned subproblems, within which continuous-space optimization is performed. Let $x \in \mathcal{X}$ denote all decision variables and h the planning objective. Our goal is to solve the hybrid optimization problem:

$$\mathcal{P}_{\text{joint}} : \min_{x \in \mathcal{X}} h(x). \tag{1}$$

Let $\Phi : \mathcal{X} \to \Psi$ denote a symbolization map, where each $\psi \in \Psi$ is a symbolic assignment. Fixing ψ induces the conditioned problem $\mathcal{P}^{\psi}_{\text{joint}} : \min_{x \in \Phi^{-1}(\psi)} h(x|\psi)$. This yields the decomposition:

$$\underbrace{\text{Solve}(\mathcal{P}_{\text{joint}})}_{\textit{Joint Solving}} = \underbrace{\min\nolimits_{\psi \in \Psi}}_{\textit{Symbolic Search}} \underbrace{\text{Solve}(\mathcal{P}^{\psi}_{\text{joint}})}_{\textit{Grounded Solving}}. \tag{2}$$

whose naïve solving complexity is $\mathcal{O}(2^{|\Psi|} \cdot \text{SolveTime}(\mathcal{P}^{\psi}_{\text{joint}}))$. In this paper, we focus on *solver-sufficient* symbols: they preserve the global optimum in Eq. (2) while reducing the overall complexity of conditioned solve $\text{Solve}(\mathcal{P}^{\psi}_{\text{joint}})$. In a planning context, such symbols should decouple temporal and spatial dependencies that would otherwise require joint reasoning, allowing each grounded solve to involve only the local variables relevant to the selected symbolic commitment.

To put these in practice, we begin with simple piecewise affine (PWA) systems (Rodrigues & How, 2003), where the solver sufficiency and complexity can be studied exactly using TU-dimension tools (Hoffman & Kruskal, 2010). We then extend this perspective to general settings involving neural networks, focusing on energy-based formulations which include frameworks such as diffusion and flow-matching policies (Chi et al., 2025; Lipman et al., 2022). In both settings, we develop algorithms for both symbol invention and symbol-guided planning. Our contributions are threefold:

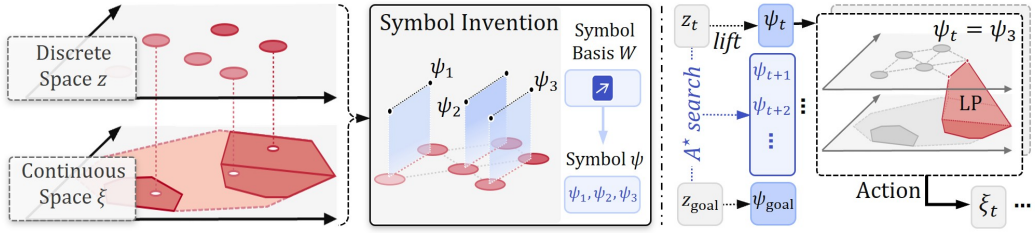
- We introduce solver-sufficient symbolization, in which symbolic commitments preserve optimality while reducing grounded solving cost. This formalizes the symbolic invention problem as an optimization problem rather than a purely descriptive interface.
- We provide a mathematical foundation for analyzing why certain human-defined concepts are useful in robotic planning, and characterize task hardness by the amount of irreducible discrete structure that must remain explicit for faithful solving.
- We develop and evaluate a symbol invention and hybrid planning framework that retains key advantages of symbolic planning, including interpretability and behavior steerability.

Overview of related work. In search-then-optimize planning (Geoffrion, 1972; Audet et al., 2026), the key step is to choose a conditioned subproblem, such as mode assignments in MILPs (Marcucci & Tedrake, 2020; Cauligi et al., 2021) and energy-based model inference with multi-seed initialization (Florence et al., 2022; Chi et al., 2025). This choice is hard because decisions are coupled. Researchers have formalized such coupling in AI planning problems (Chen & Giménez, 2007; Mao et al., 2023). Hybrid discrete-continuous planners aim to address this by introducing symbols (Konidaris et al., 2018; Garrett et al., 2020), but their symbols come from labels or simple algebraic combinations of state variables (Silver et al., 2023; Athalye et al., 2025). We instead derive symbols from the solver itself: the commitments required before make the conditioned problem cheap to solve, and the commitment number measures the problem's coupling and irreducible combinatorial hardness. We include a more comprehensive discussion of related work in Appendix A.

2 Solver Sufficiency in PWA Systems

We first build the intuition for solver-sufficient symbolization in piecewise affine (PWA) systems. In PWA, a finite-horizon decision-making problem is formulated as a mixed-integer linear program (MILP). Here, discrete variables encode *modes*, such as contacts, while continuous variables describe the continuous states and actions governed by mode-dependent affine dynamics. Such formulations are widely used in contact-rich manipulation, such as linear complementarity models (Jin & Posa, 2024, *inter alia*). Formally, let $z \in \{0,1\}^{n_d}$ denote discrete mode assignments and $\xi \in \mathbb{R}^{n_c}$ denote continuous states and actions. With $x = (z, \xi)$, Eq. (1) specializes to

$$\begin{aligned} \mathcal{P}_{\text{PWA}} : \quad \min_{z,\xi} \quad & c_d^\top z + c_c^\top \xi \\ \text{s.t.} \quad & E^d z = e^d,\ P^d z \le p^d; \quad E^c \xi = e^c,\ P^c \xi \le p^c, \\ & B^d z + B^c \xi \le b; \quad z \in \{0,1\}^{n_d},\ \xi \in \mathbb{R}^{n_c}. \end{aligned} \tag{3}$$



(a) Piece-Wise Affine System $\mathcal{P}_{\text{PWA}}$ **(b) Symbol-Guided Control**

Figure 1: **Solver-Sufficient Symbolization for PWA Systems.** Relative affine-TU groups optimization regions with a compact symbol set. Each conditional optimization subproblem $\mathcal{P}^{\psi}_{\text{joint}}$, given by an assignment of discrete symbols ψ, reduces to an LP-relaxable optimization problem.

The first set of constraints describes logical constraints over modes and linear constraints over continuous variables. The last constraint couples mode assignments with continuous states and actions. Fixing all variables in z leaves a simple linear program (LP) problem, but searching over these assignments is combinatorially expensive. Solver-sufficient symbolization seeks a more compact discrete commitment set that yields better tractability.

2.1 Affine-TU Symbolization in Integer Programs

A classical source of tractability in integer programming (IP) is *total unimodularity* (TU). It is a structural condition under which integrality constraints can be removed without affecting optimality.

Definition 2.1 (Total Unimodularity)**.** *A matrix $A \in \mathbb{Z}^{m\times n}$ is* totally unimodular *if every square submatrix of A has determinant in $\{-1, 0, 1\}$.*

For an IP $\{x \in \mathbb{Z}^n_{\geq 0} : Ax \leq b\}$, by the Hoffman–Kruskal theorem (Hoffman & Kruskal, 2010), A is TU if and only if, for every $b \in \mathbb{Z}^m$, the polyhedron $\{x \in \mathbb{R}^n_{\geq 0} : Ax \leq b\}$ is integral. TU therefore certifies that the corresponding integer program can be solved through its LP relaxation (by removing the integer constraints on x). When the original matrix A is not TU, we can create TU subproblems by fixing a few "summaries" of the decision variables. *Affine TU* formalizes this:

Definition 2.2 (Affine TU (Bader et al., 2018))**.** *Let $A \in \mathbb{Z}^{m\times n}$. A k-row* affine TU decomposition *of A consists of matrices $\widetilde{A} \in \mathbb{Z}^{m\times n}$, $U \in \mathbb{Z}^{m\times k}$, $W \in \{0, \pm 1\}^{k\times n}$ such that*

$$A = \widetilde{A} + UW, \qquad [\widetilde{A}; W] \text{ is TU}^{1}.$$

The affine TU dimension *of A is the minimum k for which such a decomposition exists.*

To see a connection to symbol invention, we can view the discrete summary $\psi = Wx \in \mathbb{Z}^k$ as the assignment to a set of *new* symbols induced by W. For a fixed assignment ψ, the original constraints $Ax \leq b$ becomes $\widetilde{A}x \leq b - U\psi$. Since $[\widetilde{A}; W]$ is TU, this conditioned IP problem has an exact LP relaxation. As a result, instead of searching for the integral assignment of x, we can search over the integral assignments for ψ and solve LPs given each assignment. We get a speed-up when $k \ll n$. This is precisely the principle of solver-sufficient symbolization: introduce discrete symbols that capture the critical decisions needed to make the remaining optimization problem easy.

2.2 Relative Affine-TU Symbolization in MILP

Affine TU applies to pure integer programs. However, for Eq. (3), we additionally need to account for the coupling between discrete modes and continuous trajectories. The equalities $E^d z = e^d$ already fix certain summaries of z, so the symbols to introduce need only capture information beyond those equalities (*relative* to E^d). This motivates the following extension:[2]

Definition 2.3 (Relative Affine-TU)**.** *A matrix $W \in \{0, \pm 1\}^{k\times n_d}$ is a* relative affine-TU certificate *for the discrete interface (E^d, P^d, B^d) if there exist integer matrices $\widetilde{P}^d, U_P, V_P$ and rational matrices U_B, V_B such that*

$$(1) P^d = \widetilde{P}^d + U_P W + V_P E^d, \quad (2) B^d = U_B W + V_B E^d, \quad (3) M := [\widetilde{P}^d; E^d; W] \text{ is TU.}$$

[1] $[A; B]$ denotes vertical concatenation. The same applies thereafter.

[2] We assume E^d is TU and $B^d \in \{0, \pm 1\}^{m_B \times n_d}$. These assumptions accommodate common PWA systems in which discrete symbols enter through mode selection and signed constraints. See discussion in Appendix B.1.

The relative affine-TU dimension $\kappa(\mathcal{P}_{\text{joint}})$ *is the minimum number of rows* k *among such certificates. Such decomposition always exists, since* W *can take* $\mathbf{I}_{n_d \times n_d}$*. Thus,* $\kappa(\mathcal{P}_{\text{joint}}) \leq n_d$*.*

To gain some insight, similarly consider fixing a symbol assignment $\psi = Wz$: as illustrated in Fig. 1 (a), the map W groups primitive mode assignments that share the same symbolic value, depicted by the blue slices labeled ψ_1, ψ_2, ψ_3. Fixing ψ therefore corresponds to choosing a particular group of assignments of all combinatorial assignments of z. Within this group, the discrete variable z can be relaxed to continuous and the inequalities reduce to $\widetilde{P}^d z \leq p^d - U_P \psi - V_P e^d$, which is an integral polytope. That is, the concrete mode decisions of the full z can be recovered after a linear optimization. Meanwhile, the coupling constraints reduce to $B^c \xi \leq b - U_B \psi - V_B e^d$, which depend only on the selected symbols. This is the mechanism underlying the symbol-guided control in Fig. 1 (b): a planner searches over symbolic commitments ψ and, once a commitment is fixed (e.g., as ψ_3 in the figure), solves the corresponding residual problem exactly through an LP relaxation.

Fundamentally, the relative affine-TU dimension $\kappa(\mathcal{P}_{\text{joint}})$ quantifies the irreducible discrete variables, implying the problem hardness is characterized by at most $2^{\kappa(\mathcal{P}_{\text{joint}})}$ modes. Such gains can be quantified in terms of formal complexity statements in Appendix B.2. Furthermore, the minimum number of symbol coordinates is bounded, as given by the following lemma.

Lemma 2.1 (Bounds on Relative Affine-TU Dimension)**.** *Under the assumptions above,* $\operatorname{rank}[B^d; E^d] - \operatorname{rank}(E^d) \leq \kappa(\mathcal{P}_{\text{joint}}) \leq n_d$*. Proofs are in Appendix B.3.*

2.3 Counterexample-Guided Symbol Invention with CPR

Determining the minimum affine-TU dimension is NP-hard. We therefore construct a certificate W using *Counterexample-Guided Partition Refinement* (CPR), a CEGAR-style procedure summarized in Algorithm 1. CPR initializes W_0 from directions in B^d not represented by E^d, guided by the rank lower bound in Lemma 2.1. Given the current W_i, it residualizes P^d against $[W_i; E^d]$ to obtain $\tilde{P}_i^d$ and forms the verification matrix $M_i = [\tilde{P}_i^d; E^d; W_i]$ in Definition 2.3. The TU oracle $\textsc{IsTU}(M_i)$ then determines whether W_i satisfies the relative affine-TU certificate; if so, CPR terminates.

When verification fails, the cause may lie not in which symbols W_i encodes, but in how it is represented as a matrix. The Sparsify step rewrites W_i with another $\{0, \pm 1\}$ basis of the same row space. If no such re-basing yields a TU matrix, the obstruction is real. The Split step then shrinks M_i: we delete rows and columns until obtaining a minimal non-TU submatrix O_i, in which removing any single column makes the matrix TU. CPR therefore promotes the variable z_c of one such column to a new symbol ($k \to k+1$): once a value is assigned to z_c, its column drops out of the conditioned problem, so the remaining columns of O_i form a TU matrix. The loop then repeats. We include more algorithm and implementation details in Appendix C.

3 Solver Sufficiency in Network–Parameterized Systems

The PWA analysis illustrates the central principle of solver-sufficient symbolization: identify discrete commitments that make the remaining optimization tractable. There, the discrete choices are explicit mode variables, and relative affine-TU certifies that fixing their symbolic summaries leaves an LP-solvable problem. We now consider decision making with neural objectives, where the same principle is useful but the characterization of difficulty is different, due to the black-box nature and large parameterization of modern networks.

Energy-based problem formulation. Let s denote the current state or planning context, and let $\mathcal{X} = [\underline{x}, \overline{x}] \subset \mathbb{R}^n$ be a bounded decision space. We represent decision-making as minimizing a state-conditioned neural energy function:

$$\mathcal{P}_{\text{NN}}: \quad \min_{x \in \mathcal{X}} h_s(x). \tag{4}$$

Here, energy denotes a scalar measure of the decision quality, with lower values indicating more desirable decisions; it need not represent physical energy. For single-step control, $x = a$ and one may use the Q function $h_s(a) = -Q(s, a)$. For long-horizon planning, $x = a_{1:H}$ and h_s can evaluate the trajectory through a learned trajectory evaluator or a world model. Unlike the conditioned LPs in the PWA setting, this energy landscape remains highly nonconvex. As illustrated on the left of Fig. 3 (a), doing gradient descent or Langevin dynamics from different initializations may lead to different local minima. Our goal here is to characterize *regions* in such optimization landscapes.

Algorithm 1 CPR Algorithm (PWA System)

Init: PWA system (E^d, P^d, B^d); TU oracle IsTU
Certificate: Relative-affine TU W

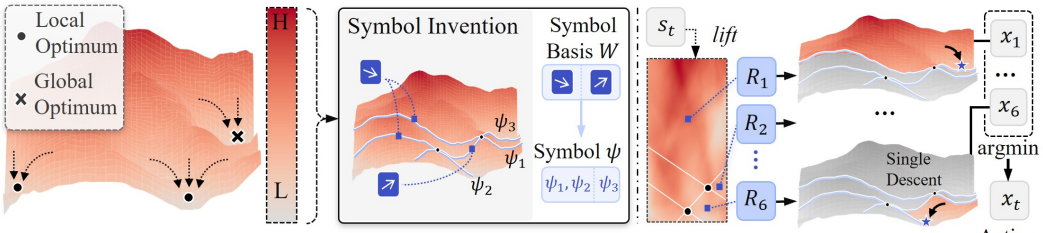
$W_0 \leftarrow B^d \bmod \text{rowspan}(E^d)$ ▷ Lemma 2.1
Repeat
  $\tilde{P}_i^d \leftarrow P^d \bmod [W_i; E^d]$
  $M_i \leftarrow [\tilde{P}_i^d;\ E^d;\ W_i]$
  if IsTU(M_i), **return** W_i
  re-base W_i in rowspan(W_i) ▷ Sparsify
  refresh M_i; **if** IsTU(M_i), **return** W_i
  $O_i \leftarrow$ minimal non-TU core of M_i
  $W^c \leftarrow$ Split$(W_i, O_i, c), \quad \forall c \in \text{cols}(O_i)$
  $W_{i+1} \leftarrow$ Select$(\{W^c\}_c;$ IsTU$)$
    ▷ New symbol invented, $k \to k+1$
Until $k = n_d$
Return W

Algorithm 2 CPR Algorithm (NN System)

Init: Network system $h_s(\cdot)$; resolution τ
Certificate: τ-descent faithfulness $W := (\mathcal{U}, \mathcal{Q})$

$(\mathcal{U}_0, \mathcal{Q}_0) \leftarrow \emptyset$; $G_s \leftarrow$ descent samples ▷ $k = 0$
Repeat
  partition G_s into subproblems by $(\mathcal{U}_i, \mathcal{Q}_i)$
  $\Phi_i \leftarrow$ subproblems that are not τ-faithful
  if $\Phi_i = \emptyset$, **return** $(\mathcal{U}_i, \mathcal{Q}_i)$
  $G_s \leftarrow G_s \cup$ samples near Φ_i
  recompute Φ_i; **return** $(\mathcal{U}_i, \mathcal{Q}_i)$ **if** $\Phi_i = \emptyset$
  $\pi \leftarrow$ lowest pass between two basins in Φ_i
  $u \leftarrow \nabla h_s$ flip across π; $q \leftarrow$ cut at $u^\top x_\pi$
  $\mathcal{U}_{i+1} \leftarrow \mathcal{U}_i \cup \{u\}$; $\mathcal{Q}_{i+1} \leftarrow \mathcal{Q}_i \cup \{q\}$
    ▷ New symbol invented, $k \to k+1$
Until iteration limit reached
Return $W := (\mathcal{U}, \mathcal{Q})$

Figure 2: **One CEGAR Skeleton, Two Instantiations.** Both algorithms share three stages: ■ VERIFY, ■ SPARSIFY (*eliminate false obstacles*), and ■ SPLIT (*pay one symbol for real obstacles*).



(a) Neural Network System $\mathcal{P}_{\text{NN}}$ **(b) Symbol-Guided Control**

Figure 3: **Solver-Sufficient Symbolization for NN Systems.** τ-descent faithfulness partitions a multimodal state-conditioned objective into regions. A symbol-guided control algorithm performs one confined descent per region and selects the lowest-value candidate across all regions.

Remark 3.1. *Our extension removes the discrete variables z in the base optimization formulation. In PWA, the raw model is affine within each mode, and the complexity of the global landscape comes from z. When generalizing this to neural network–parameterized landscapes, we can represent such a complex landscape directly over x, without declaring mode variables, as illustrated in Fig. 3 (a).*

3.1 τ-DESCENT FAITHFUL SYMBOLIZATION IN NEURAL OBJECTIVES

In the PWA setting, the tractability of the conditioned problem follows from an algebraic property of the constraint matrix. For neural objectives, conditioned solvability is instead characterized by the geometry that a local solver must navigate. In particular, a region may contain several local minima yet remain manageable if the solver can escape the barriers separating them. We model this capability through an escape budget $\tau > 0$, which bounds the temporary increase in energy above the lowest value reached so far. A path satisfying this budget is defined as follows.

Definition 3.1 (τ-Admissible Path). *Let $R \subseteq \mathcal{X}$ and $x \in R$. A continuous path $\gamma : [0, 1] \to R$ with $\gamma(0) = x$ is τ-*admissible *for h_s if*

$$h_s(\gamma(t)) \le \min_{0 \le r \le t} h_s(\gamma(r)) + \tau, \qquad \forall t \in [0, 1].$$

We denote by $\Gamma^\tau_{s,R}(x)$ the set of all such paths starting from x.

We then define a region to be easily *solvable* if this allowance suffices to reach a near-optimal point from every initialization in a region. Assume that h_s is continuous on $\mathcal{X}$.

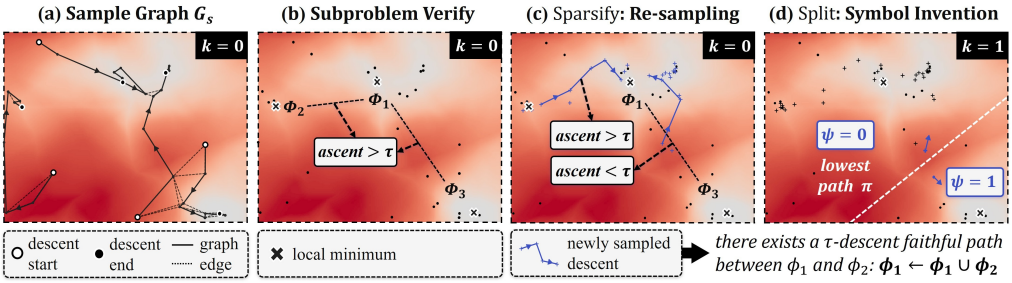


Figure 4: **Exemplar Symbol Invention in One Round.** Take Lunar Lander as the exemplar task.

Definition 3.2 (τ-Descent Faithfulness)**.** *For a nonempty region $R \subseteq \mathcal{X}$ and $x \in R$, define the best reachable endpoint value*

$$v^{\tau}_{s,R}(x) := \inf_{\gamma \in \Gamma^{\tau}_{s,R}(x)} h_s(\gamma(1)).$$

The region R is τ-descent faithful *with respect to h_s if*

$$\sup_{x \in R} v^{\tau}_{s,R}(x) \le \inf_{y \in R} h_s(y) + \tau.$$

The condition does not require convexity or a unique local minimum. It certifies accessibility under the specified path model. This provides an analogy to the relative affine-TU certificate. In both settings, a symbol induces conditioned regional problems under a hardness measurement. For PWA systems, that guarantee is the exact optimizability through an LP relaxation; for neural objectives, it is the near-optimal accessibility within an escape budget.

Given the definition, we next seek symbols that induce such conditioned regions. As illustrated in Fig. 3 (a), the symbol invention problem is to choose projection directions and quantizers that organize the decision space into regions that are τ-descent faithful, with as few directions and thresholds as possible. Formally, we seek a fixed symbolic map

$$W(x) = \left(q_1(u_1^{\top} x), \ldots, q_k(u_k^{\top} x)\right), \tag{5}$$

where $u_1, \ldots, u_k \in \mathcal{U}$ are projection directions and each $q_j \in \mathcal{Q}$ is a finite-valued quantizer. Each symbolic assignment ψ defines a region $R_\psi := \{x \in \mathcal{X} : W(x) = \psi\}$.

During execution, the current state s_t instantiates h_{s_t} and the corresponding quantizers in $W = (\mathcal{U}, \mathcal{Q})$, whose assignments partition $\mathcal{X}$ into conditioned regions. The controller then performs single confined descent in each region under the τ-admissible solver model and outputs the action sequence attaining the lowest objective value. By Proposition B.4, the value is within τ of the global optimum. The procedure is illustrated in Fig. 3 (b).

3.2 Counterexample-Guided Symbol Invention with Neural CPR

Neural CPR inherits the CEGAR style. Since the network parameters cannot be characterized explicitly, CPR constructs symbols from sampling. Given a state set $\mathcal{S}$ (e.g., from demonstration trajectories), we run multi-start descent to collect outputs, objective values, and gradients. For each state $s \in \mathcal{S}$, an m-nearest-neighbor graph G_s connects these samples, with edges storing the objective values at the midpoint and both ends. Next, neural CPR discovers a symbolic partition through sampling-based, counterexample-guided refinement (see Algorithm 2; detailed version in Appendix D). Verification is exact on the sample graph G_s, not on the continuous region.

Fig. 4 gives a concrete example of a refinement loop on state s. **(a)** Random restarts of descent settle in different basins; we link descent nodes by m-nearest neighbours into a graph G_s. **(b)** Verification partitions G_s into basins that are each internally τ-descent faithful, i.e., Φ_1, Φ_2, Φ_3. **(c)** SPARSIFY-step: such a separation may itself be an artefact of under-sampling: re-sampling descents between two basins can recover a τ-admissible path connecting them, merging ϕ_1 and ϕ_2 into a single basin at no symbolic cost. **(d)** SPLIT-steps: basins that remain separated lead to new symbols. We first estimate a lowest-barrier path π from the trapped basin toward a lower-energy basin and locate a transition point x_π where the gradient directions "*flip*" within its neighborhood. The gradient directions provide a heuristic separating direction u_{k+1}. Its projection $b_{k+1} = u_{k+1}^{\top} x_\pi$ defines a candidate quantization threshold. CPR then augments the symbolic map with this new symbol. Steps b-d will be repeated until no violations are found. Details are deferred to Appendix D.

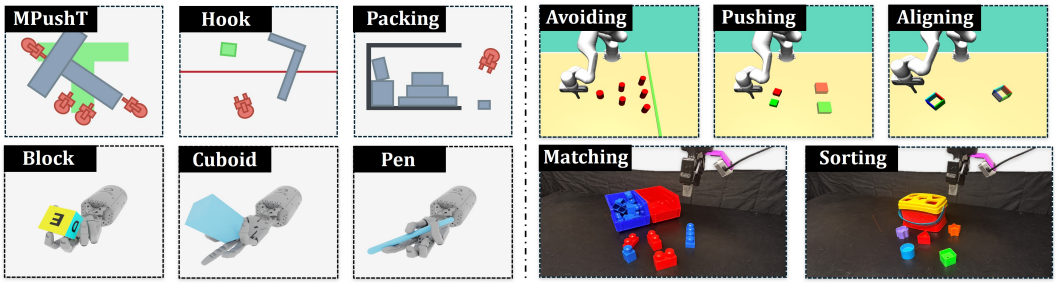


Figure 5: **Task Suites.** Dashed line divides the tasks for PWA systems and for network systems.

Table 1: **Solver Sufficiency on PWA Systems.** For inference-time speedup **IS**, E2E is 1.00×.

Method	2D Manipulation						In-Hand Reorientation					
	MPushT		Hook		Shelf		Block		Cuboid		Pen	
	SR↑	IS↑	SR↑	IS↑	SR↑	IS↑	SR↑	IS↑	SR↑	IS↑	SR↑	IS↑
- Solver-Oriented Methods												
CoCo	0.75	3.57×	0.63	**3.18**×	0.75	1.39×	0.69	**2.70**×	0.81	**1.61**×	0.75	1.28×
GBD-MPC	0.94	0.12×	0.94	0.06×	0.81	0.09×	**0.94**	0.32×	0.88	0.21×	**0.88**	0.06×
- Symbol-Oriented Methods												
PDDLStream	0.88	0.77×	0.81	0.36×	0.69	0.42×	0.81	0.67×	0.81	0.38×	0.75	0.17×
PGS	0.81	0.08×	0.69	0.37×	0.75	0.16×	0.75	0.15×	0.69	0.13×	0.81	0.04×
IVNTR	0.88	0.15×	0.81	0.22×	0.75	0.12×	**0.94**	0.10×	0.75	0.10×	0.81	0.03×
pix2pred	0.88	0.03×	0.88	0.09×	**0.94**	0.08×	**0.94**	0.05×	0.88	0.04×	0.75	0.01×
Ours	**1.00**	**4.81**×	**1.00**	3.07×	**0.94**	**2.77**×	0.88	1.53×	**0.94**	1.34×	**0.88**	**1.94**×

4 Experiments

In this section, we present extensive experimental results to answer the following questions:
Q1. Can solver-sufficient symbols preserve optimality while reducing solving cost?
Q2. Can solver-sufficient symbols explain why certain human-defined symbols are useful?
Q3. Can solver-sufficient symbols retain the interpretability and steerability of symbolic planning?

Experiment Setup. We evaluate our method extensively in both PWA (three 2D manipulation tasks and three in-hand reorientation tasks with multi-finger grippers) and network settings (three D3IL tasks to study symbolic structure discovery in multimodal decision landscapes (Jia et al., 2024), and two real-world tasks). Our task suites are shown in Fig. 5. The details can be found in Appendix F.

Baselines. For PWA systems, we compare against solver-specific methods CoCo and GBD-MPC, which respectively predict discrete modes before applying convex optimization and refine the original problem through Benders cuts (Cauligi et al., 2020; Lin, 2024) to accelerate solving. We also compare against existing symbol invention methods: PDDLStream with LLM-invented symbols, PGS, IVNTR, and pix2pred, which construct symbolic abstractions before grounded solving (Garrett et al., 2020; Silver et al., 2023; Li et al., 2025; Athalye et al., 2026). The first accelerates the same MILP without symbols, and the second uses symbols not derived from the solver. For neural network–based systems, we evaluate multimodal policies BESO, IBC, DDPM, ACT, and BeT under their energy-based variants (Reuss et al., 2023; Florence et al., 2022; Chi et al., 2025; Zhao et al., 2023; Shafiullah et al., 2022).

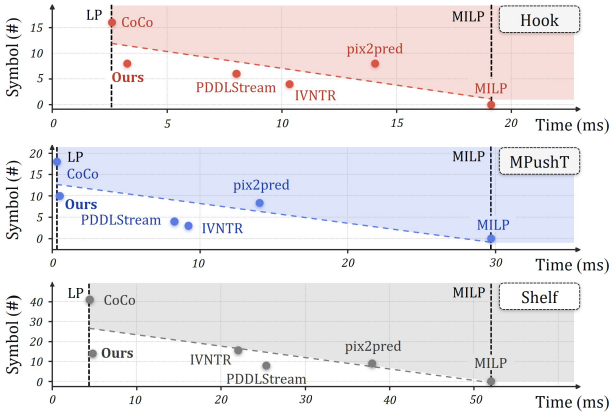
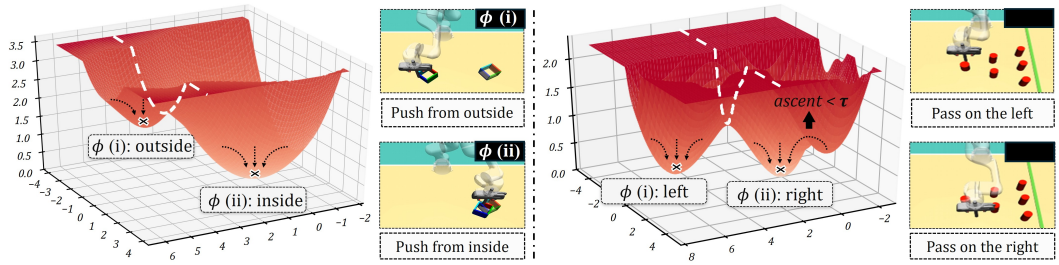


Figure 6: **Symbol Count vs. Conditioned Solving Time.** The dashed line shows the fitted trend across methods.

Table 2: **Solver Sufficiency on Network Systems.** For $\mathbf{DC}_\delta$, we take $\delta = 0.001$.

Method	Avoiding		Pushing		Aligning		Matching		Sorting	
	SR↑	$\mathbf{DC}_\delta$ ↓	**SR**↑	$\mathbf{DC}_\delta$ ↓	**SR**↑	$\mathbf{DC}_\delta$ ↓	**SR**↑	$\mathbf{DC}_\delta$ ↓	**SR**↑	$\mathbf{DC}_\delta$ ↓
DDPM	**1.00**	7.03	0.90	**1.00**	**0.94**	10.31	**0.75**	5.15	**0.88**	1.68
ACT	0.91	833.69	0.91	5.93	0.91	9.25	0.69	173.45	0.63	259.67
BESO	0.97	329.03	0.63	198.57	0.75	338.03	**0.75**	87.48	0.69	147.85
IBC	0.81	690.34	0.63	312.70	0.63	349.94	0.63	340.78	0.69	562.95
BeT	0.72	495.50	0.30	105.63	0.53	94.66	0.56	936.83	0.75	1325.94
DDPM (Symbols)	**1.00**	**2.00**	**0.91**	**1.00**	0.84	**2.41**	**0.75**	**1.20**	**0.88**	**1.00**



(a) Aligning **Task at Step=0** **(b)** Avoiding **Task at Step=4**

Figure 8: **Behavior Steering with Invented Symbols.** Different symbol values correspond to distinct basin-level behaviors, allowing steering the controller toward preferences.

4.1 Q1. Preserving Optimality while Reducing Solving Cost

We first evaluate solver sufficiency in the PWA setting using success rate (**SR**) and per-step inference-time speedup (**IS**). As shown in Table 1, our symbols largely preserve solution quality while reducing solving time. CoCo obtains efficient conditioned solves by fixing complete discrete assignments to construct an LP, but may exclude the optimum; in contrast, our method exposes only the commitments required for faithful relaxation, leaving the residual problem LP-solvable with a lower-dimensional symbolic search (Fig. 6). The remaining failures in our algorithm are mainly due to early commitments that invalidate successive transitions, caused by a too-short planning horizon.

For neural settings, we construct τ-descent-faithful symbols on top of a trained DDPM policy without modifying its parameters. We report **SR** and descents-to-coverage ($\mathbf{DC}_\delta$), the number of descents required to reach the τ-neighborhood of the global optimum with probability at least $1-\delta$. As shown in Table 2, symbol-guided solving maintains high task success while substantially reducing $\mathbf{DC}_{10^{-3}}$, indicating that near-optimal solutions can be recovered with only a few conditioned descents.

Table 3 further illustrates that our CPR-invented symbols encode action-relevant commitments, whereas baseline symbols mainly describe state or geometry. Overall, our algorithms and results provide a solver-centric interpretation of usefulness.

4.2 Q2. Explaining the Utility of Human-Defined Symbols

To investigate why certain human-defined symbols are computationally useful, we study `MPushT`, a PushT variant in which four pushers simultaneously move a T-shaped block to a target pose. We manually construct nine MILP formulations containing 127 factors in total (Appendix F.3). We compute a CPR symbolization for each formulation and evaluate each human-defined binary factor z_i along two dimensions. Task necessity (**TN**) measures the decrease in task success when the factor is disabled. Solver necessity (**SN**) measures how often the factor is covered by the CPR-invented symbols, defined as the fraction of symbol assignments $\psi' \in \Psi$ for which conditioning on $Wz = \psi'$ uniquely determines z_i. Since CPR identifies the underlying combinatorial structure that

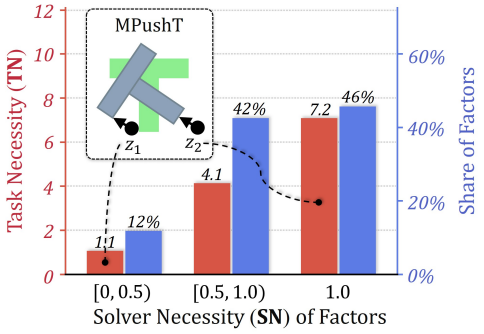


Figure 7: **Human-Defined Concept Analysis.**

Table 3: **Semantics of CPR-Invented Symbols.** We only show two invented symbols per setting.

Task	Ours	PGS	pix2pred
MPushT	$\sum_{\tau_c<0} z_c - \sum_{\tau_c>0} z_c$ ▷ is clockwise push $z_{\text{top}} - z_{\text{bottom}}$ ▷ is push above	$\xi_\theta \leq -0.9$ ▷ T orientation ≤ threshold $\xi_{\text{dist}} \leq 1.0$ ▷ near the goal	stem_pointing_down_right rotation_error_small
Block	$z_{\text{s}\to\text{ff}}$ ▷ index finger contacts object $z_{\text{th}\to\text{palm}}$ ▷ thumb contacts palm	$\xi_{\text{ff}\to\text{thu}} > 0.5$ ▷ index finger near thumb $\xi_{11} \leq 0.6$ ▷ control variable ≤ threshold	orientation_over_bound orientation_southeast

Table 4: **Behavior Steering with Symbols.** See Fig. 8 for preference definitions.

Method	Avoiding				Aligning			
	ϕ (i)		ϕ (ii)		ϕ (i)		ϕ (ii)	
	SR	**PR**	**SR**	**PR**	**SR**	**PR**	**SR**	**PR**
Ours	1.00	0.94	1.00	1.00	0.75	0.75	0.83	0.81

Table 5: **Invented Symbol Complexity Scale with Planning Horizon.**

	$H=4$		$H=8$		$H=12$		$H=16$	
Task	k	**SR**	k	**SR**	k	**SR**	k	**SR**
Avoiding	0	1.00	1	1.00	1	0.91	2	0.94
Aligning	4	0.84	5	0.75	6	0.69	8	0.63

must be preserved under relaxation, a high **SN** indicates that a factor encodes a discrete distinction that cannot simply be relaxed away by the MILP solver. By contrast, a discrete variable z_i that cannot be relaxed before the CPR symbols are fixed but can be relaxed afterwards has low **SN**, indicating that it contributes no intrinsic combinatorial hardness. As shown in Fig. 7, both **TN** and the fraction of factors with clear human semantics increase with **SN**, suggesting factors that cannot be simply relaxed are more important for task success. For example, z_2, a push at the bottom of the stem, has higher **SN** and **TN** than z_1, a push at a corner of the top bar. The decision of z_2 matters in more task instances, as the contact force of z_1 can be replaced by neighboring pushes.

4.3 **Q3.** Retaining Symbolic Interpretability and Steerability

The invented symbols also enable steering at the level of motion preferences. As shown in Fig. 8, different manipulation strategies will emerge in policies trained for `Aligning` (pushing from inside versus outside), and `Avoiding` (passing an obstacle on the left versus right). These alternative strategies correspond to distinct basins of the underlying energy landscape h_s, which CPR will discover as separate symbol regions. Because the symbol map is defined over x (Eq. (5)), fixing a symbol value ψ restricts the control command x to the corresponding region $\Phi^{-1}(\psi)$. To make these regions interpretable, we use a vision-language model to label rollout videos from different symbol regions and assign each region a behavioral label. For example, the left basin in Fig. 8(a) is labeled "pushing from outside." At inference time, we can manually set the symbol values associated with the desired preference. As shown in Table 4, this yields both high task success rates (**SR**) and high preference success rates (**PR**: the fraction of successful trials that also satisfy the preference).

Discussion: horizon-induced symbol complexity. As the planning horizon grows, the decision landscape becomes more complex through accumulated long-horizon state-action dependencies, and the required symbol dimension k generally increases (Table 5). The growth may plateau once dominant modes are exposed. Yet the resulting increase in basin complexity also makes optimization harder, leading to a moderate decline in **SR**. Tasks with clearer decision modes scale more favorably: for example, `Avoiding`, which mainly requires choosing a passing direction around obstacles, needs far fewer symbols than `Aligning`.

Limitations. So far, our formulation uses linear symbol models $u^\top x$, which may be restrictive in some cases. Yet, if task-relevant modes are sufficiently abstracted and linearly separable in a learned feature space, $u^\top x$ can still partition the landscape effectively; otherwise, nonlinear maps $u(x)$ may be preferable. We include more discussions and ablation studies in Appendix E.

5 Conclusion

We presented an inference-algorithm–centric view of symbol invention: useful symbols expose discrete commitments that simplify planning. We formalized this principle for piecewise-affine systems and extended it to neural energy landscapes. Across simulated and real-robot tasks, the invented symbols reduce solving cost while preserving solution quality and provide an interpretable interface for behavior steering. These results connect symbolic abstraction to computational utility, offering a basis for both inventing new symbols and understanding the usefulness of human-defined concepts.

AI USE STATEMENT

We used generative AI tools, to a limited extent, for tasks with required disclosure: feedback on the methodology and the interpretation of results, translation of author-drafted text, and assistance in writing and checking proofs. They were not used for the other tasks with required disclosure. We also used them to edit code written by the authors, improve readability, draft parts of the related-work and appendix text from author outlines, prepare LaTeX figure code, and search for and format references. The authors reviewed all AI-assisted work, tested the code, verified the proofs and references, and take full responsibility for the content of this paper.

REPRODUCIBILITY STATEMENT

Assumptions and full proofs are given in Appendix B, detailed pseudocode for both CPR instantiations in Appendices C and D, hyperparameters and ablations in Appendix E, and task definitions and MILP factorizations in Appendix F. Our code is included in the supplementary material.

REFERENCES

Masataro Asai and Alex Fukunaga. Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary. In *Proceedings of the aaai conference on artificial intelligence*, volume 32, 2018.

Ashay Athalye, Nishanth Kumar, Tom Silver, Yichao Liang, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Predicate invention from pixels via pretrained vision-language models. In *AAAI 2025 Workshop LM4Plan*, 2025.

Ashay Athalye, Nishanth Kumar, Tom Silver, Yichao Liang, Jiuguang Wang, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. From pixels to predicates: Learning symbolic world models via pretrained vlms. *IEEE Robotics and Automation Letters*, 2026.

Charles Audet, Pierre-Yves Bouchet, and Loïc Bourdin. A partitioned optimization framework for structure-aware problems. *Journal of Optimization Theory and Applications*, 210(1):9, 2026.

Jörg Bader, Robert Hildebrand, Robert Weismantel, and Rico Zenklusen. Mixed integer reformulations of integer programs and the affine tu-dimension of a matrix. *Mathematical Programming*, 169(2):565–584, 2018.

Kevin Black, Manuel Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. *Advances in Neural Information Processing Systems*, 38:33383–33407, 2026.

Tom Bylander. The computational complexity of propositional strips planning. *Artificial Intelligence*, 69(1-2):165–204, 1994.

Abhishek Cauligi, Preston Culbertson, Bartolomeo Stellato, Dimitris Bertsimas, Mac Schwager, and Marco Pavone. Learning mixed-integer convex optimization strategies for robot planning and control. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 1698–1705. IEEE, 2020.

Abhishek Cauligi, Preston Culbertson, Edward Schmerling, Mac Schwager, Bartolomeo Stellato, and Marco Pavone. Coco: Online mixed-integer control via supervised learning. *IEEE Robotics and Automation Letters*, 7(2):1447–1454, 2021.

Hubie Chen and Omer Giménez. Act local, think global: Width notions for tractable planning. In *ICAPS*, volume 1, pp. 5, 2007.

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)*, pp. 2730–2736. IEEE, 2022.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Rohan Chitnis, Tom Silver, Joshua B Tenenbaum, Tomas Lozano-Perez, and Leslie Pack Kaelbling. Learning neuro-symbolic relational transition models for bilevel planning. In *2022 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 4166–4173. IEEE, 2022.

Rina Dechter. Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. *Artificial Intelligence*, 41(3):273–312, 1990.

Rina Dechter. *Constraint processing*. Elsevier, 2007.

Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification. *Discrete & computational geometry*, 28(4):511–533, 2002.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on robot learning*, pp. 158–168. PMLR, 2022.

Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the international conference on automated planning and scheduling*, volume 30, pp. 440–448, 2020.

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.

Héctor Geffner and Nir Lipovetzky. Width and serialization of classical planning problems. 2012.

Arthur M Geoffrion. Generalized benders decomposition. *Journal of optimization theory and applications*, 10(4):237–260, 1972.

Mark K Ho, David Abel, Carlos G Correa, Michael L Littman, Jonathan D Cohen, and Thomas L Griffiths. People construct simplified mental representations to plan. *Nature*, 606(7912):129–136, 2022.

Alan J Hoffman and Joseph B Kruskal. Integral boundary points of convex polyhedra. *50 Years of integer programming 1958–2008*, pp. 49, 2010.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Xiaogang Jia, Denis Blessing, Xinkai Jiang, Moritz Reuss, Atalay Donat, Rudolf Lioutikov, and Gerhard Neumann. Towards diverse behaviors: A benchmark for imitation learning with human demonstrations. In *International Conference on Learning Representations*, volume 2024, pp. 41666–41686, 2024.

Wanxin Jin and Michael Posa. Task-driven hybrid model reduction for dexterous manipulation. *IEEE Transactions on Robotics*, 40:1774–1794, 2024.

George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61:215–289, 2018.

Bowen Li, Tom Silver, Sebastian Scherer, and Alexander Gray. Bilevel learning for bilevel planning. *arXiv preprint arXiv:2502.08697*, 2025.

Yichao Liang, Nishanth Kumar, Hao Tang, Adrian Weller, Joshua B Tenenbaum, Tom Silver, João F Henriques, and Kevin Ellis. Visualpredicator: Learning abstract world models with neuro-symbolic predicates for robot planning. In *International Conference on Learning Representations*, volume 2025, pp. 60416–60444, 2025.

Falk Lieder and Thomas L Griffiths. Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and brain sciences*, 43:e1, 2020.

Xuan Lin. Accelerating hybrid model predictive control using warm-started generalized benders decomposition. *arXiv preprint arXiv:2406.00780*, 2024.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Yuejiang Liu, Jubayer Hamid, Annie Xie, Yoonho Lee, Max Du, and Chelsea Finn. Bidirectional decoding: Improving action chunking via guided test-time sampling. In *International Conference on Learning Representations*, volume 2025, pp. 4594–4627, 2025.

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*, 2019.

Jiayuan Mao, Tomás Lozano-Pérez, Josh Tenenbaum, and Leslie Kaelbling. What planning problems can a relational neural network solve? *Advances in Neural Information Processing Systems*, 36:59522–59542, 2023.

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.

Hanna M Pasula, Luke S Zettlemoyer, and Leslie Pack Kaelbling. Learning symbolic models of stochastic domains. *Journal of Artificial Intelligence Research*, 29:309–352, 2007.

Michael Posa, Scott Kuindersma, and Russ Tedrake. Optimization and stabilization of trajectories for constrained dynamical systems. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1366–1373. IEEE, 2016.

Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. *arXiv preprint arXiv:2304.02532*, 2023.

Luis Rodrigues and Jonathan P How. Observer-based control of piecewise-affine systems. *International Journal of Control*, 76(5):459–477, 2003.

Nur Muhammad Shafiullah, Zichen Cui, Ariuntuya Arty Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022.

Tom Silver, Rohan Chitnis, Nishanth Kumar, Willie McClinton, Tomás Lozano-Pérez, Leslie Kaelbling, and Joshua B Tenenbaum. Predicate invention for bilevel planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 12120–12129, 2023.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Appendix Contents

A Related Works **13**

A.1 Search-then-Optimize: From MILP to Energy-Based Models 13

A.2 Symbol Invention in Neuro-Symbolic Planning 14

A.3 Planning Efficiency and Structural Coupling 14

B Theoretical Foundations **15**

B.1 Structural Assumptions on the Discrete Interface 15

B.2 Numerical Complexity of Symbolic Conditioning 15

B.3 Bounds on Relative Affine-TU Dimension 16

B.4 Bellman Factorization for Relative Affine-TU 16

B.5 Guarantee of a τ-Faithfulness 17

C Details of CPR for PWA Systems **17**

D Details of Neural CPR for Network Systems **19**

E Further Experimental Results **20**

E.1 Beyond Linear Symbol Axes 20

E.2 Ablations on Solver Resolution τ 20

E.3 Ablations on Sampling Evidence. 21

F Tasks Details **21**

F.1 Task Requirements 21

F.2 MILP Setups 22

F.3 `MPushT` MILP Variations 22

A Related Works

A.1 Search-then-Optimize: From MILP to Energy-Based Models

Many planning methods share a two-stage shape: commit to a discrete quantity, then solve a conditioned problem that is tractable once it is fixed. Optimization formalizes this as the partitioned optimization framework, where minimizing over partition indices recovers solutions of the original problem (Audet et al., 2026). For piecewise-affine models written as MILPs, the index is the mode assignment and each conditioned problem is convex; methods differ in how that assignment is obtained, by warm-starting branch-and-bound (Marcucci & Tedrake, 2020), by predicting it so that a convex program remains online (Cauligi et al., 2021), or by transferring Benders cuts across instances (Lin, 2024). In each, the discrete structure is declared by the formulation and the effort goes into searching it faster; which partition to use, and whether the conditioned problem is tractable under it, remain open (Audet et al., 2026).

When the objective is a learned network, the shape persists but no partition is declared. Energy-based and diffusion policies represent alternative valid behaviors as distinct basins of the learned objective (Florence et al., 2022; Chi et al., 2025), and which basin a run settles in is decided by its initialization. This is how multimodality is represented, not a defect, but it makes the basin sampled rather than decided. Never represented as a variable, it cannot be committed to or held fixed across

successive solves. Adjacent action chunks, drawn from independent noise, may therefore select incompatible modes (Black et al., 2026).

Existing remedies act on the generated actions rather than on the basin: averaging overlapping predictions (Zhao et al., 2023), selecting among candidates for consistency with previous decisions (Liu et al., 2025), or correcting the denoising process by inpainting (Black et al., 2026); each enforces agreement between consecutive outputs without making the basin explicit. We instead supply an explicit index together with what fixing it buys, at two strengths: for PWA systems the LP relaxation of each conditioned problem is exact (Proposition B.1), whereas for learned objectives one confined descent per region reaches within τ of the global optimum regardless of its start (Proposition B.4), certified from sampled descents and therefore one-sided over the continuous domain.

A.2 Symbol Invention in Neuro-Symbolic Planning

Symbol invention involves two questions: where candidates come from, and which to keep; existing methods differ mainly in the first. Symbols have been grounded in the preconditions and effects of given skills (Konidaris et al., 2018), taken as the discrete code of an autoencoder (Asai & Fukunaga, 2018), tied to a language vocabulary (Mao et al., 2019), enumerated from a grammar over given object-centric features (Silver et al., 2023; Chitnis et al., 2022), learned as neural classifiers supervised by operator effects (Li et al., 2025), or proposed by a pretrained vision-language model (Athalye et al., 2025; Liang et al., 2025); with the set fixed, planners interleave symbolic search with continuous samplers (Garrett et al., 2020). In each case the admissible symbols are delimited before learning begins, by a grammar, a feature set, a code size, or a pretrained vocabulary.

The second question is answered by measurement. Candidates are scored by the likelihood they assign to observed transitions (Pasula et al., 2007), or by planning performance on a distribution of training tasks under a particular planner (Silver et al., 2023). Such scores are informative but extrinsic: they report how a symbol set behaved, not what fixing its value does to the remaining problem. They also do not single out a set. The same task admits many symbol sets whose measured performance is close, and a success rate then offers no basis for choosing among them.

We take both the source and the criterion from the solver itself. Candidates are read off the structure that makes solving hard—the discrete–continuous interface for PWA systems, and the basins revealed by sampled descents for learned objectives—so no vocabulary is delimited in advance. The criterion is a property of the conditioned problem rather than a measurement: an exact LP relaxation for PWA systems, and, at the weaker strength that sampling permits, τ-descent faithfulness for learned objectives (Proposition B.4). The count k is then itself a measure of difficulty—the commitments that cannot be relaxed away—with the computable lower bound of Lemma 2.1.

A.3 Planning Efficiency and Structural Coupling

This paper is concerned with planning efficiency: what a solve actually costs on a given problem. Plan existence being PSPACE-hard in general (Bylander, 1994), efficiency is studied through finer structural measures, separating along two axes—how many steps must be searched, and how much must be considered jointly at each step. In heuristic search these are the expansions to the goal and the branching kept at each expansion; Mao et al. (2023) make the same separation explicit for relational policies, classifying problems by how a policy circuit's width and depth grow with object count and horizon. Work on the first axis replaces primitive steps with temporal abstractions (Sutton et al., 1999); work on the second prunes states whose novelty exceeds a bound (Geffner & Lipovetzky, 2012).

Both axes trace to one property: coupling among the decisions in a plan—the extent to which they cannot be made independently. Width notions make it operational: how many atoms must hold together before a subgoal can be treated in isolation (Chen & Giménez, 2007; Geffner & Lipovetzky, 2012), and the regression width of Mao et al. (2023), $|c| + \max_i |c_i|$, counts the conditions that must be maintained while a subgoal is achieved. When subgoals serialize, search factors and both axes shrink; otherwise they must be decided jointly. The induced width of a constraint graph measures the same thing (Dechter, 2007).

All of these are defined on a declared combinatorial structure: a predicate vocabulary with preconditions and effects, or a constraint graph. We measure coupling instead by whether the solver's

relaxation is faithful using affine TU tools (Hoffman & Kruskal, 2010; Bader et al., 2018) for PWA systems, sublevel persistence (Edelsbrunner et al., 2002) for learned objectives. The two are not interchangeable: total unimodularity is not a property of the constraint graph, since matrices with the same sparsity pattern can differ in it by sign alone. RATdim counts the commitments required before the LP relaxation is exact (Lemma 2.1), and fixing them decouples the discrete and continuous sides; τ-descent faithfulness asks the same question where no transition structure is declared, with a certificate established on sampled descents rather than exactly

B Theoretical Foundations

B.1 Structural Assumptions on the Discrete Interface

Assumption B.1. *The discrete matrices in Eq. (3) satisfy:* (1) E^d *is TU;* (2) $B^d \in \{0, \pm 1\}^{m_B \times n_d}$.

Physically, both are mild. Condition (1) holds when each mode variable is selected by at most one equality row, as in per-contact one-hot mode logic. Condition (2) says that a contact either enters a bridge constraint or does not, and if it does, only with a direction; the magnitude is carried by B^c and b. To enforce B^d to be a ternary matrix, we can divide each bridge row by the magnitude of its discrete coefficients, which is available for big-M gates and signed incidence.

B.2 Numerical Complexity of Symbolic Conditioning

This section quantifies the structural simplification induced by solver-sufficient symbols through the numerical complexity. We formally give the following proposition.

Proposition B.1 (RAT Integer-Dimension Complexity)**.** *(Proof in Section B.2) Under Assumption B.1, let W be a k-row relative affine-TU certificate, $\mathcal{P}^{\psi}_{\mathrm{PWA}}$ admits the formulation*

$$\mathcal{P}^{\psi}_{\mathrm{PWA}} : \min_{z,\xi} \; c_d^\top z + c_c^\top \xi \qquad \text{s.t.} \quad (z, \xi) \in \mathcal{P}_{\mathrm{PWA}}, \; Wz = \psi. \tag{6}$$

For every fixed ψ, Eq. (6) is an LP. Let m and L be the number of constraints and input encoding length. Searching over ψ and then optimizing can be solved in

$$2^{\mathcal{O}(k \log n_d)} \operatorname{poly}(n_d, n_c, m, L).$$

The same reduction can also be viewed through Bellman lens, as shown in Section B.4.

Proposition B.1 states that a k-row certificate moves the exponential dependence from the n_d primitive modes to the k symbols. We first recall Lenstra's theorem, which converts a bound on the number of integer variables into a running time.

Lemma B.1 (Lenstra Theorem)**.** *A mixed-integer linear program with n integer variables, arbitrary continuous variables, m constraints and rational data of encoding length L is solvable in*

$$2^{\mathcal{O}(n^3)} \operatorname{poly}(n, m, L).$$

Proof. After fixing the symbol ψ, the feasible set factorizes as in follows, and the objective separates accordingly as well.

$$\mathcal{F}^d(\psi) = \left\{ z \in [0,1]^{n_d} : E^d z = e^d, \; Wz = \psi, \; \widetilde{P}^d z \le p^d - U_P \psi - V_P e^d \right\}, \tag{7a}$$

$$\mathcal{F}^c(\psi) = \left\{ \xi \in \mathbb{R}^{n_c} : E^c \xi = e^c, \; P^c \xi \le p^c, \; B^c \xi \le b - U_B \psi - V_B e^d \right\}. \tag{7b}$$

By Definition 2.1, $\mathcal{F}^d(\psi)$ is an integral polytope: relaxing z to $[0,1]^{n_d}$ loses nothing, and $\mathcal{P}^{\psi}_{\mathrm{PWA}}$ is solved by a single LP. And such transform preserves the optimality of the optimization problem, since the vertices of the solution automatically fall on integer points. Therefore, the lifted program has only k exact integer variables, so Lemma B.1 replaces the direct bound $2^{\mathcal{O}(n_d^3)} \operatorname{poly}(\cdot)$ by $2^{\mathcal{O}(k^3)} \operatorname{poly}(n_d, n_c, m, L)$.

However, we can further accelerate the solution process. In practice, we does not solve the lifted program and enumerate over remaining k discrete variables; we instead searches the ψ-symbol space directly. Each row of W lies in $\{0, \pm 1\}^{n_d}$ and z is binary, so $w_j^\top z \in [-n_d, n_d]$ and $|\Psi| \le (2n_d + 1)^k$. Solving one LP under a fixed symbol commitment ψ is exact, giving $2^{\mathcal{O}(k \log n_d)} \operatorname{poly}(n_d, n_c, m, L)$. □

B.3 Bounds on Relative Affine-TU Dimension

Lemma 2.1 brackets the relative affine-TU dimension between a rank floor and the primitive dimension n_d, with the following proof.

Proof. Let W be a certificate and U_B, V_B the rational multipliers in condition (2). Then $B^d = [\,U_B \;\; V_B\,]\,[W; E^d]$, so every row of B^d lies in $\mathrm{rowspan}[W; E^d]$, whence $\mathrm{rowspan}[B^d; E^d] \subseteq \mathrm{rowspan}[W; E^d]$. Taking dimensions and using sub-additivity of rank under vertical concatenation,

$$\mathrm{rank}[B^d; E^d] \;\le\; \mathrm{rank}[W; E^d] \;\le\; \mathrm{rank}(W) + \mathrm{rank}(E^d) \;\le\; k + \mathrm{rank}(E^d),$$

the last step because W has k rows. Rearranging gives the lower bound. For the right-hand bound, Since we can choose $W = \mathbf{I}_{n_d \times n_d}$ in the worst case, so it also holds. □

B.4 Bellman Factorization for Relative Affine-TU

We first build a proposition before introducing the Bellman factorization of relative affine-TU. In our settings, the matrices E^d, P^d, B^d describe the full planning horizon and encode cross-step couplings. In many robotic formulations, however, the same local constraint structure repeats at each time step. Based on the circumstance, we propose the following proposition, where the same symbolic structure can be reused across time.

Proposition B.2 (Temporal Separation of Relative Affine-TU)**.** *Suppose*

$$E^d = I_T \otimes \bar{E}^d, \qquad P^d = I_T \otimes \bar{P}^d, \qquad B^d = I_T \otimes \bar{B}^d.$$

If $\bar{W}$ is a $\bar{k}$-row relative affine-TU certificate for $(\bar{E}^d, \bar{P}^d, \bar{B}^d)$, then $W = I_T \otimes \bar{W}$ is a relative affine-TU certificate for the full-horizon problem, with $k = T \cdot \bar{k}$.

Proposition B.1 treats $\mathcal{P}_{\mathrm{PWA}}$ as one monolithic program. When the interface repeats in time, Proposition B.2 lets the same certificate be reused at every step, and the reduction can be stated inside the dynamic program. We show that the symbol layer factorizes each Bellman backup exactly: it prunes mode branching within a backup without altering the value function.

First, we let $\bar{W}$ be a $\bar{k}$-row certificate for $(\bar{E}^d, \bar{P}^d, \bar{B}^d)$ as in Proposition B.2, write $\xi = (s, u)$, and let the one-step constraint in MDP set be

$$\mathcal{C}(s) = \big\{\bar{E}^d z = e^d,\; \bar{P}^d z \le p^d,\; \bar{E}^c \xi = e^c,\; \bar{P}^c \xi \le p^c,\; \bar{B}^d z + \bar{B}^c \xi \le b\big\}.$$

Assume the stage cost is separable and the successor state $s'(\xi)$ does not depend on z. The Bellman operator is $(\mathcal{T}V)(s) = \min_{z \in \{0,1\}^{\bar{n}_d},\, \xi} \{c_d^\top z + c_c^\top \xi + V(s'(\xi)) : \mathcal{C}(s)\}$.

Proposition B.3 (Relative affine-TU Bellman Backup)**.** *Let $\Psi = \bar{W}\{0,1\}^{\bar{n}_d}$, for every V and s,*

$$(\mathcal{T}V)(s) = \min_{\psi \in \Psi} \big[g(\psi) + J_V(s, \psi)\big],\; g(\psi) = \min_{z \in \mathcal{F}^d(\psi)} c_d^\top z,\; J_V(s, \psi) = \min_{\xi \in \mathcal{F}^c(\psi)} \big\{c_c^\top \xi + V(s'(\xi))\big\},$$

with $\mathcal{F}^d, \mathcal{F}^c$ as in Eq. (7). The identity is exact: the value function and its fixed point are unchanged.

Proof. The map W partitions $z \in \{0,1\}^{\bar{n}_d}$ into the subproblems indexed by ψ, so minimizing over z equals minimizing over ψ and then over its subproblem. This step is a rewriting of the feasible set and uses no property of $\bar{W}$.

Fixing ψ on $\{\bar{E}^d z = e^d,\; \bar{W} z = \psi\}$, conditions (1) and (2) of Definition 2.3 give $\bar{P}^d z = \widetilde{P}^d z + U_P \psi + V_P e^d$ and $\bar{B}^d z = U_B \psi + V_B e^d$. The second removes z from the coupling row, leaving $\bar{B}^c \xi \le b - U_B \psi - V_B e^d$; hence the inner feasible set is the product $\mathcal{F}^d(\psi) \times \mathcal{F}^c(\psi)$. Since the stage cost is separable and s' depends only on ξ, the inner objective splits, and the minimum of a separable objective over a product set is the sum of the two minima, giving $g(\psi) + J_V(s, \psi)$.

$\psi \in \Psi$ is integral because $\bar{W} \in \{0, \pm 1\}^{\bar{k} \times \bar{n}_d}$ and z is binary; the right-hand side of $\mathcal{F}^d(\psi)$ is therefore integral. Condition (3) makes the defining system TU, so by Definition 2.1 the polytope is integral and $g(\psi)$ is attained at a binary z even after relaxing z to $[0,1]^{\bar{n}_d}$. Thus no step of the rewriting discards a feasible point or changes an optimal value, and $\mathcal{T}$ is preserved; applying it along $V_t = \mathcal{T} V_{t+1}$ leaves every V_t unchanged. □

Remark B.1. *The backup no longer enumerates the $2^{\bar{n}_d}$ primitive mode assignments: its combinatorial part is a search over Ψ, with $|\Psi| \le (2\bar{n}_d + 1)^{\bar{k}}$, followed by one LP for g and one continuous subproblem for J_V. The factorization acts* inside *each backup and does not shorten the horizon; it is action-level pruning, not temporal abstraction.*

B.5 Guarantee of a τ-Faithfulness

Proposition B.1 is what makes $\psi = Wz$ solver-sufficient in the PWA setting. The statement below is its counterpart for Definition 3.2, and discharges the two requirements of the solver-sufficient symbols on the network side: (i) Searching over symbol assignments preserves optimality up to the solver resolution τ; and (ii) Each assignment reduces the conditioned problem to a single confined descent, with no dependence on where that descent starts.

Proposition B.4 (τ-Descent Faithfulness Guarantee)**.** *Let the symbol map $\Psi := W$ of Eq. (5) induce the conditioned subproblems $R_\psi := \Psi^{-1}(\psi)$, $\psi \in \Psi(\mathcal{X})$, and suppose every R_ψ is τ-descent faithful for h_s. Then for any initializations $x_\psi \in R_\psi$,*

$$\min_{\psi \in W(\mathcal{X})} v^{\tau}_{s,R_\psi}(x_\psi) \;\le\; \min_{x \in \mathcal{X}} h_s(x) + \tau,$$

at a cost of $|W(\mathcal{X})|$ confined descents. If moreover h_s is continuous on $\mathcal{X}$, such a W exists.

Proof. Let $R^\star$ be a subproblem containing a global minimizer, so $\min_{R^\star} h_s = \min_{\mathcal{X}} h_s$; by Definition 3.2 applied to $R^\star$, $v^{\tau}_{s,R^\star}(x) \le \min_{R^\star} h_s + \tau$ for *every* $x \in R^\star$, and the left-hand side, being a minimum over all subproblems, is at most its value at $x_\star$. For existence, uniform continuity of h_s on the compact $\mathcal{X}$ gives $\delta > 0$ with $|h_s(x) - h_s(y)| \le \tau$ whenever $\|x - y\|_\infty \le \delta$, so any symbol map makes each subproblem vary by at most τ and hence τ-descent faithful. □

C Details of CPR for PWA Systems

CPR follows CEGAR: construct a symbolization, verify it, and refine failed candidates. It groups identical columns of $[B^d; E^d; P^d]$ since they do not affect TU verification; write (E, P, B) for the reduced matrices. *Firstly*, starting from $\varnothing$, W_0 appends each row of B whose integer coefficients cannot be recovered from the currently selected rows and E, based on Lemma 2.1. Note the rank lower bound is a compression target, not a guaranteed initialization size.

Given W_i, CPR next removes from P the constraints already explained by the current symbols and E. Specifically, it constructs $\tilde{P}_i := P - U_{P,i}W_i - V_{P,i}E$. CPR then verifies the complete matrix $M_i = [\tilde{P}_i; E; W_i]$ with the TU oracle. If M_i is TU, W_i is certified and returned; otherwise, CPR first attempts a dimension-preserving repair (**Sparsify**-step) and, if needed, refines W_i with an additional commitment (**Split**-step).

1) Sparsify-Step. When M_i is non-TU, CPR first asks whether the failure is caused only by an unfavorable basis of the current symbol space (false obstruction), rather than by a missing symbol. CPR therefore searches for alternative $\{0, \pm 1\}$ bases of the same row space through bounded integer row operations, preserving k while reducing row density. It additionally performs a one-time, budgeted search at the rank floor over $\mathrm{rowspan}([B; E])$. Each candidate is accepted only if bridge recovery remains valid and the recomputed M_i passes the TU oracle; failure of this bounded search does not certify that a larger k is necessary.

2) Split-Step. If the recomputed M_i remains non-TU after **Sparsify**, CPR treats the failure as a genuine obstruction under the current symbolization and introduces an additional commitment through **Split**. It first localizes an inclusion-minimal non-TU core by greedily deleting rows and columns while preserving non-TU, then considers implicated columns that still participate in multi-entry rows of W_i. For each candidate column c, CPR removes its contribution from the existing rows and appends the unit row $e_c^\top$, making $e_c^\top z = z_c$ explicit:

$$W_i^c = \big[\, W_i - (W_i)_{:,c} e_c^\top;\; e_c^\top \,\big].$$

Each trial is residualized again to form its verification matrix $M_i^c = [\widetilde{P}_i; E; W_i^c]$ and is ranked first by TU success, then by row count. The selected split becomes W_{i+1} and re-enters the same

verification loop; once the full RAT conditions pass, the certificate is expanded back to the original column groups and returned.

Algorithm 3 Detailed CPR Algorithm (PWA Systems)

Init: PWA matrices (E^d, P^d, B^d); TU oracle IsTU
Certificate: Relative affine-TU matrix W

Group identical columns of $[B^d; E^d; P^d]$ and retain reduced matrices (E, P, B).
Construct W_0 from directions of B not recoverable from E; ▷ Lemma 2.1
Repeat
  VERIFY
  Construct $\widetilde{P}_i = P - U_{P,i}W_i - V_{P,i}E$.
  $M_i \leftarrow [\widetilde{P}_i; E; W_i]$.
  if IsTU(M_i), **return** W_i.
  ▷ If fails, first test whether the obstruction in W_i is induced by the current basis.
  SPARSIFY
  Re-base W_i within the same row space using bounded integer row operations.
  Optionally search, once and within budget, for a verified $\{0, \pm 1\}$ basis at the rank lower bound.
  Recompute $\widetilde{P}_i$ and M_i. **If** IsTU(M_i), **return** W_i.
  SPLIT
  $O_i \leftarrow$ an inclusion-minimal non-TU core of M_i.
  For each eligible $c \in \mathrm{cols}(O_i)$ **do**
    $W_i^c \leftarrow [\, W_i - (W_i)_{:,c} e_c^\top \,;\; e_c^\top \,]$; remove zero and duplicate rows.
    Recompute $\widetilde{P}_i^c$ and $M_i^c = [\widetilde{P}_i^c; E; W_i^c]$.
  end For
  Select a TU-passing W_i^c if one exists; otherwise select the candidate with the fewest rows.
  $W_{i+1} \leftarrow W_i^c$, $i \leftarrow i+1$. ▷ New symbol, $k \to k+1$
Until W_i is certified or $k = n_d$
Return W_i.

Example C.1. *We consider a PWA system with discrete parameter matrix*

$$E^d = \varnothing, \qquad P^d = [0 \quad 0 \quad -1 \quad 1], \qquad B^d = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & -1 & 1 & 1 \end{bmatrix}.$$

Since $\mathrm{rank}(B^d) = 2$*, Lemma 2.1 gives* $k_{\min} = 2$*.*

At iteration 0*, CPR initializes*

$$W_0 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & -1 & 1 & 1 \end{bmatrix},$$

corresponding to the commitments $\psi_1 = z_3 + z_4$ *and* $\psi_2 = z_1 - z_2 + z_3 + z_4$*. Because* $P^d \notin \mathrm{rowspan}(W_0)$*, we have* $\widetilde{P}_0^d = P^d$*, and*

$$M_0 = \begin{bmatrix} 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & -1 & 1 & 1 \end{bmatrix}.$$

The submatrix on the first two rows and columns $3, 4$ *has determinant*

$$\det \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} = -2,$$

so M_0 *is not TU.*

CPR first applies **SPARSIFY** *by the row operation* $w_2 \leftarrow w_2 - w_1$*, yielding*

$$W_0^{\mathrm{spa}} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 \end{bmatrix}.$$

This preserves the row space and therefore introduces no new symbol. However, $\widetilde{P}_0^d = P^d$ *still holds, and the same determinant-2 obstruction remains between* $-z_3 + z_4$ *and* $z_3 + z_4$*. Thus, re-basing alone cannot remove the failure.*

CPR then applies **SPLIT**. *The minimal non-TU core involves z_3 and z_4; choosing z_3, CPR removes its contribution from the existing aggregate and appends the explicit commitment $e_3^\top z = z_3$:*

$$W_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The resulting symbols are

$$\psi_1 = z_4, \qquad \psi_2 = z_1 - z_2, \qquad \psi_3 = z_3.$$

Now both the bridge and the discrete inequality are explained by W_1:

$$B^d = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} W_1, \qquad P^d = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} W_1.$$

Hence $\widetilde{P}_1^d = 0$, and the verification matrix reduces to

$$M_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

which is TU. CPR therefore returns $W_\star = W_1$ with $k = 3$.

D DETAILS OF NEURAL CPR FOR NETWORK SYSTEMS

Neural CPR follows the same CEGAR structure, but its oracle reads the landscape from samples rather than from a matrix. Let $\mathcal{S}$ be the set of states drawn from a robot demonstration dataset, and h_s the learned objective at state s, minimized over the decision space $\mathcal{X}$. Neural CPR starts from the empty symbol basis $(\mathcal{U}_0, \mathcal{Q}_0) = \varnothing$, so $k = 0$ and all of $\mathcal{X}$ is a single subproblem. For each $s \in \mathcal{S}$ it builds a sample graph G_s by running noisy multi-start descent on h_s and recording the visited decisions as nodes. Edges join consecutive records along a descent and m-nearest neighbours. Each edge is given the argest objective value at its two endpoints and at its midpoint, so an edge cannot report a barrier lower than the segment actually climbs.

At iteration i, the current directions $\mathcal{U}_i$ and quantizers $\mathcal{Q}_i$ assign every node in G_s to its subproblem through $(q_j(u_j^\top x))_{j=1}^k$. Each subproblem R is tested by flooding, a 0-dimensional sublevel-persistence sweep (Edelsbrunner et al., 2002) run with union–find: edges are added in increasing level, so each basin forms a pool at its own lowest value b and merges into a lower pool at the level d of the lowest ridge between them. With a_R the lowest value sampled in R and τ the declared solver resolution, a pool is flagged as violating when $d - b > \tau$, so leaving costs more than the budget, and $b > a_R + \tau$, so settling there already loses more than the budget. Violating subproblems form Φ_i, and CPR returns when $\Phi_i = \varnothing$. Otherwise it attempts the repair that costs no symbol (**SPARSIFY**) before the one that does (**SPLIT**).

1) SPARSIFY-Step. CPR first asks whether a violation reflects the sampling rather than the symbols. A basin may look sealed only because no sample crossed its escape; CPR then seeds confined descents at the flagged saddle. This operation leave k unchanged. Since added evidence can also expose basins that coarse sampling had hidden, $|\Phi_i|$ is not monotone across iterations.

2) SPLIT-Step. Violations that survive are genuine. For each surviving subproblem, CPR takes the lowest pass π between two of its violating basins and measures the gradient flip across two short segments through π, one joining the two basin floors and one along the merging edge; these are sign-aligned and averaged into a normal u_{k+1}, whose ridge point x_π sets the threshold of the new quantizer q_{k+1} at $u_{k+1}^\top x_\pi$. If u_{k+1} falls within an angle tolerance of an existing direction, only a threshold is added and k is unchanged; otherwise a new direction opens and $k \to k + 1$. The loop repeats until no subproblem is violating or a resource cap is reached.

Table 6: **Ablation of Resolution τ.** 200 descent initializations, 600 optimization steps, kNN size 48.

Aligning ($H = 4$)					Avoiding ($H = 8$)				
τ	k	Violating	SR	Time (s)	τ	k	Violating	SR	Time (s)
0.05	6	87%	0.66	2350	0.02	3	91%	0.81	1343
0.125	6	35%	0.72	1801	0.06	2	0%	1.00	612
0.25	4	0%	0.84	962	0.10	2	0%	1.00	327
0.50	2	0%	0.81	194	0.20	0	0%	0.97	14
1.00	0	0%	0.94	38	0.38	0	0%	1.00	6

Table 7: **Ablation of Sampling Evidence.** `Avoiding` with $H = 16$.

kNN	Descent Counts	k	Violating	Time (s)
6	200	1	100%	67
12	200	1	100%	360
24	200	2	83%	692
48	200	2	67%	1105
96	200	1	32%	1203
96	400	1	0%	1514

Algorithm 4 Detailed CPR Algorithm (Network Systems)

Init: objective family $\{h_s\}_{s\in\mathcal{S}}$; resolution τ
Certificate: τ-descent faithful $W := (\mathcal{U}, \mathcal{Q})$

$(\mathcal{U}_0, \mathcal{Q}_0) \leftarrow \varnothing$. ▷ $k = 0$
For each $s \in \mathcal{S}$, build G_s by noisy multi-start descent on h_s.
Set each edge level to the largest value at its endpoints and midpoint.
Repeat
  VERIFY
  Assign the nodes of G_s to subproblems by $(q_j(u_j^\top x))_{j=1}^k$.
  Flood each subproblem R: add edges in increasing level, merging pools.
  Flag a violating pool born at b and merged at d if $d - b > \tau$ and $b > a_R + \tau$.
  $\Phi_i \leftarrow$ the subproblems carrying a flagged pool.
  If $\Phi_i = \varnothing$, **return** $(\mathcal{U}_i, \mathcal{Q}_i)$.
  SPARSIFY
  For each $R \in \Phi_i$, add descents at its violating saddle.
  Recompute Φ_i. **If** $\Phi_i = \varnothing$, **return** $(\mathcal{U}_i, \mathcal{Q}_i)$.
  SPLIT
  For each surviving $R \in \Phi_i$:
    $\pi \leftarrow$ lowest pass between two violating basins of R.
    $(u_{k+1}, x_\pi) \leftarrow$ wall normal and ridge point of the flip across π.
    $\mathcal{U} \leftarrow \mathcal{U} \cup \{u_{k+1}\}$; $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{q_{k+1}\}$ cutting at $u_{k+1}^\top x_\pi$.
  $(\mathcal{U}_{i+1}, \mathcal{Q}_{i+1}) \leftarrow (\mathcal{U}, \mathcal{Q})$, $i \leftarrow i + 1$. ▷ New symbol, $k \to k + 1$
Until $\Phi_i = \varnothing$ or a resource cap is reached
Return $W := (\mathcal{U}, \mathcal{Q})$.

E Further Experimental Results

E.1 Beyond Linear Symbol Axes

Whether linear partitions suffice depends on the feature space in which x is expressed. When it is abstract enough that task-relevant modes are linearly separable, a few cuts $u^\top x$ separate them; when basin boundaries are curved in it, linear cuts can only approximate them piecewise, and more directions are needed. We therefore examine whether richer symbol axes can better capture such nonlinear boundaries. For example, replacing the linear projection $u^\top x$ with a quadratic form can reduce the required symbolic directions while improving success rate. However, increasingly expressive symbol functions transfer complexity from the network into the symbolic layer and weaken interpretability, limiting the value of higher-order representations.

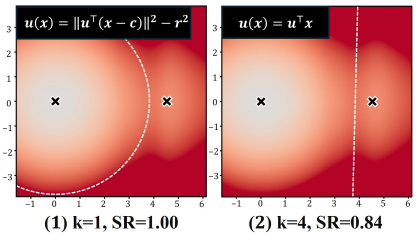


Figure 9: **Quadratic Symbol Axes.** It captures the curved decision boundary with fewer symbolic directions.

E.2 Ablations on Solver Resolution τ

We vary the escape budget τ and report the discovered symbol dimension k, the fraction of conditioned regions that remain *violating* under τ-descent faithfulness, **SR**, and symbol-discovery time (Table 6). The results support interpreting τ as a solver resolution rather than a fitted parameter. When τ falls below within-basin descent variation, CPR over-resolves noise, producing more symbols while leaving many regions unclosed. In an intermediate range, all regions close and k

decreases progressively with little loss in SR. Once τ exceeds the dominant inter-basin barriers, no explicit symbolic distinction is required ($k = 0$). Thus, τ can be selected from the empirical range between within-basin variation and inter-basin separation.

E.3 Ablations on Sampling Evidence.

We next vary the evidence available to neural CPR through graph connectivity and descent initializations (Table 7; we take $\tau = 0.38$, 600 optimization steps). With insufficient evidence, the inferred symbol dimension k changes with the sampling configuration while many conditioned regions remain violating, indicating that the observed partition is not yet supported by the sampled landscape. Increasing the evidence budget progressively reduces the violating fraction, reaching zero with $k = 1$ at 400 initializations. This trend highlights that symbol invention should be interpreted only after the sampled atlas closes: additional evidence resolves uncertain connectivity before the resulting symbolic partition is treated as stable.

F Tasks Details

F.1 Task Requirements

We evaluate six tasks (three 2D manipulation tasks, three in-hand reorientation tasks) with PWA objectives and five tasks (three D3IL tasks (Jia et al., 2024), two real-world tasks) with neural network settings. Task details can be found as follows.

- `MPushT`. 4 pushers translate and rotate a planar T-shaped object. There are a finite number of pushable points on each face. A rollout succeeds when its position and orientation errors are below 0.15 and $0.15\,\mathrm{rad}$, respectively.
- `Hook`. A planar gripper manipulates a finite-volume L-shaped hook to reach around and pull a 0.318-wide square block toward a divider. The task requires collision-free repositioning, establishing a valid hook–block contact, and sustained pulling.
- `Packing`. The controller packs 4–8 randomly selected, heterogeneous rectangles into two stable support stacks inside a one-sided shelf with a 1.70×1.31 interior. Objects begin with randomized orientations and must respect spatial supports and constraints.
- `Block`. A five-finger Shadow Hand reorients an $80\,\mathrm{mm}$ cube while maintaining a feasible multi-finger grasp in the IsaacLab simulator. Its contact structure distinguishes five fingers and six object faces. Terminate when the rotational error is at most $0.10\,\mathrm{rad}$.
- `Cuboid`. The Shadow Hand reorients a $55 \times 70 \times 110\,\mathrm{mm}$ cuboid under the same five-finger setting. Unlike the symmetric cube, its unequal dimensions make fingertip reachability and face transitions orientation dependent. Take $0.10\,\mathrm{rad}$ rotational-error tolerance.
- `Pen`. The hand reorients a capsule-shaped pen of length $200\,\mathrm{mm}$ and diameter $16\,\mathrm{mm}$. Contacts are organized across three finger roles and three axial pen regions, capturing both grasp redistribution and longitudinal contact changes. Take $0.10\,\mathrm{rad}$ success tolerance.
- `Avoiding`. A planar robot endpoint traverses three obstacle levels containing one, two, and three obstacles, separated vertically by $0.18\,\mathrm{m}$. The corresponding 2, 3, and 4 passage choices induce 24 collision-free route modes. Success requires crossing the goal line at $y = 0.35\,\mathrm{m}$ without collision, within a 250-step episode.
- `Pushing`. A robot pushes two $30\,\mathrm{mm}$ boxes into two $50\,\mathrm{mm}$ target regions. Either color-preserving or crossed assignment is accepted when both box–target center distances are at most $50\,\mathrm{mm}$. Combining the two assignments with their two completion orders produces four distinct solution modes.
- `Aligning`. A robot pushes a planar object to a randomized target pose from either its inner or outer side (two task modalities). Initial and target orientations span $[-90°, 90°]$. Success requires position and orientation errors below $18\,\mathrm{mm}$ and $0.048\pi\,\mathrm{rad}$, respectively.
- `Matching`. A robot places multiple red and blue blocks of varying shapes into fixed color-matched baskets in arbitrary order. Success requires all blocks to be placed in the basket of the corresponding color.

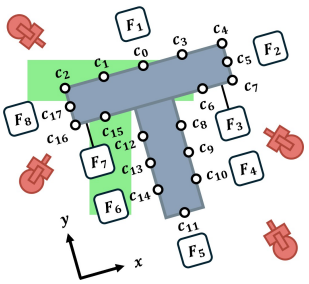


Figure 10: `MPushT` **Contact Domain.** The object boundary is discretized into faces F_i and candidate contact points c_j; the local x–y axes define the object-fixed coordinate frame .

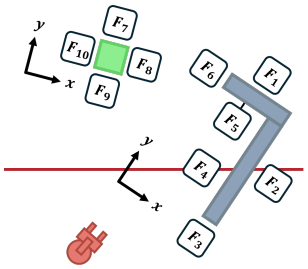


Figure 11: `Hook` **Contact Domain.** The hook and object are represented by multiple discrete contact faces F_i relative to their local x–y coordinate frames.

- `Sorting`. A robot inserts shape-specific blocks (e.g., cross and circle) into a fixed storage box through their matching apertures. Each block fits only its corresponding opening, requiring precise alignment and insertion.

F.2 MILP Setups

In this subsection, we summarize the MILP structure through its primitive discrete variables z, which define the mode and contact choices. We omit the continuous dynamics and full constraint matrices for space. We first outline the rules upon which our formulation is based, and then demonstrate the specific meaning of z (Table 8 and Table 9).

- `MPushT`. A T-shaped object is pushed to a goal pose through 18 candidate contact points distributed over eight faces. Each binary factor z_j activates a unit push at contact c_j along the inward face normal. Contacts on the same face are activated cumulatively to form contiguous pushing segments, with at most four faces active at each step (Fig. 10).
- `Hook`. A gripper manipulates a hook to pull a block toward a divider wall. The 16 binary factors encode whether the hook is held, hook–block contact, their relative placement and facing surfaces, and whether the block can reach the wall within the current step (Fig. 11).
- `Shelf`. Multiple objects are packed onto a shelf through discrete pick-and-place primitives and we take 8 objects here as an example. The 41 factors consist of eight grasp actions, placements on the shelf floor, and stacking actions among objects. An object must be placed only on a larger supporting object.
- `Block`. A five-finger hand reorients a cube through a discrete finger–face contact graph. We denote that, ff: index finger; mf: middle finger; rf: ring finger; lf: little finger; th: thumb. The 20 factors represent active fingers, admissible finger–face assignments, contacted object faces, and one idle unit. Flow constraints allow multiple simultaneous contacts, with each active finger assigned to one face and no face contacted twice.
- `Cuboid`. The cuboid task uses the same 20-factor finger–face representation as `Block`. The unequal object dimensions change the continuous contact feasibility and reachable orientations, while the discrete factors retain the same finger, face, and finger–face semantics.
- `Pen`. A five-finger hand reorients a pen using three functional finger groups—opposition, medial, and ulnar—and three axial contact zones—proximal, center, and distal. The 12 factors encode active groups, group–zone assignments, and occupied zones; two groups contact two distinct zones at each step.

F.3 `MPushT` MILP Variations

To study why certain human-defined concepts are useful, we further construct nine expert-designed MILP formulations of the same `MPushT` task, containing 127 primitive contact factors in total. Holding the task fixed while varying its discrete factorization allows us to analyze the computational role of different human-defined distinctions. We summarize the factors below.

Table 8: **Discrete Factors for 2D Manipulation.**

Task	Factors	Semantics
MPushT	$z_{0:4}$	F_1, push along $-y$; cumulative runs $c_0 \to c_4$
	z_5	F_2, push along $-x$ at c_5
	$z_{6:7}$	F_3, push along $+y$ from c_6 to c_7
	$z_{8:10}$	F_4, push along $-x$ at c_8 and along $c_9 \to c_{10}$
	z_{11}	F_5, push along $+y$ at c_{11}
	$z_{12:14}$	F_6, push along $+x$ from c_{12} to c_{14}
	$z_{15:16}$	F_7, push along $+y$ from c_{15} to c_{16}
	z_{17}	F_8, push along $+x$ at c_{17}
Hook	$z_{0:1}$	hook released / held by the gripper
	$z_{2:3}$	hook separate from / in contact with the block
	$z_{4:9}$	relative block–hook placement: body-frame axis, side, and active hook surface
	$z_{10:13}$	block face oriented toward the hook: body-frame axis and sign
	$z_{14:15}$	block cannot / can reach the divider wall within the current step
Shelf	$z_{0:7}$	grasp object o
	$z_{8:15}$	place o on the shelf floor
	$z_{16:40}$	place o on o'; permitted only when o is smaller than o'

Table 9: **Discrete Factors for In-Hand Reorientation.**

Task	Factors	Semantics
Block	$z_{0:4}$	active finger: ff, mf, rf, lf, or th
	$z_{5:12}$	admissible finger–face assignment
	$z_{13:18}$	contacted face: palm-facing, dorsal, thumb-side, little-side, distal, or proximal
	z_{19}	idle unit, allowing only two active finger contacts
Cuboid	$z_{0:19}$	same finger, finger–face, face, and idle-factor semantics as Block; contact feasibility follows the cuboid geometry
Pen	$z_{0:2}$	active finger group: opposition, medial, or ulnar
	$z_{3:8}$	finger-group assignment to proximal, center, or distal pen regions
	$z_{9:11}$	occupied pen zone: proximal, center, or distal

Table 10: **Alternative Expert-Designed MPushT Factorizations.** P_0 is described in Table 8.

Form.	Factors	Semantics
P_1	$z_{0:2}$	F_1: push at c_0, extend leftward to c_1, or move to c_3
	$z_{3,4,7,10,11}$	single pushes on F_2, F_3, F_5, F_7, F_8 at $c_5, c_6, c_{11}, c_{15}, c_{17}$
	$z_{5:6}$	F_4: push at c_8 or move to c_9
	$z_{8:9}$	F_6: push at c_{12} or extend to c_{13}
P_2	$z_{0:7}$	one occupancy bit per face $F_1, \ldots, F_8$, pushing at its innermost contact point
P_3	$z_{0:3}$	push from the top, bottom, right, or left side
	$z_{4:5}$	global counter-clockwise / clockwise spin; the first active side shifts its contact to realize the requested rotation
P_4	$z_{0:3}$	single pushes from the top, bottom, right, or left
	$z_{4:7}$	two-point couples for counter-clockwise ($z_{4:5}$) or clockwise ($z_{6:7}$) rotation
P_5	z_0	no push
	$z_{1:26}$	one-hot unit-effect targets $(f_x, f_y, s) \in \{-1, 0, +1\}^3 \setminus \{(0, 0, 0)\}$, decoded to the fewest P_0 contacts realizing the target
P_6	$z_{0:17}$	same contact-point factors as P_0, but with at most two active faces per step

(Continued on next page)

Form.	Factors	Semantics
P_7	$z_{0:7}$	same face-occupancy factors as P_2
	$z_{8:17}$	face-specific lever offsets that move or extend the active contact along F_1, F_3, F_4, F_6, or F_7
P_8	$z_{0:7}$	same face-occupancy factors as P_2
	$z_{8:11}$	one / two units of global counter-clockwise ($z_{8:9}$) or clockwise ($z_{10:11}$) spin, realized by shifting the first eligible active face