HYBRID REASONING THROUGH COMPOSITIONAL STRUCTURAL ENERGY MINIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Many real-world reasoning problems require solving for high-level discrete structure with continuous values. For example, a robot solving a manipulation problem must determine both a high-level task structure and continuous actions; a model for scene understanding infers both the objects' pairwise relations and their poses from a partial camera view. Existing machine learning approaches face important limitations: end-to-end monolithic models often struggle to solve problems that are harder than those seen during training, while compositional methods typically require the correct problem structure to be specified in advance. We introduce Compositional Structural Energy Minimization (CSEM), a general framework that learns from small training instances and jointly solves for the discrete structure and continuous solutions of harder problems. We evaluate CSEM on a wide range of hybrid reasoning problems, including task-and-motion-planning, 3D scene understanding, and reconstructing physical trajectories and contact modes from sparse observations. We demonstrate that CSEM generalizes without additional training to larger structures and longer horizons, outperforming strong learning-based baselines.

1 INTRODUCTION

Reasoning extends far beyond formal mathematics and logic. Many real-world tasks – including visual inference, robotics planning, and decision-making – can be viewed as reasoning problems because they require finding solutions that satisfy interacting constraints. These problems are challenging as they require machine-learned models to go beyond pattern recognition and identify the rules and constraints that determine a valid solution. In recent years, considerable attention has been devoted to applying generative models to reasoning problems across many domains (Lewkowycz et al., 2022; Trinh et al., 2024; Janner et al., 2022). A common approach is to train a monolithic generative model that maps each problem directly to its solution (Ajay et al., 2023; Høeg et al., 2026). Although these models can perform well on problems similar to those seen during training, they often struggle when test problems are larger, more complex, or otherwise outside the training distribution (Zhang et al., 2023; Dziri et al., 2023).

A promising alternative is compositional generative modeling, which learns reusable models of local concepts, constraints, or subproblems and combines them at inference time. Energy-based models (EBMs) provide a natural realization of this idea: their energies and gradients can be combined to form a joint objective over a larger problem. Prior work has used this property to generate images satisfying novel combinations of visual concepts (Du et al., 2020), solve packing, stacking, and task-and-motion problems from known constraint graphs (Yang et al., 2023), arrange objects by composing learned spatial relations (Xu et al., 2025), and scale combinatorial solvers to larger instances of N-Queens, SAT, graph coloring, and crosswords (Oarga & Du, 2025).

However, these compositional methods require knowing exactly *how* to compose. In many problems, identifying the right decomposition is itself a nontrivial part of reasoning. We call these problems *hybrid reasoning*. Consider bin packing, where several objects must be placed into bins of different sizes. Given a valid object-to-bin assignment, the problem can be naturally decomposed into constraints requiring each object to remain inside its assigned bin and objects in the same bin not to intersect. The assignment, however, is not known in advance, and the search space is exponentially large. Each possible assignment produces a different composition and, consequently, a

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.

different energy landscape over the continuous object poses. Thus, before optimizing a continuous solution, the model must first determine the discrete structure that defines which constraints should be composed, but the quality of the discrete structure is unknown before we have a continuous solution, leading to a circular conundrum.

We address this challenge by treating both the discrete structure G and the continuous variables $\mathbf{y}$ as unknowns. During training, we learn energy factors from small subproblems, with each factor representing one local constraint. Each factor serves two roles: its gradients refine the continuous variables toward satisfying the constraint, while its clean energy is calibrated to indicate whether that constraint is satisfied. Given a proposed structure G, we activate and sum the corresponding energy factors to construct a global energy $E_\theta(\mathbf{x}, \mathbf{y}; G)$. Thus, the graph determines the landscape optimized over $\mathbf{y}$, and changing the graph recomposes the objective itself.

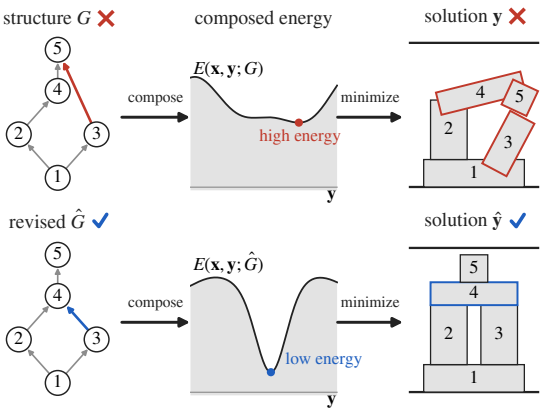


Figure 1: **Hybrid reasoning through energy-guided structural revision.** An initial support graph G induces a high-energy landscape and an unstable stacking $\mathbf{y}$. Our method revises the support relation and locally backtracks the affected poses, producing a lower-energy landscape and a stable solution $(\hat{G}, \hat{\mathbf{y}})$.

At inference time, we initialize any structurally plausible graph G_0 as an initial guess, and minimize its corresponding composed energy. We then inspect the local factor energies: a high-energy factor identifies a poorly satisfied constraint and the corresponding structural decision to reconsider. We sample valid local edits at that decision and evaluate them in parallel. For each edit, we backtrack the variables involved in the changed factors to an earlier diffusion timestep while retaining the remaining context, then minimize the recomposed landscape. We keep the candidate with the lowest aggregate factor energy and repeat this "minimize, verify, recompose" loop until all constraints are satisfied or the search budget is exhausted.

We illustrate the effectiveness of CSEM on three domains: task-and-motion planning, 3D scene graph inference, and physical mode inference. Across these domains, we train on small instances and evaluate generalization to larger structures, longer horizons, and more challenging observation conditions. For example, in multi-bin packing, CSEM trains on two-bin scenes with 4–8 objects and achieves 52% success on five-bin scenes with 11–19 objects, compared with $\sim$0% for the baselines. Across the evaluated domains, CSEM achieves substantial gains over learning-based baselines, particularly on larger planning problems and scene inference with occlusion. We further show that increasing the structural-edit budget improves packing success, enabling the same trained model to benefit from additional inference-time computation. We also ran ablations to demonstrate the contributions of structural revision, energy calibration, and the timing of verification.

Our contributions are threefold. **(1)** We formulate hybrid reasoning as compositional energy minimization in which the discrete structure determining which factors to compose is inferred jointly with the continuous solution. **(2)** We introduce CSEM, an inference framework that uses calibrated local energies to guide structural revision and continuous refinement, with local backtracking to adapt solutions to the revised composition. **(3)** We evaluate CSEM on task-and-motion planning, 3D scene inference, and trajectory reconstruction with latent physical modes, demonstrating generalization without retraining to larger instances and longer horizons, with substantial improvements over learned baselines across these domains.

2 Related Work

Reasoning as optimization. Reasoning extends beyond mathematical problem solving: many real-world inference, decision-making, and planning tasks can be formulated as constrained optimization over latent variables. In robotics, task-and-motion planning reasons over symbolic search with geometric optimization (Toussaint, 2015; Garrett et al., 2020; Agia et al., 2023), while prob-

abilistic scene models jointly optimize latent structure and object poses to explain visual observations (Gothoskar et al., 2021; 2023; Zhou et al., 2023). These methods are effective but often rely on domain-specific models or predefined structural assumptions. Learning-based approaches instead learn optimization procedures or objectives from data, including energy-based reasoners (Du et al., 2022; 2024) and graph neural network solvers (Cappart et al., 2023; Schuetz et al., 2022). However, they typically learn task-level predictors or objectives tied to the training distribution, limiting generalization to larger or harder instances. We pursue a compositional alternative: learning reusable energy landscapes for small subproblems and composing and revising them at inference time to solve harder tasks.

Energy-based and diffusion models for composition. There has been a line of work in recent years on compositional generative models. In particular, energy-based models support composition by adding component energies, corresponding to products of distributions (Du et al., 2020). For robotics constraint satisfaction problems, Diffusion-CCSP (Yang et al., 2023) composes learned constraint models to solve larger continuous problems; for puzzle solving, Oarga & Du (2025) composes subproblem energies and uses population-based optimization to improve sampling; for long-horizon planning, several methods compose short trajectory segments into long plans (Luo et al., 2025; Mishra et al., 2026; Sun et al., 2026). These approaches achieve out-of-distribution generalization, but they assume that *how* to compose is known. We provide a novel perspective in this line of work – we treat the compositional structure itself as unknown and use the calibrated energy values to infer how the energies should be composed.

Inference-time scaling through guidance and search. Diffusion models can exploit additional inference-time computation through guidance (Ye et al., 2024), particle-based sampling and resampling (Wu et al., 2023; Li et al., 2025), and verifier-guided search (Zhang et al., 2026). In particular, (Zhang et al., 2026) provides a general classical search framework for inference-time scaling. Diffusion Forcing method (Chen et al., 2024a) provided the flexibility that allows variables to be denoised asynchronously at different noise levels. Our method combines these ideas in a structural-revision loop that searches over factor compositions, revises implicated variables locally, and backtracks, to optimize for the discrete and continuous solutions.

3 METHOD

In this section, we develop a general framework for learning and solving *hybrid reasoning* tasks that require reasoning about both discrete structures and continuous values. We will first formally define hybrid reasoning and demonstrate how a wide range of domains can fall into this category. Then we describe our method, which enables us to train on simple problems and solve much harder hybrid reasoning problems at inference time.

3.1 FORMULATION AND REPRESENTATION OF HYBRID REASONING PROBLEMS

Prior energy-based reasoning methods cast a task as minimizing a *fixed* objective over continuous outputs or continuous relaxations of discrete ones (Oarga & Du, 2025; Yang et al., 2023). In a *hybrid reasoning* problem, the objective itself depends on a discrete structure that must be inferred together with the continuous values that realize it. In this subsection, we first recall the fixed-objective setting, then formally define hybrid reasoning as an optimization problem.

Reasoning as energy minimization. A reasoning task maps an input $\mathbf{x} \in \mathcal{X}$ to a solution $\mathbf{y} \in \mathcal{Y}(\mathbf{x})$, where an input may admit multiple valid solutions. We represent the task using an energy function $E_\theta(\mathbf{x}, \mathbf{y})$ that assigns lower energy to valid solutions than to invalid ones. Inference is performed by

$$\hat{\mathbf{y}} = \arg\min_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} E_\theta(\mathbf{x}, \mathbf{y}). \tag{1}$$

This optimization can be approximated using gradient descent:

$$\mathbf{y}^t = \mathbf{y}^{t-1} - \lambda \nabla_{\mathbf{y}} E_\theta\left(\mathbf{x}, \mathbf{y}^{t-1}\right), \tag{2}$$

where λ is the step size and $\mathbf{y}^0$ is sampled from an initial distribution, such as a Gaussian. After T steps, we use $\mathbf{y}^T$ as an approximate solution.

Hybrid reasoning as compositional energy minimization. In a hybrid reasoning problem, a solution consists of a discrete graph G and continuous variables $\mathbf{y} = \{\mathbf{y}_v\}_{v \in V}$. The task specifies a set $\mathcal{G}(\mathbf{x})$ of structurally plausible graphs, whose requirements are typically easy to define. For example, in bin packing, G is a bipartite graph between objects and bins, and structural plausibility requires

each object to be assigned to exactly one bin. Note that a structurally plausible graph (e.g., putting all objects into the smallest bin) does not mean that it leads to a valid solution.

We represent local constraints using a small library of reusable energy functions. For a given problem instance, let $E^i_\theta(\mathbf{x}_i, \mathbf{y}_{S_i})$ denote an instantiated energy factor acting on variables $S_i \subseteq V$, and let $z_i(G) \in \{0, 1\}$ indicate whether this factor is activated by G. The graph defines the composed energy

$$E_\theta(\mathbf{x}, \mathbf{y}; G) = \sum_{i=1}^{L} z_i(G)\, E^i_\theta\left(\mathbf{x}_i, \mathbf{y}_{S_i}\right). \tag{3}$$

Changing G therefore changes which factors are included and recomposes the energy landscape over $\mathbf{y}$. Hybrid reasoning corresponds to the joint optimization

$$(\hat{G}, \hat{\mathbf{y}}) = \arg \min_{\substack{G \in \mathcal{G}(\mathbf{x}) \\ \mathbf{y} \in \mathcal{Y}(\mathbf{x})}} E_\theta(\mathbf{x}, \mathbf{y}; G). \tag{4}$$

A naive solution would enumerate every $G \in \mathcal{G}(\mathbf{x})$ and optimize its continuous variables using equation 2. This is generally intractable because the number of valid graphs can grow exponentially. For example, assigning 10 objects to 5 bins produces 5^{10} plausible assignment graphs; naively trying each of them will not be tractable.

3.2 Training EBMs for Sampling and Verification

CSEM assumes a domain-level vocabulary of local constraints, but not the decomposition of a particular problem instance. Each constraint type is represented by an energy factor acting only on the variables it involves, while the inferred graph determines which factor instances are active. For stacking, these factors model containment, collision avoidance, and support. For 3D scene inference, relation factors score candidate spatial and parent relations, while an observation factor measures consistency with the RGB-D input. For temporal inference, mode-specific factors model local flight, impact, rolling, and resting dynamics. These local subproblems are specified directly from the domain constraints and learned independently from small training examples.

Regardless of the domain, each energy factor must support two operations: its gradients must refine continuous variables toward valid configurations, and its clean energy must indicate whether its constraint is satisfied.

Sampling with diffusion energy models. We train the energy gradients using denoising score matching (Du et al., 2023). Because inference may backtrack different variables by different amounts, we follow Diffusion Forcing (Chen et al., 2024a) and assign each variable its own timestep.

For a valid sample $\mathbf{y}^+$, we independently add noise to each variable:

$$\tilde{\mathbf{y}}_k = \sqrt{\bar{\alpha}_{t_k}}\, \mathbf{y}^+_k + \sqrt{1 - \bar{\alpha}_{t_k}}\, \boldsymbol{\epsilon}_k, \qquad \boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{5}$$

Factor i is trained by

$$\mathcal{L}^i_{\mathrm{DSM}} = \mathbb{E}\left[\sum_{\substack{k \in S_i \\ t_k > 0}} \left\| \sqrt{1 - \bar{\alpha}_{t_k}}\, \nabla_{\tilde{y}_k} E^i_\theta\left(x_i, \tilde{y}_{S_i}, t_{S_i}\right) - \epsilon_k \right\|_2^2\right]. \tag{6}$$

The noise scales multiply the corresponding variable-wise gradients, allowing a scalar energy to parameterize noise predictions under unequal corruption levels. Independent timestep sampling exposes each factor to the mixed-noise configurations encountered during local backtracking.

Verification by calibrating the clean energy. The denoising objective in equation 6 learns continuous refinement directions but does not ensure that invalid assignments have higher energy than valid ones. Prior work uses contrastive learning supervision to shape raw energy values Du et al. (2024); Oarga & Du (2025). Although these values can rank candidates for a single type of EBM, their magnitudes may not be comparable across independently trained EBM types, i.e., a large composed energy may therefore reflect either a constraint violation or a difference in scale, making it unhelpful to predict whether a sampled solution is successful.

We therefore calibrate each factor at the clean timestep. Starting from positive samples, we construct negative samples by perturbing the variables involved in the constraint. Let $c \in \{0, 1\}$ indicate

whether a clean sample satisfies the constraint. We train the negative energy as its validity logit:

$$\mathcal{L}^i_{\text{cal}} = -\mathbb{E}\left[c \log \sigma\big(-E^i_\theta(\mathbf{x}_i, \mathbf{y}_{S_i}, \mathbf{0})\big) + (1-c)\log\big(1-\sigma\big(-E^i_\theta(\mathbf{x}_i, \mathbf{y}_{S_i}, \mathbf{0})\big)\big)\right]. \tag{7}$$

Here $\sigma(z) = 1/(1+e^{-z})$ is the logistic sigmoid, so $\sigma(-E^i_\theta)$ is the calibrated probability that constraint i holds. We apply this supervision only at $\mathbf{t} = \mathbf{0}$ because adding noise can move a sample across the constraint boundary and invalidate its original label. Since every factor uses the same binary target, calibration anchors their clean energies to a common scale: lower energy indicates a more likely valid constraint.

The complete objective is $\mathcal{L}^i = \mathcal{L}^i_{\text{DSM}} + \lambda_{\text{cal}}\mathcal{L}^i_{\text{cal}}$. Thus, the same energy supplies denoising gradients during sampling and clean values for localizing failures, comparing candidates, and deciding when inference should terminate.

3.3 Energy-Guided Graph Search with Local Backtracking

Given an input $\mathbf{x}$, we aim to solve for a graph G and continuous variables $\mathbf{y}$. The two are coupled: G determines the energy landscape used to minimize $\mathbf{y}$, but G can be evaluated only after $\mathbf{y}$ has adapted to that landscape. We solve this problem by repeatedly minimizing under one initial graph, identifying a failed local relation, editing that relation, and locally backtracking the affected variables. We can start with any structurally plausible graph $G \sim \mathcal{G}(\mathbf{x})$.

Time-conditioned energy minimization. Given a proposed graph G, let $\mathcal{I}(G) = \{i : z_i(G) = 1\}$ denote the active factors. During inference, their time-conditioned composition is

$$E_\theta(\mathbf{x}, \mathbf{y}, \mathbf{t}; G) = \sum_{i \in \mathcal{I}(G)} E^i_\theta\left(\mathbf{x}_i, \mathbf{y}_{S_i}, \mathbf{t}_{S_i}\right). \tag{8}$$

For fixed G, we minimize this energy while annealing $\mathbf{t}$ to $\mathbf{0}$ using prior inference-time energy minimization methods (Oarga & Du, 2025; Zhang et al., 2026), with details in Appendix A.

Verify and edit. After minimization, we evaluate each active factor at the clean timestep:

$$e_i(G, \mathbf{y}) = E^i_\theta\left(\mathbf{x}_i, \mathbf{y}_{S_i}, \mathbf{0}\right). \tag{9}$$

A high e_i indicates a likely failed relation. To stochastically edit our graph, we sample the relation to edit according to

$$p(i \mid G, \mathbf{y}) = \frac{\exp(e_i/\tau)}{\sum_{j \in \mathcal{I}(G)} \exp(e_j/\tau)}. \tag{10}$$

The selected factor determines where the graph should change, while the domain provides its syntax-valid alternatives. We sample H such edits and evaluate them in parallel.

Locally backtrack after an edit. Suppose an edit changes G to G'. Define the changed factors and their incident variables by

$$\Delta(G, G') = \{i : z_i(G) \neq z_i(G')\}, \qquad \mathcal{A}(G, G') = \bigcup_{i \in \Delta(G, G')} S_i. \tag{11}$$

Variables affected by the edit are backtracked farther than the remaining context variables:

$$b_k(G, G') = \begin{cases} t_{\max}, & k \in \mathcal{A}(G, G'), \\ t_{\text{part}}, & k \notin \mathcal{A}(G, G'), \end{cases} \qquad t_{\max} > t_{\text{part}} \geq 0. \tag{12}$$

Starting from the clean solution under G, we re-noise each variable to its assigned timestep:

$$\tilde{\mathbf{y}}_k = \sqrt{\bar{\alpha}_{b_k}}\,\mathbf{y}_k + \sqrt{1-\bar{\alpha}_{b_k}}\,\boldsymbol{\epsilon}_k, \qquad \boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{13}$$

A factor connecting an edited variable to a context variable therefore receives mixed timesteps, such as $(t_{\max}, t_{\text{part}})$, which are included in the training objective of Sec. 3.2. Local backtracking allows affected variables to escape a basin associated with the previous graph while retaining useful context from the current solution.

The edited graph G' defines a revised, time-conditioned energy landscape $E_\theta(\mathbf{x}, \mathbf{y}, \mathbf{t}; G')$. Starting from the perturbed state $\tilde{\mathbf{y}}$, we apply annealed energy-guided refinement to obtain $\mathbf{y}'$ (Appendix A.4).

We then evaluate the clean factor energies of the candidate $(G', \mathbf{y}')$ using Eq. equation 9. These scores determine candidate selection and termination, as described below; if inference continues, the retained candidate undergoes another round of verification, structural revision, and local backtracking.

Selection and termination. Each round scores the edited proposals and the current candidate by an aggregate of their calibrated clean factor energies, $S(G, \mathbf{y}) = \mathrm{agg}_{i \in \mathcal{I}(G)} \sigma(e_i)$, where σ is the sigmoid of Eq. equation 7 and agg can be any monotone aggregator such as the mean or the maximum (Appendix A.5 gives the per-domain choice), and retains the candidate with the lowest score. Including the current candidate ensures that the score cannot increase. We stop when the largest calibrated factor energy $\max_i \sigma(e_i)$ falls below δ, or when the edit budget is exhausted, and return the best candidate encountered.

4 Task-and-Motion Planning Experiments

We evaluate CSEM on three long-horizon task-and-motion planning domains: 3D stacking, labeled-box rearrangement, and 3D multi-bin packing, as illustrated in Figure 2. Each domain requires jointly selecting a discrete task structure and continuous object configurations that satisfy geometric or physical constraints.

We train all the local energy factors on subproblems containing at most four objects. At inference time, we evaluate CSEM zero-shot on substantially larger structures. We compare our method against transformer-based planners trained on the same small-instance distributions, as well as existing compositional planners. We show that our method consistently achieves higher solve rates and retains substantially stronger performance as the problem size increases.

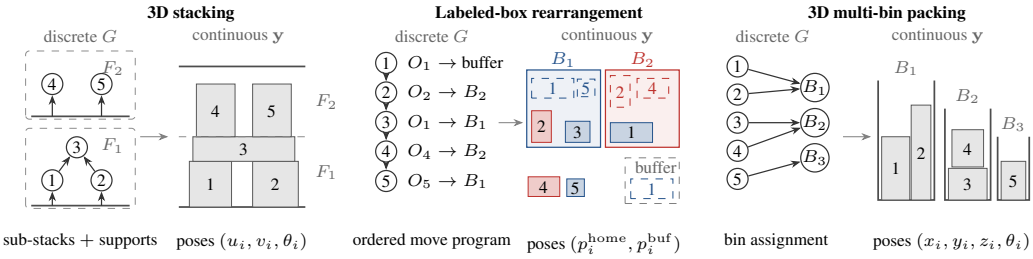


Figure 2: **Three hybrid planning domains.** Each task couples a discrete structure with continuous geometric variables. In stacking, the structure decomposes the frame and objects into bounded-size sub-stacks. In rearrangement, each misplaced object must be placed into its correct box without collision during the process, where the discrete structure specifies the order of the moving sequence. In packing, an assignment partitions objects among bins.

- **3D stacking.** Given a shelf and n blocks of varying sizes, the robot must place all blocks within the shelf while maintaining a stable stack. The discrete structure decomposes the frame and blocks into sub-stacks containing at most four blocks, while the continuous variables specify the planar poses $\mathbf{x}_i = (u_i, v_i, \theta_i)$ within each subproblem. The local energy factors enforce containment, collision avoidance, support, and sub-stack stability. We train with $n \leq 4$ and evaluate on $n \in \{4, \ldots, 8\}$, with all larger sizes tested zero-shot.
- **3D multi-bin packing.** Given K open-top bins and n blocks of varying sizes, the robot must place every block inside a bin without collisions. The discrete structure assigns each block to exactly one bin, while the continuous variables specify its pose $\mathbf{p}_i = (x_i, y_i, z_i, \theta_i)$. The local energy factors enforce bin containment and pairwise collision avoidance, while a permutation-equivariant set factor models the joint packability of 2–4 blocks assigned to the same bin. We train with K=2 and $n \in [4, 8]$, and evaluate zero-shot with $K \in \{3, 4, 5\}$ and up to 19 blocks.
- **Labeled-box rearrangement.** Given k labeled boxes, one buffer region, and n labeled objects initially misplaced, the robot must move every object into its matching box, using the buffer when necessary and avoiding collisions throughout execution. The discrete structure is an ordered rearrangement program that determines whether each object moves directly home or through the buffer, as well as the order of these moves; the continuous variables specify its home and buffer poses. The local energy factors enforce containment and collision-free placement at every step. We train with $(n, k) = (4, 2)$ and evaluate zero-shot on scenes containing up to $(n, k) = (12, 3)$.

Table 1: **Solve rate (%) when problem size grows beyond training.** Each method is trained only at the ID size and evaluated zero-shot on larger OOD problems. n: number of objects; k: number of labeled boxes; K: number of bins (small numbers give objects or blocks per scene). Success means PyBullet build-up survival for stacking and strict move-by-move replay for rearrangement and packing. Each method returns one plan per scene, and every column uses the same 100 scenes for all methods. *w/o edits* keeps CSEM's sampler and energy-based selection but performs no structural edits. 95% confidence intervals ±10 points.

	3D stacking					**Labeled-box rearrangement**						**3D multi-bin packing**			
	objects n					boxes k (objects n)						bins K (blocks n)			
	ID	OOD (larger) ⟶				ID	OOD (larger) ⟶					ID	OOD (larger) ⟶		
	4	5	6	7	8	2 4	2 6	2 8	2 10	3 6	3 12	2 4–8	3 6–12	4 8–15	5 11–19
Diffuser	71	33	9	5	7	23	1	1	0	5	0	1	0	0	0
HD	61	17	13	7	1	23	1	0	0	2	0	1	0	0	0
CHD	47	16	12	5	4	20	0	0	0	0	0	1	0	0	0
HDP	66	38	38	27	14	21	0	0	0	0	0	33	0	0	0
ccsp-rand	31	18	15	11	7	–	–	–	–	–	–	13	2	1	0
CDGS	45	31	20	10	9	29	2	1	0	15	0	5	1	0	0
CSEM (ours)	**76**	**68**	**51**	**53**	**49**	**65**	**66**	**42**	**39**	**58**	**25**	**92**	**77**	**63**	**52**
w/o edits (ablation)	69	63	49	52	42	61	48	29	25	47	14	80	43	17	8

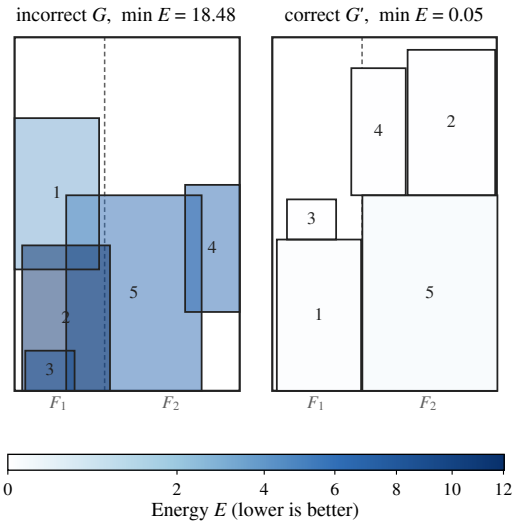


Figure 3: **Energy under an incorrect and the correct structure.** A five-block stacking scene solved by CSEM, with poses minimized under an incorrect sub-stack partition G (left) and the correct partition G' (right). Blocks are colored by their factor energies; the incorrect G leaves the minimum total energy high.

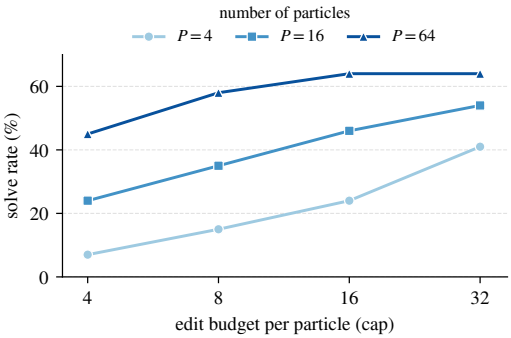


Figure 4: **Increased compute:** Packing at K=5: solve rate vs. edit budget per particle for three population sizes P.

Edit at	K=2	3	4	5
t=0 (CSEM)	**92** (3k)	**77** (5k)	**63** (11k)	**52** (19k)
t_e=27	83 (6k)	58 (12k)	36 (16k)	20 (19k)
t_e=54	83 (12k)	57 (16k)	33 (18k)	13 (18k)

Table 2: **Ablation (early editing)**: Edit the structure at an intermediate timestep t_e (packing). Solve rate (%), network evaluations per scene in parentheses.

Baselines. We compare against Diffuser (Janner et al., 2022), Hierarchical Diffuser (HD) (Chen et al., 2024b), Coupled Hierarchical Diffuser (CHD) (Hao et al., 2025), and Hybrid Diffusion Planner (HDP) (Høeg et al., 2026). We additionally evaluate CCSP (Yang et al., 2023) with a random or first-fit structure (*ccsp-rand*) on stacking and packing, and Compositional Diffusion with Guided Search (CDGS) (Mishra et al., 2026) with its original chain factorization and search procedure. CCSP has no fair rearrangement arm because specifying its constraint graph would reveal the move sequence.

Quantitative results. CSEM achieves the highest solve rate in every column of Table 1 (compute per scene in Appendix B.4). In particular, while other baselines' success rates drop quickly at out-of-distribution evaluation scales, our method maintains much stronger success rates across each domain. Thus, the performance margin increases under count-OOD generalization.

Qualitative results. We present in Figure 3 a 3D bin packing instance with corresponding energy values. We visualize the energy map in 3D and show that when the discrete structure (bin assign-

ment) is incorrect, the continuous solution given by the minimum energy is still invalid; whereas when the bin assignment is correct, the corresponding energy map gives low minimum value, which yields a valid discrete-continuous solution.

Scaling with inference-time computation. CSEM exposes test-time computation through its particle count and edit budget. In Figure 4, we report the performance of our approach on the bin packing problem (5 bins with up to 19 objects) with increasing number of particles and structural edit budget during sampling. We found that as we increase the structural edit budget as well as the number of particles, the solve rates significantly improve.

Ablations. Table 2 studies early editing, where the algorithm uses a Tweedie estimate to evaluate the solution and revise its structure before denoising finishes. This variant lowers the solve rate and increases total computation because less reliable edits require more rounds. The final row of Table 1 removes structural revision entirely by composing energies under a randomly sampled graph G. Its substantially lower solve rate confirms that revising the discrete structure is essential to hybrid reasoning. Additional ablations are in Appendix B.4.

5 CSEM FOR VISUAL INFERENCE

In this section, we show that our method can be applied to problems much beyond task-and-motion planning. We demonstrate how CSEM can be applied to visual inference, where the goal is to recover the latent discrete structure and continuous states that jointly explain incomplete observations. We demonstrate this perspective in two domains – 3D scene-graph inference and physical mode inference – showing that CSEM provides a unified framework for hybrid reasoning tasks with partial observations.

5.1 3D SCENE GRAPH INFERENCE

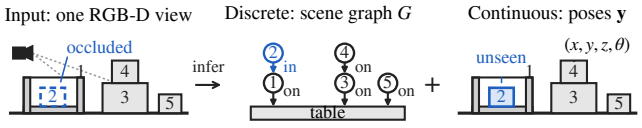


Figure 5: **3D scene inference as hybrid reasoning.** From a partial RGB-D view and given object sizes, CSEM infers the discrete support graph and continuous object poses.

Consider a tabletop scene containing multiple objects with various spatial relations. Given a single RGB-D view of the scene and known object sizes, the task is to jointly infer its latent relation graph and each object's continuous pose (Fig. 5). We are particularly interested in inferring scenes with occlusions.

We cast the task as hybrid reasoning: the discrete structure specifies each object's support parent and pairwise relations, while the continuous variables specify its pose $\mathbf{p}_i = (x_i, y_i, z_i, \theta_i)$. CSEM composes local factors for support, containment, collision avoidance, camera-relative relations, and consistency with visible points. We train on 3,000 scenes with 2–4 objects and evaluate zero-shot under increasing object counts and heavy occlusion.

We compare against an FF transformer with relation prediction (Wald et al., 2020) and pose regression (Melas-Kyriazi et al., 2024), and JointDiff (Capellera et al., 2026). As shown in Table 3, CSEM substantially improves hybrid accuracy across most OOD settings, especially under occlusion. Removing structural edits sharply reduces performance, showing that the gains arise from structural search rather than the learned energies alone.

	More objects				Occluded		
	ID	OOD ⟶			OOD ⟶		
objects n	4	6	8	10	4	6	8
FF transformer	61	56	44	**40**	30	22	18
JointDiff	54	30	18	11	38	19	15
CSEM (ours)	**77**	**68**	**45**	32	**57**	**48**	**35**
w/o edits (ablation)	41	26	17	8	32	17	11

Table 3: **3D scene inference under count and occlusion shifts.** Per-object hybrid accuracy (%): correct support parent and pose within 2 cm/10°. Models trained on $n \leq 4$ and evaluated zero-shot; 100 scenes per setting.

5.2 CSEM INFERS PHYSICAL MODES FROM SPARSE OBSERVATIONS

Given sparse, noisy observations of a ball's 3D position, the task is to jointly recover its continuous trajectory and sequence of physical modes: `flight`, `impact`, `roll`, and `rest`. At each step,

the discrete mode selects a learned dynamics factor, while the continuous variable specifies the ball's position. We train these factors on 3,000 complete trajectories of length $T{=}20$ without sparse observation patterns. At test time, we compose them over $T \in \{20, 50, 100\}$, conditioning on noisy positions every fifth step, or every tenth in the sparse setting. Longer horizons and increased observation sparsity therefore constitute separate zero-shot shifts.

We compare against JointDiff (Capellera et al., 2026), its ALiBi variant (Press et al., 2022), and an autoregressive bridge between observation anchors. Table 4 shows that CSEM retains high trajectory validity under both shifts and achieves the highest OOD mode accuracy. Removing structural edits consistently degrades both metrics, demonstrating the benefit of revising the inferred mode sequence.

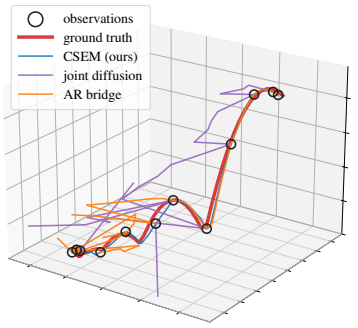


Figure 6: **Trajectory comparison.** CSEM (blue) follows the ground truth (red) through every bounce, while both baselines drift between observations.

Table 4: **Physical mode inference under horizon and observation shifts.** Accuracy (%). Methods train at $T{=}20$ (shaded) and evaluate zero-shot at longer horizons or with observations every tenth rather than fifth step. Trajectory validity measures dynamical consistency within 1.5 cm; mode accuracy measures correct mode labels.

	Trajectory validity				**Mode accuracy**			
	ID	OOD (longer, sparser) ⟶			ID	OOD (longer, sparser) ⟶		
horizon T	20	50	100	50 sparse	20	50	100	50 sparse
Joint diffusion	33	2	0	3	92	57	32	53
w/ ALiBi	39	11	1	13	**93**	62	30	54
AR bridge	40	16	8	19	91	58	32	48
CSEM (ours)	**83**	**74**	**72**	**79**	85	**67**	**36**	**57**
w/o edits (abl.)	62	51	49	54	79	62	30	53

6 Conclusion and Limitations

We introduced CSEM, a framework for hybrid reasoning that treats the composition of learned energy factors as an inference variable. Building on the line of work of compositional EBMs, CSEM jointly infers which energy factors to compose and the continuous variables that satisfy them. We learn calibrated local energies to guide structural revision, while local backtracking and continuous refinement adapt the solution to the revised composition. We applied CSEM across a wide range of real-world reasoning domains: task-and-motion planning, 3D scene inference, and physical mode inference. Our experiments demonstrate zero-shot generalization from small training instances to larger structures and longer horizons without retraining.

Two limitations motivate further work. First, in our experiments, CSEM's performance declines on the largest out-of-distribution instances. Distinguishing the effects of compounding verifier error and continuous optimization difficulty would help identify where additional training or inference-time computation is most effective. Second, applying this framework to real-world hardware setups would be an interesting future direction.

Reproducibility Statement

All learned components are small per-factor transformer encoders trained on procedurally generated datasets of 3,000 scenes per task. Architecture, optimizer, noise-schedule and budget hyperparameters are listed in Appendix A.7; the per-domain factor libraries, structure spaces and edit neighborhoods in Appendix A.6; the training procedure in Appendix A.3; and the full inference loop in Algorithm 2. Dataset generation, evaluation protocols and baseline adaptations are documented in Appendices B, C and D. Source code, dataset generators and evaluation scripts will be released upon publication.

Statement on AI Usage

Large language models were used as a writing and editing aid for this manuscript and as a coding assistant during implementation. All technical ideas, experiments and analyses are the authors' own;

all LLM-assisted text and code were reviewed and verified by the authors, who take full responsibility for the content.

REFERENCES

Christopher Agia, Toki Migimatsu, Jiajun Wu, and Jeannette Bohg. STAP: Sequencing Task-Agnostic Policies. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7951–7958, London, United Kingdom, May 2023. IEEE. ISBN 9798350323658. doi: 10.1109/ICRA48891.2023.10160220. URL https://ieeexplore.ieee.org/document/10160220/. 2

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? In *International Conference on Learning Representations (ICLR)*, 2023. 1

Guillem Capellera, Luis Ferraz, Antonio Romano, Alexandre Alahi, and Antonio Agudo. JointDiff: Bridging Continuous and Discrete in Multi-Agent Trajectory Generation. In *International Conference on Learning Representations*, volume 2026, pp. 75912–75942, April 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/7ad65363050e550c5234c4365a93442c-Abstract-Conference.html. 8, 9, 23, 24

Quentin Cappart, Didier Chételat, Elias B. Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial Optimization and Reasoning with Graph Neural Networks. *Journal of Machine Learning Research*, 24(130):1–61, 2023. ISSN 1533-7928. URL http://jmlr.org/papers/v24/21-0449.html. 3

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion Forcing: Next-token Prediction Meets Full-Sequence Diffusion. November 2024a. URL https://openreview.net/forum?id=yDo1ynArjj. 3, 4

Chang Chen, Fei Deng, Kenji Kawaguchi, Caglar Gulcehre, and Sungjin Ahn. Simple hierarchical planning with diffusion. In *International Conference on Learning Representations (ICLR)*, 2024b. 7, 19

Yilun Du, Shuang Li, and Igor Mordatch. Compositional visual generation with energy based models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 1, 3

Yilun Du, Shuang Li, Joshua Tenenbaum, and Igor Mordatch. Learning Iterative Reasoning through Energy Minimization. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 5570–5582. PMLR, June 2022. URL https://proceedings.mlr.press/v162/du22d.html. 3

Yilun Du, Conor Durkan, Robin Strudel, Joshua B. Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and MCMC. In *International Conference on Machine Learning (ICML)*, 2023. 4

Yilun Du, Jiayuan Mao, and Joshua B. Tenenbaum. Learning Iterative Reasoning through Energy Diffusion. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 11764–11776. PMLR, July 2024. URL https://proceedings.mlr.press/v235/du24f.html. 3, 4

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang (Lorraine) Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and Fate: Limits of Transformers on Compositionality. In *Advances in Neural Information Processing Systems*, volume 36, pp. 70293–70332. Curran Associates, Inc., 2023. doi: 10.52202/075280-3081. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/deb3c28192f979302c157cb653c15e90-Abstract-Conference.html. 1

Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. PDDLStream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2020. 2

Nishad Gothoskar, Marco Cusumano-Towner, Ben Zinberg, Matin Ghavamizadeh, Falk Pollok, Austin Garrett, Josh Tenenbaum, Dan Gutfreund, and Vikash Mansinghka. 3DP3: 3D Scene Perception via Probabilistic Programming. In *Advances in Neural Information Processing Systems*, volume 34, pp. 9600–9612. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/4fc66104f8ada6257fa55f29a2a567c7-Abstract.html. 3

Nishad Gothoskar, Matin Ghavami, Eric Li, Aidan Curtis, Michael Noseworthy, Karen Chung, Brian Patton, William T. Freeman, Joshua B. Tenenbaum, Mirko Klukas, and Vikash K. Mansinghka. Bayes3D: fast learning and inference in structured generative models of 3D objects and scenes, December 2023. URL http://arxiv.org/abs/2312.08715. arXiv:2312.08715 [cs.RO]. 3

Ce Hao, Anxing Xiao, Zhiwei Xue, and Harold Soh. CHD: Coupled hierarchical diffusion for long-horizon tasks. In *Conference on Robot Learning (CoRL)*, 2025. arXiv:2505.07261. 7, 19

Sigmund H. Høeg, Aksel Vaaler, Chaoqi Liu, Olav Egeland, and Yilun Du. Hybrid Diffusion for Simultaneous Symbolic and Continuous Planning. *IEEE Robotics and Automation Letters*, 11 (4):4489–4496, April 2026. ISSN 2377-3766. doi: 10.1109/LRA.2026.3664616. URL https://ieeexplore.ieee.org/document/11395608. 1, 7, 19

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning (ICML)*, 2022. 1, 7, 18

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving Quantitative Reasoning Problems with Language Models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 3843–3857. Curran Associates, Inc., 2022. doi: 10.52202/068431-0278. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/18abbeef8cfe9203fdf9053c9c4fe191-Abstract-Conference.html. 1

Xiner Li, Yulai Zhao, Chenyu Wang, Gabriele Scalia, Gokcen Eraslan, Surag Nair, Tommaso Biancalani, Shuiwang Ji, Aviv Regev, Sergey Levine, and Masatoshi Uehara. Derivative-Free Guidance in Continuous and Discrete Diffusion Models with Soft Value-based Decoding. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 95507–95545. Curran Associates, Inc., 2025. doi: 10.52202/085713-3194. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/899af0d66d8850318a20781484416152-Abstract-Conference.html. 3

Yunhao Luo, Utkarsh A. Mishra, Yilun Du, and Danfei Xu. Generative trajectory stitching through diffusion composition. *arXiv preprint arXiv:2503.05153*, 2025. 3

Luke Melas-Kyriazi, Iro Laina, Christian Rupprecht, Natalia Neverova, Andrea Vedaldi, Oran Gafni, and Filippos Kokkinos. IM-3D: Iterative Multiview Diffusion and Reconstruction for High-Quality 3D Generation. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 35310–35323. PMLR, July 2024. URL https://proceedings.mlr.press/v235/melas-kyriazi24a.html. 8

Utkarsh A. Mishra, Yongxin He, Hongyi Chen, and Danfei Xu. CDGS: Compositional diffusion with guided search. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2601.00126. 3, 7, 19

Alexandru Oarga and Yilun Du. Generalizable reasoning through compositional energy minimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2510.20607. 1, 3, 4, 5

Ofir Press, Noah A. Smith, and Mike Lewis. Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation. International Conference on Learning Representations, April 2022. doi: 10.48550/arXiv.2108.12409. URL http://arxiv.org/abs/2108.12409. arXiv:2108.12409 [cs.CL]. 9, 24

Martin J. A. Schuetz, J. Kyle Brubaker, and Helmut G. Katzgraber. Combinatorial optimization with physics-inspired graph neural networks. *Nature Machine Intelligence*, 4(4):367–377, April 2022. ISSN 2522-5839. doi: 10.1038/s42256-022-00468-6. URL https://www.nature.com/articles/s42256-022-00468-6. 3

Tao Sun, Utkarsh Aashu Mishra, Jiaxin Lu, Danfei Xu, and Iro Armeni. Energy-based Compositional Diffusion Planning, June 2026. URL http://arxiv.org/abs/2606.21646. arXiv:2606.21646 [cs.RO]. 3

Marc Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015. 2

Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, January 2024. ISSN 1476-4687. doi: 10.1038/s41586-023-06747-5. URL https://www.nature.com/articles/s41586-023-06747-5. 1

Johanna Wald, Helisa Dhamo, Nassir Navab, and Federico Tombari. Learning 3D Semantic Scene Graphs From 3D Indoor Reconstructions. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3960–3969, June 2020. doi: 10.1109/CVPR42600.2020.00402. URL https://ieeexplore.ieee.org/abstract/document/9156565. 8

Luhuan Wu, Brian Trippe, Christian Naesseth, David Blei, and John Cunningham. Practical and Asymptotically Exact Conditional Sampling in Diffusion Models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 31372–31403. Curran Associates, Inc., 2023. doi: 10.52202/075280-1363. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/63e8bc7bbf1cfea36d1d1b6538aecce5-Abstract-Conference.html. 3

Yiqing Xu, Jiayuan Mao, Linfeng Li, Yilun Du, Tomas Lozáno-Pérez, Leslie Pack Kaelbling, and David Hsu. "Set it up": Functional object arrangement with compositional generative models. *The International Journal of Robotics Research*, pp. 02783649251378198, November 2025. ISSN 0278-3649, 1741-3176. doi: 10.1177/02783649251378198. URL https://journals.sagepub.com/doi/10.1177/02783649251378198. 1

Zhutian Yang, Jiayuan Mao, Yilun Du, Jiajun Wu, Joshua B. Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Compositional diffusion-based continuous constraint solvers. In *Conference on Robot Learning (CoRL)*, 2023. 1, 3, 7, 19

Haotian Ye, Haowei Lin, Jiaqi Han, Minkai Xu, Sheng Liu, Yitao Liang, Jianzhu Ma, James Zou, and Stefano Ermon. TFG: Unified training-free guidance for diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 3

Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang, and Guy Van Den Broeck. On the Paradox of Learning to Reason from Data. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pp. 3365–3373, Macau, SAR China, August 2023. International Joint Conferences on Artificial Intelligence Organization. ISBN 9781956792034. doi: 10.24963/ijcai.2023/375. URL https://www.ijcai.org/proceedings/2023/375. 1

XiangCheng Zhang, Haowei Lin, Haotian Ye, James Y. Zou, Jianzhu Ma, Yitao Liang, and Yilun Du. Inference-time scaling of diffusion models through classical search. *International Conference on Learning Representations*, 2026:39133–39166, April 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/41c9aa6c04924c29c6469572c5f6d01b-Abstract-Conference.html. 3, 5

Guangyao Zhou, Nishad Gothoskar, Lirui Wang, Joshua B. Tenenbaum, Dan Gutfreund, Miguel Lázaro-Gredilla, Dileep George, and Vikash K. Mansinghka. 3D Neural Embedding Likelihood: Probabilistic Inverse Graphics for Robust 6D Pose Estimation. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 21568–21579, Paris, France, October 2023. IEEE. ISBN 9798350307184. doi: 10.1109/ICCV51070.2023.01977. URL https://ieeexplore.ieee.org/document/10377270/. 3

A Additional Method Details

This appendix provides a concrete decomposition of one domain, implementation details for factor training and inference, and the hyperparameters used in the main experiments.

A.1 Worked example: 3D multi-bin packing

A packing instance contains K open-top bins and n blocks with known dimensions. The discrete structure $G(i) \in \{1, \dots, K\}$ assigns each block i to a bin, while its continuous pose is $\mathbf{y}_i = (x_i, y_i, z_i, \theta_i)$. Poses are expressed in normalized bin-local coordinates.

We learn three reusable factors from small subproblems: `contain` scores whether one block lies inside its assigned bin; `cfree` scores whether two co-assigned blocks intersect; and `bin` scores the joint packability of a set of two to four blocks. The set factor is permutation-equivariant. No factor is trained on a complete packing scene.

Let $B_k(G) = \{i : G(i) = k\}$. A proposed assignment activates one containment factor per block, one collision factor per co-assigned pair, and a set factor for supported groups of two to four co-assigned blocks:

$$E_\theta(\mathbf{x}, \mathbf{y}; G) = \beta_{\mathrm{c}} \sum_{i=1}^{n} E_\theta^{\mathrm{contain}}(\mathbf{x}_i, \mathbf{y}_i) + \beta_{\mathrm{f}} \sum_{k=1}^{K} \sum_{\substack{i<j \\ i,j \in B_k(G)}} E_\theta^{\mathrm{cfree}}(\mathbf{x}_{ij}, \mathbf{y}_i, \mathbf{y}_j)$$
$$+ \beta_{\mathrm{b}} \sum_{k:\, 2 \le |B_k(G)| \le 4} E_\theta^{\mathrm{bin}}(\mathbf{x}_{B_k}, \mathbf{y}_{B_k}). \tag{14}$$

We use $\beta = (1, 2, 1)$. Changing an assignment changes both the active factors and the coordinate chart of the reassigned block, thereby recomposing the energy landscape.

A graph is *structurally plausible* when every block is assigned to exactly one bin. There are K^n such assignments, but many admit no valid continuous packing. Determining whether an assignment is feasible therefore requires minimizing its corresponding energy. Other domains follow the same pattern: graph syntax is inexpensive to enforce, whereas compatibility with a continuous solution is not known in advance.

A.2 Factor parameterization

Each constraint type is represented by a small pre-LayerNorm transformer ($d_{\mathrm{model}} = 256$, 8 heads, 3 layers). Each token contains projections of its continuous variable and problem context, together with a learned role embedding and a sinusoidal embedding of its diffusion timestep. The token representations are mean-pooled and mapped to a scalar energy by a two-layer MLP. Set factors omit positional encodings and use masked pooling to preserve permutation equivariance.

Unless stated otherwise, we use a cosine diffusion schedule with 100 levels. The temporal domain instead uses the geometric noise schedule described in Appendix D.

A.3 Factor training

Every factor is trained independently on examples of its local constraint. When a variable has $t_k = 0$, it remains an input to the factor but contributes no term to the denoising loss. The all-clean state is supervised separately by the validity-calibration objective. A training generator produces a valid configuration $\mathbf{y}^+$ and a matched negative $\mathbf{y}^-$ by perturbing one or more variables until the constraint is violated. We sample violations at different severities to provide supervision near the validity boundary.

We additionally collect hard negatives by running the solver on training-distribution scenes and labeling its terminal factor instances with the domain checker. These examples are mixed with synthetic pairs in the calibration objective. Timesteps are sampled independently for each token, exposing the factors to the mixed-noise configurations encountered during local backtracking.

All factors use Adam, gradient clipping at 1.0, batch size 256, and $\lambda_{\text{cal}} = 1$. We select checkpoints using held-out calibration performance while requiring the denoising error to remain below a validation threshold. Domain-specific settings are reported in Table 6.

A.4 Continuous minimization under a fixed graph

For a fixed graph G, MINIMIZE refines the continuous variables through an annealed denoise–refine–re-noise procedure. We use the weighted composition

$$E_\theta(\mathbf{x}, \mathbf{y}, \mathbf{t}; G) = \sum_{i \in \mathcal{I}(G)} \beta_i E_\theta^i(\mathbf{x}_i, \mathbf{y}_{S_i}, \mathbf{t}_{S_i}), \qquad \beta_i > 0, \tag{15}$$

where $\mathcal{I}(G)$ contains the active factors and β_i are fixed factor weights ($\beta_i = 1$ by default).

The solver receives a continuous state already initialized at the prescribed token-wise noise levels $\mathbf{b}$. Fresh samples start from Gaussian noise with $b_k = t_{\max}$. After a structural edit, the previous clean solution is re-noised using the forward corruption process: affected tokens use $b_k = t_{\max}$, while context tokens use $b_k = t_{\text{part}}$.

Denoising with noise-scaled energy gradients. Let $a_k = \sqrt{\bar{\alpha}_{t_k}}$ and $s_k = \sqrt{1 - \bar{\alpha}_{t_k}}$. Under the objective in Eq. equation 6, each factor's gradient with respect to variable k, multiplied by s_k, is trained to predict the forward noise on that variable. For a variable shared by several factors, we combine these noise predictions using a weighted average:

$$\hat{\boldsymbol{\epsilon}}_k = \frac{s_k}{d_k} \sum_{\substack{i \in \mathcal{I}(G) \\ k \in S_i}} \beta_i \nabla_{\mathbf{y}_k} E_\theta^i(\mathbf{x}_i, \mathbf{y}_{S_i}, \mathbf{t}_{S_i}), \qquad d_k = \sum_{\substack{i \in \mathcal{I}(G) \\ k \in S_i}} \beta_i. \tag{16}$$

We assume that every updated token belongs to at least one active factor, so $d_k > 0$. This normalization controls the aggregate prediction scale as the number of incident factors changes. The resulting clean estimate is

$$\hat{\mathbf{y}}_{0,k} = \frac{\mathbf{y}_k - s_k \hat{\boldsymbol{\epsilon}}_k}{a_k}. \tag{17}$$

This uses the standard noise-to-clean reconstruction formula with an aggregate of local predictions; we do not assume that the aggregate is the exact noise predictor of the joint distribution. At $t_k = 0$, where $a_k = 1$ and $s_k = 0$, the clean estimate is simply $\hat{\mathbf{y}}_{0,k} = \mathbf{y}_k$.

Annealing and clean refinement. We use a decreasing ladder $t_{\max} = \ell_1 > \cdots > \ell_R = 0$. At ladder level r, token k is evaluated at

$$t_k = \min(b_k, \ell_r). \tag{18}$$

Thus, its noise level never exceeds b_k: it remains at b_k while $\ell_r > b_k$ and follows the decreasing ladder thereafter. All ladder iterations are executed. Context tokens remain available for refinement even when their noise levels are lower than those of the affected tokens.

At each level, we first compute the clean estimate in Eq. equation 17. We then apply S_r normalized gradient steps to the clean composed energy, with $S_r = S$ except at the final level, where $S_R = 3S$. Unless this is the final level, we re-noise the refined estimate to the next scheduled token-wise noise levels using fresh Gaussian noise. Algorithm 1 gives the shared update procedure; the domain-specific modifications are described below. This procedure implements the continuous refinement step of Sec. 3.3, rather than an exact reverse-diffusion transition.

Algorithm 1 MINIMIZE$(\mathbf{y}, G, \mathbf{b})$: shared denoise–refine–re-noise procedure

Require: State $\mathbf{y}$ initialized at noise levels $\mathbf{b}$; active factors $\mathcal{I}(G)$; weights $\beta_i > 0$; noise schedule $\bar{\alpha}_t$; ladder $\ell_1 = t_{\max} > \cdots > \ell_R = 0$; clean steps S; step sizes η_r

$d_k \leftarrow \sum_{i \in \mathcal{I}(G):\, k \in S_i} \beta_i$ for every updated token k
for $r = 1, \ldots, R$ **do**
  $t_k \leftarrow \min(b_k, \ell_r)$ for every token k
  $a_k \leftarrow \sqrt{\bar{\alpha}_{t_k}}, \quad s_k \leftarrow \sqrt{1 - \bar{\alpha}_{t_k}}$
  For each token with $t_k > 0$, compute $\hat{\boldsymbol{\epsilon}}_k$ using Eq. equation 16
  $\hat{\mathbf{y}}_{0,k} \leftarrow (\mathbf{y}_k - s_k \hat{\boldsymbol{\epsilon}}_k)/a_k$ if $t_k > 0$; otherwise $\hat{\mathbf{y}}_{0,k} \leftarrow \mathbf{y}_k$
  $S_r \leftarrow 3S$ if $r = R$, and $S_r \leftarrow S$ otherwise
  for $j = 1, \ldots, S_r$ **do**
    $\mathbf{g}_0 \leftarrow \nabla_{\mathbf{y}} E_\theta(\mathbf{x}, \hat{\mathbf{y}}_0, \mathbf{0}; G)$
    $\hat{\mathbf{y}}_0 \leftarrow \hat{\mathbf{y}}_0 - \eta_r \mathbf{g}_0 / \mathrm{RMS}(\mathbf{g}_0)$
  end for
  if $r < R$ **then**
    $t'_k \leftarrow \min(b_k, \ell_{r+1})$
    $a'_k \leftarrow \sqrt{\bar{\alpha}_{t'_k}}, \quad s'_k \leftarrow \sqrt{1 - \bar{\alpha}_{t'_k}}$
    $\boldsymbol{\xi}_k \sim \mathcal{N}(\mathbf{0}, I), \quad \mathbf{y}_k \leftarrow a'_k \hat{\mathbf{y}}_{0,k} + s'_k \boldsymbol{\xi}_k$
  end if
end for
return $\hat{\mathbf{y}}_0$

We define $\mathrm{RMS}(\mathbf{g}) = \sqrt{\|\mathbf{g}\|_2^2/d + \epsilon_{\mathrm{num}}}$, where d is the dimensionality of $\mathbf{y}$ and $\epsilon_{\mathrm{num}} > 0$ prevents division by zero. The degree normalization in Eq. equation 16 is used only to combine local noise predictions; clean refinement uses the gradient of the composed energy in Eq. equation 15.

Population inference. With $P > 1$ particles, MINIMIZE is applied independently to each particle under its current graph. In the reported population runs, resampling occurs between structural revision rounds, as described in Sec. A.5; no resampling is performed within the denoising ladder. Population scores are evaluated on the resulting clean states, where the feasibility calibration is applied.

A.5 STRUCTURAL REVISION

Algorithm 2 summarizes the complete inference loop. CSEM first minimizes the energy induced by a structurally plausible graph. It then evaluates every active factor at the clean timestep. High factor energy indicates a likely local violation and increases the probability that the corresponding structural decision is edited.

For each proposed edit, CSEM fully re-noises variables incident to the changed factors and partially re-noises the remaining context. It then minimizes the recomposed energy and retains the best candidate. We define factor risk as $\rho_i = \sigma(e_i)$ and stop when $\max_i \rho_i < \delta$.

Algorithm 2 CSEM inference for one particle

Require: Factors $\{E_\theta^i\}$; graph syntax $\mathcal{G}(\mathbf{x})$; round cap $R_{\max}$; proposals H; levels $t_{\max}, t_{\text{part}}$; temperature τ; threshold δ

Sample $G \sim q_0(\mathcal{G}(\mathbf{x}))$ and $\mathbf{y} \sim \mathcal{N}(0, I)$
$\mathbf{y} \leftarrow \textsc{Minimize}(\mathbf{y}, G, t_{\max}\mathbf{1})$
for $r = 1, \ldots, R_{\max}$ **do**
  $e_i \leftarrow E_\theta^i(\mathbf{x}_i, \mathbf{y}_{S_i}, \mathbf{0})$ and $\rho_i \leftarrow \sigma(e_i)$ for all $i \in \mathcal{I}(G)$
  if $\max_i \rho_i < \delta$ **then**
    break
  end if
  Sample H edit targets with probability $p(i \mid G, \mathbf{y}) \propto \exp(e_i/\tau)$
  for each target i_h **do**
    Construct a syntax-valid local edit G'_h
    Set $b_k = t_{\max}$ for affected variables and $b_k = t_{\text{part}}$ otherwise
    Re-noise $\mathbf{y}$ to $\mathbf{b}$ and compute $\mathbf{y}'_h \leftarrow \textsc{Minimize}(\mathbf{y}, G'_h, \mathbf{b})$
  end for
  Retain the best of the parent and H edited candidates
end for
return $(G, \mathbf{y})$

For multiple particles, all candidates are processed in one batch and resampled when the effective sample size falls below $P/2$. Candidate scores are normalized by the number of active factors; packing instead uses the worst factor risk because a single violated constraint invalidates the solution. We report inference cost in factor-network evaluations (NFE).

A.6 Domain instantiations

Table 5 summarizes the representation and local edit used in each domain.

Table 5: **Instantiation of CSEM in each domain.** Arity is the number of continuous variables a factor reads; "always" means the factor is active under every G; the incident set is what an edit re-noises to $t_{\max}$ (everything else re-noises to t_{part}).

Domain	Discrete structure G	Continuous $\mathbf{y}$	Factors (arity; activation)	Edit / incident set
3D stacking	split of the n blocks into sub-stacks of ≤ 4; per block two parent slots (floor, one supporter, or a two-block bridge)	planar pose (u_i, v_i, θ_i) per block	within-frame (1; always), cfree (2; always), support (2; per parent slot), sub-stack value (set of ≤ 4; per leaf)	re-mask the blamed block's parent slots; incident: the blamed blocks
Rearrangement	per mover a routing bit (direct / via buffer) and a priority that orders all moves	home pose and, if staged, buffer pose per mover	contain (1; home always, buffer iff staged), cfree (2; gated by which objects coexist at that time)	flip a routing bit or resample a mover's order; incident: that mover's two pose slots
Bin packing	bin assignment $G(i) \in \{1..K\}$ per block	pose $(x_i, y_i, z_i, \theta_i)$ per block	contain (1; always), cfree (2; iff co-assigned), bin (set of 2–4 co-residents; per occupied bin)	reassign the blamed block to another bin; incident: the block and its old and new bin-mates
Scene inference	support parent per object (table / on j / in c) and three relation bits per ordered pair	pose $(x_i, y_i, z_i, \theta_i)$ per object	obs (1; always), cfree (2; always), on-table / supported-by / inside (1–2; per parent), left-of / in-front-of / above (2; every pair, gradient-active iff the bit is on)	change a parent or flip a relation bit; incident: the object, its old and new parent
Temporal modes	mode per step: flight, impact(s), roll(s), rest(s) over the scene's surfaces s	ball position per step	flight / roll / rest (3 consecutive positions), impact (4), within (1), cfree (1)	relabel one inter-observation span; incident: that span's positions

A.7 Hyperparameters

Tables 6 report the settings used for the headline results. Shared architectural and optimization settings are given above; the tables contain only values that vary by domain. All settings are selected on in-distribution validation instances and remain fixed for OOD evaluation.

Table 6: **Training settings used in the main experiments.**

	Stacking	Rearrangement	Packing	Scene	Temporal
Learning rate	10^{-4}	5×10^{-4}	5×10^{-4}	3×10^{-4}, cosine	3×10^{-4}
Training budget	80k updates	30×50k examples	30×50k examples	30×20k examples	300×50k examples
Diffusion levels	1000	100	100	100	100, geometric σ
Training distribution	3,000 scenes, $n \leq 4$	3,000 scenes, $(n, k) = (4, 2)$	3,000 scenes, $K = 2$	3,000 scenes, $n \leq 4$	3,000 trajectories, $L = 20$

Table 7: **Inference settings used for the headline results.**

	Stacking	Rearrangement	Packing	Scene	Temporal
Particles P	1 4 seeds/edit	1	16	16	16
Round cap	8 min. 4 before stopping $t_{\max} = 300$	$12n$	8	32	L
Backtracking	context clamped	99/30	99/30	99/30	span re-noising $\sigma = 0.475$
Annealing / guide steps	1000/—	12/4	12/4	16/8	$\sigma : 0.1 \to 0.002$ 80 steps/level

B Task-and-Motion Planning: Experimental Details

B.1 Datasets and evaluation

Table 8 summarizes the three planning domains. Every method returns one plan per scene, which is evaluated once without best-of-N selection or post-hoc filtering. All learned components are trained only on the in-distribution training sets.

Table 8: **Task-and-motion planning datasets and evaluation.** Every test set contains 100 scenes unless otherwise stated.

Task	Training distribution	OOD evaluation	Success criterion
3D stacking	3,000 scenes with $n \leq 4$; 45,400 local sub-stacks	$n \in \{5, 6, 7, 8\}$	Bottom-up PyBullet assembly survives every placement; pose error remains within 2 cm/10° and all blocks remain below the ceiling.
Labeled-box rearrangement	3,000 scenes at $(n, k) = (4, 2)$	$(6, 2), (8, 2), (10, 2), (6, 3), (12, 3)$	Exact geometric replay is collision-free after every move and all objects finish in their labeled boxes.
3D multi-bin packing	3,000 scenes with $K = 2$ and $n \in [4, 8]$	$K \in \{3, 4, 5\}$, with up to 19 blocks	Every intermediate placement is contained and collision-free under strict geometric replay.

Scenes are generated to be feasible. Rearrangement varies object count, box count, and packing tightness; packing holds the per-bin fill distribution approximately fixed so that increasing K primarily enlarges the assignment space.

B.2 Stacking-specific instantiation

Stacking uses divide-and-conquer because stability must be evaluated over an assembled sub-stack rather than an isolated pair of blocks. CSEM partitions the scene into sub-stacks of at most four blocks and infers an acyclic support structure within each leaf. A shared transformer predicts block poses and support parents together with a scene-level stability score trained from PyBullet labels (held-out AUC 0.936).

At inference, CSEM samples one leaf solution and accepts it when the stability score exceeds 0.5. Otherwise, it revises the blamed support parents, re-noises the affected poses to $t = 300$, and evaluates four backtracking samples. Each leaf permits at most eight revision rounds; persistent failure triggers a new scene partition, with at most 24 leaf solves per scene. The no-edit ablation uses the same partition search and leaf sampler but disables support-graph revision.

B.3 Baselines

All baselines use the same object features, pose parameterization, training data, and validation protocol as CSEM. Diffuser (Janner et al., 2022) generates continuous plans with trajectory diffusion.

HD (Chen et al., 2024b) and CHD (Hao et al., 2025) add hierarchical trajectory representations. HDP (Høeg et al., 2026) jointly diffuses symbolic and continuous plans and is adapted to variable object counts using relative parent tokens and sinusoidal slot embeddings.

CCSP (Yang et al., 2023) normally assumes a supplied constraint graph. For a non-oracle comparison, *ccsp-rand* uses a random valid support structure for stacking and first-fit-decreasing assignments for packing. We omit CCSP from rearrangement because constructing its constraint graph would reveal the move plan. CDGS (Mishra et al., 2026) uses its original chain composition, iterative resampling, and likelihood pruning.

B.4 Compute and matched-budget search

Table 9 reports CSEM and the baselines' mean numbers of factor-network evaluations (NFEs) per scene. For reference, Diffuser, HDP, and CCSP use approximately 1k network evaluations; HD and CHD use 4–5k; and CDGS uses approximately 50–220k, depending on the task. CSEM therefore spends more inference computation than most baselines, but we do not equalize budgets by drawing best-of-N samples from them, because such a comparison is not well defined. The baselines are single-shot generators with no learned notion of solution validity: without a verifier, selecting among N samples reduces to picking one at random, so the expected solve rate of best-of-N equals that of a single sample, and additional NFEs cannot improve it. Selecting instead with the physics simulator or the geometric replay checker would place the evaluation oracle inside the planner; the resulting method is no longer zero-shot, and the comparison is no longer fair. Selecting with the energy verifier in our method would then make it an ablation rather than a baseline comparison.

Packing configurations. All packing numbers in this paper come from the same solver (calibrated checkpoints, P=16, H=2 clones per round, energy guidance μ=0.01) in three configurations that use the settings described below. **(A) Headline** (Table 1, Table 2, Figure 4): bin-level set factor on, risk bar δ=0.55, round cap 8; mean edits spent 6/26/83/158 per scene at K=2–5 (NFE 2.7k/4.9k/11.1k/19.2k). **(B) Fixed budget**: no risk bar, 3 rounds, exactly 96 edits per scene (12.4k NFE); at K=2 this arm was run before the set factor existed and with summed-energy selection (99%), at K=3 with the set factor and max-risk selection (83%). **(C) Fixed rounds**: the full stack of (A) with the risk bar removed, 3 rounds at $K\leq 3$ and 8 rounds at $K\geq 4$ (Table 13). Where a table mixes configurations, the sentence before it says which cells are which. The CSEM packing row of Table 9 reports configuration (B) at K=2, 3 and (A) at K=4, 5.

Table 9: Mean network evaluations (NFE, $\times 10^3$) per scene for the methods evaluated in Table 1. Single-chain baselines return one plan per scene; CDGS uses $B = 16$. CSEM uses the headline configurations for stacking and rearrangement. For packing, CSEM uses configuration (B) at $K = 2, 3$ and configuration (A) at $K = 4, 5$, as defined above. One NFE is one forward pass of a denoiser (baselines) or one factor-instance evaluation (CSEM).

	stacking n					rearrangement (n, k)						packing K			
	4	5	6	7	8	4,2	6,2	8,2	10,2	6,3	12,3	2	3	4	5
Diffuser	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
HD	3	3	4	4	5	1.9	2.5	3.1	3.7	2.5	4.4	3.9	5.2	6.7	8.3
CHD	3	3	4	4	5	1.9	2.5	3.1	3.7	2.5	4.4	3.9	5.2	6.7	8.3
HDP	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ccsp-rand	1	1	1	1	1	–	–	–	–	–	–	1	1	1	1
CDGS	54	54	54	54	54	72	126	144	199	126	217	50	66	78	97
CSEM	6	108	100	95	90	0.5	1.9	3.3	4.9	1.4	7.8	12.4	12.4	11.1	19.2

To isolate the search procedure, we replace CSEM's verifier-guided edits and local backtracking with annealed Metropolis–Hastings over the same energies and matched NFE. Each proposal changes one random discrete decision with probability 0.2 or perturbs one object's pose otherwise. The temperature decreases from 5 to 1 and the proposal scale from 0.3 to 0.01. Packing uses 16 chains and rearrangement uses one, matching CSEM's particle counts. The CSEM reference row for packing is configuration (B) at K=2, 3 and (A) at K=4, 5; the MH iteration counts match those cells' NFE.

Table 10: **Metropolis–Hastings over CSEM's energies at matched NFE** (solve %, 100 scenes per cell). Same energies, same budget, random proposals instead of learned transport and blame. Best per column in bold.

	rearrangement (n, k)						packing K			
	4,2	6,2	8,2	10,2	6,3	12,3	2	3	4	5
CSEM (ours)	**65**	**66**	**42**	**39**	**58**	**25**	**99**	**83**	**63**	**52**
w/o edits	61	48	29	25	47	14	80	43	17	8
MH over the same energies	58	53	37	32	52	**25**	44	11	0	0

MH is competitive on rearrangement, whose structural space contains one routing bit and one priority per object, but degrades rapidly on packing as the assignment space grows. This comparison indicates that verifier-guided revision is most beneficial when discrete proposals are strongly coupled to continuous feasibility.

B.5 Additional packing analyses

Table 11 uses configuration (A) throughout: the t=0 row is (A) itself, and the t_e rows move verification and editing to noise level t_e with the same particles, risk bar and round cap.

Table 11: **When to verify: editing the structure before denoising finishes** (packing; solve % with mean NFE per scene in parentheses; 100 scenes per cell; 16 particles; same energies and stopping rule).

		bins K			
Verify and edit at	round cap	2	3	4	5
t=0 (CSEM)	8	**92** (2.7k)	**77** (4.9k)	**63** (11.1k)	**52** (19.2k)
t_e=27	8	83 (6.0k)	58 (11.5k)	36 (16.4k)	20 (18.5k)
t_e=27	16	88 (7.4k)	63 (18.2k)	47 (28.3k)	22 (32.8k)
t_e=54	8	83 (12.1k)	57 (16.3k)	33 (17.9k)	13 (18.3k)

Editing time. Table 11 compares editing after full denoising with editing at intermediate diffusion levels. Early verification reduces the cost of an individual proposal but produces less reliable edits, lowering solve rate without reducing total NFE. Increasing its round cap recovers some performance but remains worse than editing at the clean timestep.

Inference-time scaling. Figure 4 varies the population $P \in \{4, 16, 64\}$ and the round cap $\in \{2, 4, 8, 16\}$ of configuration (A) at K=5; the P=16, cap-8 point is the Table 1 cell. It shows that packing performance improves with both particle count and edit budget. Larger populations provide the stronger gain at fixed per-particle budget because they explore and resample across more candidate structures.

More ablations. Table 12 separates the effects of energy calibration, continuous guidance, adaptive revision, and the bin-level set factor. Calibration produces the largest in-distribution gain, while the set factor becomes increasingly important as the number of bins grows. The population oracle indicates that the remaining large-scale errors are primarily failures to reach a valid structure rather than failures to select one already present in the population.

Rows of Table 12 are configuration (B) unless a row adds a component: the InfoNCE and no-guidance rows use summed-energy selection and no guidance (6.9k NFE); the risk-bar row is cap 8 (256 edits) without the set factor; the full row is (B) at K=3 and the headline configuration (A) at K=4, 5, which is why it differs from Table 1 at K=3.

Table 12: **Bin packing, cumulative ablation ladder** (strict-replay solve %, 100 scenes per cell; best of the terminal population in parentheses). Arms stack cumulatively; edit budget 96 unless noted.

	K=2 (ID)	K=3	K=4	K=5
contrastive (InfoNCE) energies	8	9	—	—
calibrated energies, no guidance	90 (99)	68 (75)	36 (38)	6 (6)
+ energy guidance	**99** (99)	80 (89)	44 (48)	28 (28)
+ worst-instance keep-best	—	82 (89)	—	—
+ risk-bar loop, 256-edit cap	—	—	54 (56)	45 (46)
+ bin-level set factor (full)	—	**83** (94)	**63** (74)	**52** (56)

Table 13 uses configuration (C) for both rows and runs the ablation where the local structural edits are completely randomly selected, instead of guided by energy values. From the results below, we can see that for OOD settings, editting with energy guidance is better than randomly editting.

Table 13: **Editing target ablation at matched edit counts** (packing; solve %, 100 scenes per cell). Both arms run a fixed number of rounds without the risk-bar stop, so every scene spends exactly 96 ($K \leq 3$) or 256 ($K \geq 4$) edits; only the edit-target distribution differs. We found that for OOD settings ($K = 3 - 5$), targeted editing with energy is indeed more effective than random editing, which means that using energy values to guide graph editing is important.

	K=2	K=3	K=4	K=5
editing with energy (CSEM)	95	**83**	**87**	**59**
random editing	**99**	72	68	46

B.6 Additional rearrangement analyses

Rearrangement factors are evaluated at every intermediate state. A factor is active only when its objects coexist under the hypothesized move order. Removing these temporal gates reduces solve rate to 30% at both $(n, k) = (6, 2)$ and $(6, 3)$; restoring them yields 57% and 61%, while the full adaptive method reaches 66% and 58%.

We also evaluate 50 harder scenes at $(n, k) = (20, 5)$. CSEM and all baselines solve 0 scenes. Most failures occur within the first four moves and are associated with verifier false positives and occupancies beyond those represented during factor training. We report this result as a current failure regime rather than including it in the main comparison.

B.7 Kinematic extension

Setup. We extend rearrangement and packing with an analytic 2.5D manipulation model: a fixed-base arm in front of the workspace, a reach annulus, top-down grasps, and jaw plates of thickness t_j=0.02 along a grasp axis $a \in \{x, y\}$ chosen per move. Packing additionally uses a grip depth d_g=0.05 below the block top.

Each move follows a prescribed primitive: approach, descend, grasp, lift, carry above all walls and bins, descend, release, and retract. The replay checks reachability of the pick and place poses, jaw clearance against objects present in the source and target regions, and containment of the jaw-extended footprint. For packing, it also checks the descent corridor against previously placed blocks. These checks evaluate feasibility under the prescribed manipulation model.

The analytic replay provides training labels and evaluates final plans but is never queried during inference. We regenerate the training and evaluation scenes with jaw margins, using 3,000 training scenes per task, 100 scenes per evaluation setting for CSEM, and 50 per setting for the rearrangement baselines.

Factors and solver. We add two factors trained using the procedure in Appendix A.3: *jaw containment*, a unary factor requiring the jaw-extended footprint to fit inside the target region, and *jaw clearance*, a pairwise factor requiring the plates to avoid neighbouring objects and their grip bands. Both factors condition on the grasp axis and jaw geometry.

Grasp-axis choices are handled within the jaw-energy evaluation rather than through structural edits. We use a soft minimum over the two axes for gradient computation and a hard minimum for verification, blame assignment, and stopping. The decoder emits an axis for each move, which the replay checks explicitly. The structural-revision procedure is unchanged; the jaw factors join the active factor set. We use the main-experiment inference settings: $P{=}1$ for rearrangement and configuration (A) for packing.

We compare three variants on the kinematic scenes. The *geometric reference* runs the geometric solver and evaluates its outputs using geometric checks only. The *control* runs the geometric solver and evaluates its outputs using the kinematic replay, accepting any feasible grasp axis. CSEM adds the jaw factors and is evaluated by the kinematic replay using its decoded axes. External baselines are retrained on the kinematic datasets and evaluated with any feasible grasp axis accepted.

The geometric reference uses a less restrictive evaluation criterion and is provided for context; its achieved solve rate is not an upper bound on other solvers. Reported NFE ranges are 0.6–9.8k for CSEM versus 0.3–1.9k for the control on rearrangement, and 2.3–16.2k versus 2.2–12.4k on packing.

Table 14: **Kinematic extension.** Solve rates (%) under the analytic kinematic replay, except for the italicized geometric reference, which uses geometric checks only. The control omits the jaw factors; CSEM adds them while retaining the structural-revision procedure. Packing columns "2 (val.)" and "2 (test)" are both in-distribution settings and denote the validation and test splits, respectively.

	rearrangement (n, k)						packing K				
	4,2 (ID)	6,2	6,3	8,2	10,2	12,3	2 (val.)	2 (test)	3	4	5
Diffuser	20	4	4	0	0	0	4	1	0	0	0
HD	4	0	2	0	0	0	0	0	0	0	0
CHD	12	2	4	2	0	0	1	0	0	0	0
HDP	24	0	0	0	0	0	41	44	2	0	0
ccsp-rand	–	–	–	–	–	–	15	13	1	0	0
CDGS	20	8	8	2	0	0	5	4	0	0	0
geometric reference	*74*	*66*	*64*	*65*	*39*	*47*	*94*	*91*	*80*	*73*	*64*
control (no jaw factors)	39	23	25	12	3	2	47	41	17	18	16
CSEM (+ jaw factors)	**71**	**66**	**59**	**60**	**25**	**39**	**80**	**84**	**52**	**38**	**30**

Table 14 shows that adding jaw factors improves solve rates over the control in every evaluated setting. At the largest rearrangement setting, $(n, k) = (12, 3)$, success increases from 2% to 39%; at the largest packing setting, $K = 5$, it increases from 16% to 30%. These results support incorporating additional manipulation constraints through the factor library while retaining the structural-revision procedure, with the additional inference cost reported above.

C 3D Scene Inference: Experimental Details

C.1 Data and observations

We generate tabletop scenes in PyBullet containing n boxes and up to two open-top containers. An acyclic support graph is sampled first; objects are then placed on the table, on another object, or inside a container using rejection sampling. We simulate each scene and retain it only if the objects remain stable and all sampled relations hold after settling.

We render one RGB-D image with instance segmentation from a randomized camera. Each object's visible depth pixels are unprojected into a world-frame point cloud of at most 512 points. Every method receives the same inputs: visible points, object dimensions, container geometry, and camera parameters. Object identities are randomly permuted. Objects with no visible pixels receive a learned no-observation embedding.

The count-shift sets increase the number of objects while preserving the training camera distribution. The occlusion sets place the camera closer to the table at a lower elevation; in the most difficult setting, 21% of objects have less than 30% of their surface visible.

C.2 Factor library and structure

We represent each pose as $(x, y, z, \cos 2\theta, \sin 2\theta)$ and learn eight factors: `obs`, `cfree`, `on-table`, `supported-by`, `inside`, `left-of`, `in-front-of`, and `above`. The observation factor transforms the visible points into the hypothesized object frame and encodes them with a small PointNet. Its gradients therefore move the pose toward one whose box surface agrees with the observed points.

Observation-factor negatives use translated, rotated, or mismatched object clouds. Structural-factor negatives violate support, containment, collision, or relative-position constraints at varying severities. Every factor obtains at least 0.95 held-out calibration AUC at the smallest tested violation.

The discrete structure contains one support-parent variable per object and three relation bits per ordered pair. Parent assignments must be acyclic, respect the maximum stack depth, and place containers on the table. For relation bits, an active predicate contributes $\log \sigma(-E)$ and an inactive predicate contributes $\log \sigma(E)$. We average these terms across factor instances when comparing structures with different numbers of active relations. Only active relations contribute gradients during continuous refinement.

C.3 Inference and evaluation

During inference, parent edit re-noises the edited object and its old and new parents; a relation edit re-noises the two incident objects. Initial parents are sampled from the syntactic prior, and relation bits are initialized from the factor scores at the initial poses.

Mean NFE ranges from 42k at $n = 4$ to approximately 118k at $n = 8$–10. Runtime is 4–20 minutes per scene on eight CPU cores, depending on object count and occlusion.

We report *hybrid accuracy*: the fraction of objects with the correct supporter whose translation error is below 2 cm per axis and yaw error below 10°, accounting for the 90° symmetry of square footprints.

C.4 Baselines

Feed-forward transformer. The FF baseline uses a four-layer transformer over object tokens containing the same observations provided to CSEM. Separate heads predict object poses, support parents, and pairwise relations. It is trained with pose regression, parent cross-entropy, and relation BCE. At inference, parent predictions are projected onto the same acyclic graph syntax used by CSEM.

JointDiff. JointDiff (Capellera et al., 2026) jointly denoises continuous poses and discrete structure tokens. We use an eight-layer transformer with the same object conditioning as CSEM and tune the relative discrete-loss weight over $\{0.1, 1, 10\}$ on the in-distribution validation split, selecting $\lambda = 1$. We report the first generated sample rather than oracle best-of-N performance.

D Temporal Mode Inference: Experimental Details

D.1 Data

We simulate a ball moving under gravity in a 3D room containing up to two box obstacles. Candidate contact surfaces include the floor, walls, ceiling, and obstacle faces. Each step has one latent mode: `flight`, `impact`(s), `roll`(s), or `rest`(s), where s identifies a scene surface. Restitution, tangential retention, and rolling friction are randomized within fixed physical ranges. An event-driven simulator generates trajectories and rejects ambiguous multi-contact and edge-contact steps.

Observations are ball positions corrupted by 2 cm Gaussian noise. Positions at the first two steps and then every fifth step are observed; the sparse setting observes every tenth step. Velocities and physical modes are never observed. Factors are trained on 3,000 complete trajectories of length $L = 20$ without conditioning on an observation pattern.

Table 15: **Temporal-inference evaluation sets.** Observation stride is the number of steps between observed positions.

Setting	Horizon L	Observation stride	Scenes
In-distribution	20	5	100
Long	50	5	40
Long	100	5	20
Sparse	50	10	40

Longer horizons and increased observation sparsity are therefore separate zero-shot shifts for CSEM.

D.2 Baselines

JointDiff (Capellera et al., 2026) uses a transformer to jointly denoise continuous positions and predict discrete modes. The ALiBi variant (Press et al., 2022) replaces absolute temporal positions with linear attention biases to improve horizon extrapolation. The autoregressive bridge predicts positions sequentially between consecutive observation anchors. All baselines receive the same scene geometry, observations, and candidate mode table as CSEM.

The baselines are trained on $L = 20$ trajectories. If observation strides 5 and 10 are both used during their training, the sparse setting is OOD only for CSEM; we retain this advantage for the baselines but do not describe it as a shared zero-shot comparison. Each baseline generates 16 samples, but reported non-oracle results use the first sample. Oracle best-of-16 values are used only as a diagnostic.

D.3 Results and ablations

Table 16: **Temporal mode inference** (%). Methods train at $L = 20$. Trajectory validity measures consistency with the inferred dynamics; mode accuracy uses ground-truth mode labels.

	Trajectory validity				Mode accuracy			
Horizon L	20	50	100	50 sparse	20	50	100	50 sparse
JointDiff	33	2	0	3	92	57	32	53
+ ALiBi	39	11	1	13	**93**	62	30	54
Autoregressive bridge	40	16	8	19	91	58	32	48
CSEM	**83**	**74**	**72**	**79**	85	**67**	**36**	**57**
w/o edits	62	51	49	54	79	62	30	53
max-risk selection	69	64	68	66	82	64	35	**57**

CSEM maintains substantially higher trajectory validity as the horizon increases and observations become sparse. Mode-accuracy differences are smaller, indicating that the main challenge is producing positions that precisely satisfy the inferred dynamics rather than identifying broad motion phases. Removing structural edits reduces trajectory validity by 21–25 points.

At $L = 20$, uniform blame obtains 81% trajectory validity, compared with 83% for calibrated blame, because the revision budget covers most editable spans. Ground-truth modes raise CSEM's trajectory validity to 90%, leaving a seven-point continuous-optimization gap. Oracle best-of-16 trajectory validity for JointDiff, ALiBi, and the autoregressive bridge is respectively $50/6/1/8$, $55/18/4/27$, and $45/18/9/22$ across the four settings.

D.4 QUALITATIVE EXAMPLE

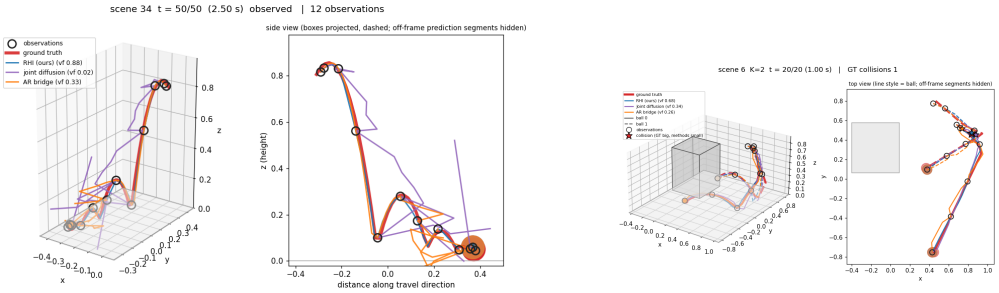


Figure 7: **Temporal inference from sparse observations.** Left: CSEM follows the ground-truth trajectory through two floor impacts, while the autoregressive and joint-diffusion baselines drift between observation anchors. Right: extension to a two-ball collision. Ground truth is red and observed positions are circled.

E STATISTICAL SIGNIFICANCE OF THE MAIN RESULTS

All results use one training seed per method and one plan per scene. For the solve rates of Table 1 (100 scenes per cell) we report 95% Wilson intervals; for the per-object hybrid accuracy of Table 3 and the per-scene trajectory validity and mode accuracy of Table 4 (100 scenes in distribution, 40 and 20 scenes at T=50 and T=100) we report 95% scene-level bootstrap intervals of the mean. Because every method is evaluated on the same scenes, we also test each margin with a paired test on the per-scene outcomes: an exact McNemar test for solve rates and a sign test for the per-scene metrics. Table 17 lists, for every CSEM cell, its interval and the p-values against the no-edit ablation and against the strongest baseline in that column.

Table 17: **Intervals and paired tests for Tables 1, 3 and 4.** CI: 95% Wilson (solve rates) or scene-level bootstrap (per-object and per-scene metrics). p: paired McNemar (solve rates) or sign test (other metrics) on the shared scenes; a dagger marks a cell where the baseline is significantly *better*.

Table	cell	CSEM	95% CI	p vs w/o edits	strongest baseline (p)
1, stacking n	4	76	67–83	0.09	Diffuser 71 (0.49)
	5	68	58–76	0.42	HDP 38 (<0.001)
	6	51	41–61	0.83	HDP 38 (0.09)
	7	53	43–62	1.00	HDP 27 (0.001)
	8	49	39–59	0.30	HDP 14 (<0.001)
1, rearrangement (n, k)	4,2	65	55–74	0.65	CDGS 29 (<0.001)
	6,2	66	56–75	0.006	CDGS 2 (<0.001)
	6,3	58	48–67	0.13	CDGS 15 (<0.001)
	8,2	42	33–52	0.06	CDGS 1 (<0.001)
	10,2	39	30–49	0.03	all 0 (<0.001)
	12,3	25	18–34	0.02	all 0 (<0.001)
1, packing K	2	92	85–96	<0.001	HDP 33 (<0.001)
	3	77	68–84	<0.001	ccsp-rand 2 (<0.001)
	4	63	53–72	<0.001	ccsp-rand 1 (<0.001)
	5	52	42–62	<0.001	all 0 (<0.001)
3, scene n	4	77	72–82	—	FF 61 (<0.001)
	6	68	64–72	—	FF 56 (<0.001)
	8	45	40–49	—	FF 44 (0.43)
	10	32	28–35	—	FF 40 (<0.001)†
	occ. 4	57	52–63	—	JointDiff 38 (<0.001)
	occ. 6	48	44–53	—	FF 22 (<0.001)
	occ. 8	35	32–39	—	FF 18 (<0.001)
4, trajectory validity T	20	83	81–86	—	AR bridge 40 (<0.001)
	50	74	71–77	—	AR bridge 16 (<0.001)
	100	72	70–74	—	AR bridge 8 (<0.001)
	50 sparse	79	76–82	—	AR bridge 18 (<0.001)
4, mode accuracy T	20	85	82–88	—	ALiBi 92 (<0.001)†
	50	67	62–72	—	ALiBi 61 (0.006)
	100	36	31–42	—	Joint diffusion 32 (<0.001)
	50 sparse	57	52–61	—	ALiBi 54 (0.004)

Every CSEM margin over the strongest baseline is significant at $p < 0.01$ except stacking n=4 and n=6 and scene n=8; FF is significantly better at scene n=10, and all temporal baselines are significantly better in-distribution on mode accuracy. Against the no-edit ablation, the gain is significant at every packing size and at three of six rearrangement sizes, and within noise at every stacking size, where revision acts inside the leaf solver (Appendix B.2).