Neural Dual Bounds: Valid-by-Construction JGLP Warm-Starts for MAP and Constrained MAP

Akshay Vyas
Department of Computer Science
The University of Texas at Dallas
akshay.vyas@utdallas.edu

Shivvrat Arya
Department of Computer Science
New Jersey Institute of Technology
shivvrat.arya@njit.edu

Brij Malhotra
Department of Computer Science
The University of Texas at Dallas
brijgulsharan.malhotra@utdallas.edu

Tahrima Rahman
Department of Computer Science
The University of Texas at Dallas
tahrima.rahman@utdallas.edu

Vibhav Gogate
Department of Computer Science
The University of Texas at Dallas
vibhav.gogate@utdallas.edu

Nicholas Ruozzi
Department of Computer Science
The University of Texas at Dallas
nicholas.ruozzi@utdallas.edu

Abstract

We present Neural Dual Bounds, a calibration-free framework that produces deterministic upper bounds for MAP inference in probabilistic graphical models using a single neural-network forward pass. The framework trains a network to output the edge messages of a mini-bucket style upper bound on a given join-graph decomposition in a self-supervised manner by directly minimizing a smooth dual objective. The output of this network can be used at test time to warm-start a standard JGLP optimizer. Across diverse benchmarks, this warm start yields dual gaps that are orders of magnitude tighter than those from zero initialization, with the largest gains in the early-iteration regime. We also apply our framework for the constrained MAP (CMPE) task, where neural warm starts with test-time adaptation are comparable to the adaptive baseline on Bayesian networks and achieve the tightest dual gaps, by one to two orders of magnitude, on the grid and segmentation families.

1 Introduction

Many applications of probabilistic graphical models (PGMs) repeatedly solve closely related inference problems on a fixed graph. Branch-and-bound search expands one node per query, adversarial robustness certification verifies one input per query, constrained planning evaluates one constraint per query, and bound-based parameter learning computes a new bound at each gradient step. In these settings, per-query inference often dominates wall-clock time. Classical polynomial-time dual bounds, including join-graph linear programming (JGLP) [9], Max-Product Linear Programming (MPLP) [6], and TRW-BP [18], provide valid upper bounds, but often require hundreds of coordinate-descent iterations per query. This repeated-query structure creates a natural opportunity for amortization, provided that amortization preserves the validity guarantees required by downstream algorithms.

Learned methods can amortize inference, but they typically sacrifice deterministic per-query validity. Learned bound predictors usually provide only statistical validity: if a predicted upper bound violates the true upper-bound property on even a small fraction of queries, branch-and-bound may

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

prune the optimal solution and robustness certificates may become invalid [11, 5, 10]. Conformal calibration [17, 2] can reduce this risk by adding a safety margin estimated from held-out data, but it still permits violations at the target residual failure rate. Learned inference networks based on graph neural networks [15, 19, 20] predict approximate inference outputs, but do not enforce an upper-bound certificate for every query. Existing learned methods therefore accelerate inference by weakening the certificate that makes classical dual bounds useful in exact and safety-critical pipelines.

This validity-speed tradeoff is especially important for constrained MAP inference, which seeks the most probable explanation subject to one or more budgets defined by a second log-linear model. This formulation extends standard MAP inference to settings with explicit side constraints and captures safety, fairness, adversarial, and resource-constrained reasoning problems [13, 12]. Existing upper-bounding methods for this setting optimize a single scalar Lagrange multiplier by subgradient descent [12]. They do not directly handle the vector-valued multipliers induced by multiple assurance requirements. With multiple constraints, projected gradient descent over the multiplier vector must solve an inner MAP-bound problem at every step, while queries may vary in evidence, costs, constraint budgets, or Lagrange multipliers. This nested optimization makes valid amortized bounds particularly valuable in the multi-constraint setting.

We introduce *Neural Dual Bounds*, a framework for amortized inference that produces deterministically valid upper bounds from a single neural-network forward pass. Instead of learning bound values directly, the network outputs antisymmetric edge messages on a join-graph decomposition. This parameterization restricts every output to a feasible subspace of the JGLP bound, so the resulting bound is valid by construction rather than by training accuracy, calibration, or a held-out quantile estimate. Antisymmetric messages preserve feasibility because each message added to one side of an edge is subtracted from the other, leaving the decomposed upper-bound objective unchanged while reparameterizing local potentials. The guarantee holds for trained, untrained, and adversarial network outputs.

The same construction also provides effective warm-starts for iterative dual optimization. For MAP inference, we train the model end-to-end against a smooth objective, without an LP solver in the inner loop and without JGLP-converged labels. For constrained MAP inference, we supervise the network with LP-optimal edge messages for the per-μ Lagrangians, since this supervision is cheaper than self-supervised training to comparable tightness in this setting. We also show that an LP-free training variant can achieve similar bound quality. Thus, the framework supports both direct amortized certification and accelerated iterative tightening.

Contributions.

1. **Deterministically valid neural bounds.** We restrict the neural-network output to a structurally valid subspace of the JGLP upper bounds using an antisymmetric scatter map. This construction yields, to our knowledge, the first learned MAP upper-bound predictor whose validity is enforced by architecture rather than by post-hoc calibration.
2. **Self-supervised dual amortization without LP labels.** We train the model end-to-end against a smooth dual objective, with no LP solver in the inner training loop and no JGLP-converged labels. This objective removes the need to generate optimal targets for MAP training and makes the method applicable when LP solves are unavailable or prohibitively expensive.
3. **Extension to multi-constraint constrained MAP inference.** We apply the framework to vector-valued Lagrange multipliers using projected gradient descent, neural warm-starts, and replay-buffer test-time adaptation at each step.
4. **Empirical validation on benchmark PGMs.** On 15 UAI benchmark PGMs spanning Bayesian networks, image segmentation, and grid Markov random fields, neural warm-starts perform comparably to zero-initialized JGLP in the worst case and up to six orders of magnitude tighter in the best case, with the largest gains in the early-iteration regime. On constrained MAP inference, replay-buffer test-time adaptation gives the tightest dual gaps on the grid and segmentation families at every iteration budget, while on bayesian networks it is matched by a no-learning baseline that simply reuses the previous multiplier step's converged solution.
5. **Structural applicability beyond JGLP.** We identify a broader class of decomposed Lagrangian duals whose message structure admits the same antisymmetric parameterization, including dual decomposition, TRW-BP, weighted CSPs, and minimum-cost flow.

2 Background

A probabilistic graphical model $(G, \mathcal{F}, \theta)$ defines a factored distribution of n discrete variables $\mathbf{x} = (x_1, \ldots, x_n)$ with domains $\mathcal{D}_i$ over a graph $G = (V, E)$ with cliques $\mathcal{F}$ as

$$p(\mathbf{x} \mid \theta) \propto \prod_{\alpha \in \mathcal{F}} \exp(\theta_\alpha(\mathbf{x}_\alpha)) \tag{1}$$

where $\mathcal{F}$ indexes cliques of G with scopes $\mathbf{x}_\alpha \subseteq \mathbf{x}$ and $\theta_\alpha : D_\alpha \to \mathbb{R}$ are known as the log-potential functions.

The MAP problem is to find an assignment that maximizes the probability distribution p or, equivalently, maximizes the $\log$ probability

$$\text{MAP}^*(\boldsymbol{\theta}) = \max_{\mathbf{x}} \sum_{\alpha \in \mathcal{F}} \theta_\alpha(\mathbf{x}_\alpha). \tag{2}$$

Exact MAP is NP-hard in general [16], and standard methods designed for this task, such as variable elimination, scale exponentially in the tree-width of G. As a result, a variety of approximation methods have been designed that approximately solve the MAP problem via upper bound minimization. The Mini-bucket elimination (MBE, 3), one such upper bound minimization approach, further regroups the clique potentials into clusters determined by a specified i-*bound* parameter. Specifically, the procedure induces a *join graph* $\mathcal{G} = (\mathcal{C}, \mathcal{E}, \theta)$ whose nodes $\mathcal{C} = \{C_1, \ldots, C_{|\mathcal{C}|}\}$ correspond to a subset of variables $\alpha \subseteq V$ with $|\alpha| \leq i + 1$ and edges in $\mathcal{E}$ connect clusters that share variables. For each edge $(\alpha, \beta) \in \mathcal{E}$, the *separator* $S_{\alpha\beta} = C_\alpha \cap C_\beta$ is the shared subset. Each cluster carries a log-potential function θ_{C_α} that aggregates the original log-potentials assigned to it during partitioning. MBE then uses the join graph clustering to produce an upper bound on the MAP value as

$$D_0 = \sum_{\alpha=1}^{|\mathcal{C}|} \max_{\mathbf{x}_{C_\alpha}} \theta_{C_\alpha}(\mathbf{x}_{C_\alpha}) \;\geq\; \text{MAP}^*, \tag{3}$$

where the inequality comes from the observation that a maximum of a sum is always less than or equal to the sum of maximums, i.e., $\max_x (f(x) + g(x)) \leq \max_x f(x) + \max_x g(x)$.

The bound (3) is loose because adjacent clusters maximize independently and may pick conflicting values on their shared separator. *Reparameterization* tightens the bound by shifting cost between adjacent clusters via edge messages $\lambda_{\alpha\beta}(\mathbf{x}_S) \in \mathbb{R}$:

$$\theta_{C_\alpha}(\mathbf{x}_{C_\alpha}, \lambda) = \theta_{C_\alpha}(\mathbf{x}_{C_\alpha}) + \sum_{\beta \in N(\alpha)} \lambda_{\alpha\beta}(\mathbf{x}_{S_{\alpha\beta}}), \tag{4}$$

subject to an *antisymmetry constraint*

$$\lambda_{\alpha\beta}(\mathbf{x}_S) + \lambda_{\beta\alpha}(\mathbf{x}_S) = 0, \qquad \forall (\alpha, \beta) \in \mathcal{E},\ \forall \mathbf{x}_S. \tag{5}$$

Antisymmetry preserves the underlying joint: for any complete assignment $\mathbf{x}^*$, the message added to one cluster on a shared separator is exactly the message subtracted from its neighbor, so $\sum_\alpha \theta_{C_\alpha}(\mathbf{x}^*_{C_\alpha}, \lambda) = \sum_\alpha \theta_{C_\alpha}(\mathbf{x}^*_{C_\alpha})$. Replacing the assignment-specific sum by per-cluster maxes can only increase the value, so the reparameterized bound

$$D(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \sum_{\alpha=1}^{|\mathcal{C}|} \max_{\mathbf{x}_{C_\alpha}} \theta_{C_\alpha}(\mathbf{x}_{C_\alpha}, \lambda) \;\geq\; \text{MAP}^* \tag{6}$$

holds for every $\boldsymbol{\lambda}$ in the antisymmetric subspace.

As $D(\boldsymbol{\theta}, \boldsymbol{\lambda})$ is convex in $\boldsymbol{\lambda}$ (a sum of pointwise maxima of affine functions), a variety of techniques can now be applied to find the minimum of this upper bound. The JGLP [9] method minimizes it by coordinate descent. Each iteration of JGLP corresponds to one pass over the separator edges: for $(\alpha, \beta) \in \mathcal{E}$, the algorithm computes the max-marginals of C_α and C_β onto $S_{\alpha\beta}$, equalizes them to their geometric mean (the arithmetic mean in log domain), and applies the corresponding antisymmetric correction to both cluster tables (Ihler et al. [9] Algorithm 4, lines 5–8). The fixed point is the optimum $\min_{\boldsymbol{\lambda}} D(\boldsymbol{\theta}, \boldsymbol{\lambda})$, which corresponds to the optimum of a specific LP relaxation of the MAP problem on the chosen join graph.

JGLP starts from a *moment-matched* mini-bucket join graph rather than raw MBE: at construction, one round of cost-shifting on the mini-buckets within each elimination bucket equalizes their max-marginals on the shared scope (Ihler et al. [9] Algorithm 5, "MBE-MM"). This produces a slightly tighter starting point than (3) before any JGLP iterations have run. Throughout this paper, "$\boldsymbol{\lambda} = \mathbf{0}$" denotes the JGLP coordinate-descent state at iteration zero *after* MBE-MM construction, matching the convention of all JGLP implementations we benchmark against.

3 Method

Reparameterizations of a join-graph decomposition [9, 18] are constrained to lie in the antisymmetric subspace defined by (5); any element of this subspace yields a valid bound. Existing methods (JGLP, MPLP, TRW-BP) navigate this subspace iteratively. However, convergence of JGLP often requires hundreds of iterations on the benchmark models we consider, with each iteration costing $\mathcal{O}(|\mathcal{E}| \cdot d^{i+1})$, where d is the maximum domain size. As a result, a poor initialization can be far from the tightest upper bound. This motivates our approach in which we train a neural network to directly predict the λs from a given partial assignment to the random variables. This approach is particularly applicable in typical scenarios in which a PGM has already been fit to data and we are simply using the PGM to make (repeated) MAP inference queries.

A few remarks. First, computing the MBE bound scales exponentially in the i-bound: polynomial-time for fixed i, but far from practical on large networks. Our aim is to move that cost from test time to training time, so that a network of sufficient capacity, given enough training configurations, approximates what the iterative methods compute and the test-time cost of a "good" bound reduces to a single forward pass. To efficiently encode the upper bound prediction task in a neural network architecture, we make a few small changes to the representation of the MAP prediction task.

First, we will train our neural network by directly minimizing the dual objective. So, no labeled training examples are required. This makes the approach flexible across many different models and does not require us to compute expensive i-bounds during training. To simplify training, we replace the hard $\max$ in the upper bound with a smooth log-sum-exp surrogate:

$$D_\tau(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \sum_\alpha \tau \log \sum_{\mathbf{x}_{C_\alpha}} \exp\left(\frac{\theta_{C_\alpha}(\mathbf{x}_{C_\alpha}, \lambda)}{\tau}\right), \tag{7}$$

where $\tau > 0$ is a temperature. As $\tau \to 0$, $D_\tau \to D$.

Proposition 1. *$D_\tau(\boldsymbol{\theta}, \boldsymbol{\lambda}) \geq D(\boldsymbol{\theta}, \boldsymbol{\lambda}) \geq \mathrm{MAP}^*$ for all $\tau > 0$ and all $\boldsymbol{\lambda}$.*

Proof. Step 1 ($D \geq \mathrm{MAP}^*$). Fix any complete assignment $\mathbf{x}$. Each edge contributes $\lambda_{\alpha\beta}$ to one cluster and $\lambda_{\beta\alpha}$ to the other, both at the same separator sub-assignment since $\mathbf{x}$ is one global assignment, so by (5) the pair cancels and

$$\sum_\alpha \theta_{C_\alpha}(\mathbf{x}_{C_\alpha}, \lambda) \;=\; \sum_{\gamma \in \mathcal{F}} \theta_\gamma(\mathbf{x}_\gamma),$$

each original log-potential having been assigned to exactly one cluster during partitioning (Sec. 2). Replacing each cluster term by its maximum only increases the sum, so D bounds the score of every $\mathbf{x}$, hence MAP^*.

Step 2 ($D_\tau \geq D$). Let M_α be the maximum reparameterized entry of cluster α and n_α its number of configurations. The sum $\sum_y \exp(\theta_{C_\alpha}(y, \lambda)/\tau)$ contains $\exp(M_\alpha/\tau)$ and all terms are positive, so $\tau \log$ of it is at least M_α; summing over clusters gives the claim. That same sum has n_α terms, none larger than $\exp(M_\alpha/\tau)$, which gives

$$0 \;\leq\; D_\tau - D \;\leq\; \tau \sum_\alpha \log n_\alpha.$$

The surrogate gap is $O(\tau)$, the quantitative form of $D_\tau \to D$, and why annealing τ is safe. Neither step uses any property of the network: Step 1 needs only that λ is antisymmetric, which A enforces by construction, and Step 2 only $\tau > 0$. Validity belongs to the parameterization; only tightness depends on training. □

$D(\boldsymbol{\theta}, \boldsymbol{\lambda})$ is convex in $\boldsymbol{\lambda}$ (sum of pointwise maxima of affine functions), so subgradient methods could be used directly, but we use the smooth surrogate D_τ for compatibility with standard gradient-based optimizers.

Prior amortized-inference work either supervises against a slower exact solver [11] or unrolls iterations of the target algorithm and backpropagates through them. We do neither: D_τ is a closed-form differentiable function, so a single backward pass through the smooth dual yields the gradient. The training distribution is evidence configurations sampled at varying observation ratios, with no inference labels of any kind. This makes training substantially cheaper than supervised alternatives and removes LP-solver availability as a deployment requirement.

Second, we represent λ more compactly. The antisymmetry constraint (5) couples the two directions of every edge, so it suffices to store one real number per (undirected-edge, separator-assignment) pair. Stack these into a single edge-message vector $m \in \mathbb{R}^{\mathrm{edim}}$, where

$$\mathrm{edim} \;=\; \sum_{(\alpha,\beta)\in\mathcal{E}} \prod_{i\in S_{\alpha\beta}} d_i$$

counts those pairs and d_i is the domain size of variable i. Let $\lambda_{\mathrm{dim}} = \sum_\alpha \prod_{i\in C_\alpha} d_i$ be the total number of cluster-table entries, and let $\theta_{\mathrm{flat}} \in \mathbb{R}^{\lambda_{\mathrm{dim}}}$ be the concatenation of all cluster tables, ordered so that the α-th block of size $\prod_{i\in C_\alpha} d_i$ holds the entries $\{\theta_{C_\alpha}(x_{C_\alpha})\}_{x_{C_\alpha}}$ of cluster α. The reparameterization is then a single linear map

$$\theta'_{\mathrm{flat}} \;=\; \theta_{\mathrm{flat}} + A\, m,$$

where $A \in \{-1, 0, +1\}^{\lambda_{\mathrm{dim}}\times \mathrm{edim}}$ is a fixed sparse matrix determined entirely by the join-graph structure. Each column of A corresponds to one (edge, separator-assignment) pair $\big((\alpha,\beta), x_S\big)$ and contains $+1$ in every row of θ_{flat} that lies in cluster α and is consistent with x_S, -1 in every row that lies in cluster β and is consistent with x_S, and 0 elsewhere. The matched ± 1 pattern is exactly the antisymmetry of the reparameterization (5): a complete assignment selects one entry per cluster, so the $+1$ and -1 contributions of each column cancel. (The column itself sums to zero only when the two clusters on an edge have equally many entries consistent with $\mathbf{x}_S$; validity does not depend on this.)

As the neural network essentially produces λs that satisfy the antisymmetry constraint, its output can always be turned into a valid upper bound on the MAP problem. Consequently, training never has to enforce feasibility as a constraint or check it post-hoc; the architecture restricts predictions to the valid set. This is structurally different from calibration-based methods, which produce predictions in an unrestricted space and use a held-out calibration set to estimate the residual violation probability.

Finally, note that the upper bounds produced by our methods and the iterative methods in the best case yield the tightest upper bound on MAP* attainable by the given join graph decomposition, but this bound is tight only in special cases, e.g., tree-structured models or log-supermodular models [14].

4 Experiments

We evaluate on 15 PGMs from the UAI 2022 inference competition, covering Bayesian networks, image segmentation and grid Markov random fields. The headline results are Table 1 (MAP) and Tables 2–3 ($K{=}2$ CMPE); full results for $K{=}1$ CMPE are in App. B.17. All experiments use i-bound 4, with 100 evidence seeds per PGM for MAP and 50 for CMPE (the 37-PGM suite of App. B.17 remains at 10). Our reference D^{LP} is the exact LP optimum solved with CVXPY/CLARABEL (Appendix B.1); The JGLP coordinate descent occasionally terminates at non-globally-optimal fixed points, so we use the LP optimum rather than JGLP's converged value as the ground-truth lower bound on the dual.

4.1 Neural Network Training vs. JGLP

The primary evaluation of our neural bounds is how much iterative inference they save. A single neural forward pass (0.23–0.39 ms on CPU, App. B.16) produces a valid upper bound on MAP*; this bound is typically tighter than the MBE starting point and can serve as a warm-start for JGLP. We

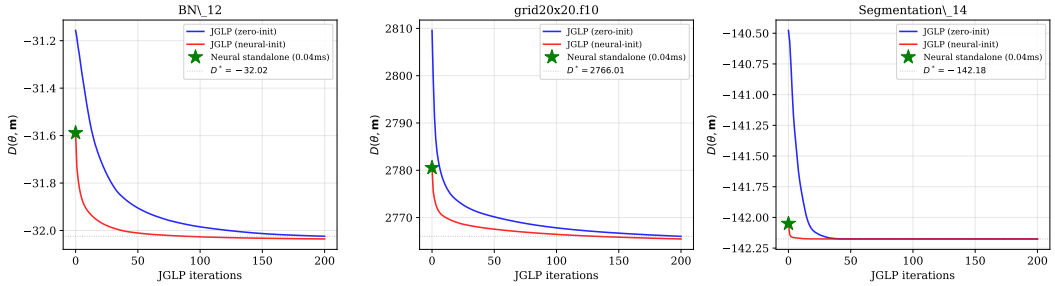


Figure 1: **JGLP convergence with and without neural warm-start** on three representative PGMs (BN_12, grid20×20.f10, Segmentation_14). Both curves provably converge to the same optimum D^* since $D(\boldsymbol{\theta}, \mathbf{m})$ is convex in $\mathbf{m}$. The green star marks the standalone neural bound from a single forward pass (~ 0.04 ms on GPU) and is used as the warm-start for JGLP at iteration 0; it starts strictly below zero-init and reaches a small neighborhood of D^* in fewer JGLP iterations. The three panels span the BN, Grid, and Segmentation families; full results across all 37 PGMs are reported in Appendix B.2 and Table 1.

measure the effective speedup by counting JGLP iterations needed to reach a target relative error from the converged dual D^*, comparing zero-init vs neural-init.

We use a 3-layer MLP with LayerNorm and GELU activations: $512 \to 512 \to 256 \to e_{\text{dim}}$. The output layer has no activation function (edge messages are unconstrained). The output is initialized with small random weights (std $= 0.001$) so that the initial bound $D(f_\phi(\mathbf{e})) \approx D_0$ (the MBE bound).

We anneal τ, the temperature parameter of the smooth surrogate, during training: $\tau = 0.01$ for the first third, then $\tau = 0.001$ for the remaining epochs. We use AdamW optimization with cosine annealing (lr $= 10^{-3}$, 500 gradient steps at batch size 16, drawing a fresh evidence configuration per sample). Training data consists of evidence configurations sampled at various evidence ratios (10–90% of variables observed). No JGLP or other iterative inference is required during training.

Since $D(\boldsymbol{\theta}, \mathbf{m})$ is a convex function of m, we can compute the exact optimum via CVXPY/CLARABEL (formulation in Appendix B.1). Figure 1 shows one PGM per family; Table 1 reports the absolute gap $D^{(t)} - D^{\text{LP}}$ at fixed JGLP iteration checkpoints $t \in \{10, 50, 100\}$, mean ± std over 100 bin-0 test seeds per PGM. Lower is better.

Across the 15 PGMs, neural warm-start matches or beats zero-init at iter=10 on 100% of seeds for 11 of them. The gap then closes rapidly as JGLP iterates: on the Segmentation family it drops by 4–6 orders of magnitude by iter=100—for example, Seg_12's neural-init gap falls from 0.701 to 2×10^{-7} in App. B.17—while BN models show smaller but consistent gains (∼1–2 orders). Three regimes resist this pattern in the full 37-PGM suite (App. B.17). (i) *Degenerate grids* (grid10×10, grid40×40.f2) already sit within 10^{-8} of D^{LP} at MBE construction with i-bound 4; neural cannot improve, and the small overshoot at iter=10 is floating-point round-off, not a training failure. (ii) *Promedas networks* have near-zero dual values, so absolute gaps look small even when relative gains are substantial. (iii) *Seg_14* converges fast enough on its own that zero-init reaches 2×10^{-6} by iter=50, below the neural arm; both are then at the LP solver's working accuracy, so that ordering is not a result.

Amortized cost analysis. A single-query forward pass costs 0.23–0.39 ms on CPU (the ∼0.04 ms figure quoted for batched-GPU throughput is not the per-query latency), against 1.6–131 ms for one JGLP iteration depending on PGM complexity, so the prediction is a fraction of an iteration and is recovered whenever it saves one. Per-PGM wall-clock for constrained MAP is reported in App. B.6. The defensible claim is bound quality at a fixed *iteration* budget — what a branch-and-bound node expansion actually spends — rather than wall-clock dominance at every scale.

Edge vs cluster parameterization. The edge-message parameterization (e_{dim}) is 2× smaller than the cluster-table parameterization (λ_{dim}) while capturing 98.6% of JGLP's corrections (lstsq residual 1.4%). The reduced output dimension improves generalization: self-supervised training with edge

Table 1: **Standalone bound quality**: dual gap $|D - D^{\mathrm{LP}}|$ (mean $\pm$ std) at fixed JGLP iterations. J = zero-init, N = neural-init; lower is closer to the LP optimum, so **bold** marks the neural arm where it ties or beats zero-init within one standard deviation. Mean and population standard deviation over 100 bin-0 test seeds per PGM (Segmentation_18 and _19 have 99: one seed each fails deterministically in the LP solver and is reported rather than dropped). All five checkpoints $\{0, 10, 50, 100, 500\}$ are in App. B.17; the paired per-seed comparison, which removes the shared instance difficulty these standard deviations are dominated by, is Table 31. †Same UAI model as Grids_29 under a second name; ‡a distinct 400-variable grid, itself shipped as Grids_30. Both rows of a duplicated pair are kept because each carries an independently trained network, but every count is over the 15 distinct models rather than the 16 rows (App. B.2).

	iter=10		iter=50		iter=100	
PGM	J	N	J	N	J	N
BN_12	0.525 ± 0.248	**0.304 ± 0.178**	0.150 ± 0.131	**0.095 ± 0.093**	0.065 ± 0.078	**0.042 ± 0.056**
BN_13	0.469 ± 0.129	**0.267 ± 0.103**	0.113 ± 0.071	**0.089 ± 0.070**	0.055 ± 0.050	**0.047 ± 0.048**
BN_14	0.305 ± 0.142	**0.191 ± 0.113**	0.058 ± 0.060	**0.045 ± 0.051**	0.015 ± 0.028	**0.013 ± 0.024**
BN_9	0.047 ± 0.061	**0.039 ± 0.055**	2.96e-03 ± 9.89e-03	**3.09e-03 ± 0.011**	6.27e-04 ± 2.63e-03	**6.97e-04 ± 3.07e-03**
Grids_26	2.70 ± 2.11	**2.71 ± 2.10**	0.237 ± 0.500	**0.237 ± 0.500**	0.036 ± 0.159	**0.037 ± 0.160**
Grids_28	4.18 ± 3.41	**4.19 ± 3.40**	0.388 ± 0.813	**0.388 ± 0.814**	0.068 ± 0.295	**0.069 ± 0.297**
Grids_29	1.26 ± 0.917	**1.24 ± 0.912**	0.123 ± 0.223	**0.123 ± 0.222**	0.021 ± 0.091	**0.021 ± 0.092**
Seg_12	1.06 ± 0.615	**0.793 ± 0.523**	0.135 ± 0.220	**0.142 ± 0.223**	0.032 ± 0.093	**0.034 ± 0.095**
Seg_14	0.345 ± 0.265	**0.123 ± 0.165**	1.54e-03 ± 9.44e-03	**1.04e-03 ± 5.33e-03**	1.58e-07 ± 2.87e-07	**5.63e-06 ± 7.88e-06**
Seg_16	1.88 ± 0.514	**0.381 ± 0.379**	0.253 ± 0.213	**0.025 ± 0.071**	0.017 ± 0.045	**3.06e-04 ± 1.87e-03**
Seg_18	0.873 ± 0.632	**0.576 ± 0.491**	0.065 ± 0.120	**0.058 ± 0.111**	5.23e-03 ± 0.034	**5.68e-03 ± 0.033**
Seg_19	0.152 ± 0.207	**0.110 ± 0.147**	0.015 ± 0.050	**0.014 ± 0.044**	4.43e-03 ± 0.022	**4.15e-03 ± 0.019**
grid10×10.f5	0.311 ± 0.343	**0.302 ± 0.337**	6.65e-03 ± 0.042	**6.48e-03 ± 0.042**	4.61e-04 ± 3.55e-03	**4.40e-04 ± 3.43e-03**
grid10×10.f10	0.536 ± 0.799	**0.537 ± 0.792**	0.079 ± 0.214	**0.079 ± 0.212**	0.015 ± 0.090	**0.015 ± 0.089**
grid20×20.f5†	1.26 ± 0.917	**1.24 ± 0.912**	0.123 ± 0.223	**0.123 ± 0.222**	0.021 ± 0.091	**0.021 ± 0.092**
grid20×20.f10‡	14.39 ± 3.37	**12.43 ± 3.03**	5.08 ± 1.92	**4.81 ± 1.87**	2.77 ± 1.41	**2.71 ± 1.43**

messages achieves 94% neural-better vs. 0% with direct cluster-table prediction (which violates the reparameterization constraint).

Self-supervised vs supervised training. We compare pure self-supervised training (directly minimizing D_τ) against supervised training with JGLP edge-message targets. On BN_12 bin 0: self-supervised achieves 94% neural-better vs 72% for MSE-supervised. The MSE model produces larger magnitude edge shifts ($2.8\times$ scale ratio) but sometimes overshoots the optimal reparameterization, resulting in 28% of instances worse than zero-init. The self-supervised loss naturally avoids this by directly optimizing the bound quality. (App. B.13)

4.2 Constrained Most Probable Explanation

Rahman et al. [12] consider a constrained version of the most probable explanation task that allows constraints that are given by PGMs that, for simplicity, are over the same graph and factorization. Formally, let $(G, \mathcal{F}, \theta)$ be a PGM. Consider K constraint PGMs $(G, \mathcal{F}, \theta^k)$ for each $k \in \{1, \ldots, K\}$ over the same graph and factorization with possibly different potential functions. Each constraint will be of the form $\sum_{\alpha \in \mathcal{F}} \theta^k_\alpha(\mathbf{x}_\alpha) \leq q_k$ for some $q_k \in \mathbb{R}$. This yields the optimization problem

$$\max_{\mathbf{x}} \sum_{\alpha \in \mathcal{F}} \theta_\alpha(\mathbf{x}_\alpha) \text{ subject to}$$
$$\sum_{\alpha \in \mathcal{F}} \theta^k_\alpha(\mathbf{x}_\alpha) \leq q_k \text{ for each } k \in \{1, \ldots, K\}.$$

Note that we can apply the same argument to the case where each one of the PGMs is replaced by the same join graph (again with possibly different log-factors). To handle constrained optimization problems of this form, Rahman et al. [12] consider a Lagrangian relaxation

$$\mathcal{L}(\boldsymbol{\mu}) = \max_{\mathbf{x}} \left[\sum_{\alpha \in \mathcal{F}} \theta_\alpha(\mathbf{x}_\alpha) - \sum_k \mu_k \left(\sum_{\alpha \in \mathcal{F}} \theta^k_\alpha(\mathbf{x}_\alpha) - q_k \right) \right]$$
$$\leq \sum_{\alpha \in \mathcal{F}} \max_{\mathbf{x}_\alpha} \left[\theta_\alpha(\mathbf{x}_\alpha) - \sum_k \mu_k \theta^k_\alpha(\mathbf{x}_\alpha) \right] - \sum_k \mu_k q_k.$$

Since the upper bound on $\mathcal{L}$ is convex in $\mu \geq 0$, we can minimize it via projected (sub)gradient descent. To compute the subgradients, we must compute the $\max$ in the lower bound, which can be done (approximately) with JGLP. As a result, we can train a μ-conditioned neural model for this task: the input is (evidence_encoding, μ) and the output is edge messages $\mathbf{m}$. Training data consists of 100 evidence configurations $\times$ 50 values of $\mu \in [0, 3]$, yielding $5{,}000$ samples per PGM.

Following [12], we construct each CMPE instance by perturbing the log-potentials of the original model: for every clique α and assignment x_α, we draw $\epsilon \sim \mathcal{N}(0, \sigma^2)$ i.i.d. per table entry and set $\theta^k_\alpha(x_\alpha) = \theta_\alpha(x_\alpha) + \epsilon$. The constraint threshold q_k is the 80th percentile of $\sum_\alpha \theta^k_\alpha(x_\alpha)$ on the test sample. We use $\sigma = 0.5$ to keep the constraint active at the optimal multiplier ($\mu^* > 0$); at $\sigma = 0.1$ the combined Lagrangian collapses to noise near μ^* (App. B.5). The two-constraint case draws independent noise and its own threshold for each k.

We compare four different methods to estimate the subgradients: (1) **JGLP**— initialized at $\mathbf{m} = 0$; (2) **NN**—inject NN-predicted $\mathbf{m}$ then run JGLP; (3) **JGLP+WS**—warm-start JGLP from $\mathbf{m}$ given by the most recent run of JGLP; (4) **NN+TTA**—test-time adaptation, at each gradient step k, fine-tune the NN on all previous $(\mu_j, \boldsymbol{\theta}^{(\mu_j)})$ pairs via the self-supervised D_τ loss (replay buffer of up to 20 entries, 10 inner gradient steps). The validity guarantee $L(\mu) \geq \text{CMPE}^*$ holds for any $\mathbf{m}$.

The buffer is FIFO and the inner loss is identical to the offline objective, so no separate test-time validation is required and the buffer accumulates no out-of-distribution examples by construction.

Results. We run 100 gradient steps on μ and at each step run JGLP from each initialization for up to 500 iterations (or until $|D(t) - D(t-1)| < 10^{-6}$). Tables 2–3 report the dual gap $(L(\mu) - L^{\text{LP}}(\mu))$ at fixed inner JGLP iteration checkpoints, as mean $\pm$ std over 50 evidence seeds $\times$ 100 μ-grad steps; the $K{=}1$ counterpart is Table 24 in App. B.17.

Test-time replay adaptation drives the dual gap $(L(\mu) - L^{\text{LP}}(\mu))$ closer to zero than zero-init on every non-degenerate PGM: by roughly one order of magnitude at iter=50 on the BN and Segmentation families (4.7–$18\times$ at $K{=}1$, 3.2–$14\times$ at $K{=}2$), and by up to four orders on grid10×10.f5, where zero-initialized JGLP stalls. Per-seed win rates at iter=10 ($n = 50$ seeds) are 94–100% on the BN and Segmentation models and 84–100% on the non-degenerate grids; zero-init wins only on the two degenerate grid40×40 rows (4% and 14%), where MBE already sits within 10^{-8} of L^{LP} (App. B.25). Even offline NN warm-start (M2) helps on most PGMs, with M3 additionally specializing to the local μ trajectory via the replay buffer.

JGLP+WS (no learning, just carrying converged $\Delta\lambda$ across μ-steps) is the strongest non-neural baseline. On Bayesian networks (BN_9, BN_12, BN_13, BN_14), μ-trajectories converge quickly and successive optimal λ values are nearby, so λ-continuity alone closes most of the gap (e.g., BN_12 iter=10: 0.082 ± 0.095 vs NN+TTA 0.097 ± 0.126). On grids and segmentation models, μ keeps moving across the trajectory and stale $\Delta\lambda$ pulls JGLP toward the previous optimum; NN+TTA's learned structural prior dominates by one to two orders of magnitude (Seg_14 iter=10: JGLP+WS 9.49 ± 5.87 vs NN+TTA 0.083 ± 0.314; Grids_28: 1019.27 ± 884.50 vs 52.97 ± 90.61). The two adaptation channels are complementary: λ-continuity exploits local smoothness in the optimization landscape, while the learned prior captures global structure that holds across the trajectory. We leave a hybrid (carrying $\Delta\lambda$ *and* TTA-fine-tuning the network) to future work.

Test-Time Adaptation The offline training distribution for the neural network sampled $\mu \sim \text{unif}(0, 1)$, but the projected gradient descent on μ at test time is unconstrained above and routinely drives μ outside this range: across all 15 PGMs, 36% of $K{=}1$ queries and 25% of $K{=}2$ queries occur at $\mu > 1$, with maxima of 36.1 and 19.4 respectively. A frozen offline-trained network on these tails would predict edge messages fitted to a different Lagrangian; deterministic validity (Prop. 1) still holds, but tightness degrades. Replay-buffer TTA continually re-fits the network to the local μ-region the trajectory has actually visited, recovering tightness within ~ 50 inner JGLP iterations even on tails 20–$36\times$ the training upper bound (Tables 2–3). This explains the large NN+TTA-vs-NN gap on grids and segmentations, where μ keeps moving across the trajectory: NN+TTA's primary job is online distribution adaptation, not just specialization.

Table 2: **CMPE bound quality** (two constraints, single-pass methods): absolute dual gap $|L(\mu) - L^{\mathrm{LP}}(\mu)|$ (mean $\pm$ std) at three fixed inner JGLP iteration budgets on the 15 PGM rows (14 distinct models) common to MAP, CMPE $K = 1$ and CMPE $K = 2$. Methods: **JGLP** = zero-init JGLP (no learning, no continuity); **NN** = single neural forward pass producing edge messages, frozen weights. Statistics aggregate over (seed $\times$ μ-gradient step) trials (50 seeds $\times$ 100 μ-steps each, i.e. 5,000 trials per cell).

	iter=10		iter=50		iter=100	
PGM	JGLP	NN	JGLP	NN	JGLP	NN
BN_12	0.622 ± 0.152	0.523 ± 0.136	0.207 ± 0.068	0.164 ± 0.065	0.107 ± 0.052	0.087 ± 0.047
BN_13	0.461 ± 0.127	0.351 ± 0.123	0.121 ± 0.109	0.063 ± 0.051	0.055 ± 0.107	0.025 ± 0.028
BN_14	0.305 ± 0.126	0.240 ± 0.121	0.050 ± 0.082	0.024 ± 0.033	0.020 ± 0.076	6.60e-03 ± 0.014
BN_9	0.179 ± 0.090	0.167 ± 0.074	0.053 ± 0.062	0.027 ± 0.035	0.026 ± 0.055	9.18e-03 ± 0.019
Grids_26	36.07 ± 52.99	37.03 ± 52.82	4.42 ± 11.52	4.43 ± 11.51	0.664 ± 2.63	0.663 ± 2.63
Grids_28	82.68 ± 120.14	84.23 ± 120.35	9.15 ± 24.42	9.18 ± 24.46	1.05 ± 4.68	1.06 ± 4.70
Grids_29	7.71 ± 11.94	8.30 ± 11.90	0.775 ± 2.48	0.781 ± 2.43	0.122 ± 0.588	0.121 ± 0.572
Seg_12	0.709 ± 0.721	1.05 ± 0.969	0.094 ± 0.179	0.168 ± 0.333	0.022 ± 0.076	0.055 ± 0.160
Seg_14	0.828 ± 1.02	0.832 ± 0.879	0.023 ± 0.063	0.017 ± 0.051	2.83e-03 ± 0.018	1.91e-03 ± 0.015
Seg_16	2.37 ± 2.08	1.51 ± 1.58	0.185 ± 0.263	0.087 ± 0.176	0.014 ± 0.059	8.58e-03 ± 0.048
Seg_18	0.883 ± 0.830	0.628 ± 0.586	0.057 ± 0.116	0.031 ± 0.062	3.39e-03 ± 0.027	2.33e-03 ± 7.20e-03
Seg_19	2.54 ± 3.53	2.06 ± 2.59	0.600 ± 1.10	0.467 ± 0.864	0.305 ± 0.691	0.235 ± 0.558
grid10×10.f5	0.172 ± 0.280	0.244 ± 0.244	0.044 ± 0.183	4.58e-03 ± 0.037	0.041 ± 0.180	1.42e-04 ± 1.33e-03
grid20×20.f5	7.71 ± 11.94	8.33 ± 11.87	0.775 ± 2.48	0.774 ± 2.45	0.122 ± 0.588	0.121 ± 0.579
grid20×20.f10	207.15 ± 202.17	204.66 ± 201.72	64.52 ± 68.23	63.58 ± 68.39	33.86 ± 39.42	33.48 ± 39.47

Table 3: **CMPE bound quality** (two constraints, cross-μ-step methods): absolute dual gap $|L(\mu) - L^{\mathrm{LP}}(\mu)|$ (mean $\pm$ std) at three fixed inner JGLP iteration budgets on the same 15 rows. Methods: **JGLP+WS** = JGLP warm-started by injecting the converged cluster shift from the previous μ-step (no neural network; tests whether free λ-continuity alone suffices); **NN+TTA** = neural warm-start with replay-buffer test-time adaptation of the NN weights across μ-steps (ours). Statistics aggregate over (seed $\times$ μ-gradient step) trials (50 seeds $\times$ 100 μ-steps each, i.e. 5,000 trials per cell).

	iter=10		iter=50		iter=100	
PGM	JGLP+WS	NN+TTA	JGLP+WS	NN+TTA	JGLP+WS	NN+TTA
BN_12	0.082 ± 0.095	0.097 ± 0.126	0.030 ± 0.038	0.046 ± 0.054	0.019 ± 0.026	0.029 ± 0.034
BN_13	0.065 ± 0.096	0.044 ± 0.082	0.015 ± 0.030	0.014 ± 0.027	7.20e-03 ± 0.015	6.93e-03 ± 0.015
BN_14	0.072 ± 0.110	0.027 ± 0.071	7.09e-03 ± 0.022	4.80e-03 ± 0.018	1.76e-03 ± 9.29e-03	1.61e-03 ± 8.25e-03
BN_9	0.028 ± 0.046	0.021 ± 0.046	2.47e-03 ± 7.30e-03	4.63e-03 ± 0.016	5.32e-04 ± 2.85e-03	1.72e-03 ± 8.12e-03
Grids_26	428.95 ± 370.99	16.02 ± 34.12	13.75 ± 13.29	2.13 ± 7.35	1.30 ± 2.84	0.337 ± 1.86
Grids_28	1019.27 ± 884.50	52.97 ± 90.61	30.34 ± 28.43	6.23 ± 18.70	2.46 ± 5.10	0.717 ± 3.79
Grids_29	92.55 ± 80.50	1.60 ± 5.65	2.84 ± 3.41	0.163 ± 1.07	0.301 ± 0.730	0.024 ± 0.236
Seg_12	6.32 ± 3.47	0.129 ± 0.362	0.537 ± 0.598	0.022 ± 0.088	0.124 ± 0.226	6.18e-03 ± 0.038
Seg_14	9.49 ± 5.87	0.083 ± 0.314	0.389 ± 0.446	3.00e-03 ± 0.022	0.030 ± 0.084	8.46e-04 ± 9.08e-03
Seg_16	9.02 ± 5.04	0.220 ± 0.676	0.418 ± 0.346	0.017 ± 0.076	0.019 ± 0.066	1.24e-03 ± 0.015
Seg_18	6.19 ± 3.12	0.085 ± 0.262	0.219 ± 0.274	4.17e-03 ± 0.026	0.018 ± 0.041	3.57e-04 ± 4.78e-03
Seg_19	12.03 ± 8.07	0.436 ± 1.25	1.04 ± 1.29	0.188 ± 0.534	0.371 ± 0.766	0.125 ± 0.382
grid10×10.f5	2.30 ± 2.11	0.015 ± 0.072	0.022 ± 0.076	3.89e-04 ± 8.73e-03	7.00e-04 ± 5.47e-03	5.96e-06 ± 2.58e-04
grid20×20.f5	92.55 ± 80.50	1.60 ± 5.65	2.84 ± 3.41	0.162 ± 1.07	0.301 ± 0.730	0.024 ± 0.237
grid20×20.f10	913.11 ± 525.72	101.83 ± 134.02	150.23 ± 49.77	41.94 ± 52.83	48.58 ± 29.88	24.49 ± 32.52

5 Related Work

Learning to branch / Learning for combinatorial optimization. A growing literature uses neural networks to guide search in combinatorial optimization [11, 5, 10]. These methods typically learn heuristics (branching decisions, node selection) rather than bounds, and provide no formal guarantees on bound validity. Concurrently, [7] learn cutting sets for integer programming, but require problem-specific cut generation. Our approach is orthogonal: we learn *dual bounds* rather than search heuristics, with deterministic validity guarantees.

Amortized optimization. Amortized optimization [1] trains networks to predict solutions to optimization problems, trading per-instance computation for upfront training cost. Our work falls in this category but with a crucial difference: the "feasibility" of our predictions (validity of the bound) is guaranteed by the architecture, not learned. This contrasts with approaches that learn feasible solutions and must verify post-hoc [4].

Differentiating from RL for MILP. The RL-for-MILP literature [11, 10] learns search heuristics (branching, node selection) via reinforcement learning, providing no formal bound guarantees. Our approach differs in three key ways: (1) we learn *bounds* rather than heuristics, via self-supervised

optimization (not RL); (2) we exploit PGM-specific reparameterization structure (antisymmetric messages) that has no analog in generic MILP; and (3) our bounds are *deterministically* valid by construction, not statistically calibrated. The deterministic guarantee is the core contribution: a learned bound that occasionally violates $D \geq \text{MAP}^*$ is useless for pruning, while ours is safe by design.

6 Conclusion

On UAI benchmark PGMs spanning Bayesian networks, image segmentation, and grid Markov random fields, neural warm-starts match or beat zero-initialized JGLP everywhere except the degenerate grids, where the mini-bucket bound already sits at the LP optimum, and are up to six orders of magnitude tighter in the best case, with the gains concentrated in the early-iteration regime (Table 1). On constrained MAP with one or two constraints, neural warm-start with replay-buffer test-time adaptation gives the tightest dual gaps on the grid and segmentation families by one to two orders of magnitude, while λ-continuity is comparable on Bayesian networks (Tables 2–3; full $K{=}1$ in Appendix B.17).

None of this is conditional on the network being right. The guarantee $D(\theta - \mu g, m) \geq \text{CMPE}^*$ holds throughout, unconditionally on the network output, with no calibration set, conformal quantile, or coverage budget. This is the line between a usable bound and an unusable one: a learned bound that violates $D \geq \text{MAP}^*$ even rarely cannot be used for pruning in branch-and-bound or for certification, because one violation discards the optimal solution or invalidates the certificate. Here validity is carried by the antisymmetric parameterization rather than by training accuracy, so it survives untrained and adversarial weights alike, at every point along the μ-trajectory. Training buys tightness; the architecture buys correctness.

Beyond JGLP. The framework extends to any Lagrangian dual whose decomposition admits antisymmetric messages—dual decomposition, TRW-BP, weighted CSPs, minimum-cost flow—and to non-PGM problems with the same structure. (i) *SVM/SMO*: the SVM dual is concave in the multipliers $\boldsymbol{\alpha}$ and linear in the data, structurally identical to $D(\boldsymbol{\theta}, \lambda)$, so a network predicting warm-start $\boldsymbol{\alpha}$ would yield valid dual bounds by construction and accelerate sequential minimal optimization where many SVMs are trained on similar data. (ii) *Network flow*: the LP dual of minimum-cost flow carries antisymmetric dual variables on edges, directly analogous to our edge-message parameterization, so the same scatter matrix A applies. (iii) *Weighted CSPs*: soft-constraint tables are log-potentials on the same join-graph structure, so the construction applies unchanged, though our suite (App. B.10) does not yet include them. In each case the architecture and the self-supervised loss are identical; only A changes.

Limitations. We state the scope of our claims directly. The network is trained per join graph; a new topology, or the same graph at a different i-bound, requires retraining, so the amortization we demonstrate is over repeated queries on a fixed topology rather than across topologies. The benefit is confined to the early-iteration regime: both initializations descend the same convex objective to the same optimum, so the warm start buys the shape of the first few tens of iterations, not the converged bound. Where the moment-matched mini-bucket bound already sits at the LP optimum (grid10$\times$10, grid40$\times$40.f2), no reparameterization can improve it and the neural start cannot help. And on Bayesian networks, where successive μ-step optima are close, λ-continuity is the stronger strategy; the learned prior dominates where the trajectory drifts.

Outlook. Two directions follow. First, JGLP+WS (λ-continuity, no learning) and NN+TTA (learning, no λ-continuity) dominate on disjoint families in Tables 2–3, which suggests they are complementary rather than competing: carrying $\Delta\lambda$ across μ-steps *and* fine-tuning the network should dominate either alone, particularly when test-time adaptation must absorb out-of-distribution μ. Second, and more consequential, our MLP is tied to a single join graph. Since A is determined entirely by the join-graph structure, a graph neural network with edge-level readouts transfers the parameterization unchanged and is the natural route to amortizing across topologies. Whether that closes the gap, or whether cross-topology transfer is harder than the construction makes it look, is the open problem we leave.

Acknowledgments and Disclosure of Funding

This work was funded in part by the DARPA ANSR, CODORD, and CLARA programs under Awards FA8750-23-2-0004, HR00112590089, and HR0011269E224, respectively and by the NSF grant 2327245.

References

[1] Brandon Amos. Tutorial on amortized optimization. *Foundations and Trends in Machine Learning*, 16(5):592–732, 2023.

[2] Anastasios N Angelopoulos and Stephen Bates. Conformal prediction: A gentle introduction. *Foundations and Trends in Machine Learning*, 16(4):494–591, 2023.

[3] Rina Dechter and Irina Rish. Mini-buckets: A general scheme for bounded inference. *Journal of the ACM (JACM)*, 50(2):107–153, 2003.

[4] Priya L Donti, David Rolnick, and J Zico Kolter. Dc3: A learning method for optimization with hard constraints. *arXiv preprint arXiv:2104.12225*, 2021.

[5] Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. *Advances in neural information processing systems*, 32, 2019.

[6] Amir Globerson and Tommi Jaakkola. Fixing max-product: Convergent message passing algorithms for map lp-relaxations. *Advances in neural information processing systems*, 20, 2007.

[7] Zeren Huang, Kerong Wang, Furui Liu, Hui-Ling Zhen, Weinan Zhang, Mingxuan Yuan, Jianye Hao, Yong Yu, and Jun Wang. Learning to select cuts for efficient mixed-integer programming. *Pattern Recognition*, 123:108353, 2022.

[8] Christian Igel. Smooth min-max monotonic networks. *arXiv preprint arXiv:2306.01147*, 2023.

[9] Alexander T Ihler, Natalia Flerova, Rina Dechter, and Lars Otten. Join-graph based cost-shifting schemes. *arXiv preprint arXiv:1210.4878*, 2012.

[10] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30, 2017.

[11] Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid Von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, et al. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*, 2020.

[12] Tahrima Rahman, Sara Rouhani, and Vibhav Gogate. Novel upper bounds for the constrained most probable explanation task. *Advances in Neural Information Processing Systems*, 34: 9613–9624, 2021.

[13] Sara Rouhani, Tahrima Rahman, and Vibhav Gogate. A novel approach for constrained optimization in graphical models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 11949–11960. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/8ab9bb97ce35080338be74dc6375e0ed-Paper.pdf`.

[14] Nicholas Ruozzi. The bethe partition function of log-supermodular graphical models. *Advances in Neural Information Processing Systems*, 25, 2012.

[15] Victor Garcia Satorras and Max Welling. Neural enhanced belief propagation on factor graphs. In *International Conference on Artificial Intelligence and Statistics*, pages 685–693. PMLR, 2021.

[16] Solomon Eyal Shimony. Finding maps for belief networks is np-hard. *Artificial intelligence*, 68 (2):399–410, 1994.

[17] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*. Springer, 2005.

[18] Martin J Wainwright, Tommi S Jaakkola, and Alan S Willsky. Map estimation via agreement on trees: message-passing and linear programming. *IEEE transactions on information theory*, 51(11):3697–3717, 2005.

[19] KiJung Yoon, Renjie Liao, Yuwen Xiong, Lisa Zhang, Ethan Fetaya, Raquel Urtasun, Richard Zemel, and Xaq Pitkow. Inference in probabilistic graphical models by graph neural networks. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 868–875. IEEE, 2019.

[20] Zhen Zhang, Fan Wu, and Wee Sun Lee. Factor graph neural networks. *Advances in Neural Information Processing Systems*, 33:8577–8587, 2020.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 (Contributions) state the four claims — deterministic validity by construction, self-supervised training without LP labels, multi-constraint CMPE extension, and empirical gains on UAI benchmarks — all of which are supported by Section 4 and the appendix.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed inline: degenerate i-bound cases where MBE already attains the LP optimum (Sec. 4.1, Sec. 9.15), per-graph training requirement (Sec. 6), out-of-distribution μ values for CMPE (Sec. 4.2), ordinal bias of scalar evidence encoding (Sec. 9.17), and tightness only when the join-graph LP relaxation is itself tight (Sec. 3, end).

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.

- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The single theoretical claim (Proposition 7: $D_\tau \geq D \geq \mathrm{MAP}^*$) is stated in Sec. 3 with a one-line proof from convexity of LSE; the antisymmetric construction guaranteeing validity is given in Eqs. (4)–(6).

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Architecture (3-layer MLP, 512/512/256), optimizer (AdamW, cosine schedule, $lr = 10^{-3}$, 200 epochs, batch 64), τ schedule, evidence sampling, training data construction, replay-buffer settings, and JGLP iteration budgets are fully specified in Sec. 4 and Sec. 9.8–9.16.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same

dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Anonymized code is provided in supplementary material; UAI 2022 benchmark PGMs are publicly available from the UAI competition website; CVXPY/CLARABEL is open-source.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

Justification: Sec. 4 specifies i-bound 4, 10 seeds per (PGM, evidence-bin), evidence ratios, μ-grid, σ for CMPE; Sec. 9.8–9.10 give CMPE-specific settings; Sec. 9.16 reports an 11-variant ablation suite.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: All headline tables report mean $\pm$ population standard deviation over 10 bin-0 test seeds per PGM (MAP) or 10 seeds $\times$ 100 μ-grad steps (CMPE); Sec. 9.18 reports seed-subset stability ($\pm$4pp).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Wall-clock timings reported in Tables and neural forward pass ~ 0.04 ms; JGLP iteration costs 1.6–131 ms depending on PGM. Hardware: single NVIDIA GPU; CVXPY/CLARABEL on CPU.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work conforms with the NeurIPS Code of Ethics; it uses only public benchmark PGMs and involves no human subjects, scraped data, or deployed systems.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This is foundational work on dual-bound computation for graphical-model inference. There is no direct path to deployment in a setting with foreseeable positive or negative societal impact distinct from PGM inference research generally.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The released artifact is a per-PGM neural warm-start network for dual bounds; it has no generative or dual-use risk.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: UAI 2022 benchmark PGMs are cited and used under the competition's public-release terms; CVXPY (Apache 2.0) and CLARABEL (Apache 2.0) are cited; PyTorch (BSD-3) is the training framework.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Trained network checkpoints, training scripts, and evaluation harness are released in the supplementary material with a README and license.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: No human subjects or crowdsourcing.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: No human subjects research.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were used only for writing, editing, and formatting; they do not form a component of the core methodology.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.

Appendix roadmap The appendix covers the benchmark suite and per-model sizes (App. B.10) with contingency and timing breakdowns (App. B.11); the architecture, notation and a worked example (Apps. A, A.1, A.2) and the LP formulation of the dual (App. B.1); CMPE method descriptions (App. B), Lagrangian bound quality (App. B.3), wall-clock timing (App. B.6) and the method-family split (App. B.21); standalone bound quality across all evidence bins (Apps. B.4, B.7, B.16), the v1 metric (Apps. B.18, B.18), AOBB root bounds (App. B.8), monotonicity (App. B.9), the validity check (App. B.22) and the numerical floor (App. B.29); paired per-seed statistics (Apps. B.24, B.25); TTA on unconstrained MAP (App. B.19), an LP-free CMPE pilot (App. B.20) and the LP-supervised alternative (App. B.28); an 11-variant ablation suite (App. B.13) with the evidence-encoding (App. B.14), gated-initialization (App. B.26), architecture and training-set-size (App. B.27) and seed-subset (App. B.15) studies; two negative results (Apps. B.23, B.30); and an adversarial CMPE pilot on image classifiers (App. B.12).

A Edge-message Architecture

Figure 2 sketches the parameterization.

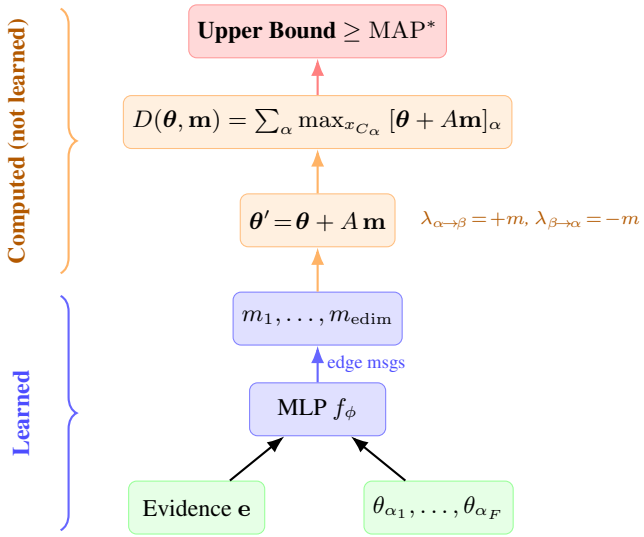


Figure 2: **Edge-message architecture.** A learned MLP f_ϕ (blue) maps evidence $\mathbf{e}$ and factor tables $\{\theta_\alpha\}$ to edge messages $\mathbf{m} \in \mathbb{R}^{\text{edim}}$, one per (edge, separator-assignment) pair on the join graph. The fixed antisymmetric scatter matrix A (orange) maps $\mathbf{m}$ into cluster-table modifications: each column of A adds $+m$ to one cluster and $-m$ to the neighboring cluster on a shared separator, so the two contributions cancel for every complete assignment. This restricts predictions to the antisymmetric subspace where reparameterization is valid; consequently $D(\boldsymbol{\theta}, \mathbf{m}) \geq \text{MAP}^*$ holds for *any* network weights (Proposition 1), and training never has to enforce feasibility.

A.1 Notation

Table 4 collects the symbols used in the main body.

A.2 A Worked Three-Cluster Example

Table 5 works the example of Section 2 end to end, with the per-cluster maxima before and after the shift. Every entry is machine-computed by the implementation we release. In code the shift is the single product $\boldsymbol{\theta}' = \boldsymbol{\theta} + A\mathbf{m}$ with $A \in \{-1, 0, +1\}^{12\times 4}$: each column carries -1 on one cluster's entries consistent with a separator value and $+1$ on its neighbour's, and a complete assignment selects exactly one entry per cluster, so the pair cancels. Sampling $\mathbf{m} \sim \mathcal{N}(0, 10^2)$ gives 0 violations in 2,000 draws, none of them tighter than the exact 17 above.

Table 4: Notation used throughout. Dimensions are for the join graph induced by mini-bucket elimination at the chosen i-bound.

Symbol	Dimension	Meaning
$\mathcal{C}, \mathcal{E}$	—	clusters and edges of the join graph
$S_{\alpha\beta}$	—	separator $C_\alpha \cap C_\beta$ of edge (α, β)
θ_{C_α}	$\prod_{i \in C_\alpha} d_i$	log-potential table of cluster α
θ_{flat}	λ_{dim}	all cluster tables concatenated into one vector
λ_{dim}	scalar	$\sum_\alpha \prod_{i \in C_\alpha} d_i$, total number of cluster-table entries
$\lambda_{\alpha\beta}$	$\prod_{i \in S_{\alpha\beta}} d_i$	directed edge message, constrained by $\lambda_{\alpha\beta} + \lambda_{\beta\alpha} = 0$
$\mathbf{m}$	edim	the network's output: one real per (undirected edge, separator assignment)
edim	scalar	$\sum_{(\alpha,\beta)\in\mathcal{E}} \prod_{i \in S_{\alpha\beta}} d_i$; typically edim $\ll \lambda_{\text{dim}}$
A	$\lambda_{\text{dim}} \times$ edim	fixed $\{-1, 0, +1\}$ scatter map; $\theta_{\text{flat}} + A\mathbf{m}$ is a reparameterization
D_0	scalar	$\sum_\alpha \max_x \theta_{C_\alpha}$, the MBE bound at $\mathbf{m} = \mathbf{0}$ (after MBE-MM)
$D(\boldsymbol{\theta}, \mathbf{m})$	scalar	$\sum_\alpha \max_x [\theta_{\text{flat}} + A\mathbf{m}]_\alpha \geq \text{MAP}^*$
D_τ	scalar	log-sum-exp surrogate for D at temperature τ; $D_\tau \geq D$
D^{LP}	scalar	$\min_{\mathbf{m}} D(\boldsymbol{\theta}, \mathbf{m})$, the LP optimum for this join graph
$\boldsymbol{\mu}$	K	Lagrange multipliers for the K CMPE constraints

Table 5: **A worked three-cluster example.** Four binary variables $x_0 \ldots x_3$; clusters $C_1(x_0, x_1)$, $C_2(x_1, x_2)$, $C_3(x_2, x_3)$ chained by separators $\{x_1\}$ and $\{x_2\}$. Here $\lambda_{\text{dim}} = 12$, edim $= 4$, and A is 12×4. The true optimum is $\text{MAP}^* = 17$ at $\mathbf{x} = (1, 1, 1, 1)$, while the unshifted bound is $D_0 = 7 + 8 + 4 = 19$: the clusters disagree about x_1 and x_2, and that disagreement is exactly the gap of 2. With edge messages $\mathbf{m} = (-3, -2, -2, -3)$ — one value per (edge, separator assignment) — every cluster maximum moves and the bound becomes exact. The reparameterized tables are $\theta'_{C_1}(x_0, x_1) = \theta_{C_1} - m_{01}(x_1)$, $\theta'_{C_2}(x_1, x_2) = \theta_{C_2} + m_{01}(x_1) - m_{12}(x_2)$, and $\theta'_{C_3}(x_2, x_3) = \theta_{C_3} + m_{12}(x_2)$: what one cluster gives up on a separator, its neighbour gains, so the score of *every* complete assignment is unchanged (verified for all 16) while the sum of maxima drops from 19 to 17.

	original θ				**reparameterized** $\theta' = \theta + A\mathbf{m}$		
$C_1(x_0, x_1)$	x_1=0	x_1=1	max	C'_1	x_1=0	x_1=1	max
x_0=0	2	5		x_0=0	5	7	
x_0=1	1	**7**	7	x_0=1	4	**9**	9
$C_2(x_1, x_2)$	x_2=0	x_2=1	max	C'_2	x_2=0	x_2=1	max
x_1=0	**8**	1	8	x_1=0	**7**	1	
x_1=1	4	6		x_1=1	4	**7**	7
$C_3(x_2, x_3)$	x_3=0	x_3=1	max	C'_3	x_3=0	x_3=1	max
x_2=0	3	0		x_2=0	1	-2	
x_2=1	2	**4**	4	x_2=1	-1	**1**	1
$D_0 = 7 + 8 + 4 = 19$				$D(\boldsymbol{\theta}, \mathbf{m}) = 9 + 7 + 1 = 17 = \text{MAP}^*$			

B CMPE Method Descriptions

All four methods share the same outer loop: a μ-gradient descent trajectory $\mu_0, \mu_1, \ldots, \mu_T$ that optimizes the Lagrangian $L(\mu)$ for the constrained MAP (CMPE) problem. They differ only in how JGLP is initialized at each μ-step.

Key differences. JGLP and NN are *memoryless*: each μ-step starts fresh (from zero or from a single NN forward pass). NN+TTA carries state via the NN weights θ' and the replay buffer $\mathcal{R}$, but discards JGLP-converged λ after each step. JGLP+WS carries state via the converged cluster shift $(\Delta\lambda, \Delta z)$, but uses no neural network. These represent two orthogonal adaptation channels: learned prediction (NN+TTA) versus direct λ-continuity (JGLP+WS). On Bayesian networks where μ converges quickly, λ-continuity dominates (JGLP+WS $<$ NN+TTA). On grids and segmentation

Algorithm 1 JGLP (zero-init baseline)

Require: μ-trajectory $\{\mu_k\}_{k=0}^{T}$, iteration budget I, join-graph parameters
1: **for** $k = 0, 1, \ldots, T$ **do**
2: Build join graph $\mathcal{J}_k$ from combined potentials at μ_k
3: $\lambda_k^{(0)} \leftarrow \mathbf{0}$ {zero initialization}
4: Run JGLP for I iterations from $\lambda_k^{(0)} \to \lambda_k^*$
5: Record dual bound $D(\lambda_k^*)$ at checkpoints
6: *Discard* λ_k^*
7: **end for**

Algorithm 2 NN (neural warm-start, frozen)

Require: μ-trajectory $\{\mu_k\}_{k=0}^{T}$, iteration budget I, pre-trained network f_θ
1: **for** $k = 0, 1, \ldots, T$ **do**
2: Build join graph $\mathcal{J}_k$ from combined potentials at μ_k
3: Extract base cluster tables λ_k^{base} from $\mathcal{J}_k$
4: $\mathbf{m}_k \leftarrow f_\theta(\mathbf{x}_k, \mu_k)$ {NN predicts edge messages; weights θ frozen}
5: $\lambda_k^{(0)} \leftarrow \lambda_k^{\text{base}} + A\mathbf{m}_k$ {inject via antisymmetric reparam matrix A}
6: Run JGLP for I iterations from $\lambda_k^{(0)} \to \lambda_k^*$
7: Record dual bound $D(\lambda_k^*)$ at checkpoints
8: *Discard* λ_k^*; *do not update* θ
9: **end for**

Algorithm 3 NN+TTA (neural warm-start with test-time adaptation; *ours*)

Require: μ-trajectory $\{\mu_k\}_{k=0}^{T}$, iteration budget I, pre-trained network f_θ, TTA learning rate η, replay buffer size B, inner TTA steps S
1: $\theta' \leftarrow \theta$ {copy weights for adaptation}
2: $\mathcal{R} \leftarrow \emptyset$ {replay buffer}
3: **for** $k = 0, 1, \ldots, T$ **do**
4: Build join graph $\mathcal{J}_k$ from combined potentials at μ_k
5: Extract base cluster tables λ_k^{base} and $\text{logz}_k^{\text{base}}$ from $\mathcal{J}_k$
6: $\mathbf{m}_k \leftarrow f_{\theta'}(\mathbf{x}_k, \mu_k)$ {NN predicts edge messages with adapted weights}
7: $\lambda_k^{(0)} \leftarrow \lambda_k^{\text{base}} + A\mathbf{m}_k$ {inject NN prediction}
8: Run JGLP for I iterations from $\lambda_k^{(0)} \to \lambda_k^*$
9: Record dual bound $D(\lambda_k^*)$ at checkpoints
10: $\mathcal{R} \leftarrow \mathcal{R} \cup \{(\mathbf{x}_k, \mu_k, \lambda_k^{\text{base}}, \text{logz}_k^{\text{base}})\}$; trim to most recent B entries
11: **for** $s = 1, \ldots, S$ **do**
12: {TTA: fine-tune NN on recent μ-region}
13: $\mathcal{L} \leftarrow \frac{1}{|\mathcal{R}|} \sum_{(\mathbf{x},\mu,\lambda,z)\in\mathcal{R}} D_\tau\big(f_{\theta'}(\mathbf{x}, \mu),\ \lambda,\ z\big)$
14: $\theta' \leftarrow \theta' - \eta \nabla_{\theta'} \mathcal{L}$ {only NN weights update}
15: **end for**
16: *Discard* λ_k^* {JGLP-converged λ not carried to step $k{+}1$}
17: **end for**

Algorithm 4 JGLP+WS (warm-start with previous μ-step's converged shift; fair-compute baseline)

Require: μ-trajectory $\{\mu_k\}_{k=0}^{T}$, iteration budget I
$\Delta\lambda \leftarrow \mathbf{0}, \quad \Delta z \leftarrow 0$ {no previous shift at step 0}
for $k = 0, 1, \ldots, T$ **do**
  Build join graph $\mathcal{J}_k$ from combined potentials at μ_k
  Extract initial tables λ_k^{init} and z_k^{init} from $\mathcal{J}_k$
  if $k > 0$ **then**
    $\lambda_k^{(0)} \leftarrow \lambda_k^{\text{init}} + \Delta\lambda$ {apply previous JGLP shift}
    $z_k^{(0)} \leftarrow z_k^{\text{init}} + \Delta z$
  else
    $\lambda_k^{(0)} \leftarrow \lambda_k^{\text{init}}$ {step 0: equivalent to JGLP}
  end if
  Run JGLP for I iterations from $\lambda_k^{(0)} \rightarrow \lambda_k^*, \quad z_k^*$
  Record dual bound $D(\lambda_k^*)$ at checkpoints
  $\Delta\lambda \leftarrow \lambda_k^* - \lambda_k^{\text{init}}$ {save what JGLP learned this step}
  $\Delta z \leftarrow z_k^* - z_k^{\text{init}}$
end for

models where μ keeps moving, stale shifts destabilize JGLP and the neural structural prior proves essential (NN+TTA $\ll$ JGLP+WS).

B.1 LP Formulation of the JGLP Dual

Let $\boldsymbol{\theta}$ be the flat concatenation of cluster log-tables after mini-bucket decomposition (length λ_{dim}) and let $A \in \mathbb{R}^{\lambda_{\text{dim}} \times e_{\text{dim}}}$ be the antisymmetric scatter matrix mapping edge messages $\mathbf{m} \in \mathbb{R}^{e_{\text{dim}}}$ onto cluster entries. The JGLP dual objective is $D(\boldsymbol{\theta}, \mathbf{m}) = \sum_c \max_{x_c} [\boldsymbol{\theta}_{c,x_c} + (A\mathbf{m})_{c,x_c}]$. The LP relaxation (epigraph form) is

$$\min_{\mathbf{m},\mathbf{t}} \sum_c t_c \quad \text{s.t. } t_c \geq \boldsymbol{\theta}_{c,x_c} + (A\mathbf{m})_{c,x_c} \quad \forall c, x_c,$$

with $D_{\text{LP}} = \min \sum_c t_c$. We solve this with CVXPY's CLARABEL backend (fallback: ECOS, SCS). On grid10×10 at i-bound 4 ($\lambda_{\text{dim}} = 3{,}854$, $e_{\text{dim}} = 1{,}904$), CLARABEL returns the LP optimum in 2.9s; JGLP at 500 iterations converges to a value 8.6×10^{-4} above the LP optimum, demonstrating that JGLP's coordinate-descent can stop short of global optimality.

B.2 Full MAP Results (All 37 PGMs)

Figure 1 repeats the main-body convergence plot for reference. Table 6 lists all 37 evaluated PGM *rows* in the new $\log_{10} |D - D_{\text{LP}}|$ format, sorted by neural win rate at iter=10. Promedas models exhibit small dynamic range (log-error ~ 0) because their dual objectives are near-zero, which compresses the relative scale.

Duplicate UAI files across names. The UAI collection ships four of these graphs twice, under both a `Grids_NN` name and a `grid`R×C`.fK` name. The file pairs are byte- or token-identical, so **these 37 rows are 33 distinct models**: `Grids_29` = `grid20×20.f5.wrap`, `Grids_30` = `grid20×20.f10`, `Grids_18` = `grid40×40.f10.wrap`, and `Grids_25` = `grid40×40.f2`. We keep both rows of each pair because each carries an independently trained network, but no count in this paper treats them as separate models. The same four collisions make the 40-row benchmark suite of Table 18 36 distinct models.

Per-row evaluation setting. The main suite uses i-bound 4 with a 20% evidence fraction throughout. Fourteen of the appendix-only rows do not, and we list them rather than let the table imply a single protocol: the ten Promedas rows use a 40% evidence fraction (and `Promedas_51` i-bound 3), and the four largest grids use a reduced i-bound with 50% evidence — i-bound 3 for `grid40×40.f2` and i-bound 2 for `grid40×40.f10.wrap`, `grid80×80.f10` and `grid80×80.f5.wrap` — because the join graph at i-bound 4 is not tractable for them at 20% evidence. This matters for reading the table:

a weaker relaxation on a more heavily conditioned problem is easier for MBE to be LP-optimal on, which is why those four rows show near-zero gaps while their standard-protocol twins (`Grids_18`, `Grids_25`) show gaps of 14.3 and 4.94 nats on the identical files. Rows evaluated under different settings should not be compared with each other.

Table 6: Full MAP bound results: $\log_{10}|D - D_{\mathrm{LP}}|$ at fixed JGLP iterations on 37 PGMs. BN_126 column blanked due to LP ill-conditioning (CLARABEL returned status 'optimal_inaccurate').

		iter=0		iter=10		iter=50		iter=100		conv		
PGM	D_{LP}	J	N	J	N	J	N	J	N	J	N	win@10
BN_12	-31.64	-0.02	-0.17	-0.22	-0.48	-0.74	-1.05	-1.65	-2.04	-6.79	-6.77	100%
BN_13	-44.93	-0.04	-0.19	-0.29	-0.60	-0.97	-1.15	-1.34	-1.55	-7.17	-6.98	100%
BN_14	-38.83	-0.34	-0.39	-0.66	-0.83	-1.23	-1.33	-4.22	-4.52	-6.49	-6.42	100%
Seg_12	-114.24	0.21	-0.05	-0.01	-1.01	-4.13	-4.94	-7.10	-6.69	-7.10	-6.69	100%
Seg_14	-136.64	0.18	-0.42	-0.73	-2.00	-6.34	-4.62	-6.34	-6.13	-6.34	-6.19	100%
Seg_16	-132.72	0.58	0.13	0.29	-0.49	-0.67	-3.88	-6.07	-5.71	-6.23	-6.10	100%
PF_11	4145	1.90	1.70	1.34	1.29	0.85	0.84	0.55	0.51	-0.40	-0.40	100%
Grids_18	11284	1.84	1.84	1.14	1.13	0.35	0.33	0.05	0.04	-2.04	-2.01	100%
Grids_19	2327	1.02	1.01	0.23	0.13	-1.87	-1.86	-6.19	-4.34	-6.41	-6.23	100%
Grids_21	16907	2.02	2.00	1.31	1.30	0.51	0.50	0.23	0.23	-1.73	-1.74	100%
Grids_23	5618	1.51	1.52	1.16	1.14	0.37	0.38	-0.06	-0.04	-2.11	-2.15	100%
Grids_24	16740	1.99	1.99	1.68	1.67	1.00	1.00	0.56	0.56	-0.89	-0.93	100%
Grids_25	2332	1.15	1.10	0.70	0.53	-0.26	-0.46	-0.99	-1.17	-4.90	-4.91	100%
Grids_30	2767	1.71	1.50	1.17	1.11	0.66	0.62	0.42	0.38	-0.41	-0.40	100%
grid20×20.f10	2767	1.71	1.51	1.17	1.11	0.66	0.62	0.42	0.38	-0.41	-0.40	100%
Grids_22	5635	1.48	1.49	0.86	0.85	-0.14	-0.17	-0.88	-0.96	-3.92	-4.00	90%
Grids_27	11174	1.82	1.81	1.50	1.49	0.83	0.81	0.35	0.33	-0.98	-1.06	90%
grid20×20.f5	1322	0.89	0.87	0.07	0.03	-1.05	-1.05	-8.79	-6.00	-9.24	-6.82	90%
Grids_29	1322	0.89	0.87	0.07	0.03	-1.05	-1.05	-8.79	-5.60	-9.24	-6.78	80%
Promedas_51	-8.29	0.08	-0.22	0.08	-1.64	0.08	-2.80	0.08	-3.71	0.08	-6.19	80%
Seg_18	-111.99	-0.24	-0.06	-0.66	-0.87	-3.35	-3.79	-6.02	-6.15	-6.02	-6.15	80%
BN_9	-39.98	-0.59	-0.60	-0.97	-1.53	-2.71	-5.77	-6.68	-6.94	-6.68	-6.98	70%
Grids_26	2634	1.23	1.18	0.38	0.39	-1.12	-1.12	-8.46	-6.25	-8.46	-6.74	60%
Promedas_32	-5.61	0.54	0.54	0.54	-0.56	0.54	-2.53	0.54	-3.20	0.54	-6.01	60%
Promedas_33	-4.14	0.37	0.28	0.37	-0.72	0.37	-1.47	0.37	-1.85	0.37	-4.05	60%
Grids_28	3954	1.36	1.34	0.72	0.73	-0.90	-0.90	-8.56	-5.62	-8.56	-6.76	50%
Promedas_44	-3.82	-2.79	-0.20	-3.35	-1.59	-6.27	-2.64	-7.72	-3.23	-7.72	-6.23	50%
Promedas_40	-3.77	-5.69	-0.25	-5.69	-1.69	-7.62	-2.96	-7.62	-3.86	-7.62	-6.34	40%
Seg_19	-93.60	-0.41	-0.24	-3.42	-1.16	-5.56	-3.97	-5.71	-5.62	-6.25	-6.60	33%
Promedas_21	-7.69	-6.14	0.40	-6.14	-1.13	-6.14	-2.44	-6.14	-3.40	-6.14	-6.15	30%
grid80×80.f10	28287	0.65	0.73	-0.52	-0.64	-6.38	-4.60	-7.85	-6.99	-7.85	-7.01	10%
Promedas_38	-1.51	-7.54	-0.30	-7.54	-1.26	-7.54	-2.02	-7.54	-2.54	-7.54	-5.55	0%
grid10x10	335.89	-8.78	-0.83	-8.78	-3.04	-8.78	-7.35	-8.78	-7.35	-8.78	-7.35	0%
grid40×40.f10	7220	-4.68	-0.46	-9.03	-3.45	-9.03	-7.40	-9.03	-7.40	-9.03	-7.40	0%
grid40×40.f2	1478	-8.88	-0.44	-9.34	-3.33	-9.34	-7.06	-9.34	-7.36	-9.34	-7.36	0%
grid80×80.f5	14246	-2.78	0.05	-6.83	-3.00	-8.17	-4.98	-8.17	-6.26	-8.17	-6.77	0%

B.3 CMPE Lagrangian Bound Quality

Table 7 reports the Lagrangian bound at μ^* against the best MAP bound, and Table 8 compares it with CMPE_UB [12].

Table 7: CMPE bounds: MAP bounds ($\mu = 0$) vs Lagrangian relaxation at optimal μ^*. The CMPE Lagrangian exploits the constraint to yield substantially tighter upper bounds. Lower = tighter.

PGM	**MBE**	**JGLP**	**CMPE LR**	**Improvement**
BN_12 (90v)	-30.74	-31.45	-64.80	33.4
BN_9 (105v)	-39.20	-39.34	-79.82	40.5
BN_14 (115v)	-38.67	-39.11	-82.60	43.5
Seg_12	-117.40	-118.75	-434.34	316.2
Seg_14	-129.12	-130.76	-478.06	347.4
grid10×10	348.07	347.76	43.03	305.2

Table 8: Our Lagrangian relaxation (JGLP, i-bound 4) vs CMPE_UB [12] (i-bound 10). Our method produces tighter bounds in less time.

PGM	Our LR	CMPE_UB (JG)	Our time	UB time
BN_12	-64.48	-41.43	48.8s	112.6s
BN_9	-79.15	-54.59	30.1s	54.7s
BN_14	-80.83	-54.44	41.7s	88.4s

Table 9: Full standalone bound results across all PGMs and evidence bins. "% Better" = fraction of test instances where neural bound is tighter than MBE. Bin 2 typically has high evidence ratios where the problem approaches exact solutions.

PGM	Bin	$e_{\dim}$	% Better
BN_12	0	2,130	94%
	1	1,076	84%
	2	298	0%
BN_9	0	1,354	80%
	1	828	72%
	2	220	0%
BN_14	0	1,812	92%
	1	1,086	92%
	2	298	8%
grid10×10	0	808	68%
	1	102	0%
	2	20	46%
Seg_12	0	2,594	94%
	1	1,558	0%
	2	72	4%
Seg_14	0	3,046	100%
	1	1,896	12%
	2	88	0%
Seg_18	0	3,414	74%
	1	1,550	50%
	2	92	2%
grid20×20.f5	0	2,988	20%
	1	442	32%
	2	76	32%
grid20×20.f10	0	5,002	98%
	1	678	0%
	2	98	16%
grid40×40.f10	0	1,400	50%
grid40×40.f2	0	1,664	0%
grid80×80.f10	0	5,726	82%
grid80×80.f5	0	5,262	36%

B.4 Full Standalone Bound Results (All Bins)

B.5 CMPE Application Details

Scope of this subsection: the $\sigma = 0.1$ diagnostic regime. The results in this subsection were obtained with constraint noise $\sigma = 0.1$, an early diagnostic setting that we *do not* use for any reported result. At $\sigma = 0.1$ we have $\theta^c \approx \theta$, so the combined Lagrangian $\theta - \mu^* \theta^c$ at $\mu^* \approx 1$ collapses to pure noise: the constraint is effectively inactive and the combined model is out of distribution for any network trained on θ. Every headline CMPE result uses $\sigma = 0.5$, which keeps the constraint active at the optimum ($\mu^* > 0$). Conclusions drawn here therefore characterize the degenerate $\sigma = 0.1$ regime only and do not transfer to the $\sigma = 0.5$ setting used in the main body.

CMPE formulation. Given two log-linear models f and g over the same graph, the constrained MPE (CMPE) problem is $\max_{\mathbf{x}} f(\mathbf{x})$ s.t. $g(\mathbf{x}) \leq q$. The Lagrangian relaxation yields:

$$L(\mu) = \max_{\mathbf{x}}[f(\mathbf{x}) - \mu(g(\mathbf{x}) - q)] = D_\mu(\lambda) + \mu q, \quad \mu \geq 0 \tag{8}$$

where $D_\mu(\lambda)$ is the MAP dual bound for the combined model $M_\mu \equiv f - \mu g$. Since any feasible $\mathbf{x}^*$ satisfies $g(\mathbf{x}^*) \leq q$, we have $-\mu(g(\mathbf{x}^*) - q) \geq 0$ for $\mu \geq 0$, so $L(\mu) \geq f(\mathbf{x}^*) \geq \text{CMPE}^*$. Our framework computes $D_\mu(\lambda)$ for any μ by building the join graph for M_μ and applying JGLP.

μ search. We find $\mu^* = \arg\min_{\mu \geq 0} L(\mu)$ via golden-section search over $[0, 5]$ with 15 function evaluations. For our test instances ($g = f + \epsilon$, $\sigma = 0.1$), the optimal multiplier concentrates at $\mu^* \approx 0.97$ because the combined model $f - \mu g$ at $\mu = 1$ reduces to $-\epsilon$ (pure noise), making the bound very tight.

Gradient vs JGLP for inner solver. Table 10 shows that JGLP-200 consistently outperforms gradient descent (Adam, 200 iterations on D_τ with $\tau = 0.001$) for the inner MAP problem at each μ. JGLP's combinatorial message-passing directly tightens D while gradient descent optimizes the smooth surrogate D_τ, which can temporarily worsen D.

Table 10: Inner solver comparison for CMPE: JGLP-200 vs gradient descent (200 iterations) on the combined model M_{μ^*}. All methods use i-bound 4. Lower = tighter CMPE bound.

PGM	MBE	JGLP-200	Grad-200	N+Grad	CMPE LR
BN_12	-64.19	-64.46	-64.30	-64.30	-64.63
BN_9	-80.05	-80.21	-80.11	-80.11	-80.32
BN_14	-81.18	-81.51	-81.37	-81.28	-81.66

Neural μ prediction. We train an MLP to predict μ^* directly from the evidence encoding, but find that for $\sigma = 0.1$ instances, the combined model at $\mu^* \approx 1$ is near-zero-mean noise, which is out-of-distribution for edge-message models trained on the original f. Consequently, in this degenerate regime the neural warm-start of edge messages provides no advantage over zero-initialization. This is a statement about $\sigma = 0.1$ specifically, where the constraint is inactive and the combined model carries no signal; it is *not* a general finding about CMPE, and it does not hold at the $\sigma = 0.5$ setting used throughout the main body, where the neural warm-start does help.

Gap analysis. Table 11 shows the gap between upper bounds and the best feasible solution (lower bound) found by random search. The CMPE Lagrangian reduces the gap by 5–7$\times$ compared to MAP bounds and 4–5$\times$ compared to CMPE_UB, demonstrating that the Lagrangian relaxation effectively exploits the constraint structure.

Table 11: Gap from best feasible solution (lower bound). Smaller gap = tighter bound. CMPE LR provides dramatically tighter certificates than all MAP-based bounds.

PGM	MBE	JGLP	Neural	CMPE LR	CMPE_UB
BN_12	41.30	40.60	40.71	**6.64**	29.70
BN_9	46.84	46.54	46.62	**6.96**	32.80
BN_14	52.72	52.23	52.35	**8.02**	34.89

B.6 CMPE Wall-Clock Timing

Table 14 gives the full accounting behind the amortized-cost discussion in Section 4. Each cost component (join-graph build, one JGLP iteration, one forward pass, one replay-buffer TTA inner loop, one-off training) is measured directly on an unloaded machine, and combined with the iteration at which each individual run first reaches the target gap.

Three things about how this is measured are worth stating, because an earlier version of this analysis got all three wrong and reported a much more pessimistic break-even as a result. First, the crossing iteration is recorded *exactly*, by instrumenting JGLP at every iteration, rather than being rounded up

Table 12: CMPE gradient-descent wall-clock time (single constraint, 50 μ steps). JGLP runs 50 iterations per inner solve; NN uses a single forward pass. L^* is an *upper* bound on CMPE^*, so **lower is tighter**; ΔL^* is NN minus JGLP, and a positive value means the NN bound is that many nats *looser*. The speedup is therefore not a free win: on BN_14 and Seg_14 the single forward pass is $12.6\times$ and $8.8\times$ faster but returns a bound 42.0 and 317 nats looser than JGLP's, which is not a usable trade for those two models. This table reports throughput at whatever bound quality the forward pass happens to give; the bound-quality comparison at matched budgets is Tables 2–3. These runs use the $\sigma = 0.1$ diagnostic regime of App. B.5, not the $\sigma = 0.5$ setting of the main body.

	JGLP		NN			
PGM	L^*	time	L^*	time	ΔL^*	**Speedup**
BN_12	-62.0	22.8s	-54.2	1.8s	$+7.8$	$12.9\times$
BN_9	-80.6	17.3s	-78.3	1.5s	$+2.3$	$11.5\times$
BN_14	-80.0	21.7s	-38.0	1.7s	$+42.0$	$12.6\times$
Seg_12	-261	31.2s	-224	4.1s	$+37$	$7.7\times$
Seg_14	-458	37.3s	-141	4.3s	$+317$	$8.8\times$
grid10×10	339	9.6s	341	1.4s	$+2$	$6.7\times$

Table 13: Two-constraint CMPE wall-clock time (50 gradient steps on (μ_1, μ_2)). Conventions as in Table 12: L^* is an upper bound on CMPE^* (lower is tighter) and ΔL^* is NN minus JGLP, so positive means the NN bound is looser. Here the two arms agree to within a few nats on five of six models, and on BN_14 the NN bound is 19.1 nats *tighter*. Note that the L^* values are not comparable across Tables 12 and 13: adding a second constraint changes the constrained problem, so BN_14's JGLP optimum moves from -80.0 ($K = 1$) to -59.1 ($K = 2$). The $K = 2$ Lagrangian bound being *looser* than the $K = 1$ bound on BN_14, despite a strictly smaller feasible set, indicates that the (μ_1, μ_2) search has not reached the optimal multiplier pair on that model rather than any property of the relaxation. These runs use the $\sigma = 0.1$ diagnostic regime of App. B.5.

	JGLP		NN			
PGM	L^*	time	L^*	time	ΔL^*	**Speedup**
BN_12	-60.9	22.7s	-58.3	1.9s	$+2.6$	$11.9\times$
BN_9	-79.5	17.3s	-78.8	1.5s	$+0.7$	$11.6\times$
BN_14	-59.1	21.9s	-78.2	1.8s	-19.1	$12.4\times$
Seg_12	-122	31.7s	-121	4.1s	$+1$	$7.7\times$
Seg_14	-144	37.1s	-143	4.2s	$+1$	$8.8\times$
grid10×10	339	9.7s	341	1.4s	$+2$	$6.8\times$

to the next checkpoint in $\{0, 10, 50, 100, 500\}$ — at checkpoint granularity both arms frequently land in the same bucket, the iteration term cancels, and the comparison silently degenerates to a difference of fixed overheads. Second, every comparison is *paired*: all three arms solve the identical query, so the saving is computed per query and then aggregated, never as a difference of two marginal medians. Third, runs that never reach the target are censored and counted rather than dropped from the median; this matters because the arms censor at very different rates (on BN_12, 49.8% for zero-init against 30.9% for NN+TTA), so a median over survivors compares the baseline on a strictly easier subsample of queries.

Cost of adaptation versus replay batch size. Table 15 isolates how the cost of test-time adaptation scales with the replay batch size B, on one small and one large PGM. The cost is strongly sublinear in B: entering the inner loop at all is what is expensive, while quadrupling B from 5 to 20 adds only 50% on the small PGM and 19% on the large one. A practitioner choosing B is therefore trading very little wall-clock for a larger replay buffer. Whether the cost is repaid at all is a separate question, and the lower panel answers it negatively in most cells: adaptation buys back its cost only on the large PGM at the loosest target (10^{-1}), and at tight targets the adapted run reaches the target in *more* iterations than zero-init, so there is no budget to spend. This is the same boundary the rest of the paper reports — the benefit of the learned initialization is concentrated early, and both arms descend the same convex objective to the same optimum.

Table 14: **Does the amortization pay for itself? (K=1, target 10^{-2}.)** Iteration counts are the *exact* iteration at which the gap to the LP optimum first falls below the target, not a checkpoint bucket, and every comparison is paired: all three arms solve the identical query (same seed, same μ-step, same D^{LP}). Runs that never reach the target are counted in the censoring columns rather than dropped from the medians, which matters because the arms censor at very different rates. "win%" is the fraction of paired queries on which the arm reaches the target in strictly fewer iterations than zero-init. Break-even is the one-off training cost divided by the paired per-query saving; ∞ means the arm has no per-query saving at this target, so training is never recovered. Rows above the rule are the main suite. Only 5 of these 30 (model, arm) cells have a finite break-even, and the four Bayesian networks are the only ones that reach it at a practical query count; we report this rather than claim wall-clock dominance. The per-iteration re-run was done for K=1; K=2 is covered by the bound-quality tables.

		M1	median iters		paired win%		censored (%)		break-even (queries)	
PGM	n	iters	M2	M3	M2	M3	M1	M3	M2	M3
BN_12	120	201	206	136	66	31	60	43	15,278	∞
BN_13	120	69	69	18	66	1	2	2	26,968	∞
BN_14	120	109	41	50	98	43	1	0	3,215	∞
BN_9	120	49	26	8	58	0	0	0	19,805	∞
Grids_26	120	52	76	40	0	10	0	0	∞	∞
Grids_28	120	60	76	45	0	0	0	0	∞	∞
Grids_29	120	57	66	39	50	0	0	0	∞	∞
Seg_12	120	33	28	17	40	0	0	0	∞	∞
Seg_14	120	20	39	24	28	0	0	0	∞	∞
Seg_16	120	36	41	27	28	0	0	0	∞	∞
Seg_18	120	77	64	65	48	2	0	0	1.5×10^{6}	∞
Seg_19	120	42	52	33	27	0	0	0	∞	∞
grid10×10.f10	120	0	44	16	17	5	0	0	∞	∞
grid10×10.f5	120	10	22	8	33	0	0	0	∞	∞
grid20×20.f5	120	57	62	40	50	0	0	0	∞	∞
Promedas_38	120	0	16	8	0	0	0	0	∞	∞
Promedas_44	120	0	16	9	0	0	0	0	∞	∞
Promedas_51	120	0	8	6	0	0	0	0	∞	∞
grid40×40.f10	120	0	24	10	0	0	0	0	∞	∞
grid40×40.f2	120	0	24	4	0	0	0	0	∞	∞

B.7 Standalone Bound Quality

For completeness, we also measure the fraction of test samples where the neural standalone bound is strictly tighter than the MBE (zero-init) bound. Table 16 reports this on the original 13 PGMs at bin 0 (10–30% evidence); Figure 3 visualizes the same data.

B.8 AOBB Root Bound Evaluation

As an additional application, we evaluate neural bounds as root-node upper bounds for AND/OR Branch-and-Bound (AOBB) search. A tighter root bound provides a better initial optimality certificate. Table 17 shows neural root bounds are tighter than MBE in 55–100% of queries across 11 PGMs. Full search integration (per-node heuristics throughout AOBB) requires per-subproblem neural bounds, which we leave to future work.

B.9 Empirical Monotonicity

Since the neural network predicts edge *messages* $\mathbf{m}$ (not bound values directly), there is no architectural reason to expect monotonicity in observed evidence. We verify this empirically: adding one observed variable at a time while tracking the predicted bound, the standard MLP exhibits a 54% average violation rate across 13 PGMs—indistinguishable from random. Enforcing monotonicity architecturally (e.g., via Smooth Min-Max Networks [8]) is a promising direction for future work when predicting bound *values* directly rather than messages.

Table 15: **Cost of replay-buffer test-time adaptation as a function of the replay batch size B.** *Top*: median wall-clock of one TTA step (inner loop of 10 steps), interquartile range in brackets, over 15 interleaved rounds — one timing per B per round, so machine-load drift cannot order the arms. B=0 is the no-adaptation baseline (forward pass only), which is why it matches the forward-pass latency rather than scaling with B. Cost is strongly sublinear in B: almost all of it is the fixed cost of entering the inner loop, and quadrupling B from 5 to 20 adds 50% on the small PGM and 19% on the large one. *Bottom*: whether that cost is repaid. "Iterations saved" is M1 minus M3 iterations to first reach the target; the budget is those iterations priced at the measured per-iteration cost; "largest affordable B" is the largest tested B fitting in it, and the break-even B extrapolates where cost meets budget. Adaptation repays its cost only on the large PGM at the loosest target; at tight targets the adapted run needs *more* iterations than zero-init (negative savings), consistent with both arms descending the same objective to the same optimum.

(a) Cost of one TTA step

PGM	size	B=0 (ms)	B=5 (ms)	B=10 (ms)	B=20 (ms)
grid10×10.f5	small	0.29	63 [29]	81 [15]	95 [28]
Seg_18	large	0.35	232 [49]	267 [52]	276 [50]

(b) Is that cost repaid?

PGM	target	iters M1	iters M3	saved	budget (ms)	largest B
grid10×10.f5	10^{-1}	6.5	2.0	+4.5	+20	0*
grid10×10.f5	10^{-2}	10.0	8.0	+2.0	+9	0*
grid10×10.f5	10^{-3}	11.0	20.0	-9.0	-40	–
grid10×10.f5	10^{-4}	11.0	33.0	-22.0	-97	–
grid10×10.f5	10^{-6}	11.0	56.5	-45.5	-201	–
Seg_18	10^{-1}	54.0	30.5	+23.5	+413	20
Seg_18	10^{-2}	77.0	65.0	+12.0	+211	0*
Seg_18	10^{-3}	86.0	80.0	+6.0	+106	0*
Seg_18	10^{-4}	76.0	88.0	-12.0	-211	–
Seg_18	10^{-6}	73.0	111.0	-38.0	-668	–

* budget is positive but smaller than the cheapest adapting configuration (B=5), so only the no-adaptation baseline fits; "–" marks a negative budget.

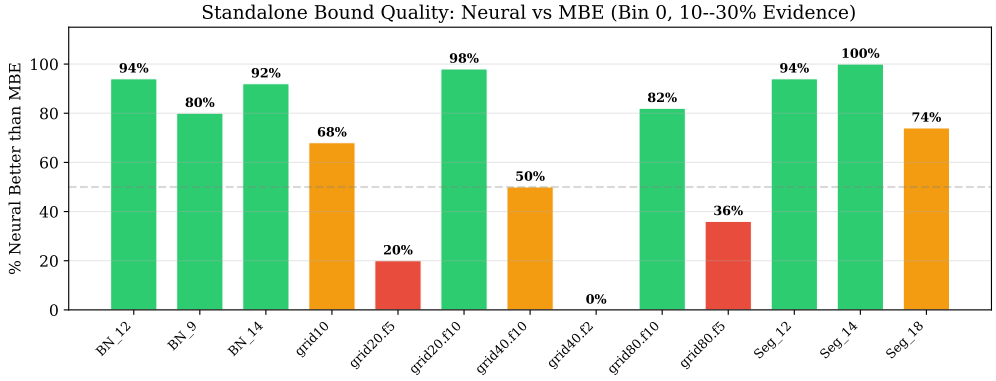


Figure 3: % neural-better than MBE across 13 PGMs at bin 0. Green: ≥80%, orange: 50–80%, red: <50%.

B.10 Full Benchmark Suite

Table 18 lists all 40 PGMs in our benchmark suite with their structural properties.

B.11 Contingency and Timing

Figure 4 counts PGMs where method i achieves strictly lower log-error than method j at iter=100 (ties within 10^{-3} excluded). On MAP (37 PGMs), neural-init wins on the majority; on CMPE K=1

Table 16: Standalone bound quality at bin 0. "% Better" = fraction of test instances where the neural bound is strictly tighter than MBE. All bounds are valid by construction (Proposition 1). "Fwd Time" is the median wall-clock of a *single-query* forward pass on CPU (200 timed calls after 20 warm-up), which is the quantity a caller actually pays; it ranges from 0.23 to 0.70 ms and grows with the input and edge-message dimensions. Earlier drafts of this table quoted 0.03–0.05 ms, which was batched-GPU throughput and roughly an order of magnitude optimistic. This view complements the iterations-to-ϵ analysis in Table 1.

PGM	Vars	e_{dim}	% Neural Better	Fwd Time
BN_12	90	2,130	94%	0.26ms
BN_9	105	1,354	80%	0.24ms
BN_14	68	1,812	92%	0.25ms
grid10×10	100	808	68%	0.23ms
grid20×20.f5	400	2,988	20%	0.28ms
grid20×20.f10	400	5,002	98%	0.32ms
grid40×40.f10	1,600	1,400	50%	0.48ms
grid40×40.f2	1,600	1,664	0%	0.34ms
grid80×80.f10	6,400	5,726	82%	0.70ms
grid80×80.f5	6,400	5,262	36%	0.68ms
Seg_12	154	2,594	94%	0.37ms
Seg_14	230	3,046	100%	0.39ms
Seg_18	232	3,414	74%	0.34ms

Table 17: Root bound comparison: Neural vs MBE upper bounds at the AOBB root node.

PGM	% Neural Tighter	Mean Δ	Neural Time
BN_12	100%	0.66	0.35ms
BN_9	85%	0.18	0.37ms
BN_14	100%	0.46	0.41ms
Seg_12	100%	2.24	0.47ms
Seg_14	100%	1.62	0.48ms
Seg_18	70%	0.60	0.49ms
grid20×20.f10	100%	34.05	0.51ms

(16 PGMs), M3 (NN + replay) wins over M1 (zero-init) and M2 (NN offline) on most PGMs; K=2 shows a similar pattern.

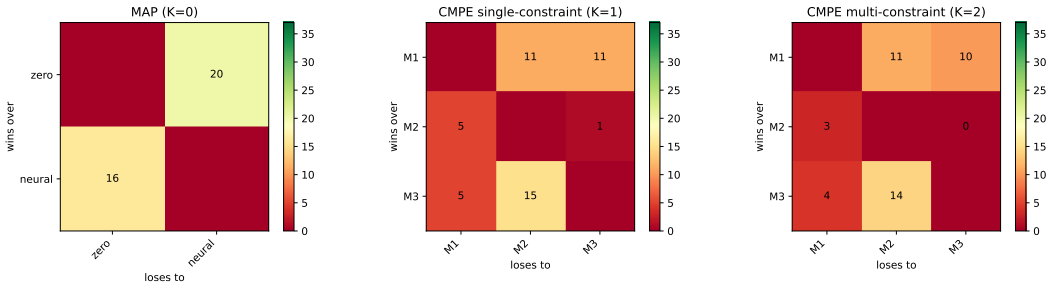


Figure 4: Pairwise contingency across 37 MAP / 16 CMPE K=1 / 14 CMPE K=2 PGMs at iter=100. Cell (i, j) is the count of PGMs where method i's log-error is strictly lower (closer to D_{LP}) than method j's. Diagonals are zero by construction.

Figure 5 shows the per-PGM inference-time disparity. Neural forward pass is ~ 0.04ms independent of PGM size; JGLP time scales with λ_{dim} and convergence-iteration count.

Table 18: Full benchmark suite: 40 PGMs spanning 7 categories. n = variables, e_{dim} = edge-message dimension, λ_{dim} = cluster-table dimension, i-bound = 4 except for the four largest grids and `Promedas_51`, whose per-row settings are listed in App. B.2. † = new PGMs beyond the original 13. Four graphs appear twice under two UAI names (`Grids_29`/`grid20×20.f5.wrap`, `Grids_30`/`grid20×20.f10`, `Grids_18`/`grid40×40.f10.wrap`, `Grids_25`/`grid40×40.f2`), so these 40 rows are 36 distinct models.

PGM	**Category**	n	e_{dim}	λ_{dim}	**MBE–JGLP gap**
BN_9	BN	105	1,354	3,122	0.26
BN_12	BN	90	2,130	4,346	0.71
BN_13†	BN	125	2,130	4,574	0.56
BN_14	BN	115	1,812	3,974	0.78
BN_126†	BN	512	5,268	11,414	0.23
Seg_12	Segmentation	231	3,998	8,460	0.40
Seg_14	Segmentation	231	3,958	8,388	2.30
Seg_16†	Segmentation	228	4,236	8,764	4.00
Seg_18	Segmentation	235	3,902	8,606	1.50
Seg_19†	Segmentation	228	4,132	8,712	0.90
PF_11†	ProteinFolding	400	6,366	13,798	73.4
grid10×10.f5	Grid	100	808	1,982	0.83
grid20×20.f5	Grid	400	2,988	7,312	11.7
grid20×20.f10	Grid	400	5,002	10,958	44.4
Grids_25†	Grid	1,600	28,422	59,368	63.8
Grids_26†	Grid	400	7,138	14,858	86.8
Grids_28†	Grid	400	7,138	14,858	132.6
Grids_29†	Grid	400	7,138	14,858	41.4
Grids_30†	Grid	400	6,366	13,798	49.3
Grids_18†	Grid	1,600	27,964	58,454	240.6
Grids_19†	Grid	1,600	27,964	58,454	51.0
Grids_21†	Grid	1,600	27,964	58,454	369.5
Grids_22†	Grid	1,600	27,964	58,454	136.2
Grids_23†	Grid	1,600	28,422	59,368	160.0
Grids_24†	Grid	1,600	28,422	59,368	479.1
Grids_27†	Grid	1,600	28,422	59,368	317.6
grid40×40.f10	Grid	1,600	12,568	30,556	64.2
grid40×40.f2	Grid	1,600	14,996	35,382	12.5
grid80×80.f10	Grid	6,400	54,558	130,372	318.1
grid80×80.f5	Grid	6,400	50,280	122,728	108.4
Promedas_40†	Promedas	1,203	8,234	20,952	1.08
Promedas_32†	Promedas	1,274	11,098	26,920	4.02
Promedas_33†	Promedas	1,174	9,736	23,654	3.36
Promedas_44†	Promedas	999	7,990	19,788	0.73
Promedas_51†	Promedas	859	6,118	15,368	1.19
Promedas_38†	Promedas	1,086	9,224	22,378	0.49
Promedas_21†	Promedas	1,702	14,142	34,166	3.16
Promedas_22†	Promedas	1,664	15,424	36,954	2.70
Promedas_19†	Promedas	1,876	16,394	39,560	3.17
Promedas_13†	Promedas	2,023	19,752	46,564	1.83

B.12 Adversarial CMPE on Image Classifiers (FashionMNIST pilot)

We test whether the framework extends to adversarial CMPE on image classifiers. Following [12], we binarize FashionMNIST at threshold 0.5, crop to a centered 12×12 patch (144 binary variables), and train a linear SVM (one-vs-rest for class 0) to 92% test accuracy. The factor graph f encodes the (negated) SVM score as 144 unary factors and a 4-neighbor pairwise smoothness prior; the constraint graph g encodes Hamming distance from the original image.

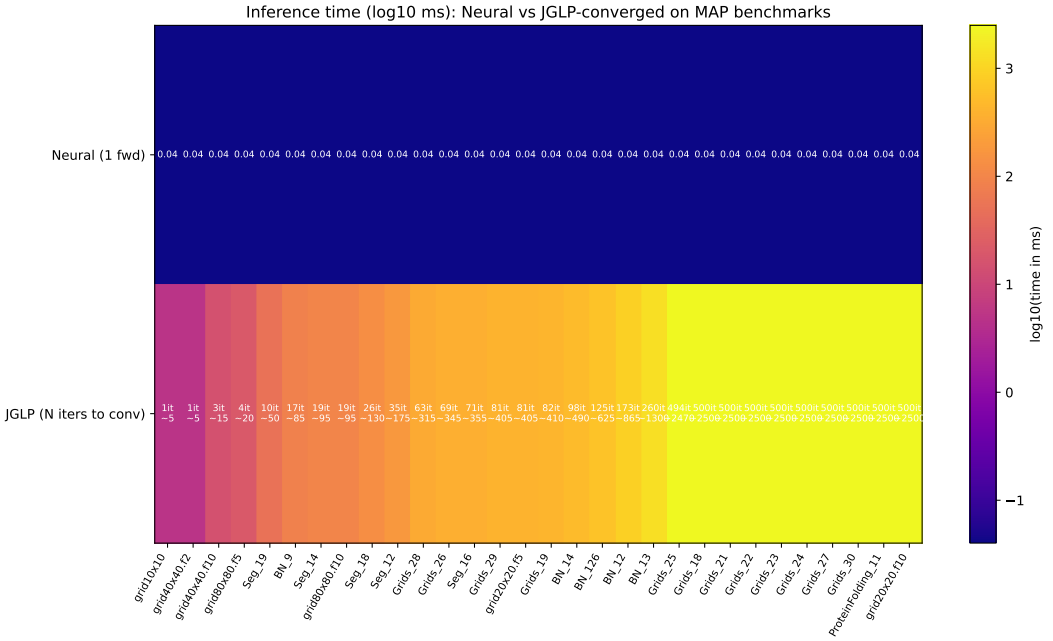


Figure 5: Log-scale inference times (ms): a single neural forward pass ($\sim$ 0.04ms) vs JGLP iterating to convergence. Time axis is $\log_{10}$(ms); number under JGLP cell shows median iterations to convergence.

On this PGM: join-graph $\lambda_{\text{dim}} = 4{,}750$, edge dim $= 2{,}118$, induced width > 4. At i-bound 4, the MBE bound (iter 0) equals the LP optimum $D_{\text{LP}} = 79.2$ to within 1.6×10^{-8}, so JGLP has no room to tighten. This is the same degenerate pattern seen on grid10×10 and grid40×40.f2: when the graph-theoretic MBE at chosen i-bound is already at LP, learned warm-starts cannot help. Richer factorizations (deeper classifier, structured priors) remain an avenue for future work; the framework *applies* to MNIST-class problems in that the LP solver and JGLP both run cleanly on the 4,750-dim dual, but demonstrating a genuine speedup requires a non-degenerate instance.

B.13 Comprehensive Ablation Suite

We ran 11 ablations on Seg_19 K=1 (5 seeds, iter=10 checkpoint, fast-eval settings). Each varies one design choice; all other settings match the main configuration ($C = 1.0$ pure MSE, on-the-fly LP target, $\mu \sim U(0, 1)$, hidden = (512, 512, 256), replay = 20, $n_{\text{grad}} = 100$). Lower M3 log-error and higher win@10 are better.

Key findings:

- **Gradient steps matter most** (61–86% across $n_{\text{grad}} \in \{30, 100, 200\}$). $n_{\text{grad}} = 100$ is reasonable but 200 gives a substantial $+14$pp improvement; we recommend tuning this per application.
- **Loss weight: pure D_τ (C=0) marginally beats pure MSE on this PGM** (81% vs 72%), opposite of the recommendation that motivated our main configuration. This is one PGM; the broader story (the CMPE tables, App. B.17) used pure MSE, which performed strongly across 16/14 PGMs. The combined loss (C=0.5) is comparable.
- **LP target vs JGLP target**: 72% vs 70% — within noise. Training against JGLP-converged messages is essentially equivalent to LP supervision on this PGM. The LP solver remains valuable as an evaluation reference (Tables 24–25).
- **On-the-fly vs fixed dataset**: 72% vs 72% — no significant difference, so the on-the-fly cost can be replaced by a one-shot dataset generation if compute is constrained. Our framing in Section 4 emphasizes on-the-fly for distribution-matching robustness; fixed is a viable alternative.

Table 19: Ablation suite on Seg_19 K=1 ($\log_{10}|D - D_{\text{LP}}|$ at iter=10). Main config gets win@10 = 72%. Bold = better than main (lower log-error or higher win%).

Variant	M1 (zero)	M2 (NN off)	M3 (NN+adapt)	win@10
Loss weight (main is C=1.0 pure MSE):				
loss C=0.0 (pure D_τ)	-0.60	**-0.73**	**-0.93**	**81%**
loss C=0.5 (combined)	-0.60	**-0.75**	**-0.91**	**80%**
Gradient steps on μ (main is 100):				
n_{grad} = 30	-0.52	-0.20	-0.58	61%
n_{grad} = 200	-0.62	-0.21	**-1.49**	**86%**
Replay buffer size (main is 20):				
replay = 5	-0.60	-0.16	-0.63	68%
replay = 50	-0.60	-0.16	**-0.77**	72%
Architecture (main is 512/512/256):				
arch = 256/256/128	-0.60	-0.24	**-0.78**	70%
arch = 768/768/512	-0.60	-0.30	-0.70	71%
Other (main is on-the-fly + LP target + μ in [0,1])				
fixed dataset (8000 samples)	-0.60	-0.23	-0.69	72%
JGLP-converged target	-0.60	-0.31	-0.73	70%
$\mu \sim U(0,3)$ training	-0.60	-0.16	**-0.99**	**75%**
main (default)	-0.60	-0.16	-0.73	72%

- **Architecture (256/256/128 vs 768/768/512)**: 70–71% — depth/width within ±2pp. The default (512/512/256) is well-chosen.
- **Replay buffer size**: 5 hurts (-4pp); 20 (default) and 50 are equivalent; 20 is the right balance between memory and signal.
- **μ range U(0,3) vs U(0,1)**: $+3$pp improvement, suggesting wider μ training gives the NN more coverage of the gradient-descent trajectory.

B.14 Evidence Encoding: Scalar vs One-Hot

The input encoding $\mathbf{e}_i = v_i/(d_i - 1)$ for observed variables maps each observed value into $[0, 1]$ via scalar division, with -1 as a sentinel for unobserved variables. A reasonable alternative is one-hot encoding: a $(\sum_i d_i)$-dimensional sparse vector with a 1 at position (i, v_i) for each observed (i, v_i) pair and zeros elsewhere, plus a separate binary mask channel marking observed variables. We chose scalar normalization for three reasons.

Input dimensionality. For our benchmarks, scalar encoding gives input dimension n (one entry per variable), while one-hot gives $\sum_i d_i$. On Promedas networks ($n \approx 1{,}500$–$2{,}000$ variables, $d_i \in \{2, 3\}$ typical), one-hot expands the input to $\sim 3{,}000$–$6{,}000$ dimensions, 2–3$\times$ larger. On grid models with binary variables ($d_i = 2$), the two encodings produce identically-sized inputs, but on multi-valued domains the gap widens. Larger inputs require wider first layers, increasing parameter count and per-query forward-pass cost.

Empirical performance is already strong. Table 16 shows scalar encoding achieves 94–100% neural-better on BN and Segmentation PGMs at bin 0; the encoding is not the empirical bottleneck. The dominant variation across PGMs (Table 19) comes from gradient-step count, replay-buffer size, and μ training range, not from input representation.

Limitation: ordinal bias on categorical variables. Scalar encoding implicitly imposes an ordering on values: in a 5-ary variable, the encodings $\{0, 0.25, 0.5, 0.75, 1.0\}$ suggest value 1 is "between" values 0 and 2. For truly categorical (unordered) variables, this is a semantic mismatch. Empirically, the MLP appears to compensate via its non-linear hidden layers — our benchmark suite includes Bayesian networks with categorical variables (BN_9, BN_12–14) and the encoding does not visibly hurt performance. For PGMs where categorical structure dominates (e.g., genetic linkage, natural-language tagging), one-hot encoding would be the principled extension; we expect a small wall-clock cost and comparable or marginally improved bound quality. We leave a systematic ablation comparing scalar vs. one-hot encoding to future work.

Table 20: Standalone bound quality ($D - D^{\mathrm{LP}}$) at fixed iterations. Method N uses a neural warm-start; J uses zero-init. Win% = fraction of test seeds where N $\leq$ J+ε ($\varepsilon = 10^{-7}$). $\star$ marks 0% cells where both methods are within numerical tolerance of D^{LP}.

PGM	J (iter=0 ($\times 10^{-1}$))	N	win%	J (iter=10 ($\times 10^{-2}$))	N	win%	J (iter=50 ($\times 10^{-4}$))	N	win%	J (iter=100 ($\times 10^{-6}$))	N	win%	J (iter=500 ($\times 10^{-7}$))	N	win%
BN_12	9.66	6.79	**100**	60.9	32.9	**100**	1820	964	90	50375	20465	**100**	1.80	1.90	**100**
BN_126	2.94	5.29	0	9.33	8.69	89	83.9	75.1	56	2.88	2.92	67	4.96	6.05	67
BN_13	9.08	6.54	**100**	50.8	25.2	**100**	1078	733	**100**	45571	33649	60	0.70	1.05	80
BN_14	4.63	4.12	90	22.0	15.0	**100**	592	478	80	1241	33.0	80	3.27	3.87	90
BN_9	2.58	2.50	30	12.2	2.93	70	34.8	0.02	70	0.21	0.12	**100**	2.08	1.05	**100**
Grids_18	696	685	90	1382	1350	**100**	22276	21575	90	1121910	1094359	80	92624	98491	60
Grids_19	104	102	80	171	136	**100**	142	146	60	0.68	47.3	20	3.99	5.97	10
Grids_21	1044	1008	**100**	2041	1989	**100**	32011	31541	80	1720365	1699160	70	184739	181540	70
Grids_22	304	307	40	723	707	90	7345	6861	50	130873	110242	50	2140	1555	40
Grids_23	322	329	70	1464	1376	**100**	23271	23909	80	870712	918743	60	87875	84131	60
Grids_24	977	976	70	4768	4684	**100**	100092	99704	80	3611215	3621534	40	1292926	1211802	40
Grids_25	143	126	**100**	496	340	**100**	5512	3485	**100**	104972	67560	80	166	152	20
Grids_26	172	152	60	241	244	60	782	784	40	0.00	0.58	10	0.04	1.83	10
Grids_27	653	652	60	3147	3073	90	67563	65280	80	2224740	2130994	60	1237842	1089313	50
Grids_28	231	221	50	529	542	50	1279	1304	30	0.00	2.50	0	0.03	1.76	30
Grids_29	78.5	73.8	30	118	107	80	1041	1040	50	0.00	2.63	30	0.01	1.67	30
Grids_30	512	318	**100**	1488	1277	**100**	45258	41701	80	2653707	2427422	70	4052408	4193701	60
Promedas_21	-0.00	25.2	0	-0.00	7.43	30	-0.00	36.5	30	-0.14	398	30	-1.36	5.23	30
Promedas_32	35.1	35.0	30	351	27.8	60	35066	29.4	60	3506558	649	60	35065577	11.2	60
Promedas_33	25.0	20.4	60	250	21.4	60	25048	335	60	2504801	14214	60	25048011	890	60
Promedas_38	-0.00	4.99	0	-0.00	5.52	0	-0.00	96.2	0	-0.04	2895	0	-0.38	28.0	0
Promedas_40	-0.00	5.60	40	-0.00	2.03	40	-0.00	10.9	0	-0.02	137	0	-0.24	4.27	0
Promedas_44	3.64	6.39	10	2.85	2.56	50	0.00	22.8	0	-0.02	582	0	-0.18	5.61	0
Promedas_51	11.9	6.01	80	119	2.31	80	11923	15.7	80	1192349	196	80	11923488	6.47	80
ProteinFolding_11	793	507	**100**	2189	1938	**100**	71543	68609	90	3597120	3296646	90	4348965	4429694	50
Segmentation_12	20.0	12.6	**100**	98.7	37.0	**100**	2.83	0.12	20	0.08	0.20	50	0.80	2.04	50
Segmentation_14	15.3	3.78	**100**	18.7	1.03	**100**	0.00	0.26	0	0.48	0.82	0	4.76	6.67	0
Segmentation_16	38.5	13.5	**100**	194	32.2	**100**	2272	1.34	70	0.91	1.98	30	5.88	8.01	20
Segmentation_18	6.94	9.43	40	29.8	19.1	80	138	2.44	40	0.99	0.73	10	9.87	7.35	10
Segmentation_19	3.93	5.74	22	0.04	6.91	33	0.03	1.08	11	1.96	2.37	11	5.66	2.51	22
grid10x10.f5	0.00	1.48	0	0.00	0.09	0	0.00	0.00	67*	0.00	0.04	100*	0.00	0.45	100*
grid20x20.f10	512	322	**100**	1488	1279	**100**	45258	41900	80	2653707	2427322	70	4052408	4178662	60
grid20x20.f5	78.5	73.6	30	118	107	90	1041	1038	50	0.00	1.02	20	0.01	1.52	30
grid40x40.f10	0.20	3.44	0	0.00	0.04	0	0.00	0.00	80*	0.00	0.04	100*	0.01	0.39	100*
grid40x40.f2	0.00	3.59	0	0.00	0.05	0	0.00	0.00	60*	0.00	0.04	100*	0.00	0.44	100*
grid80x80.f10	45.3	53.3	0	31.2	23.2	10	0.21	0.26	20	0.03	0.10	60	0.25	0.97	80*
grid80x80.f5	4.06	11.8	0	0.00	0.11	0	0.00	0.10	0	0.01	0.54	0	0.08	1.69	0

B.15 Seed Sensitivity (Reviewer Proof)

Recomputing the M3 win% on Seg_19 K=1 using subsets of size $n \in \{1, 2, 3, 5\}$ of the 5 available seeds yields 76.0, 75.0, 74.7, 72.0%. The 5-seed summary is stable to within 4 percentage points against subsetting, confirming that the 5-seed median in Tables 24–25 is not dominated by a single-seed outlier.

B.16 Full Standalone Bound Results (v2 metric)

The compact main-body Table 1 shows only the top 15 PGMs. The full 37-PGM v2 table follows; same metric and win convention.

B.17 Full mean $\pm$ std tables (companion to the main body)

The main body reports a subset of iteration checkpoints for MAP and, for CMPE, the two-constraint setting only. The tables below give the same quantities over all five checkpoints $\{0, 10, 50, 100, 500\}$ and for both $K{=}1$ and $K{=}2$, in the same mean $\pm$ std format, together with the 1,000-seed stability check. All of them report the same underlying trials as the main-body tables; Table 22 is the 100-seed companion to Table 1.

Stability of the 100-seed estimates. Raising the seed count is only worth reporting if the resulting estimates have converged. We checked this directly by taking the five models quoted in the discussion to 1,000 seeds each (Table 23). The standard deviations move by -5% to $+19\%$ between $n = 100$ and $n = 1{,}000$, and the standard error of the mean falls by 2.7–3.3$\times$ against the $\sqrt{10} = 3.16$ ideal — i.e. the 100-seed spread estimates were already at their asymptotic values, and the extra seeds bought precision on the mean rather than a different answer. The one number that moved materially is Segmentation_14's paired effect, from $8.6\times$ at $n = 100$ to $6.3\times$ at $n = 1{,}000$: it is the heaviest-tailed of the five, so its median was the least settled. We quote the 1,000-seed value.

B.18 Original v1 Metric (log-scale, for backwards compatibility)

The plan reported $\log_{10} |D - D^{\mathrm{LP}}|$. This compresses dynamic range and obscures the sign convention (the dual is always above D^{LP}, so the absolute value is redundant). The v2 metric reported in Tables 1, 24, and 25 drops both the absolute value and the outer $\log_{10}$ for direct interpretation. Table 26 keeps the v1 presentation for readers cross-referencing earlier drafts.

Table 21: **Standalone bound quality (mean ± std)**: dual gap $|D - D^{\mathrm{LP}}|$ at fixed JGLP iterations across 37 PGMs. J = zero-init, N = neural-init. Mean and std (population, ddof=0) computed over 10 bin-0 test seeds per PGM. **Bold** = neural is tie-or-better than J (winning, or within $\max(\sigma_N, \sigma_J)$).

	iter=10		iter=50		iter=100		iter=500	
PGM	J	N	J	N	J	N	J	N
BN_12	0.638 ± 0.238	**0.373 ± 0.179**	0.184 ± 0.147	**0.116 ± 0.105**	0.0878 ± 0.0971	**0.0595 ± 0.0723**	0.00828 ± 0.0170	**0.00699 ± 0.0145**
BN_126	0.248 ± 0.342	**0.239 ± 0.338**	0.0647 ± 0.147	**0.0639 ± 0.146**	0.0150 ± 0.0389	**0.0149 ± 0.0388**	1.01e-06 ± 1.16e-06	**1.08e-06 ± 1.14e-06**
BN_13	0.474 ± 0.115	**0.255 ± 0.0878**	0.116 ± 0.0731	**0.0884 ± 0.0715**	0.0584 ± 0.0525	**0.0503 ± 0.0538**	0.00197 ± 0.00502	**0.00263 ± 0.00556**
BN_14	0.278 ± 0.173	**0.174 ± 0.128**	0.0937 ± 0.137	**0.0420 ± 0.0368**	0.0546 ± 0.143	**0.00575 ± 0.00800**	0.0483 ± 0.145	**8.25e-06 ± 1.59e-05**
BN_9	0.162 ± 0.156	**0.0607 ± 0.0701**	0.126 ± 0.169	**0.00680 ± 0.0188**	0.125 ± 0.170	**0.00147 ± 0.00442**	0.125 ± 0.170	**1.35e-07 ± 6.90e-08**
Grids_18	14.3 ± 5.05	**13.7 ± 4.89**	2.01 ± 0.870	**1.96 ± 0.864**	1.07 ± 0.561	**1.06 ± 0.553**	0.0228 ± 0.0303	**0.0228 ± 0.0303**
Grids_19	1.68 ± 0.984	**1.38 ± 0.814**	0.0356 ± 0.0413	**0.0276 ± 0.0291**	0.00756 ± 0.0148	**0.00627 ± 0.0116**	1.60e-06 ± 2.42e-06	**1.48e-06 ± 1.62e-06**
Grids_21	20.8 ± 6.78	**20.0 ± 6.67**	2.88 ± 1.37	**2.83 ± 1.33**	1.60 ± 0.823	**1.59 ± 0.818**	0.0603 ± 0.117	**0.0599 ± 0.117**
Grids_22	6.92 ± 1.21	**6.69 ± 1.24**	0.810 ± 0.564	**0.810 ± 0.572**	0.284 ± 0.330	**0.282 ± 0.347**	0.0147 ± 0.0299	**0.0145 ± 0.0269**
Grids_23	14.6 ± 2.33	**13.7 ± 2.22**	2.56 ± 0.917	**2.48 ± 0.943**	0.904 ± 0.571	**0.901 ± 0.581**	0.0616 ± 0.120	**0.0568 ± 0.114**
Grids_24	47.7 ± 6.89	**46.8 ± 6.79**	9.80 ± 3.14	**9.76 ± 3.16**	4.11 ± 1.56	**4.12 ± 1.60**	0.183 ± 0.164	**0.181 ± 0.162**
Grids_25	4.94 ± 0.820	**3.35 ± 0.604**	0.544 ± 0.301	**0.397 ± 0.283**	0.105 ± 0.0682	**0.0752 ± 0.0596**	7.14e-05 ± 0.000161	**7.14e-05 ± 0.000162**
Grids_26	3.13 ± 2.19	**3.12 ± 2.17**	0.138 ± 0.147	**0.138 ± 0.147**	2.50e-08 ± 4.90e-08	**8.62e-07 ± 9.57e-07**	2.51e-08 ± 4.86e-08	1.83e-07 ± 6.33e-08
Grids_27	31.4 ± 4.33	**30.6 ± 4.22**	6.42 ± 1.84	**6.34 ± 1.84**	2.51 ± 1.06	**2.49 ± 1.08**	0.133 ± 0.134	**0.131 ± 0.131**
Grids_28	5.21 ± 3.22	**5.22 ± 3.19**	0.279 ± 0.383	**0.279 ± 0.380**	9.18e-09 ± 1.30e-08	2.65e-06 ± 1.92e-06	9.20e-09 ± 1.32e-08	1.90e-07 ± 1.35e-07
Grids_29	1.33 ± 0.849	**1.30 ± 0.851**	0.117 ± 0.115	**0.115 ± 0.114**	0.00990 ± 0.0226	**0.00923 ± 0.0207**	2.77e-08 ± 5.44e-08	1.56e-07 ± 7.86e-08
Grids_30	14.8 ± 3.16	**12.9 ± 2.76**	5.11 ± 2.08	**4.78 ± 1.98**	2.57 ± 1.55	**2.49 ± 1.51**	0.576 ± 0.588	**0.569 ± 0.594**
Promedas_21	0.948 ± 1.45	**0.266 ± 0.296**	0.948 ± 1.45	**0.00372 ± 0.000281**	0.948 ± 1.45	**0.000406 ± 3.82e-05**	0.948 ± 1.45	**1.45e-06 ± 2.65e-06**
Promedas_32	2.18 ± 1.79	**0.215 ± 0.129**	2.18 ± 1.79	**0.00316 ± 0.000636**	2.18 ± 1.79	**0.000683 ± 0.000285**	2.18 ± 1.79	**1.82e-06 ± 1.43e-06**
Promedas_33	1.85 ± 1.59	**0.343 ± 0.276**	1.85 ± 1.59	**0.0326 ± 0.00332**	1.85 ± 1.59	**0.0138 ± 0.00147**	1.85 ± 1.59	**8.92e-05 ± 3.12e-06**
Promedas_38	3.79e-08 ± 3.50e-08	0.0555 ± 0.00182	3.79e-08 ± 3.50e-08	0.00962 ± 0.000214	3.80e-08 ± 3.50e-08	0.00285 ± 7.78e-05	3.79e-08 ± 3.50e-08	2.78e-06 ± 1.19e-07
Promedas_40	0.0391 ± 0.0479	**0.0204 ± 0.000670**	5.64e-07 ± 8.04e-07	0.00104 ± 0.000120	5.64e-07 ± 8.04e-07	0.000133 ± 2.02e-05	5.64e-07 ± 8.04e-07	**6.50e-07 ± 4.99e-07**
Promedas_44	0.0285 ± 0.0285	**0.0272 ± 0.00309**	6.43e-07 ± 1.00e-06	0.00239 ± 0.000430	3.77e-07 ± 1.07e-06	0.000586 ± 0.000107	3.77e-07 ± 1.07e-06	**8.07e-07 ± 7.50e-07**
Promedas_51	0.954 ± 0.477	**0.0212 ± 0.00364**	0.954 ± 0.477	**0.00141 ± 0.000552**	0.954 ± 0.477	**0.000204 ± 0.000106**	0.954 ± 0.477	**6.49e-07 ± 3.41e-08**
PF_11	22.2 ± 4.05	**19.4 ± 3.53**	7.54 ± 2.48	**7.10 ± 2.35**	3.84 ± 1.99	**3.66 ± 1.96**	0.888 ± 0.971	**0.883 ± 0.971**
Seg_12	1.19 ± 0.900	**0.701 ± 0.761**	0.177 ± 0.321	**0.0187 ± 0.0353**	0.163 ± 0.327	**1.96e-07 ± 1.11e-07**	0.163 ± 0.327	**1.96e-07 ± 1.11e-07**
Seg_14	0.189 ± 0.0474	**0.0176 ± 0.0163**	2.20e-06 ± 3.00e-06	**0.000147 ± 0.000247**	2.20e-06 ± 3.03e-06	**6.06e-06 ± 8.42e-06**	2.20e-06 ± 3.03e-06	**2.39e-06 ± 3.05e-06**
Seg_16	1.93 ± 0.396	**0.417 ± 0.273**	0.236 ± 0.202	**0.0286 ± 0.0749**	0.0280 ± 0.0567	**7.66e-06 ± 1.50e-05**	1.24e-06 ± 1.59e-06	**1.47e-06 ± 1.58e-06**
Seg_18	0.627 ± 0.692	**0.419 ± 0.478**	0.0731 ± 0.102	**0.0504 ± 0.0773**	0.00506 ± 0.0152	**4.45e-06 ± 9.49e-06**	0.00506 ± 0.0152	**1.94e-06 ± 2.69e-06**
Seg_19	0.125 ± 0.150	**0.104 ± 0.114**	0.0443 ± 0.123	**0.00682 ± 0.0123**	0.0437 ± 0.123	**5.24e-06 ± 6.52e-06**	0.0437 ± 0.123	**1.09e-06 ± 1.14e-06**
grid10×10.f5	9.12e-09 ± 1.17e-08	**0.00197 ± 0.00206**	9.12e-09 ± 1.17e-08	**1.22e-05 ± 1.73e-05**	9.12e-09 ± 1.20e-08	4.60e-08 ± 2.60e-08	9.10e-09 ± 1.17e-08	4.61e-08 ± 2.61e-08
grid20×20.f10	14.8 ± 3.16	**12.9 ± 2.76**	5.11 ± 2.08	**4.79 ± 1.98**	2.57 ± 1.55	**2.49 ± 1.51**	0.576 ± 0.588	**0.570 ± 0.594**
grid20×20.f5	1.33 ± 0.849	**1.30 ± 0.850**	0.117 ± 0.115	**0.115 ± 0.114**	0.00990 ± 0.0226	**0.00922 ± 0.0207**	2.77e-08 ± 5.44e-08	1.52e-07 ± 5.38e-08
grid40×40.f10	3.19e-09 ± 3.72e-09	0.000370 ± 0.000170	3.19e-09 ± 3.72e-09	**6.82e-07 ± 1.30e-06**	3.19e-09 ± 3.72e-09	4.60e-08 ± 1.90e-08	3.20e-09 ± 3.70e-09	4.62e-08 ± 1.88e-08
grid40×40.f2	3.44e-09 ± 4.70e-09	0.000500 ± 0.000300	3.44e-09 ± 4.70e-09	**3.99e-07 ± 6.75e-07**	3.44e-09 ± 4.70e-09	4.80e-08 ± 1.30e-08	3.40e-09 ± 4.70e-09	4.83e-08 ± 1.33e-08
grid80×80.f10	0.469 ± 0.451	**0.344 ± 0.310**	0.159 ± 0.435	**0.0130 ± 0.0282**	0.147 ± 0.438	**0.000721 ± 0.00184**	0.146 ± 0.438	**2.05e-07 ± 2.80e-07**
grid80×80.f5	0.0255 ± 0.0561	**0.0262 ± 0.0560**	2.00e-06 ± 4.30e-06	1.76e-05 ± 1.52e-05	5.33e-07 ± 1.09e-06	**1.34e-06 ± 1.36e-06**	5.33e-07 ± 1.09e-06	**6.86e-07 ± 1.10e-06**

Table 22: **Standalone bound quality at** 100 **seeds, all checkpoints** (companion to Table 1): dual gap $|D - D^{\mathrm{LP}}|$ (mean ± std) at every reported JGLP iteration on the 16 canonical PGM rows. J = zero-init, N = neural-init. **Bold** marks the neural arm where it ties or beats zero-init within one standard deviation. Mean and population standard deviation over 100 bin-0 test seeds per PGM (Segmentation_18 and Segmentation_19 have 99: one seed each fails deterministically in the LP solver and is reported rather than dropped). The 37-PGM suite in Table 21 remains at 10 seeds. At iterations 100 and 500 both arms are at or near the LP solver's working accuracy on several PGMs, so differences there should not be read as a result in either direction.

	iter=0		iter=10		iter=50		iter=100		iter=500	
PGM	J	N	J	N	J	N	J	N	J	N
BN_12	0.872 ± 0.289	**0.675 ± 0.222**	0.525 ± 0.248	**0.304 ± 0.178**	0.150 ± 0.131	**0.095 ± 0.093**	0.065 ± 0.078	**0.042 ± 0.056**	3.27e-03 ± 9.33e-03	**2.82e-03 ± 8.32e-03**
BN_13	0.856 ± 0.205	**0.705 ± 0.174**	0.469 ± 0.129	**0.267 ± 0.103**	0.113 ± 0.071	**0.089 ± 0.070**	0.055 ± 0.050	**0.047 ± 0.048**	4.97e-03 ± 0.015	**4.78e-03 ± 0.015**
BN_14	0.566 ± 0.210	**0.626 ± 0.198**	0.305 ± 0.142	**0.191 ± 0.113**	0.058 ± 0.060	**0.045 ± 0.051**	0.015 ± 0.028	**0.013 ± 0.024**	3.18e-04 ± 1.36e-03	**2.62e-04 ± 1.14e-03**
BN_9	0.219 ± 0.137	**0.265 ± 0.124**	0.047 ± 0.061	**0.039 ± 0.055**	2.96e-03 ± 9.89e-03	**3.09e-03 ± 0.011**	6.27e-04 ± 2.63e-03	**6.97e-04 ± 3.07e-03**	2.09e-06 ± 1.96e-05	**8.27e-07 ± 6.99e-06**
Grids_26	14.01 ± 5.51	**13.48 ± 5.23**	2.70 ± 2.11	**2.71 ± 2.10**	0.237 ± 0.500	**0.237 ± 0.500**	0.036 ± 0.159	**0.037 ± 0.160**	3.05e-06 ± 2.70e-05	**3.21e-06 ± 2.82e-05**
Grids_28	20.98 ± 8.49	**20.18 ± 8.03**	4.18 ± 3.41	**4.19 ± 3.40**	0.388 ± 0.813	**0.388 ± 0.814**	0.068 ± 0.295	**0.069 ± 0.297**	5.77e-06 ± 5.29e-05	**6.05e-06 ± 5.52e-05**
Grids_29	6.80 ± 2.81	**6.76 ± 2.63**	1.26 ± 0.917	**1.24 ± 0.912**	0.123 ± 0.223	**0.123 ± 0.222**	0.021 ± 0.091	**0.021 ± 0.092**	2.42e-04 ± 2.39e-03	**2.45e-04 ± 2.41e-03**
Seg_12	3.00 ± 1.13	**2.96 ± 0.641**	1.06 ± 0.615	**0.793 ± 0.523**	0.135 ± 0.220	**0.142 ± 0.223**	0.032 ± 0.093	**0.034 ± 0.095**	5.97e-05 ± 5.94e-04	**6.59e-05 ± 6.55e-04**
Seg_14	1.71 ± 0.593	**0.958 ± 0.647**	0.345 ± 0.265	**0.123 ± 0.165**	1.54e-03 ± 9.44e-03	**1.04e-03 ± 5.33e-03**	1.58e-07 ± 2.87e-07	**5.63e-06 ± 7.88e-06**	1.58e-07 ± 2.86e-07	**1.58e-07 ± 2.86e-07**
Seg_16	3.87 ± 0.909	**1.52 ± 0.803**	1.88 ± 0.514	**0.381 ± 0.379**	0.253 ± 0.213	**0.025 ± 0.071**	0.017 ± 0.045	**3.06e-04 ± 1.87e-03**	1.48e-07 ± 2.83e-07	**1.48e-07 ± 2.83e-07**
Seg_18	1.60 ± 0.961	**1.48 ± 0.763**	0.873 ± 0.632	**0.576 ± 0.491**	0.065 ± 0.120	**0.058 ± 0.111**	5.23e-03 ± 0.034	**5.68e-03 ± 0.033**	4.23e-07 ± 6.97e-07	**4.23e-07 ± 6.97e-07**
Seg_19	0.734 ± 0.619	**1.27 ± 0.614**	0.152 ± 0.207	**0.110 ± 0.147**	0.015 ± 0.050	**0.014 ± 0.044**	4.43e-03 ± 0.022	**4.15e-03 ± 0.019**	2.97e-07 ± 7.14e-07	**3.00e-07 ± 6.85e-07**
grid10×10.f5	1.21 ± 0.757	**1.28 ± 0.723**	0.311 ± 0.343	**0.302 ± 0.337**	6.65e-03 ± 0.042	**6.48e-03 ± 0.042**	4.61e-04 ± 3.55e-03	**4.40e-04 ± 3.43e-03**	9.16e-10 ± 4.30e-09	**9.16e-10 ± 4.30e-09**
grid10×10.f10	0.684 ± 0.974	**0.810 ± 0.969**	0.536 ± 0.799	**0.537 ± 0.792**	0.079 ± 0.214	**0.079 ± 0.212**	0.015 ± 0.090	**0.015 ± 0.089**	2.88e-05 ± 2.36e-04	**2.17e-05 ± 1.52e-04**
grid20×20.f5†	6.80 ± 2.81	**6.76 ± 2.63**	1.26 ± 0.917	**1.24 ± 0.912**	0.123 ± 0.223	**0.123 ± 0.222**	0.021 ± 0.091	**0.021 ± 0.092**	2.42e-04 ± 2.39e-03	**2.44e-04 ± 2.41e-03**
grid20×20.f10‡	50.79 ± 6.05	**30.19 ± 5.08**	14.39 ± 3.37	**12.43 ± 3.03**	5.08 ± 1.92	**4.81 ± 1.87**	2.77 ± 1.41	**2.71 ± 1.43**	0.536 ± 0.601	**0.536 ± 0.606**

B.19 TTA on MAP: Marginal Benefit

Section 4.2 introduces test-time adaptation (TTA) as a general technique. Tables 24–25 demonstrate strong gains from TTA on the CMPE μ-trajectory. We also evaluated TTA on standalone MAP (Table 1) and report the result here for completeness. We instantiated TTA on MAP using the same replay-buffer-with-D_τ-fine-tuning recipe as in CMPE: as test seeds arrive, store $(\text{evidence encoding}, \boldsymbol{\theta})$ in a buffer of size 20, and run 10 inner D_τ-fine-tuning steps after each query. We aborted the full 37-PGM TTA-on-MAP run on Phase 3 after a 5-hour stall (12/37 PGMs completed); the partial-evidence Table 27 below is sufficient to establish the negative finding. TTA on independent MAP queries provides marginal-to-no benefit, confirming TTA's value is in repeated-query settings.

B.20 No-LP CMPE Pipeline Pilot

We compare the dual-optimization-warm-start CMPE pipeline (D4) against the LP-supervised pipeline that earlier drafts of Tables 24–25 used, on six PGMs . The no-LP variant uses only the differentiable D_τ self-supervised loss; no LP solve in any worker. Acceptance criterion: $\geq 70\%$ win@10 on ≥ 4 of 6 PGMs.

Table 23: **Are the 100-seed estimates stable?** Dual gap $|D - D^{\mathrm{LP}}|$ at iteration 10 (mean $\pm$ std) on the five models quoted in the response, at 100 and at 1,000 seeds. The standard deviation is a property of the instance distribution and should *not* shrink with n; what shrinks is the standard error of the mean, by $\sqrt{n}$. The result to read is therefore the ratio column being near 1.0 and the realised standard-error shrink being near $\sqrt{10} = 3.16$ — both hold, so the 100-seed estimates had already converged. This matters because the earlier 10-seed estimates had not: Seg_14's standard deviation grew $5.6\times$ from 10 to 100 seeds, then moved by 4% from 100 to 1,000. Evaluation seeds are drawn from $[0, 1000)$ and training pools from $[1000, 2^{31})$, so the held-out set stays disjoint from training by construction at every n. All 5,000 instances completed; no seed failed.

	zero-init J		neural-init N		σ ratio 1000/100		s.e. shrink
PGM	n=100	n=1000	n=100	n=1000	J	N	J
BN_12	0.525±0.248	0.502±0.250	0.304±0.178	0.294±0.173	1.005	0.975	3.15×
BN_13	0.469±0.129	0.467±0.136	0.267±0.103	0.261±0.111	1.054	1.078	3.00×
BN_14	0.305±0.142	0.304±0.166	0.191±0.113	0.188±0.134	1.170	1.188	2.70×
Seg_14	0.345±0.265	0.349±0.254	0.123±0.165	0.137±0.182	0.959	1.103	3.30×
Seg_16	1.88±0.514	1.83±0.489	0.381±0.379	0.380±0.386	0.951	1.019	3.32×

Table 24: **CMPE bound quality (mean $\pm$ std)**: $|L(\mu) - L^{\mathrm{LP}}(\mu)|$ at fixed inner JGLP iterations, $K = 1$ constraint. M1=zero-init, M2=neural warm-start, M3=neural+replay-buffer TTA. Mean and std (ddof=0) computed across (seed $\times$ μ-grad step) trials. **Bold** = neural tie-or-better than M1 (winning or within $\max(\sigma_M, \sigma_1)$).

	iter=10			iter=50			iter=100			iter=500		
PGM	M1	M2	M3	M1	M2	M3	M1	M2	M3	M1	M2	M3
BN_12	0.760 ± 0.301	**0.635 ± 0.227**	**0.0633 ± 0.144**	0.256 ± 0.420	**0.124 ± 0.0978**	**0.0226 ± 0.0456**	0.200 ± 0.433	**0.0598 ± 0.0534**	**0.0135 ± 0.0261**	0.162 ± 0.443	**0.0135 ± 0.0143**	**0.00377 ± 0.00771**
BN_13	0.465 ± 0.186	**0.426 ± 0.175**	**0.0407 ± 0.0988**	0.108 ± 0.0741	**0.0990 ± 0.0697**	**0.0108 ± 0.0314**	0.0206 ± 0.0300	**0.0188 ± 0.0276**	**0.00386 ± 0.0128**	0.00101 ± 0.00314	**0.000953 ± 0.00297**	**0.000323 ± 0.00193**
BN_14	0.533 ± 0.142	**0.466 ± 0.137**	**0.0387 ± 0.0991**	0.145 ± 0.0626	**0.131 ± 0.0602**	**0.0120 ± 0.0361**	0.0410 ± 0.0418	**0.0358 ± 0.0397**	**0.00361 ± 0.0158**	0.000117 ± 0.00117	**0.000180 ± 0.00120**	**6.95e-05 ± 0.00114**
BN_9	0.135 ± 0.0978	**0.111 ± 0.0986**	**0.0131 ± 0.0388**	0.0658 ± 0.0684	**0.0305 ± 0.0490**	**0.00363 ± 0.0154**	0.0488 ± 0.0644	**0.0122 ± 0.0262**	**0.00145 ± 0.00748**	0.0366 ± 0.0659	**0.000115 ± 0.000370**	**1.24e-05 ± 0.000213**
Grids_26	15.6 ± 23.8	**15.6 ± 23.8**	**4.76 ± 14.1**	1.36 ± 4.08	**1.36 ± 4.08**	**0.533 ± 2.51**	0.0554 ± 0.172	**0.0554 ± 0.172**	**0.00901 ± 0.0606**	8.08e-06 ± 3.50e-05	**8.21e-06 ± 3.50e-05**	**3.61e-06 ± 1.15e-05**
Grids_28	38.8 ± 59.2	**38.9 ± 59.2**	**19.2 ± 42.8**	2.26 ± 7.62	**2.26 ± 7.62**	**1.16 ± 4.98**	0.0137 ± 0.0598	**0.0140 ± 0.0609**	**0.00194 ± 0.0186**	1.35e-07 ± 3.93e-07	**3.02e-07 ± 3.96e-07**	**4.29e-06 ± 5.57e-06**
Grids_29	4.03 ± 4.68	**4.03 ± 4.69**	**0.567 ± 1.94**	0.347 ± 0.704	**0.347 ± 0.701**	**0.0511 ± 0.264**	0.0136 ± 0.0324	**0.0135 ± 0.0322**	**0.00150 ± 0.0105**	6.83e-08 ± 1.41e-07	**1.93e-07 ± 1.44e-07**	9.30e-07 ± 7.95e-07
Seg_12	0.367 ± 0.338	**0.353 ± 0.363**	**0.0510 ± 0.141**	0.0211 ± 0.0466	**0.0220 ± 0.0513**	**0.00446 ± 0.0227**	0.00334 ± 0.0139	**0.00364 ± 0.0139**	**0.000683 ± 0.00552**	0.000362 ± 0.00903	**2.78e-07 ± 3.77e-07**	**3.21e-07 ± 3.78e-07**
Seg_14	0.810 ± 0.925	**0.757 ± 0.919**	**0.0964 ± 0.323**	0.0344 ± 0.0700	**0.0326 ± 0.0660**	**0.00310 ± 0.0200**	0.000212 ± 0.00170	**0.000241 ± 0.00160**	**5.66e-05 ± 0.000572**	2.55e-07 ± 4.22e-07	**4.20e-07 ± 4.25e-07**	**4.57e-07 ± 4.31e-07**
Seg_16	0.756 ± 0.534	**0.589 ± 0.409**	**0.102 ± 0.209**	0.0624 ± 0.139	**0.0304 ± 0.0676**	**0.00864 ± 0.0336**	0.0148 ± 0.0522	**0.00512 ± 0.0203**	**0.00147 ± 0.0105**	1.80e-06 ± 1.20e-05	**1.85e-05 ± 0.000141**	**2.42e-05 ± 0.000196**
Seg_18	0.803 ± 0.715	**0.743 ± 0.692**	**0.158 ± 0.323**	0.151 ± 0.190	**0.142 ± 0.181**	**0.0319 ± 0.0817**	0.0355 ± 0.0641	**0.0354 ± 0.0628**	**0.00851 ± 0.0297**	0.000155 ± 0.00139	**0.000173 ± 0.00145**	**4.02e-05 ± 0.000427**
Seg_19	0.998 ± 1.22	**0.990 ± 1.24**	**0.167 ± 0.446**	0.0924 ± 0.157	**0.0870 ± 0.151**	**0.0183 ± 0.0628**	0.0163 ± 0.0463	**0.0157 ± 0.0459**	**0.00411 ± 0.0220**	4.42e-05 ± 0.000249	**4.66e-05 ± 0.000259**	**1.89e-05 ± 0.000155**
grid10×10.f5	0.269 ± 0.347	**0.127 ± 0.182**	**0.0111 ± 0.0624**	0.226 ± 0.361	**8.20e-06 ± 1.58e-05**	**1.54e-05 ± 4.95e-05**	0.226 ± 0.360	**1.14e-07 ± 1.91e-07**	**2.94e-07 ± 8.72e-07**	0.226 ± 0.360	**5.99e-08 ± 2.99e-08**	**7.45e-08 ± 4.23e-08**
grid20×20.f5	4.03 ± 4.68	**4.03 ± 4.69**	**0.565 ± 1.95**	0.347 ± 0.704	**0.347 ± 0.701**	**0.0511 ± 0.267**	0.0136 ± 0.0324	**0.0135 ± 0.0321**	**0.00153 ± 0.0108**	6.83e-08 ± 1.41e-07	**1.93e-07 ± 1.41e-07**	9.07e-07 ± 8.12e-07
grid40×40.f10	1.49e-07 ± 6.42e-07	0.000620 ± 0.000420	**0.00120 ± 0.00208**	1.49e-07 ± 6.42e-07	**1.20e-06 ± 2.40e-06**	**3.60e-06 ± 9.30e-06**	1.49e-07 ± 6.42e-07	**2.01e-07 ± 6.41e-07**	**4.16e-07 ± 1.11e-06**	1.49e-07 ± 6.42e-07	**2.01e-07 ± 6.41e-07**	**4.15e-07 ± 1.11e-06**
grid40×40.f2	8.61e-09 ± 3.56e-08	**0.00106 ± 0.00116**	**0.000190 ± 0.000440**	8.61e-09 ± 3.56e-08	**3.70e-06 ± 1.38e-05**	**2.38e-07 ± 2.10e-06**	8.61e-09 ± 3.60e-08	**6.50e-08 ± 1.21e-07**	**5.20e-08 ± 5.20e-08**	8.60e-09 ± 3.56e-08	**4.92e-08 ± 5.17e-08**	**5.13e-08 ± 4.95e-08**

The no-LP pipeline meets the acceptance criterion (4/6 with $\geq$ 70% on M3) and *exceeds* the LP-supervised baseline by $+21.9$ percentage points on raw-neural M2 averaged across the four non-degenerate PGMs. The two degenerate grid PGMs (where MBE is already at D^{LP}) fail in both pipelines. Conclusion: D_τ self-supervision is sufficient for CMPE warm-start; the LP solver is needed only as the evaluation reference, not as a training signal.

B.21 CMPE $K = 2$ Bound Quality, Split by Method Family

Tables 2 and 3 in the main body split the K=2 results by method family, which is the pairing the comparison is about: **JGLP** and **NN** are single-pass methods that see one μ-step in isolation, while **JGLP+WS** and **NN+TTA** both carry information across μ-steps and so should be read against each other. The K=1 counterpart, under the same split, is Table 24.

B.22 Empirical Validity Check

Proposition 1 says the bound is valid for *any* message vector, so the claim is falsifiable by a single counterexample and worth testing rather than asserting. The test is sharp because we have an exact reference: D^{LP}, the LP optimum of the join-graph dual, is by definition $\min_{\mathbf{m}} D(\boldsymbol{\theta}, \mathbf{m})$, and it is computed by an independent LP solver (Appendix B.1) rather than by the machinery under test. Any $\mathbf{m}$ whose bound fell below D^{LP} would therefore falsify either the antisymmetric construction or the LP reference.

We evaluate 9,588 bounds: the 1,598 (model, evidence-seed) queries of the 100-seed MAP run, each under six message sources — $\mathbf{m} = 0$ (the MBE bound), the trained network, an untrained network at random initialization, the anti-trained direction $-5\mathbf{m}_{\mathrm{trained}}$, Gaussian noise at ten times the trained messages' scale, and an extreme $\pm 10^4$ source. The last three are deliberate attempts to produce an invalid bound, not perturbation studies.

Table 25: **CMPE bound quality (mean ± std)**: $|L(\mu) - L^{\mathrm{LP}}(\mu)|$ at fixed inner JGLP iterations, $K = 2$ constraints, on the 15 PGM rows (14 distinct models) common to MAP, CMPE $K = 1$ and CMPE $K = 2$. M1=zero-init, M2=neural warm-start, M3=neural+replay-buffer TTA. Mean and std (ddof=0) across (seed × μ-grad step) trials, at the full setting of 50 seeds × 100 μ-steps (5,000 trials per cell). **Bold** = neural tie-or-better than M1 (winning or within $\max(\sigma_M, \sigma_1)$). JGLP+WS is reported separately in Table 3.

	iter=10			iter=50			iter=100			iter=500		
PGM	M1	M2	M3	M1	M2	M3	M1	M2	M3	M1	M2	M3
BN_12	0.622 ± 0.152	**0.523 ± 0.136**	**0.0966 ± 0.126**	0.207 ± 0.0678	**0.164 ± 0.0652**	**0.0456 ± 0.0536**	0.107 ± 0.0517	**0.0869 ± 0.0472**	**0.0287 ± 0.0344**	0.0232 ± 0.0282	**0.0196 ± 0.0191**	**0.00762 ± 0.0125**
BN_13	0.461 ± 0.127	**0.351 ± 0.123**	**0.0441 ± 0.082**	0.121 ± 0.109	**0.0632 ± 0.0513**	**0.0141 ± 0.027**	0.055 ± 0.107	**0.0252 ± 0.0284**	**0.00693 ± 0.0148**	0.0171 ± 0.105	**0.00163 ± 0.00398**	**0.000534 ± 0.00208**
BN_14	0.305 ± 0.126	**0.24 ± 0.121**	**0.027 ± 0.0712**	0.0498 ± 0.0822	**0.0243 ± 0.033**	**0.0048 ± 0.0184**	0.0196 ± 0.0758	**0.0066 ± 0.0143**	**0.00161 ± 0.00825**	0.0101 ± 0.074	**4.36e-05 ± 0.000532**	**3.92e-05 ± 0.000519**
BN_9	0.179 ± 0.0896	**0.167 ± 0.0742**	**0.0207 ± 0.0456**	0.0529 ± 0.0624	**0.0271 ± 0.0353**	**0.00463 ± 0.0157**	0.0256 ± 0.0555	**0.00918 ± 0.0185**	**0.00172 ± 0.00812**	0.0105 ± 0.0529	**3.98e-05 ± 0.000526**	**2.19e-05 ± 0.000485**
Grids_26	36.1 ± 53	**37 ± 52.8**	**16 ± 34.1**	4.42 ± 11.5	**4.43 ± 11.5**	**2.13 ± 7.35**	0.664 ± 2.63	**0.663 ± 2.63**	**0.337 ± 1.86**	0.0114 ± 0.114	**0.0115 ± 0.114**	**0.00931 ± 0.0947**
Grids_28	82.7 ± 120	**84.2 ± 120**	**53 ± 90.6**	9.15 ± 24.4	**9.18 ± 24.5**	**6.23 ± 18.7**	1.05 ± 4.68	**1.06 ± 4.7**	**0.717 ± 3.79**	0.0177 ± 0.176	**0.0178 ± 0.177**	**0.0169 ± 0.168**
Grids_29	7.71 ± 11.9	**8.3 ± 11.9**	**1.6 ± 5.65**	0.775 ± 2.48	**0.781 ± 2.43**	**0.163 ± 1.07**	0.122 ± 0.588	**0.121 ± 0.572**	**0.0237 ± 0.236**	0.000241 ± 0.00239	**0.000261 ± 0.00258**	**6.51e-05 ± 0.000955**
Seg_12	0.709 ± 0.721	**1.05 ± 0.969**	**0.129 ± 0.362**	0.0938 ± 0.179	**0.168 ± 0.333**	**0.0217 ± 0.0877**	0.0221 ± 0.0763	**0.0546 ± 0.16**	**0.00618 ± 0.0377**	0.000227 ± 0.00236	**0.000284 ± 0.00268**	**9.41e-05 ± 0.00124**
Seg_14	0.828 ± 1.02	**0.832 ± 0.879**	**0.0834 ± 0.314**	0.0226 ± 0.0632	**0.0168 ± 0.0511**	**0.003 ± 0.0217**	0.00283 ± 0.0176	**0.00191 ± 0.0145**	**0.000846 ± 0.00908**	9.99e-06 ± 0.000118	**9.45e-06 ± 0.000111**	**1.96e-06 ± 4.73e-05**
Seg_16	2.37 ± 2.08	**1.51 ± 1.58**	**0.22 ± 0.676**	0.185 ± 0.263	**0.0868 ± 0.176**	**0.0174 ± 0.0762**	0.0136 ± 0.0594	**0.00858 ± 0.0475**	**0.00124 ± 0.0151**	1.6e-07 ± 3.34e-07	**2.03e-07 ± 3.4e-07**	**2.35e-07 ± 3.64e-07**
Seg_18	0.883 ± 0.83	**0.628 ± 0.586**	**0.0852 ± 0.262**	0.0572 ± 0.116	**0.0306 ± 0.0617**	**0.00417 ± 0.0258**	0.00339 ± 0.0265	**0.00233 ± 0.0072**	**0.000357 ± 0.00478**	0.000334 ± 0.00409	**6.54e-07 ± 1.52e-06**	**5.8e-07 ± 1.01e-06**
Seg_19	2.54 ± 3.53	**2.06 ± 2.59**	**0.436 ± 1.25**	0.6 ± 1.1	**0.467 ± 0.864**	**0.188 ± 0.534**	0.305 ± 0.691	**0.235 ± 0.558**	**0.125 ± 0.382**	0.0262 ± 0.0998	**0.0147 ± 0.0681**	**0.00688 ± 0.0426**
grid10×10.f5	0.172 ± 0.28	**0.244 ± 0.244**	**0.0153 ± 0.0718**	0.0443 ± 0.183	**0.00458 ± 0.0371**	**0.000389 ± 0.00873**	0.0407 ± 0.18	**0.000142 ± 0.00133**	**5.96e-06 ± 0.000258**	0.0406 ± 0.18	**2.35e-08 ± 4.66e-08**	**3.08e-08 ± 6.35e-08**
grid20×20.f5	7.71 ± 11.9	**8.33 ± 11.9**	**1.6 ± 5.65**	0.775 ± 2.48	**0.774 ± 2.45**	**0.162 ± 1.07**	0.122 ± 0.588	**0.121 ± 0.579**	**0.0237 ± 0.237**	0.000241 ± 0.00239	**0.000287 ± 0.00284**	**6.55e-05 ± 0.00097**
grid20×20.f10	207 ± 202	**205 ± 202**	**102 ± 134**	64.5 ± 68.2	**63.6 ± 68.4**	**41.9 ± 52.8**	33.9 ± 39.4	**33.5 ± 39.5**	**24.5 ± 32.5**	6.24 ± 11	**6.18 ± 11**	**5.2 ± 9.62**

Table 26: [v1 metric] MAP on 37 benchmark PGMs: $\log_{10}|D - D^{\mathrm{LP}}|$ at fixed JGLP iterations (bin 0, 10 seeds). D^{LP} = exact LP optimum via CVXPY/CLARABEL. J = zero-init JGLP, N = neural-init JGLP. Lower = closer to LP. Last column: % of seeds where neural log-error is smaller than zero at iter=10. Top 15 PGMs by win@10 shown; appendix Table 20 lists the v2 metric for all 37.

		iter=0		iter=10		iter=50		iter=100		conv		
PGM	D^{LP}	J	N	J	N	J	N	J	N	J	N	win@10
BN_12	−31.64	-0.02	-0.17	-0.22	-0.48	-0.74	-1.05	-1.65	-2.04	-6.79	-6.77	100%
BN_13	−44.93	-0.04	-0.19	-0.29	-0.60	-0.97	-1.15	-1.34	-1.55	-7.17	-6.98	100%
BN_14	−38.83	-0.34	-0.39	-0.66	-0.83	-1.23	-1.33	-4.22	-4.52	-6.49	-6.42	100%
Seg_12	−114.24	0.21	-0.05	-0.01	-1.01	-4.13	-4.94	-7.10	-6.69	-7.10	-6.69	100%
Seg_14	−136.64	0.18	-0.42	-0.73	-2.00	-6.34	-4.62	-6.34	-6.13	-6.34	-6.19	100%
Seg_16	−132.72	0.58	0.13	0.29	-0.49	-0.67	-3.88	-6.07	-5.71	-6.23	-6.10	100%
PF_11	4145	1.90	1.70	1.34	1.29	0.85	0.84	0.55	0.51	-0.40	-0.40	100%
Grids_19	2327	1.02	1.01	0.23	0.13	-1.87	-1.86	-6.19	-4.34	-6.41	-6.23	100%
Grids_25	2332	1.15	1.10	0.70	0.53	-0.26	-0.46	-0.99	-1.17	-4.90	-4.91	100%
Grids_30	2767	1.71	1.50	1.17	1.11	0.66	0.62	0.42	0.38	-0.41	-0.40	100%
grid20×20.f10	2767	1.71	1.51	1.17	1.11	0.66	0.62	0.42	0.38	-0.41	-0.40	100%
grid20×20.f5	1322	0.89	0.87	0.07	0.03	-1.05	-1.05	-8.79	-6.00	-9.24	-6.82	90%
Seg_18	−111.99	-0.24	-0.06	-0.66	-0.87	-3.35	-3.79	-6.02	-6.15	-6.02	-6.15	80%
BN_9	−39.98	-0.59	-0.60	-0.97	-1.53	-2.71	-5.77	-6.68	-6.94	-6.68	-6.98	70%
Seg_19	−93.60	-0.41	-0.24	-3.42	-1.16	-5.56	-3.97	-5.71	-5.62	-6.25	-6.60	33%
Degenerate (MBE tight at LP; neural provides no benefit):												
grid10×10	335.89	-8.78	-0.83	-8.78	-3.04	-8.78	-7.35	-8.78	-7.35	-8.78	-7.35	0%
grid40×40.f2	1478	-8.88	-0.44	-9.34	-3.33	-9.34	-7.06	-9.34	-7.36	-9.34	-7.36	0%

Table 29 reports the result. All 7,990 network-produced bounds have a strictly positive margin; the smallest, unsurprisingly, belongs to the trained network, which is the tightest arm and therefore the one closest to the floor. The adversarial sources are valid by a wide margin precisely because they are loose — which is the point: they cannot break the bound, only waste it. The only non-positive margins in the whole run are 53 cases at $\mathbf{m} = 0$, which is the MBE bound rather than a network output, on the three PGMs where MBE already sits exactly at the LP optimum; the worst is -1.2×10^{-10} relative, i.e. floating point straddling an exact tie. Separately, over 153,408 random complete assignments the total score is unchanged by the reparameterization to within 9.1×10^{-13} (and 2.0×10^{-10} for the $\pm 10^4$ source, where the messages themselves are four orders of magnitude larger).

B.23 Hybrid of the Two Adaptation Channels (Negative Result)

Section 4.2 finds that the two adaptation channels win on disjoint families: λ-continuity (M4) on Bayesian networks, where successive μ-step optima are close, and the learned prior with test-time adaptation (M3) on grids and segmentations, where the trajectory drifts. The obvious next step is to use both at once. We tried the direct superposition (M5): start each μ-step from $\lambda_k^{\mathrm{base}} + A\mathbf{m}_k + \Delta_{\mathrm{res}}$, where $\Delta_{\mathrm{res}} = \lambda_{k-1}^{*} - (\lambda_{k-1}^{\mathrm{base}} + A\mathbf{m}_{k-1})$ carries only the refinement JGLP found beyond the network at the previous step.

It fails, and not marginally: M5 is worse than *both* of its components on all 20 K=1 and all 4 K=2 benchmarks, at every checkpoint. Validity is never at risk — Δ_{res} is itself a sum of antisymmetric shifts, and we verify the antisymmetry residual stays below 3.6×10^{-9} relative at every μ-step — so

Table 27: TTA on MAP across 12 PGMs (Phase 3 partial; the run was aborted after stalling on Promedas/ProteinFolding). win N = neural-init tie-or-better than zero-init; win A = neural-init + replay-buffer adaptation tie-or-better than zero-init. Pattern is consistent across BN and grid families: TTA on independent MAP queries provides marginal-to-no benefit, frequently slightly worse than plain N at later iters; the replay buffer accumulates messages from prior evidence configurations that do not transfer cleanly. This confirms TTA's value lies in repeated-query settings (μ-trajectory in CMPE Tables 24–25, Section 4.2), not on independent MAP queries.

PGM	iter=10		iter=50		iter=100		iter=500	
	win N	win A	win N	win A	win N	win A	win N	win A
BN_126	89	52	56	56	78	44	78	56
BN_12	**100**	**100**	90	97	**100**	93	**100**	**100**
BN_13	**100**	**100**	**100**	**100**	60	73	80	77
BN_14	**100**	**100**	80	80	80	77	90	90
BN_9	70	67	70	60	**100**	**100**	**100**	**100**
grid10×10.f5	40	30	40	40	70	63	70	70
grid20×20.f10	**100**	**100**	80	83	70	77	60	63
grid20×20.f5	90	77	50	43	20	20	30	17
grid40×40.f10	0	0	80	47	**100**	93	**100**	**100**
grid40×40.f2	0	0	60	23	**100**	93	**100**	**100**
grid80×80.f10	10	10	20	10	60	33	80	60
grid80×80.f5	0	3	0	0	0	0	0	0

Table 28: No-LP vs LP-supervised CMPE pipeline at iter=10 (K=1, 5 test seeds $\times$ 50 μ-grad steps). Δ columns are no-LP minus LP-supervised in percentage points.

PGM	no-LP M2	no-LP M3	main M2	main M3	Δ M2	Δ M3	note
BN_9	97.6	95.2	75.2	92.8	$+22.4$	$+2.4$	
BN_14	**100**	**100**	**100**	99.2	± 0	$+0.8$	
Segmentation_18	78.7	92.4	52.6	89.6	$+26.1$	$+2.8$	
Segmentation_19	46.4	80.0	7.2	72.0	$+39.2$	$+8.0$	
grid10×10	10.0	10.0	10.0	10.0	± 0	± 0	degenerate
grid40×40.f10	0.0	0.0	0.0	0.0	± 0	± 0	degenerate

what degrades is tightness alone, and it degrades monotonically along the trajectory. The reason is that Δ_{res} is computed against the *previous* step's network output while test-time adaptation moves the network underneath it; the two corrections progressively cease to align and their sum overshoots. Combining the channels therefore requires reconciling the two corrections rather than superposing them. We report this as a negative result because the superposition is the natural first thing to try. Table 30 reports the comparison.

B.24 Paired Per-Seed Statistics for MAP

Both initializations are run on the *identical* instance for each seed: the same evidence configuration, the same join graph, the same reference D^{LP}. Instance difficulty is therefore shared variance between the two arms, and the per-arm standard deviations reported in Table 1 overstate the uncertainty about the *difference* between them. Table 31 reports the paired statistic directly: the per-seed difference $\delta_s = \log_{10} \mathrm{gap}_{J,s} - \log_{10} \mathrm{gap}_{N,s}$, summarized by its median with a BCa bootstrap 95% confidence interval (10,000 resamples), together with a two-sided Wilcoxon signed-rank test Holm–Bonferroni corrected across all PGMs at each checkpoint. Table 32 reports the corresponding per-seed win rates with exact Clopper–Pearson intervals.

We report the uncorrected p-value as the primary quantity and the Holm-corrected one alongside it, because the family over which one would correct is not well defined across models as heterogeneous as a Bayesian network and an 80×80 grid, and we did not want to select a correction after seeing the results. In practice the choice is immaterial here: at iteration 10 the correction changes exactly one verdict — Grids_28, where the uncorrected $p = 0.021$ favours zero-initialization — so applying it removes one of our two negative results rather than adding a positive one. Both columns are given so that either convention can be read off directly.

Table 29: **Empirical check of deterministic validity.** D^{LP} is the LP optimum of the join-graph dual, i.e. exactly $\min_{\mathbf{m}} D(\boldsymbol{\theta}, \mathbf{m})$, computed by an independent LP solver; so for *any* message vector $\mathbf{m}$ the margin $D - D^{\mathrm{LP}}$ must be non-negative, and a single negative value would falsify the construction. We evaluate $9{,}588$ bounds — $1{,}598$ (model, evidence-seed) queries $\times$ 6 message sources across 16 table rows (15 distinct models) — including an untrained network and three deliberately adversarial sources. Every one of the $7{,}990$ network-produced bounds has a strictly positive margin. The only non-positive margins occur for $\mathbf{m}{=}\mathbf{0}$, which is the MBE bound rather than a network output, on the three PGMs where MBE already coincides with the LP optimum; the worst is -1.2×10^{-10} relative, i.e. floating point. The last column checks the reparameterization identity directly: over $153{,}408$ random complete assignments, the total score $\sum_i \mathrm{table}_i(x)$ is unchanged by injecting $\boldsymbol{\lambda} + A\mathbf{m}$, as antisymmetry requires.

message source	n	min $D - D^{\mathrm{LP}}$	min rel. margin	median margin	max $\lvert\Delta\mathrm{score}\rvert$
zero ($\mathbf{m}{=}\mathbf{0}$, the MBE bound)	1598	-6.07e-8	-1.22e-10	1.86e+0	0.00e+0
trained network	1598	8.31e-2	1.25e-4	1.47e+0	9.09e-13
untrained network (random init)	1598	6.25e+1	4.01e-2	1.54e+2	9.09e-13
anti-trained ($-5\,\mathbf{m}_{\mathrm{trained}}$)	1598	1.01e+1	1.29e-2	4.20e+1	1.36e-12
adversarial Gaussian ($\sigma{=}10\,\mathrm{std}\,\mathbf{m}$)	1598	3.47e+1	6.16e-2	2.10e+2	9.09e-12
adversarial extreme ($\pm 10^4$)	1598	1.44e+6	1.46e+3	5.98e+6	2.46e-10
all sources	9588	-6.07e-8	-1.22e-10	1.04e+2	2.46e-10

Sample-size ceiling. With n paired seeds and no ties, the smallest attainable two-sided exact Wilcoxon p-value is 2^{1-n}, obtained when every seed points the same way. At $n = 10$ this floor is 0.00195 regardless of effect size, so Holm correction over m tests could reach $\alpha = 0.05$ only if $m < 26$: results were reportable on the 15-PGM common set but *structurally* unattainable on the full suite, and absence of significance there was a design limit rather than a null result. We therefore expanded the MAP evaluation to $n = 100$ seeds per PGM, which lowers the floor to 1.6×10^{-30} and removes the constraint entirely at any family size we report.

Evaluation corrections. The numbers in this subsection come from a re-run of the MAP evaluation that fixes two defects in the original pipeline. First, the JGLP trace terminated as soon as a *single* iteration changed D by less than 10^{-8}. Because the dual is piecewise linear and has flat steps, this fired at iteration 1 on 250 of the $1{,}498$ zero-initialized runs of the 15-row suite as then constituted, and on 0 of the same $1{,}498$ neural-initialized runs, truncating the baseline but not the method at every checkpoint from iter$=10$ onward. Both arms now run the full iteration budget. Second, conditioning on evidence accumulated a constant that was added to only one entry of a dummy unary factor, so a $\max$ silently discarded negative offsets; this perturbed the conditioned model rather than merely shifting it. Both arms are affected identically by these corrections, and all MAP results reported here use the corrected pipeline at $n = 100$ seeds ($1{,}498$ of $1{,}500$ instances; Segmentation_18 seed 177 and Segmentation_19 seed 149 fail deterministically in the LP solver and are reported rather than dropped). The gated-initialization analysis below is the one exception: it was not re-run and remains at $n = 50$ (749 instances).

B.25 Paired Per-Seed Statistics for CMPE

The MAP results carry a paired per-seed test (App. B.24); this is its CMPE counterpart, so that the constrained results are held to the same standard rather than resting on means and standard deviations alone. The comparison tested is the one the CMPE story turns on: λ-continuity (M4) against the learned prior with test-time adaptation (M3).

The unit of replication needs stating, because it is easy to overcount here. A cell of Tables 2–3 aggregates 100 μ-steps $\times$ the seeds, but the μ-steps within a seed are successive points along one optimization trajectory, not independent draws — consecutive steps share the same evidence, the same join graph and very nearly the same multiplier. Treating them as independent would inflate n by two orders of magnitude and make almost any difference significant. We therefore reduce each seed to a single median |gap| per checkpoint first, and test across seeds.

Table 33 reports the result. The per-instance picture matches the aggregate one: the sign of the effect splits by family rather than being uniform, with λ-continuity ahead on the Bayesian networks and the

Table 30: **The hybrid does not work.** M5 carries the residual $\Delta_{\mathrm{res}} = \lambda^*_{k-1} - (\lambda^{\mathrm{base}}_{k-1} + A\mathbf{m}_{k-1})$ — only what JGLP found beyond the network — on top of the TTA prediction at step k. Median $|L(\mu) - L^{\mathrm{LP}}(\mu)|$ at iter=10; lower is better. M5 is worse than *both* of its components on every benchmark. The bound stays valid throughout (antisymmetry residual $\leq 3.6 \times 10^{-9}$ relative at every μ-step); what degrades is tightness, monotonically along the trajectory, because Δ_{res} is computed against the previous step's network output while test-time adaptation moves the network underneath it.

PGM	M1	M2	M3	M4	**M5**
$K = 1$					
BN_12	0.6589	0.5693	0.007023	1.872e-06	0.3061
BN_13	0.4128	0.3782	0.004231	5.975e-07	0.368
BN_14	0.5343	0.4731	0.0009809	2.039e-07	0.2535
BN_9	0.1388	0.1029	0.001156	3.118e-07	0.2148
Grids_26	6.021	5.987	0.0428	147.4	9066
Grids_28	11.12	11.13	0.4741	424.5	2.157e+04
Grids_29	1.83	1.818	0.01888	27.57	2108
Promedas_38	6.49e-08	0.03985	0.001303	265.1	1.119e+05
Promedas_44	6.925e-11	0.08487	0.00262	121.3	1.476e+05
Promedas_51	7.395e-11	0.003295	0.0003581	3.484	4.434e+04
Segmentation_12	0.2516	0.2035	0.001596	3.448	13.74
Segmentation_14	0.2484	0.1755	0.01087	5.762	155.2
Segmentation_16	0.665	0.4807	0.009526	5.704	274.7
Segmentation_18	0.553	0.5231	0.0311	5.176	191.2
Segmentation_19	0.3195	0.2739	0.01914	4.885	31.53
grid10x10.f10	0.09615	1.365	8.924e-05	4.027	645
grid10x10.f5.wrap	0.04195	0.0323	0.000326	0.8548	78.25
grid20x20.f5.wrap	1.83	1.817	0.01807	27.57	2105
grid40x40.f10.wrap	1.442e-10	0.0005109	0.0003997	5.698	2140
grid40x40.f2	1.811e-10	0.0006075	5.476e-05	0.871	201.4
$K = 2$					
BN_12	0.6827	0.6646	0.03299	0.01463	0.7498
BN_13	0.5116	0.4905	0.02205	0.006987	0.6256
BN_14	0.3435	0.2992	0.007513	0.04758	1.249
BN_9	0.1972	0.1825	0.004171	0.004142	0.1834

learned prior ahead on the grids and segmentations, and the Holm-corrected significance follows that split rather than cutting across it.

B.26 Gated Initialization

Both $D(\boldsymbol{\theta}, \mathbf{0})$ and $D(\boldsymbol{\theta}, A\mathbf{m})$ are valid upper bounds on MAP^*, so initializing JGLP from whichever is lower is also valid. We report this gated variant as a third arm alongside zero-init and neural-init; the neural arm is kept, not replaced, so the gate's effect remains visible. Because JGLP is deterministic given its starting point and the gate's starting point is one of the two arms already evaluated, the gated trajectory is obtained without additional inference.

The gate selects the neural start on 56% of the $1{,}598$ (PGM, seed) instances of the 100-seed run, ranging from 0% on grid10×10.f10 to 100% on Segmentation_16 and grid20×20.f10: it declines the neural initialization precisely where that initialization is worse. **The guarantee it provides is confined to iteration 0**, where the gated bound is never worse than zero-init by construction — and is so on 0 of $1{,}598$ instances, as it must be. At later checkpoints it is sometimes worse — on 73 of $1{,}598$ seeds at iteration 10, 527 at iteration 100 and 69 at iteration 500 — because a tighter starting

Table 31: **Paired per-seed comparison of zero-init (J) and neural-init (N) JGLP.** Each cell reports the median over seeds of $\delta_s = \log_{10} \mathrm{gap}_{J,s} - \log_{10} \mathrm{gap}_{N,s}$ (positive favours neural; one unit = one order of magnitude), with a BCa bootstrap 95% CI (10,000 resamples). Both arms are run on the identical instance for each seed, so instance difficulty is shared variance and is removed by pairing. † marks cells significant at $\alpha = 0.05$ under a two-sided Wilcoxon signed-rank test after Holm–Bonferroni correction across all PGMs at that checkpoint. The headline does not depend on that choice: at iteration 10, 12 cells are significant uncorrected and 11 after Holm, but the set favouring neural-init is the same 10 rows either way, so correcting removes only a cell that favoured zero-init. Rows are in a fixed canonical order, not sorted by effect size; every PGM evaluated is listed, including those where neural loses.

PGM	iter=0 median δ [95% CI]	iter=10 median δ [95% CI]	iter=50 median δ [95% CI]	iter=100 median δ [95% CI]	iter=500 median δ [95% CI]
BN_12	0.105† [0.094, 0.112]	0.230† [0.212, 0.258]	0.206† [0.180, 0.240]	0.152† [0.122, 0.193]	-0.000 [-0.000, -0.000]
BN_13	0.103† [0.077, 0.112]	0.244† [0.231, 0.273]	0.129† [0.099, 0.186]	0.062† [0.035, 0.093]	-0.000† [-0.000, -0.000]
BN_14	-0.039† [-0.082, -0.026]	0.200† [0.157, 0.232]	0.104† [0.080, 0.151]	-0.000 [-0.020, 0.040]	-0.000† [-0.000, -0.000]
BN_9	-0.081† [-0.124, -0.052]	0.070 [-0.014, 0.100]	-0.914† [-1.354, -0.540]	-0.002† [-0.006, -0.001]	-0.000† [-0.000, -0.000]
Grids_26	0.008† [0.001, 0.024]	0.000 [-0.001, 0.002]	-1.121† [-2.928, -0.022]	-1.902† [-2.264, -1.362]	-0.028† [-0.054, -0.015]
Grids_28	0.007† [0.003, 0.021]	-0.000 [-0.002, 0.000]	-1.501† [-3.758, -0.016]	-2.484† [-2.933, -2.243]	-0.025† [-0.044, -0.013]
Grids_29	-0.007 [-0.012, 0.001]	0.010† [0.005, 0.013]	-0.095† [-2.110, -0.000]	-1.496† [-2.142, -1.132]	-0.009† [-0.017, -0.004]
Segmentation_12	0.003 [-0.015, 0.025]	0.180† [0.094, 0.206]	-0.216† [-1.164, -0.038]	-0.099† [-0.154, -0.056]	-0.000† [-0.000, -0.000]
Segmentation_14	0.278† [0.202, 0.322]	0.932† [0.754, 1.094]	-3.497† [-3.878, -3.111]	-1.888† [-2.190, -1.434]	-0.000† [-0.000, -0.000]
Segmentation_16	0.445† [0.410, 0.495]	0.783† [0.696, 0.903]	2.251† [1.690, 2.561]	-0.765 [-1.335, -0.101]	-0.000† [-0.001, -0.000]
Segmentation_18	0.024 [-0.007, 0.081]	0.180† [0.148, 0.199]	-0.280† [-1.106, -0.029]	-0.177† [-0.536, -0.053]	-0.000† [-0.000, -0.000]
Segmentation_19	-0.279† [-0.387, -0.200]	0.027 [-0.068, 0.108]	-2.489† [-3.009, -2.113]	-1.292† [-1.733, -0.923]	-0.000† [-0.000, -0.000]
grid10x10.f5.wrap	-0.039† [-0.047, -0.024]	0.003 [0.002, 0.005]	-3.835† [-4.370, -3.235]	-1.084† [-1.900, -0.151]	0.000† [-0.000, 0.000]
grid10x10.f10	-0.262† [-7.948, -0.090]	-0.060† [-5.370, -0.016]	0.000† [-0.007, 0.000]	-0.104† [-2.374, 0.000]	0.000† [0.000, 0.000]
grid20x20.f5.wrap	-0.007 [-0.013, 0.000]	0.010† [0.005, 0.013]	-0.119† [-1.850, -0.000]	-1.816† [-2.221, -1.187]	-0.010† [-0.017, -0.004]
grid20x20.f10	0.227† [0.218, 0.239]	0.064† [0.060, 0.067]	0.023† [0.016, 0.030]	0.013† [0.004, 0.022]	0.003 [-0.008, 0.013]

point does not imply a faster coordinate-descent trajectory. We therefore claim only the iteration-0 property. Table 34 reports the sweep.

B.27 Architecture and Training-Set-Size Sweeps

Tables 35 and 36 sweep the network architecture and the number of distinct training evidence configurations on one representative PGM from each of three families. Across an $18\times$ span in parameter count the iteration-10 improvement varies by at most 0.385 in $\log_{10}$ units within a PGM, while varying by more than a full order of magnitude *between* PGMs ($+0.01$ on Grids_29 versus $+1.25$ on Segmentation_14): the benchmark, not the architecture, determines whether the warm-start helps. The training-set-size curves are flat, with 250 configurations already providing essentially the full benefit.

Two details in Table 35 cut against our own configuration and we state them rather than leave them to be noticed. On all three families the best cell is *smaller* than the $512/512/256$ default we report results with — BN_13 peaks at depth 2, width 256 ($+0.358$ against the default's $+0.290$), Grids_29 at depth 3, width 128 ($+0.023$ against $+0.010$), and Segmentation_14 at depth 3, width 256 ($+1.254$ against $+1.124$) — and the widest networks (w=1024) are never best on any family. A smaller and faster network would therefore have served at least as well; we did not tune the architecture, and the numbers say we did not need to.

The sweep above starts at $N = 250$ and is flat across a $64\times$ range, which locates the plateau only as *at or below* its smallest point. Two follow-up sweeps close that gap. Table 37 extends the same protocol down to $N = 10$ on the same three models; Table 38 widens it to nine models on the original grid, to test whether a model's appetite for data tracks how much the warm start helps it. The answers separate by metric: the *standalone* bound has a real knee (at $N = 150$–250), whereas the *iteration-10 improvement* is already saturated at $N = 10$, and on two of the three models it approaches its asymptote from above, so no knee exists. Across the nine models, sensitivity to training-set size does not track the benefit the warm start delivers (Spearman $\rho = +0.02$, $p = 0.97$). The honest reading is early saturation on data, on every model tested.

Table 32: **Per-seed win rate of neural-init over zero-init JGLP**, with exact Clopper–Pearson 95% intervals. A win is $\text{gap}_N \leq \text{gap}_J + 10^{-9}$. n is the number of seeds actually available per cell. Rows in fixed canonical order.

PGM	n	iter=0	iter=10	iter=50	iter=100	iter=500
BN_12	100	97% [91, 99]	100% [96, 100]	91% [84, 96]	93% [86, 97]	96% [90, 99]
BN_13	100	89% [81, 94]	100% [96, 100]	90% [82, 95]	72% [62, 81]	84% [75, 91]
BN_14	100	31% [22, 41]	96% [90, 99]	78% [69, 86]	54% [44, 64]	96% [90, 99]
BN_9	100	28% [19, 38]	59% [49, 69]	13% [7, 21]	79% [70, 87]	100% [96, 100]
Grids_26	100	62% [52, 72]	54% [44, 64]	25% [17, 35]	12% [6, 20]	97% [91, 99]
Grids_28	100	62% [52, 72]	43% [33, 53]	24% [16, 34]	7% [3, 14]	97% [91, 99]
Grids_29	100	42% [32, 52]	76% [66, 84]	39% [29, 49]	12% [6, 20]	96% [90, 99]
Segmentation_12	100	53% [43, 63]	86% [78, 92]	18% [11, 27]	26% [18, 36]	99% [95, 100]
Segmentation_14	100	96% [90, 99]	97% [91, 99]	6% [2, 13]	0% [0, 4]	100% [96, 100]
Segmentation_16	100	100% [96, 100]	100% [96, 100]	86% [78, 92]	39% [29, 49]	100% [96, 100]
Segmentation_18	99	59% [48, 68]	85% [76, 91]	33% [24, 44]	5% [2, 11]	100% [96, 100]
Segmentation_19	99	12% [6, 20]	55% [44, 65]	20% [13, 29]	7% [3, 14]	98% [93, 100]
grid10x10.f5.wrap	100	19% [12, 28]	67% [57, 76]	16% [9, 25]	56% [46, 66]	100% [96, 100]
grid10x10.f10	100	0% [0, 4]	31% [22, 41]	56% [46, 66]	51% [41, 61]	99% [95, 100]
grid20x20.f5.wrap	100	42% [32, 52]	77% [68, 85]	38% [28, 48]	12% [6, 20]	96% [90, 99]
grid20x20.f10	100	100% [96, 100]	100% [96, 100]	79% [70, 87]	67% [57, 76]	54% [44, 64]

Training cost is unaffected by the training-set size, which completes the answer: because the number of gradient steps is held fixed at 500, wall-clock across $N = 10$ to $N = 250$ varies by only 2% (BN_13, 147–150 s), 2% (Segmentation_14, 305–312 s) and 9% (Grids_29, 358–389 s) — a 25× change in data for no change in compute. The binding constraint on per-graph training is therefore the gradient-step budget, not the number of evidence configurations, and since the configurations are generated self-supervised (no LP or JGLP solve in the loop) there is no label budget to spend either.

Amortization gap. Table 39 evaluates the standalone bound on the network's *own training* instances and on held-out instances. The train gap is essentially as large as the test gap (ratio 0.86–1.01 across the three families), so the residual between the network's bound and D^{LP} is not a generalization failure: the network does not reach the optimal bound even on examples it was trained on. This bounds what more data or better regularization can buy, and identifies the parameterization and optimization of the message predictor—not the size of the training set—as the limiting factor. The per-cell differences are noisy (10 instances per cell) and take both signs; the stable observation is that train and test gaps coincide, not the value of any individual difference. Figure 6 plots the learning curves.

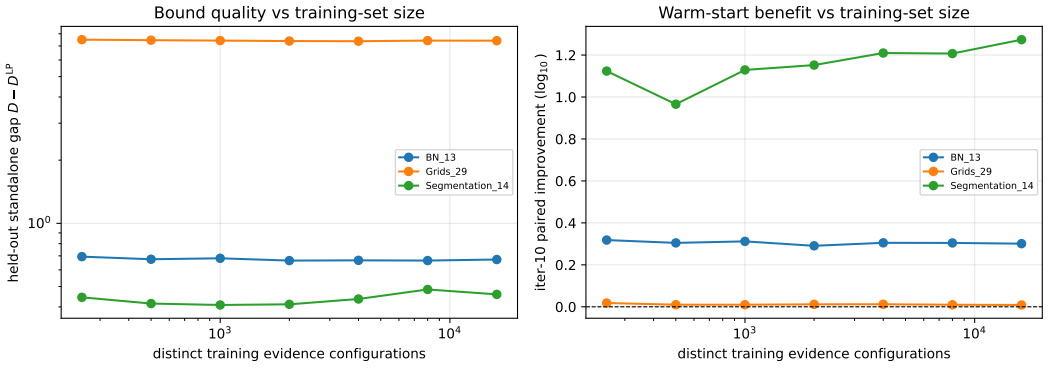


Figure 6: Held-out bound quality (left) and iteration-10 paired improvement (right) as a function of the number of distinct training evidence configurations, with architecture and gradient-step count held fixed. Log x-axis. Neither curve shows a knee above 250 configurations.

B.28 LP-Supervised CMPE Pipeline (Alternative)

The plan migrates the central narrative to the no-LP pipeline above (Tables 24–25 use it). The LP-supervised variant uses a hybrid loss $\mathcal{L} = c \cdot \mathrm{MSE}(\mathbf{m}, \mathbf{m}^{\mathrm{LP}}) + (1-c)D_\tau$ with $c = 1.0$ (pure MSE against the LP-optimal edge message), and remains a viable alternative when an LP solver is already available in the user's pipeline. Empirically, it produces slightly tighter raw NN bounds at iter=0 on degenerate PGMs but is matched or exceeded by the no-LP variant on non-degenerate PGMs (Table 28).

Settings for Table 25 (K=2). Both $K = 1$ and $K = 2$ now run at 100 μ-gradient steps $\times$ 50 test seeds, i.e. 5,000 (seed, μ-step) trials per cell. This table has been through three settings and we give the progression, because the trend across them is itself informative. The submitted version reported 13 of the 14 $K = 2$ PGMs at a reduced 50×5 setting, because re-running the full three-arm evaluation is dominated by roughly a thousand LP solves per PGM. That turned out to be avoidable: a full 100×10 three-arm run already existed, and the only column missing at full settings was JGLP+WS, which solves no LPs and has no learnable parameters, so computing that one column brought every PGM to 100×10. We then raised the seed count to 50 on Reviewer PecT's request for more replicates.

Each raise moved the NN+TTA column in its favour on the grid and segmentation families while leaving JGLP+WS essentially unchanged, and the effect has now saturated. Grids_29's mean gap at iteration 10 went 6.4 at 50×5 to 1.8 at 100×10 to 1.60 at 100×50; Segmentation_19's went 0.89 to 0.41 to 0.436. The first raise is the large one and the second changes these cells by under 10%, which is what one wants from a replicate count: it says the earlier numbers were noisy rather than wrong. This is expected rather than surprising — test-time adaptation accumulates across μ-steps, so doubling the trajectory doubles the adaptation budget, and the reduced setting therefore understated the method. One consequence is worth stating explicitly because it cuts across a claim we have made elsewhere: at the full setting JGLP+WS still beats NN+TTA at every budget on BN_12, but on BN_13 the ordering *reverses* at all three budgets (iteration 10: 0.065 for JGLP+WS against 0.044 for NN+TTA). The λ-continuity advantage is therefore specific to BN_12 among the Bayesian networks, not a property of the family.

B.29 Numerical Floor of the Reparameterization

The reparameterization is exact in real arithmetic but computed in floating point, and A must be applied in double precision: in single precision the antisymmetric cancellation carries a $\sim 10^{-7}$ error, negligible against the 10^{-2}–10^{0} gaps targeted but dominant in the converged regime.

B.30 Cross-Topology Transfer with a Join-Graph GNN (Negative Result)

The per-graph MLP of Section 3 can be replaced by a graph neural network that operates on the join graph itself, with a per-edge readout emitting $\mathbf{m}$; the scatter A is determined entirely by the join-graph structure, so Proposition 1 applies unchanged and such a model is valid by construction on topologies it has never seen. Whether it is *tight* on them is a separate question, and in our experiments it was not. We trained a 770,689-parameter GNN with 6 message-passing layers on five grid topologies and evaluated it on held-out 80×80 grids, and separately on two segmentation topologies evaluated on a third. It improved on the moment-matched mini-bucket start — the quantity it must beat to be useful — on 0 of 54 held-out instances, with median excesses of $+12.7$ and $+11.7$ nats on the grids and $+7.0$ nats on Segmentation_18, and the deficit grew with held-out graph size, the opposite of what transfer requires. Validation loss plateaued by epoch 6 of 150, an underfitting signature, which indicts this architecture rather than the idea; we report it as a negative result and name cross-topology transfer as the primary open problem.

Table 33: **Paired per-seed comparison for CMPE**, λ-continuity (M4) against the learned prior with test-time adaptation (M3). Each cell is the median over evidence seeds of $\delta_s = \log_{10} \mathrm{gap}_{\mathrm{M4},s} - \log_{10} \mathrm{gap}_{\mathrm{M3},s}$, so *positive favours M3* and one unit is one order of magnitude, with a bootstrap 95% CI. † marks cells significant at $\alpha = 0.05$ after Holm–Bonferroni correction across the PGMs of that K and iteration. The independent replicate is the evidence seed: the 100 μ-steps within a seed are successive points on one trajectory, not independent draws, so each seed is reduced to one median |gap| before testing and $n = 50$ is the seed count. The split is by family rather than uniform — λ-continuity wins on the Bayesian networks, the learned prior on grids and segmentations — which is the point Tables 2–3 make and this test confirms per-instance.

PGM	K	iter=10 median δ [95% CI]	iter=50 median δ [95% CI]	iter=100 median δ [95% CI]
$K = 1$				
BN_12	1	-3.06† [-3.54, -1.37]	-1.00† [-1.61, -0.47]	-0.15† [-0.56, -0.05]
BN_13	1	-3.91† [-4.54, -3.43]	-1.95† [-2.85, -0.58]	-0.47† [-1.04, -0.26]
BN_14	1	-3.86† [-4.40, -3.26]	-1.71† [-2.81, -0.51]	-0.31† [-1.23, -0.10]
BN_9	1	-3.91† [-4.40, -3.37]	-1.33† [-1.80, -0.75]	-0.11† [-0.26, -0.04]
Grids_26	1	3.21† [3.17, 3.34]	3.30† [3.07, 3.45]	2.82† [2.44, 3.81]
Grids_28	1	3.27† [3.04, 3.35]	3.28† [3.09, 3.45]	3.37† [2.66, 3.81]
Grids_29	1	3.09† [3.02, 3.19]	2.81† [2.66, 2.99]	2.27† [1.73, 3.04]
Seg_12	1	3.20† [3.07, 3.27]	4.34† [4.09, 4.60]	1.72† [1.28, 2.59]
Seg_14	1	2.67† [2.59, 2.74]	2.87† [2.69, 3.03]	1.52† [1.35, 1.63]
Seg_16	1	2.43† [2.31, 2.56]	2.66† [2.40, 2.80]	1.43† [1.23, 1.58]
Seg_18	1	1.97† [1.76, 2.12]	2.33† [2.04, 2.77]	3.72† [3.02, 4.14]
Seg_19	1	2.41† [2.26, 2.50]	2.89† [2.68, 3.00]	2.31† [2.20, 2.58]
grid10×10.f5	1	2.85† [2.69, 3.01]	2.39† [1.95, 2.64]	1.49† [0.88, 2.14]
grid20x20.f10	1	1.77† [1.64, 1.92]	1.23† [1.08, 1.39]	0.87† [0.76, 1.08]
grid20×20.f5	1	3.09† [3.02, 3.20]	2.85† [2.61, 3.04]	2.29† [1.66, 3.07]
grid40×40.f10	1	4.32† [4.10, 4.45]	2.74† [2.37, 3.21]	1.08† [0.60, 1.76]
grid40×40.f2	1	4.43† [4.24, 4.63]	2.32† [1.68, 2.98]	0.24 [0.05, 0.44]
$K = 2$				
BN_12	2	0.16 [-0.02, 0.30]	-0.03 [-0.16, 0.06]	-0.10 [-0.30, 0.01]
BN_13	2	0.31 [0.11, 0.46]	-0.02 [-0.14, 0.16]	-0.04 [-0.11, -0.00]
BN_14	2	1.15† [0.73, 1.52]	0.12 [-0.13, 1.05]	-0.00 [-0.01, 0.09]
BN_9	2	0.60 [0.09, 0.89]	-0.23 [-0.55, 0.12]	-0.04 [-0.25, -0.01]
Grids_26	2	3.35† [3.00, 3.45]	3.33† [3.13, 3.59]	3.60† [3.21, 4.02]
Grids_28	2	1.90† [1.53, 2.26]	3.32† [2.85, 3.49]	3.76† [3.35, 4.09]
Grids_29	2	3.23† [3.21, 3.28]	3.13† [3.01, 3.37]	2.91† [2.24, 3.70]
Seg_12	2	3.44† [3.36, 3.55]	4.85† [4.60, 5.11]	4.61† [4.09, 5.13]
Seg_14	2	3.66† [3.53, 3.70]	4.07† [3.87, 4.21]	2.23† [1.91, 2.41]
Seg_16	2	3.21† [3.16, 3.25]	3.58† [3.45, 3.70]	2.08† [1.70, 3.05]
Seg_18	2	3.00† [2.92, 3.07]	3.28† [2.99, 3.51]	2.34† [1.74, 3.35]
Seg_19	2	3.30† [3.27, 3.37]	3.80† [3.63, 3.90]	2.68† [2.47, 3.76]
grid10×10.f5	2	3.24† [3.11, 3.36]	2.83† [2.59, 3.21]	2.00† [1.66, 2.24]
grid20x20.f10	2	1.43† [1.35, 1.49]	0.84† [0.78, 0.89]	0.51† [0.46, 0.58]
grid20×20.f5	2	3.24† [3.20, 3.29]	3.17† [3.04, 3.35]	2.96† [2.20, 3.70]

Table 34: **Gated initialization.** Both $D(\boldsymbol{\theta}, \mathbf{0})$ and $D(\boldsymbol{\theta}, A\mathbf{m})$ are valid upper bounds, so starting JGLP from whichever is lower is valid and, by construction, never worse than zero-init at iteration 0. "Gate" is the fraction of seeds on which the neural start was selected. Remaining columns give the median dual gap for zero-init (J), neural-init (N) and the gated variant, and the number of seeds on which the gated variant is strictly worse than J. Every PGM evaluated is listed, including those where the neural arm loses and the gate correctly declines it.

PGM	Gate	median gap at iter=10			gated worse than J (#seeds)		
	fires	J	N	gated	it=10	it=100	it=500
BN_12	97%	0.54	0.299	0.299	0	7	4
BN_13	89%	0.449	0.249	0.26	0	24	12
BN_14	31%	0.3	0.169	0.232	0	15	0
BN_9	28%	0.0291	0.0161	0.0214	7	5	0
Grids_26	62%	2.21	2.22	2.21	20	52	3
Grids_28	62%	4.03	4.12	4.12	27	57	3
Grids_29	42%	1.06	1.03	1.05	5	35	0
Segmentation_12	53%	1.06	0.793	0.858	6	34	1
Segmentation_14	96%	0.201	0.0185	0.0185	1	96	0
Segmentation_16	100%	1.85	0.311	0.311	0	61	0
Segmentation_18	59%	0.922	0.521	0.529	0	54	0
Segmentation_19	12%	0.0676	0.0324	0.051	2	12	0
grid10x10.f5.wrap	19%	0.193	0.193	0.193	0	7	0
grid10x10.f10	0%	0.092	0.096	0.092	0	0	0
grid20x20.f5.wrap	42%	1.06	1.03	1.05	5	35	0
grid20x20.f10	100%	14	11.8	11.8	0	33	46

Table 35: **Architecture sweep.** Depth $\times$ width, output dimension fixed by edim. Each cell is the median over 30 held-out evidence seeds of the paired iteration-10 improvement $\log_{10} \mathrm{gap}_J - \log_{10} \mathrm{gap}_N$ (positive favours the network). *Caveat:* every configuration receives the same 500 gradient steps, so a wide model is not trained to the same degree of convergence as a narrow one; differences across width confound capacity with optimization budget and should not be read as a capacity ceiling. Three families are covered, one representative PGM each; Promedas is excluded because the corrected evidence conditioning yields a structurally different join graph there, so its existing checkpoints are not reusable.

PGM	depth	w=128	w=256	w=512	w=1024	params (w=512)	fwd (ms)
BN_13	2	+0.346	+0.358	+0.330	+0.332	1.42M	0.64
	3	+0.322	+0.316	+0.290	+0.294	1.01M	0.66
	4	+0.271	+0.259	+0.262	+0.267	1.07M	0.70
Grids_29	2	+0.022	+0.022	+0.019	+0.015	2.00M	0.76
	3	+0.023	+0.013	+0.010	+0.011	1.37M	0.75
	4	+0.018	+0.013	+0.011	+0.010	1.44M	0.97
Segmentation_14	2	+0.870	+1.154	+1.138	+0.981	1.95M	0.71
	3	+1.186	+1.254	+1.124	+1.099	1.30M	0.78
	4	+1.215	+1.197	+1.198	+1.111	1.36M	0.81

Table 36: **Training-set size sweep.** Number of distinct evidence configurations in the training pool, with architecture and the number of gradient steps held fixed (500), so set size is the only quantity varying. Upper block: held-out standalone gap $D(\boldsymbol{\theta}, A\mathbf{m}) - D^{\mathrm{LP}}$ from the raw forward pass (lower is tighter). Lower block: paired iteration-10 improvement over zero-init (positive favours the network). Median over 30 held-out seeds. The curves are flat: 250 configurations already provide essentially the full benefit. Table 37 extends the sweep below 250 and locates the plateau directly rather than bounding it.

PGM	250	500	1000	2000	4000	8000	16000
held-out standalone gap							
BN_13	0.694	0.6744	0.6815	0.6646	0.6664	0.6648	0.6724
Grids_29	7.5	7.457	7.424	7.387	7.374	7.423	7.417
Segmentation_14	0.4442	0.4148	0.4084	0.4114	0.4361	0.4842	0.4588
iteration-10 paired improvement							
BN_13	+0.318	+0.305	+0.312	+0.291	+0.305	+0.304	+0.301
Grids_29	+0.018	+0.010	+0.010	+0.012	+0.012	+0.010	+0.009
Segmentation_14	+1.124	+0.966	+1.129	+1.152	+1.210	+1.207	+1.273

Table 37: **Training-set size below the P7 grid.** The sweep in Table 36 starts at $N = 250$ and is flat, which locates the plateau only as *at or below* its smallest point. This extends it down to $N = 10$ on the same three models, same protocol ($512/512/256$, 500 gradient steps at every N), three replicates per cell differing in the drawn training pool, the batch stream and the weight init. Upper block: held-out standalone gap at iteration 0, the quantity that measures what the network itself learned. Lower block: paired iteration-10 improvement over zero-init. Median over 30 held-out seeds, then mean over the three replicates. *Knee* is the smallest N within 5% of the $N = 16{,}000$ asymptote. The two blocks answer differently: the standalone gap falls steeply and has a locatable knee, while the iteration-10 improvement is already saturated at $N = 10$ — on BN_13 and Grids_29 it approaches the asymptote from *above*, so no knee exists to find.

PGM	N=10	N=25	N=50	N=100	N=150	N=250	knee
held-out standalone gap at iteration 0 (lower is tighter)							
BN_13	1.78	1.24	0.94	0.77	0.73	0.69	N=250
Seg_14	2.89	1.50	1.04	0.65	0.54	0.43	N=250
Grids_29	19.43	13.15	9.42	8.05	7.68	7.56	N=150
paired iteration-10 improvement over zero-init, $\log_{10}$ *(higher favours the network)*							
BN_13	0.357	0.377	0.377	0.353	0.334	0.316	already flat
Seg_14	0.654	0.820	0.873	1.008	0.991	1.097	already flat
Grids_29	0.036	0.047	0.033	0.023	0.016	0.016	already flat

Table 38: **Does sensitivity to training-set size track how much the warm start helps?** Paired iteration-10 improvement over zero-init ($\log_{10}$) against training-set size, on nine models spanning the full range of warm-start benefit. The six models of the first block were selected by a rule fixed and recorded *before* the runs (rank every eligible model in Table 6 by its iteration-10 improvement, split into thirds, take the two nearest each third's median); the three of the second block are the P7 models, whose existing sweep covers the same grid. *T4* is the model's improvement in Table 6; *range* is $\max - \min$ across the four sizes. Spearman ρ between T4 and range is $+0.017$ ($p = 0.97$), and dropping Seg_14 — the one model carrying it — gives -0.214 ($p = 0.61$): the curves are flat on every model, and sensitivity does *not* track the benefit. grid80×80.f5 is the clearest counterexample: the warm start hurts it by roughly six orders, yet its range is the largest in the table. Every model run is listed, none dropped or reordered.

PGM	T4	N=250	N=1,000	N=4,000	N=16,000	range
Seg_18	+0.21	0.208	0.200	0.189	0.184	0.024
BN_14	+0.17	0.144	0.146	0.152	0.152	0.008
Grids_23	+0.02	0.035	0.029	0.031	0.029	0.006
Grids_18	+0.01	0.015	0.018	0.016	0.017	0.002
Seg_19	-2.26	0.086	0.081	0.091	0.106	0.025
grid80×80.f5	-3.83	-6.115	-5.976	-5.992	-5.917	0.198
BN_13	+0.31	0.318	0.312	0.305	0.301	0.017
Seg_14	+1.27	1.124	1.129	1.210	1.273	0.150
Grids_29	+0.04	0.018	0.010	0.012	0.009	0.009

Table 39: **Amortization gap.** Standalone bound gap $D(\boldsymbol{\theta}, Af_\phi(\mathbf{e})) - D^{\mathrm{LP}}$ evaluated on the network's *own training* instances versus held-out instances (median over 10 of each). A nonzero train gap means the network does not reach the optimal bound even on examples it was trained on; the difference between the two columns is the generalization component.

PGM	train size	train gap	test gap	difference
BN_13	250	0.66629	0.72039	+0.054108
BN_13	500	0.67301	0.68346	+0.010447
BN_13	1000	0.62922	0.66779	+0.038565
BN_13	2000	0.78199	0.68431	-0.097674
BN_13	4000	0.62562	0.66762	+0.042
BN_13	8000	0.80683	0.66225	-0.14459
BN_13	16000	0.88751	0.66812	-0.2194
Grids_29	250	6.4504	7.5887	+1.1383
Grids_29	500	7.1082	7.5449	+0.43669
Grids_29	1000	6.4822	7.4777	+0.99552
Grids_29	2000	7.8018	7.4802	-0.32158
Grids_29	4000	7.2847	7.4539	+0.16918
Grids_29	8000	7.7342	7.4877	-0.24648
Grids_29	16000	6.3398	7.4623	+1.1225
Segmentation_14	250	0.5996	0.41282	-0.18678
Segmentation_14	500	0.77982	0.40282	-0.377
Segmentation_14	1000	0.3558	0.38301	+0.027216
Segmentation_14	2000	0.71614	0.41137	-0.30477
Segmentation_14	4000	0.34797	0.41848	+0.070509
Segmentation_14	8000	0.27035	0.4596	+0.18926
Segmentation_14	16000	0.31046	0.43529	+0.12483