Differentiating Network Design Objectives for Balancing Cost and Distance

Zishang Chen[1]* **Longhe Lin**[2]* **Zhiyuan Julian Su**[3] **Qi Qi**[1]†‡
[1]Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China
[2]School of Cyber Science and Engineering, Southeast University, Nanjing, Jiangsu, China
[3]Pratt School of Engineering, Duke University, Durham, NC, USA
zishangchen@ruc.edu.cn, linlonghe@seu.edu.cn
zhiyuan.j.su@duke.edu, qi.qi@ruc.edu.cn

Abstract

Many network design tasks must decide which directed arcs to build so that multiple sources can reach a root efficiently. Building fewer or cheaper arcs reduces construction cost, but may force long routing paths; building more arcs shortens routes, but increases construction cost. The Cost–Distance problem formalizes this trade-off, yet it has remained difficult to optimize with gradient-based methods because the selected network, the induced routes, and the routing cost are tightly coupled. We propose Cost Distance Policy Gradient (CDPG), a differentiable framework that treats local next-hop choices as a routing policy, prevents unstable cyclic routing through Dynamic Acyclic Dropout, evaluates the objective with a truncated value solver, and uses the induced acyclic support for quality-preserving TopoRounding. Under the stated conditions, CDPG has a fixed-accuracy $\mathcal{O}(m \log n)$ efficiency guarantee. Experiments cover 1,487 instances in UAV logistics networks, transportation networks, and four synthetic graph families; across this suite, CDPG shows a strong time–quality trade-off compared with other differentiable baselines, Cost–Distance-specific algorithms, and the commercial solver. Our code is available at: `https://github.com/Sakuraced/Cost-Distance-Policy-Gradient-CDPG`.

1 Introduction

Suppose a logistics operator must design a delivery backbone for a city. Many demand locations need to send flow to one depot, while the operator can activate only a subset of candidate roads, drone corridors, or transfer arcs. Activating fewer arcs lowers construction cost, but forces demand to travel along longer routes; activating more arcs shortens delivery paths, but increases the construction cost paid for the network. The decision is therefore not simply to find shortest paths on a given graph, but to choose a shared directed subnetwork whose arcs can serve many sources at once.

This setting is captured by a fundamental class of shared network design objectives: the *Cost–Distance* problem [1]. The directed version considered here asks which arcs should be selected so that many weighted sources can reach a designated root, minimizing the sum of selected-arc construction cost and total source-to-root routing distance. It models the same sparse-versus-short tension in transportation, sensor, and communication networks [2, 3, 4]. Approximation-preserving reductions

*Equal contribution.
†Corresponding author.
‡Also with, Beijing Key Laboratory of Research on Large Models and Intelligent Governance. Also with, Engineering Research Center of Next-Generation Intelligent Search and Recommendation, MOE.

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

further connect Cost–Distance to Facility Location, Multicast Tree Design, and Buy-at-Bulk, making it a representative objective for shared infrastructure design.

Cost–Distance is also a natural target for machine learning for combinatorial optimization (ML4CO), but only if the discrete network-design objective can be turned into a differentiable surrogate. In *unsupervised learning* (UL), such surrogates allow training without optimal labels [5, 6, 7]. In *post-inference refinement* (PIR), a model-generated solution is further improved by gradient descent on the same surrogate [8]. In *direct gradient descent* (DGD), model training is bypassed and the surrogate is optimized directly on each instance [9, 10, 11, 12]. These pipelines differ in initialization and reuse, but they share the same bottleneck: the combinatorial objective and constraints must be encoded into a stable differentiable form.

Existing differentiable methods do not provide such a surrogate for Cost–Distance. The main obstacle is that edge selection and routing are inseparable. Once a source chooses a next hop, all downstream choices determine whether it reaches the root and how much routing distance it accumulates. Therefore, a relaxation cannot treat edges as independent fractional variables; it must also represent the source-to-root routes induced by those variables. Differentiable shortest-path methods address a related routing primitive [13].

In this work, we propose **Cost Distance Policy Gradient (CDPG)**, the first gradient-based framework that casts the Cost–Distance problem into a differentiable form. We instantiate the surrogate as an end-to-end DGD solver that optimizes each instance directly. The core idea is to relax a hard rooted tree into a next-hop routing policy. Each active node maintains a probability distribution over feasible parents, and following these local choices defines a stochastic route. This lets us evaluate routing distance through a Markov decision process (MDP) and a Bellman equation.

Two additional ingredients make this surrogate usable for optimization. First, probabilistic routing can form cycles, allowing probability mass to circulate without reaching the root and making expected distance ill-behaved. CDPG introduces *Dynamic Acyclic Dropout* (DAD), which dynamically keeps the policy on an exact root-oriented acyclic support while staying aligned with the Cost–Distance objective. This structure also enables *TopoRounding*, which converts the soft policy into a hard tree while preserving the relaxed solution quality in linear time.

Second, CDPG evaluates routing through fixed-depth sparse propagation over adjacency lists, with an unreached-flow penalty accounting for the remaining demand. This gives a differentiable computation without explicit matrix inversion during optimization. The formal analysis in Section 5 shows that, under the stated structural conditions, the relaxed objective is smooth, the truncated estimator has controlled relative error, accepted stop-gradient DAD updates satisfy a monotone convergence guarantee, and the resulting procedure has $\mathcal{O}(m \log n)$ complexity.

We evaluate CDPG on 1,487 Cost–Distance instances covering dense UAV logistics networks, 8 sparse transportation regions, and 4 synthetic graph families.

2 Related Work

Cost–Distance. Work on Cost-Distance centers on specialized algorithms and hardness. [1] proposed the first randomized hierarchical aggregation algorithm, with an $\mathcal{O}(\log |S|)$ performance guarantee and running time $\mathcal{O}(|S|^2(m + n \log n))$; [14] derandomized it; and [15] improved the running time to $\mathcal{O}(|S|(m + n \log n))$. On hardness, [16] proved that this problem cannot be approximated better than $\Omega(\log \log n)$ unless $\mathrm{NP} \subseteq \mathrm{DTIME}\big(n^{\mathcal{O}(\log \log \log n)}\big)$. The classical path-flow relaxation [14] couples terminal-specific path flows to shared edge capacities. CDPG uses a shared next-hop policy to induce all source routes and evaluates the coupled construction and routing objective through Bellman propagation.

Differentiable relaxation. Many discrete structures have been relaxed for differentiable optimization, including categorical sampling by Gumbel-Softmax [17], DAG constraints by NOTEARS [18], DAG-GNN [19], and Nocurl [20], and matching or assignment-style problems by Sinkhorn-based relaxations [21, 22, 23]. DataSP differentiates shortest paths with respect to edge weights [13]. Cost–Distance also selects the network on which routing takes place, so its surrogate must couple shared construction costs with demand-weighted path lengths.

ML4CO. ML4CO spans supervised learning [24, 25], reinforcement learning [26, 27, 28], and objective-driven learning and search [5, 6, 7, 8, 9, 10].

3 Cost-Distance Problem Formulation

Intuitively, we are given a directed graph of possible arcs and must build a root-reaching directed subnetwork that routes all demand to one root. A source node carries demand, the root is the destination, and zero-weight Steiner nodes may be used as intermediate connectors. The same selected arc can serve many sources, so the model must balance two effects: selecting an arc pays its construction cost once, while every source using that arc contributes routing distance.

Consider a directed graph $\mathcal{G} \coloneqq (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes, $m = |\mathcal{E}|$ directed arcs, adjacency matrix $X \in \{0,1\}^{n \times n}$, arc construction cost $C \in \mathbb{R}_{>0}^{n \times n}$, and arc routing distance $D \in \mathbb{R}_{>0}^{n \times n}$, where $X_{ij} = 1$ iff the directed arc $(v_i, v_j) \in \mathcal{E}$. Given a root r and non-root sources $S \subseteq \mathcal{V} \setminus \{r\}$, Cost-Distance selects an adjacency matrix $X' \leq X$ of a directed subgraph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$, where $S \subseteq \mathcal{V}' \subseteq \mathcal{V}$, $\mathcal{E}' \subseteq \mathcal{E}$, and every source in S can reach r in $\mathcal{G}'$, that minimizes

$$\mathcal{J}(X') \coloneqq \underbrace{\sum_{i,j} X'_{ij} C_{ij}}_{\Omega_C(X')} + \underbrace{\sum_i w_i \, L_D^{X'}(v_i, r)}_{\Omega_D(X')} . \tag{1}$$

Here $w \in \mathbb{R}_{\geq 0}^n$ is node flow, with $w_i > 0$ iff $v_i \in S$, and $L_D^{X'}(v, r)$ is the directed shortest-path distance from v to r in the selected subgraph under distance D. The two terms are the construction cost $\Omega_C(X')$, paid once per selected directed arc, and the weighted routing distance $\Omega_D(X')$. For example, when two sources share a backbone arc, its construction cost is charged once, while both demands contribute to routing distance along that arc.

Positive construction costs allow unused arcs to be removed. Within a feasible subnetwork, choosing a shortest-path next hop toward the root at each root-reachable non-root node preserves source distances. Positive routing distances ensure that these choices contain no directed cycle. Pruning nodes not reached from a source then yields a root-oriented tree with no larger objective, motivating the parent-choice representation below.

4 Methodology

To optimize the Cost–Distance problem, we propose CDPG, illustrated in Figure 1. CDPG relaxes parent choices into a differentiable next-hop policy, evaluates expected routing cost, updates policy parameters by gradient descent, and applies quality-preserving TopoRounding; Appendix B gives the full algorithm.

4.1 Problem Relaxation

We relax the binary edge-selection matrix X' into a row-wise policy $P \in [0,1]^{n \times n}$. For each active non-root node v_i, $P_{i:}$ is a categorical distribution over feasible next-hop parents, while the root row and inactive Steiner rows are zero. Thus P is a stochastic policy over local edge decisions, not a fractional arborescence; a hard X' is produced only after sampling or TopoRounding in Eq. (10), which makes decisions only for nodes that carry source flow.

The same row distribution is used whenever flow arrives at a node, regardless of its source. Sources therefore share routing preferences after their flows meet, while their demands enter evaluation through w. A Steiner node can serve as a merging point when it receives source flow; nodes outside the source-reachable support have zero policy rows.

The edge mask $0 \leq P \leq X$ removes invalid arcs while allowing directed cycles. A closed cyclic class traps routing mass and gives $\mathbb{E}[\Omega_D] = +\infty$. DAD maintains a root-oriented acyclic support to ensure termination.

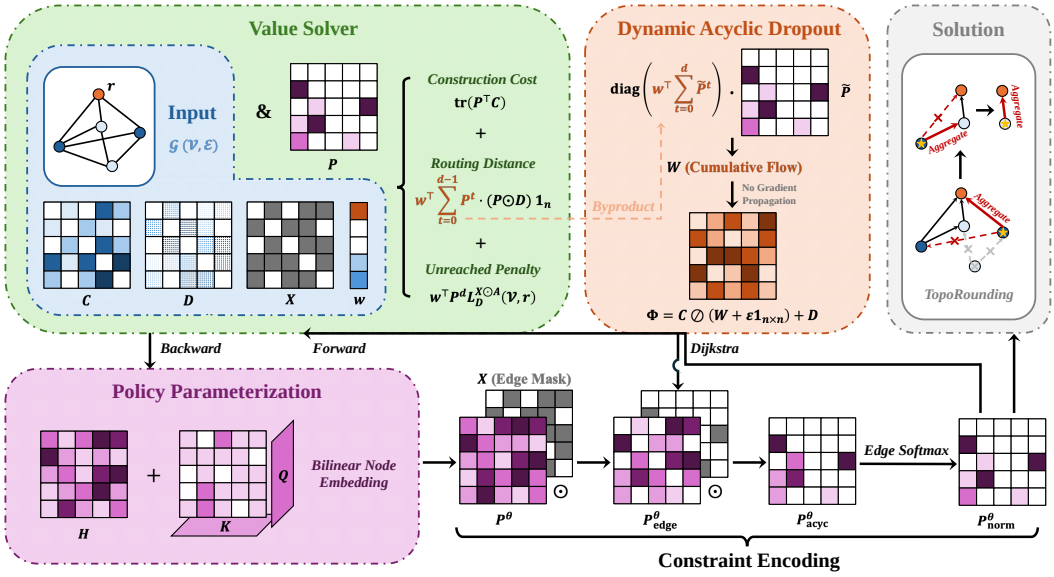

Figure 1: The CDPG pipeline. Edge logits and bilinear node embeddings parameterize a next-hop policy. DAD selects a root-oriented acyclic support, and row normalization gives routing probabilities. Value Solver evaluates construction and induced routing costs. Gradients update policy parameters under the current support; the acceptance test controls mask updates. Finally, TopoRounding converts the optimized policy into a discrete arborescence.

$$\min_P \ \mathcal{J}(P) \coloneqq \mathbb{E}[\Omega_C] + \mathbb{E}[\Omega_D], \tag{2}$$

$$\text{s.t. } 0 \le P \le X, \quad \mathrm{tr}(P^k) = 0,\ \forall k \in \mathbb{N}^+, \quad \sum_j P_{rj} = 0, \tag{3}$$

$$\sum_j P_{ij} = \mathbb{1}\left\{ v_i \in S \ \vee \ \sum_k P_{ki} > 0 \right\}, \quad i \neq r. \tag{4}$$

Constraint (3) enforces original edge support, acyclicity, and an absorbing root. The trace condition is exact for nonnegative P: any positive directed cycle contributes to $\mathrm{tr}(P^k)$ for some k, while zero traces exclude closed walks. Constraint (4) is the active-row condition: sources and Steiner nodes receiving positive incoming probability select one next-hop parent in expectation, while unused Steiner nodes keep zero rows. Acyclicity then ensures every active path terminates at the root, and zero-flow Steiner nodes are never selected in the final hard tree.

4.2 Value Solver

Value Solver estimates $\mathcal{J}(P)$ by computing $\mathbb{E}[\Omega_C]$ and $\mathbb{E}[\Omega_D]$. The construction term is direct: $\mathbb{E}[\Omega_C] = \mathrm{tr}(P^\top C)$. For routing, we use an MDP only as an evaluation device: the fixed policy P samples an outgoing edge from the current node, transitions to the next node, and receives reward $r(u,v) = -D_{uv}$. For a trajectory starting at v_i, write $s_0 = i$, $s_{\tau_i} = r$, and $R_i = \sum_{t=0}^{\tau_i - 1} \gamma^t (-D_{s_t s_{t+1}})$. Let $V_{\gamma,i} = \mathbb{E}[R_i]$ and $\mathbf{V}_\gamma$ collect node values; because rewards are negative distances, $\mathbb{E}[\Omega_D] = -w^\top \mathbf{V}_1$.

Proposition 4.1 (Bellman value)**.** *For a policy table P supported on a root-oriented DAG, the vector value satisfies*

$$\mathbf{V}_\gamma = -(I - \gamma P)^{-1}(P \odot D)\mathbf{1}_n. \tag{5}$$

Here $\mathbf{1}_n$ denotes a vector of ones, and $\odot$ is element-wise multiplication. Since DAD makes P nilpotent, $(I-P)^{-1}$ exists, so $-\mathbf{V}_1$ gives the exact expected routing distance; see Appendix A.1.

Explicit dense inversion in Proposition 4.1 costs $\mathcal{O}(n^3)$. During optimization, we use Neumann iteration [29] to express evaluation as a fixed sequence of sparse propagations.

$$\mathbf{V}_1 = \underbrace{-\sum_{t=0}^{+\infty} P^t (P \odot D)\mathbf{1}_n}_{\text{cyclic}} = \underbrace{-\sum_{t=0}^{n-1} P^t (P \odot D)\mathbf{1}_n}_{\text{acyclic}}.$$

Proposition 4.2 (Finite Neumann expansion)**.** *If P is supported on the DAD acyclic mask, then $P^n = 0$ and the Neumann expansion is exact after at most $n-1$ powers; see Appendix A.2.*

The acyclic expansion still has length n, we truncate it and add a penalty μ for unreached weight:

$$\mathbb{E}[\Omega_D] \geq \mathbb{E}[\Omega_D^{\text{eval}}] \coloneqq w^\top \sum_{t=0}^{d-1} P^t (P \odot D)\mathbf{1}_n + \mu, \quad \mu \coloneqq w^\top P^d L_D^{X \odot A}(\mathcal{V}, r). \tag{6}$$

The row vector $w^\top P^d$ tracks the demand reaching each node after d transitions. The terminal cost attaches this demand to shortest remaining paths on the same support, giving the lower-bound evaluation in Eq. (6) while retaining a fixed propagation depth. Theorem 5.2 shows that $d = \Theta(\log n)$ gives constant relative error under residual-flow contraction and polynomially bounded distance range. Appendix E.2 further validates this choice empirically by sweeping d around $\lceil \log_2 n \rceil$. Algorithm 1 evaluates the sum by setting $q_0 = w$, accumulating $q_t^\top (P \odot D)\mathbf{1}_n$, and updating $q_{t+1} = P^\top q_t$. Each supported arc transfers mass $q_{t,i} P_{ij}$ to its head node and contributes that mass times D_{ij} to routing distance. These accumulations share one adjacency-list scan per step, giving $\mathcal{O}(md)$ work for d steps.

4.3 Unconstrained Policy Parameterization

CDPG assigns logits only to supported edges. For each candidate edge $(i, j) \in \mathcal{E}$, CDPG forms the unconstrained logit $z_{ij} = h_{ij} + l^{-1/2} q_i^\top k_j$, where $h \in \mathbb{R}^m$ is an edge-specific logit vector indexed by $\mathcal{E}$, $q_i, k_j \in \mathbb{R}^l$ are the i-th and j-th columns of $Q, K \in \mathbb{R}^{l \times n}$, and $\theta = \{h, Q, K\}$. Q and K are initialized using a standard Gaussian distribution $\mathcal{N}(0, 1)$, and then normalized $\|Q\|_F = \|K\|_F = 1$ during optimization. The bilinear node embedding is inspired by VGAE [30], where adjacency is reconstructed via inner products. Separate tables Q and K give a node different representations as the tail and head of a directed arc. Changing one embedding affects several incident-edge logits together, sharing parameters across local routing choices. The direct logits h retain independent control of each candidate arc, including edges whose endpoints have similar embeddings. Scoring only supported arcs gives $\mathcal{O}(ml)$ logit-generation cost.

4.4 Constraint Encoding

We map the logits z to a feasible policy through three operations. **(i)** The feasible-edge restriction is enforced by materializing z_{ij} only for $(i, j) \in \mathcal{E}$. **(ii)** DAD constructs an exact acyclic support A and decides which Steiner rows are active. **(iii)** Row-wise normalization is applied on the DAD-active non-root rows; all other Steiner rows and the root row are set to zero:

$$P_{ij} \coloneqq \frac{\exp(z_{ij})}{\sum_{k \in \mathcal{N}_A(i)} \exp(z_{ik})}, \quad \mathcal{N}_A(i) \coloneqq \{k \in \mathcal{V} : X_{ik} A_{ik} = 1\}, \quad i \in \mathcal{V}_{\text{act}},\ j \in \mathcal{N}_A(i), \tag{7}$$

and $P_{ij} = 0$ otherwise. Here $\mathcal{V}_{\text{act}}$ is the set of non-root rows retained by the accepted DAD mask: it contains sources and Steiner nodes reached from sources. Rows outside $\mathcal{V}_{\text{act}}$ are zero and receive no gradient in that DAD epoch. This yields the probabilistic policy $P = f(\theta; A)$ stored on sparse adjacency lists. The resulting policy is evaluated by $\mathcal{J}_d(\theta; A) = \mathbb{E}[\Omega_C] + \mathbb{E}[\Omega_D^{\text{eval}}]$, and gradients are backpropagated through the entire encoding–evaluation pipeline to optimize θ end-to-end.

Dynamic Acyclic Dropout. Let A^t denote the accepted stop-gradient mask, $P^t = f(\theta_t; A^t)$, and $F_t(\theta) \coloneqq \mathcal{J}_d(\theta; A^t)$. CDPG takes a fixed-mask gradient step and proposes a new support:

$$\begin{aligned} \theta_{t+1} &= \theta_t - \eta \nabla_\theta \mathcal{J}_d(\theta_t; A^t), \\ \tilde{P}^{t+1} &= f(\theta_{t+1}; A^t), \qquad \widehat{A}^{t+1} = \text{DAD}(\tilde{P}^{t+1}), \\ A^{t+1} &= \begin{cases} \widehat{A}^{t+1}, & \mathcal{J}_d(\theta_{t+1}; \widehat{A}^{t+1}) \leq \mathcal{J}_d(\theta_{t+1}; A^t), \\ A^t, & \text{otherwise.} \end{cases} \end{aligned} \tag{8}$$

Changing A changes each row's feasible parent set and renormalizes its probabilities even at fixed logits. The acceptance test therefore evaluates both supports at the same updated parameters. Between refreshes, the accepted support and its terminal distances are reused while the policy parameters change.

The operator $\mathrm{DAD}(\cdot)$ constructs $\widehat{A}^{t+1}$ from $\tilde{P}^{t+1}$ without backpropagating through the structural choice. Define the old edge-flow matrix directly by $W \coloneqq \mathrm{diag}\Big(w^\top (I - \tilde{P}^{t+1})^{-1}\Big)\, \tilde{P}^{t+1}$. On the accepted DAG, cumulative flow is computed by one topological propagation, so the inverse notation above requires only $\mathcal{O}(m)$ work. The flow-weighted construction term favors sharing an arc among demands that use it. DAD uses the fixed-W edge metric $\Phi \coloneqq C \oslash (W + \varepsilon \cdot \mathbf{1}_{n\times n}) + D$. Here $\oslash$ denotes element-wise division and $\varepsilon > 0$ avoids division by zero. DAD computes all-node shortest distances $\ell_i^\Phi \coloneqq L_\Phi^X(v_i, r)$, forms the raw root-oriented DAG B^{t+1}, and then keeps only rows reachable from sources:

$$
\begin{aligned}
B_{ij}^{t+1} &\coloneqq \mathbb{1}\big\{X_{ij} = 1,\ v_i \neq r,\ \ell_i^\Phi > \ell_j^\Phi\big\}, \\
\mathcal{R}(B^{t+1}) &\coloneqq \{v : v \text{ is reachable from some source in } B^{t+1}\}, \\
\widehat{A}_{ij}^{t+1} &\coloneqq \mathbb{1}\{v_i \in \mathcal{R}(B^{t+1})\} B_{ij}^{t+1}.
\end{aligned} \tag{9}
$$

Thus every retained transition strictly decreases $L_\Phi^X(\cdot, r)$, so cycles are impossible. The reachable-set factor is the Steiner pruning step: if no source reaches a Steiner node in B^{t+1}, its outgoing row is removed from the evaluated policy and final rounding.

Proposition 4.3 (Support of the fixed-W optimum)**.** *For fixed old edge-flow W, every optimum of the fixed-W routing objective is supported on the DAD mask A induced by $L_\Phi^X(\cdot, r)$ and source-reachable rows. This is shown in Appendix A.3.*

Therefore, DAD constructs a flow-aware candidate from the distance-to-root order induced by Φ, enforcing acyclicity while retaining the fixed-W optimum support. The acceptance test in (8) transfers this structural update to the actual truncated objective $\mathcal{J}_d$.

Discussion (Comparison with soft DAG penalties). Soft DAG penalties (e.g., NOTEARS, DAG-GNN, Nocurl) incur $\mathcal{O}(n^3)$ cost and only discourage cycles. DAD instead constructs a policy-consistent exact acyclic mask with much lower overhead.

4.5 TopoRounding

Rounding must preserve solution quality without dominating runtime. Principled objective-relaxation rounding can certify quality by evaluating n candidate objectives [6], which would cost $\mathcal{O}(nm \log n)$ for Cost–Distance. DAD's exact DAG enables a linear alternative. After optimization, CDPG converts the final soft policy $P = f(\theta^*; A^T)$ into a hard arborescence by a linear pass on the terminal mask A^T. Before rounding, it computes the exact unit-discount value vector $\mathbf{V}_1$ on A^T by one reverse topological pass; see Appendix A.4. TopoRounding initializes $\widehat{w}_i \leftarrow w_i$ and maintains a queue of zero-indegree nodes in A^T. When node i is popped, it selects a hard parent only if its accumulated hard flow $\widehat{w}_i > 0$:

$$
p(i) \in \arg\min_{j : A_{ij}^T = 1} \big\{C_{ij} + \widehat{w}_i \big(D_{ij} - V_{1,j}\big)\big\}, \qquad X_{i,p(i)}^{\mathrm{topo}} = 1. \tag{10}
$$

Here $-V_{1,j}$ is the expected remaining distance from parent j to the root. Topological processing accumulates all incoming hard flow before selecting a parent, so the score balances one construction charge against demand-weighted downstream routing. The flow is then aggregated by $\widehat{w}_{p(i)} \leftarrow \widehat{w}_{p(i)} + \widehat{w}_i$. If $\widehat{w}_i = 0$, no hard edge is added. In both cases, TopoRounding removes i's outgoing edges from the queue graph, so zero-flow Steiner nodes cannot block later nodes. The conditional-expectation proof gives $\mathcal{J}(X^{\mathrm{topo}}) \leq \mathcal{J}(P)$; see Appendix A.4. Computing $\mathbf{V}_1$ and rounding both scan terminal adjacency lists once, so the rounding cost is $\mathcal{O}(m)$.

5 Theoretical Analysis

The analysis establishes smoothness (Theorem 5.1), controls evaluation error (Theorem 5.2), and combines descent with accepted-mask updates (Theorem 5.3). Sparse implementation costs then yield Theorem 5.4. Proofs and constants appear in Appendix A.5–A.8.

Theorem 5.1 (Smoothness of the relaxed objective)**.** *For a fixed acyclic mask A, $F_A(\theta) \coloneqq \mathcal{J}_d(\theta; A)$ is $L_{\theta,d}(A)$-smooth on bounded parameter sets: $\|\nabla F_A(\theta_1) - \nabla F_A(\theta_2)\|_F \le L_{\theta,d}(A)\|\theta_1 - \theta_2\|_F$. An explicit bound for $L_{\theta,d}(A)$ is given in Appendix A.5.*

Theorem 5.2 (Estimation relative error)**.** *For $P = f(\theta; A)$ on a DAD acyclic support, let $r_d \coloneqq w^\top P^d \mathbf{1}_n / \|w\|_1$ be the residual-flow ratio after d transitions, let $\|D\|_{\min} \coloneqq \min_{X_{ij}A_{ij}=1} D_{ij}$, $\|D\|_{\max} \coloneqq \max_{X_{ij}A_{ij}=1} D_{ij}$, and let $\chi_D \coloneqq \|D\|_{\max}/\|D\|_{\min}$. The relative error satisfies*

$$(\mathbb{E}[\Omega_D] - \mathbb{E}[\Omega_D^{\text{eval}}])/\mathbb{E}[\Omega_D] \le n r_d \, \|D\|_{\max}/\|D\|_{\min}.$$

When $r_d \le \rho_{\text{res}}^d$ for some $\rho_{\text{res}} \in (0,1)$, $d = \mathcal{O}(\log(n\chi_D))$ is sufficient for constant relative error; if $\chi_D \le n^{\mathcal{O}(1)}$, this becomes $d = \mathcal{O}(\log n)$.

In our experiments, we instantiate this logarithmic-depth regime with $d = \lceil \log_2 n \rceil$; Appendix E.2 provides numerical evidence that this setting is close to the accuracy–runtime knee.

Theorem 5.3 (Convergence rate of CDPG)**.** *Fix T. Let A^t be the accepted stop-gradient DAD mask, $F_t(\theta) \coloneqq \mathcal{J}_d(\theta; A^t)$, $L \coloneqq \sup_{t<T} L_{\theta,d}(A^t) < \infty$, $F_{\inf}$ be a common lower bound, and $\Delta_0 \coloneqq F_0(\theta_0) - F_{\inf}$. Assume each candidate mask is accepted only when $F_{t+1}(\theta_{t+1}) \le F_t(\theta_{t+1})$. For $\theta_{t+1} = \theta_t - \eta \nabla F_t(\theta_t)$, $\eta = \kappa/L$, and $0 < \kappa \le 1$,*

$$\min_{t<T} \frac{\|\nabla F_t(\theta_t)\|^2}{L\Delta_0} \le \frac{1}{\kappa(1-\kappa/2)T}.$$

For the relative stationarity tolerance $\varepsilon_{\text{rel}} > 0$, defined by $\min_{t<T} \|\nabla F_t(\theta_t)\|^2 \le \varepsilon_{\text{rel}}^2 L\Delta_0$, it is sufficient that $T_{\text{rel}} \ge [\kappa(1-\kappa/2)\varepsilon_{\text{rel}}^2]^{-1}$. Thus fixed ε_{rel} gives $T_{\text{rel}} = \mathcal{O}(1)$.

Why relative stationarity. CDPG optimizes a relaxed policy point that is then passed to TopoRounding; the absolute scale of the continuous surrogate is not itself the output. Thus the natural stopping notion is scale-invariant. Under positive rescaling $F_t \mapsto aF_t$, $a > 0$, both $\|\nabla F_t\|^2$ and $L\Delta_0$ scale by a^2, while $L \mapsto aL$ gives $\eta \mapsto \eta/a$ for $\eta = \kappa/L$. Thus the normalized stationarity criterion, the GD update $\eta\nabla F_t$, and the accepted-mask test remain unchanged. Adaptive implementations such as RMSprop have the same scale-insensitive behavior up to numerical stabilizers. "Fixed" means that ε_{rel} is chosen independent of n, m, d; reducing it only multiplies the iteration bound by $1/\varepsilon_{\text{rel}}^2$.

Theorem 5.4 (Overall computational complexity)**.** *For root-reachable directed graphs stored in sparse adjacency-list form, under the logarithmic-depth conditions of Theorem 5.2, CDPG costs $\mathcal{O}(T_{\text{rel}} m \log n)$. For fixed relative stationarity tolerance ε_{rel}, this gives fixed-accuracy $\mathcal{O}(m \log n)$.*

6 Experiment

We evaluate CDPG on real UAV logistics networks, transportation networks, and synthetic graphs. The experiments compare runtime and objective quality, isolate component contributions, and track DAD's optimization dynamics. Appendix E further reports CDPG performance under varying source-set sizes, routing distance weight scales, and details the parameter analysis.

6.1 Experimental Setup

Dataset Introduction: We evaluate CDPG on 1,487 Cost–Distance instances in total, covering multiple UAV logistics networks, transportation networks, and synthetic graphs.

- **GBA-CDNet** is a UAV logistics network benchmark for the Guangdong–Hong Kong–Macao Greater Bay Area. It uses logistics locations and road-based travel distances queried from Tencent Location Service, and contributes 9 large, dense UAV-network instances.
- **TN-CDNet** converts open-source Transportation Networks for Research into Cost–Distance instances; our benchmark uses 8 different medium-to-large sparse regions, and each region contributes multiple sink-centered instances.
- **CD-RandNet** contributes 1,000 small-to-medium synthetic instances generated from: Erdos–Renyi random graphs (ER) [31], Watts–Strogatz small-world graphs (WS) [32], stochastic block model graphs (SBM) [33], and Barabasi–Albert preferential-attachment graphs (BA) [34].

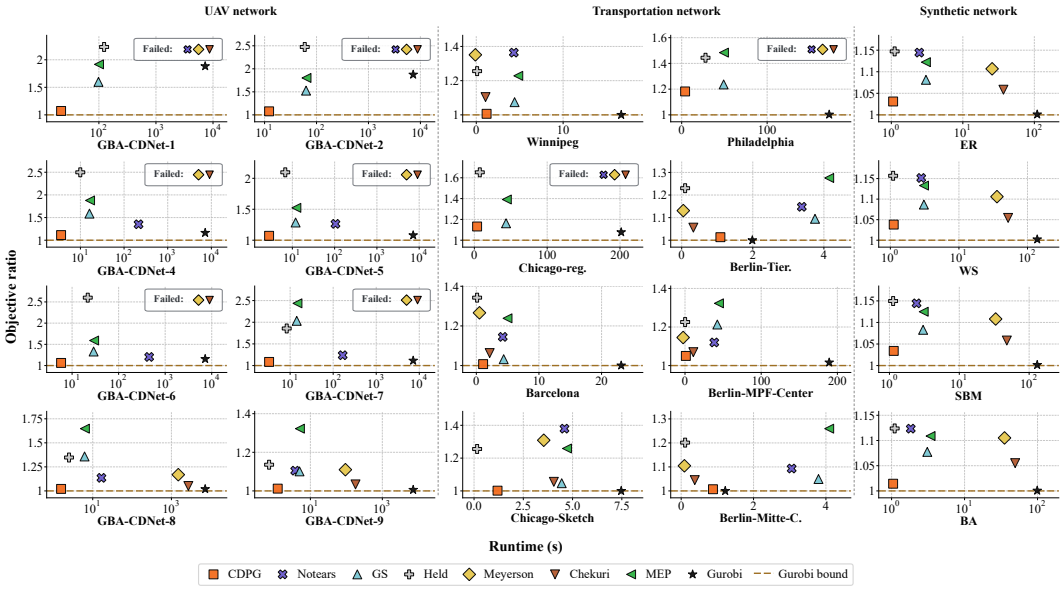

Figure 2: Runtime–quality comparison of CDPG and compared methods. Objectives are normalized by the Gurobi lower bound. Failed runs exceed the shared time limit for their dataset. Lower objective and runtime values are better.

TN-CDNet's 478 instances vary roots and demands on shared regional road graphs. Each synthetic family spans five sizes from 200 to 2,000 nodes. Appendix C gives the construction rules, instance counts, and weight synthesis details.

Evaluation Metrics and Baselines: We assess performance using the Cost-Distance objective $\mathcal{J}$ and runtime, with results averaged over 20 runs. CDPG is compared with three baseline categories:

- **Gradient-Based Relaxation** We compare with Notears [18], which enforces acyclicity with a continuous penalty, and sampling-based relaxations including Gumbel Softmax (GS) [17] and Maximum Entropy Policy (MEP) [35]. These baselines differ from CDPG in constraint handling and in whether decisions are deterministically softened or stochastically sampled.
- **Cost–Distance-Specific Algorithms** We compare with Held Routing [15], Meyerson Baseline [1], and Chekuri Baseline [14]. Held Routing runs in $\mathcal{O}(|S|(m + n \log n))$, while the hierarchical aggregation baselines require $\mathcal{O}(|S|^2(m + n \log n))$.
- **Commercial Solver** The Gurobi Baseline formulates Cost–Distance as a mixed-integer linear program and solves it with Gurobi Optimizer through `gurobipy` 13.0.1 [36]. Gurobi can return a lower bound (equal to the optimal value only when Gurobi certifies optimality within the prescribed time budgets). We use the lower bound as the Gurobi lower bound for normalization.

For the main comparison, we evaluate the final discrete network using $\mathcal{J}(X')$: each selected arc is charged once, and routing distance is weighted by source demand. All methods on an instance share the same normalization bound; the plotted Gurobi point is its best feasible incumbent.

Implementations: In CDPG, the practical optimizer is terminated at iteration T when $|\mathcal{J}^{(T)} - \mathcal{J}^{(T-1)}|/|\mathcal{J}^{(T-1)}| < 10^{-5}$. This objective-change rule is an implementation proxy; Theorem 5.3 analyzes the corresponding normalized stationarity criterion for gradient descent. All adopted hyperparameters are selected once from the parameter analysis on GBA-CDNet-3 in Appendix E.5 and then used unchanged for every instance: value-solver depth $d = \lceil \log_2 n \rceil$, DAD refresh interval 10 (refresh every 10 optimizer iterations and accept only if $\mathcal{J}_d(\theta^+; \widehat{A}) \leq \mathcal{J}_d(\theta^+; A)$), node-embedding rank $l = 12$, learning rate $\eta = 0.06$, and the RMSprop optimizer. All experiments were conducted using the GPU: RTX 5090 and the CPU: 25 vCPU Intel(R) Xeon(R) Platinum 8470Q.

All methods receive the same per-instance time limit within each dataset: 7200 seconds on GBA-CDNet and 200 seconds on TN-CDNet and CD-RandNet. The longer GBA limit accommodates its dense graphs and slower baselines; the shorter limit keeps evaluation of the other 1478 instances computationally manageable. Runs exceeding the limit are marked as failed.

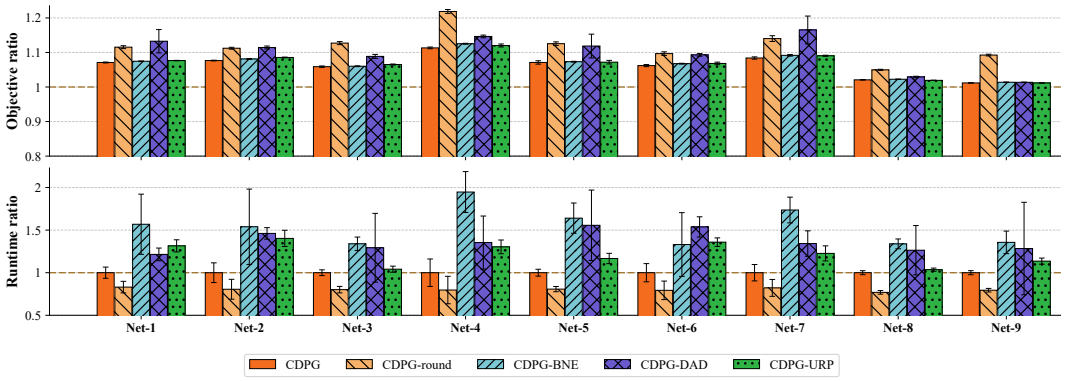

Figure 3: Ablation study on GBA-CDNet. The upper panel reports objective values normalized by the Gurobi bound, and the lower panel reports runtime normalized by CDPG time; lower is better.

6.2 Overall Performance

Figure 2 summarizes runtime–quality trade-offs across all benchmark families; lower and further left is better. The numerical values underlying Figure 2 are reported in Appendix Tables 3 and 4. GBA-CDNet-3 is not shown in Figure 2 because it is reserved for hyperparameter selection.

The largest application instance, GBA-CDNet-1, has $17{,}408$ nodes, $1{,}137{,}357$ directed arcs, and $16{,}440$ sources. CDPG solves it in 21.75 seconds with an objective ratio of 1.071 to the Gurobi lower bound. Held requires 123.38 seconds and has a ratio of 2.130 on this instance, so CDPG is about $5.7\times$ faster and reduces the objective by 49.7%.

Across all 20 panels, CDPG obtains the best objective among non-commercial methods while staying in the low-runtime region. Its normalized objective remains close to the Gurobi lower bound: 1.012–1.113 on UAV networks, 1.002–1.182 on transportation networks, and 1.014–1.038 on synthetic graph families. On Winnipeg, Held runs in 0.10 seconds compared with CDPG's 1.19, while the objective ratios are 1.223 and 1.006, respectively. On ER, WS, SBM, and BA graphs, CDPG remains close to the bound across all four topology families.

Gurobi supplies the normalization bound; on large dense instances it often exhausts the solver time budget and returns worse incumbents than CDPG. The source-size sweep in Appendix E.3 further separates sparse- and dense-demand regimes: Held is competitive with few sources, while CDPG improves the objective in 59 of 64 settings with $0.1|S|$–$0.8|S|$ active sources and its runtime remains nearly flat as the source set grows.

Reweighting routing distance by factors from 0.1 to 10 leaves topology, sources, and construction costs fixed (Appendix E.4). CDPG improves on Held in 52 of 56 graph–scale pairs; the mean CDPG/Held objective ratio decreases from 0.961 to 0.733 between these endpoints.

6.3 Ablation Study

Figure 3 compares full CDPG with four component variants on the nine UAV-network instances. CDPG-round evaluates the learned relaxed policy without applying TopoRounding, CDPG-DAD removes the accepted dynamic acyclic support update, CDPG-BNE removes the bilinear node embedding and keeps only direct edge logits, and CDPG-URP removes the unreached-flow penalty used by the truncated value solver.

CDPG-round has slightly lower runtime and consistently the worst objective. This indicates that TopoRounding efficiently converts the relaxed policy into a higher-quality hard arborescence. The DAD ablation causes the largest objective degradation among the structural variants, showing the benefit of adapting feasible parent sets as the policy evolves. Removing BNE has a smaller effect on objective quality, showing the benefit of node-shared bilinear structure alongside direct edge logits. Removing URP keeps objectives close to CDPG but increases measured runtime.

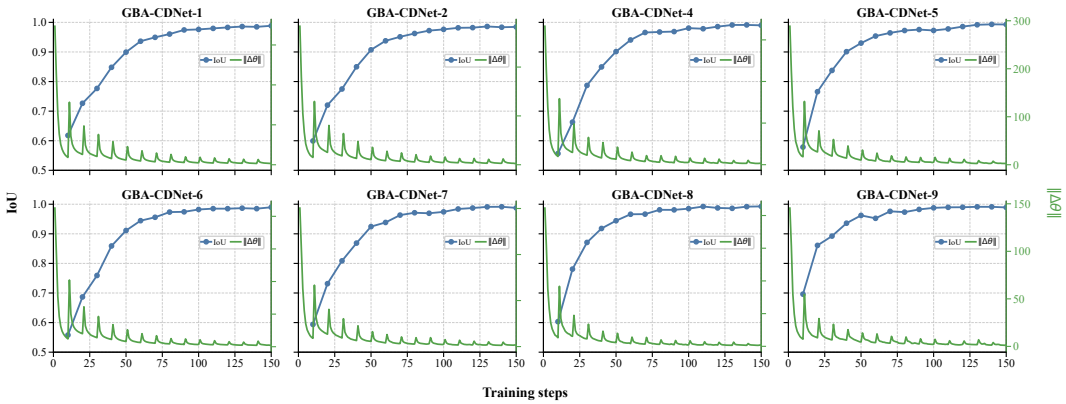

Figure 4: Optimization dynamics under DAD on GBA-CDNet. Structural consistency is measured by the IoU, and the parameter update magnitude $\|\Delta\theta\|$ is shown on the right y-axis.

6.4 Effect of DAD on Optimization

We track consecutive accepted acyclic structures on eight GBA-CDNet instances to inspect how DAD changes the feasible support during training. Structural stability is measured by Intersection-over-Union (IoU) [37], $\mathrm{IoU} = |A^t \cap A^{t-1}|/|A^t \cup A^{t-1}|$, where A^t and A^{t-1} are consecutive accepted DAD structures.

Figure 4 shows a consistent pattern across dense UAV networks. DAD first changes the support substantially, which allows the policy to escape poor initial parent choices and search over root-oriented acyclic supports aligned with the current Cost–Distance trade-off. As training proceeds, the IoU curves increase and saturate near one, meaning that consecutive masks become highly overlapping and the structural search stabilizes. The parameter update magnitude $\|\Delta\theta\| = \|\theta_{t+1} - \theta_t\|$ also exhibits larger corrections early and then decays to small oscillations.

A separate refresh-interval sweep on the eight GBA test graphs uses three seeds per setting. For intervals $q \in \{5, 10, 20, 50\}$, mean objective ratios relative to $q = 10$ remain within 0.9977–1.0066, and runtime ratios within 0.9635–1.0814. At $q = 1$, mean objective and runtime rise by 3.16% and 56.53%, respectively; $q = 2$ also increases both. These results support a moderate refresh rate that balances structural guidance and reconstruction overhead; Appendix E.7 reports the full sweep.

6.5 Scalability

We further evaluate ER and BA graphs with 5k, 10k, 20k, and 50k nodes, using five independently generated instances per setting. At 50k nodes, the graphs contain approximately two million directed arcs and 35k sources. Ratios are computed on each instance before averaging over the five matched comparisons. As graph size grows from 5k to 50k, CDPG's runtime ratios relative to Held decrease from 0.584 to 0.241 on ER and from 0.614 to 0.246 on BA. The corresponding objective ratios also decrease, from 0.985 to 0.930 and from 0.996 to 0.980, respectively. At 50k nodes, GS has runtime ratios 1.222 (ER) and 1.247 (BA), with objective ratios 0.988 and 1.027, respectively. Appendix E.6 gives all paired-normalized results. The same configuration is used across settings; adaptation consists of re-optimizing the policy on each graph.

7 Conclusion

CDPG is an end-to-end differentiable framework for Cost–Distance: it optimizes a relaxed next-hop policy, evaluates routing with a differentiable value solver, keeps supports acyclic through DAD, and recovers a hard arborescence with TopoRounding. The theory gives finite value computation, controlled truncation error, and fixed-accuracy complexity $\mathcal{O}(m \log n)$ under the stated conditions. Across 1,487 dense UAV, sparse transportation, and synthetic instances, CDPG gives the strongest non-commercial runtime–quality trade-off, and the ablations and appendix studies confirm the stability of DAD, routing-aware rounding, and the fixed hyperparameter setting.

Acknowledgments and Disclosure of Funding

This work was supported by National Natural Science Foundation of China (No.62472428), Public Computing Cloud, Renmin University of China, the fund for building world-class universities (disciplines) of Renmin University of China, and Qiushi Academic Project of Renmin University of China (NO.RUC26QSDL126).

The authors declare no competing interests.

References

[1] Adam Meyerson, Kamesh Munagala, and Serge Plotkin. Cost-distance: Two metric network design. *SIAM Journal on Computing*, 38(4):1648–1659, 2008.

[2] Francisco Saldanha da Gama. Facility location in logistics and transportation: An enduring relationship. *Transportation Research Part E: Logistics and Transportation Review*, 166:102903, 2022. ISSN 1366-5545.

[3] Hongyu Gong, Luoyi Fu, Xinzhe Fu, Lutian Zhao, Kainan Wang, and Xinbing Wang. Distributed multicast tree construction in wireless sensor networks. *IEEE Transactions on Information Theory*, 63(1):280–296, 2017.

[4] Chandra Chekuri and Rhea Jain. Approximation Algorithms for Hop Constrained and Buy-At-Bulk Network Design via Hop Constrained Oblivious Routing. In Timothy Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms (ESA 2024)*, volume 308 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 41:1–41:21, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-338-6.

[5] Nikolaos Karalias and Andreas Loukas. Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. *Advances in Neural Information Processing Systems*, 33:6659–6672, 2020.

[6] Haoyu Peter Wang, Nan Wu, Hang Yang, Cong Hao, and Pan Li. Unsupervised learning for combinatorial optimization with principled objective relaxation. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 31444–31458. Curran Associates, Inc., 2022.

[7] Sebastian Sanokowski, Sepp Hochreiter, and Sebastian Lehner. A diffusion model framework for unsupervised neural combinatorial optimization. In *Proceedings of the 41st International Conference on Machine Learning*, pages 43346–43367, 2024.

[8] Haoyu Peter Wang and Pan Li. Unsupervised learning for combinatorial optimization needs meta learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[9] Martin JA Schuetz, J Kyle Brubaker, and Helmut G Katzgraber. Combinatorial optimization with physics-inspired graph neural networks. *Nature Machine Intelligence*, 4(4):367–377, 2022.

[10] Wei Li, Rongjian Liang, Anthony Agnesina, Haoyu Yang, Chia-Tung Ho, Anand Rajaram, and Haoxing Ren. Dgr: Differentiable global router. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, DAC '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400706011.

[11] Mingju Liu, Yingjie Li, Jiaqi Yin, Zhiru Zhang, and CUNXI YU. Differentiable combinatorial scheduling at scale. In *Forty-first International Conference on Machine Learning*, 2024.

[12] Ismail Alkhouri, Cedric Le Denmat, Yingjie Li, CUNXI YU, Jia Liu, Rongrong Wang, and Alvaro Velasquez. Differentiable quadratic optimization for the maximum independent set problem. In *Forty-second International Conference on Machine Learning*, 2025.

[13] Alan Lahoud, Erik Schaffernicht, and Johannes Stork. Datasp: A differential all-to-all shortest path algorithm for learning costs and predicting paths with context. In *Uncertainty in Artificial Intelligence*, pages 2094–2112. PMLR, 2024.

[14] Chandra Chekuri, Sanjeev Khanna, and Joseph (Seffi) Naor. A deterministic algorithm for the cost-distance problem. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, page 232–233, USA, 2001. Society for Industrial and Applied Mathematics. ISBN 0898714907.

[15] Stephan Held and Edgar Perner. Cost-distance steiner trees for timing-constrained global routing. In *2025 62nd ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2025.

[16] Julia Chuzhoy, Anupam Gupta, Joseph (Seffi) Naor, and Amitabh Sinha. On the approximability of some network design problems. *ACM Trans. Algorithms*, 4(2), May 2008. ISSN 1549-6325.

[17] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *The fifth International Conference on Learning Representations*, 2017.

[18] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. Dags with no tears: Continuous optimization for structure learning. *Advances in neural information processing systems*, 31, 2018.

[19] Yue Yu, Jie Chen, Tian Gao, and Mo Yu. Dag-gnn: Dag structure learning with graph neural networks. In *International conference on machine learning*, pages 7154–7163. PMLR, 2019.

[20] Yue Yu, Tian Gao, Naiyu Yin, and Qiang Ji. Dags with no curl: An efficient dag structure learning approach. In *International conference on machine learning*, pages 12156–12166. Pmlr, 2021.

[21] Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. Learning latent permutations with gumbel-sinkhorn networks. In *International Conference on Learning Representations*, 2018.

[22] Yujia Xie, Hanjun Dai, Minshuo Chen, Bo Dai, Tuo Zhao, Hongyuan Zha, Wei Wei, and Tomas Pfister. Differentiable top-k with optimal transport. *Advances in neural information processing systems*, 33:20520–20531, 2020.

[23] Yimeng Min, Yiwei Bai, and Carla P Gomes. Unsupervised learning for solving the travelling salesman problem. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 47264–47278. Curran Associates, Inc., 2023.

[24] Yang Li, Jiale Ma, Wenzheng Pan, Runzhong Wang, Haoyu Geng, Nianzu Yang, and Junchi Yan. Unify ml4tsp: Drawing methodological principles for tsp and beyond from streamlined design space of learning and search. In *The Thirteenth International Conference on Learning Representations*, 2025.

[25] Wenzheng Pan, Hao Xiong, Jiale Ma, Wentao Zhao, Yang Li, and Junchi Yan. Unico: On unified combinatorial optimization via problem reduction to matrix-encoded general tsp. In *The Thirteenth International Conference on Learning Representations*, 2025.

[26] Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21188–21198, 2020.

[27] Haoyu Geng, Runzhong Wang, Fei Wu, and Junchi Yan. Gal-vne: Solving the vne problem with global reinforcement learning and local one-shot neural prediction. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 531–543, 2023.

[28] Yuan Jiang, Zhiguang Cao, Yaoxin Wu, Wen Song, and Jie Zhang. Ensemble-based deep reinforcement learning for vehicle routing problems under distribution shift. *Advances in Neural Information Processing Systems*, 36:53112–53125, 2023.

[29] Anne Greenbaum. *Iterative methods for solving linear systems*. SIAM, 1997.

[30] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[31] Paul Erdos and Alfred Renyi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960.

[32] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, 1998.

[33] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983.

[34] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[35] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

[36] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. Gurobi Optimization, LLC, 2026. URL `https://www.gurobi.com`.

[37] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 658–666, 2019.

[38] Qingyu Tan, Zhenkun Wang, Yew-Soon Ong, and Kin Huat Low. Evolutionary optimization-based mission planning for uas traffic management (utm). In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 952–958. IEEE, 2019.

[39] Kevin Dorling, Jordan Heinrichs, Geoffrey G. Messier, and Sebastian Magierowski. Vehicle routing problems for drone delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(1):70–85, 2017. doi: 10.1109/TSMC.2016.2582745.

A Proofs

A.1 Bellman Value Identity

Proof. Let $k(P) \coloneqq (P \odot D)\mathbf{1}_n$, so $k_i(P) = \sum_j P_{ij}D_{ij}$ is the expected one-step distance from v_i. For a fixed policy P, the Bellman equation with reward $-D$ is

$$\begin{aligned} V_{\gamma,i} &= \sum_j P_{ij}\big(-D_{ij} + \gamma V_{\gamma,j}\big) \\ &= -\sum_j P_{ij}D_{ij} + \gamma \sum_j P_{ij}V_{\gamma,j} \\ &= -k_i(P) + \gamma(P\mathbf{V}_\gamma)_i. \end{aligned} \tag{11}$$

Stacking (11) over all nodes gives

$$\begin{aligned} & \mathbf{V}_\gamma = -k(P) + \gamma P\mathbf{V}_\gamma \\ \iff \quad & (I - \gamma P)\mathbf{V}_\gamma = -(P \odot D)\mathbf{1}_n \\ \iff \quad & \mathbf{V}_\gamma = -(I - \gamma P)^{-1}(P \odot D)\mathbf{1}_n. \end{aligned} \tag{12}$$

The inverse exists on a DAD-supported policy because Proposition 4.2 proves $P^n = 0$. Let $S_n \coloneqq \sum_{t=0}^{n-1} P^t$. Then

$$\begin{aligned} (I - P)S_n = S_n(I - P) = \sum_{t=0}^{n-1} P^t - \sum_{t=1}^{n} P^t = I - P^n = I, \\ (I - P)^{-1} = S_n = \sum_{t=0}^{n-1} P^t. \end{aligned} \tag{13}$$

Substituting $\gamma = 1$ in (12) and using (13),

$$\begin{aligned} \mathbf{V}_1 &= -(I - P)^{-1}(P \odot D)\mathbf{1}_n \\ &= -\sum_{t=0}^{n-1} P^t(P \odot D)\mathbf{1}_n, \\ \mathbb{E}[\Omega_D] &= -w^\top \mathbf{V}_1 = w^\top \sum_{t=0}^{n-1} P^t(P \odot D)\mathbf{1}_n. \end{aligned} \tag{14}$$

□

A.2 Acyclicity and Finite Neumann Expansion

Proof. The DAD mask keeps only edges that strictly decrease the Φ-distance to the root. Therefore every positive transition in the normalized policy satisfies $P_{ij} > 0 \Rightarrow A_{ij} = 1 \Rightarrow L^X_\Phi(v_i, r) > L^X_\Phi(v_j, r)$. Let $\pi(1), \ldots, \pi(n)$ sort nodes by increasing $L^X_\Phi(\cdot, r)$, let Π be the corresponding permutation matrix, and define $\widetilde{P} = \Pi P \Pi^\top$. If $\widetilde{P}_{ab} > 0$, then $P_{\pi(a),\pi(b)} > 0$, so $L^X_\Phi(v_{\pi(a)}, r) > L^X_\Phi(v_{\pi(b)}, r)$ and therefore $a > b$. Thus $\widetilde{P}$ is strictly lower triangular. A standard induction for strictly lower-triangular matrices gives $(\widetilde{P}^t)_{ab} = 0$ whenever $b \geq a - t + 1$, $t = 1, \ldots, n$. For $t = n$, this condition holds for every valid pair (a, b). Hence $\widetilde{P}^n = 0$, $P^n = \Pi^\top \widetilde{P}^n \Pi = 0$, and $P^t = 0$ for all $t \geq n$. The Neumann expansion therefore terminates as

$$(I - P)^{-1} = \sum_{t=0}^{\infty} P^t = \sum_{t=0}^{n-1} P^t + \sum_{t=n}^{\infty} P^t = \sum_{t=0}^{n-1} P^t.$$

□

A.3 Support of the Fixed-W Optimum

Proof. Fix the current policy $\tilde{P}$. The old cumulative node-flow row vector is $w^\top (I - \tilde{P})^{-1}$, so

$$W \coloneqq \operatorname{diag}\Big(w^\top (I - \tilde{P})^{-1}\Big)\,\tilde{P}, \qquad W_{ij} = \big[w^\top (I - \tilde{P})^{-1}\big]_i \tilde{P}_{ij}.$$

For fixed W, DAD uses the positive edge metric $\Phi \coloneqq C \oslash (W + \varepsilon \mathbf{1}_{n\times n}) + D$, with $\Phi_{ij} > 0$. The fixed-W surrogate separates over sources:

$$\begin{aligned}\min_{\{P_{(s)}\}_{s\in S}} \mathcal{J}_{\text{fix}}(\{P_{(s)}\}; W) &\coloneqq \sum_{s\in S} w_s \operatorname{tr}\Big(P_{(s)}^\top \Phi\Big) \\ &= \sum_{s\in S} w_s \sum_{(i,j)\in \text{path}(s,r)} \Phi_{ij}. \end{aligned} \tag{15}$$

Because (15) is a nonnegative weighted sum of independent source-to-root path costs, an optimum routes each source along a Φ-shortest path.

Let $\ell_i^\Phi = L_\Phi^X(v_i, r)$. If edge $i \to j$ lies on a Φ-shortest source-to-root path, the suffix from j to r is also shortest, so $\ell_i^\Phi = \Phi_{ij} + \ell_j^\Phi > \ell_j^\Phi$. By the definition of B in (9), this implies $B_{ij} = 1$. Since the path starts from a source, every tail node v_i on that path is in $\mathcal{R}(B)$, and therefore $A_{ij} = \mathbb{1}\{v_i \in \mathcal{R}(B)\}B_{ij} = 1$. Thus every edge used by any fixed-W optimum is retained. If P^Φ denotes the aggregate support of such an optimum, then $P_{ij}^\Phi > 0 \Rightarrow A_{ij} = 1$, and consequently

$$\operatorname{tr}\big((\mathbf{1}_{n\times n} - A)^\top P^\Phi\big) = \sum_{i,j} (1 - A_{ij}) P_{ij}^\Phi = 0.$$

□

A.4 TopoRounding Bound

Proof. Fix the terminal DAG A^T and the final normalized policy $P = f(\theta^*; A^T)$. Let $\mathcal{I}_A = \{i \neq r : \sum_j A_{ij}^T > 0\}$ be the active rows. Since A^T is acyclic, the exact value vector is computed by the reverse topological recursion

$$V_{1,r} = 0, \qquad V_{1,i} = \sum_{j: A_{ij}^T = 1} P_{ij}\big(V_{1,j} - D_{ij}\big), \quad i \in \mathcal{I}_A. \tag{16}$$

Equivalently, $-V_{1,i}$ is the expected remaining routing distance from i.

Initialize the hard flow by $\widehat{w}_i \leftarrow w_i$. Consider an augmented random completion that independently chooses one parent $J_i \sim P_{i:}$ for every $i \in \mathcal{I}_A$, including rows that may later receive zero hard flow. Its expected construction cost and routing cost are

$$\begin{aligned}\mathbb{E}[\Omega_C^{\text{aug}}] &= \sum_{i\in\mathcal{I}_A} \sum_{j: A_{ij}^T = 1} P_{ij} C_{ij} = \operatorname{tr}(P^\top C), \\ \mathbb{E}[\Omega_D^{\text{aug}}] &= -w^\top \mathbf{V}_1, \end{aligned} \tag{17}$$

so $\mathbb{E}[\mathcal{J}^{\text{aug}}] = \operatorname{tr}(P^\top C) - w^\top \mathbf{V}_1 = \mathcal{J}(P)$. The actual TopoRounding output deletes zero-flow rows, and therefore its objective is no larger than the augmented completion objective obtained with the same positive-flow choices.

Let $\mathcal{F}_t$ be the choices fixed after the first t positive-flow queue pops. When node i is the next positive-flow pop, all hard flow entering i has already been accumulated by the topological order, so $\widehat{w}_i$ is fixed under $\mathcal{F}_t$. If parent j is chosen, the only parent-dependent term in the conditional augmented objective is $g_i(j; \widehat{w}_i) \coloneqq C_{ij} + \widehat{w}_i(D_{ij} - V_{1,j})$, where D_{ij} is the distance to the parent and $-V_{1,j}$ is the expected remaining distance after reaching j. Thus, for a term B_t independent of the next parent,

$$\mathcal{E}_t \coloneqq \mathbb{E}[\mathcal{J}^{\text{aug}} \mid \mathcal{F}_t] = B_t + \sum_{j: A_{ij}^T = 1} P_{ij} g_i(j; \widehat{w}_i).$$

Since $P_{i:}$ is a distribution on the same feasible parent set, the minimizer chosen by TopoRounding satisfies

$$\mathcal{E}_{t+1} = B_t + \min_{j:A^T_{ij}=1} g_i(j;\widehat{w}_i) \le B_t + \sum_{j:A^T_{ij}=1} P_{ij} g_i(j;\widehat{w}_i) = \mathcal{E}_t.$$

Zero-flow pops add no hard edge and do not increase the actual objective. Iterating the conditional-expectation inequality over all positive-flow pops gives $\mathcal{J}(X^{\mathrm{topo}}) \le \mathbb{E}[\mathcal{J}^{\mathrm{aug}}] = \mathrm{tr}(P^\top C) - w^\top \mathbf{V}_1 = \mathcal{J}(P)$. □

A.5 Smoothness Bound

Proof. Fix an acyclic mask A, and let $\mathcal{P}_A$ be the set of row-normalized policies supported on $X \odot A$. Write

$$L_A \coloneqq L_D^{X\odot A}(\mathcal{V}, r), \quad k(P) \coloneqq (P \odot D)\mathbf{1}_n, \quad \mathcal{J}_d(P) \coloneqq \mathrm{tr}(P^\top C) + R_d(P) + M_d(P),$$

where $R_d(P) \coloneqq w^\top \sum_{t=0}^{d-1} P^t k(P)$ and $M_d(P) \coloneqq w^\top P^d L_A$. The construction term is linear in P, so only R_d and M_d contribute to the Lipschitz constant of $\nabla_P \mathcal{J}_d$. Define

$$\beta_d \coloneqq \sup_{P\in\mathcal{P}_A} \max_{0\le t\le d} \|P^t\|_2, \qquad \delta \coloneqq \|P_1 - P_2\|_F.$$

Since $P^0 = I$, $\beta_d \ge 1$. For $1 \le t \le d$,

$$\begin{aligned} P_1^t - P_2^t &= \sum_{a=0}^{t-1} P_1^a (P_1 - P_2) P_2^{t-1-a}, \\ \|P_1^t - P_2^t\|_F &\le \sum_{a=0}^{t-1} \|P_1^a\|_2 \, \delta \, \|P_2^{t-1-a}\|_2 \\ &\le t\beta_d^2 \delta. \end{aligned} \tag{18}$$

The same expansion gives the differential

$$\begin{aligned} \mathrm{D}(P^t)[H] &= \sum_{a=0}^{t-1} P^a H P^{t-1-a}, \\ \|\mathrm{D}(P^t)[H]\|_F &\le t\beta_d^2 \|H\|_F. \end{aligned} \tag{19}$$

For the terminal penalty,

$$\mathrm{D}M_d(P)[H] = w^\top \mathrm{D}(P^d)[H] L_A = \sum_{a=0}^{d-1} w^\top P^a H P^{d-1-a} L_A.$$

By duality, $\nabla_P M_d(P) = \sum_{a=0}^{d-1} (P^a)^\top w L_A^\top (P^{d-1-a})^\top$. Let $b = d-1-a$. Using (18),

$$\begin{aligned} &\|\nabla_P M_d(P_1) - \nabla_P M_d(P_2)\|_F \\ &\le \|w\|_2 \|L_A\|_2 \sum_{a=0}^{d-1} \Big(\|P_1^a - P_2^a\|_F \|P_1^b\|_2 + \|P_2^a\|_2 \|P_1^b - P_2^b\|_F \Big) \\ &\le \|w\|_2 \|L_A\|_2 \sum_{a=0}^{d-1} \big(a\beta_d^3 + b\beta_d^3 \big)\delta \\ &\le \|w\|_2 \|L_A\|_2 d^2 \beta_d^3 \delta. \end{aligned} \tag{20}$$

This is the conservative β_d^3 bound; it does not require $\beta_d \le 1$.

For the routing term, let $\mathcal{K}_D[H] \coloneqq (H \odot D)\mathbf{1}_n$,

$$\Lambda_D \coloneqq \sup_{\|H\|_F=1} \|\mathcal{K}_D[H]\|_2 \le \|D\|_F, \qquad K_{A,D} \coloneqq \sup_{P\in\mathcal{P}_A} \|k(P)\|_2 < \infty.$$

For $R_t(P) \coloneqq w^\top P^t k(P)$, the differential is

$$\begin{aligned} \mathrm{D}R_t(P)[H] &= w^\top \mathrm{D}(P^t)[H]k(P) + w^\top P^t \mathcal{K}_D[H] \\ &= w^\top \sum_{a=0}^{t-1} P^a H P^{t-1-a} k(P) + w^\top P^t \mathcal{K}_D[H]. \end{aligned} \tag{21}$$

The first part of (21) is controlled exactly as in (20), with $K_{A,D}$ replacing $\|L_A\|_2$. The second part varies only through P^t, and (18) gives

$$\|\nabla_P R_t(P_1) - \nabla_P R_t(P_2)\|_F \le \|w\|_2 \left(t^2 \beta_d^3 K_{A,D} + (2t+1)\beta_d^3 \Lambda_D\right) \delta. \tag{22}$$

Summing (22) over $t = 0, \dots, d-1$,

$$\begin{aligned} \|\nabla_P R_d(P_1) - \nabla_P R_d(P_2)\|_F &\le \|w\|_2 \left(K_{A,D}\beta_d^3 \sum_{t=0}^{d-1} t^2 + \Lambda_D \beta_d^3 \sum_{t=0}^{d-1} (2t+1) \right) \delta \\ &= \|w\|_2 \left(K_{A,D}\beta_d^3 \frac{d(d-1)(2d-1)}{6} + \Lambda_D \beta_d^3 d^2 \right) \delta. \end{aligned} \tag{23}$$

Combining (20) and (23),

$$\begin{aligned} &\|\nabla_P \mathcal{J}_d(P_1) - \nabla_P \mathcal{J}_d(P_2)\|_F \le L_{P,d} \|P_1 - P_2\|_F, \\ &L_{P,d} \coloneqq \|w\|_2 \left[\|L_A\|_2 d^2 \beta_d^3 + K_{A,D}\beta_d^3 \frac{d(d-1)(2d-1)}{6} + \Lambda_D \beta_d^3 d^2 \right]. \end{aligned} \tag{24}$$

Since $\mathcal{P}_A$ is compact and $\mathcal{J}_d$ is a finite polynomial in P, the policy-space gradient is bounded by $G_{P,d} \coloneqq \sup_{P \in \mathcal{P}_A} \|\nabla_P \mathcal{J}_d(P)\|_F < \infty$.

Now compose with the fixed-mask encoding $P = f_A(\theta)$. On the bounded parameter domain, let B_1 bound the operator norm of the Jacobian $\mathrm{D}f_A$, and let B_2 bound the Lipschitz constant of this Jacobian. The chain rule yields

$$\begin{aligned} &\|\nabla_\theta (\mathcal{J}_d \circ f_A)(\theta_1) - \nabla_\theta (\mathcal{J}_d \circ f_A)(\theta_2)\|_F \\ &\le B_1^2 L_{P,d} \|\theta_1 - \theta_2\|_F + B_2 G_{P,d} \|\theta_1 - \theta_2\|_F \\ &= \left(B_1^2 L_{P,d} + B_2 G_{P,d}\right) \|\theta_1 - \theta_2\|_F. \end{aligned} \tag{25}$$

For the sparse score map $z_{ij} = h_{ij} + l^{-1/2} q_i^\top k_j$, $(i,j) \in \mathcal{E}$, the logits are materialized only on the edge list. On bounded parameter sets, this bilinear edge-list map and the row-wise softmax have bounded first derivatives and Lipschitz Jacobians; these constants are absorbed into B_1 and B_2. Thus $F_A(\theta) = \mathcal{J}_d(f_A(\theta))$ is $L_{\theta,d}(A)$-smooth with $L_{\theta,d}(A) \coloneqq B_1^2 L_{P,d} + B_2 G_{P,d}$. □

A.6 Estimation Relative Error

Proof. Let

$$k(P) = (P \odot D)\mathbf{1}_n, \quad L_A = L_D^{X \odot A}(\mathcal{V}, r), \quad V = \sum_{t=0}^{n-1} P^t k(P),$$

and define $\|D\|_{\min} = \min_{X_{ij}A_{ij}=1} D_{ij}$, $\|D\|_{\max} = \max_{X_{ij}A_{ij}=1} D_{ij}$. If $d \ge n$, nilpotency gives zero truncation error, so assume $d < n$. By Proposition 4.2,

$$\begin{aligned} \mathbb{E}[\Omega_D] &= w^\top V = w^\top \sum_{t=0}^{n-1} P^t k(P), \\ \mathbb{E}[\Omega_D^{\mathrm{eval}}] &= w^\top \sum_{t=0}^{d-1} P^t k(P) + w^\top P^d L_A. \end{aligned} \tag{26}$$

The terminal vector L_A is the shortest remaining distance on the same acyclic support, whereas V is the policy-induced remaining distance. Hence $0 \le L_A \le V$. Also,

$$\begin{aligned} V &= \sum_{t=0}^{d-1} P^t k(P) + P^d V \qquad , \\ \Delta_d &\coloneqq \mathbb{E}[\Omega_D] - \mathbb{E}[\Omega_D^{\text{eval}}] \\ &= w^\top P^d V - w^\top P^d L_A \qquad \text{by (26)} \\ &= w^\top P^d (V - L_A) \ge 0. \end{aligned} \tag{27}$$

Every active root-reaching path has at most $n-1$ edges, so $V_i \le n\|D\|_{\max}$. Each positive-flow source is non-root and must pay at least one positive supported edge to reach r, so $L_{A,i} \ge \|D\|_{\min}$ on source nodes. Therefore $V \le n\|D\|_{\max}\mathbf{1}_n$ and $w^\top V \ge w^\top L_A \ge \|w\|_1\|D\|_{\min}$. Combining these bounds with (27),

$$\begin{aligned} \frac{\Delta_d}{\mathbb{E}[\Omega_D]} &= \frac{w^\top P^d (V - L_A)}{w^\top V} \\ &\le \frac{w^\top P^d V}{w^\top L_A} \qquad \text{since } 0 \le L_A \le V \\ &\le \frac{n\|D\|_{\max}\, w^\top P^d \mathbf{1}_n}{\|D\|_{\min}\|w\|_1} \\ &= n\, r_d\, \frac{\|D\|_{\max}}{\|D\|_{\min}}, \qquad r_d \coloneqq \frac{w^\top P^d \mathbf{1}_n}{\|w\|_1}. \end{aligned} \tag{28}$$

Finally let $\chi_D = \|D\|_{\max}/\|D\|_{\min}$. If $r_d \le \rho_{\text{res}}^d$ for some $\rho_{\text{res}} \in (0,1)$, then any constant relative tolerance $\varepsilon > 0$ is ensured by

$$\begin{aligned} & n\rho_{\text{res}}^d \chi_D \le \varepsilon \\ \iff \quad & d \ge \log_{1/\rho_{\text{res}}}\left(\frac{n\chi_D}{\varepsilon}\right) = \mathcal{O}(\log(n\chi_D)). \end{aligned} \tag{29}$$

If $\chi_D \le n^{\mathcal{O}(1)}$, this becomes $d = \mathcal{O}(\log n)$. □

A.7 Convergence Rate

Proof. Let $F_t(\theta) = \mathcal{J}_d(\theta; A^t)$, $g_t = \nabla F_t(\theta_t)$, and $L = \sup_{t<T} L_{\theta,d}(A^t)$. Theorem 5.1 makes every accepted fixed-mask surrogate L-smooth. For $\theta_{t+1} = \theta_t - \eta g_t$,

$$\begin{aligned} F_t(\theta_{t+1}) &\le F_t(\theta_t) + \langle g_t, \theta_{t+1} - \theta_t \rangle + \frac{L}{2}\|\theta_{t+1} - \theta_t\|^2 \\ &= F_t(\theta_t) - \eta\|g_t\|^2 + \frac{L\eta^2}{2}\|g_t\|^2 \\ &= F_t(\theta_t) - \eta\left(1 - \frac{L\eta}{2}\right)\|g_t\|^2. \end{aligned} \tag{30}$$

The DAD acceptance rule evaluates the old and candidate masks at the same parameter θ_{t+1}, and accepts only if $F_{t+1}(\theta_{t+1}) \le F_t(\theta_{t+1})$. Combining this accepted-mask inequality with (30),

$$F_{t+1}(\theta_{t+1}) \le F_t(\theta_t) - \eta\left(1 - \frac{L\eta}{2}\right)\|g_t\|^2. \tag{31}$$

Summing (31) over $t = 0, \ldots, T-1$ telescopes:

$$\begin{aligned} \eta\left(1 - \frac{L\eta}{2}\right)\sum_{t=0}^{T-1}\|\nabla F_t(\theta_t)\|^2 &\le F_0(\theta_0) - F_T(\theta_T) \\ &\le F_0(\theta_0) - F_{\text{inf}} =: \Delta_0. \end{aligned} \tag{32}$$

Set $\eta = \kappa/L$ with $0 < \kappa \leq 1$. Dividing (32) by T gives

$$\begin{aligned} \min_{0\leq t<T} \|\nabla F_t(\theta_t)\|^2 &\leq \frac{L\Delta_0}{\kappa(1-\kappa/2)T}, \\ \frac{\min_{0\leq t<T} \|\nabla F_t(\theta_t)\|^2}{L\Delta_0} &\leq \frac{1}{\kappa(1-\kappa/2)T}. \end{aligned} \tag{33}$$

Thus the relative stationarity condition $\min_{t<T} \|\nabla F_t(\theta_t)\|^2 \leq \varepsilon_{\mathrm{rel}}^2 L\Delta_0$ is reached whenever $T_{\mathrm{rel}} \geq [\kappa(1-\kappa/2)\varepsilon_{\mathrm{rel}}^2]^{-1}$. For fixed κ and $\varepsilon_{\mathrm{rel}}$, this is $\mathcal{O}(1)$. □

A.8 Overall Computational Complexity

Proof. Use sparse adjacency-list storage. Sparse edge-logit generation costs $\mathcal{O}(ml)$, and masking, row softmax on supported edges, sparse vector propagation, and element-wise sparse operations each cost $\mathcal{O}(m)$. Because the accepted mask is acyclic, cumulative flow $w^\top(I-P)^{-1}$ can be computed by one topological propagation on the DAG, also in $\mathcal{O}(m)$. A Dijkstra pass on the original sparse graph costs $\mathcal{O}(m + n\log n)$. Thus one iteration has

$$\begin{aligned} C_{\mathrm{DAD}} &= \mathcal{O}(m) + \mathcal{O}(m + n\log n) = \mathcal{O}(m + n\log n), \\ C_{\mathrm{forward}} &= \mathcal{O}(ml) + d\,\mathcal{O}(m) + \mathcal{O}(m) = \mathcal{O}(dm) \quad \text{for fixed } l, \\ C_{\mathrm{backward}} &= d\,\mathcal{O}(m) + \mathcal{O}(ml) = \mathcal{O}(dm) \quad \text{for fixed } l, \\ C_{\mathrm{accept}} &= C_{\mathrm{DAD}} + 2C_{\mathrm{forward}} = \mathcal{O}(m + n\log n + dm), \\ C_{\mathrm{round}} &= \mathcal{O}(m). \end{aligned} \tag{34}$$

Here C_{forward} includes sparse edge-logit generation, masking, d sparse propagations, and the terminal penalty; C_{accept} includes the candidate-mask construction and the two value evaluations used by the accept/reject test; C_{round} includes the exact reverse-topological value pass and the TopoRounding scan.

Under the logarithmic-depth condition in Theorem 5.2, take $d = \mathcal{O}(\log n)$. For root-reachable directed graphs, $n = \mathcal{O}(m)$, so

$$\begin{aligned} C_{\mathrm{total}} &= \mathcal{O}\big(T_{\mathrm{rel}}\big(C_{\mathrm{accept}} + C_{\mathrm{forward}} + C_{\mathrm{backward}}\big) + C_{\mathrm{round}}\big) \\ &= \mathcal{O}(T_{\mathrm{rel}}(dm + m + n\log n) + m) \\ &= \mathcal{O}(T_{\mathrm{rel}} m\log n + T_{\mathrm{rel}} n\log n + m) \\ &= \mathcal{O}(T_{\mathrm{rel}} m\log n). \end{aligned} \tag{35}$$

With fixed relative stationarity tolerance, Theorem 5.3 gives $T_{\mathrm{rel}} = \mathcal{O}(1)$, and therefore CDPG has fixed-accuracy complexity $\mathcal{O}(m\log n)$. The dominant edge-logit generation, sparse propagation, and element-wise operations are parallelizable on GPUs. □

B Overall Learning Process of CDPG

Algorithm 2 gives the complete CDPG pipeline, and Algorithm 1 is the differentiable value solver called inside each training step. CDPG starts from the input graph X, edge construction costs C, edge routing distances D, and node weights w. It first initializes the sparse policy parameters $\theta = \{h, Q, K\}$, sets the value-solver depth $d = \lceil \log_2 n \rceil$, and builds the initial DAD mask A^0 from a warm-start flow estimate. This mask removes policy support that would violate the root-oriented acyclic structure, while keeping source-reachable edges that can still contribute to the routing objective. At iteration t, CDPG forms edge-list logits $z_{ij} = h_{ij} + l^{-1/2} q_i^\top k_j$ for $(i,j) \in \mathcal{E}$, applies the DAD mask, and normalizes outgoing feasible edges with edge-softmax. The relaxed Cost–Distance objective is then evaluated as the sum of the expected construction cost and the routing distance term. The routing distance term is computed by Algorithm 1: it propagates the residual node weights through $P^\top$ for d steps, accumulates the expected one-step distances, and finally attaches the residual mass to the exact shortest-path distance on the current DAD support. After back-propagating this objective, CDPG updates the policy parameters, recomputes an edge-flow-aware score Φ, proposes a refreshed DAD mask, and accepts this mask only when it does not worsen the current surrogate objective. Once the objective-change stopping rule is met, CDPG computes the final policy on the accepted DAD support and applies TopoRounding to obtain the discrete edge set X'.

Algorithm 1 Routing Distance Estimation

Input: policy matrix P, routing distance D, node weight w, distance iteration d, terminal distance vector $L_D^{X\odot A}(\mathcal{V}, r)$

Output: $\mathbb{E}[\Omega_D^{\text{eval}}]$

Initialize $\mathbb{E}[\Omega_D^{\text{eval}}] \leftarrow 0$, $q \leftarrow w$, $k \leftarrow (P \odot D) \cdot \mathbf{1}_n$, $i \leftarrow 1$
while $i \leq d$ **do**
  $\mathbb{E}[\Omega_D^{\text{eval}}] \leftarrow \mathbb{E}[\Omega_D^{\text{eval}}] + q^\top k$
  $q \leftarrow P^\top q$ (parallelized in $\mathcal{O}(m)$ time).
  $i \leftarrow i + 1$
end while
$\mathbb{E}[\Omega_D^{\text{eval}}] \leftarrow \mathbb{E}[\Omega_D^{\text{eval}}] + q^\top L_D^{X\odot A}(\mathcal{V}, r)$
return $\mathbb{E}[\Omega_D^{\text{eval}}]$

Table 1: Statistics of the 9 GBA-CDNet UAV-network subgraphs.

Statistic	Graph 1	Graph 2	Graph 3	Graph 4	Graph 5	Graph 6	Graph 7	Graph 8	Graph 9
n	17408	13687	2650	6145	5138	8834	5737	3126	1629
m	1137357	733838	84694	237055	172451	399774	185772	88638	41043
$\lvert S\rvert$	16440	12798	2346	5473	4489	8123	5204	2691	1174

C Details of Datasets

All datasets are converted into the same directed Cost–Distance instance format (G, r, S, C, D, w): $G = (V, E)$ is the directed candidate network, r is the root or destination center, $S = \{v : w_v > 0\}$ is the source set, C is the one-time construction cost on selected directed arcs, and D is the per-unit routing distance cost accumulated by source demand. This section gives the construction details omitted from the main text.

C.1 GBA-CDNet: UAV logistics Networks

GBA-CDNet is a UAV logistics benchmark for the Guangdong–Hong Kong–Macao Greater Bay Area. We construct the benchmark from two commercial data sources. First, Tencent Location Service[4] is used to obtain logistics-relevant candidate locations and processed inter-node travel-cost estimates. Second, aggregated mobile-signaling records provided by Smart Footprint Data Technology Co., Ltd. for August 2022 are used to derive mobility-based demand proxies for candidate nodes.

Due to privacy protection and commercial data-use restrictions, the raw mobile-signaling records are not publicly released. The released benchmark contains only the processed graph instances used in our experiments: anonymized node identifiers, normalized relative demand proportions, processed edge weights, root-node indicators, and subgraph membership. It does not include user identifiers, individual trajectories, OD-level signaling records, raw timestamps, exact geographic coordinates, raw API responses, or absolute population counts. Low-count or sparse mobility observations are suppressed before aggregation.

The resulting benchmark should therefore be interpreted as a privacy-preserving, processed graph benchmark derived from real-world mobility and travel-cost signals, rather than a release of the underlying proprietary data sources. After preprocessing, the network is partitioned into 9 subgraphs, each centered on a designated core node. Nodes are assigned to the geographically closest core, yielding a proximity-based decomposition. Table 1 summarizes the 9 UAV-network instances.

C.1.1 Spatial Sampling and Core Node Selection

We traverse and search locations with logistics attributes in the Greater Bay Area at a spatial accuracy of 0.25 square kilometers. To avoid excessive duplication in extremely dense urban blocks, only one point is retained within every 2500 square meters. Nine first-level core nodes are manually selected, and the remaining nodes are assigned to their nearest core to form the subgraphs.

[4] `https://lbs.qq.com/`

Algorithm 2 CDPG

Input: adjacency matrix X, construction cost C, routing distance D, root r, node weight w
Output: X'

Set source set $S \leftarrow \{v_i : w_i > 0,\ i \neq r\}$
Initialize policy parameters $\theta = \{h, Q, K\}$:
$h, Q, K \sim \mathcal{N}(0,1)$, optimizer $\mathrm{OPT} \leftarrow \mathrm{RMSprop}(\eta = 0.06)$, distance iteration $d \leftarrow \lceil \log_2 n \rceil$, node embedding rank $l \leftarrow 12$, DAD refresh interval $q_{\mathrm{DAD}} \leftarrow 10$, objective-change tolerance $\epsilon_{\mathrm{stop}} \leftarrow 10^{-5}$
Initialize A^0 with the DAD rule using warm-start edge flow $W^{(0)} = X$, $\Phi^{(0)} = C \oslash (W^{(0)} + \varepsilon \mathbf{1}_{n\times n}) + D$, all-node distances $L^X_{\Phi^{(0)}}(\cdot, r)$, and the source-reachable set in (9)
Compute $L_D^{X\odot A^0}(\mathcal{V}, r) \leftarrow$ Dijkstra(D, $X \odot A^0$)
Initialize $t \leftarrow 0$ and $\mathcal{J}_{\mathrm{prev}} \leftarrow +\infty$
while true do
  Generate sparse logits for $(i,j) \in \mathcal{E}$: $z_{ij} \leftarrow h_{ij} + l^{-1/2} q_i^\top k_j$ keep only entries with $A^t_{ij} = 1$
  Normalize z with edge-softmax using (7) to obtain P
  Compute construction cost: $\Omega_C \leftarrow \mathrm{tr}\left(P^\top C\right)$
  Compute routing distance term with Algorithm 1
  $\Omega_D \leftarrow \mathrm{RDE}(P, D, w, d, L_D^{X\odot A^t}(\mathcal{V}, r))$
  $\mathcal{J}^{(t)} \leftarrow \Omega_C + \Omega_D$
  if $t > 0$ and $\left|\mathcal{J}^{(t)} - \mathcal{J}_{\mathrm{prev}}\right| / |\mathcal{J}_{\mathrm{prev}}| < \epsilon_{\mathrm{stop}}$ **then**
    break
  end if
  Update policy parameters with RMSprop: $\theta^+ \leftarrow \mathrm{OPT}\left(\theta, \nabla_\theta \mathcal{J}^{(t)}(\theta)\right)$
  if $(t+1) \bmod q_{\mathrm{DAD}} = 0$ **then**
    Generate $P^+_{\mathrm{keep}} \leftarrow f(\theta^+; A^t)$
    Compute $W^+ \leftarrow \mathrm{diag}\Big(w^\top (I - P^+_{\mathrm{keep}})^{-1}\Big) P^+_{\mathrm{keep}}$ and $\Phi^+ \leftarrow C \oslash (W^+ + \varepsilon \mathbf{1}_{n\times n}) + D$
    Build $\widehat{A}^{t+1}$ from P^+_{keep} using all-node distances $L^X_{\Phi^+}(\cdot, r)$ and the source-reachable set (9)
    Compute $L_D^{X\odot \widehat{A}^{t+1}}(\mathcal{V}, r) \leftarrow$ Dijkstra(D, $X \odot \widehat{A}^{t+1}$)
    Evaluate $\mathcal{J}_{\mathrm{keep}} \leftarrow \mathcal{J}_d(\theta^+; A^t)$ and $\mathcal{J}_{\mathrm{cand}} \leftarrow \mathcal{J}_d(\theta^+; \widehat{A}^{t+1})$
    if $\mathcal{J}_{\mathrm{cand}} \leq \mathcal{J}_{\mathrm{keep}}$ **then**
      $A^{t+1} \leftarrow \widehat{A}^{t+1}, \quad L_D^{X\odot A^{t+1}}(\mathcal{V}, r) \leftarrow L_D^{X\odot \widehat{A}^{t+1}}(\mathcal{V}, r)$
    else
      $A^{t+1} \leftarrow A^t, \quad L_D^{X\odot A^{t+1}}(\mathcal{V}, r) \leftarrow L_D^{X\odot A^t}(\mathcal{V}, r)$
    end if
  else
    $A^{t+1} \leftarrow A^t, \quad L_D^{X\odot A^{t+1}}(\mathcal{V}, r) \leftarrow L_D^{X\odot A^t}(\mathcal{V}, r)$
  end if
  $\theta \leftarrow \theta^+, \quad \mathcal{J}_{\mathrm{prev}} \leftarrow \mathcal{J}^{(t)}, \quad t \leftarrow t+1$
end while
Generate final policy $P \leftarrow f(\theta; A^t)$ using edge-softmax normalization
Compute exact $\mathbf{V}_1$ on A^t by the reverse topological pass in (16)
Generate X' from P using zero-indegree TopoRounding from (10)
return X'

C.1.2 Node Weights

Mobile signaling data is commonly used to estimate spatial demand. We assign each network node a standardized service area of 1 km^2. In high-density urban zones where multiple nodes fall within a 1000 m^2 region, the total service demand of that region is evenly distributed among all nodes located in it. Nodes with positive demand are sources; zero-demand nodes can serve as Steiner relay nodes.

C.1.3 Routing Distance and Construction Cost

In highly urbanized areas, UAV paths are often planned above existing urban roads [38]. We therefore use road-based walking distance as the routing distance proxy in GBA-CDNet. For primary nodes,

distances to all other nodes are queried directly. For the remaining candidate edges, distances are queried in increasing straight-line-distance bands of 500 m, 1000 m, and 1500 m; the final network keeps edges whose endpoints are within 2000 m in straight-line distance.

The construction cost is proportional to path length with a constant coefficient 0.1. The routing distance calibration follows standard UAV delivery cost modeling practice that accounts for flight distance [39].

C.2 TN-CDNet: Open Transportation Networks

TN-CDNet is constructed from the open-source Transportation Networks for Research repository[5], a widely used collection of TNTP traffic-assignment networks. A TNTP network file describes directed road arcs with attributes such as start node, end node, capacity, length, free-flow time, and BPR congestion-function parameters. A demand file stores origin–destination (OD) traffic demand by origin blocks.

We do not solve the original traffic-assignment problem. Instead, each TNTP network is converted into a Cost–Distance network-design benchmark. The conversion uses the default script settings `TN_NAME=all`, `TN_PROFILE=part`, and `TN_GRAPH_MODE=road_only`. If TNTP metadata provides `FIRST THRU NODE`, nodes with smaller indices are treated as zone or centroid nodes rather than road-design nodes; they are used only to project OD demand onto reachable road nodes. For small networks where all nodes allow through traffic, such as Sioux Falls, zone nodes are retained.

For edge weights, construction cost C_e is set to road length; if length is invalid, capacity is used as a fallback. Routing distance D_e is set to free-flow time; if free-flow time is missing or invalid, it is set to 1. For demands, each destination is treated as a candidate sink r, and all origins with positive demand to that destination become sources. If an origin or destination is a centroid, its demand is projected to its reachable road connector nodes and split across them when multiple connectors are available. Destinations with no reachable positive-demand source are skipped. Thus, each TN region yields multiple sink-centered Cost–Distance instances sharing the same processed road graph but differing in r, S, and w.

The main benchmark uses 8 different medium-to-large sparse TN regions. Table 2 reports the actual instance counts and source-set statistics used under the default `part` profile.

Table 2: TN-CDNet statistics for the 8 transportation-network regions used in the main benchmark. N and M are the processed road-only graph sizes shared by all instances of the same region.

Region	**Instances**	N	M	$S_{\min}$	S_{mean}	$S_{\max}$
Winnipeg	100	893	2284	5	72.40	215
Philadelphia	5	11864	30789	2387	3814.80	4299
chicago-regional	5	11189	35436	906	1357.00	1571
Berlin-Tiergarten	26	333	560	85	90.88	94
Barcelona	108	820	1957	5	181.45	208
Berlin-Mitte-Prenzlauerberg-Friedrichshain-Center	98	876	1410	331	331.98	335
Chicago-Sketch	100	933	2950	255	327.17	372
Berlin-Mitte-Center	36	361	583	118	119.06	121
Total / range	**478**	333–11864	560–35436	5–2387	–	94–4299

C.3 CD-RandNet: Synthetic Graphs

CD-RandNet is a controlled synthetic benchmark with 1,000 small-to-medium graph structures; each generated graph defines one directed Cost–Distance instance. The root is fixed to $r = 0$. We use node sizes $n \in \{200, 500, 1000, 1500, 2000\}$ and target expected template-edge count $m^\star = \lambda n$ with $\lambda = 10$. We first sample an undirected topology template, then replace each unordered template edge by two opposite directed arcs during loading so that all methods use the same directed-neighborhood interface.

[5] `https://github.com/bstabler/TransportationNetworks`

C.3.1 Random-Graph Families

We use four graph families. **Erdos–Renyi (ER)** [31] samples each unordered pair independently with

$$p = \frac{2\lambda}{n-1},$$

so $\mathbb{E}[|E|] \approx \lambda n$. This provides a uniform random-graph baseline. **Watts–Strogatz (WS)** [32] starts from a ring lattice with $k = 20$ neighbors per node, giving $|E| = nk/2 = 10n$, and rewires each lattice edge with probability $\beta = 0.2$. This creates high local clustering with a small number of long-range shortcuts. **Stochastic Block Model (SBM)** [33] partitions nodes into $B = 5$ balanced communities. With $p_{\text{in}}/p_{\text{out}} = 8$, the probabilities are calibrated by

$$p_{\text{in}} M_{\text{in}} + p_{\text{out}} M_{\text{out}} = 10n,$$

where M_{in} and M_{out} are the numbers of within-block and between-block node pairs. **Barabasi–Albert (BA)** [34] starts from a complete graph of 11 nodes and adds each new node with $m = 10$ preferential-attachment edges, yielding approximately $10n$ edges and a hub-dominated degree distribution.

C.3.2 Connectivity Guarantee

All sampled topology templates must be connected to the root before directed-arc expansion. If an ER, WS, or SBM sample is disconnected, we compute its connected components, identify the component containing $r = 0$, and repeatedly add one template edge from a non-root component to the current root component. After each template edge is expanded into two directed arcs, all nodes are root-reachable. This repair only adds cross-component template edges and leaves the sampled structure otherwise unchanged. BA templates are connected by construction and typically require no repair.

C.3.3 Node and Edge Weights

Each node first receives a base demand weight

$$w_v \sim \text{Uniform}(0.1, 0.5).$$

Then a Steiner ratio $\alpha = 0.3$ is applied: $\lfloor 0.3n \rfloor$ nodes are sampled uniformly and assigned zero weight. The root is always forced to $w_0 = 0$. Hence positive-weight nodes are sources, and zero-weight non-root nodes are Steiner nodes.

For every unordered template edge e, construction cost and routing distance are generated independently and assigned to both directed arcs obtained from e:

$$\tilde{C}_e \sim \mathcal{N}(500, 10^2), \qquad \tilde{D}_e \sim \mathcal{N}(20, 5^2),$$

and truncated to positive values by

$$C_e = \max(\tilde{C}_e, 10^{-6}), \qquad D_e = \max(\tilde{D}_e, 10^{-6}).$$

This yields heterogeneous edge weights while preventing degenerate zero or negative costs.

D Details of Other Methods

To guarantee the reproducibility and fair evaluation of all experimental results, we provide a detailed overview of the parameter settings used for all compared methods employed in this study. Each method was configured using either default values commonly reported in the literature or optimized through grid search or empirical validation on the validation set.

D.1 Gradient-Based Relaxation

D.1.1 Notears

We follow the acyclicity characterization proposed in the NOTEARS framework[18], which expresses the DAG constraint as a smooth function:

$$h(P) := \text{tr}\left(e^{P \odot P}\right) - n,$$

To enforce acyclicity during optimization, we adopt the quadratic penalty method by incorporating $h(P)^2$ into the objective function:

$$\min_P \mathcal{J}(P) + h(P)^2,$$

D.1.2 Gumbel Softmax (GS)

We use the Gumbel-Softmax trick[17] to obtain differentiable samples from a categorical distribution. By injecting Gumbel noise into the logits and applying a softmax with temperature $\tau = 1$, the resulting sample approximates a one-hot vector while remaining differentiable, enabling gradient-based optimization over discrete decisions. During training, the sampling batch size is 32 for an iteration with learning rate of 0.06.

D.1.3 Maximum Entropy Policy (MEP)

We adopt the maximum entropy reinforcement learning framework[35], which encourages policies that are both reward-maximizing and stochastic. Specifically, the policy maximizes the expected return plus an entropy regularization term, leading to exploration-aware behaviors:

$$\pi^* = \arg\max_\pi \mathbb{E}_\pi \left[\sum_t r_t + \alpha \mathcal{H}(\pi(\cdot|s_t)) \right],$$

where $\alpha = 1.0$ controls the trade-off between reward and entropy. During training, the sampling batch size is 32 for an iteration with learning rate of 0.06.

D.2 Cost–Distance-Specific Algorithms

D.2.1 Held Routing

Held Routing [15] employs a modified Dijkstra algorithm that iteratively constructs routing trees by minimizing a hybrid cost-distance objective. The method dynamically adjusts edge weights during path exploration to balance construction cost and routing distance.

The algorithm iteratively constructs a Steiner tree via terminal merging, guided by a modified cost function. In each iteration, a pair of terminals u, v minimizing

$$L(u, v) := \text{dist}_{G, c+\min\{w(u),w(v)\}\cdot d}(u, v) + b(u, v)$$

is selected, where $w(\cdot)$ denotes routing distance weights, $c(e)$ is edge construction cost, $d(e)$ is edge routing distance, and $b(u, v)$ is a bifurcation routing distance penalty. The algorithm uses sink-specific distance functions $\ell_u(e) := c(e) + w(u) \cdot d(e)$ to propagate labels in parallel from all sinks. After merging, a new Steiner vertex inherits routing distance weights and may serve as the root in the next iteration. This Cost–Distance-specific algorithm has an $O(\log |S|)$ performance guarantee and runs in $O(|S|(n \log n + m))$ time while accounting for bifurcation-induced routing distance penalties through a distributed model:

$$\text{dist}_T(r, t) = \sum_{(u,v)\in P} (d(u, v) + \lambda_v \cdot d_{\text{bif}})$$

where $\lambda_v \in [\eta, 1 - \eta]$ is determined by subtree weights. The parameter settings are consistent with those in the original paper.

D.2.2 Meyerson Baseline

The Meyerson baseline [1] is a randomized hierarchical aggregation algorithm. The initial active terminals are all positive-demand source nodes. The algorithm repeatedly reduces the number of active terminals until all demand has been connected to the root.

In each round, the algorithm constructs two classes of candidate connections. For a single active terminal u, the candidate is the shortest path from the root r to u under edge weights

$$c_e + w_u \ell_e,$$

where w_u is the current aggregated demand weight of terminal u. For a pair of active terminals (u, v), the algorithm computes shortest paths in both directions with the pair-dependent edge weight

$$c_e + \beta(u, v)\ell_e, \quad \beta(u, v) = \frac{2w_u w_v}{w_u + w_v}.$$

This coefficient reflects the weighted routing distance effect of merging two demand clusters. In the implementation, the lower-cost feasible path among $u \to v$ and $v \to u$ is used as the candidate connection for the pair.

All candidates are sorted by increasing cost and a greedy matching is constructed. If a terminal is matched with the root, its root path is added to the edge union and the terminal is removed from the active set. If two non-root terminals are matched, they are merged into a new terminal with weight

$$w_{uv} = w_u + w_v.$$

The merge center is chosen randomly: u is retained with probability

$$\frac{w_u}{w_u + w_v},$$

and v is retained with probability

$$\frac{w_v}{w_u + w_v}.$$

This randomness is controlled by a fixed random seed. At the end of each round, unmatched terminals and newly selected merge centers form the active set for the next round.

After all aggregation rounds, the algorithm obtains an edge union formed by multiple shortest paths. A final shortest-path search is run from the root on this union graph, and the path from each source to the root is extracted to form the final tree. This method does not perform gradient training and therefore has no learning rate or training-epoch parameter. In our experiments it uses the full original graph as the candidate edge set and runs on CPU. The GBA batch experiments impose an outer time limit of two hours per subgraph; the transportation-network and random-graph experiments impose an outer time limit of approximately 200 seconds per instance.

D.2.3 Chekuri Baseline

The Chekuri baseline [14] uses the same hierarchical aggregation framework as Meyerson, with the same candidate-path construction and greedy matching. The core difference is how the merge center is selected for a matched pair of non-root terminals.

For a matched pair (u, v), Chekuri evaluates both possible merge centers deterministically. If u is selected as the merge center, the score is

$$S(u) = C(P_{v \to u}) + w_v L(P_{v \to u}) + \gamma \widehat{D}(r, u; w_u + w_v).$$

Here $C(P)$ is the construction cost of path P, $L(P)$ is its routing distance, w_v is the demand weight merged into center u, and $\widehat{D}(r, u; W)$ estimates the future root-connection cost. This estimate is obtained by one shortest-path computation with edge weights

$$c_e + W\ell_e, \quad W = w_u + w_v.$$

The score $S(v)$ is computed symmetrically when v is used as the merge center. The algorithm chooses the endpoint with the smaller score; ties are broken by selecting the smaller node index.

We set $\gamma = 1.0$ to control the contribution of the future root-connection estimate. Larger γ makes the center choice favor endpoints that are easier to connect to the root in later rounds, whereas smaller γ makes the choice more dependent on the local pair-path cost. This baseline also uses the full original graph as the candidate edge set, runs on CPU, and uses the same evaluation propagation settings as the other compared methods.

D.3 Commercial Solver

D.3.1 Gurobi Baseline

The Gurobi baseline formulates Cost–Distance as a mixed-integer linear program and solves it with Gurobi Optimizer through `gurobipy` 13.0.1 [36]. The formulation sends flow from sources to the

root. For each candidate directed edge (i, j), where i is the current node and j is the next hop toward the root, we introduce a binary construction variable

$$x_{ij} \in \{0, 1\},$$

which indicates whether the edge is built, and a continuous flow variable

$$f_{ij} \geq 0,$$

which indicates how much demand is routed from i to j.

The objective is

$$\min_{x,f} \sum_{(i,j)\in E} c_{ij}x_{ij} + \ell_{ij}f_{ij}.$$

This objective exactly matches the sum of construction cost and routing distance flow cost. Let the total demand be

$$W = \sum_i w_i.$$

Each edge satisfies the capacity-linking constraint

$$0 \leq f_{ij} \leq W x_{ij}.$$

Thus only built edges can carry flow. Each positive-demand non-root node selects exactly one outgoing parent edge:

$$\sum_j x_{ij} = 1, \quad w_i > 0,\ i \neq r.$$

Each zero-demand Steiner node may remain unused, but if it is used it selects at most one parent edge:

$$\sum_j x_{ij} \leq 1, \quad w_i = 0,\ i \neq r.$$

Every non-root node satisfies flow conservation:

$$\sum_j f_{ij} - \sum_h f_{hi} = w_i, \quad i \neq r.$$

The total flow received by the root equals the total demand:

$$\sum_h f_{hr} = W.$$

After optimization, all edges with $x_{ij} > 0.5$ are returned as the final solution.

The solver relative optimality gap `MIPGap` is set to 10^{-4}, and we set `MIPFocus=1` to bias the search toward finding high-quality feasible solutions. The number of threads is fixed to 1, which makes runtimes more comparable across instances. GBA-CDNet uses a 7200-second Gurobi truncation budget, while TN-CDNet and CD-RandNet use a 200-second budget per instance. If the time budget is exhausted before optimality is certified, we request both Gurobi's best feasible incumbent and its current solver lower bound. The Gurobi bound reported for normalization is this lower bound at termination; it equals the optimum only when Gurobi certifies optimality within the prescribed time budgets. The pre-filter candidate edge set is the full set of available original graph edges.

The size of this MILP is determined by the actual candidate directed-edge set passed to Gurobi. Let $N = |\mathcal{V}|$, let M' be the number of candidate directed edges after masking, with the root treated as absorbing and therefore having no outgoing arc variables, and let $S = |\{i \in \mathcal{V} \setminus \{r\} : w_i > 0\}|$ be the number of positive-demand non-root sources. If M denotes the original directed-edge count, then $M' \leq M$. The model has M' binary construction variables and M' continuous flow variables, hence $2M'$ variables in total.

The capacity-linking constraints contribute M' inequalities. The outgoing-parent selection constraints contribute one constraint for each non-root node, namely S equalities for positive-demand sources and $N - 1 - S$ inequalities for zero-demand Steiner nodes. Flow conservation contributes $N - 1$ equalities, and the root-receipt constraint contributes one additional equality. Hence the total number of constraints is

$$M' + 2(N - 1) + 1 = M' + 2N - 1.$$

Table 3: Numerical objective ratios for Figure 2. Lower is better.

Dataset	Panel	#Inst.	CDPG	Notears	GS	Held	Meyerson	Chekuri	MEP	Gurobi
GBA	GBA-1	1	**1.071***	–	1.599	2.130	–	–	1.917	1.884
	GBA-2	1	**1.077***	–	1.526	2.351	–	–	1.802	1.875
	GBA-4	1	**1.113***	1.352	1.587	2.374	–	–	1.878	1.163
	GBA-5	1	**1.071**	1.264	1.286	2.005	–	–	1.523	1.081
	GBA-6	1	**1.062***	1.203	1.331	2.467	–	–	1.590	1.155
	GBA-7	1	**1.084***	1.239	2.032	1.725	–	–	2.437	1.115
	GBA-8	1	1.021	1.136	1.358	1.288	1.127	1.116	1.647	**1.019**
	GBA-9	1	1.012	1.105	1.101	1.106	1.089	1.066	1.322	**1.006**
TN	Winnipeg	100	1.006	1.364	1.074	1.223	1.328	1.140	1.228	**1.000**
	Philadelphia	5	1.182	–	1.237	1.400	–	–	1.484	**1.003**
	Chicago-reg.	5	1.132	–	1.163	1.598	–	–	1.392	**1.077**
	Berlin-Tier.	26	1.013	1.148	1.095	1.205	1.113	1.083	1.275	**1.000**
	Barcelona	108	1.007	1.145	1.032	1.314	1.246	1.095	1.238	**1.001**
	Berlin-MPF-Center	98	1.050	1.120	1.213	1.196	1.125	1.102	1.323	**1.016**
	Chicago-Sketch	100	1.002	1.380	1.047	1.221	1.285	1.093	1.259	**1.000**
	Berlin-Mitte-C.	36	1.007	1.093	1.049	1.178	1.088	1.071	1.259	**1.000**
Rand	ER	250	1.031	1.144	1.081	1.134	1.098	1.073	1.122	**1.001**
	WS	250	1.038	1.151	1.087	1.143	1.096	1.069	1.133	**1.002**
	SBM	250	1.034	1.144	1.083	1.137	1.099	1.073	1.125	**1.001**
	BA	250	1.014	1.123	1.077	1.113	1.097	1.067	1.109	**1.001**

Table 4: Numerical runtime values in seconds for Figure 2. Lower is better.

Dataset	Panel	#Inst.	CDPG	Notears	GS	Held	Meyerson	Chekuri	MEP	Gurobi
GBA	GBA-1	1	**21.75***	–	98.33	123.38	–	–	100.51	7285.45
	GBA-2	1	**12.21***	–	62.84	59.32	–	–	64.09	7259.52
	GBA-4	1	**3.61***	214.68	16.10	9.92	–	–	16.64	7213.48
	GBA-5	1	**2.82***	104.68	11.94	6.80	–	–	12.44	7212.69
	GBA-6	1	**5.81***	452.90	29.01	21.72	–	–	29.76	7230.19
	GBA-7	1	**3.25***	166.44	14.26	8.34	–	–	14.80	7214.71
	GBA-8	1	**1.57***	16.65	6.20	2.50	1884.71	1872.33	6.33	7206.61
	GBA-9	1	1.24	3.78	4.99	**0.71**	121.13	120.60	5.12	7203.07
TN	Winnipeg	100	1.19	4.33	4.41	**0.10**	0.35	0.35	4.84	16.69
	Philadelphia	5	**4.02***	–	49.11	27.88	–	–	49.48	172.62
	Chicago-reg.	5	**3.78***	–	43.52	7.48	–	–	44.58	201.65
	Berlin-Tier.	26	1.09	3.38	3.75	**0.10**	0.16	0.15	4.15	1.99
	Barcelona	108	1.09	4.22	4.37	**0.13**	1.13	1.14	4.98	23.13
	Berlin-MPF-Center	98	1.12	38.43	42.43	**0.16**	2.76	2.73	44.57	189.26
	Chicago-Sketch	100	1.19	4.59	4.45	**0.16**	3.74	3.73	4.72	7.47
	Berlin-Mitte-C.	36	0.88	3.05	3.80	**0.11**	0.20	0.20	4.09	1.22
Rand	ER	250	**1.06**	2.46	3.05	1.11	29.17	30.24	3.05	109.63
	WS	250	1.12	2.82	3.08	**1.10**	41.00	42.33	3.10	138.59
	SBM	250	1.15	2.44	3.03	**1.13**	37.80	39.05	3.09	130.49
	BA	250	**1.05**	1.83	3.08	1.10	40.00	40.53	3.42	98.97

Equivalently, the number of equality constraints is

$$S + (N - 1) + 1 = N + S,$$

and the number of inequality constraints is

$$M' + (N - 1 - S) = M' + N - 1 - S.$$

E Additional Experimental Results

E.1 Numerical Values for Runtime–Quality Comparison

Tables 3 and 4 list the numerical values used to plot Figure 2. Objective ratios are reported to three decimal places, and runtimes are reported to two decimal places. An asterisk denotes one-sided significance at $p < 0.05$ relative to the second-best method in the same row.

E.2 Value-Solver Depth Sweep

The exact acyclic Neumann expansion in Proposition 4.2 may require up to $n-1$ propagation steps, while Theorem 5.2 motivates using $d = \Theta(\log n)$. To examine whether the default depth is sufficient in practice, we sweep $d = \alpha\lceil \log_2 n \rceil$, with $\alpha \in \{0.2, 0.4, \ldots, 1.8, 2, 3\}$, on the eight GBA-CDNet evaluation instances used in the main figure. Settings with $\alpha > 1$ provide higher-depth references for the truncated value solver but require proportionally more propagations.

Figure 5 shows that the objective improves rapidly before $d = \lceil \log_2 n \rceil$, whereas increasing the depth beyond this point yields limited additional objective reduction on most panels and consistently increases per-outer-iteration time. This supports the logarithmic value-solver depth used throughout the main experiments as a favorable accuracy–runtime trade-off.

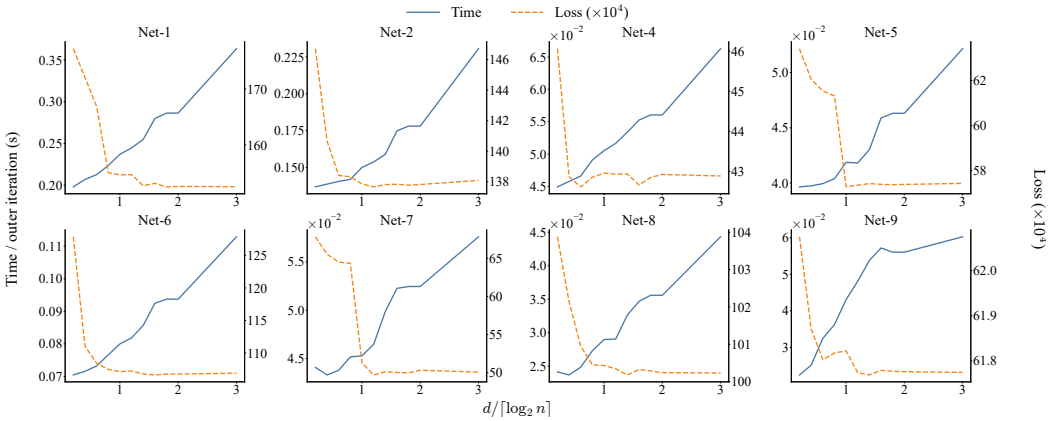

Figure 5: Value-solver depth sweep on GBA-CDNet. The horizontal axis reports the multiplier of $\lceil \log_2 n \rceil$ used for d. The left axis reports per-outer-iteration runtime with moving-average smoothing, and the right axis reports the Cost–Distance objective loss ($\times 10^4$).

E.3 Varying Source-Set Size on GBA-CDNet

To isolate the effect of the number of demand sources, we construct source-sweep variants of GBA-CDNet while keeping the graph topology, root, and edge weights fixed. For each instance, a subset $S' \subseteq S$ is randomly retained as active sources and all remaining original sources are assigned zero demand, so they can still be used as Steiner nodes. The first four settings retain approximately $0.5\log_2 n$, $\log_2 n$, $4\log_2 n$, and $16\log_2 n$ sources; the remaining settings retain $0.1|S|, 0.2|S|, \ldots, 0.8|S|$. CDPG and Held are evaluated on the same source-size schedule. Figure 6 reports the absolute Cost–Distance objective and runtime on the same eight GBA-CDNet panels used in the main figures.

The curves show two regimes. With logarithmic source budgets, Held can be cheaper because it connects only a few terminals. Once fractional portions of the dense UAV demand are active, CDPG becomes consistently stronger, improving the objective in 59 of 64 $0.1|S|$–$0.8|S|$ panel-setting pairs. Runtime follows the expected pattern: CDPG stays nearly flat because its training budget is fixed by graph size, while Held grows with active sources; CDPG is faster in 41 of 64 fractional-source settings.

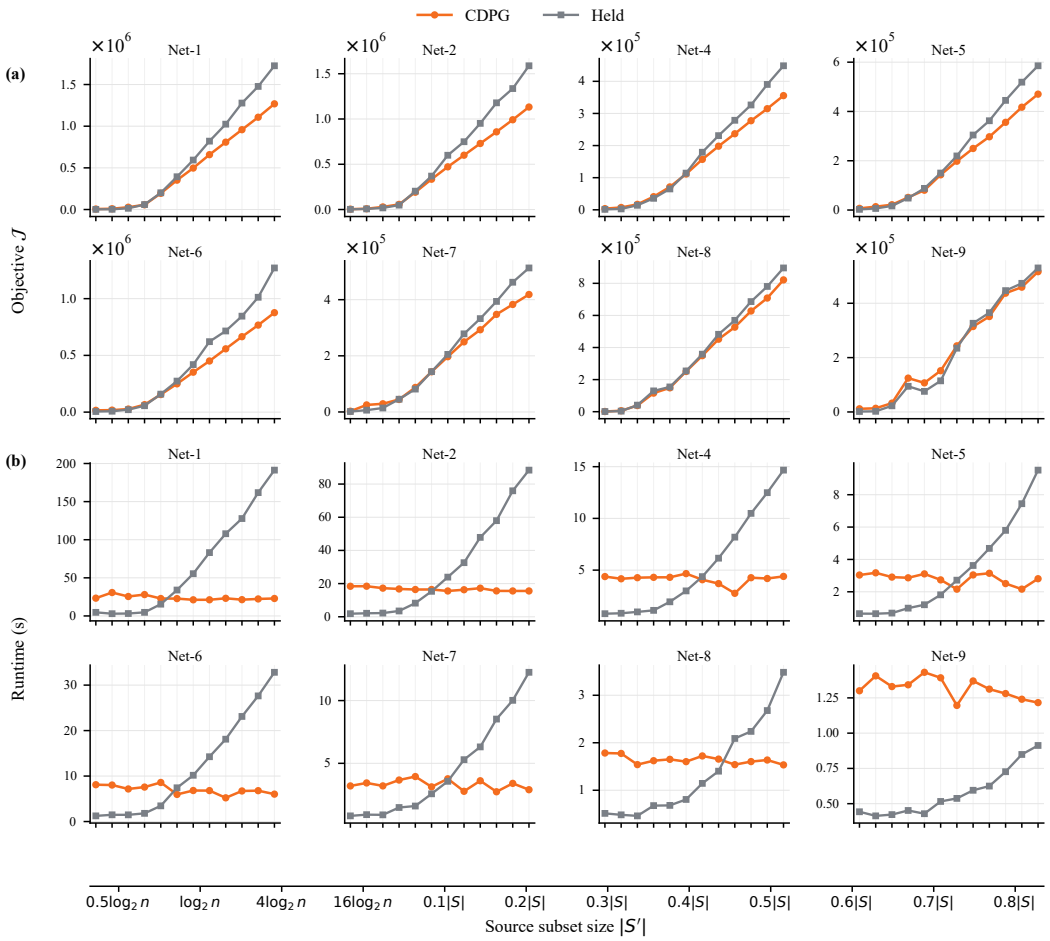

Figure 6: Source-set-size sweep on GBA-CDNet. The upper block reports objective values and the lower block reports runtime. Each block uses a 2×4 grid over UAV-network instances, and the shared bottom axis lists the source-subset sizes used in every panel.

E.4 Varying Routing Distance Weight Scale on GBA-CDNet

We further test sensitivity to the relative weight of routing distance in the Cost–Distance objective. For each GBA-CDNet instance, we keep the graph topology, root, source set, construction costs, and demands fixed, and multiply the routing distance component by a scale in $\{0.1, 0.2, 0.5, 1, 2, 5, 10\}$. CDPG and Held are evaluated on the same scale schedule. Figure 7 reports objective values and runtime on the same eight GBA-CDNet panels used in Appendix E.3.

Figure 7 shows that CDPG remains robust when the routing distance term is reweighted over two orders of magnitude. Across the 56 panel–scale pairs, CDPG obtains a lower objective than Held in 52 cases and is faster in 49 cases. The advantage strengthens as routing distance receives larger weight: the average CDPG/Held objective ratio decreases from 0.961 at scale 0.1 to 0.733 at scale 10, and CDPG wins on all eight GBA panels for every scale from 0.5 to 10. This demonstrates that CDPG is not tuned to a single cost–distance balance; it remains effective in both construction-dominant and routing distance-dominant regimes, with an average runtime ratio of 0.454 relative to Held.

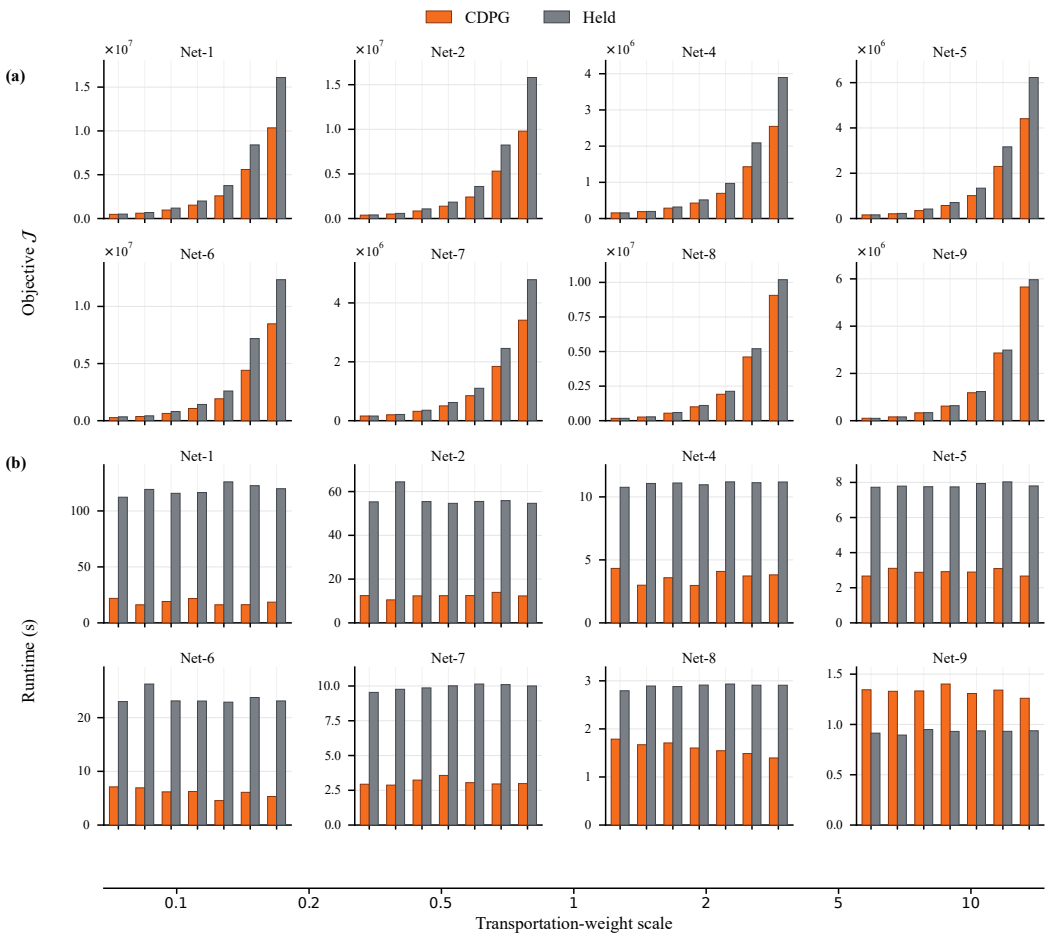

Figure 7: Routing distance weight scale sweep on GBA-CDNet. The upper block reports objective values and the lower block reports runtime. Each block uses a 2×4 grid over UAV-network instances, excluding GBA-CDNet-3 for consistency with the main appendix panels. Each scale forms one cluster with CDPG and Held bars.

E.5 Parameter Analysis

We select CDPG hyperparameters once on GBA-CDNet-3 and then fix them for every other instance. We tune the optimizer over {Adagrad, Adam, NAdam, RMSprop}, the DAD refresh interval over $\{1, 2, 5, 10, 20, 50\}$, the learning rate η over $\{0.02, 0.06, 0.10, 0.14\}$, the node-embedding rank l over $\{2, 4, 6, 8, 10, 12\}$, and the value-solver depth through $d/\lceil \log_2 n \rceil \in \{1, 2, 3\}$. The fixed main-experiment setting is RMSprop, DAD refresh interval 10, $\eta = 0.06$, $l = 12$, and $d = \lceil \log_2 n \rceil$.

Table 5 reports the $\eta \times q \times l$ loss grid on GBA-CDNet-3 setting after fixing $d = \lceil \log_2 n \rceil$ and using RMSprop. The minimum is 2.3995×10^5 at $(\eta, q, l) = (0.06, 10, 12)$, and the maximum is 2.5114×10^5 at $(0.14, 50, 10)$. The relative spread is only 4.66%, indicating stability across practical hyperparameter settings.

E.6 Large-Scale ER and BA Graphs

We evaluate two graph families at four scales, generating five independent instances for every family–scale pair, for a total of 40 additional instances. CDPG, Held, and GS are evaluated on the same instances. Table 6 reports the mean directed-arc and source counts. For each method, runtime and objective are divided by Held's values on the same instance before averaging the five paired ratios. Held therefore has value one for both metrics. These experiments use the fixed CDPG configuration from the main benchmark.

Table 5: Hyperparameter loss grid on GBA-CDNet-3 ($\times 10^5$).

q	$\eta = 0.02$						$\eta = 0.06$					
	rank l						rank l					
	2	4	6	8	10	12	2	4	6	8	10	12
1	2.4446	2.4500	2.4560	2.4304	2.4449	2.4507	2.4378	2.4418	2.4317	2.4364	2.4408	2.4336
2	2.4341	2.4342	2.4475	2.4328	2.4365	2.4315	2.4249	2.4289	2.4323	2.4149	2.4414	2.4119
5	2.4253	2.4186	2.4291	2.4156	2.4284	2.4199	2.4164	2.4140	2.4121	2.4079	2.4073	2.4043
10	2.4261	2.4189	2.4294	2.4038	2.4181	2.4188	2.4078	2.4040	2.4108	2.4033	2.4098	**2.3995**
20	2.4137	2.4128	2.4149	2.4018	2.4100	2.4056	2.4079	2.4028	2.4102	2.4003	2.4020	2.4009
50	2.4081	2.4020	2.4139	2.4011	2.4045	2.4045	2.4061	2.4005	2.4315	2.4211	2.4210	2.4182
q	$\eta = 0.10$						$\eta = 0.14$					
	rank l						rank l					
	2	4	6	8	10	12	2	4	6	8	10	12
1	2.4452	2.4514	2.4478	2.4423	2.4458	2.4539	2.4377	2.4424	2.4586	2.4498	2.4683	2.4769
2	2.4268	2.4242	2.4262	2.4174	2.4329	2.4275	2.4201	2.4252	2.4489	2.4274	2.4467	2.4522
5	2.4114	2.4103	2.4229	2.4229	2.4218	2.4228	2.4206	2.4225	2.4531	2.4425	2.4409	2.4423
10	2.4069	2.4083	2.4217	2.4232	2.4261	2.4240	2.4144	2.4264	2.4457	2.4519	2.4609	2.4409
20	2.4031	2.4059	2.4178	2.4230	2.4257	2.4239	2.4068	2.4247	2.4516	2.4611	2.4803	2.4449
50	2.4269	2.4277	2.4385	2.4373	2.4447	2.4252	2.4358	2.4247	2.4787	2.4856	2.5114	2.4786

Table 6: Large-scale ER and BA results. Graph counts and paired runtime/objective ratios are means over five instances per row. Lower ratios are better.

Family	n	$\overline{m}$	$\overline{\lvert S \rvert}$	Runtime / Held		Objective / Held	
				CDPG	GS	CDPG	GS
ER	5,000	200,016	3,499.0	0.584	1.296	0.985	1.015
ER	10,000	399,958	6,999.4	0.357	1.274	0.968	1.009
ER	20,000	799,868	13,999.4	0.271	1.247	0.944	0.999
ER	50,000	1,999,789	34,999.2	0.241	1.222	0.930	0.988
BA	5,000	199,580	3,499.2	0.614	1.338	0.996	1.024
BA	10,000	399,580	6,999.4	0.358	1.279	0.986	1.010
BA	20,000	799,580	13,999.4	0.272	1.268	0.982	1.016
BA	50,000	1,999,580	34,999.4	0.246	1.247	0.980	1.027

Across all eight settings, CDPG has lower mean runtime and objective ratios than Held. At the largest scale, its runtime ratios correspond to speedups of approximately $4.15\times$ on ER and $4.07\times$ on BA. GS has runtime ratios of 1.222 and 1.247 at this scale, with objective ratios of 0.988 and 1.027, respectively. The paired comparison measures both computational cost and solution quality as graph size increases.

E.7 DAD Refresh-Interval Sweep

To isolate refresh frequency, we evaluate $q \in \{1, 2, 5, 10, 20, 50\}$ on the eight GBA test graphs, excluding GBA-CDNet-3, with three seeds per setting. All 144 runs complete successfully. For each graph and seed, final objective and runtime are normalized by the corresponding $q = 10$ run; Table 7 reports means over the 24 paired comparisons per setting. Other hyperparameters remain fixed.

The practical range $q \geq 5$ gives similar objectives, with a maximum mean increase of 0.66% relative to the default. Updating at every iteration increases runtime by 56.53%. At $q = 20$ and $q = 50$, fewer mask reconstructions are offset by more optimizer iterations before the stopping criterion is met. A moderate interval therefore provides useful structural updates while keeping their overhead low.

Table 7: DAD refresh frequency on eight GBA test graphs and three seeds. Ratios are normalized by the matched default $q = 10$ run; lower is better.

Refresh interval q	Final objective / default	Runtime / default
1	1.0316	1.5653
2	1.0383	1.1868
5	1.0066	0.9635
10	1.0000	1.0000
20	0.9995	1.0536
50	0.9977	1.0814

F Future Work

This paper instantiates CDPG mainly as an instance-wise direct gradient descent (DGD) solver for Cost–Distance. A natural next step is to use the same differentiable Cost–Distance objective as an unsupervised surrogate for unsupervised learning (UL) and post-inference refinement pipelines. In that setting, a parametric model could be trained across many instances without optimal labels, using CDPG to provide the training signal and feasibility-aware routing evaluation. Such an amortized solver may learn transferable graph representations, reduce per-instance optimization time, and preserve high solution quality on unseen Cost–Distance instances. More broadly, the policy-based surrogate design suggests a path for extending similar unsupervised objectives to other ML4CO problems whose discrete decisions induce structured downstream costs. Another direction is to replace total routing distance with a min–max criterion on source distances, and develop the corresponding evaluation and rounding analysis for that objective.

G Broader Impacts

This work has positive broader impacts on logistics, transportation, and communication infrastructure planning. By improving the optimization of Cost–Distance network design, CDPG can help construct networks with lower construction cost and shorter routing distance, thereby supporting more efficient resource allocation and sustainable infrastructure design. The proposed method is general and can potentially benefit applications such as UAV logistics networks, transportation systems, sensor networks, and communication networks.

The work is primarily algorithmic and does not directly involve human subjects, personal decision making, or autonomous deployment. Therefore, we do not anticipate direct negative societal impacts. Potential risks are mainly associated with downstream use, such as the quality, privacy, or bias of input demand data, and should be addressed through domain-specific validation, data governance, and compliance with relevant safety and regulatory requirements.

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 introduction state the Cost–Distance problem setting, the CDPG surrogate, DAD, TopoRounding, and the benchmark contribution. The theoretical and experimental claims are supported by the theorems and the main runtime–quality comparisons.

   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: The paper discusses that the current study is limited to DGD-style per-instance optimization. Using the proposed surrogate to train generalizable neural networks is left as future work.

   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 paper states the assumptions for the smoothness, estimation, convergence, and complexity results in the theorem statements. Complete proofs and intermediate bounds are provided in the appendix and are cross-referenced from the main text.

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: The paper specifies the benchmark construction, objective, baselines, implementation choices, time budgets, and hyperparameter selection protocol. The appendix provides additional dataset details, Gurobi formulation, numerical values, and sensitivity studies needed to reproduce the main conclusions.

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: The code and processed benchmark instances are released in an anonymized repository with the submission. Raw mobile-signaling records and raw map-provider API responses are not released because of privacy protection and commercial data-use restrictions, but the released processed instances contain the information needed to reproduce the reported optimization experiments.

   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: The experimental section describes the datasets, evaluation metric, compared methods, hardware, time budgets, and optimizer settings. Additional details on dataset generation, baseline formulations, and hyperparameter selection are given in the appendix.

   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: Appendix Tables 3 and 4 mark results that are significant under one-sided tests at $p < 0.05$, comparing the best method against the second-best method in the same row.

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: The experimental section reports the GPU and CPU used for the main experiments and gives the time budgets for each benchmark family. Runtime tables and figures provide the observed execution times for the reported methods.

   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 research is algorithmic, uses anonymized and aggregated processed benchmark data, and does not involve deployment on users or personal decision making. The data release avoids exposing raw private mobility records.

   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: [Yes]

    Justification: The appendix discusses positive impacts for logistics and infrastructure planning and notes potential downstream risks related to data quality, privacy, and biased demand inputs. It also states that such risks should be handled through domain validation, data governance, and regulatory compliance.

    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: [Yes]

    Justification: The released benchmark omits raw mobile-signaling records, raw map-provider API responses, exact coordinates, user identifiers, trajectories, OD records, timestamps, and absolute population counts. Only processed and anonymized instance data needed for reproducibility are released.

    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: The paper credits the external data sources used to construct the benchmarks, including Tencent Location Service, Smart Footprint Data Technology Co., Ltd., and the open-source transportation-network source. The raw commercial and privacy-sensitive data are used under restrictions and are not redistributed.

    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: The paper introduces GBA-CDNet and documents its construction, instance statistics, released fields, and privacy-preserving preprocessing. The code repository contains the processed benchmark instances used in the experiments.

    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: The paper does not involve crowdsourcing experiments or direct interaction with human subjects. The mobility-derived demand data are used only after aggregation and privacy-preserving processing.

    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: The study does not conduct crowdsourcing or human-subject experiments. It uses processed aggregate demand information provided under commercial and privacy restrictions.

    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 are not used as an important, original, or non-standard component of the proposed method, theory, experiments, or benchmark construction.

    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.