From Ensembles to One Graph: Consensus DAGs with Minimal Information Loss

Efthyvoulos Drousiotis[1], Shashi Lakra[1], Matthew Feng[1], Fahim Razai[1], Ana L. Carvalho[1], Paul G. Spirakis[2], Simon Maskell[3], and Peter Green[4]

[1] Department of Health Data Science, Institute of Population Health, University of Liverpool, Liverpool, UK
[2] School of Computer Science and Informatics, University of Liverpool, Liverpool, UK
[3] Department of Electrical Engineering and Electronics, School of Engineering, University of Liverpool, Liverpool, UK
[4] Department of Mechanical and Aerospace Engineering, School of Engineering, University of Liverpool, Liverpool, UK
{Efthyvoulos.Drousiotis,Shashi.Lakra,Wenjie.Feng, Fahim.Razai,AnaC,P.Spirakis,SMaskell,PlGreen}@liverpool.ac.uk

Abstract. Ensemble-based Directed Acyclic Graph (DAG) structure learning, via Bayesian posterior sampling or resampling/bootstrapping, often produces multiple candidate DAGs annotated with graph weights and edge reliabilities. We propose a principled method to aggregate such an ensemble into a single consensus DAG, with an explicit trade-off between agreement with the ensemble and sparsity. Our confidence-weighted objective reduces, up to an additive constant, to maximising a linear edge-score under an acyclicity constraint, yielding a maximum-weight acyclic subgraph formulation. We solve this using (i) an Integer Linear Programming (ILP) approach that provides optimality certificates when solved to completion and (ii) a scalable greedy algorithm. A synthetic study over increasing graph sizes demonstrates the accuracy–efficiency trade-off between exact optimisation and the greedy heuristic.

Keywords: Directed Acyclic Graphs · Consensus Graphs · Bayesian networks · Maximum-weight acyclic subgraph · Graph aggregation

1 Introduction

1.1 Motivation

Many contemporary methods for probabilistic and causal structure learning output not a single Directed Acyclic Graph (DAG) [1–3], but an *ensemble* of candidate DAGs—for example from Bayesian posterior sampling, bootstrap resampling, or repeated stochastic [4, 5] runs of a learning algorithm. These ensembles can be used to represent structural uncertainty and often come with additional

metadata such as graph-level weights (reflecting model quality or posterior mass) and edge-level confidence scores (reflecting support for individual relations).

Despite their potential usefulness, ensembles can be awkward to interpret, communicate to domain experts, and integrate into downstream pipelines that typically expect a single graph. As a consequence, practitioners often collapse an ensemble into a "consensus" DAG using ad hoc rules such as thresholding edge frequencies, taking unions or intersections of edges [6, 7], or manually editing an aggregated graph. Such procedures are local in the sense that they are edge-wise, as they decide on each arc independently using marginal support statistics. They typically ignore heterogeneity in graph weights and edge confidences. Moreover, they do not provide a principled mechanism to trade off *sparsity* (a compact, interpretable graph) against *fidelity* (agreement with, or low loss relative to, the ensemble). In addition, purely edgewise aggregation can violate acyclicity, requiring a post hoc repair step that is not aligned with any explicit optimality criterion.

These limitations motivate a global optimisation view of consensus DAG construction. Given weighted, confidence-annotated input DAGs, we seek a single acyclic graph that minimises an explicit information-loss functional relative to the ensemble while allowing control over graph size. This formulation makes the meaning of "minimal information loss" precise in our setting (with respect to the defined loss, ensures the output is a valid DAG by construction, and yields both provably optimal (ILP, when solved to optimality) and scalable algorithms for consensus structure selection.

1.2 Related Work

Ensemble-based DAG learning arises naturally in Bayesian posterior sampling, bootstrap aggregation, and stability selection, where practitioners obtain collections of candidate DAGs together with edge reliabilities or frequencies. A common operational way to summarise such ensembles is to compute edge strengths (e.g., posterior probabilities or bootstrap frequencies) and then threshold them to form an aggregated network, sometimes followed by ad hoc post-processing to remove cycles. This is supported by widely used toolkits such as `bnlearn` [8]. While simple and interpretable, thresholding is inherently local and does not optimise a global objective over acyclic graphs.

A more formal notion of a *consensus Bayesian network structure* is studied via conditional-independence maps. In this view, a consensus DAG is required to be an independence map i.e., a DAG whose d-separation statements are all valid independences in the target model, of each input network, while minimising model complexity (e.g., number of parameters). Deciding whether a given DAG is a consensus structure is NP-hard, motivating heuristics based on variable orderings and minimal independence maps [9]. This approach provides rigorous combinatorial foundations, but it does not directly incorporate arbitrary graph weights, edge confidence scores, or an explicit information-loss functional.

Closest in spirit to our optimisation view are approaches that reduce consensus structure selection to a *Maximum Weight Acyclic Subdigraph* (MWAS)

problem. For instance, [10] derive directed edge weights from ensemble edge frequencies via a Beta-mixture model and then computes a score-optimal consensus BN by solving MWAS with an integer-programming approach. More recent work on *structural fusion* likewise combines edge-level evidence into a global score and then performs structure-level optimisation to output a single fused DAG [11,12]. These methods highlight the value of global acyclicity-constrained optimisation. Still, they are typically tailored to frequency-derived scores or specific fusion pipelines, and sparsity is controlled indirectly rather than through an explicit penalty.

Finally, we note that the formulation proposed in the current work connects to the broader literature on exact and approximate optimisation for DAG structure learning. Learning an optimal BN structure is NP-complete under standard scores [13], and exact solvers often rely on integer programming with explicit acyclicity constraints or order-based encodings [14]. In parallel, continuous relaxations such as NOTEARS provide differentiable characterisations of acyclicity, enabling gradient-based optimisation over weighted adjacency matrices [15]. From a graph-theoretic perspective, MWAS is NP-hard [16]; moreover, unless $P = NP$, no polynomial-time algorithm can guarantee an approximation factor strictly larger than $1/2$ in general (i.e., always achieving more than half of the optimal total weight) [17], motivating fast heuristics when exact optimisation is computationally expensive.

In contrast to the above, we propose a general consensus-DAG framework that (i) accepts arbitrary graph weights, (ii) incorporates edge confidences (and optionally node confidences), and (iii) defines an explicit confidence-weighted information-loss objective with a tunable sparsity penalty. This objective reduces to a linear edge-score maximisation under an acyclicity constraint, enabling both an exact Integer Linear Programming (ILP) solver and a scalable greedy heuristic.

1.3 Contributions

This work makes the following contributions.

- **Confidence-weighted objective for consensus DAG fusion.** We propose an aggregation objective for distilling an ensemble of DAGs into a single consensus DAG, where each input graph carries a weight and present edges may be annotated with confidence scores (and optional node reliabilities). The objective penalises weighted Hamming disagreements and includes an explicit sparsity term, under an acyclicity constraint.
- **Equivalence to maximum-weight acyclic subgraph (MWAS).** We show that minimising the proposed loss is equivalent (up to an additive constant) to maximising a linear edge-score subject to acyclicity. The resulting edge scores provide a principled way to combine graph weights and confidence-weighted evidence for/against each directed edge.
- **Two optimisation algorithms with complementary guarantees.** We give (i) an ILP formulation based on topological-order variables that can

certify global optimality when solved to completion (and returns the best feasible solution under a time limit), and (ii) a scalable greedy heuristic that adds edges in decreasing score order while maintaining acyclicity.
- **Synthetic benchmarking protocol for the optimality-efficiency trade-off.** We design a controlled synthetic generator that produces weighted, confidence-annotated DAG ensembles with tunable conflicts, and we evaluate both solvers across graph sizes (50 repetitions per size), reporting runtime, objective value, and agreement statistics.

2 Problem Setup

Let $\mathcal{V} = \{1, \dots, d\}$ be a fixed set of vertices. We are given an ensemble of M directed acyclic graphs (DAGs)

$$\mathcal{G} = \{G^{(m)}\}_{m=1}^{M}, \qquad G^{(m)} = (\mathcal{V}, \mathcal{E}^{(m)}),$$

represented by adjacency matrices $\mathbf{A}^{(m)} \in \{0,1\}^{d\times d}$, where $A_{ij}^{(m)} = 1$ iff $(i \to j) \in \mathcal{E}^{(m)}$ and $A_{ii}^{(m)} = 0$. Each graph $G^{(m)}$ also comes with a nonnegative *graph weight* w_m satisfying $\sum_{m=1}^{M} w_m = 1$. Each input graph may additionally be annotated with (i) a graph-level weight w_m and (ii) edge-level confidences $c_{ij}^{(m)}$ for present edges. These quantities are treated as *inputs* to the aggregation problem and can be produced by any upstream structure-learning pipeline (e.g., posterior sampling or bootstrap resampling). If weights are unavailable, we set $w_m = 1/M$; if edge confidences are unavailable, we set $c_{ij}^{(m)} = 1$ for all present edges.

Our goal is to construct a single *consensus DAG* $G = (\mathcal{V}, \mathcal{E})$ with adjacency matrix $\mathbf{A} \in \{0,1\}^{d\times d}$ such that $\mathbf{A}$ is acyclic and summarises the ensemble with minimal loss.

For each ensemble member m and each ordered pair (i, j) with $i \neq j$, we are given an *edge confidence* $c_{ij}^{(m)} \in [0,1]$ which is interpreted as the reliability of the directed relation $(i \to j)$ *when it is present* in $G^{(m)}$ (e.g., posterior edge probability, bootstrap selection frequency, or an internal score mapped to $[0,1]$). When $A_{ij}^{(m)} = 0$ (the edge is absent), no confidence value is required. Absence is handled separately via an asymmetry parameter introduced below.

Optionally, each graph m may also provide *node reliabilities* $v^{(m)} \in [0,1]^d$, where $v_i^{(m)}$ captures confidence in variable/node i (e.g., measurement quality, missingness, or node-level stability), allowing disagreements involving unreliable variables to be down-weighted. When available, we fold node reliabilities into a pairwise factor

$$g_{ij}^{(m)} = \mathrm{combine}\Big(v_i^{(m)}, v_j^{(m)}\Big) \in [0,1], \tag{1}$$

where combine can be, e.g., geometric mean ($\sqrt{ab}$), minimum ($\min\{a,b\}$), or product (ab). If node reliabilities are unavailable, we set $g_{ij}^{(m)} \equiv 1$.

Mismatch weights and absence-evidence parameter. We introduce an asymmetry parameter $\alpha \in [0, 1]$ to control how strongly *absence* of an edge in $G^{(m)}$ is treated as negative evidence. For each m and ordered pair (i, j), define the mismatch weight

$$\omega_{ij}^{(m)} = w_m \, g_{ij}^{(m)} \Big(A_{ij}^{(m)} \, c_{ij}^{(m)} + (1 - A_{ij}^{(m)}) \, \alpha \Big). \tag{2}$$

Thus, disagreements with a *present* edge $(i \to j)$ in $G^{(m)}$ are penalised proportionally to its confidence $c_{ij}^{(m)}$, while disagreements with an *absent* edge are penalised with strength scaled by α. Setting $\alpha = 1$ treats presence and absence symmetrically, whereas $\alpha = 0$ ignores absence as evidence.

In the next section, we define a loss that measures disagreement between the consensus adjacency matrix $\mathbf{A}$ and the ensemble $\{\mathbf{A}^{(m)}\}$ using the weighted Hamming mismatch $|A_{ij} - A_{ij}^{(m)}|$ scaled by $\omega_{ij}^{(m)}$, together with an explicit sparsity penalty on the number of edges in $\mathbf{A}$.

3 Methodology

3.1 Weighted Information Loss

We measure disagreement between the ensemble and a single consensus adjacency matrix $\mathbf{A}$ via a confidence-weighted Hamming mismatch plus a sparsity penalty:

$$\mathcal{L}_\lambda(\mathbf{A}) \;=\; \sum_{m=1}^{M} \sum_{i \neq j} \omega_{ij}^{(m)} \left| A_{ij} - A_{ij}^{(m)} \right| \;+\; \lambda \sum_{i \neq j} A_{ij}, \tag{3}$$

subject to $\mathbf{A}$ encoding a DAG and $A_{ii} = 0$. Here $\lambda \geq 0$ controls sparsity.

Here the graph weights w_m are absorbed into $\omega_{ij}^{(m)}$ (Section 2), so graphs with smaller w_m contribute less to the total loss.

The coefficients $\omega_{ij}^{(m)}$ (Section 2) modulate the cost of an edgewise mismatch. When $(i \to j)$ is present in $G^{(m)}$, mismatches are weighted by the edge confidence $c_{ij}^{(m)}$; when it is absent, mismatches are weighted by α, controlling how strongly absence is treated as negative evidence. The final term $\lambda \sum_{i \neq j} A_{ij}$ is an ℓ_0-style sparsity penalty encouraging compact consensus graphs.

3.2 Reduction to a Maximum-Weight Acyclic Subgraph Problem

Because $A_{ij}, A_{ij}^{(m)} \in \{0, 1\}$, we can expand the absolute value:

$$|A_{ij} - A_{ij}^{(m)}| \;=\; A_{ij}^{(m)} + A_{ij} - 2 A_{ij} A_{ij}^{(m)}.$$

Collecting terms depending on A_{ij} yields

$$\mathcal{L}_\lambda(\mathbf{A}) \;=\; C \;-\; \sum_{i \neq j} S_{ij} A_{ij}, \tag{4}$$

where $C = \sum_{m=1}^{M} \sum_{i \neq j} \omega_{ij}^{(m)} A_{ij}^{(m)}$ is a constant independent of $\mathbf{A}$, and the edge score matrix $\mathbf{S} \in \mathbb{R}^{d \times d}$ is

$$S_{ij} \;=\; \sum_{m=1}^{M} \omega_{ij}^{(m)} \big(2A_{ij}^{(m)} - 1\big) \;-\; \lambda, \tag{5}$$

Note that $\omega_{ij}^{(m)}$ already includes the graph weight w_m and (when available) the node-reliability factor $g_{ij}^{(m)}$. Therefore minimizing (3) is equivalent to maximizing $\sum_{i \neq j} S_{ij} A_{ij}$ subject to acyclicity i.e., obtaining

$$\max_{\mathbf{A} \in \{0,1\}^{d \times d}} \; \sum_{i \neq j} S_{ij} A_{ij} \quad \text{s.t. } \mathbf{A} \text{ is acyclic and } A_{ii} = 0. \tag{6}$$

Equation (6) defines the maximum-weight acyclic subgraph (MWAS) problem. Thus, consensus selection reduces to choosing a highest-scoring subset of directed edges that forms a DAG.

3.3 Exact ILP Solver

We introduce binary decision variables $x_{ij} \in \{0, 1\}$ indicating whether the directed edge $(i \to j)$ is selected, and integer order variables $r_i \in \{0, \ldots, d-1\}$ for each node. Using a big-M construction with $M = d$, we enforce acyclicity via

$$r_i + 1 \;\leq\; r_j + d\,(1 - x_{ij}) \qquad \forall i \neq j. \tag{7}$$

When $x_{ij} = 1$ the constraint reduces to $r_i + 1 \leq r_j$, enforcing $r_i < r_j$ along every selected edge; when $x_{ij} = 0$ it is non-binding. Distinct ranks are not required: enforcing a strict increase along each selected edge suffices to exclude directed cycles. (Optionally, we may restrict variables to candidate edges with $S_{ij} > 0$ without changing the optimum; see Lemma 1.)

The resulting mixed-integer linear program is

$$\max_{\{x_{ij}\},\{r_i\}} \quad \sum_{i \neq j} S_{ij}\, x_{ij} \tag{8}$$

$$\begin{aligned} \text{s.t.} \quad & r_i + 1 \leq r_j + d\,(1 - x_{ij}) \quad \forall i \neq j, \\ & x_{ij} \in \{0, 1\}, \;\; r_i \in \{0, \ldots, d-1\}, \;\; x_{ii} = 0. \end{aligned} \tag{9}$$

We solve (8)–(9) with a MILP solver, yielding an optimal consensus DAG when solved to completion, or the best incumbent integer-feasible solution found under a time limit. Algorithm 1 summarises the formulation.

3.4 Greedy Acyclic Edge Addition

To obtain a faster alternative to the exact ILP optimisation, we also implement a greedy heuristic for the MWAS objective (6). The procedure constructs an

Algorithm 1: Exact ILP solver for consensus DAG selection. With a time limit, the solver returns the best incumbent integer-feasible solution found.

Input: Edge-score matrix $S \in \mathbb{R}^{d \times d}$, time limit T (seconds)
Output: Consensus adjacency matrix $A \in \{0,1\}^{d \times d}$ (a DAG)
Create a MILP with binary variables $x_{ij} \in \{0,1\}$ for all $i \neq j$
Create integer order variables $r_i \in \{0, \ldots, d-1\}$ for all i

Objective: maximize $\sum_{i \neq j} S_{ij}\, x_{ij}$

Acyclicity constraints (big-M, with $M = d$):
for $i \leftarrow 1$ **to** d **do**
  for $j \leftarrow 1$ **to** d **do**
    if $i \neq j$ **then**
      Add constraint $r_i + 1 \leq r_j + d\,(1 - x_{ij})$

Solve the MILP with time limit T
Set $A_{ij} \leftarrow x_{ij}$ for all $i \neq j$ and $A_{ii} \leftarrow 0$
return A

acyclic edge set by processing candidate edges in decreasing score order and accepting an edge only if it preserves acyclicity. Unlike the ILP solver, which can certify global optimality when it terminates with status *Optimal*, the greedy method provides no optimality guarantee; it is intended as a scalable heuristic that often achieves near-optimal objective values in practice.

1. Compute the score matrix $\mathbf{S}$ in (5).
2. Sort all directed edges $(i \to j)$ with $S_{ij} > 0$ by decreasing score.
3. Initialize $\mathbf{A} = \mathbf{0}$.
4. Iterate through the sorted edges and add $(i \to j)$ if and only if doing so does not create a directed cycle.

Cycle checks are performed by reachability (testing whether i is reachable from j before adding $i \to j$), ensuring the output is always a valid DAG. The runtime is dominated by sorting $O(d^2 \log d)$ candidate edges plus incremental reachability checks.

Algorithm 2 summarises the greedy acyclic edge-addition heuristic used in our experiments.

4 Theoretical Properties

We collect four basic properties of the consensus objective. Let $\mathcal{V} = \{1, \ldots, d\}$ and let $\{A^{(m)}\}_{m=1}^{M}$ be adjacency matrices of input DAGs on $\mathcal{V}$. For each m and each ordered pair (i, j) with $i \neq j$, let $\omega_{ij}^{(m)} \geq 0$ be the mismatch weight defined in Section 2 (Eq. (2)), and let $\lambda \geq 0$.

Algorithm 2: Greedy acyclic edge-addition heuristic. REACHABLE$(u, v; \mathcal{N})$ returns true iff v is reachable from u via directed paths under adjacency lists $\mathcal{N}$.

Input: Edge-score matrix $S \in \mathbb{R}^{d\times d}$
Output: Consensus adjacency matrix $A \in \{0,1\}^{d\times d}$ (a DAG)
$A \leftarrow \mathbf{0}$
for $i \leftarrow 1$ **to** d **do**
  $\mathcal{N}(i) \leftarrow \emptyset$ `// adjacency list of outgoing neighbours`
$\mathcal{E} \leftarrow \{(i, j, S_{ij}) : i \neq j,\ S_{ij} > 0\}$
Sort $\mathcal{E}$ in descending order of score
foreach $(i, j, s) \in \mathcal{E}$ **do**
  if REACHABLE$(j, i; \mathcal{N})$ **then**
    continue `// adding` $i \to j$ `would create a cycle`
  $A_{ij} \leftarrow 1$
  $\mathcal{N}(i) \leftarrow \mathcal{N}(i) \cup \{j\}$
return A

Define the loss

$$\mathcal{L}_\lambda(A) \;=\; \sum_{m=1}^{M} \sum_{i\neq j} \omega_{ij}^{(m)} \, |A_{ij} - A_{ij}^{(m)}| \;+\; \lambda \sum_{i\neq j} A_{ij}, \tag{10}$$

where $A \in \{0,1\}^{d\times d}$ is constrained to be acyclic and $A_{ii} = 0$. Let $\mathcal{A}$ denote the set of all such DAG adjacency matrices.

Edge scores. For $i \neq j$, define the aggregated edge score

$$S_{ij} \;=\; \sum_{m=1}^{M} \omega_{ij}^{(m)} \big(2A_{ij}^{(m)} - 1\big) \;-\; \lambda. \tag{11}$$

Theorem 1 (Score reduction). *There exists a constant C independent of A such that, for all feasible $A \in \mathcal{A}$,*

$$\mathcal{L}_\lambda(A) \;=\; C \;-\; \sum_{i\neq j} S_{ij} A_{ij}. \tag{12}$$

Consequently, minimising $\mathcal{L}_\lambda$ over $\mathcal{A}$ is equivalent to maximising $\sum_{i\neq j} S_{ij}A_{ij}$ over $\mathcal{A}$ (a maximum-weight acyclic subgraph problem).

Proof. For binary $A_{ij}, A_{ij}^{(m)} \in \{0,1\}$, $|A_{ij} - A_{ij}^{(m)}| = A_{ij}^{(m)} + A_{ij} - 2A_{ij}A_{ij}^{(m)}$. Substituting into (10) gives

$$\mathcal{L}_\lambda(A) = \underbrace{\sum_{m=1}^{M} \sum_{i\neq j} \omega_{ij}^{(m)} A_{ij}^{(m)}}_{=:C} + \sum_{i\neq j} A_{ij} \Big(\sum_{m=1}^{M} \omega_{ij}^{(m)} (1 - 2A_{ij}^{(m)}) \Big) + \lambda \sum_{i\neq j} A_{ij}.$$

Collecting the coefficients of A_{ij} yields (12) with S_{ij} as in (11). Since C is constant in A, minimising $\mathcal{L}_\lambda$ is equivalent to maximising $\sum_{i \neq j} S_{ij} A_{ij}$ over $A \in \mathcal{A}$.

Proposition 1 (Edgewise optimum without the acyclicity constraint). *If the acyclicity constraint is dropped (only requiring $A_{ii} = 0$), an optimal solution is obtained independently for each (i, j) by*

$$A^\star_{ij} \;=\; \mathbb{I}\{S_{ij} > 0\}, \qquad i \neq j. \tag{13}$$

Proof. Dropping acyclicity reduces the problem to maximising $\sum_{i \neq j} S_{ij} A_{ij}$ over independent binary variables A_{ij}, so each term is maximised by $A_{ij} = 1$ iff $S_{ij} > 0$.

Lemma 1 (Optimal solutions may be chosen to contain only positive-score edges). *Let $A^\star$ be an optimal solution of the DAG-constrained problem. Then there exists an optimal solution $\widetilde{A} \in \mathcal{A}$ with $\widetilde{A} \subseteq A^\star$ (edgewise) such that $\widetilde{A}_{ij} = 1 \Rightarrow S_{ij} > 0$.*

Proof. If a feasible DAG A contains an edge (i, j) with $S_{ij} \leq 0$, removing that edge preserves acyclicity and does not decrease the objective $\sum_{u \neq v} S_{uv} A_{uv}$. Repeating removes all non-positive-score edges without worsening the objective, yielding an optimal $\widetilde{A}$ with only positive-score edges.

Theorem 2 (Greedy optimality when the positive-score subgraph is acyclic). *Let $E^+ = \{(i, j) : S_{ij} > 0\}$ and let $G^+ = (\mathcal{V}, E^+)$. If G^+ is acyclic, then the DAG A^+ defined by $A^+_{ij} = \mathbb{I}\{S_{ij} > 0\}$ is optimal for the DAG-constrained problem. Moreover, the greedy algorithm that processes edges in decreasing S_{ij} order and adds an edge iff it preserves acyclicity returns A^+.*

Proof. If G^+ is acyclic then A^+ is feasible. Any feasible DAG omitting some $(i, j) \in E^+$ has strictly smaller objective since adding (i, j) preserves acyclicity and increases $\sum_{i \neq j} S_{ij} A_{ij}$. The greedy procedure encounters all edges in E^+ before any non-positive edges; since none of them creates a cycle under the assumption, greedy adds them all and returns A^+.

5 Experiments

We evaluate the greedy heuristic and the ILP solver on a controlled synthetic generator that produces ensembles of weighted, confidence-annotated DAGs on a fixed node set $\mathcal{V} = \{1, \ldots, d\}$. We sweep

$$d \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\},$$

and run 50 independent repetitions per d. Each repetition generates an ensemble of size $M = 50$. All experiments use the *hard* setting $\delta = 1$ (we do not sweep δ), because it induces sufficiently strong conflicts among high-scoring edges to expose differences between the greedy heuristic and the ILP baseline. Randomness is controlled by a global seed (SEED= 0) and, for each repetition, an independent 32-bit seed is sampled and used to initialise a fresh RNG.

5.1 Synthetic ensemble generator

Ground-truth DAG. For each repetition we sample a base topological order π_0 uniformly at random and generate a ground-truth DAG A^{true} by sampling each forward edge (consistent with π_0) independently with probability

$$p_{\text{true}} = \text{clip}\big(p_0(1 + 0.5\delta),\, 0.05,\, 0.6\big), \qquad \text{with } p_0 = 0.25.$$

Thus, in the hard setting $\delta = 1$, we have $p_{\text{true}} = 0.375$. Here $\text{clip}(x, \ell, u) = \min\{u, \max\{\ell, x\}\}$.

Per-graph topological orders via disjoint triples. To induce systematic disagreements across ensemble members, we partition nodes into disjoint triples (a random partition from a random permutation of nodes). In the hard setting we activate *all* triples and, for each ensemble member m, assign to each triple one of three equally likely cyclic modes:

$$(a, b, c), \quad (b, c, a), \quad (c, a, b).$$

These local triple orders are then concatenated in the order in which the triples first appear along π_0, followed by any leftover nodes (if $d \not\equiv 0 \pmod 3$), producing a per-graph order π_m. Each generated graph is guaranteed acyclic, as edges are only inserted forward with respect to π_m.

Noisy ensemble members. Each ensemble member $A^{(m)}$ is generated from A^{true} in two steps. First, each true edge is kept with probability $1 - \text{fnr}$ where

$$\text{fnr} = 0.03 + 0.30\delta,$$

and, conditional on being kept, is oriented to be consistent with π_m: if π_m ranks i before j we set $i \to j$, otherwise we reverse it to $j \to i$. Second, we add spurious forward edges (consistent with π_m) independently with probability

$$\text{fpr} = 0.01 + 0.30\delta$$

among ordered pairs not already occupied.

6 Results

We report results over 50 independent repetitions for each graph size d in the hard setting ($\delta = 1$). In the hard setting, the generator activates the maximum fraction of triple-order perturbations, inducing many conflicting high-score edges across ensemble members and thereby creating frequent cycle-breaking trade-offs for the consensus solvers. The error bars denote the standard error of the mean. The ILP solver (CBC via PuLP) is run with a time limit of 700s. In this sweep, it returns status *Optimal* on all instances, so $\mathcal{L}(A_{\text{ILP}})$ is a certified global optimum of our objective.

d	t_g (s)	t_{ILP} (s)	speedup	$\Delta\mathcal{L}$	rel. gap	Ham.
10	1.20×10^{-4}	1.72×10^{-1}	1.44×10^{3}	9.89×10^{-3}	2.74%	2.62
50	1.04×10^{-2}	3.30	3.19×10^{2}	5.33×10^{-2}	2.81%	15.08
100	1.31×10^{-1}	7.69×10^{1}	5.88×10^{2}	1.03×10^{-1}	2.62%	29.16

Table 1: Hard setting (δ=1), 50 repetitions per d. Greedy runtime t_g vs ILP runtime t_{ILP}, speedup t_{ILP}/t_g, objective gap $\Delta\mathcal{L} = \mathcal{L}(A_g) - \mathcal{L}(A_{\mathrm{ILP}})$ (ILP is optimal in all runs), relative gap $\Delta\mathcal{L}/\mathcal{L}(A_{\mathrm{ILP}})$, and Hamming distance $\|A_g - A_{\mathrm{ILP}}\|_1$.

Figure 1(a) shows runtime versus d (logarithmic y-axis). Greedy remains extremely fast across the full sweep (from $\approx 10^{-4}$s at $d = 10$ to $\approx 10^{-1}$s at $d = 100$), whereas the ILP runtime grows steeply with d (from $\approx 10^{-1}$s at $d = 10$ to $\approx 10^{2}$s at $d = 100$). The resulting speedups range from a few hundred$\times$ to over $10^3\times$ (Table 1).

Figure 1(b) reports the optimality gap in our objective, $\Delta(d) = \mathcal{L}(A_g) - \mathcal{L}(A_{\mathrm{ILP}}) \geq 0$. The mean gap increases mildly with d but remains small in absolute and relative terms. Across the sweep the relative loss gap stays around $\approx 2.6\%$–2.8% (Table 1). Thus, despite its purely local edge-addition rule, greedy typically finds solutions that are very close to optimal under the proposed information-loss objective.

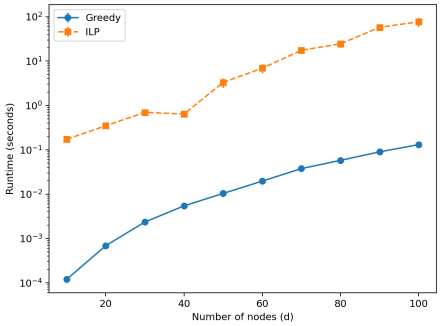
Exact equality of adjacency matrices is a strict notion. A single edge swap already yields $A_g \neq A_{\mathrm{ILP}}$. To quantify *magnitude* of disagreement, figure 1(c) plots the Hamming distance $\|A_g - A_{\mathrm{ILP}}\|_1$. Even at $d = 100$, the two solutions differ by only ≈ 29 directed edges on average, out of up to $d(d-1)/2 = 4950$ edges in a complete DAG order. This explains why the mismatch indicator can be close to one (Fig. 2) for larger d while the objective gap remains modest.

Figure 1(d) further relates solution quality to structural difference by plotting the per-instance objective gap against the Hamming distance. Instances with larger structural deviation tend to have larger gaps, but the overall gaps remain small, indicating that many alternative acyclic edge sets achieve very similar total score. This is consistent with the hard generator producing numerous near-ties among high-scoring edges, where resolving cycles requires global trade-offs that greedy and ILP may break differently.

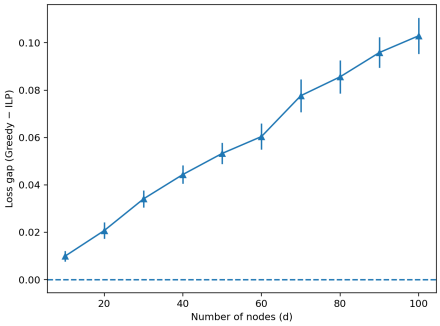
7 Discussion

Our experiments highlight a clear optimality-efficiency trade-off. The ILP solver provides certificates of global optimality for moderate graph sizes, making it valuable both as a benchmark and as a practical solver when d is not too large. However, its runtime increases rapidly with d, reflecting the combinatorial nature of selecting a maximum weight acyclic subgraph.

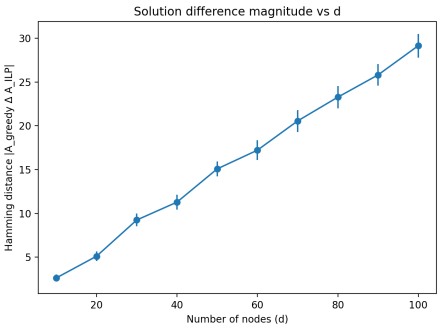
In contrast, the greedy heuristic scales smoothly and delivers solutions that are near-optimal in the proposed objective throughout the sweep. Importantly,



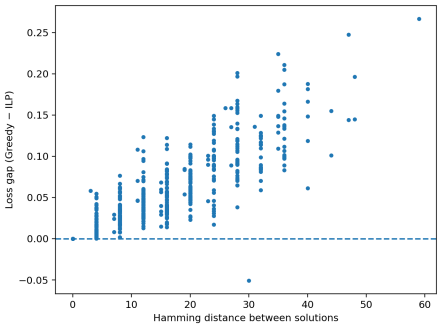
(a) Runtime vs. d (log y-axis).



(b) Objective gap.



(c) Hamming distance vs. d.



(d) Gap vs. structural deviation.

Fig. 1: Hard setting ($\delta = 1$), 50 repetitions per d; error bars are standard error of the mean.

the frequent event $A_g \neq A_{\text{ILP}}$ at larger d should not be over-interpreted. The Hamming distance between the two solutions remains small relative to the total number of feasible edges, and the objective gaps remain around only a few percent. Empirically, this indicates that the hard instances contain many high-scoring edges whose scores are close, so there are multiple acyclic edge sets with very similar total scores. Greedy may commit early to one high-scoring edge that later blocks a slightly better combination, whereas the ILP can globally reconfigure a small set of edges to eliminate cycles while improving the total score.

From a practitioner's perspective, these results suggest a simple workflow. Use greedy as the default solver for large graphs (fast, near-optimal), and use ILP when certificates are required or when validating the heuristic on smaller instances. When interpretability and sparsity are priorities, the sparsity penalty λ provides an explicit knob to control edge density; exploring the loss-sparsity trade-off (e.g., a Pareto curve over λ) is a natural extension for application-driven studies.

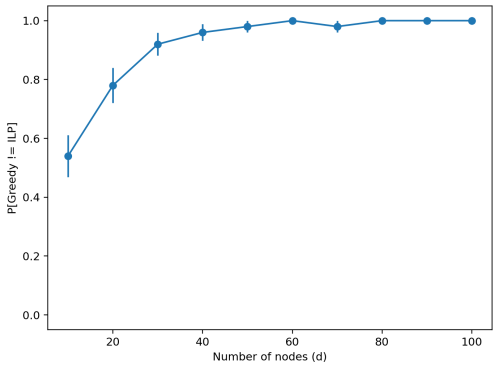


Fig. 2: Mismatch rate $\mathbb{P}(A_{\mathrm{g}} \neq A_{\mathrm{ILP}})$ in the hard setting ($\delta = 1$).

8 Conclusion

We introduced a principled formulation for collapsing an ensemble of confidence-annotated DAGs into a single *consensus* DAG. Our objective measures weighted Hamming disagreement with the ensemble—allowing heterogeneous graph weights and edge/node reliabilities—together with an optional sparsity penalty, while enforcing global acyclicity. We showed that minimising this loss is equivalent to a maximum-weight acyclic subgraph problem with an explicit edge-score matrix, clarifying the role of the acyclicity constraint and the connection to standard edgewise thresholding.

To solve the resulting optimisation, we presented two complementary solvers. A fast greedy acyclic-add heuristic and an exact ILP baseline. Experiments on a controlled hard synthetic generator demonstrate a clear accuracy-efficiency trade-off. Greedy scales smoothly to $d = 100$ with sub-second runtimes, whereas ILP runtime grows rapidly, while still providing certificates of optimality. Despite frequent structural mismatches between greedy and ILP solutions, the greedy solver attains near-optimal objective values across the sweep, indicating a "many near-optima" regime where multiple distinct DAGs have very similar loss.

Overall, our results suggest using ILP when a certified optimum is required (e.g., benchmarking, small graphs), and greedy as a scalable default for larger instances or repeated analyses. Future work includes extending the formulation to incorporate additional structural constraints (e.g., bounded in-degree), studying approximation guarantees under score-structure assumptions, and evaluating the approach on real ensembles produced by Bayesian posterior sampling and bootstrap-based structure learning.

References

1. L. Zhou and H. Fujita, "Posterior probability based ensemble strategy using optimizing decision directed acyclic graph for multi-class classification," *Information*

Sciences, vol. 400, pp. 142–156, 2017.
2. R. Wang and J. Peng, "Learning directed acyclic graphs via bootstrap aggregating," *arXiv preprint arXiv:1406.2098*, 2014.
3. S. Chowdhury, R. Wang, Q. Yu, C. J. Huntoon, L. M. Karnitz, S. H. Kaufmann, S. P. Gygi, M. J. Birrer, A. G. Paulovich, J. Peng, *et al.*, "Dagbagm: learning directed acyclic graphs of mixed variables with an application to identify protein biomarkers for treatment response in ovarian cancer," *BMC bioinformatics*, vol. 23, no. 1, p. 321, 2022.
4. H. Lähdesmäki and I. Shmulevich, "Learning the structure of dynamic bayesian networks from time series and steady state measurements," *Machine Learning*, vol. 71, no. 2, pp. 185–217, 2008.
5. S. Acid, L. M. de Campos, and J. G. Castellano, "Learning bayesian network classifiers: Searching in a space of partially directed acyclic graphs," *Machine learning*, vol. 59, no. 3, pp. 213–235, 2005.
6. N. Friedman, M. Goldszmidt, and A. Wyner, "Data analysis with bayesian networks: A bootstrap approach," *arXiv preprint arXiv:1301.6695*, 2013.
7. M. Scutari and R. Nagarajan, "On identifying significant edges in graphical models," in *Proceedings of workshop on probabilistic problem solving in biomedicine*, pp. 15–27, Springer-Verlag Bled, Slovenia, 2011.
8. M. Scutari, "Learning bayesian networks with the bnlearn r package," *Journal of statistical software*, vol. 35, pp. 1–22, 2010.
9. J. M. Peña, "Finding consensus Bayesian network structures," *Journal of Artificial Intelligence Research*, vol. 42, pp. 661–687, 2011.
10. H. Fröhlich and G. W. Klau, "Reconstructing consensus Bayesian network structures with application to learning molecular interaction networks," in *German Conference on Bioinformatics (GCB)*, vol. 34 of *OASIcs*, pp. 46–55, 2013.
11. J. M. Puerta, J. A. Aledo, J. A. Gámez, and J. D. Laborda, "Efficient and accurate structural fusion of Bayesian networks," *Information Fusion*, vol. 66, pp. 155–169, 2021.
12. J. D. Laborda Sicilia, J. M. Puerta, and J. A. Gámez, "Parallel structural learning of Bayesian networks via iterative structural fusion," *Knowledge-Based Systems*, 2024.
13. D. M. Chickering, "Learning bayesian networks is np-complete," in *Learning from data: Artificial intelligence and statistics V*, pp. 121–130, Springer, 1996.
14. M. Bartlett and J. Cussens, "Integer linear programming for the Bayesian network structure learning problem," *Artificial Intelligence*, vol. 244, pp. 258–271, 2017.
15. X. Zheng, B. Aragam, P. K. Ravikumar, and E. P. Xing, "Dags with no tears: Continuous optimization for structure learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.
16. V. Guruswami, R. Manokaran, and P. Raghavendra, "Beating the random ordering is hard: Inapproximability of maximum acyclic subgraph," in *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pp. 573–582, IEEE, 2008.
17. R. Hassin and S. Rubinstein, "Approximations for the maximum acyclic subgraph problem," *Information Processing Letters*, vol. 51, no. 3, pp. 133–140, 1994.