MILP Alternatives to MaxSAT for Synthesizing Pareto-Optimal Interpretations of Black-Box Models

Aryan Katiyar Himanshu Gangwal

Department of Computer Science
Indian Institute of Technology Bombay

November 2025

Outline

Motivation

- Black-box models (NNs, ensembles, etc.) are accurate but hard to interpret.
- Framework of Torfah et al. synthesizes **bounded decision diagrams (DDs)** as global explanations.
- They optimize a trade-off:

$$\Delta_{\mathcal{C}} \text{ (correctness)} \quad \text{vs.} \quad \Delta_{\mathcal{E}} \text{ (explainability)}.$$

- Original backend: **weighted MaxSAT**, repeatedly solved to enumerate a Pareto frontier.
- In many workflows, we only need the **first Pareto point** (maximizing $\Delta_{\mathcal{C}} + \Delta_{\mathcal{E}}$).

Goal of This Work

Main question

Can we obtain the **first Pareto-optimal explanation** faster by replacing MaxSAT with suitable MILP encodings, while preserving the original semantics?

- Keep the **same DD template**, predicates, and scoring.
- Study structural issues in the original encoding (repetition, single-node collapse).
- Propose **repetition constraints** and alternative weight schemes.
- Implement and compare several **MILP/IP formulations**.

Template and Variables

- Internal nodes: $N = \{1, \ldots, k\}$ (root $= 1$).
- Leaves: L (output labels).
- Predicates: P with discrete branches $B(p)$.
- Samples: S with black-box labels.

Boolean variables:

- $\lambda_{i,p}$: node i tests predicate p.
- $\tau_{i,c,j}$: transition from node i to j on branch c.
- $m_{i,s}$: sample s reaches node/leaf i.
- u_i: node i is reachable from root.
- $\lambda'_{i,p} \leftrightarrow u_i \wedge \lambda_{i,p}$ (predicate on a used node).

Hard and Soft Blocks

- **Hard constraints**
  - φ_E: structural constraints (one predicate per node, valid transitions, acyclicity).
  - φ_S: sample propagation / consistency.
- **Soft blocks**
  - Correctness:

$$\varphi_{\Delta_C} : \quad (m_{1,s}) \text{ for each sample } s.$$

  - Explainability:

$$\Delta_{\mathcal{E}} = \sum_{p,i} w(p) \cdot \lambda'_{i,p} + \sum_{i} W_{\text{unused}} \cdot \neg u_i.$$

$$\varphi_{\Delta_{\mathcal{E}}} : \ (\lambda'_{i,p}) \text{ and } (\neg u_i) \text{ with weights.}$$

Datasets (1/3)

- **Adult Income (UCI)**
  - Binary classification: income $>$ 50K.
  - 3 features (from 14).

$$\langle \text{age: } 4,\ \text{hours\_per\_week: } 4,\ \text{workclass: } 6 \rangle$$

- **Banknote Authentication (UCI)**
  - Genuine vs. forged banknotes.
  - 3 features, all discretized into 4 buckets.

$$\langle \text{variance: } 4,\ \text{skewness: } 4,\ \text{curtosis: } 4 \rangle$$

- **California Housing**
  - Predict median house prices.
  - 2 key features (from 10).

$$\langle \text{population: } 4 \to 1,\ \text{median\_income: } 2 \to 1 \rangle$$

Datasets (2/3)

- **AutoTaxi Simulator (ICML benchmark)**
  - Synthetic RL environment: autonomous taxi decisions.
  - 3 state variables:

$$\langle \text{clouds: } 6,\ \text{day\_time: } 4,\ \text{init\_pos: } 4 \rangle$$

- **Iris (UCI)**
  - 3-class flower classification.
  - 3 attributes (each discretized into 4 buckets).

$$\langle \text{sepal\_length: } 4,\ \text{petal\_length: } 4,\ \text{petal\_width: } 4 \rangle$$

Datasets (3/3)

- **Loan Acquisition**
  - Predict loan behaviour.
  - 4 discretized features:

$$\langle \text{age: 4, monthly\_income: 2, dependents: 2, credit\_score: 3} \rangle$$

- **Theorem Prover Dataset**
  - Symbolic reasoning for automated theorem proving.
  - 2 features:

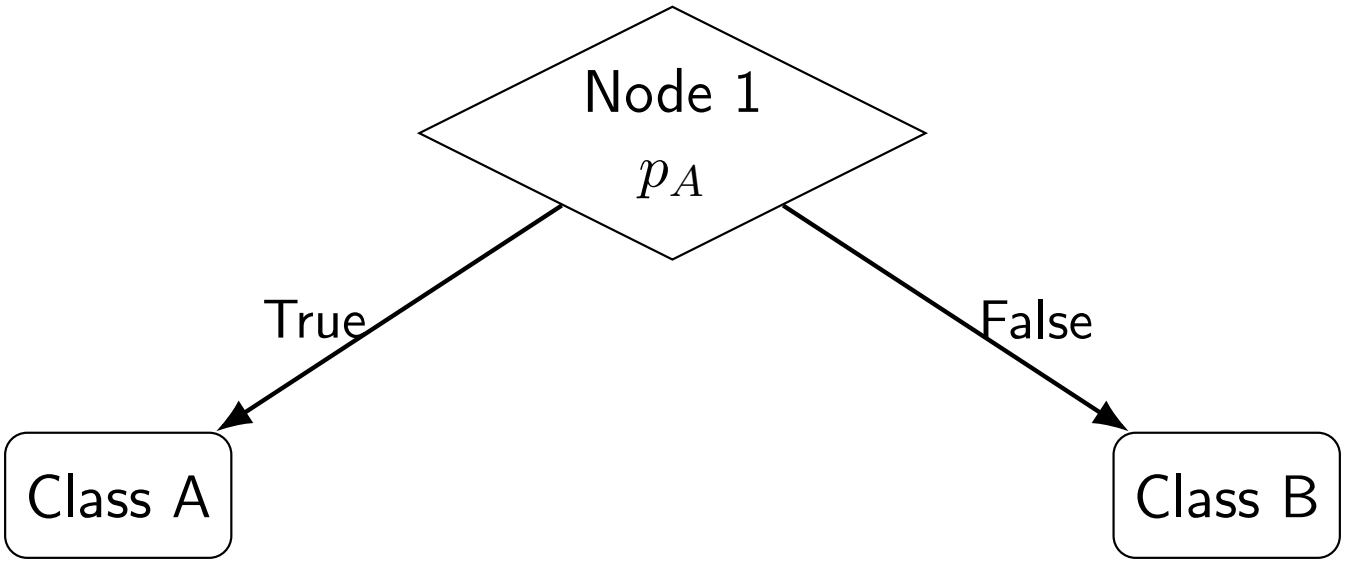
$$\langle \text{F1: 4, F10: 3} \rangle$$

All datasets are discretized into interpretable buckets, enabling compact decision diagrams.

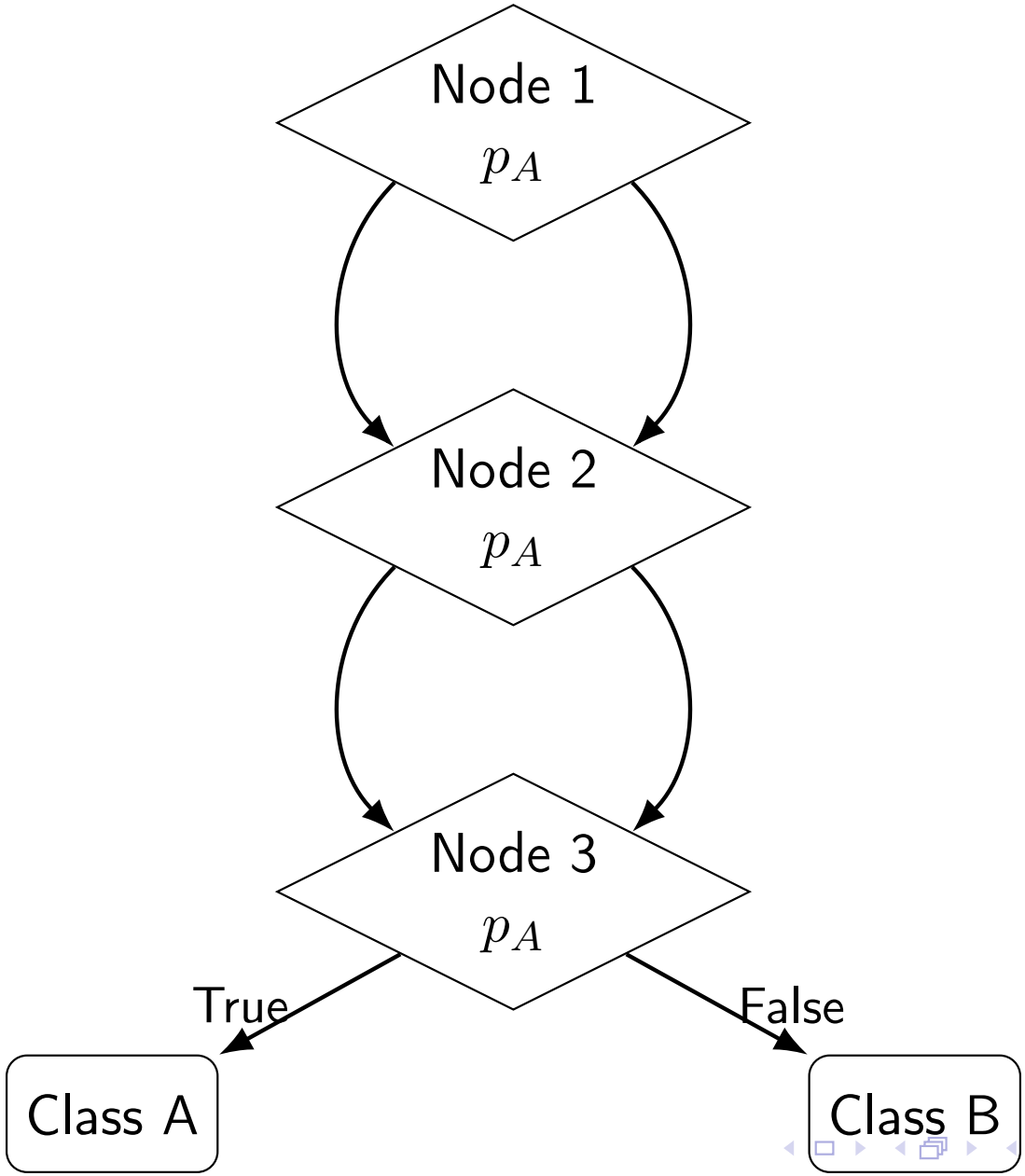
Redundancy Loophole

- If $w(p)$ is large and W_{unused} is small, solver can **repeat** a preferred predicate along a path.
- Example:
  - $w(p_A) = 11$, $W_{\text{unused}} = 10$, $k = 5$.
  - Compact tree: single p_A node, 4 unused nodes.
    Score: $(1 \times 11) + (4 \text{ unused} \times 10) = \mathbf{51}$.
  - Redundant tree: three p_A nodes, 2 unused nodes.
    Score: $(3 \times 11) + (2 \text{ unused} \times 10) = \mathbf{53}$.
- Redundant tree gets **higher** $\Delta_{\mathcal{E}}$ despite identical behaviour.

Compact Tree



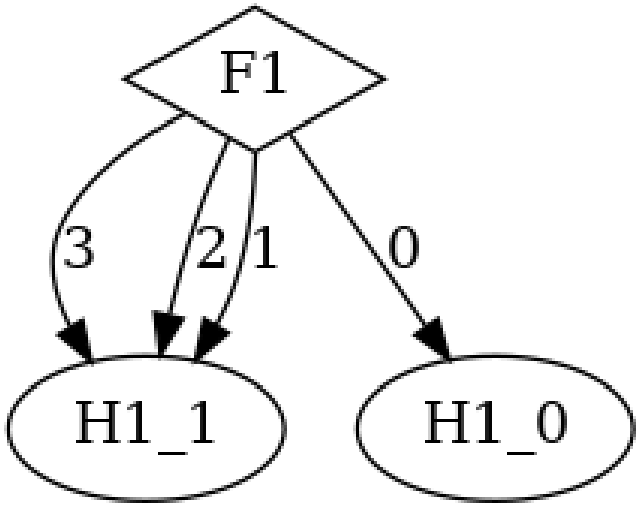
Redundant Tree



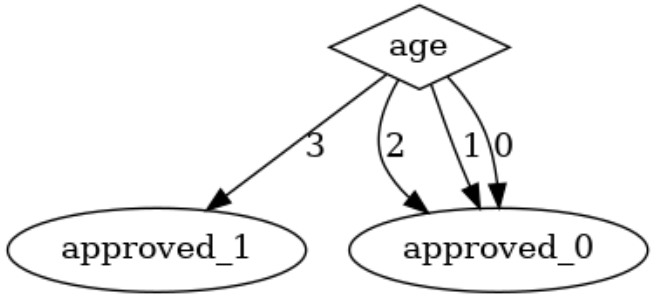
Original Fix: Aggressive Weight Scaling

- Paper sets
$$W_{\text{unused}} = \max_{p} w(p) + 1.$$
- Effect:
  - Any additional node has a negative net gain: $w(p) - W_{\text{unused}} \leq -1$.
  - Prevents predicate repetition.
- **Side effect:** Single-node degeneracy.
  - Penalty dominates correctness gains.
  - Learned DDs collapse to depth-1 trees on most datasets.
  - At that point, MaxSAT is overkill compared to a simple "best single split" scan.

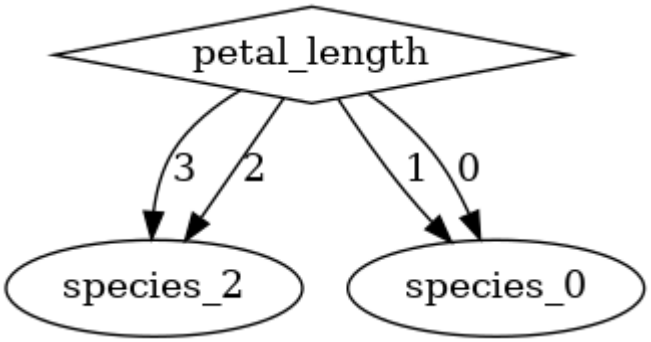
Results (1/2)



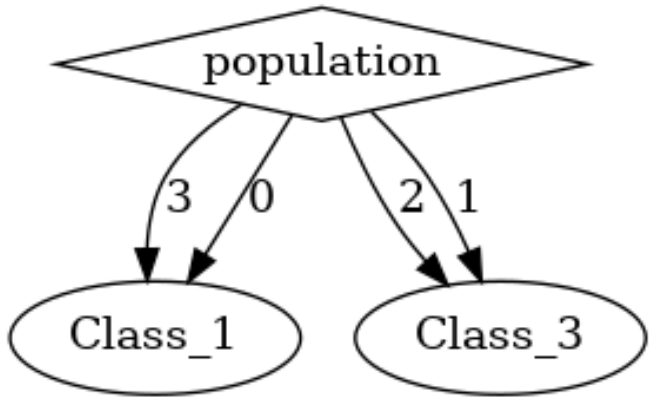
(a) Theorem Prover



(b) Loan Acquisition

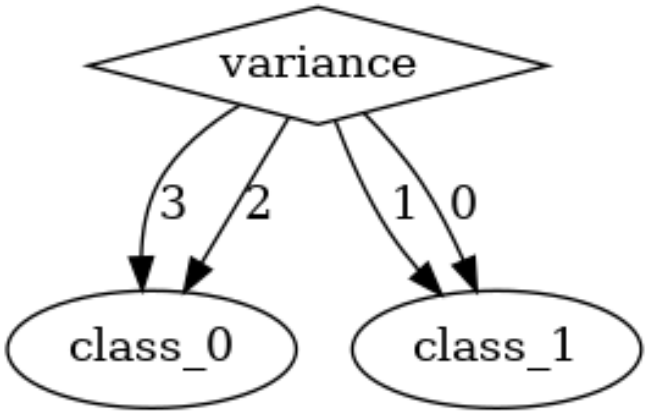


(c) Iris

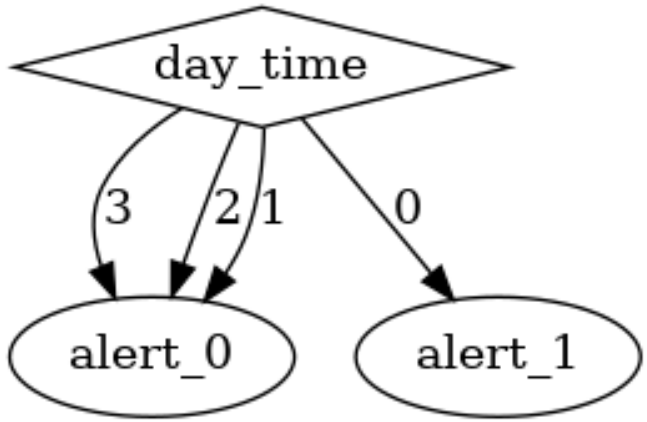


(d) California Housing

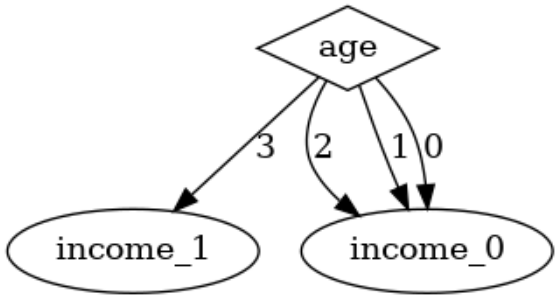
Results (2/2)



(a) Banknote



(b) Autotaxi

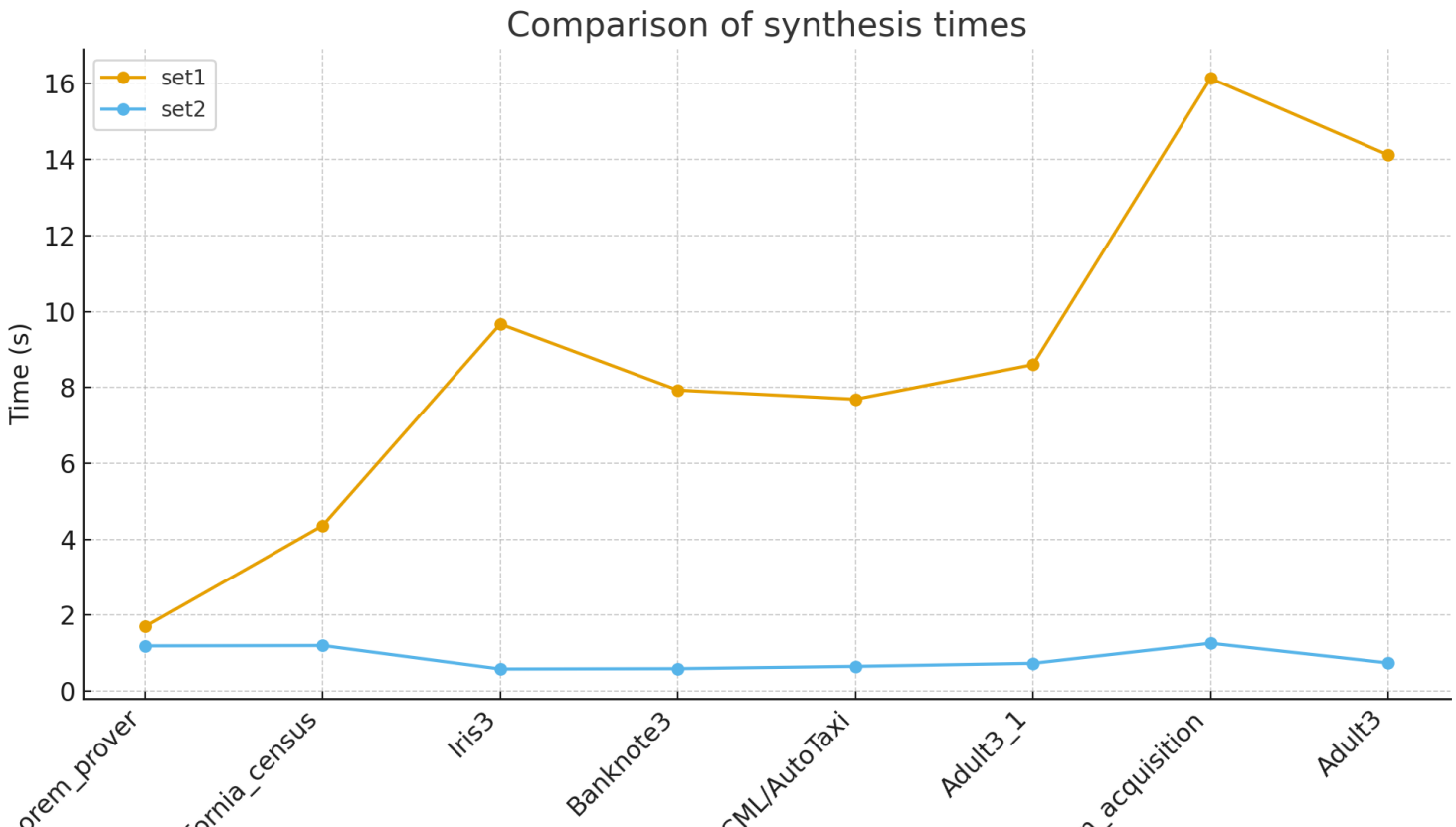


(c) Adult Income

Figure: Plots (e)-(g). We can clearly see that this formulation typically drives the solver toward the Pareto point that employs only a single node, even though up to three nodes were allowed.

Empirical Evidence

- Across 7 datasets (Adult, Banknote, California, AutoTaxi, Iris, Loan, Theorem Prover), the MaxSAT solution with $W_{\text{unused}} = \max w(p) + 1$ almost always uses 1 node.
- Simple $O(|P| \cdot |S|)$ scan over features is faster than MaxSAT in this regime.



Hard Non-Repetition via Reachability

Idea: Enforce structural non-repetition as a hard constraint instead of via weights.

- Introduce reachability variables $R_{i,j}$:

$$R_{i,j} \iff \bigvee_{c} \tau_{i,c,j} \ \vee \bigvee_{k} (R_{i,k} \wedge R_{k,j}).$$

- Non-repetition constraint:

$$\forall p, \forall i < j : \ (\lambda'_{i,p} \wedge R_{i,j}) \Rightarrow \neg \lambda'_{j,p}.$$

- Then we can relax W_{unused} (no need for "max+1 hack").

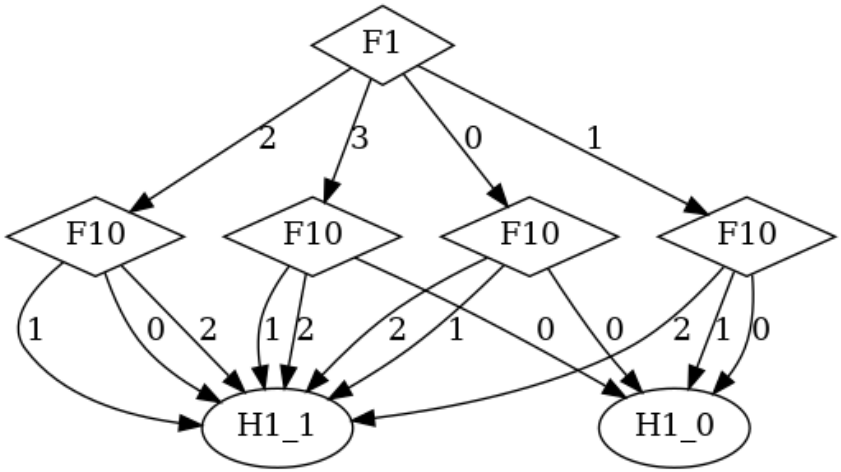
New Weight Schemes

- With non-repetition enforced, we can use:

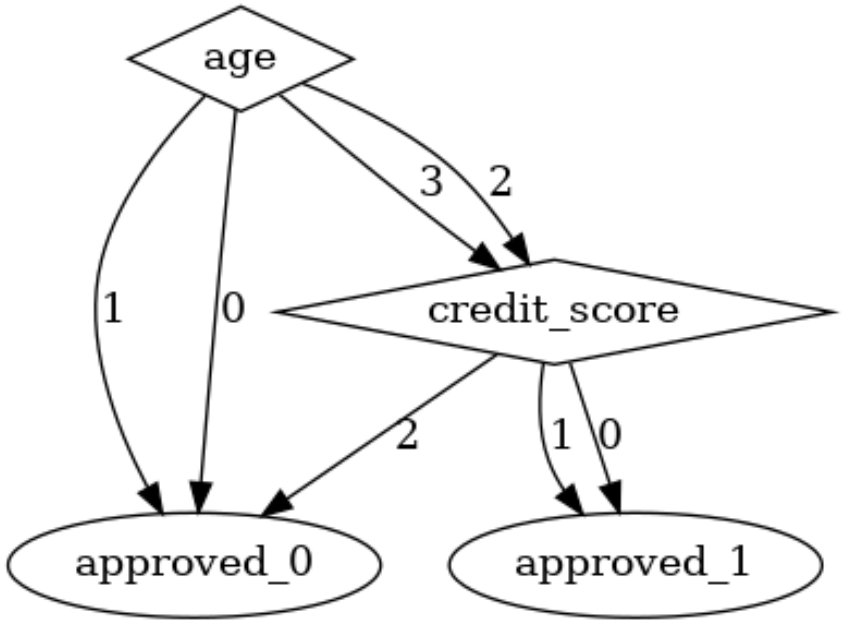
$$W_{\text{unused}} = \frac{1}{|P|} \sum_{p} w(p)$$

- Effect:
  - Trees can grow beyond depth 1 when it improves $\Delta_{\mathcal{C}}$.
  - Still a mild preference for compactness.
- But reachability + non-repetition increases encoding size which leads to a significant increase in the computation time as we shall see.

Results (1/4)



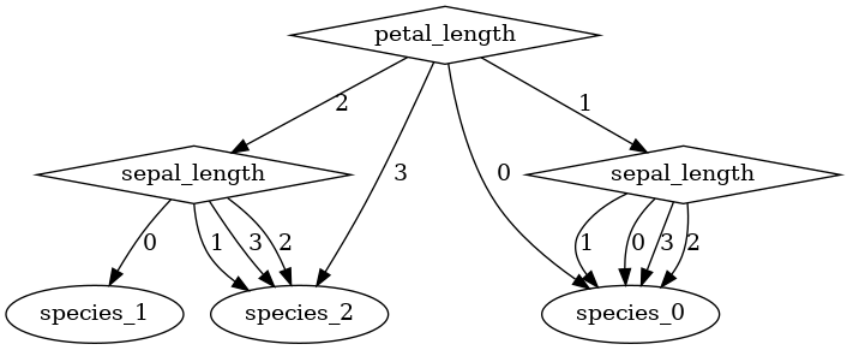
(a) Theorem Prover



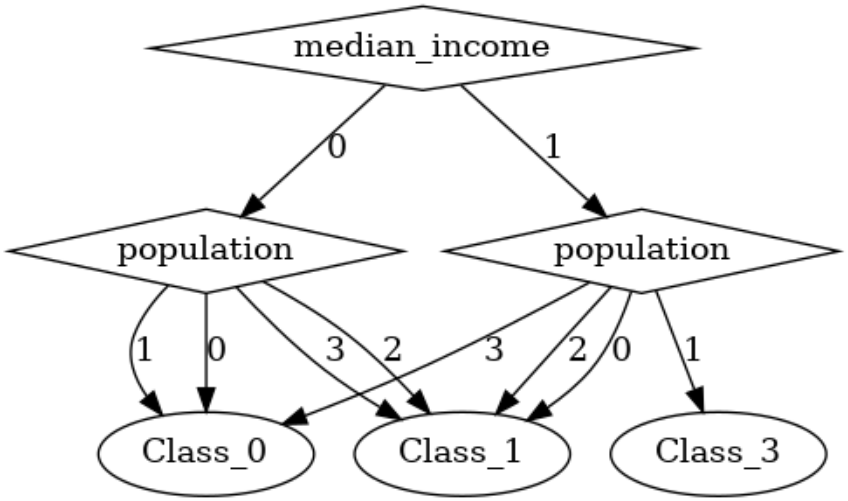
(b) Loan Acquisition

Figure: Plots (a)–(b) for the updated weighting scheme (used nodes $u_i = 1$).

Results (2/4)



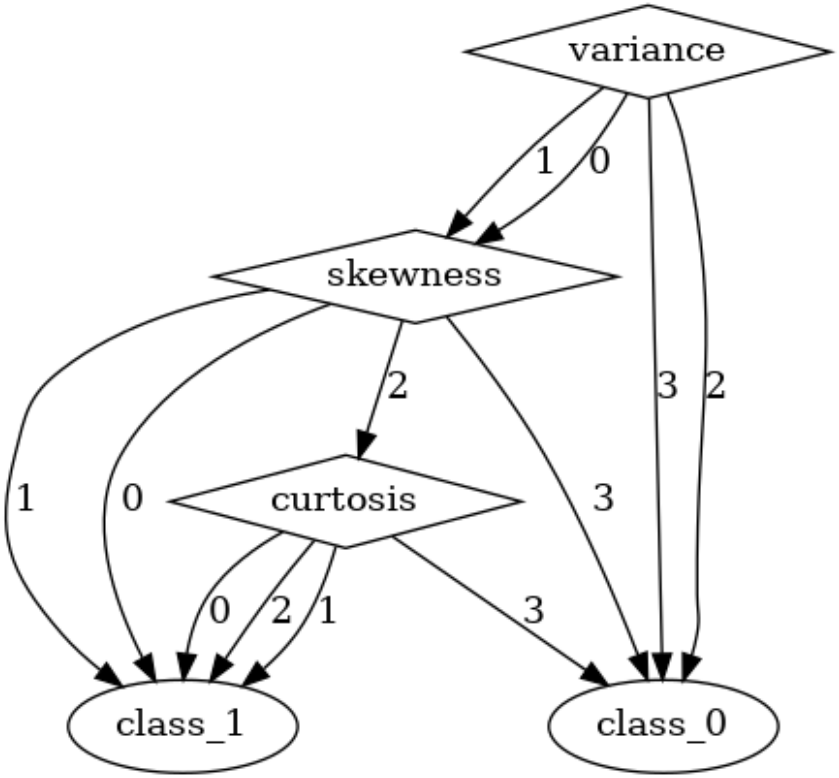
(a) Iris



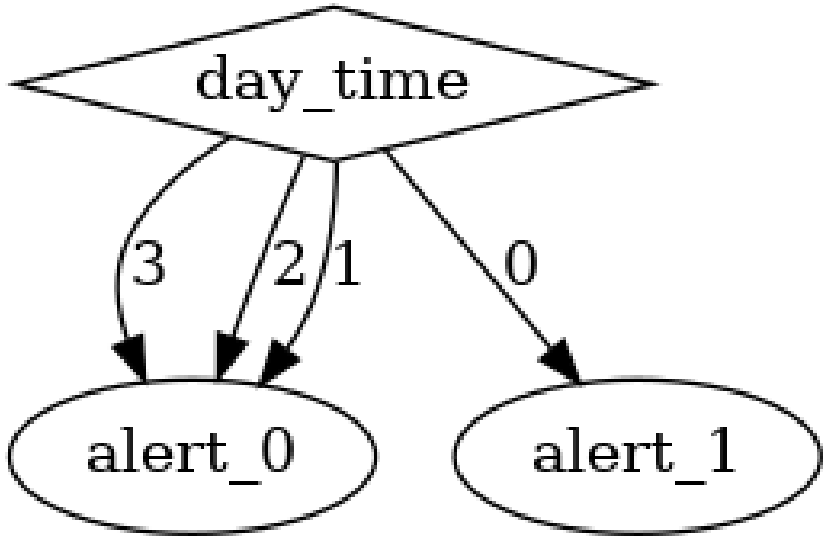
(b) California Housing

Figure: Plots (c)–(d) for the updated weighting scheme.

Results (3/4)



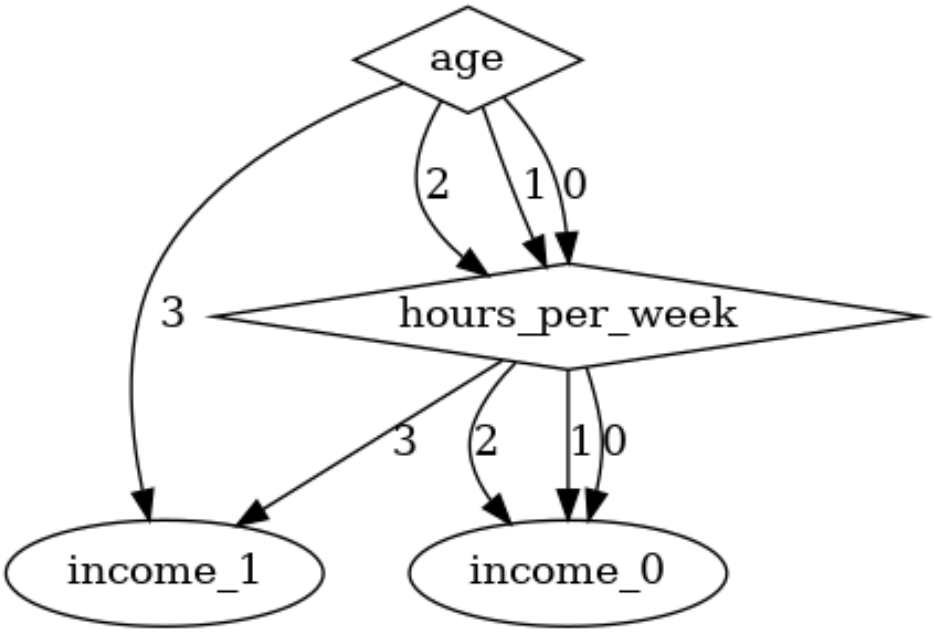
(a) Banknote



(b) Autotaxi

Figure: Plots (e)–(f). Autotaxi is the main exception where node usage does not increase.

Results (4/4)



(a) Adult Income

Figure: Plot (g). Overall, most datasets now use more nodes, but some redundancy appears (see Iris in Fig. 4a).

Shortcomings of Mean-Based Weighting

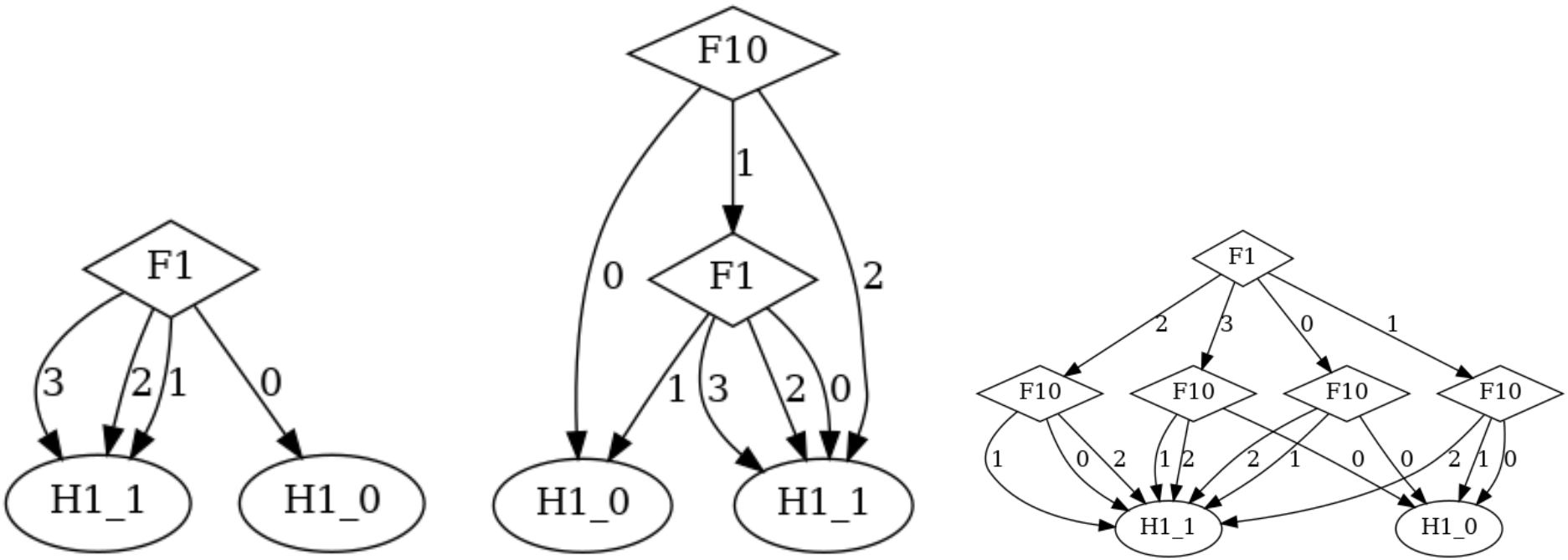
- **Previous fix:** Set the unused-node reward to the *mean* feature weight.
  - This avoids the extreme penalty of the $\max w + 1$ scheme.
  - But it can still introduce **structural redundancy**.
- **Redundancy example (Fig. 4a):**
  - A right `sepal_length` node is redundant
  - It does not improve correctness but is kept because its feature weight $w(p)$ is slightly $>$ mean.
  - Mean-based weighting thus **prevents collapse** to a single node, but may **still incentivize unnecessary nodes**.
- **Key issue:** Mean-based scheme treats "using a node" and "not using a node" as nearly equivalent.
- **Proposed ε-method:**

$$w_{\text{unused}} = \max(\text{feature\_weights}) + \varepsilon \cdot \text{mean}(\text{feature\_weights}),$$

with small ε (e.g. $\varepsilon = 0.1$).

  - Keeps a modest advantage for unused nodes.
  - Avoids both extremes: single-node collapse and redundancy

Effect of Different Weighting Schemes



(a) max_weight + 1 (b) ε-method (0.1) (c) Mean-based weight

Figure: Comparison of unused-node weighting schemes. The ε-method reaches a middle ground between trivial one-node trees (max_weight+1) and redundant structures induced by pure mean-based weighting. This is for theorem prover dataset.

Additional Structural Constraint (Impractical in Practice)

- We also explored a **hard constraint** to rule out nodes whose use is pointless.
- Target: nodes where *all* outgoing transitions go to the *same* successor, i.e. the split has no real effect.

$$\lambda'_{i,p} \implies \neg \left(\bigwedge_{c=1}^{\max(B(p))} \tau_{i,c,j} \right) \qquad \text{for all } j.$$

- Intuition:
  - If node i tests predicate p ($\lambda'_{i,p} = 1$),then it should **not** be the case that every branch c leads to the same child j.
- **Problem:** The constraint is very expensive:
  - Quantifies over all successors j and all buckets c.
  - In practice, makes solving significantly slower and almost impractical.
- Takeaway:
  - The ε weighting is a more practical way to reduce redundancy.
  - Heavy structural constraints give cleaner diagrams but at a high computational cost.

Runtime Impact of Reachability(1/2)

Dataset	Default method [s]	Reachability constraint [s]
theorem_prover	1.70	1.71
california_census	4.36	4.35
Iris3	9.67	9.47
Banknote3	8.06	17.11
ICML/AutoTaxi	7.69	600+(timeout)
Adult3_1	8.60	46.25
loan_acquisition	16.63	207.83

Table: Running time for maxsat in the default scenario and after introducing the reachability constraint. The increased computation time on the introduction of reachability constraints is more evident in the case of larger data sets, which have more features. This suggests an increase in computational complexity.

Runtime Impact of Reachability(1/2)

Dataset	Default method [s]	Reachability constraint [s]
theorem_prover	2.10	10.53
california_census	4.40	4.60
Iris3	9.77	10.67
Banknote3	8.30	10.58
ICML/AutoTaxi	8.55	26.18
Adult3_1	9.40	12.21
loan_acquisition	16.87	193.87

Table: Running time for Naive MILP in the default scenario and after introducing the reachability constraint. The increased computation time on the introduction of reachability constraints is more evident in the case of larger data sets, although it is significantly less than that in the case of the default method.

Weighted MaxSAT and MILP

- **Weighted MaxSAT:** Given a CNF formula with hard and soft clauses, each soft clause j has a weight w_j.
- Goal: maximize the total weight of satisfied soft clauses (or equivalently minimize the total weight of violated ones).
- **MILP encodings:** Represent Boolean variables and clauses using linear constraints over binary variables.
- We compare several MILP encodings for clauses in Weighted MaxSAT:
  - Naive slack-based encoding
  - Extended formulation (indicator-based)
  - Pseudo-Boolean strengthened encoding
  - Quadratic MILP with McCormick linearization

Naive Slack-Based Encoding: Idea

- Consider a clause C_j (a disjunction of literals).
- In SAT, a clause is satisfied if

$$\sum_{l \in C_j} x_l \geq 1,$$

  where $x_l \in \{0, 1\}$ is the truth value of literal l.
- For **hard** clauses, this constraint must always hold.
- For **soft** clauses in Weighted MaxSAT, violations are allowed but penalized.
- **Idea:** Introduce a slack variable $s_j \in \{0, 1\}$ indicating whether clause j is violated.

Naive Slack-Based Encoding: Formulation

- For each soft clause C_j, introduce binary slack variable s_j:

$$\sum_{l \in C_j} x_l + s_j \geq 1.$$

- Interpretation:
  - If all literals are false ($\sum_{l \in C_j} x_l = 0$), constraint forces $s_j = 1$.
  - If the clause is already true ($\sum_{l \in C_j} x_l \geq 1$), s_j can be 0.
- Objective:

$$\text{Minimize} \quad Z = \sum_{j \in \text{Soft}} w_j \, s_j.$$

- Works with standard MILP solvers, but treats violation as a simple additive patch.

Extended (Indicator-Based) Encoding: Idea

- Introduce a binary indicator y_j to represent whether clause C_j is satisfied.

$$y_j \in \{0, 1\}, \quad y_j = 1 \iff C_j \text{ is satisfied.}$$

- Use inequalities to *bound* y_j from above and below in terms of the literals.
- This gives a stronger relaxation than a slack constraint.

Extended Encoding: Mathematical Formulation

- Let C_j be a clause with $k_j = |C_j|$ literals.
- Binary decision variable $y_j \in \{0, 1\}$: clause satisfaction.

Upper bound (Implication of falsity):

$$y_j \leq \sum_{l \in C_j} x_l.$$

- If all literals are false ($\sum x_l = 0$), then $y_j \leq 0$, forcing $y_j = 0$.

Lower bound (Relaxation tightening):

$$y_j \geq \frac{1}{k_j} \sum_{l \in C_j} x_l.$$

- In integer solutions, this does not change feasibility.
- In the LP relaxation, it ties y_j more tightly to the literals.

Extended Encoding: Objective

- We penalize unsatisfied soft clauses via $1 - y_j$:

$$\text{Minimize} \quad Z = \sum_{j \in \text{Soft}} w_j(1 - y_j).$$

- $y_j = 1$ (clause satisfied) $\Rightarrow$ no cost.
- $y_j = 0$ (clause violated) $\Rightarrow$ cost w_j.
- Compared to slack-based encoding:
  - Explicit satisfaction variable y_j is structurally meaningful.
  - Both upper and lower bounds strengthen the LP relaxation.

Extended Encoding: Example Clause

Example: $C_1 = (x_1 \vee x_2 \vee x_3)$ with weight w.

- Variables: x_1, x_2, x_3 and indicator y_1.
- Constraints:

$$y_1 \leq x_1 + x_2 + x_3$$

$$y_1 \geq \frac{1}{3}(x_1 + x_2 + x_3).$$

- LP relaxation effect:
  - Suppose $x_1 = 0.3, x_2 = 0, x_3 = 0$.
  - Then $y_1 \geq 0.1$ from the lower bound.
  - The cost associated with y_1 is more tightly linked to x_1, x_2, x_3 than in a naive encoding.

PB-Strengthened Encoding: Idea

- Based on J.N. Hooker's suggestion in *Mixed logical-linear programming*.
- Start from a standard Pseudo-Boolean (PB) style constraint:

$$\sum_{l \in C_j} x_l + r_j \geq 1,$$

  where r_j is a relaxation (violation) variable.
- Add a **redundant** but strengthening constraint on falsified literals:

$$\sum_{l \in C_j} \bar{x}_l \leq (k_j - 1) + k_j r_j,$$

  where $\bar{x}_l = 1 - x_l$.
- This is a cover-type inequality, tightening the LP relaxation.

PB-Strengthened Encoding: Formulation

- For each clause C_j (length k_j):

  Variables: $x_l \in \{0,1\},\ \bar{x}_l = 1 - x_l,\ r_j \in \{0,1\}.$

  Satisfaction lower bound: $$\sum_{l \in C_j} x_l + r_j \geq 1.$$

  Falsity upper bound: $$\sum_{l \in C_j} \bar{x}_l \leq (k_j - 1) + k_j r_j.$$

- Objective:

$$\text{Minimize} \quad Z = \sum_{j \in \text{Soft}} w_j r_j.$$

PB-Strengthened Encoding: Geometric Perspective

- Over binary domain ($x_l \in \{0, 1\}$), the two constraints are logically redundant.
- Over continuous domain ($x_l \in [0, 1]$), they define different half-spaces.
- Their intersection:
  - Cuts off fractional corners of the polytope.
  - Approximates the convex hull of valid clause assignments more closely.

PB-Strengthened Encoding: Algorithmic Advantage

- The falsity upper bound has **Knapsack** structure:

$$\sum_{l \in C_j} \bar{x}_l \leq (k_j - 1) + k_j r_j.$$

- Modern solvers (Gurobi, CPLEX) detect knapsack constraints and apply:
  - Cover cuts
  - Lifted cover inequalities
  - Other specialized cut-generation routines
- This leads to:
  - Stronger cutting planes
  - Tighter root-node bounds
  - More effective presolve and branching
- Overall, PB-strengthened encoding exposes more exploitable structure than a naive row.

Quadratic MILP Encoding: Idea

- Instead of representing satisfaction via sums, explicitly encode clause **violation** as a product.
- For clause C_j:
  - Introduce unsatisfaction indicators $u_{j,l}$ for each literal l.
  - Clause is violated iff *all* literals are false:

$$P_j = \prod_{l \in C_j} u_{j,l}.$$

- Binary variable y_j indicates satisfaction:

$$y_j + P_j = 1.$$

- Use McCormick linearization to handle the product in an MILP framework.

Quadratic Encoding: Unsatisfaction Indicators

- For each literal l in clause C_j:

$$u_{j,l} = \begin{cases} 1 - x_i & \text{if } l = x_i \text{ (positive literal)}, \\ x_i & \text{if } l = \neg x_i \text{ (negative literal)}. \end{cases}$$

- $u_{j,l} = 1$ if literal l is false under assignment x.
- Clause violation product:

$$P_j = \prod_{l \in C_j} u_{j,l}.$$

- Satisfaction link:

$$y_j + P_j = 1 \quad \Rightarrow \quad \begin{cases} P_j = 1 \Rightarrow y_j = 0 \text{ (clause violated)} \\ P_j = 0 \Rightarrow y_j = 1 \text{ (clause satisfied)} \end{cases}$$

McCormick Linearization

- For binary variables a, b and product $z = a \cdot b$, McCormick envelope:

$$z \leq a, \quad z \leq b, \quad z \geq a + b - 1.$$

- For a clause with k literals:
  - Introduce $k - 1$ auxiliary variables to compute the product sequentially.
  - Each multiplication adds 3 linear constraints.
  - Total of $3(k - 1)$ linear constraints per clause.

Quadratic Encoding: Example

Example: $C_1 = (x_1 \vee \neg x_2 \vee x_3)$.

- Unsatisfaction indicators:

$$u_1 = 1 - x_1, \quad u_2 = x_2, \quad u_3 = 1 - x_3.$$

- Compute product $P = u_1 u_2 u_3$ via:
  1. Intermediate product $z_1 = u_1 u_2$:

$$z_1 \leq u_1, \quad z_1 \leq u_2, \quad z_1 \geq u_1 + u_2 - 1.$$

  2. Final product $P = z_1 u_3$:

$$P \leq z_1, \quad P \leq u_3, \quad P \geq z_1 + u_3 - 1.$$

- Satisfaction link:

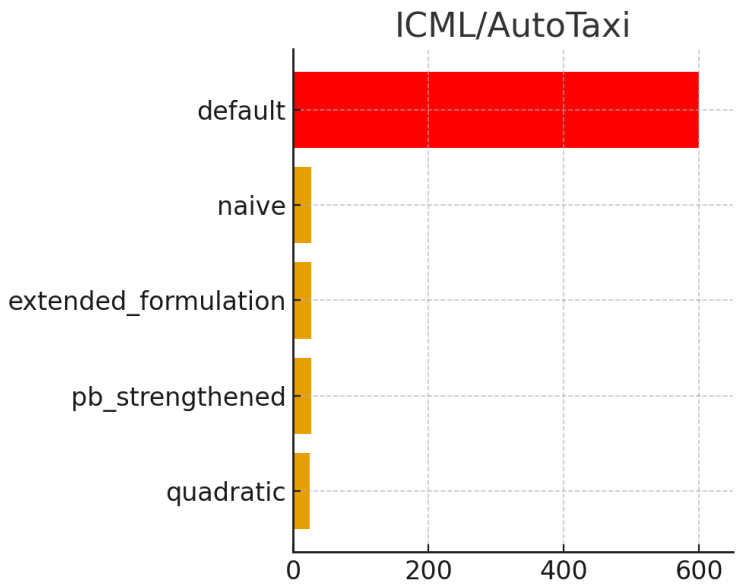
$$y_1 + P = 1.$$

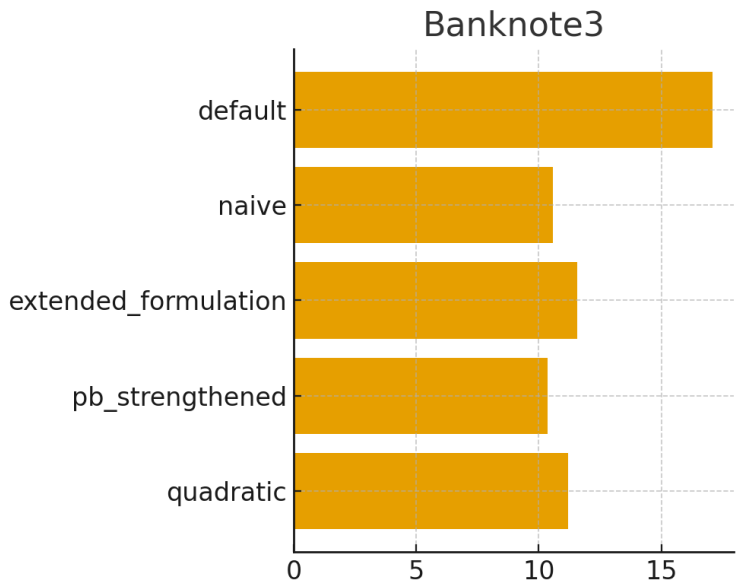
Runtime Comparison: Selected Methods

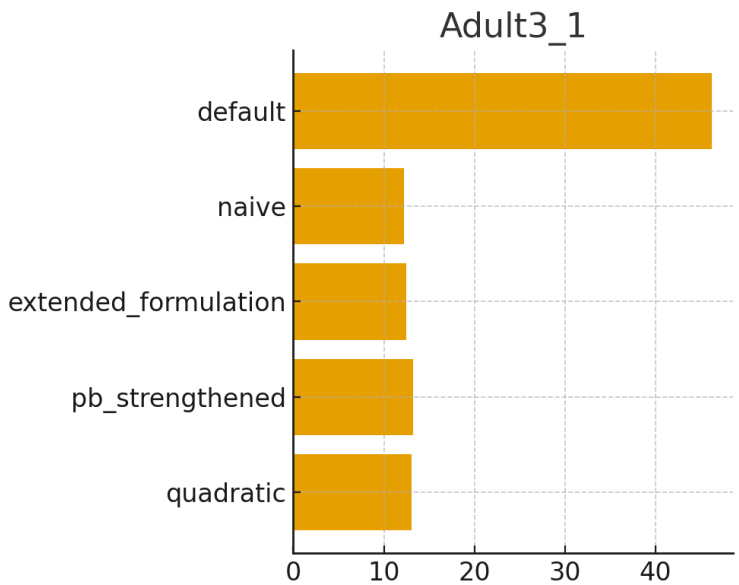
Dataset	Default	Naive MILP	Extended	PB-Strength.	Quadratic
theorem_prover	1.71	10.53	11.26	12.07	10.64
california_census	4.35	4.60	4.52	4.50	4.54
Iris3	9.47	10.67	10.79	10.91	10.72
Banknote3	17.11	10.58	10.53	10.52	10.53
ICML/AutoTaxi	600.00*	26.18	24.32	24.02	24.10
Adult3_1	46.25	12.21	12.48	13.24	13.04
loan_acquisition	207.83	193.87	204.58	174.62	178.33

*Timed out at 600 seconds.

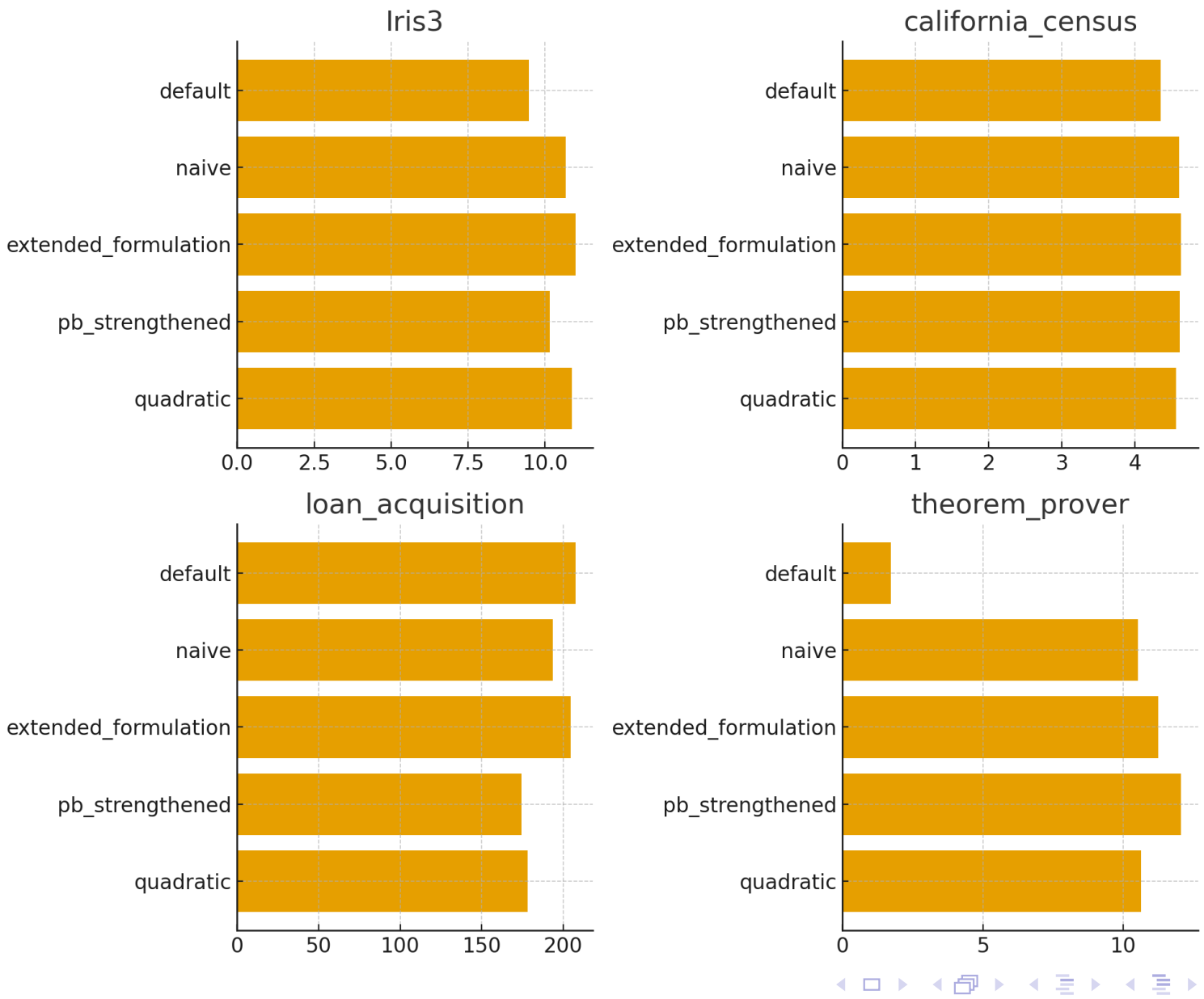
Runtime Comparison (Reachability + New Weights)







Runtime Comparison (Reachability + New Weights)



Results: Interpretation

- **Default MaxSAT** is very efficient under the original encoding.
- With added reachability constraints:
  - Default MaxSAT performance degrades sharply.
  - AutoTaxi instance times out at 600 seconds.
- MILP encodings:
  - Slightly slower on small benchmarks.
  - Scale more consistently as structural constraints are added.
- PB-strengthened and Quadratic encodings:
  - Achieve the best overall performance among MILP variants.
  - Substantially reduce solve times on larger datasets (Adult, Loan).

Summary and Takeaways

- Several MILP encodings can be used for Weighted MaxSAT clauses:
  - Naive slack-based
  - Extended indicator-based
  - PB-strengthened
  - Quadratic (with McCormick linearization)
- Strengthening formulations by:
  - Adding indicator variables and bounds
  - Exploiting knapsack-like structure
  - Explicitly modeling clause violation via products

  leads to tighter LP relaxations.
- In experiments:
  - Default MaxSAT is sensitive to additional structural constraints.
  - MILP-based approaches, especially PB-strengthened and quadratic, provide a robust alternative.