Decision Diagrams for Constraint Reasoning and Optimization

Willem-Jan van Hoeve
Carnegie Mellon University

CP 2026

Motivation

- Binary Decision Diagrams
  - Fundamental data structure that is widely used in computer science
  - Compact representation of Boolean functions
- How can decision diagrams be used for combinatorial optimization?
  - Constraint programming, dynamic programming, integer programming, …

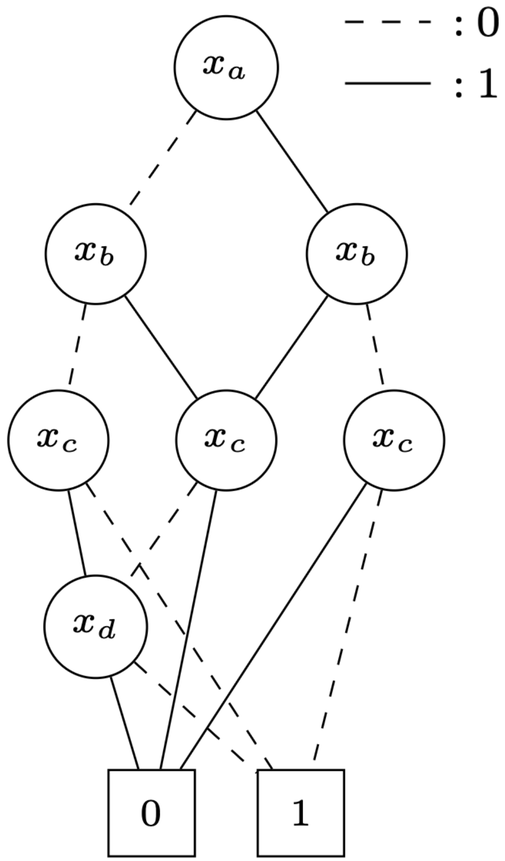
Background references:

- Bergman, Cire, van Hoeve, Hooker. Decision Diagrams for Optimization. *Springer*, 2016.
- Van Hoeve. An Introduction to Decision Diagrams for Optimization. *INFORMS*, 2024.

Decision Diagrams Compactly Represent Boolean Functions

$$f(x) = (\neg x_a \vee \neg x_c) \wedge (\neg x_b \vee \neg x_c) \wedge (\neg x_b \vee \neg x_d) \wedge (\neg x_c \vee \neg x_d)$$

x_a	x_b	x_c	x_d	$f(x)$
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	1	0	0	1
1	0	0	1	1
0	0	1	1	0
⋮	⋮	⋮	⋮	⋮
1	1	1	1	0



ROBDD: Reduced Ordered Binary Decision Diagram

ROBDD:

- Recursive ordered decomposition of $f(x)$
- Merge equivalent states (sub-functions)
- Remove redundant states

[Boole1854, Shannon1949, Lee1959, Akers1978, Bryant1986, Bryant1992, Minato1993,…]

Variants of Decision Diagrams

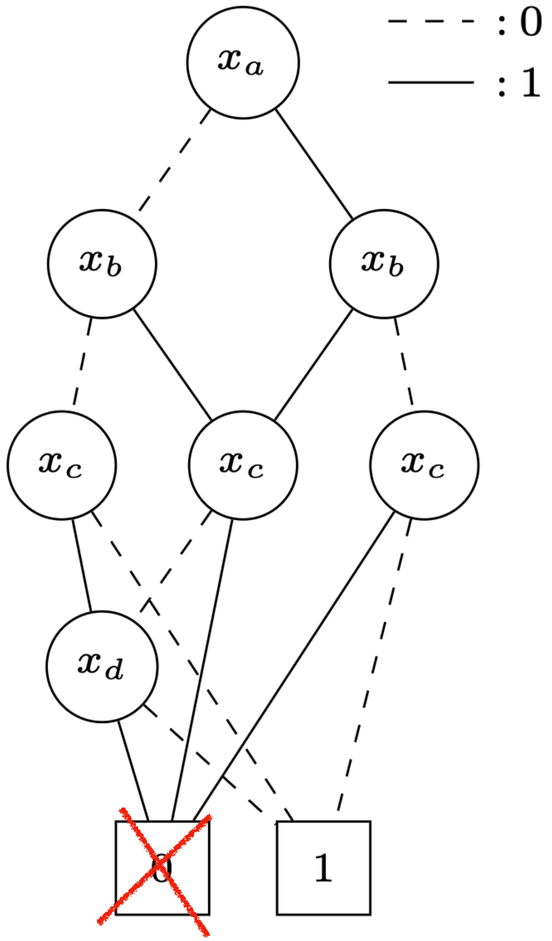
- Boolean function representation
  - Reduced Ordered Binary Decision Diagram [Akers1978, Bryant1992, Bryant1986,…]
  - SAT solving (verification problems), CP set constraints [Hawkins, Lagoon, Stuckey, 2005]
  - Sentential Decision Diagram for knowledge compilation [Darwiche&Marquis2002, Darwiche2011]
- Finite-domain function representation
  - Multi-valued Decision Diagram
  - Constraint propagation [Andersen et al. 2007, Cheng&Yap 2008,…]
  - Product configuration [Andersen et al. 2010, Berndt et al. 2012]
- Finite numeric function representation
  - Algebraic Decision Diagram (multiple terminals) [Bahar et al. 1993,…]
  - Weighted model counting [Dudek, Phan & Vardi 2020]

Optimization View of Decision Diagrams

$$f(x) = (\neg x_a \vee \neg x_c) \wedge (\neg x_b \vee \neg x_c) \wedge (\neg x_b \vee \neg x_d) \wedge (\neg x_c \vee \neg x_d)$$

Integer Program:

$$\begin{aligned} \text{s.t.} \quad & x_a + x_c \leq 1 \\ & x_b + x_c \leq 1 \\ & x_b + x_d \leq 1 \\ & x_c + x_d \leq 1 \\ & x_a, x_b, x_c, x_d \in \{0, 1\} \end{aligned}$$



Optimization view:

- Literals → binary variables
- Arcs → assignments
- Paths → solutions

Adapt representation:

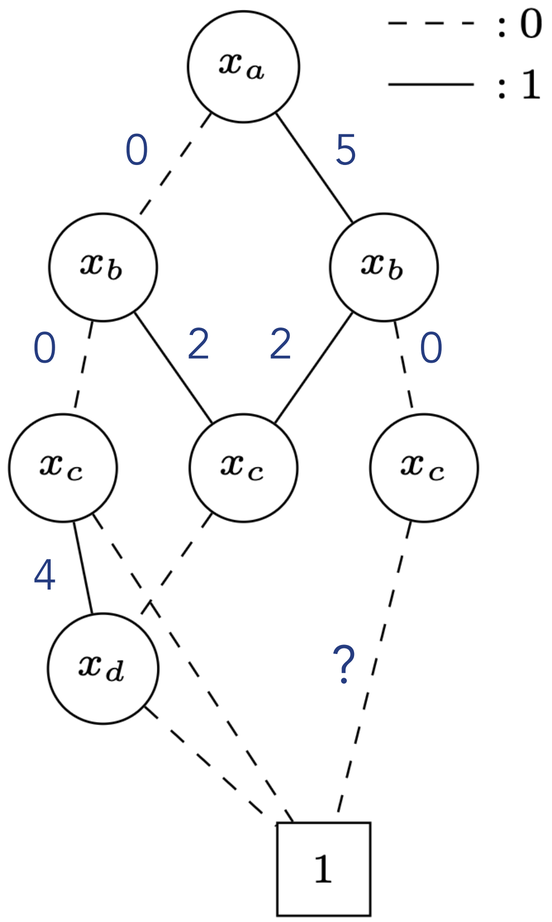
- No need for infeasible paths to 0-terminal

Optimization View of Decision Diagrams

$$f(x) = (\neg x_a \vee \neg x_c) \wedge (\neg x_b \vee \neg x_c) \wedge (\neg x_b \vee \neg x_d) \wedge (\neg x_c \vee \neg x_d)$$

Integer Program:

$$\begin{aligned} \max \quad & 5x_a + 2x_b + 4x_c + 3x_d \\ \text{s.t.} \quad & x_a + x_c \leq 1 \\ & x_b + x_c \leq 1 \\ & x_b + x_d \leq 1 \\ & x_c + x_d \leq 1 \\ & x_a, x_b, x_c, x_d \in \{0, 1\} \end{aligned}$$



Optimization view:

- Literals → binary variables
- Arcs → assignments
- Paths → solutions

Adapt representation:

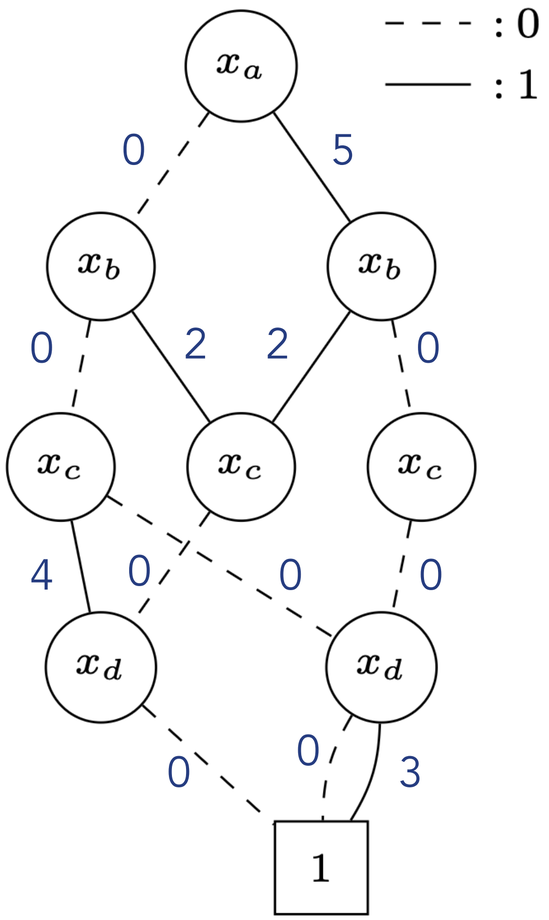
- No need for infeasible paths to 0-terminal
- Represent weights — may need redundant states

Optimization View of Decision Diagrams

$$
\begin{aligned}
f(x) = \; & (\neg x_a \vee \neg x_c) \wedge \\
& (\neg x_b \vee \neg x_c) \wedge \\
& (\neg x_b \vee \neg x_d) \wedge \\
& (\neg x_c \vee \neg x_d)
\end{aligned}
$$

Integer Program:

$$
\begin{aligned}
\max \quad & 5x_a + 2x_b + 4x_c + 3x_d \\
\text{s.t.} \quad & x_a + x_c \leq 1 \\
& x_b + x_c \leq 1 \\
& x_b + x_d \leq 1 \\
& x_c + x_d \leq 1 \\
& x_a, x_b, x_c, x_d \in \{0, 1\}
\end{aligned}
$$



Optimization view:

- Literals → binary variables
- Arcs → assignments
- Paths → solutions

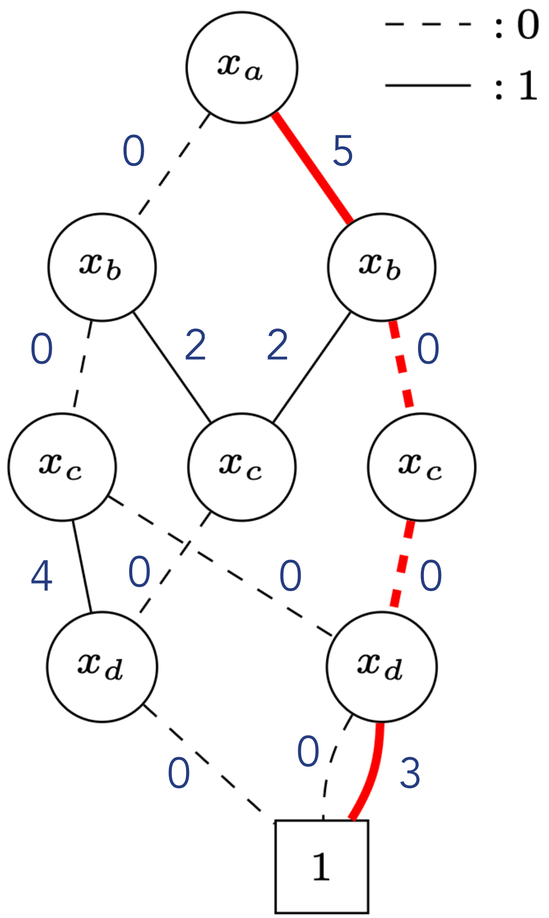
Adapt representation:

- No need for infeasible paths to 0-terminal
- Represent weights — may need redundant states

Optimization View of Decision Diagrams

Integer Program:

$$
\begin{aligned}
\max \quad & 5x_a + 2x_b + 4x_c + 3x_d \\
\text{s.t.} \quad & x_a + x_c \leq 1 \\
& x_b + x_c \leq 1 \\
& x_b + x_d \leq 1 \\
& x_c + x_d \leq 1 \\
& x_a, x_b, x_c, x_d \in \{0, 1\}
\end{aligned}
$$



Optimization:

- Find longest path from root to terminal
- Linear time complexity

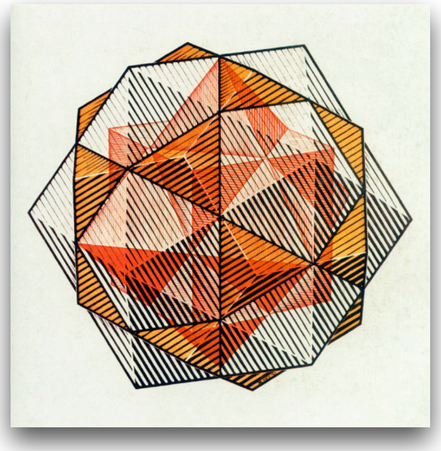
(DALL-E, 2024)

Dynamic Programming

(Durer, 1514)

Constraint Programming

(Escher, 1961)



Integer Programming

(DALL-E, 2024)

Column Elimination

(DALL-E, 2024)

Dynamic Programming

Dynamic Programming for Optimization

- Dynamic Programming is…
  - Recursion (Bellman's equation)
  - An algorithm (Dijkstra's shortest path algorithm)
  - Tabular computation (Needleman-Wunsch for sequence alignment)
- Often used for combinatorial optimization problems
  - Knapsack problem, lot sizing, vehicle routing, machine scheduling,…
  - Requires dedicated implementation!
- Opportunity: **Model+Solve** approach like CP, SAT, MIP
  - Generic solver can embed efficient techniques from various domains

[Bergman,Cire,vH,Hooker IJOC 2016] [Gillard, Schaus, Coppé IJCAI2020]
[Kuroiwa,Beck ICAPS 2023] [Michel,vH ECAI2024+]

Compiling Decision Diagram from Dynamic Program

- Input model: Dynamic Program
  - A set of *states* $\mathcal{S}$ with initial state r and target state t
  - A *label generation function* $\lambda : \mathcal{S} \to 2^{\mathcal{U}}$ representing decisions ($\mathcal{U}$ is the universe of decisions)
  - A state *transition function* $\tau : \mathcal{S} \times \mathcal{U} \to \mathcal{S}$
  - A transition *cost function* $c : \mathcal{S} \times \mathcal{U} \to \mathbb{R}$

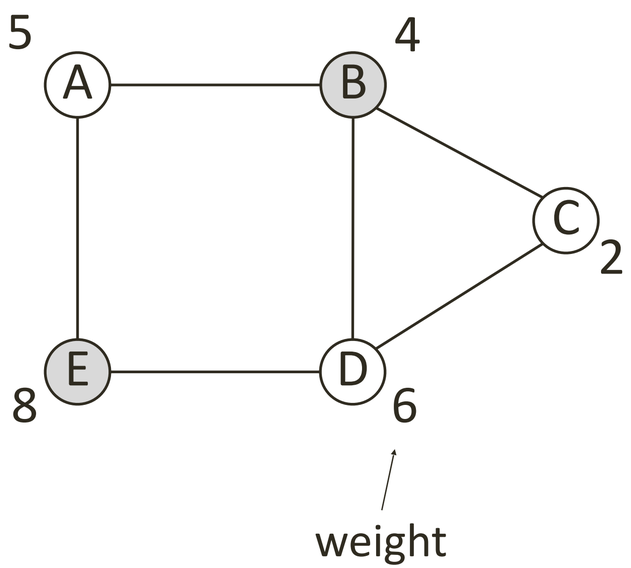
- Optimization problem:

$$\begin{aligned} \max \quad & g(t) \\ \text{s.t.} \quad & g(r) = K \\ & g(s') = \max_{\substack{s \in \mathcal{S}, \ell \in \lambda(s) \\ \tau(s,\ell) = s'}} g(s) + c(s,\ell) \quad \forall s' \in \mathcal{S} \setminus \{r\} \end{aligned}$$

where $g : \mathcal{S} \to \mathbb{R}$ is the state value function

This is the 'forward' form of the Bellman recursion, used for DD compilation

Example Application: Independent Set Problem



Independent set in a graph:

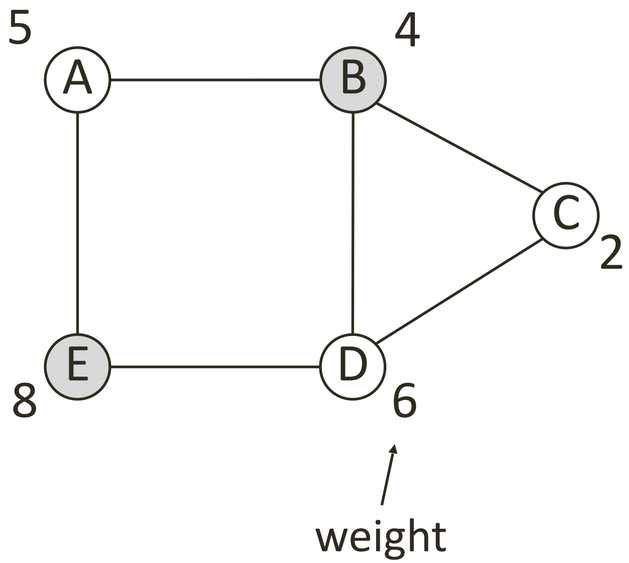
- Subset of non-adjacent vertices

Maximum Independent Set Problem:

- Find independent set with maximum weight

- Classical combinatorial optimization problem (equivalent to maximum clique in complement graph)
- Wide applications, ranging from scheduling to social network analysis
- *Not* typically solved with dynamic programming

Integer Programming Formulation



Independent set in a graph:

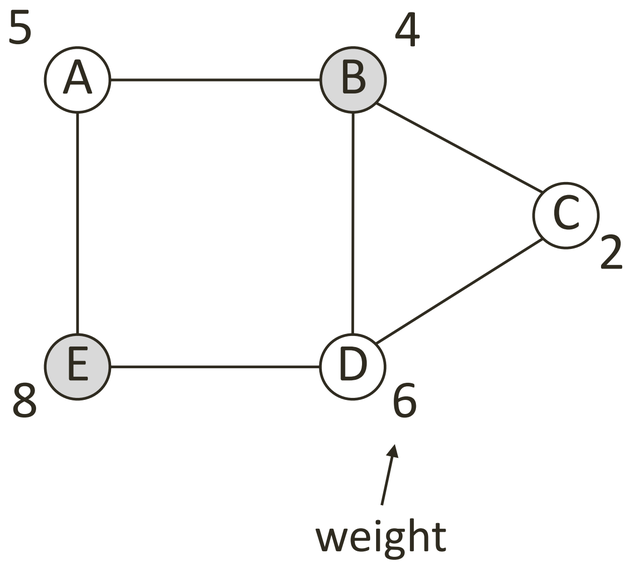
- Subset of non-adjacent vertices

Maximum Independent Set Problem:

- Find independent set with maximum weight

$$\begin{aligned}
\text{max} \quad & 5x_A + 4x_B + 2x_C + 6x_D + 8x_E \\
\text{subject to} \quad & x_A + x_B \leq 1 \\
& x_A + x_E \leq 1 \\
& x_B + x_C \leq 1 \\
& x_B + x_D \leq 1 \\
& x_C + x_D \leq 1 \\
& x_D + x_E \leq 1 \\
& x_A, x_B, x_C, x_D, x_E \in \{0,1\}
\end{aligned}$$

Dynamic Programming Formulation



Independent set in a graph:

- Subset of non-adjacent vertices

Maximum Independent Set Problem:

- Find independent set with maximum weight

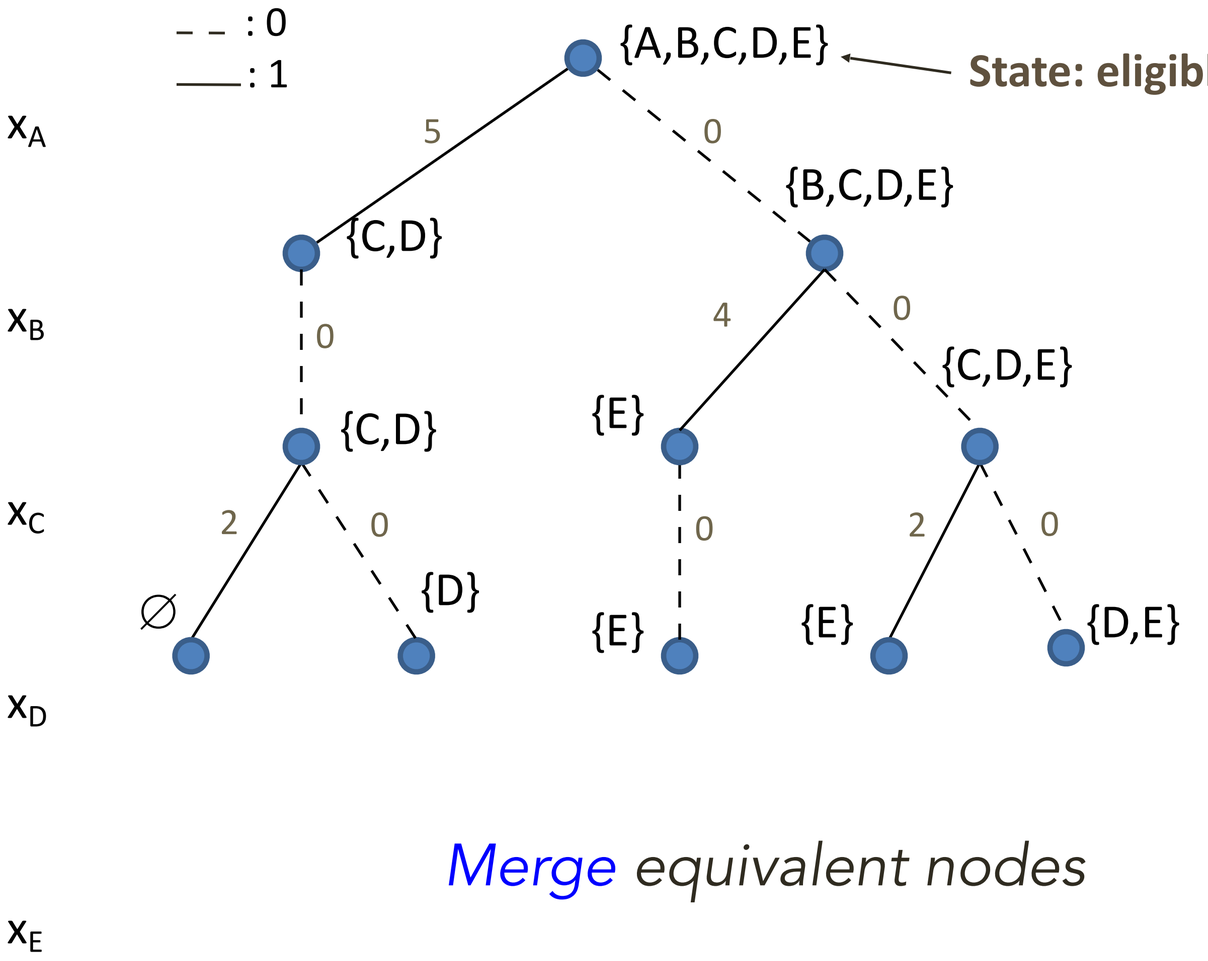
- Our Model: **Dynamic Program**
  - **State:** set of vertices that can be added to independent set
  - **Recursion:**

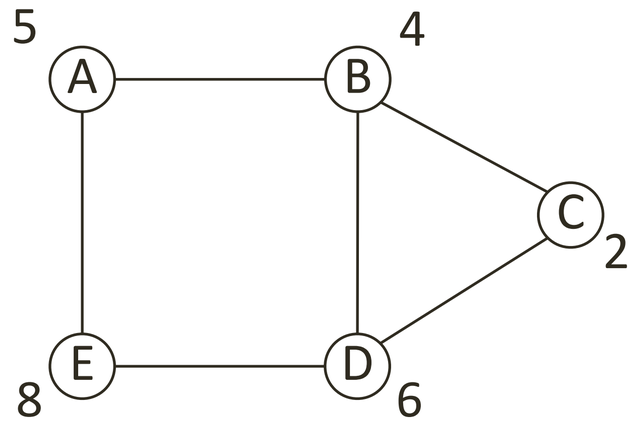
$$g(J) = \max_{j \in J} \{w_j + g(J \setminus N(j))\}$$

value function — state — immediate value (weight) — vertex j and neighbors

  - Boundary condition: $g(\varnothing) = 0$
  - Optimal value: $g(V)$

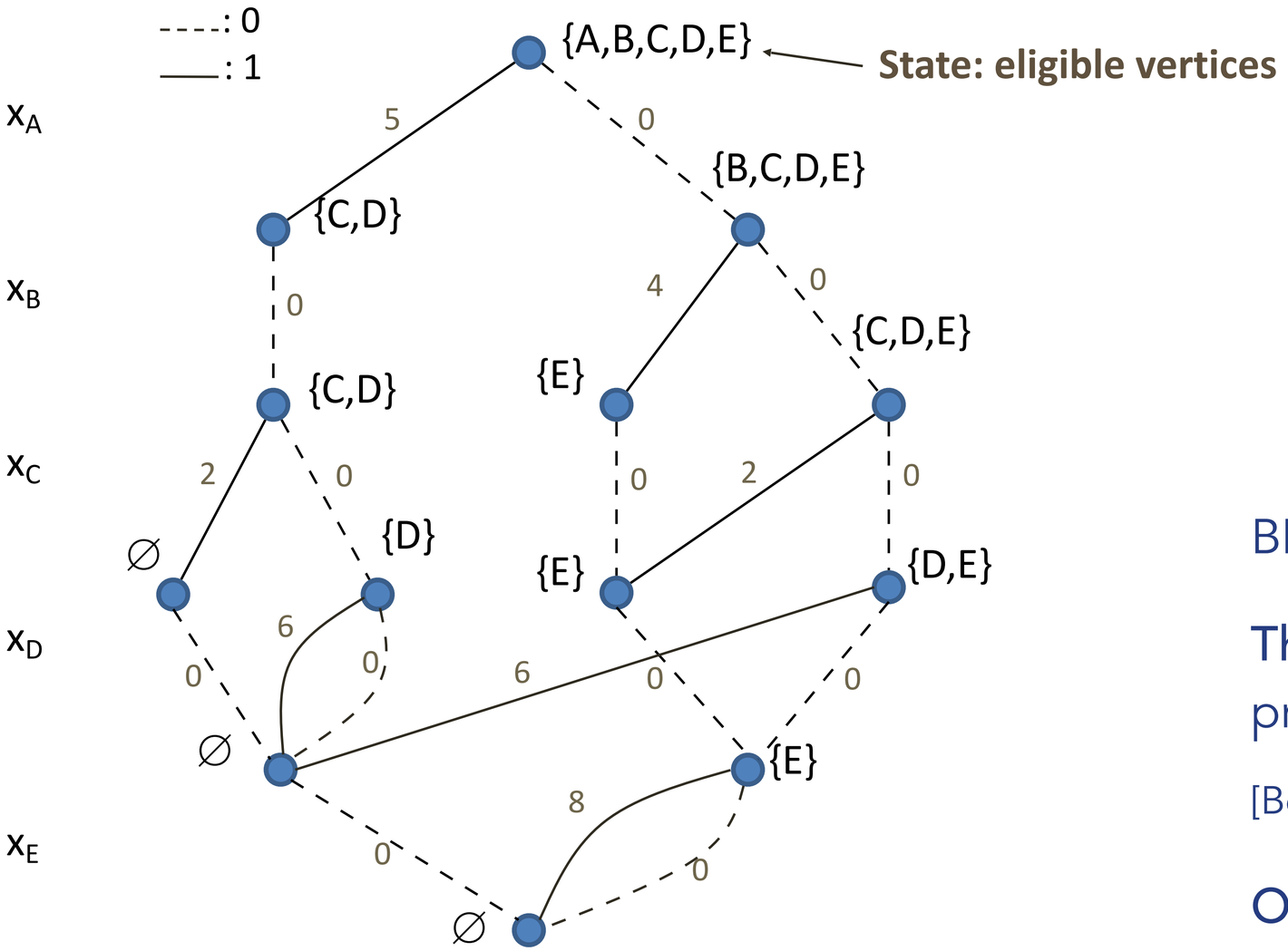
BDD Compilation for Maximum Independent Set

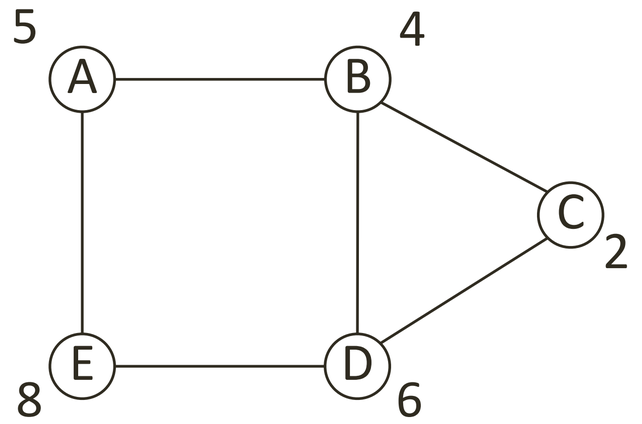




Merge equivalent nodes

BDD Compilation for Maximum Independent Set





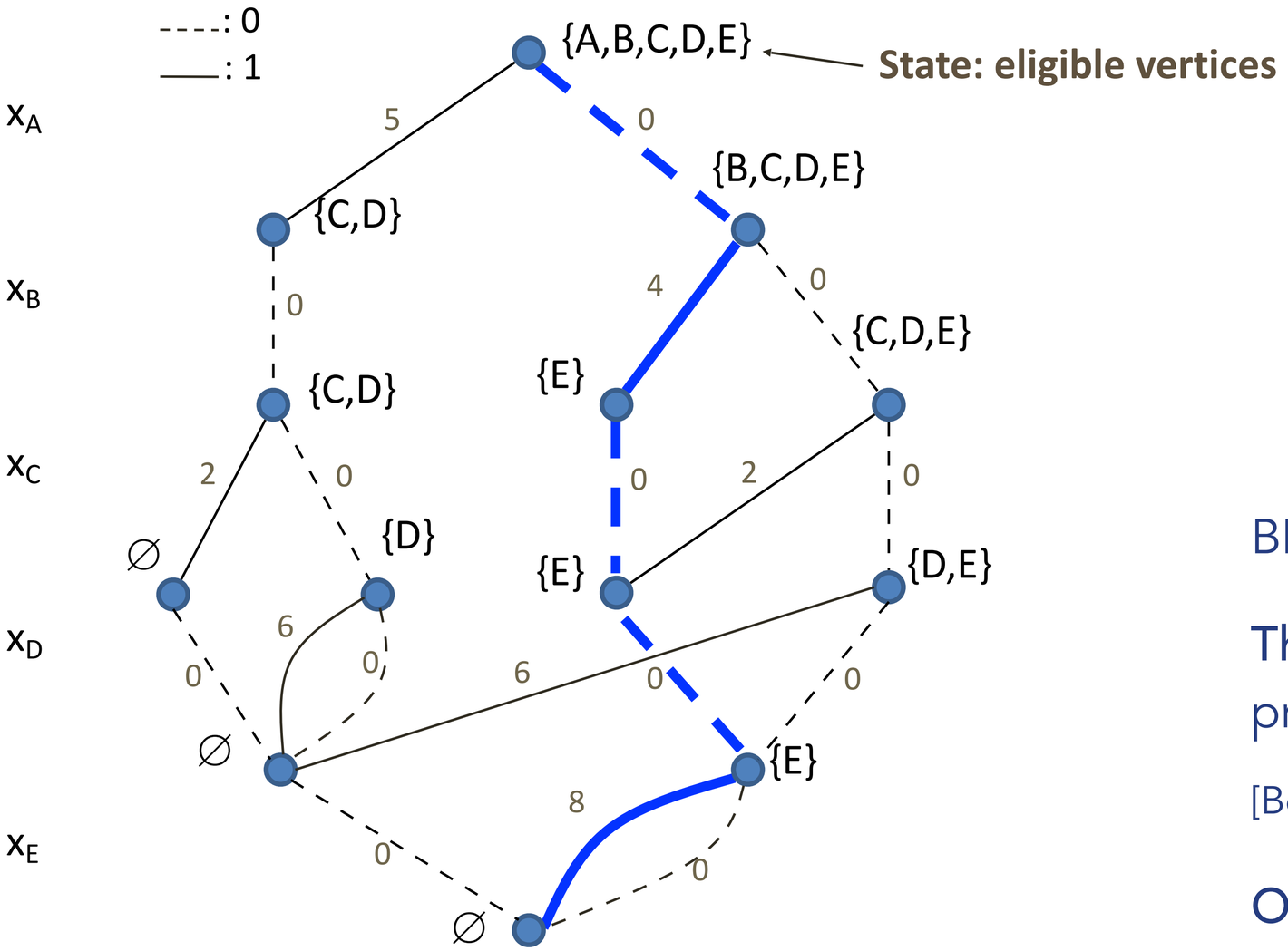
BDD is the DP state-transition graph

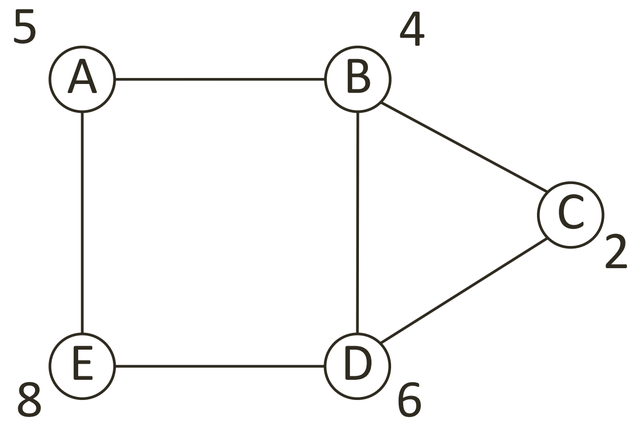
Theorem: This top-down compilation procedure generates the exact ROBDD

[Bergman, Cire, vH, Hooker, IJOC 2014]

Optimal solution: Longest path

BDD Compilation for Maximum Independent Set





BDD is the DP state-transition graph

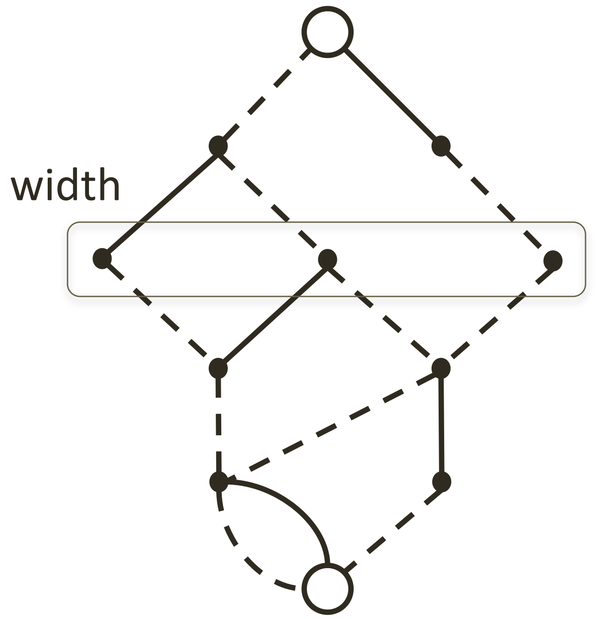
Theorem: This top-down compilation procedure generates the exact ROBDD

[Bergman, Cire, vH, Hooker, IJOC 2014]

Optimal solution: Longest path

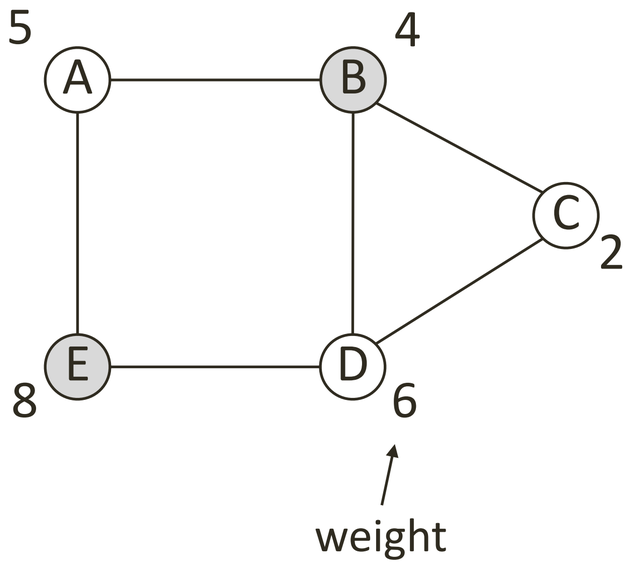
Relaxed Decision Diagrams: Polynomial Size

- Limit the size of the diagram to a maximum width
- Merge non-equivalent nodes
  - Define *node merging rule* to safely aggregate states
- Requirements for relaxation
  1. Must represent a superset of exact solutions
  2. The path costs are valid (w.r.t. exact solutions)
- Provides discrete relaxation
  - Strength is controlled by the maximum width



[Andersen, Hadzic, Hooker, Tiedemann, CP 2007]
[Bergman, Cire, vH, Hooker, CPAIOR 2011, IJOC 2016]

Update Model with State Merging Operator



Independent set in a graph:

- Subset of non-adjacent vertices

Maximum Independent Set Problem:

- Find independent set with maximum weight

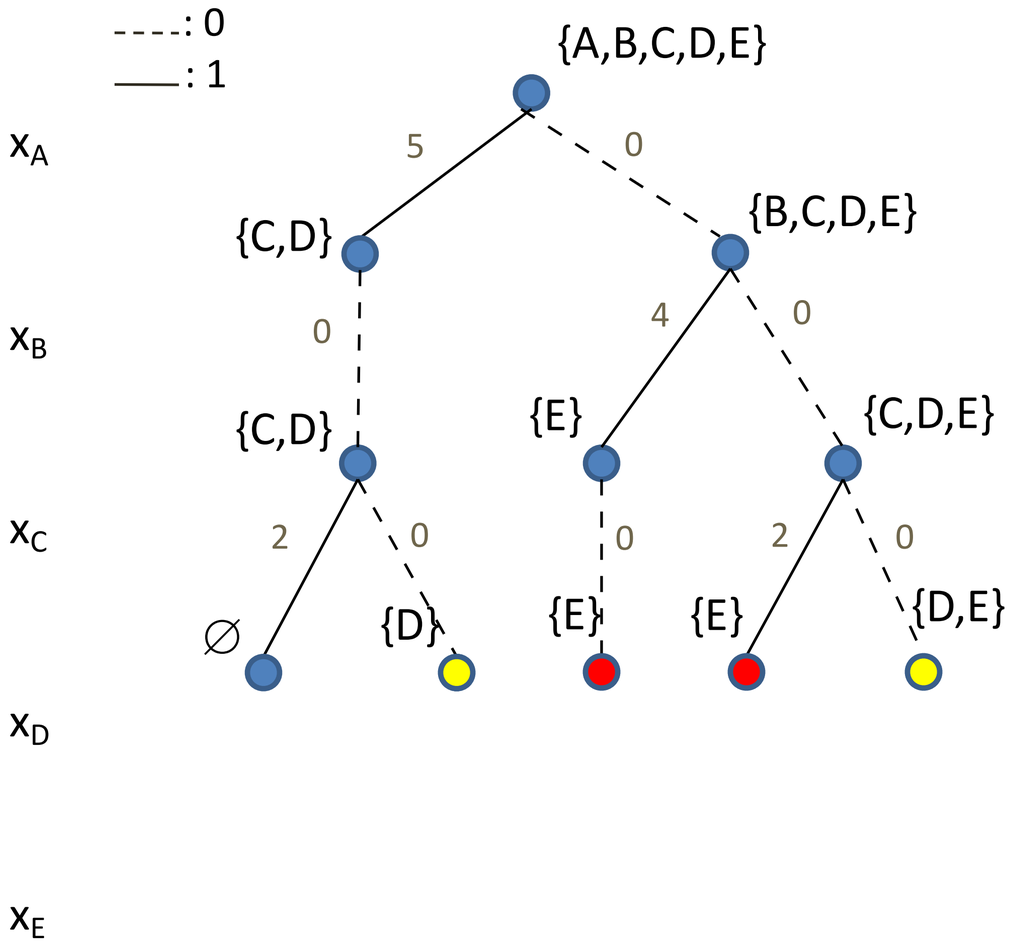
- Our Model: **Dynamic Program**
  - **State:** set of vertices that can be added to independent set
  - **Recursion:**

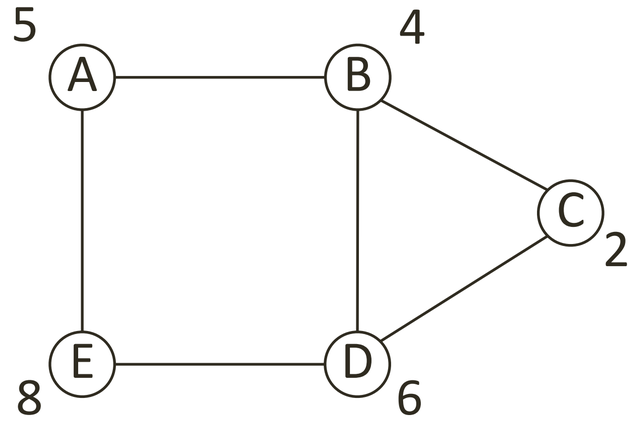
$$g(J) = \max_{j \in J} \{w_j + g\left(J \setminus N(j)\right)\}$$

  - **Boundary condition:** $g(\varnothing) = 0$
  - **Optimal value:** $g(V)$
  - **State merging:** $\oplus(M) = \bigcup_{J \in M} J$

merging operator — set of states

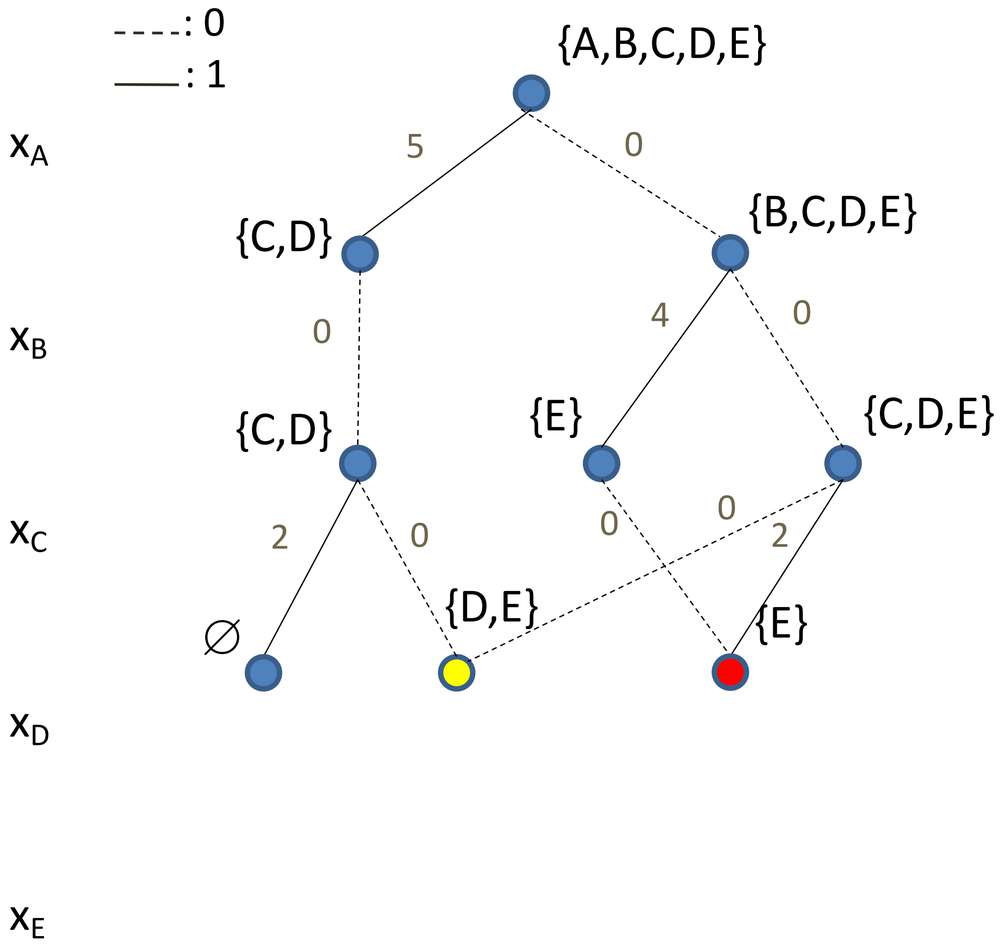
Independent Set Problem: Relaxed DD

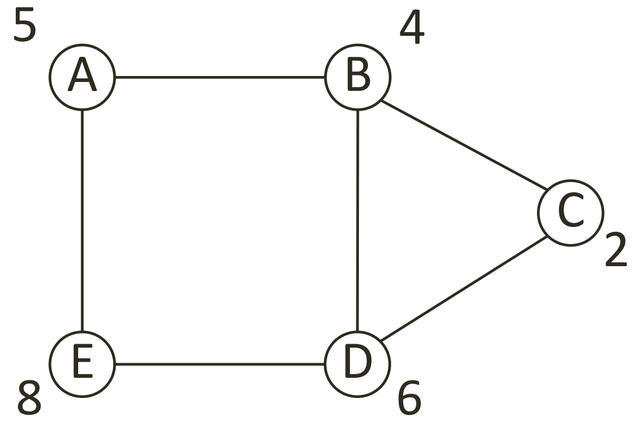




Maximum width = 3

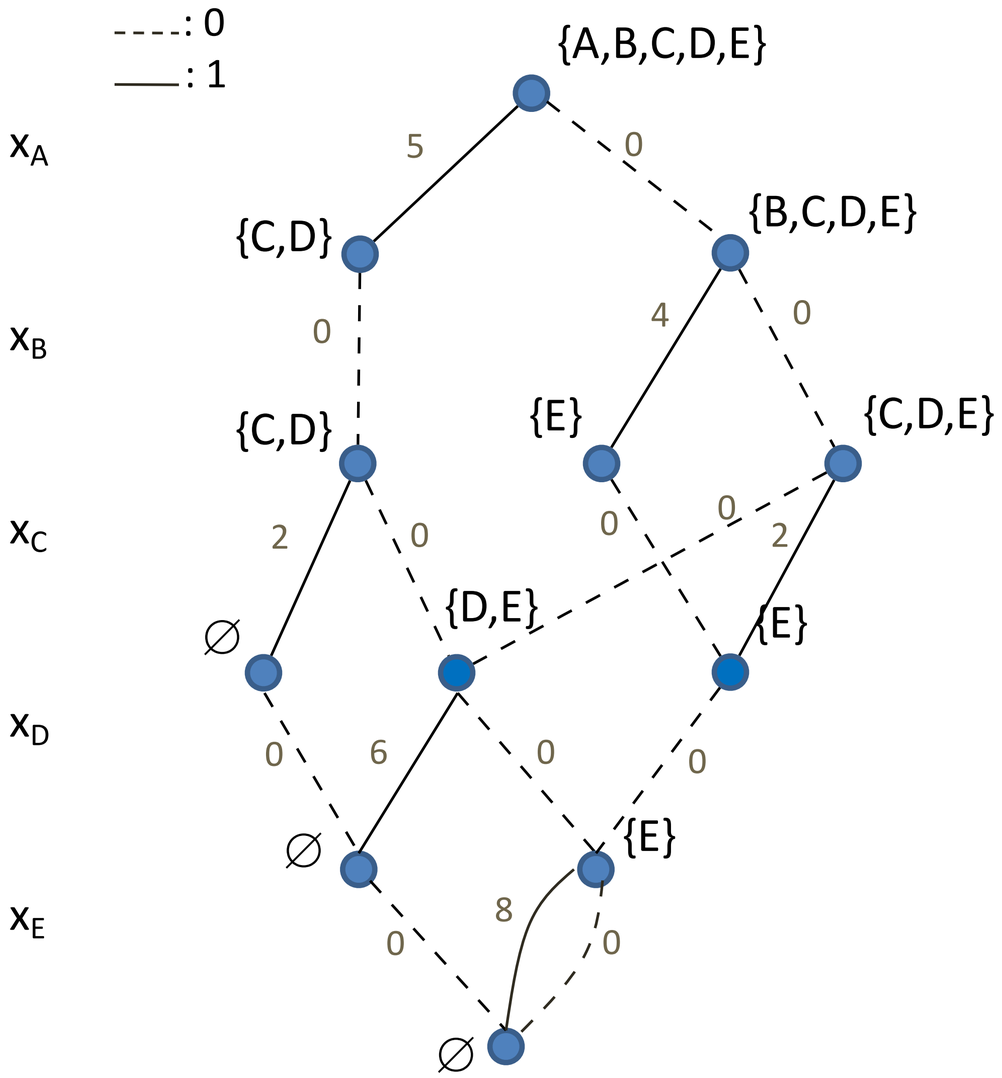
Independent Set Problem: Relaxed DD

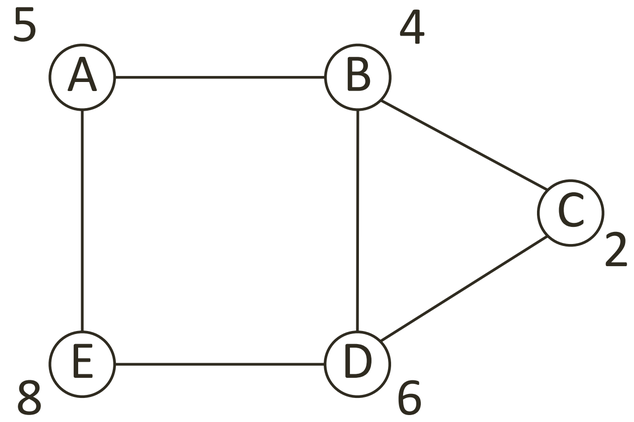




Maximum width = 3

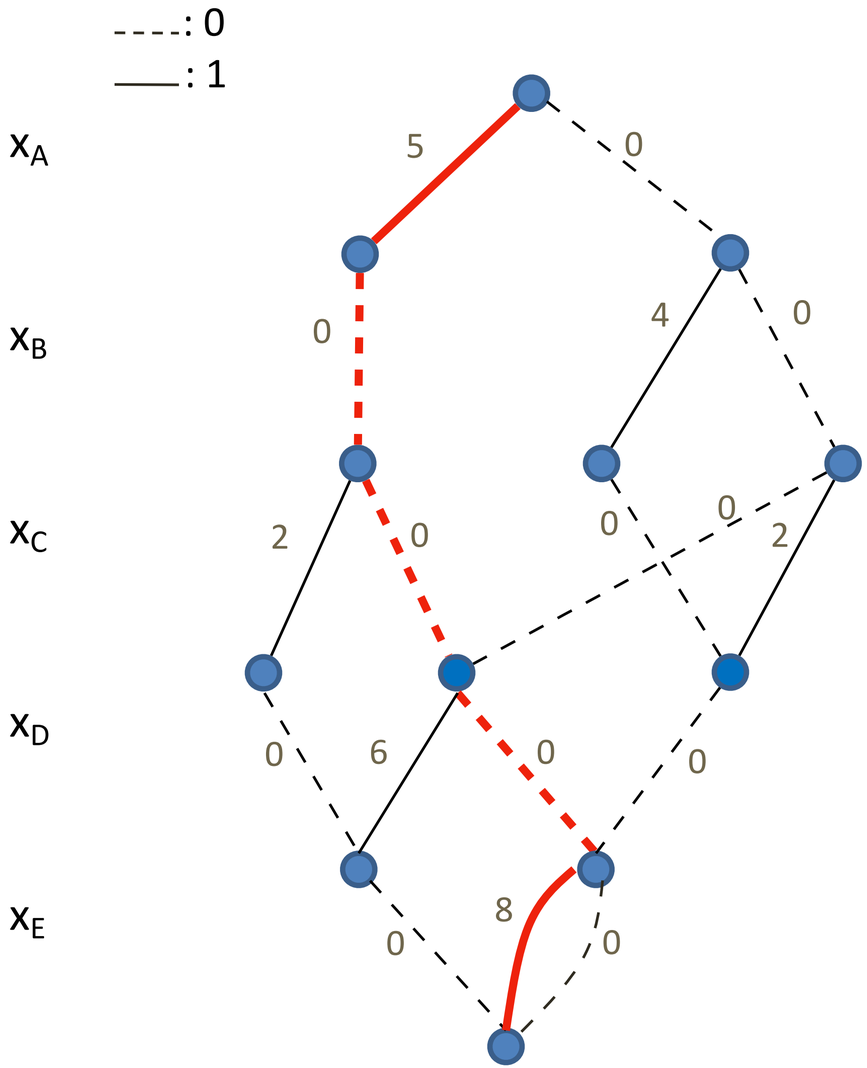
Independent Set Problem: Relaxed DD

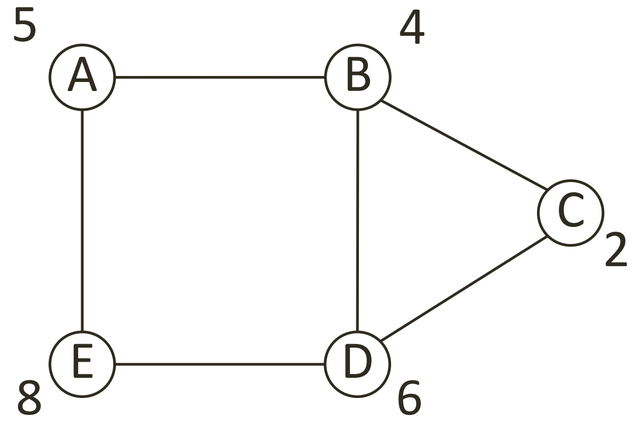




Maximum width = 3

Relaxed Decision Diagram Provides Dual Bound





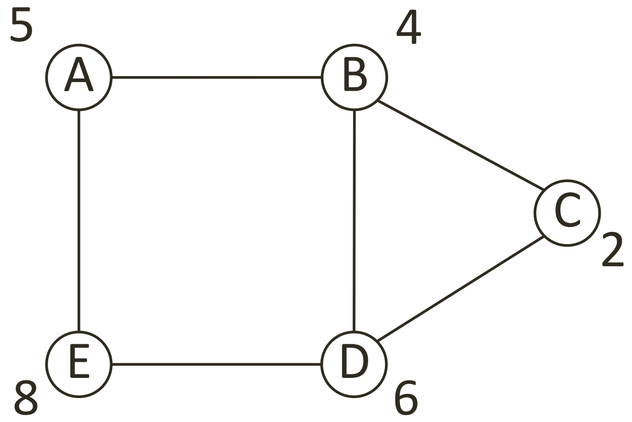
Maximum width = 3

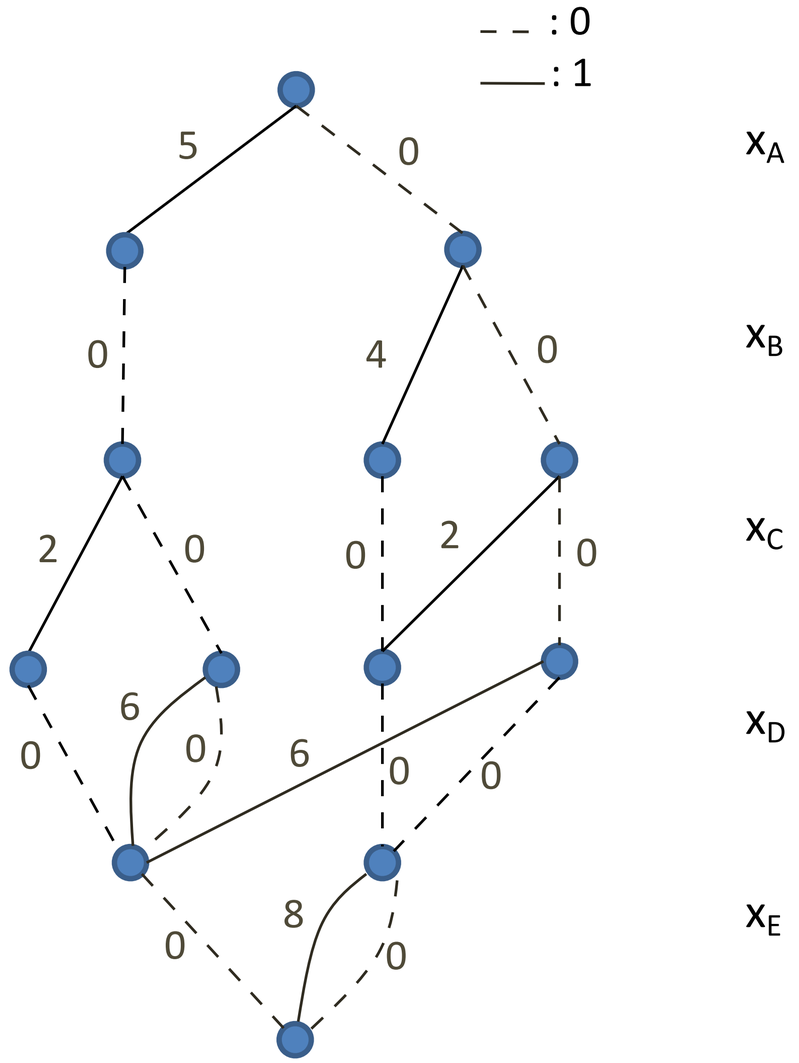
Longest path:

- x = (1, 0, 0, 0, 1)
- Dual (upper) bound = 13

Restricted Decision Diagrams: Primal Bound

- Under-approximation of the feasible set

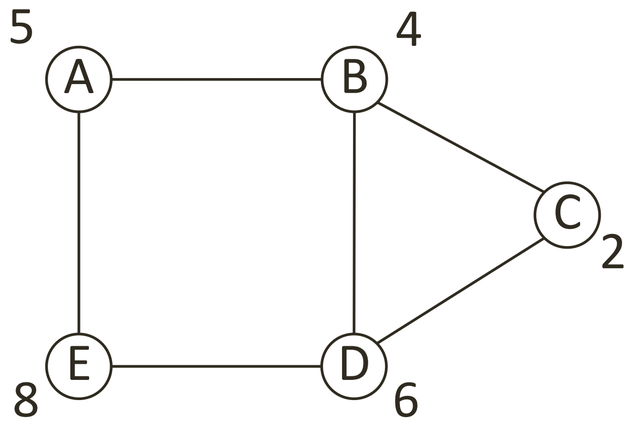




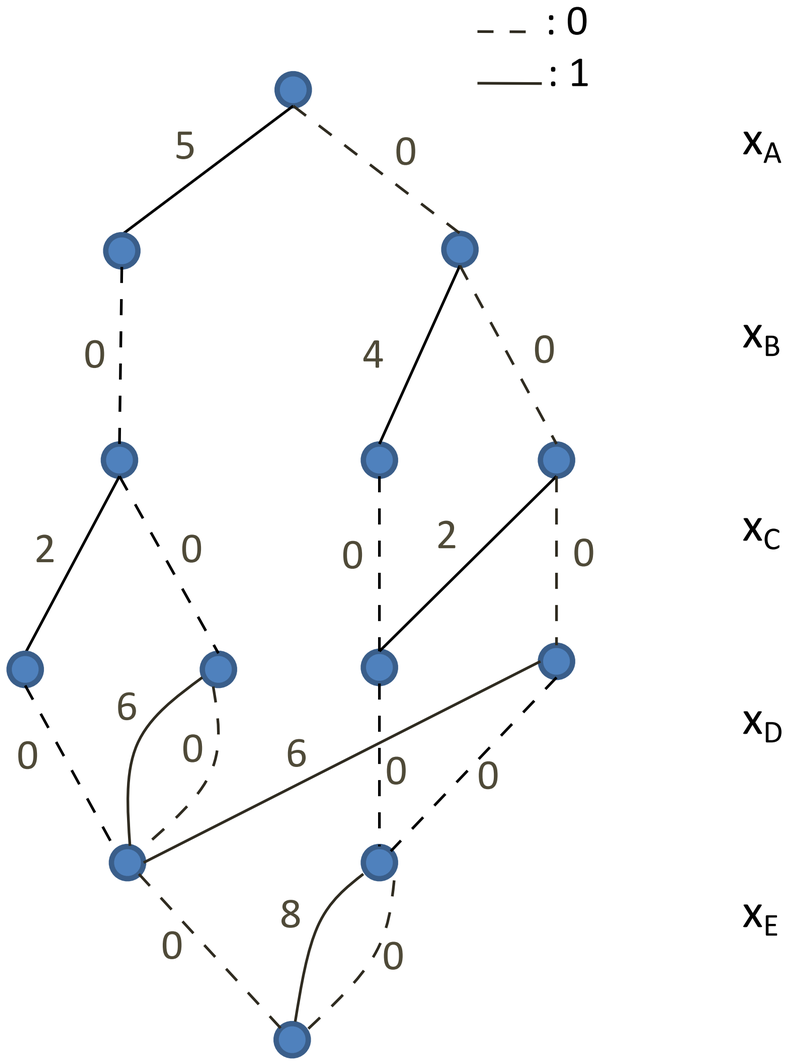
[Bergman, Cire, vH, Yunes, J Heur. 2014]

Restricted Decision Diagrams: Primal Bound

- Under-approximation of the feasible set



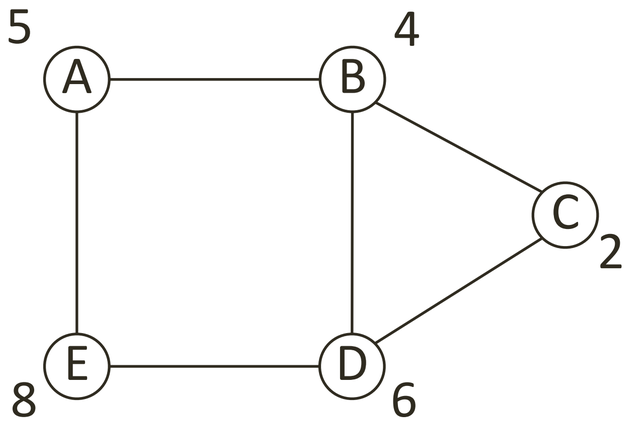
Maximum width = 3



[Bergman, Cire, vH, Yunes, J Heur. 2014]

Restricted Decision Diagrams: Primal Bound

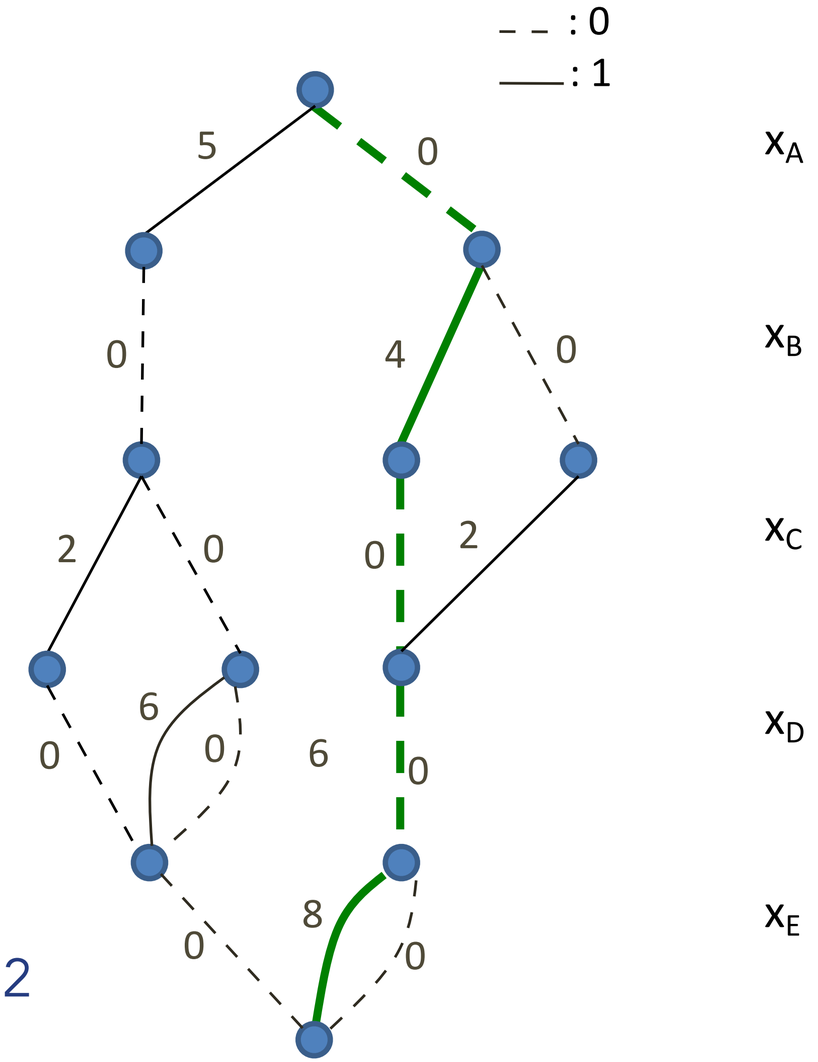
- Under-approximation of the feasible set



Maximum width = 3

Longest path:

- x = (0, 1, 0, 0, 1)
- Primal (lower) bound = 12



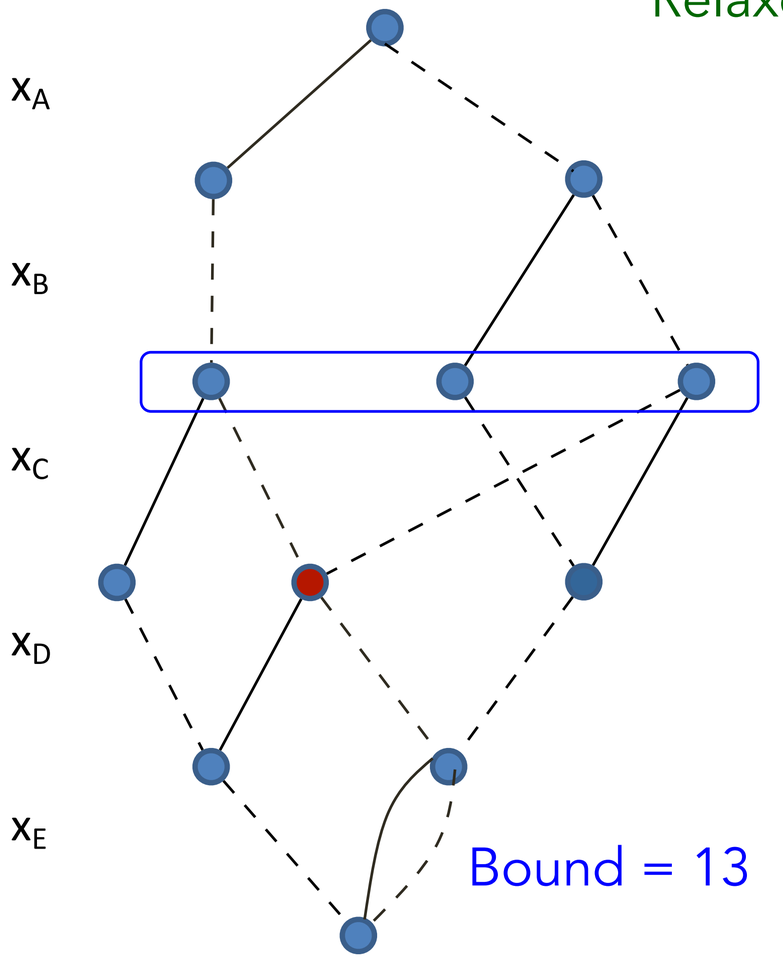
[Bergman, Cire, vH, Yunes, J Heur. 2014]

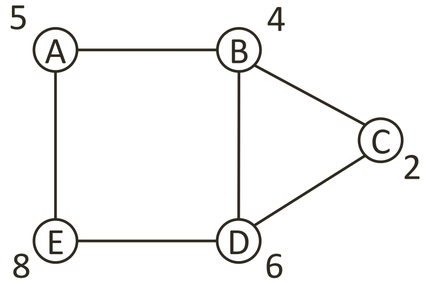
Exact Search Method

- Branch-and-bound scheme based on decision diagrams
  - Dual bounds: Relaxed decision diagrams
  - Primal bounds: Restricted decision diagrams
  - Branching is done on the *nodes* of the diagram

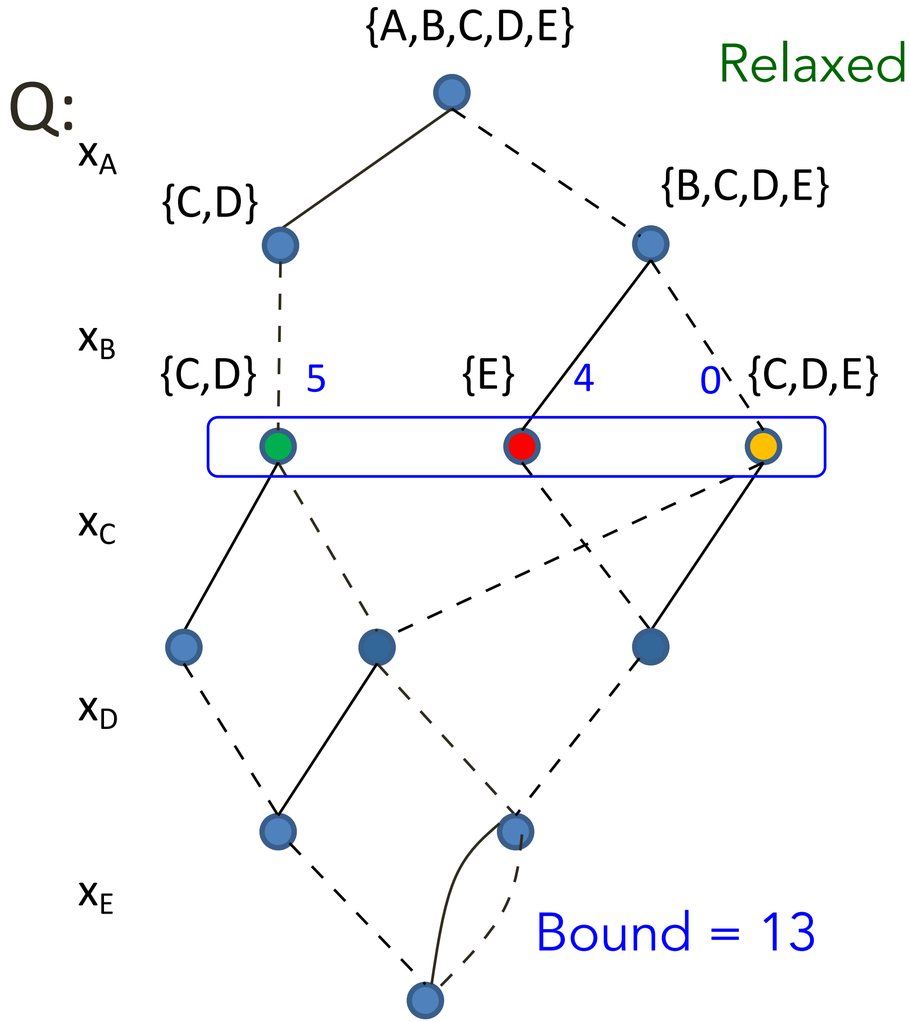
[Bergman,Cire,vH,Hooker IJOC 2016]

State-Based Branch-and-Bound





Node Queue

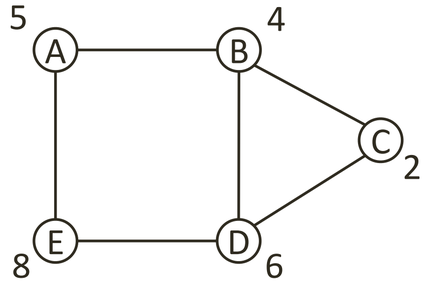


Relaxed BDD (width ≤ 3)

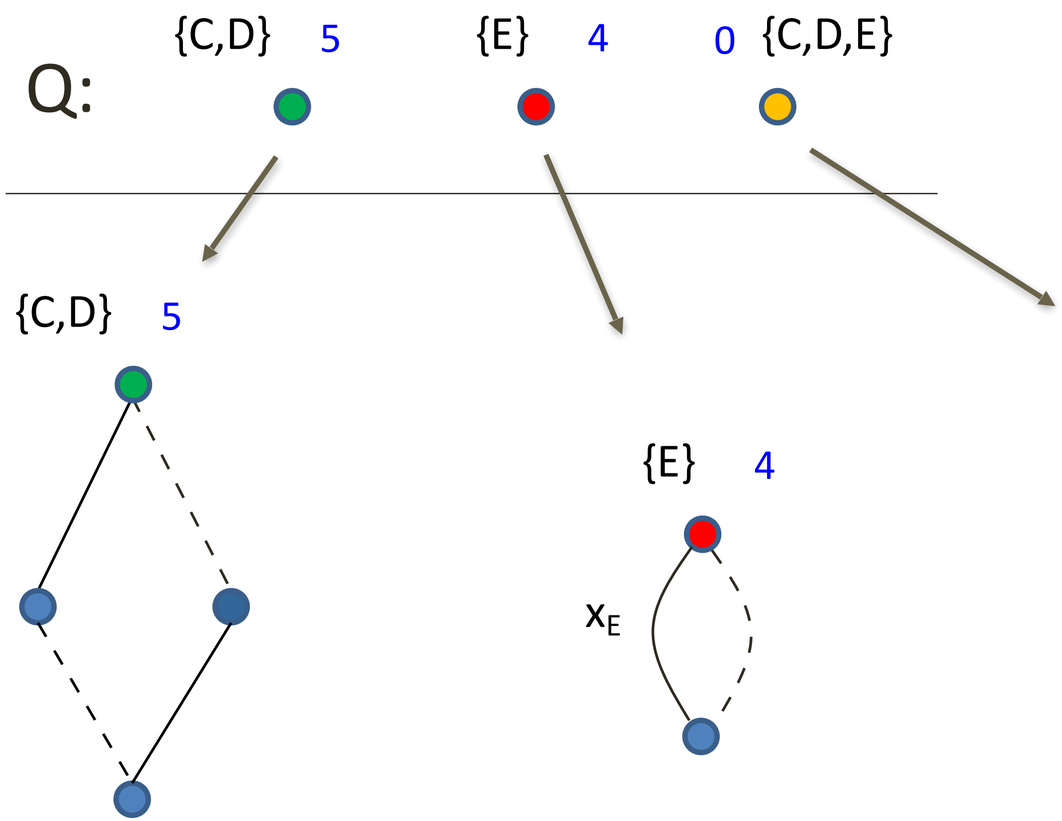
Upper bound = 13

Last Exact Layer

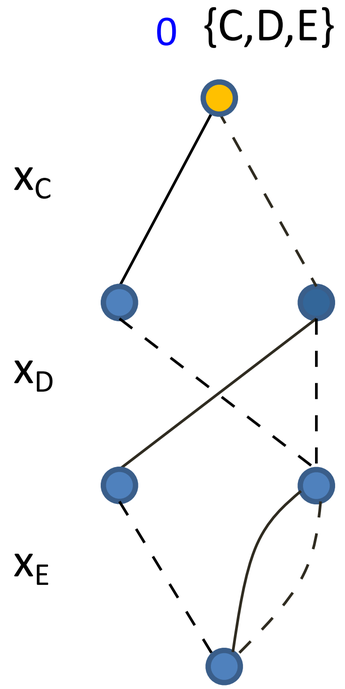
Bound = 13

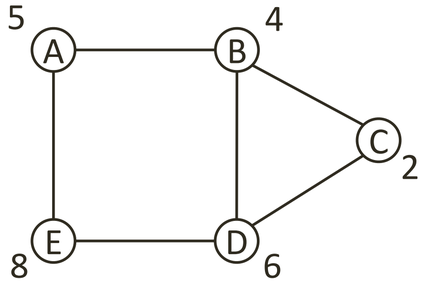


Node Queue



Upper bound = 13





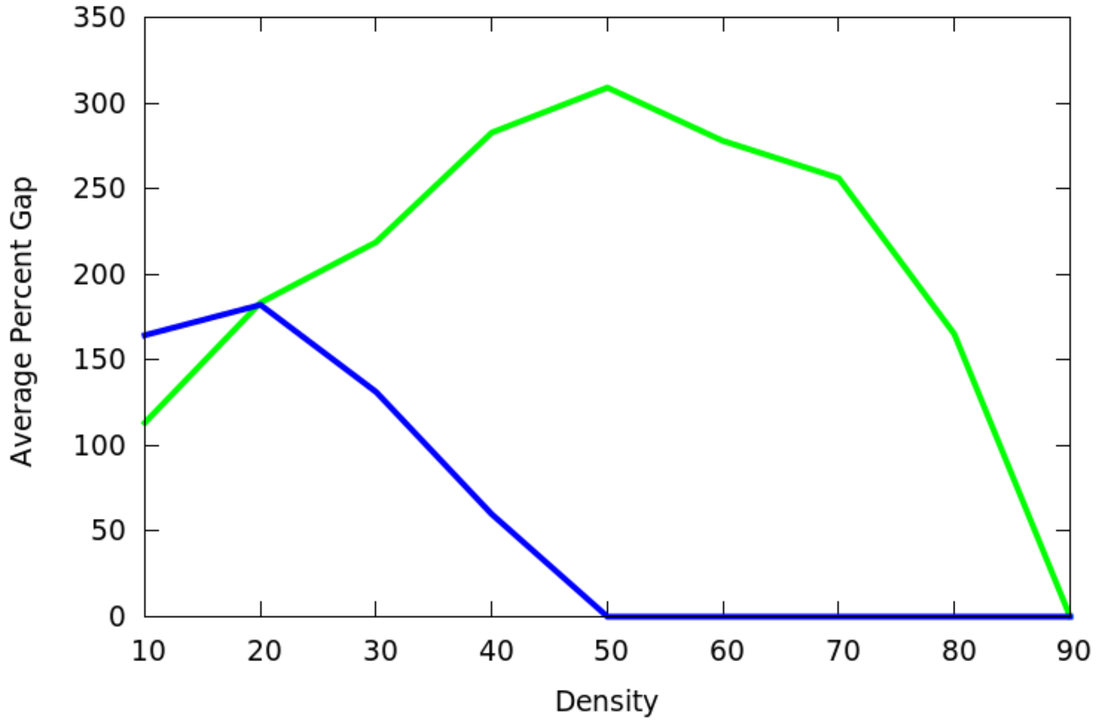
Exact solution: 11

Exact solution: 12

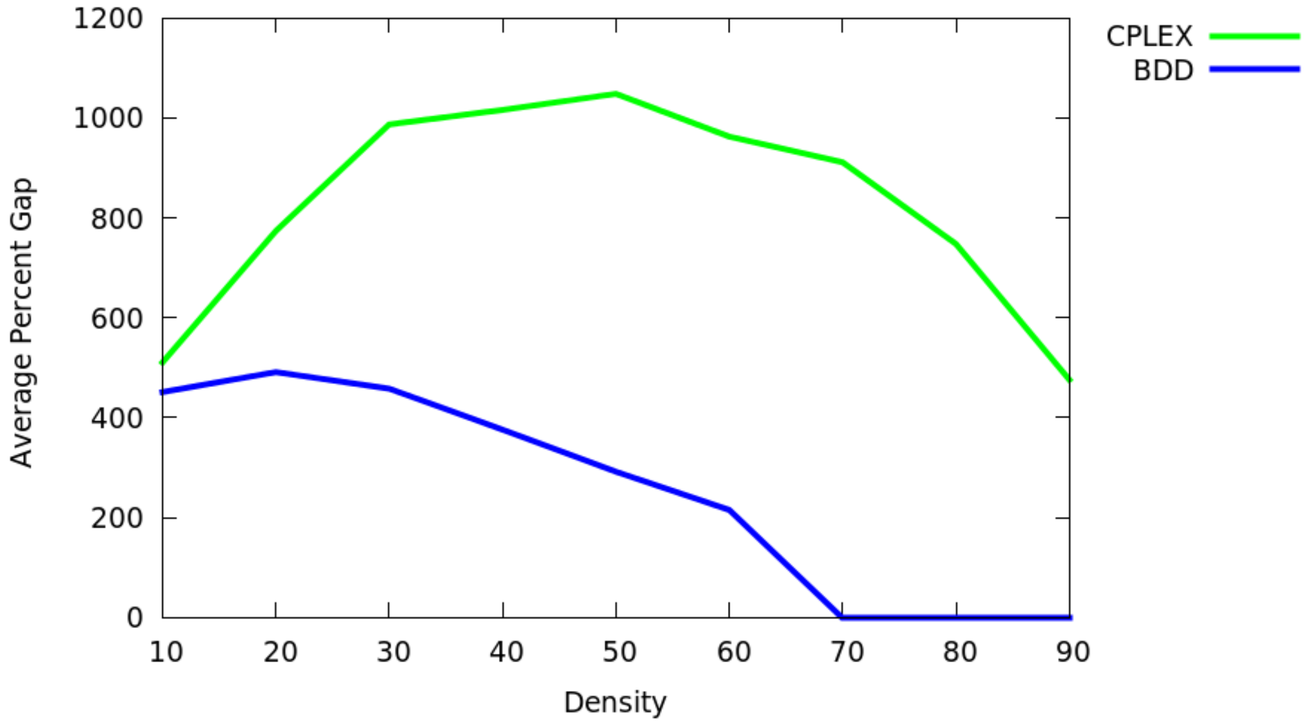
Exact solution: 10

Optimal solution: 12

Comparing Decision Diagrams with CPLEX



MISP with n=500 variables



MISP with n=1500 variables

- Randomly generated Maximum Independent Set Problem instances
- Average percent optimality gap with 1h time limit

Model+Solve for Dynamic Programming

Solver/System	Language	Reference
Ddo/DDOLib	Rust (+Python API), Java	[Gillard, Schaus, Coppé IJCAI2020+]
Peel-and-Bound	Julia	[Rudich, Cappart, Rousseau CP2022+]
CODD	C++	[Michel, vH ECAI2024+]
Nextroute	Go (+Python API)	[O'Neil/Nextmv] (industrial solver)
DIDP	Rust (+Python API)	[Kuroiwa, Beck, ICAPS2023+]

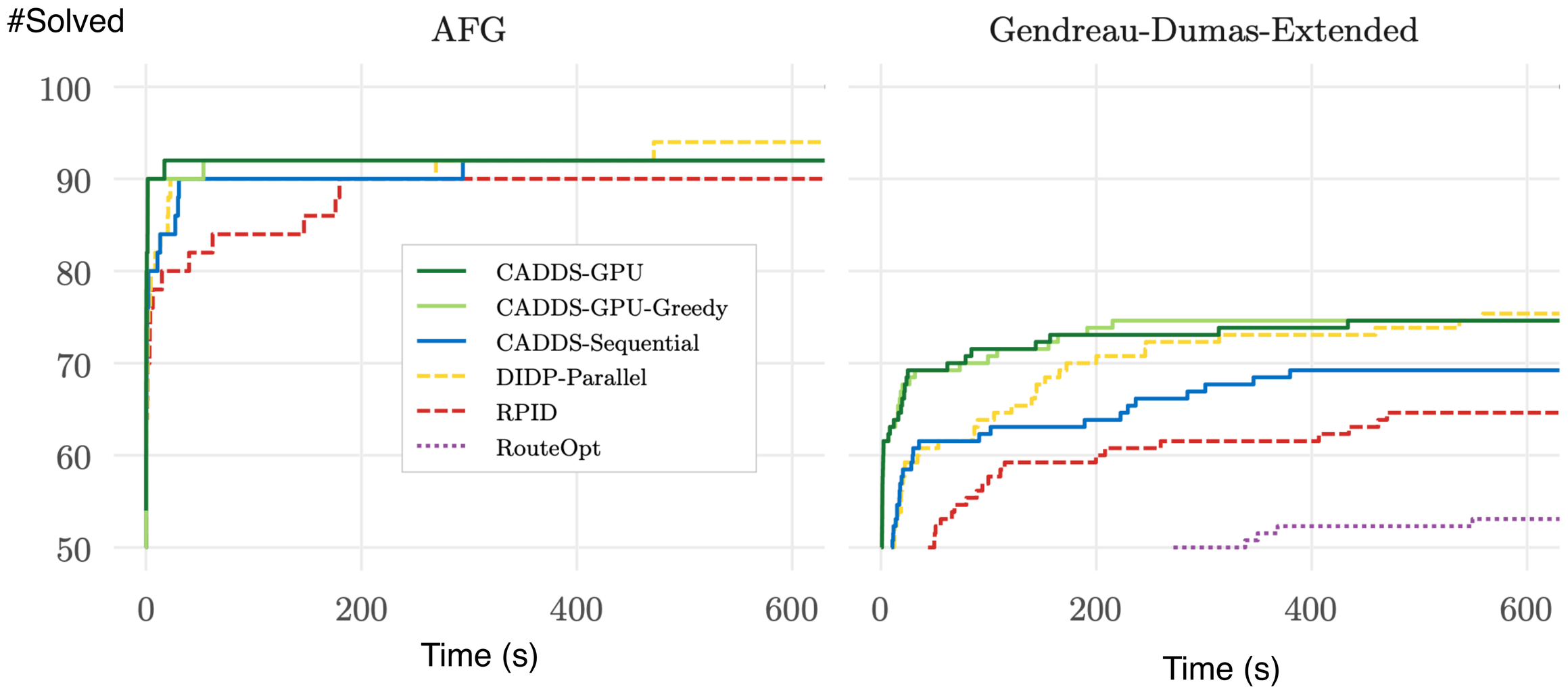
↘

- Inspired by heuristic state-based search from automated planning
- Fastest DIDP solver uses Complete Anytime Beam Search (CABS)

Additional Features

- Dominance rules [Coppé,Gillard,Schaus CPAIOR2024]
  - Prune provably dominated states (similar to dynamic programming)
- State-based (local) bounds [Gillard,Coppé,Schaus,Ciré CPAIOR2021]
  - Prune sub-optimal states based on (combinatorial) local bound
- Bounds based on state aggregation [Coppé,Gillard,Schaus CP2023]
  - Requires a mapping from original states to abstract states
- Search and compilation strategies
  - Peel-and-Bound: incremental compilation [Rudich,Cappart,Rousseau CP2022]
  - **Complete Anytime Decision Diagram Search** [Tardivo,Michel,vH CPAIOR2026]
  - GPUs for relaxed DDs in branch-and-bound [Tardivo,Michel,vH CP2026]

TSP with Time Windows: CADDS+GPU in CODD



[Tardivo,Michel,vH CPAIOR2026]

(Durer, 1514)

Constraint Programming

Constraint Programming = Propagate (+ Search)

- Constraint propagation algorithm for individual constraints
  - remove inconsistent values from variable domains
  - propagate updated domains to other constraints

$$alldifferent(x_1, x_2, x_3, x_4)$$

$$x_1 + x_2 + x_3 \geq 9$$

$$x_i \in \{1,2,3,4\} \quad (i = 1,2,3,4)$$

No propagation…

- Domains are weak communication mechanism

- **MDDs** can communicate more information
- Generic and explicit representation

Propagate Decision Diagrams!

$$alldifferent(x_1, x_2, x_3, x_4)$$

$$x_1 + x_2 + x_3 \geq 9$$

$$x_i \in \{1,2,3,4\} \quad (i = 1,2,3,4)$$

Propagate decision diagrams

- Remove inconsistent arcs from diagram
- Refine the diagram
- Use relaxed diagrams of polynomial size

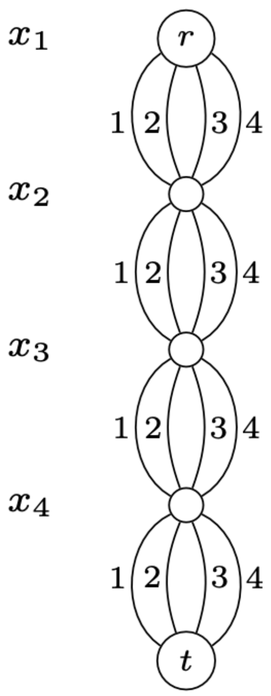
[Andersen, Hadzic, Hooker, Tiedemann, CP2007]

Improved Optimization

- Evaluate objective to retrieve dual bound

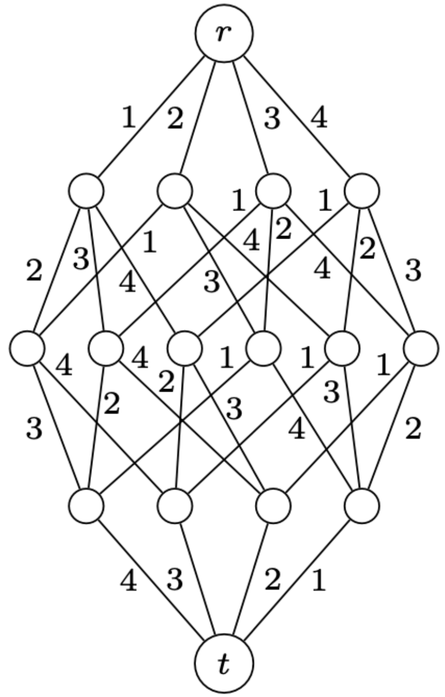
[Hoda,vH,Hooker CP2010] [Cire,vH OR2013] [Gentzel,Michel,vH,2023]

domain relaxation



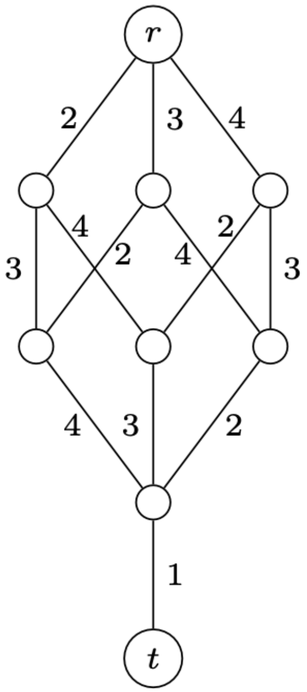
256 solutions

propagate alldifferent



24 solutions

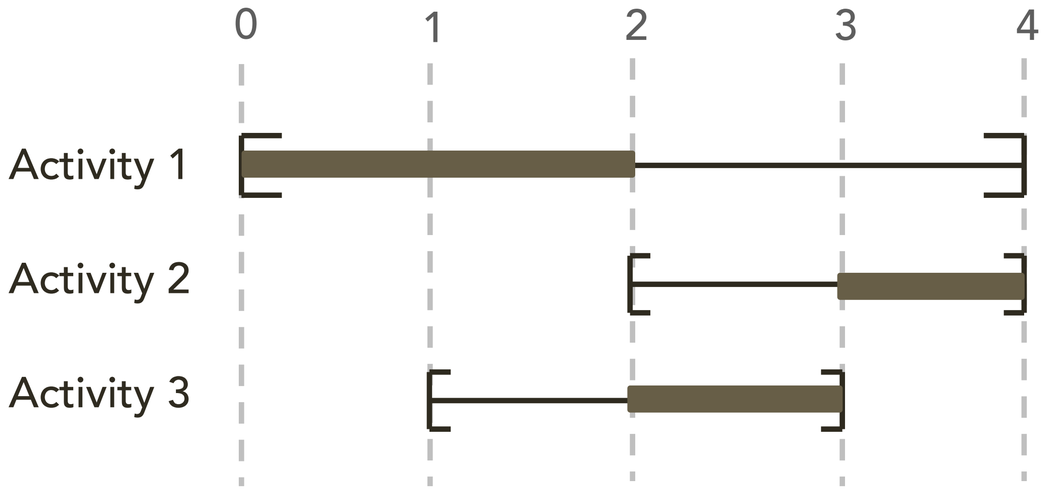
propagate $x_1 + x_2 + x_3 \geq 9$



6 solutions

Example Application: Disjunctive Scheduling

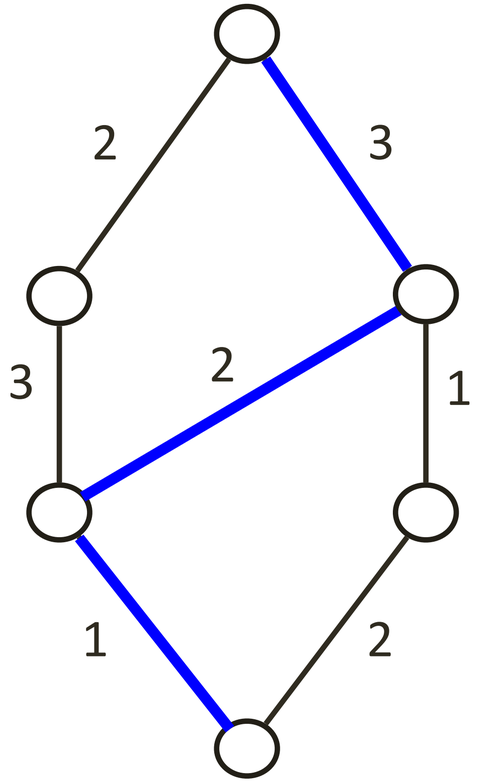
- Activities
  - Processing time: p_i
  - Release time: r_i
  - Deadline: d_i
- Resource
  - Nonpreemptive
  - Process one activity at a time
- Objective
  - Minimize makespan, sum of completion times, tardiness, …
- Optional side constraints
  - Precedence constraints, sequence-dependent setup times, …



Decision Diagram: Solution = Permutation

Act	r_i	p_i	d_i
1	3	4	12
2	0	3	11
3	1	2	10

precedence: 3 ≪ 1



π_1

π_2

π_3

Path 3 – 2 – 1 :

$6 \leq \text{start}_1 \leq 8$

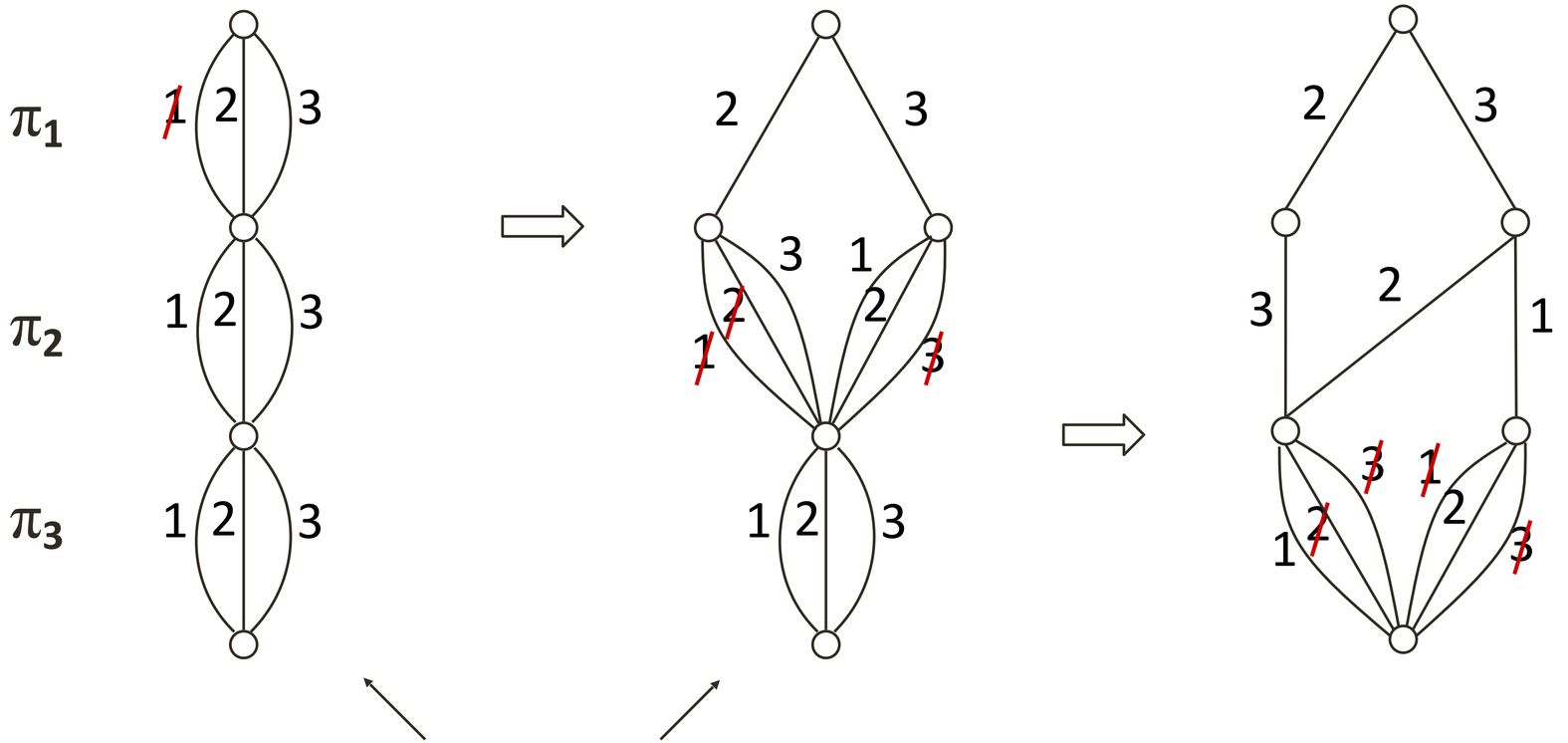
$3 \leq \text{start}_2 \leq 5$

$1 \leq \text{start}_3 \leq 3$

Solution: sequence of activities $\pi_1, \pi_2, \ldots, \pi_n$

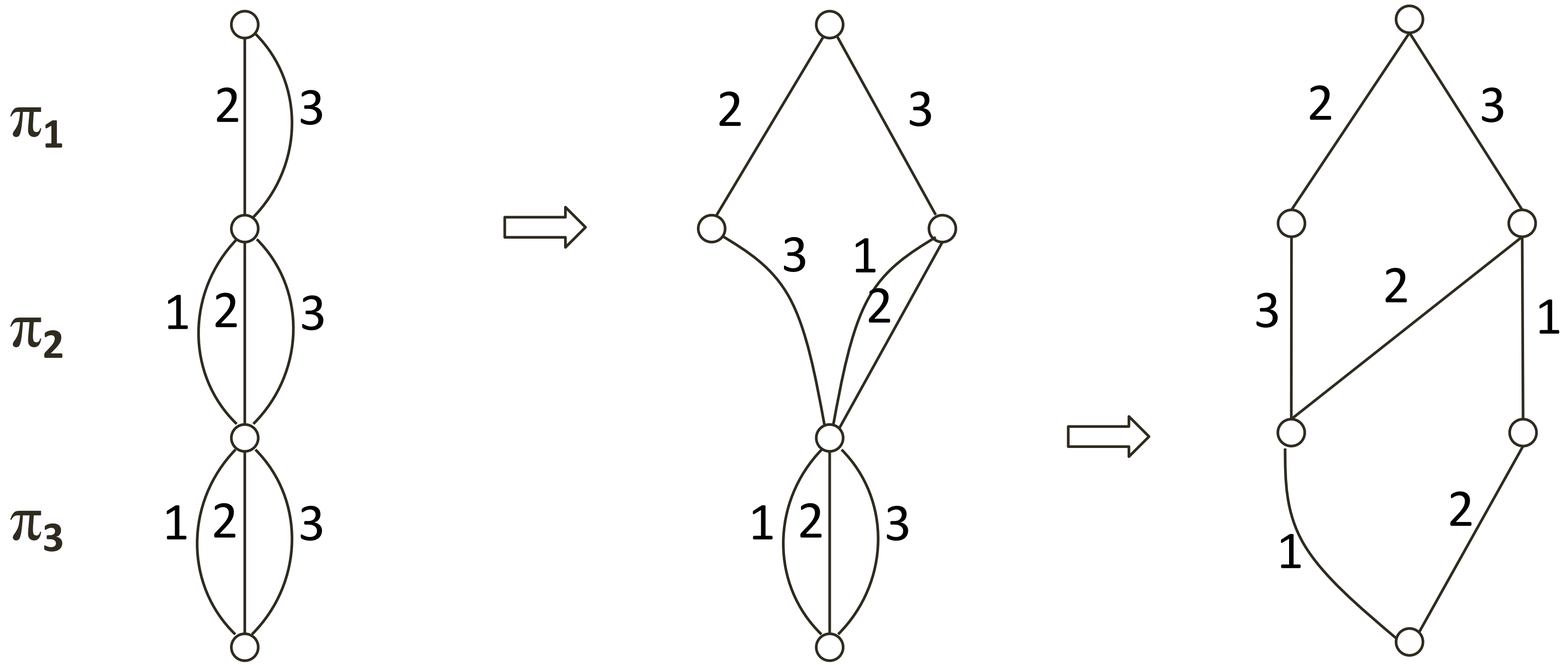
Top-down MDD compilation: Example

precedence:
$3 \ll 1$



Top-down MDD compilation: Example

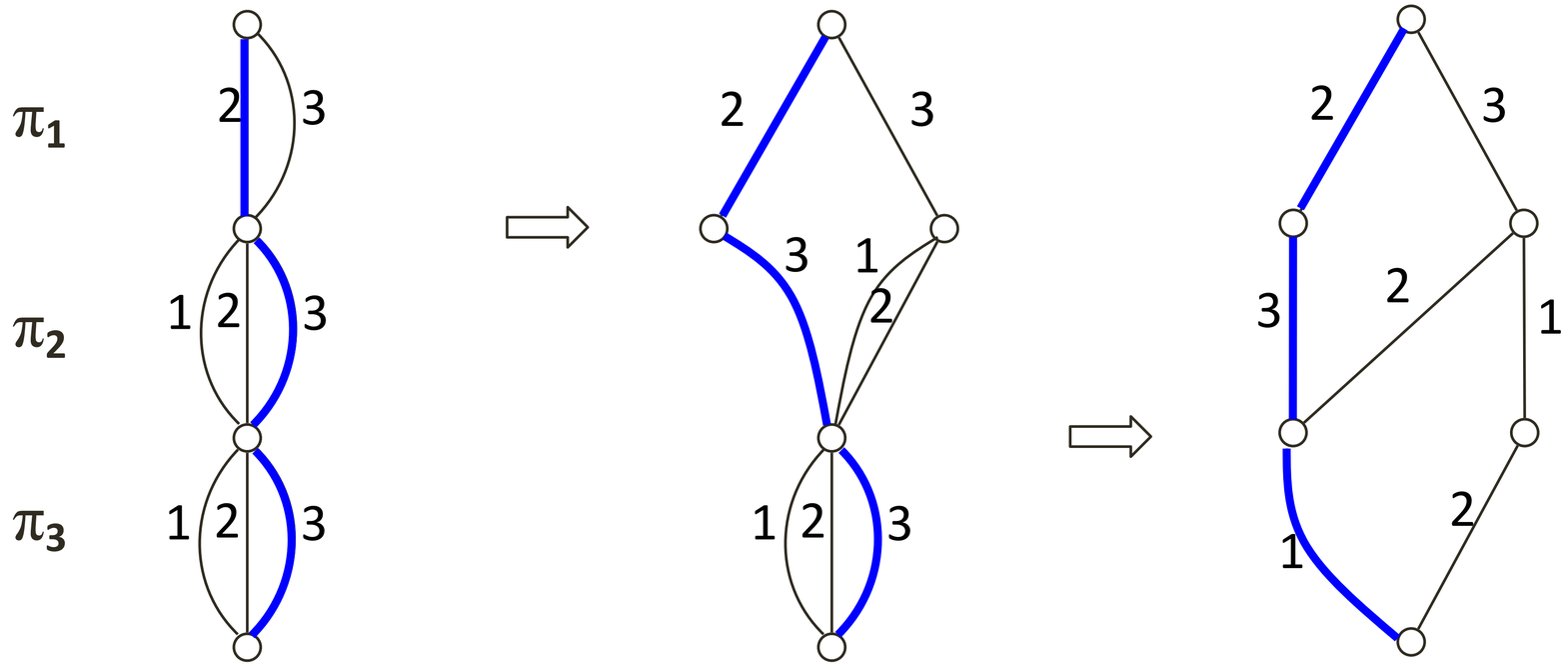
precedence:
$3 \ll 1$



Top-down MDD compilation: Example

precedence:
$3 \ll 1$

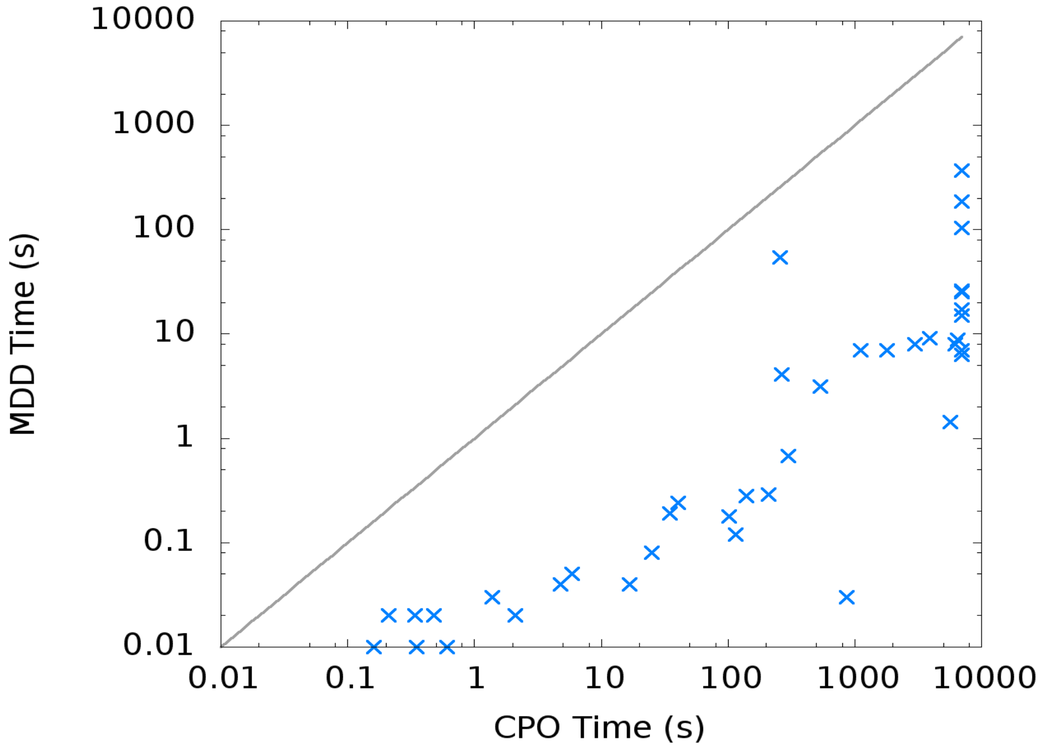
Act	r_i	p_i	d_i
1	3	4	12
2	0	3	11
3	1	2	10



minimize makespan: lower bound = 7 lower bound = 7 optimum = 9

Theorem: All implied precedences can be inferred from a relaxed or exact MDD in polynomial time, in the size of the MDD. [Cire, vH, OR2013]

Sequencing and Scheduling Applications



TSP with Time Windows

- 20-60 cities (Dumas/Ascheuer instances)
- max MDD width: 16

Compare MDD with CP Optimizer

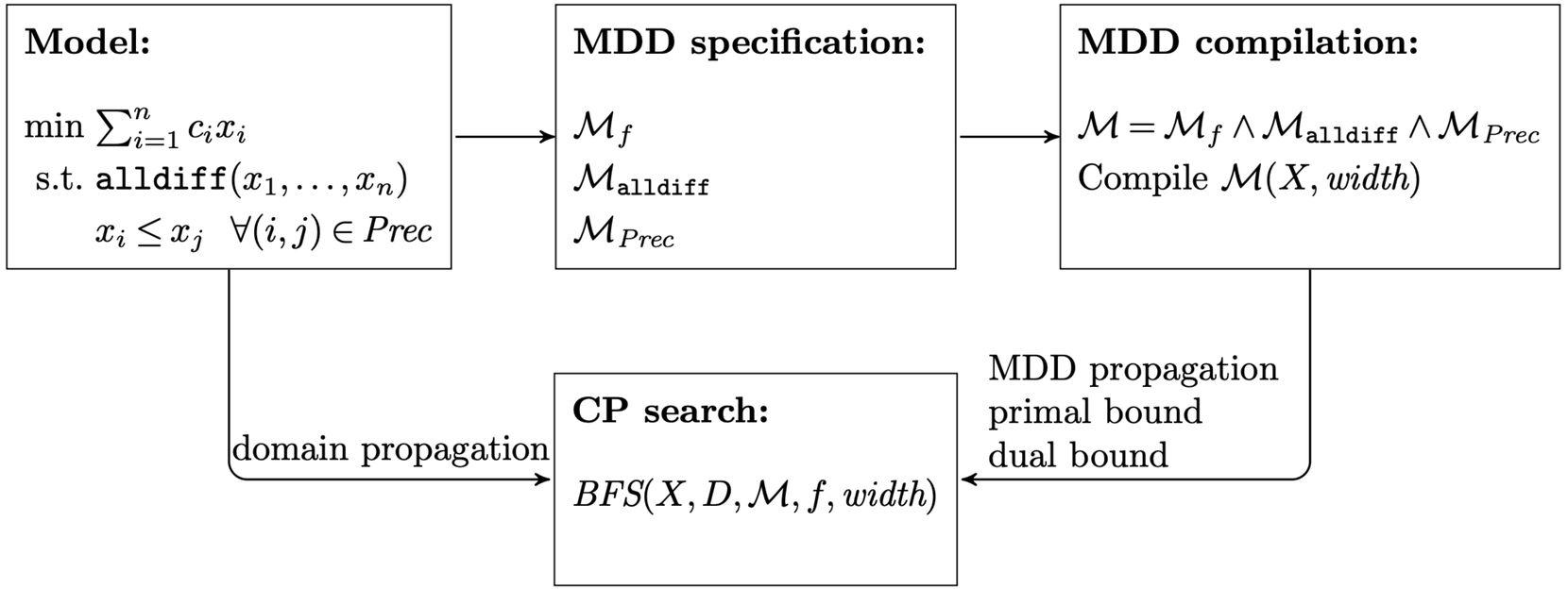
instance	vertices	bounds	CPO best	CPO time (s)	CPO+MDD, width 2048 best	CPO+MDD, width 2048 time (s)
br17.10	17	55	55	0.01	55	4.98
br17.12	17	55	55	0.01	55	4.56
ESC07	7	2125	2125	0.01	2125	0.07
ESC25	25	1681	1681	TL	1681	48.42
p43.1	43	28140	28205	TL	28140	287.57
p43.2	43	[28175, 28480]	28545	TL	**28480**	**279.18**
p43.3	43	[28366, 28835]	28930	TL	**28835**	**177.29**
p43.4	43	83005	83615	TL	83005	88.45
ry48p.1	48	[15220, 15805]	18209	TL	16561	TL
ry48p.2	48	[15524, 16666]	18649	TL	17680	TL
ry48p.3	48	[18156, 19894]	23268	TL	22311	TL
ry48p.4	48	[29967, 31446]	34502	TL	**31446**	**96.91**
ft53.1	53	[7438, 7531]	9716	TL	9216	TL
ft53.2	53	[7630, 8026]	11669	TL	11484	TL
ft53.3	53	[9473, 10262]	12343	TL	11937	TL
ft53.4	53	14425	16018	TL	14425	120.79

Sequential Ordering Problem (TSPLib)

- TSP + precedence constraints
- max MDD width: 2048

Closed 3 instances for the first time [Cire&vH 2013]

Haddock: Generic MDD-Constraint Propagation

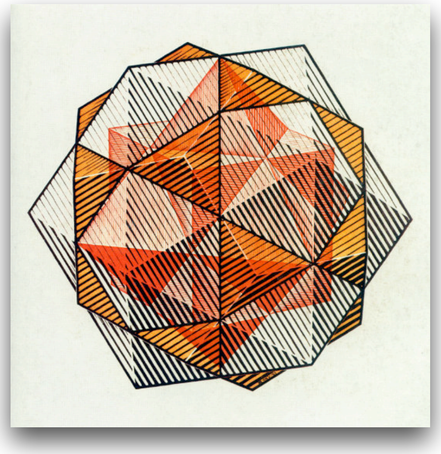



```
int main() {
  int N = ...;
  vector<int> cost{...};
  auto Prec = {...};
  int maxWidth = ...;

  auto cp = makeSolver();
  auto mdd = new MDDRelax(cp, maxWidth);

  auto vars = intVarArray(cp, N, 0, N - 1);
  auto z = makeIntVar(cp, 0, 1000);

  auto obj = minimize(z);

  sumMDD(mdd, vars, cost, z);
  allDiffMDD(mdd, vars);
  for (auto& [i, j] : Prec) {
    precedenceMDD(mdd, vars, i, j);
  }

  cp->post(mdd);
  ...
}
```

- Declarative specification language for MDD propagation
  - Define state, transition function, relaxation function
- Automatic compilation of MDD propagators in MiniCP

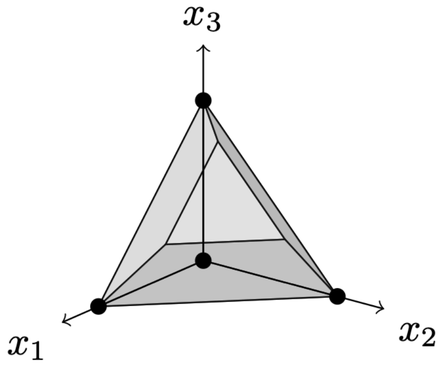
[Gentzel, Michel, vH, CP2020, CP2022, CPAIOR2023]

(Escher, 1961)



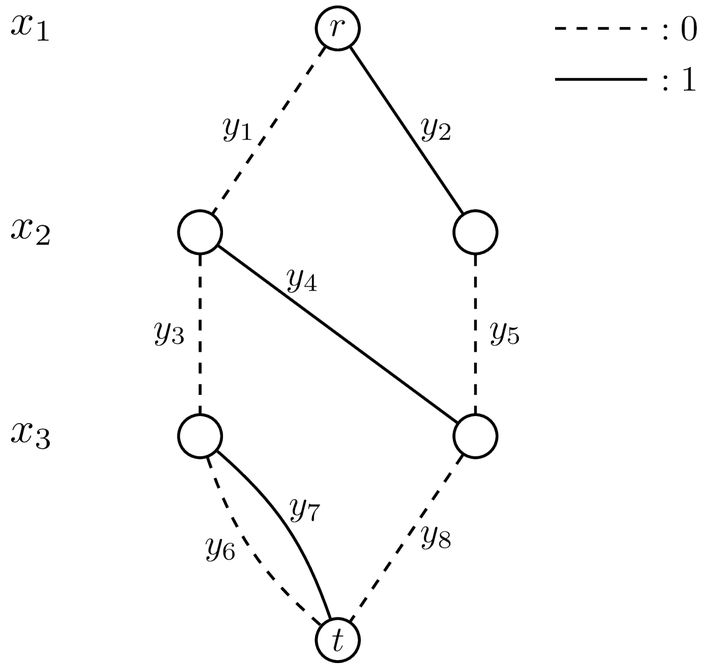
Integer Programming

Representing Integer Feasible Sets



$$\{x \in \{0,1\}^3 : \quad x_1 + x_2 - x_3 \leq 1,$$
$$x_1 - x_2 + x_3 \leq 1,$$
$$-x_1 + x_2 + x_3 \leq 1,$$
$$x_1 + x_2 + x_3 \leq 2\}$$

Integer Program Representation



Feasible Set as Decision Diagram

$$y_1 + y_2 = 1$$

$$y_1 = y_3 + y_4$$
$$y_2 = y_5$$
$$y_3 = y_6 + y_7$$
$$y_4 + y_5 = y_8$$

$$y_6 + y_7 + y_8 = 1$$

$$0 \leq y_i \leq 1, \ i \in \{1, \ldots, 8\}$$

$$x_1 = 0y_1 + 1y_2$$
$$x_2 = 0y_3 + 1y_4 + 0y_5$$
$$x_3 = 0y_6 + 1y_7 + 0y_8$$

Network Flow + Mapping

Theorem [Behle07]: Projecting the decision diagram network flow reformulation onto the original space yields the convex hull.

Applications of Decision Diagrams in MIP

1. Reformulate (non-linear) components as decision diagram, and add the network flow model to the integer program

   E.g., Linearize nonlinear components [BergmanCire18] [BergmanLozano21] [Cardonha+25], Multi-objective [BergmanBodurCardonhaCire22], Bin Packing [MehraniCardonhaBergman22]

2. Generate cutting planes based on (relaxed) decision diagrams

   [Behle+05] [Behle07] [TjandraatmadjaVH19] [DavarniaVH21] [Castro+22]

3. DD-based dual bounds in MIP branch-and-bound search tree

   [TjandraatmadjaVH21] also uses MDD propagation and MDD-Lagrangian bounds

4. **Column Elimination:** Solve network flow over the relaxed decision diagram and iteratively remove infeasible paths [vH2020,2022] [Tang,vH, TS2023], [Karahalios,vH 2023+]

(DALL-E, 2024)

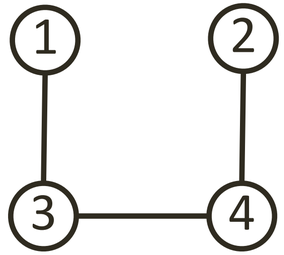
Column Elimination

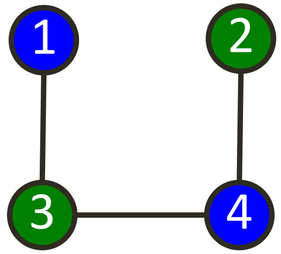
Example Application: Graph Coloring

- Assign a color to each vertex such that adjacent vertices have a different color. Minimize the number of colors.
- MIP model 1:

$$\begin{aligned}
\min \quad & \sum_{k \in K} y_k \\
\text{s.t.} \quad & \sum_{k \in K} x_{ik} = 1 && \forall i \in V \\
& x_{ik} + x_{jk} \leq y_k && \forall (i,j) \in E, k \in K \\
& x_{ik} \in \{0,1\} && \forall i \in V, k \in K \\
& y_k \in \{0,1\} && \forall k \in K
\end{aligned}$$

Dantzig-Wolfe Decomposition: Convexify subset of constraints





$G = (V, E)$

- Relatively weak linear programming relaxation

Graph Coloring: Dantzig-Wolfe Reformulation

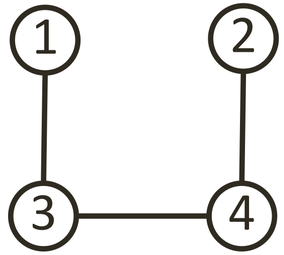
- Each integer solution is a collection of independent sets (subset of non-adjacent vertices), or color classes
- MIP model 2:

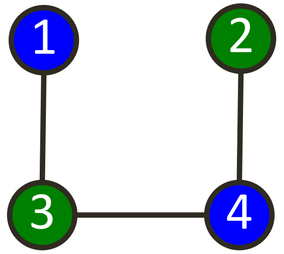
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
1 :	0	1	0	0	0	1	1	0
2 :	0	0	1	0	0	1	0	1
3 :	0	0	0	1	0	0	0	1
4 :	0	0	0	0	1	0	1	0

$$\min \quad \sum_{i \in I} x_i$$

$$\text{s.t.} \quad \sum_{i : v \in i} x_i = 1 \quad \forall v \in V$$

$$x_i \in \{0, 1\} \quad \forall i \in I$$



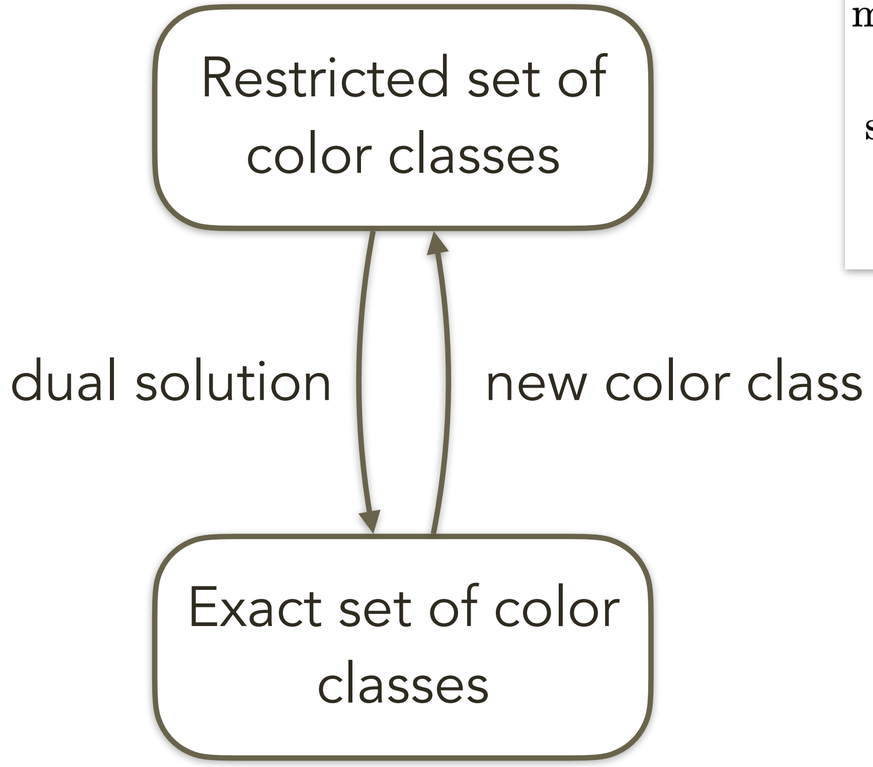


$$I = \{\{\},\{1\},\{2\},\{3\},\{4\},\{1,2\},\{1,4\},\{2,3\}\}$$

- Stronger LP relaxation (but exponential number of variables!)

Solving Linear Program with Column Generation

Master Problem: Linear Program



$$\begin{aligned} \min \quad & \sum_{i \in I'} x_i & \\ \text{s.t.} \quad & \sum_{i : v \in i} x_i = 1 & \forall v \in V \\ & x_i \in \{0,1\} & \forall i \in I' \end{aligned}$$

Integer Optimality: Embed Column Generation LPs within Branch-and-Price

[Desrosiers et al. 1984, Barnhart et al. 1998, Mehrotra and Trick 1996]

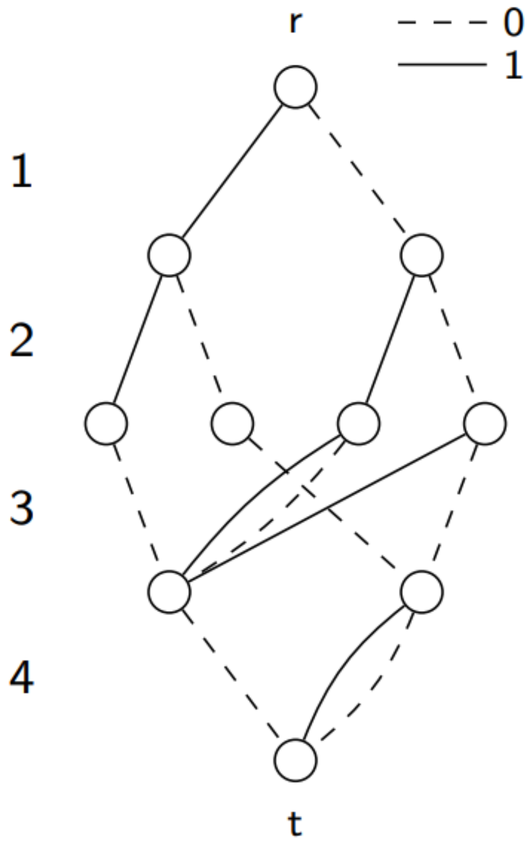
Pricing Problem: Dynamic Program*

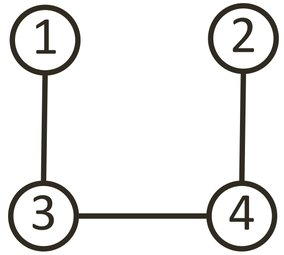
* When pricing problem is NP-hard: Use state-space relaxations or other methods to compute a dual bound

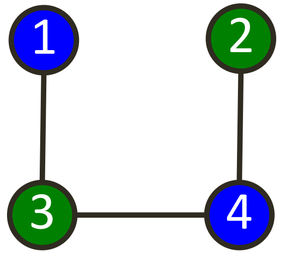
Graph Coloring: Decision Diagram Representation

- We can compactly represent all independent sets (columns) as a decision diagram!

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
$1:$	0	1	0	0	0	1	1	0
$2:$	0	0	1	0	0	1	0	1
$3:$	0	0	0	1	0	0	0	1
$4:$	0	0	0	0	1	0	1	0







$G = (V, E)$

Graph Coloring: Arc-Flow Reformulation

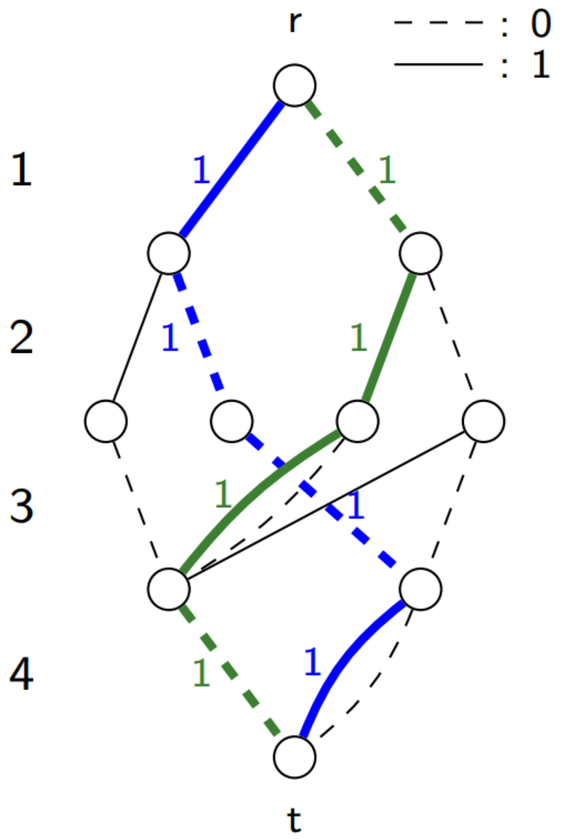
- MIP model 3:

$$\min \quad \sum_{a \in \delta^+(r)} y_a$$

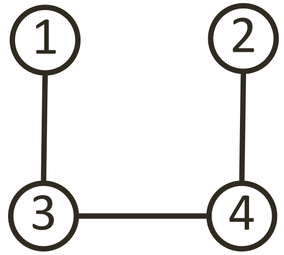
$$\text{s.t.} \quad \sum_{a:\ell(a)=v} y_a = 1 \qquad \forall v \in V$$

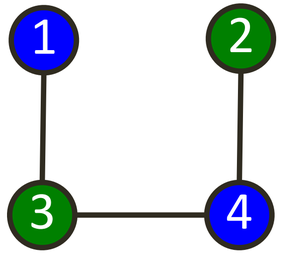
$$\sum_{a \in \delta^-(i)} y_a = \sum_{a \in \delta^+(i)} y_a \qquad \forall i \in N \setminus \{r,t\}$$

$$y_a \in \{0, 1, 2, \ldots, n\} \qquad \forall a \in A$$



$D = (N, A)$



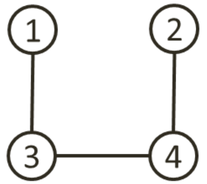


$G = (V, E)$

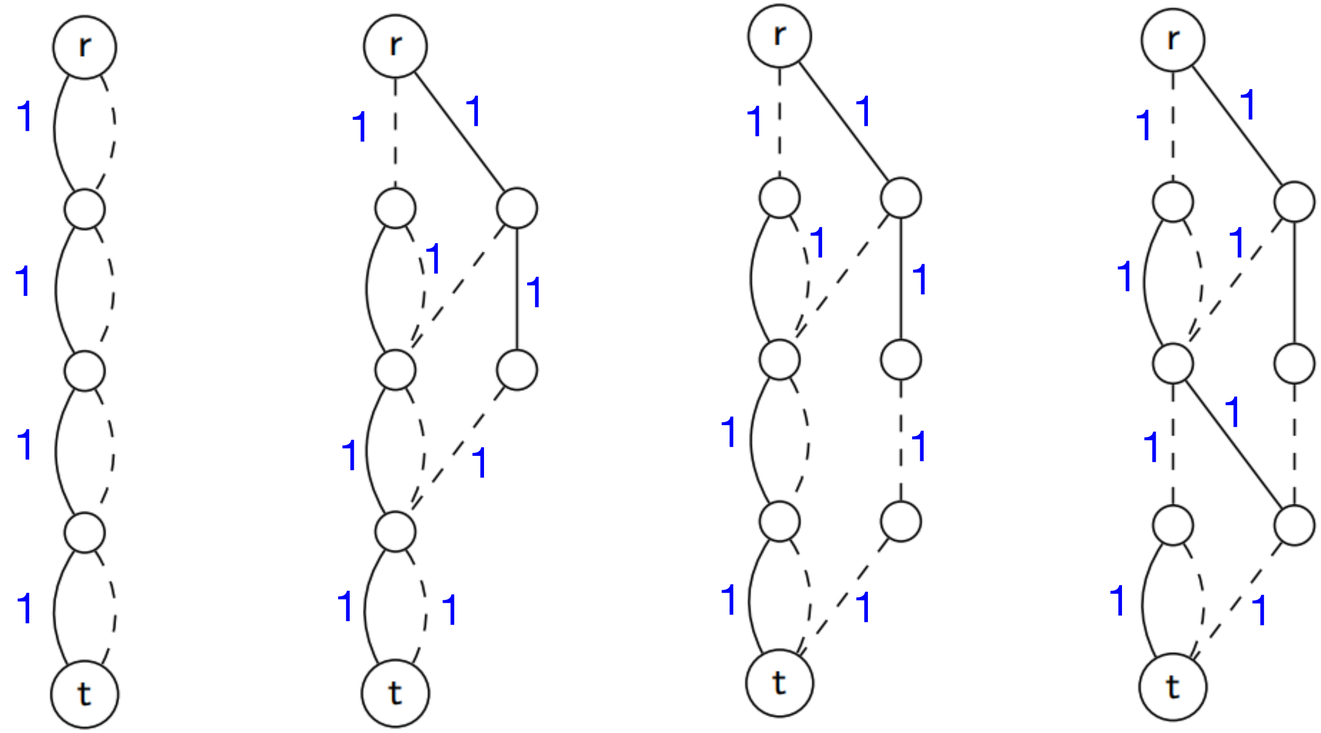
Two Main Computational Challenges

- Exact decision diagrams can be of exponential size
  - Remedy: Use smaller *relaxed* decision diagrams instead
  - Provides lower bound on coloring number
- Solving the arc flow formulation is NP-hard
  - Less relevant in practice: MIP solvers scale well
  - But we can also use LP relaxation (polynomial)

Column Elimination: Iterative Refinement



input graph



lower bound:	1	2	2	2
flow paths:	(1,1,1,1)	(1,1,0,1) (0,0,1,0)	(1,1,0,0) (0,0,1,1)	(1,0,0,1) (0,1,1,0)
conflicting edge:	(1,3)	(2,4)	(3,4)	-

Optimal!

Analysis of Overall Procedure

Lemma: Conflicts can be found in polynomial time (in the size of the decision diagram) via a path decomposition of the flow

Lemma: Eliminating k conflicts increases the size by at most $O(kn)$ nodes

- Eliminating one conflict increases each layer by at most one node

Lemma: In each iteration, compilation via conflict elimination produces a valid lower bound

Lemma: Eliminating all conflicts yields the exact decision diagram

Theorem: The algorithm terminates with an optimal solution (if time permits)

[vH IPCO 2020, Math. Prog. 2022]

Recall Arc-Flow Reformulation

- MIP model 3:

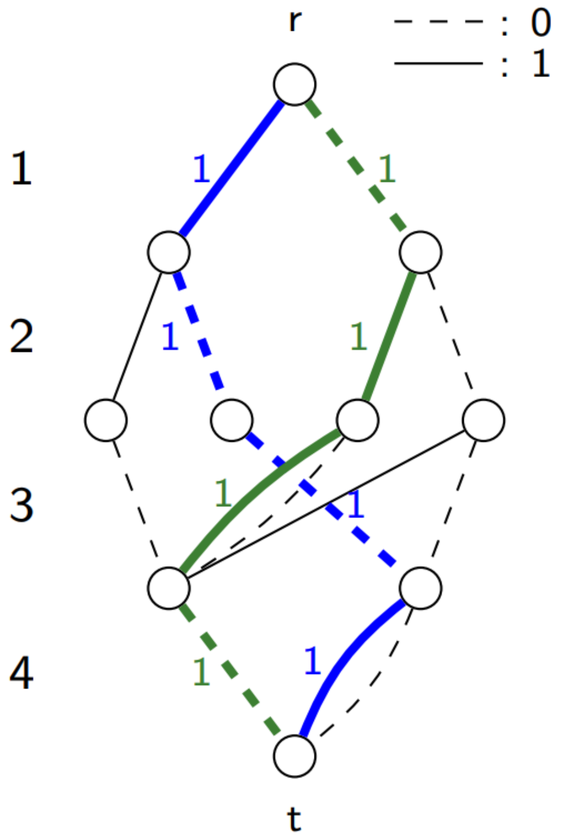
$$\min \quad \sum_{a \in \delta^+(r)} y_a$$

$$\text{s.t.} \quad \sum_{a:\ell(a)=v} y_a = 1 \qquad \forall v \in V$$

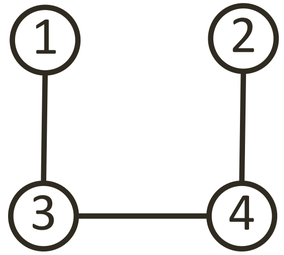
$$\sum_{a \in \delta^-(i)} y_a = \sum_{a \in \delta^+(i)} y_a \qquad \forall i \in N \setminus \{r,t\}$$

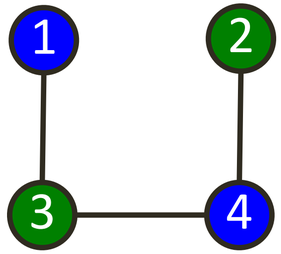
$$y_a \in \{0, 1, 2, \ldots, n\} \qquad \forall a \in A$$

Arc Flow Reformulation: Convexify integer points as network flow



$D = (N, A)$





$G = (V, E)$

Characterization of General Arc-Flow Bounds

$$
\begin{aligned}
z := \max \quad & \mathbf{cx} \\
\text{s.t.} \quad & \mathbf{Ax} \leq \mathbf{b} \\
X \quad & \boxed{\begin{aligned} &\mathbf{Dx} \leq \mathbf{d} \\ &\mathbf{x} \in \mathbb{Z}^n_+, \end{aligned}}
\end{aligned}
$$

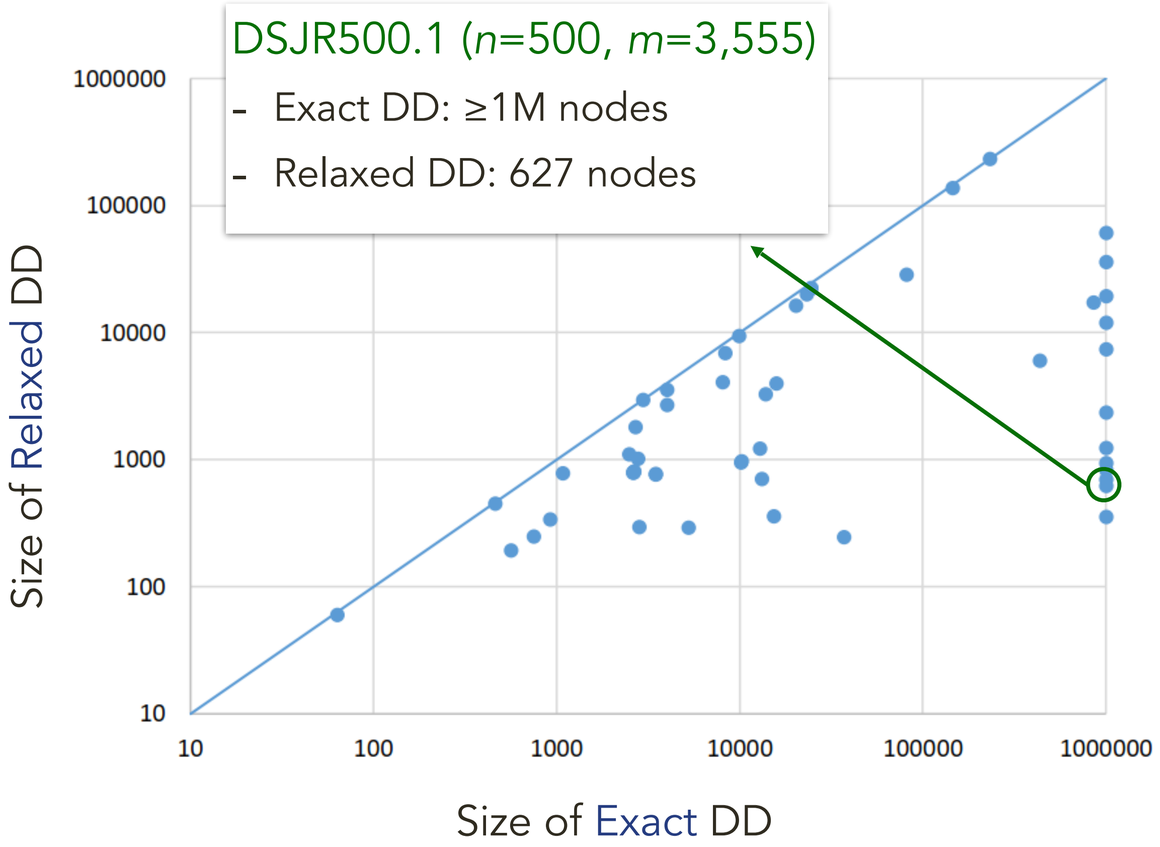
- Dantzig-Wolfe bound z^{LP}_{DW} = decomposition bound z_D
- How about Arc-Flow bound z^{AF}_{LP} ?
  - Depends on the arc flow reformulation
  - Dynamic programming state space, decision diagram, problem-specific, …

Definition: Arc-Flow = network + projection matrix resulting in $\mathrm{conv}(X)$

Theorem: Under this definition, $z_D = z^{LP}_{DW} = z^{LP}_{AF} \leq z_{LP}$ [Yamín, vH, Ralphs, 2026]

Corollary: Immediately applies to relaxed DDs and DP state-space relaxations

Performance of Column Elimination



DIMACS Graph Coloring Instances

- Network flow model over decision diagram
- Optimality can be proven with small DD

[vH 2020,2022]

Vehicle Routing Problem with Time Windows

- For several instances column elimination finds better bounds than VRPSolver [Pessoa+20]
- Column Elimination **closes open instance** C2_10_1 with 1,000 locations

Graph Multi-Coloring Problem

- Column Elimination **closes five open** benchmark instances

Pickup-and-Delivery Problem with Time Windows

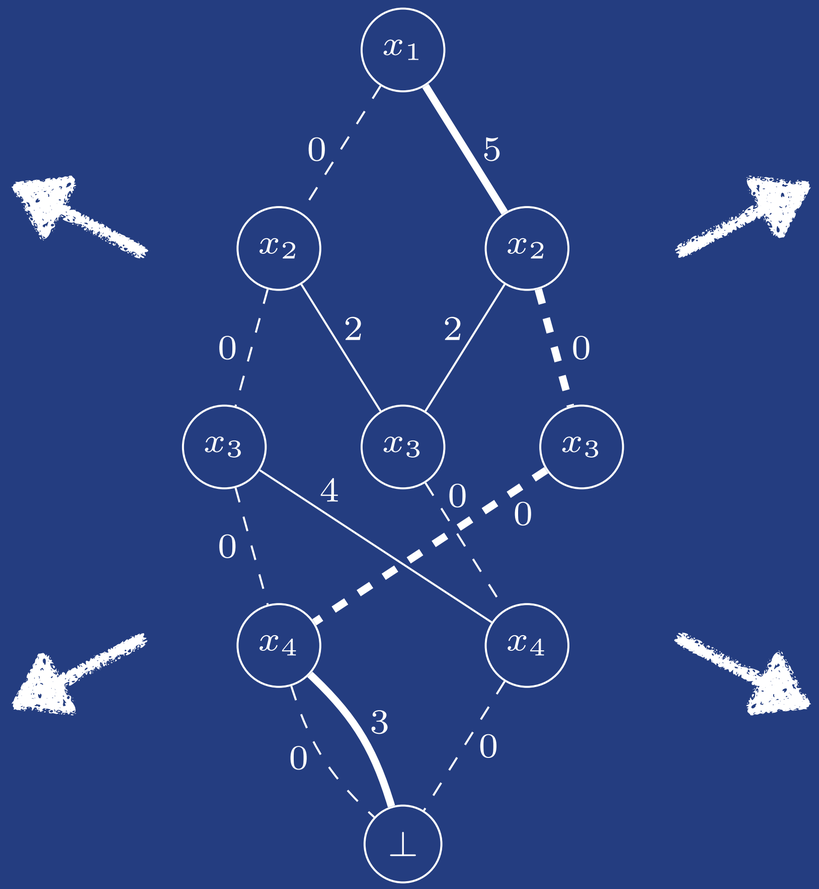
- Column Elimination **closes six open** benchmark instances with 1,000 locations

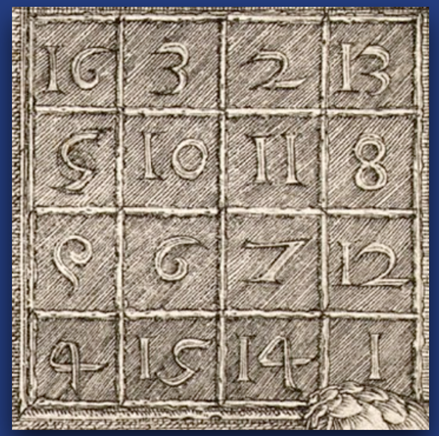
[Karahalios & vH, 2026]

Implemented in optimization solver Hexaly (7/2025)

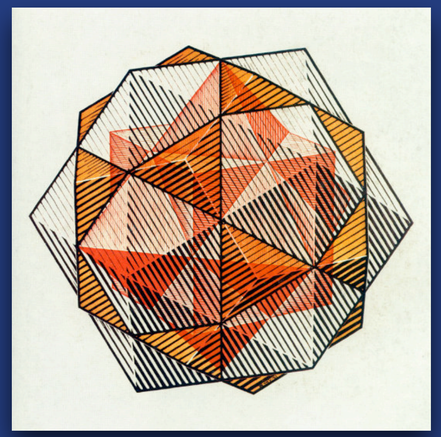
- Routing, scheduling, packing,… [Coppé JFPC 2026]

Dynamic Programming





Constraint Programming



Integer Programming

Column Elimination