Logic, Optimization and Data Analytics

John Hooker
Carnegie Mellon University

United Technologies Research Center, Cork, Ireland
August 2015

Thesis

- **Logic** and **optimization** have an **underlying unity.**
  - Ideas from one area can solve problems in the other.
  - Ideas from both can be applied to **data analytics**.

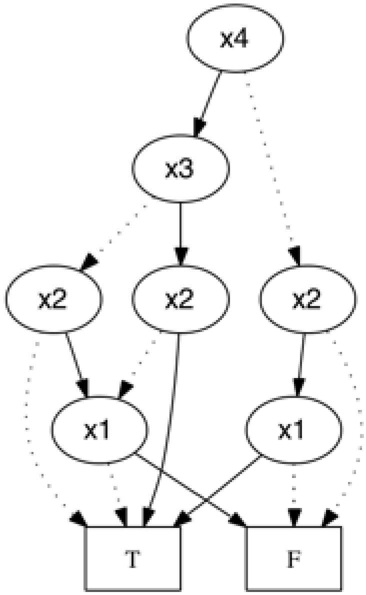
Research Programs

- Apply **logic-based methods** to **optimization**.
  - Decision diagrams.
  - Logic-based Benders decomposition
- Apply **optimization methods** to **logical inference** and **data analytics**.
  - Logic-based Benders decomposition.
  - Linear, integer, and nonlinear programming.

Decision Diagrams

- Graphical encoding of a boolean function
  - Historically used for circuit design & verification
  - Adapt to optimization and CP

Hadžić and JH (2006, 2007)



Decision Diagrams

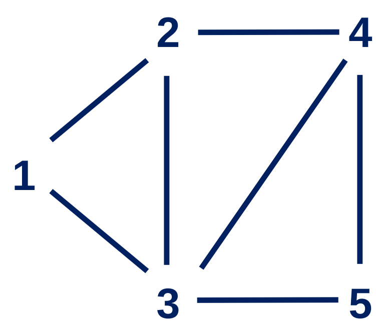
- Collaborators…
  - Henrik Andersen
  - David Bergman*
  - André Ciré**
  - Tarik Hadžić
  - Samid Hoda
  - Willem-Jan van Hoeve
  - Barry O’Sullivan
  - Peter Tiedemann

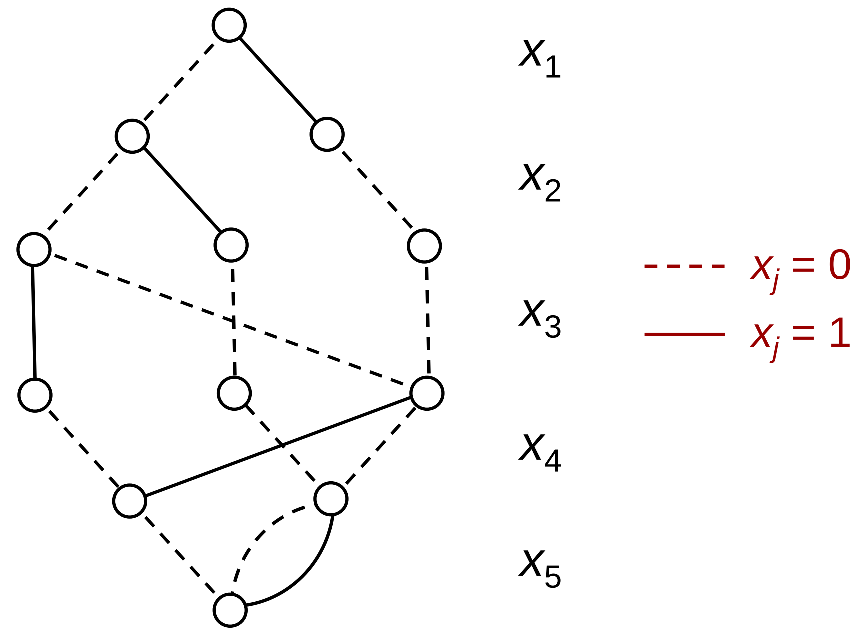
*****2014 *Doctoral Thesis Award*, Association for CP
**2014 *Best Student Paper Award,* INFORMS Computing Society

Decision Diagrams

Example:
Stable set problem

Find max-weight subset
of nonadjacent vertices

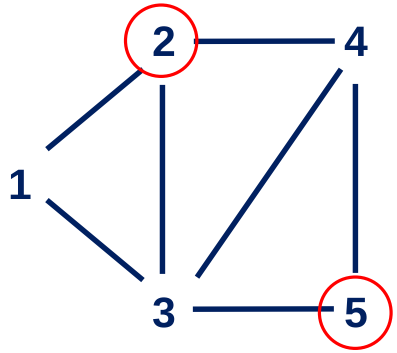


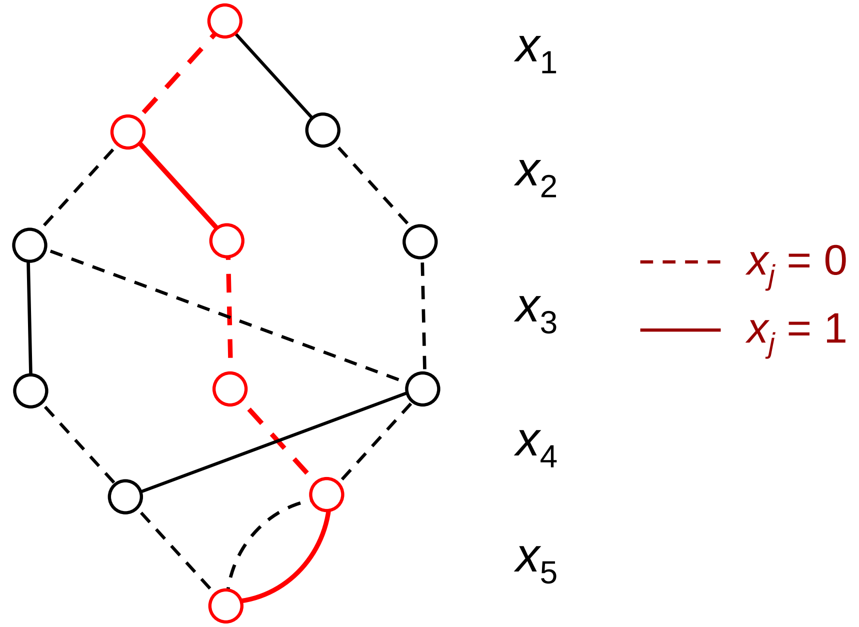


Decision Diagrams

Example:
Stable set problem

Find max-weight subset
of nonadjacent vertices



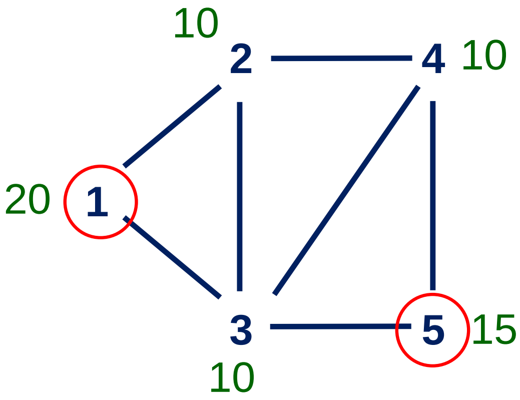


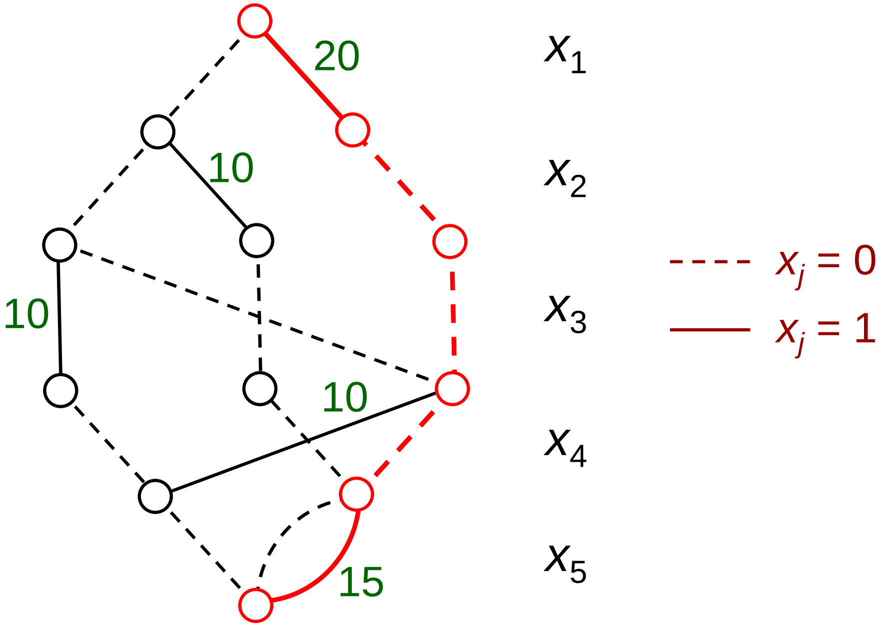
Each path corresponds to a feasible solution.

Decision Diagrams

Example:
Stable set problem

Find max-weight subset
of nonadjacent vertices





Longest path corresponds to an optimal solution.

Decision Diagrams

- Key idea: Use **relaxed** decision diagrams
  - Represent a superset of the feasible set
    - … with a **limited-width** diagram.
  - First used to strengthen **propagation** in CP solvers.
    - Reduced 1 million node search trees to 1 node.

Andersen, Hadžić, JH, Tiedemann (2007)

Decision Diagrams

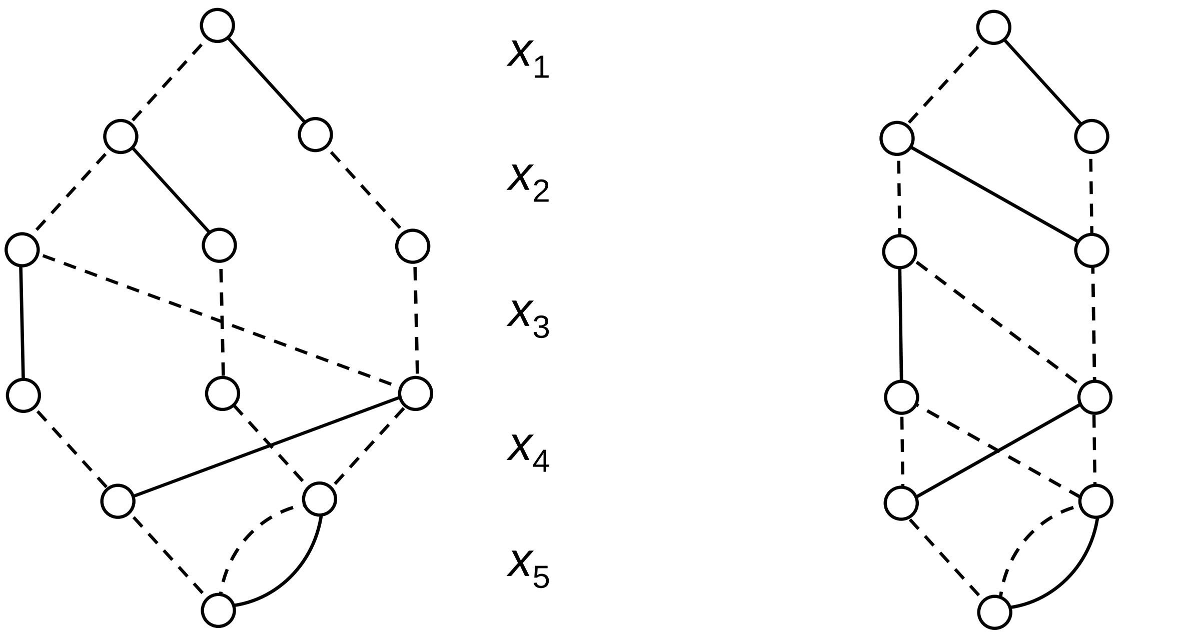
- Key idea: Use **relaxed** decision diagrams
  - Represent a superset of the feasible set
    - … with a **limited-width** diagram.
  - First used to strengthen **propagation** in CP solvers.
    - Reduced 1 million node search trees to 1 node.

Andersen, Hadžić, JH, Tiedemann (2007)

- Shortest (longest) path in the decision diagram provides a **bound** on optimal value.
  - Leads to general-purpose **optimization** method.

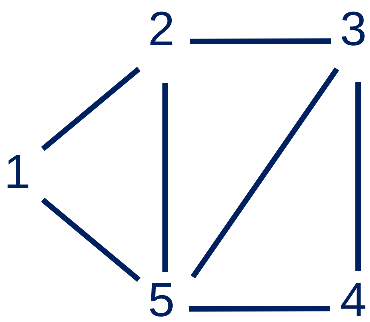
Bergman, Ciré, van Hoeve, JH (2013)

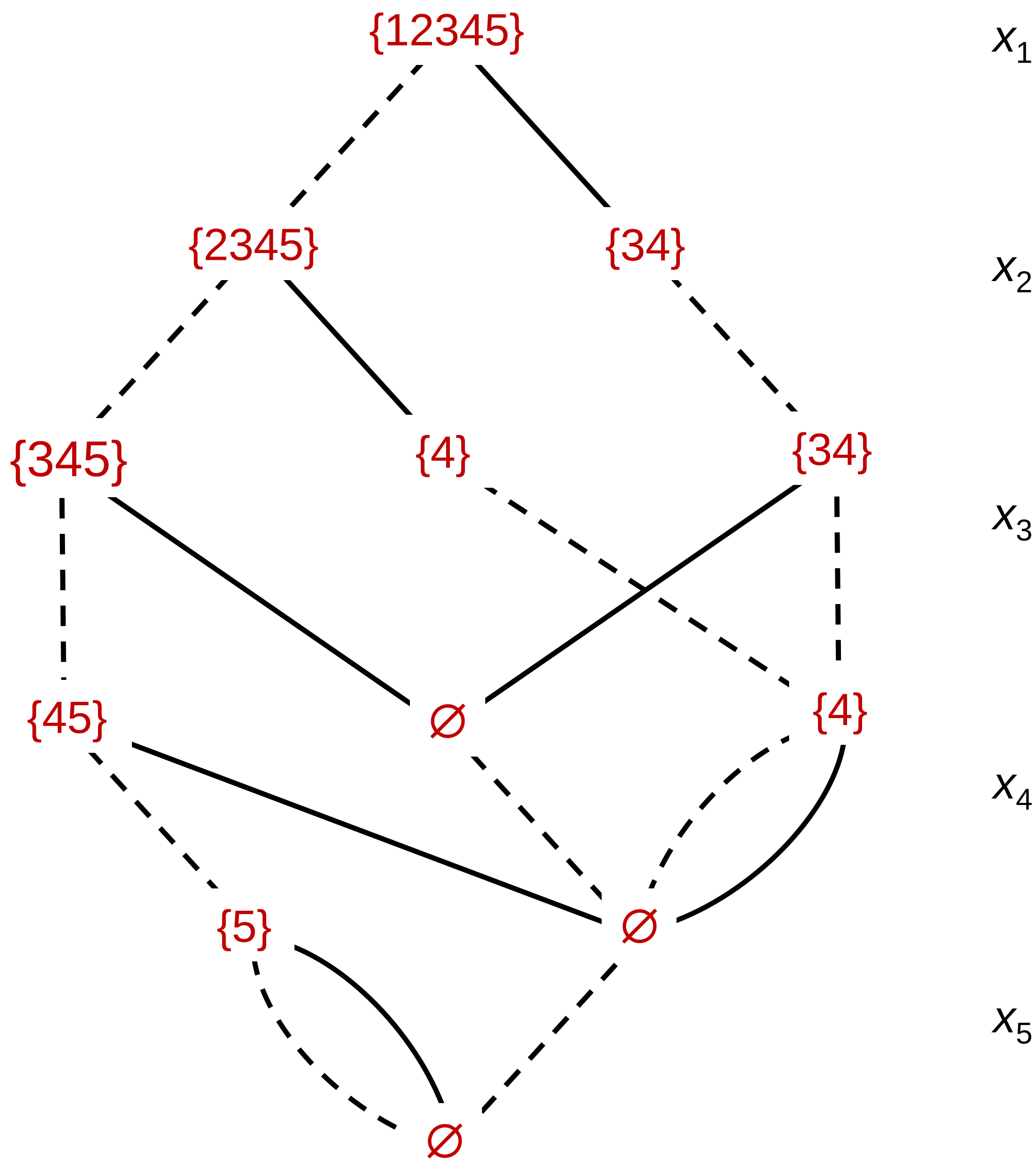
Decision Diagrams



Exact diagram
Width = 3

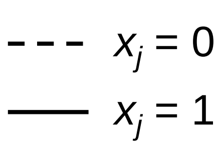
Relaxed diagram
Width = 2

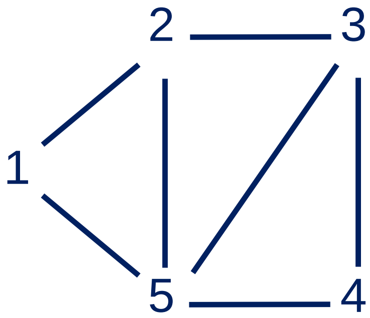




Exact DD for
stable set problem

To build DD,
associate **state**
with each node





{12345}

x_1

x_2

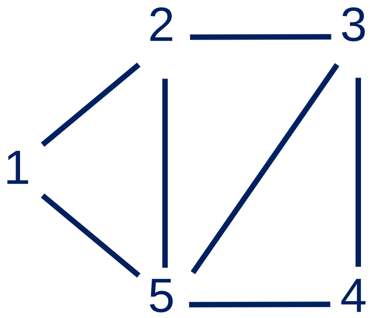
Exact DD for
stable set problem

x_3

x_4

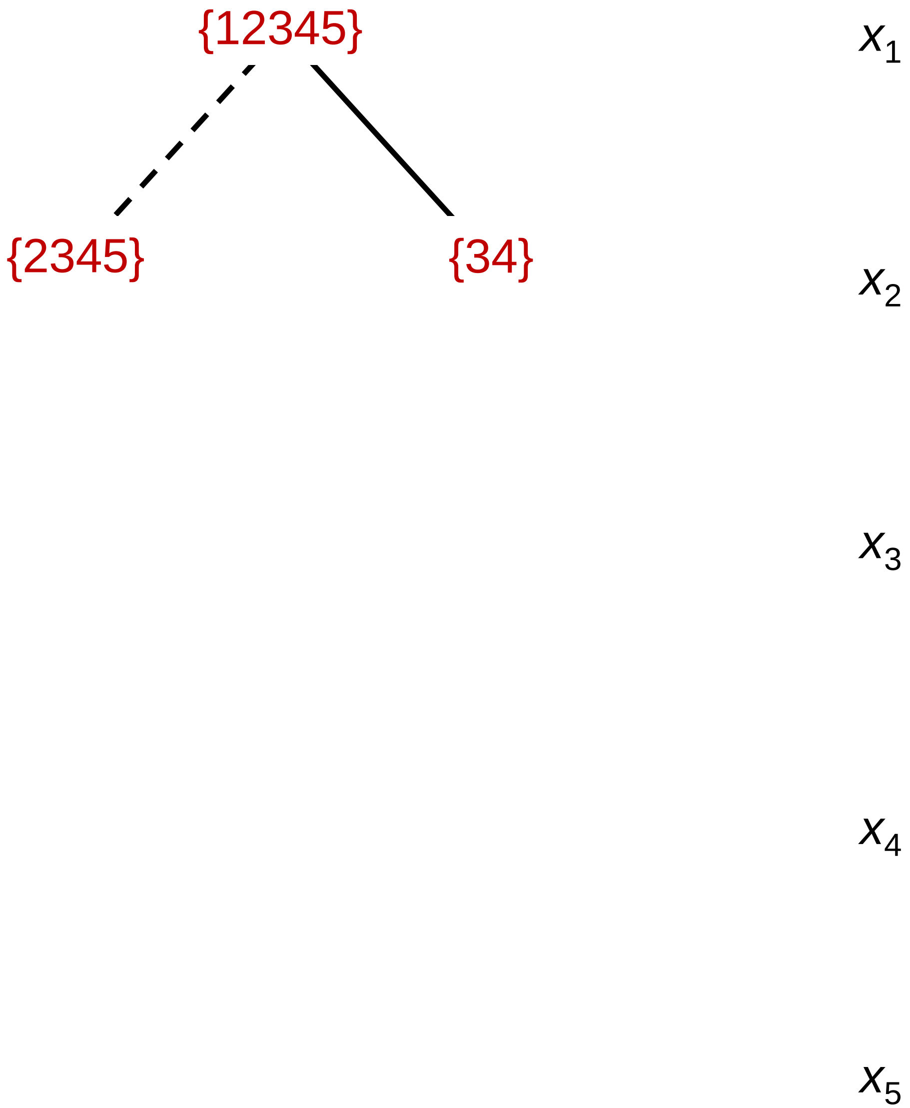
To build DD,
associate **state**
with each node

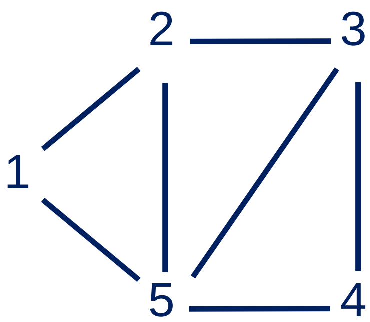
x_5

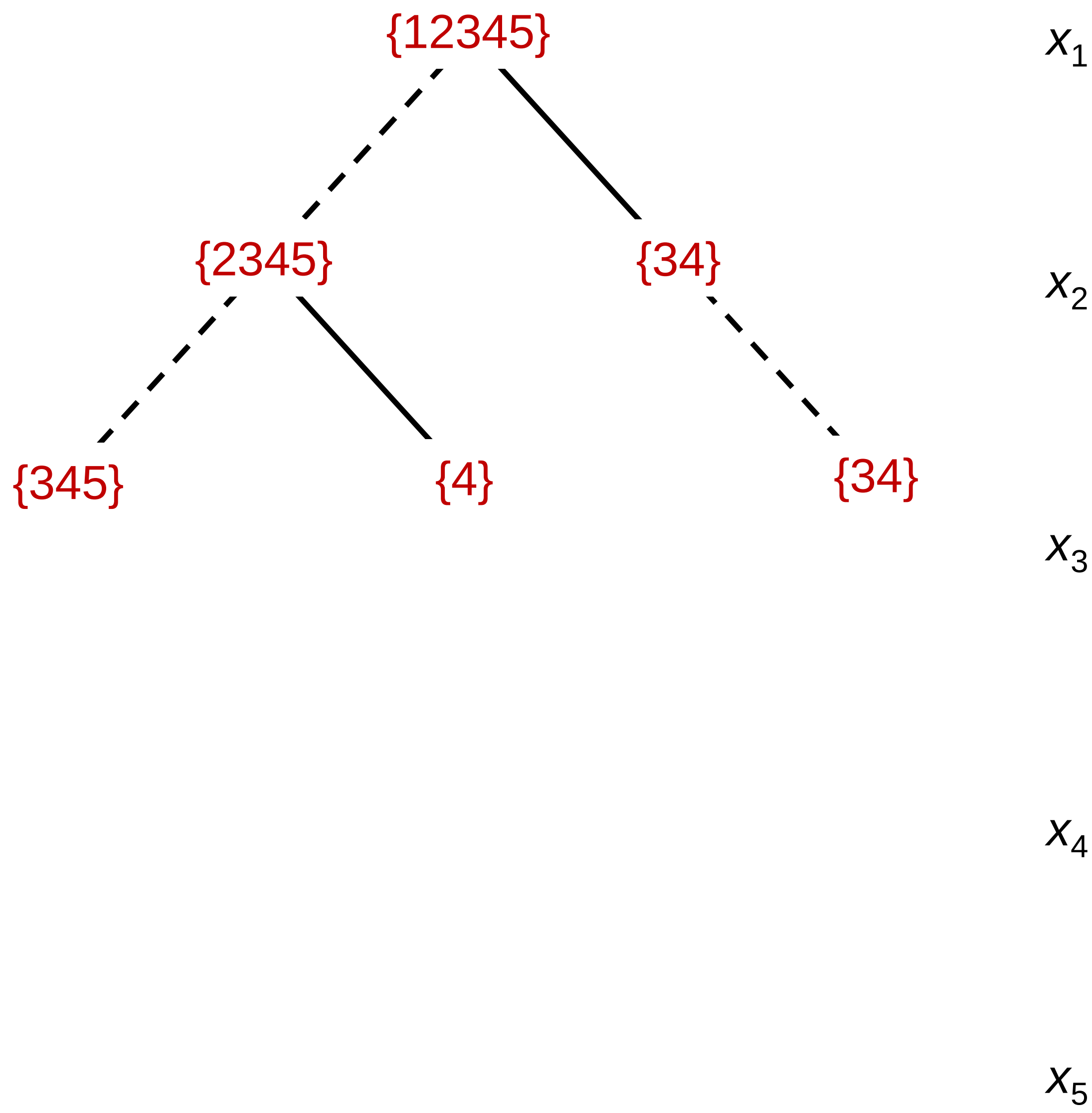


Exact DD for
stable set problem

To build DD,
associate **state**
with each node

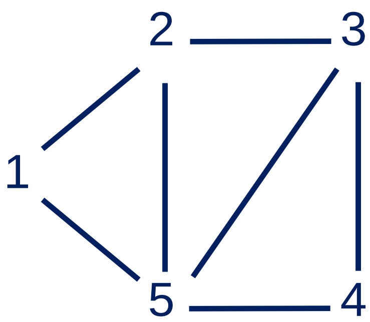


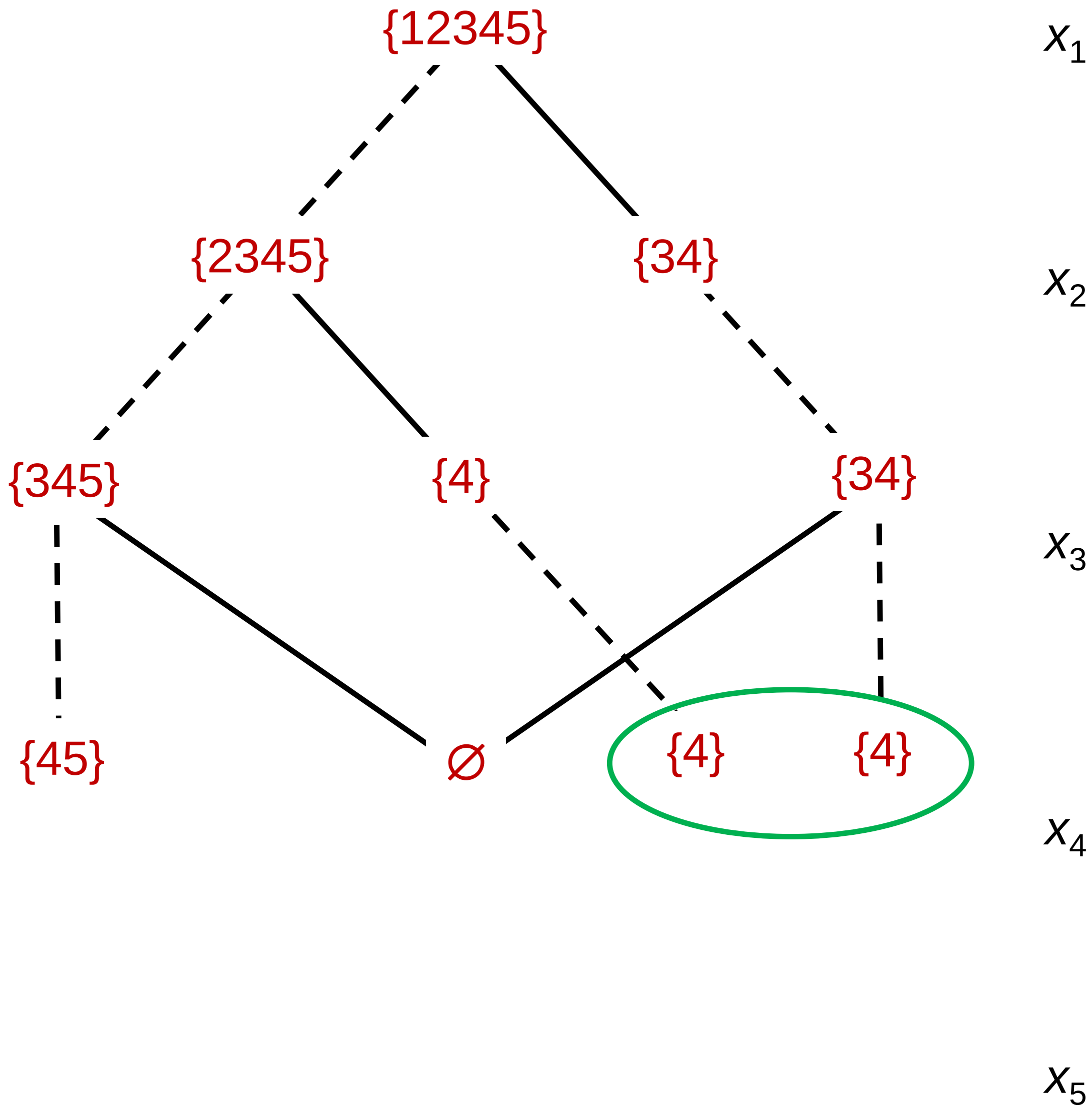




Exact DD for stable set problem

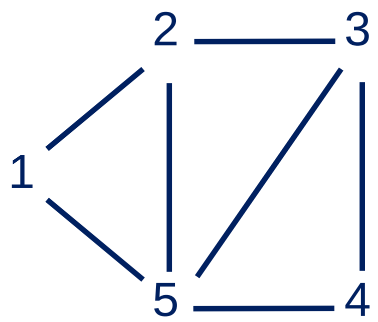
To build DD, associate **state** with each node

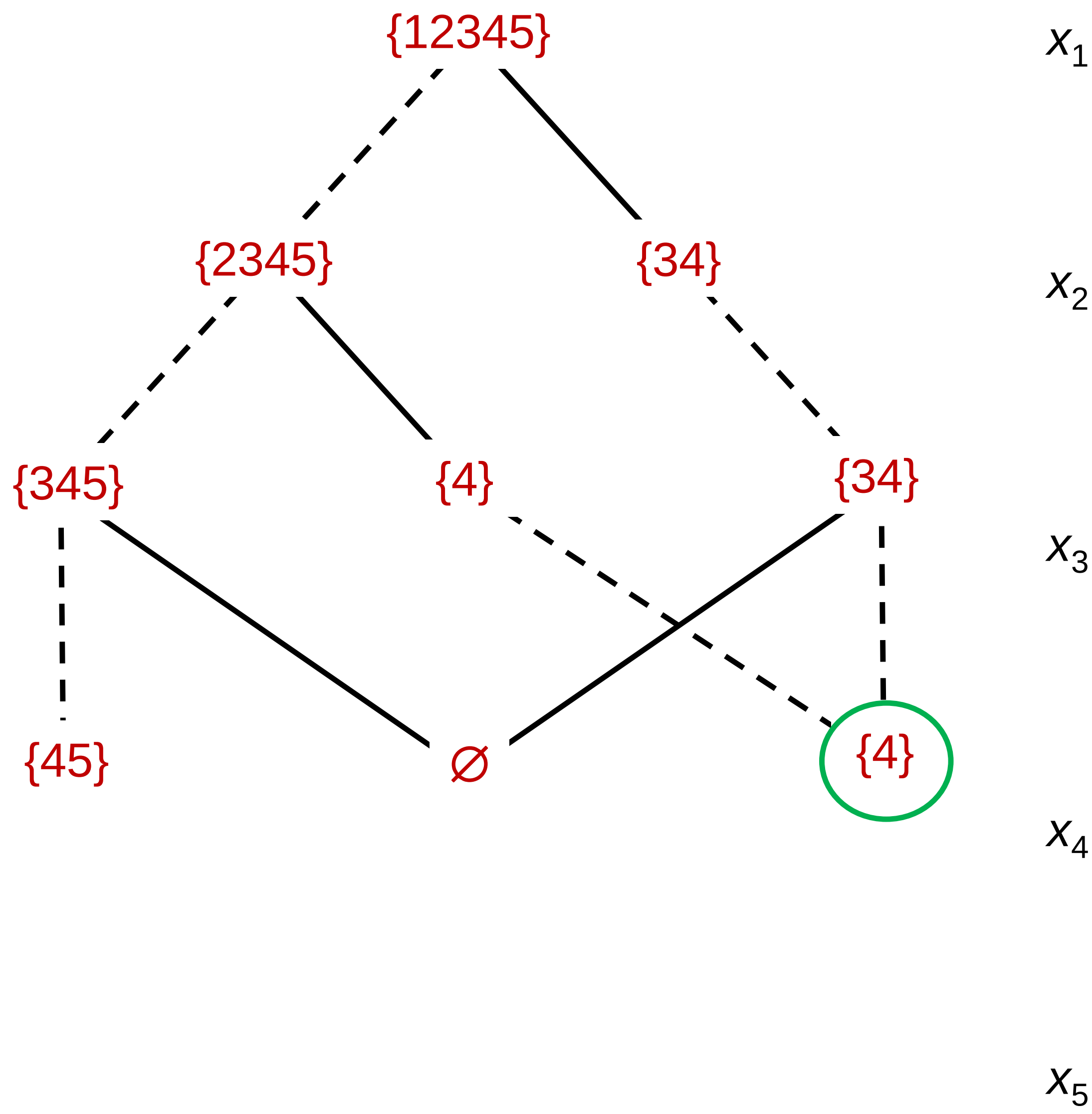




Exact DD for stable set problem

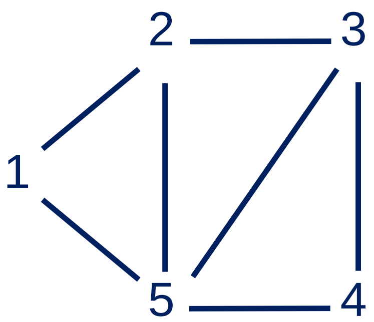
Merge nodes that correspond to the same state

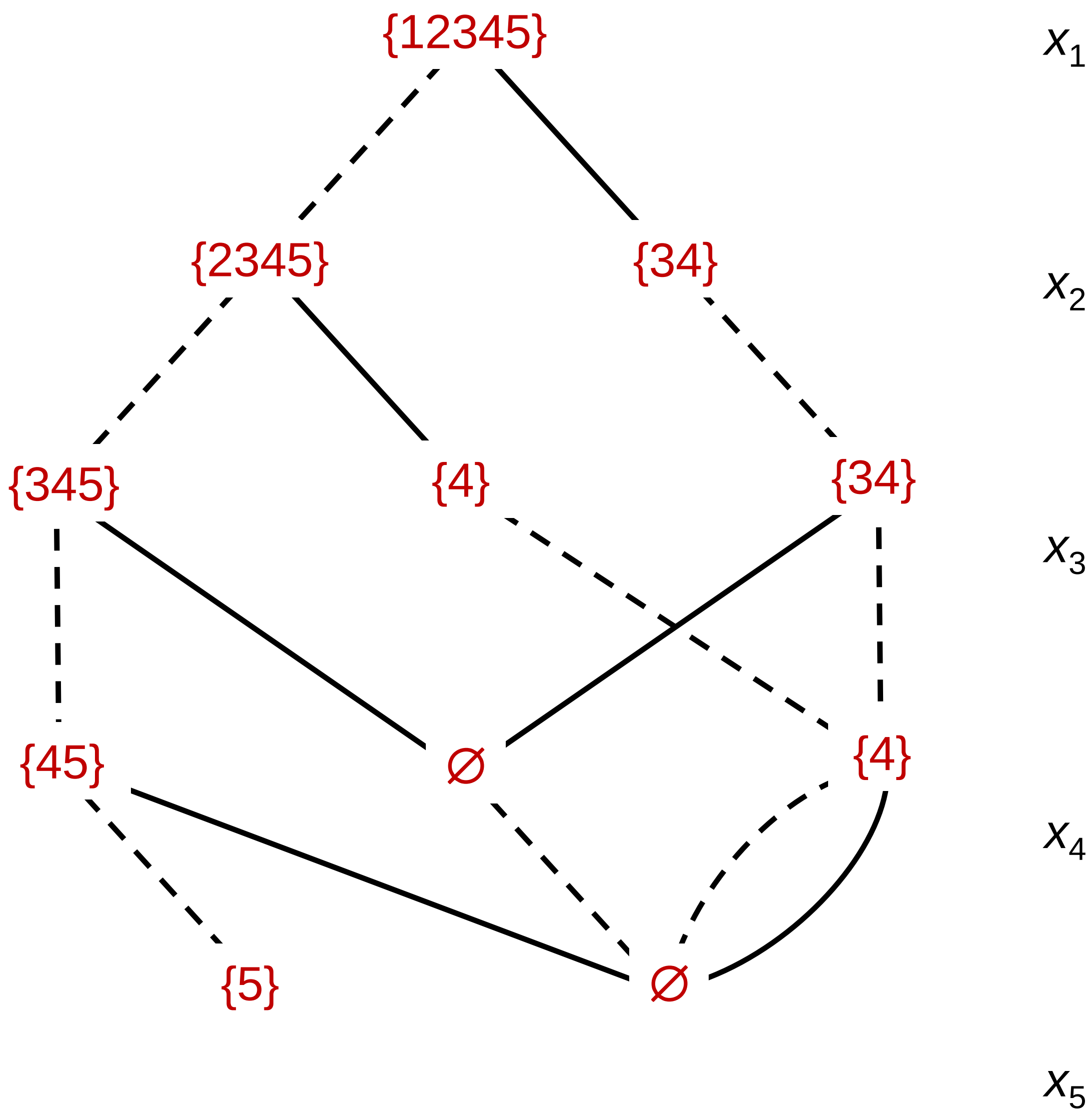




Exact DD for
stable set problem

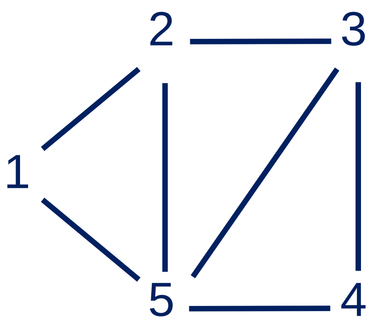
Merge nodes that
correspond to the
same state

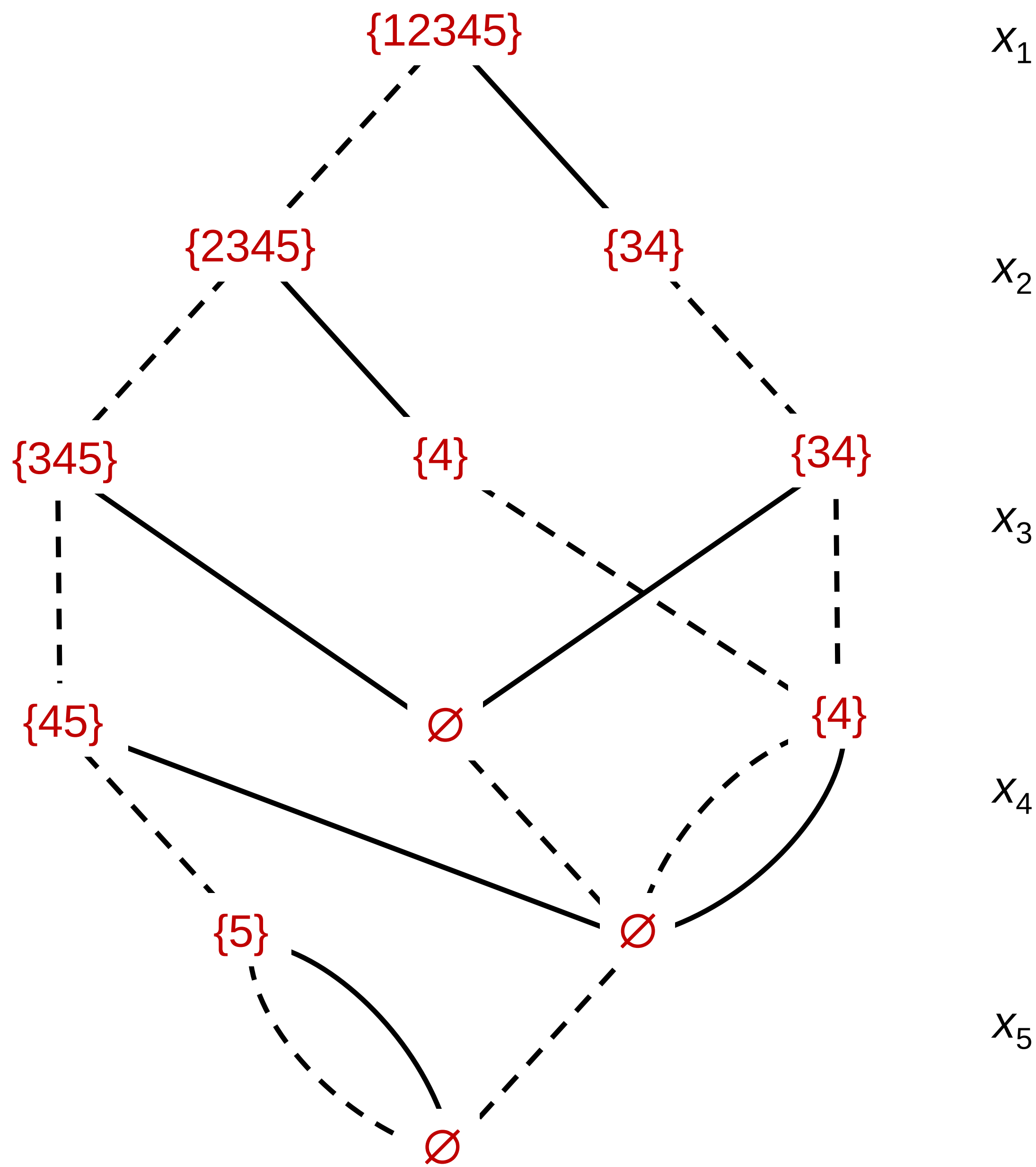




Exact DD for
stable set problem

To build DD,
associate **state**
with each node



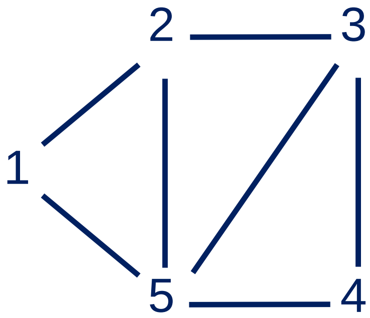


Exact DD for
stable set problem

To build DD,
associate **state**
with each node

Relaxation Bounding

- To obtain a bound on the objective function:
  - Use a **relaxed** BDD
  - Analogous to LP relaxation in IP
  - This relaxation is **discrete**.
  - Doesn’t require the linear inequality formulation of IP.



{12345}

x_1

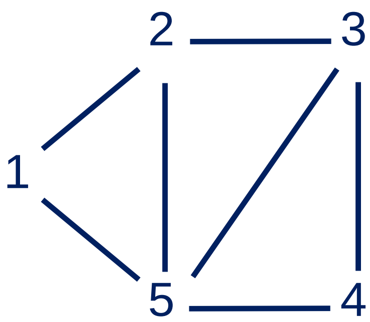
x_2

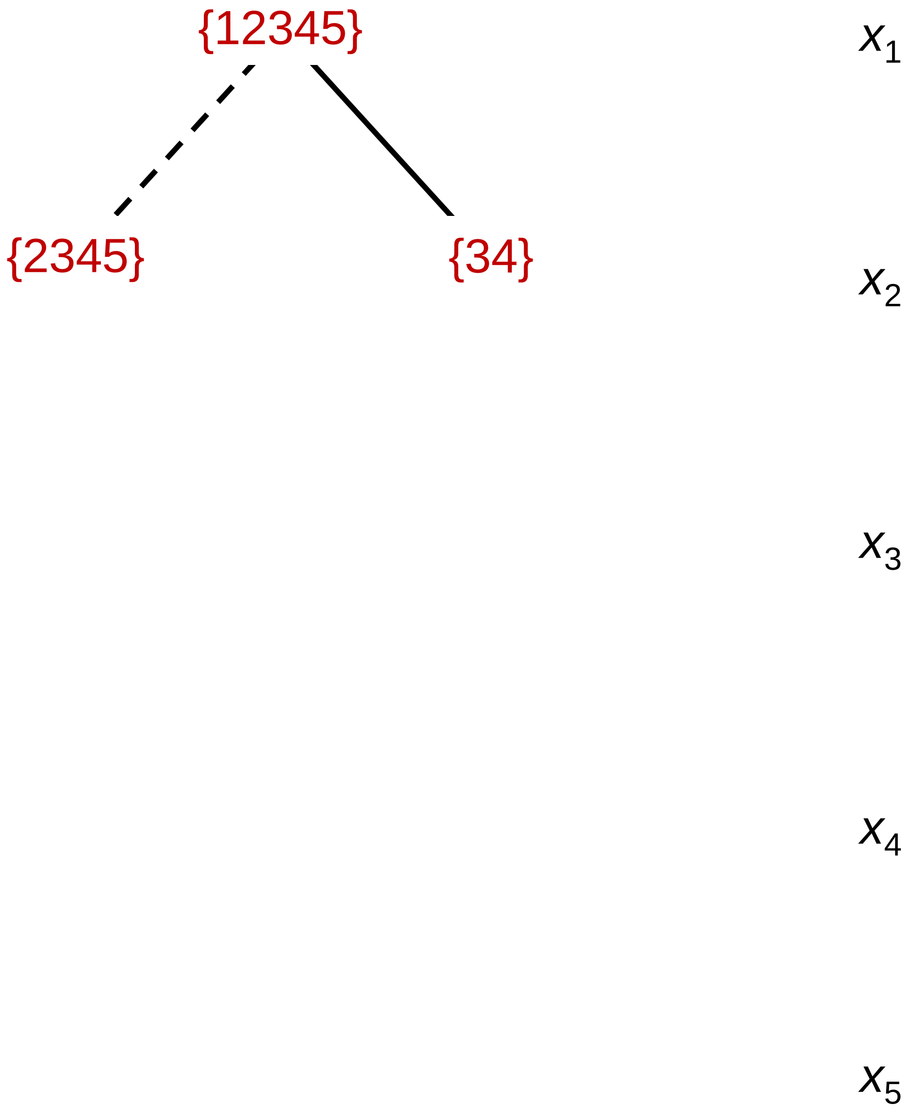
x_3

x_4

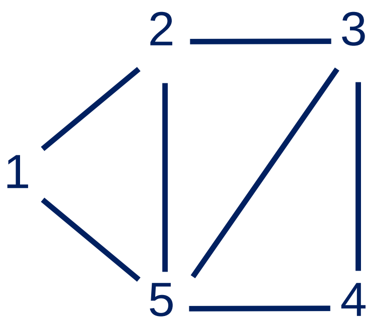
x_5

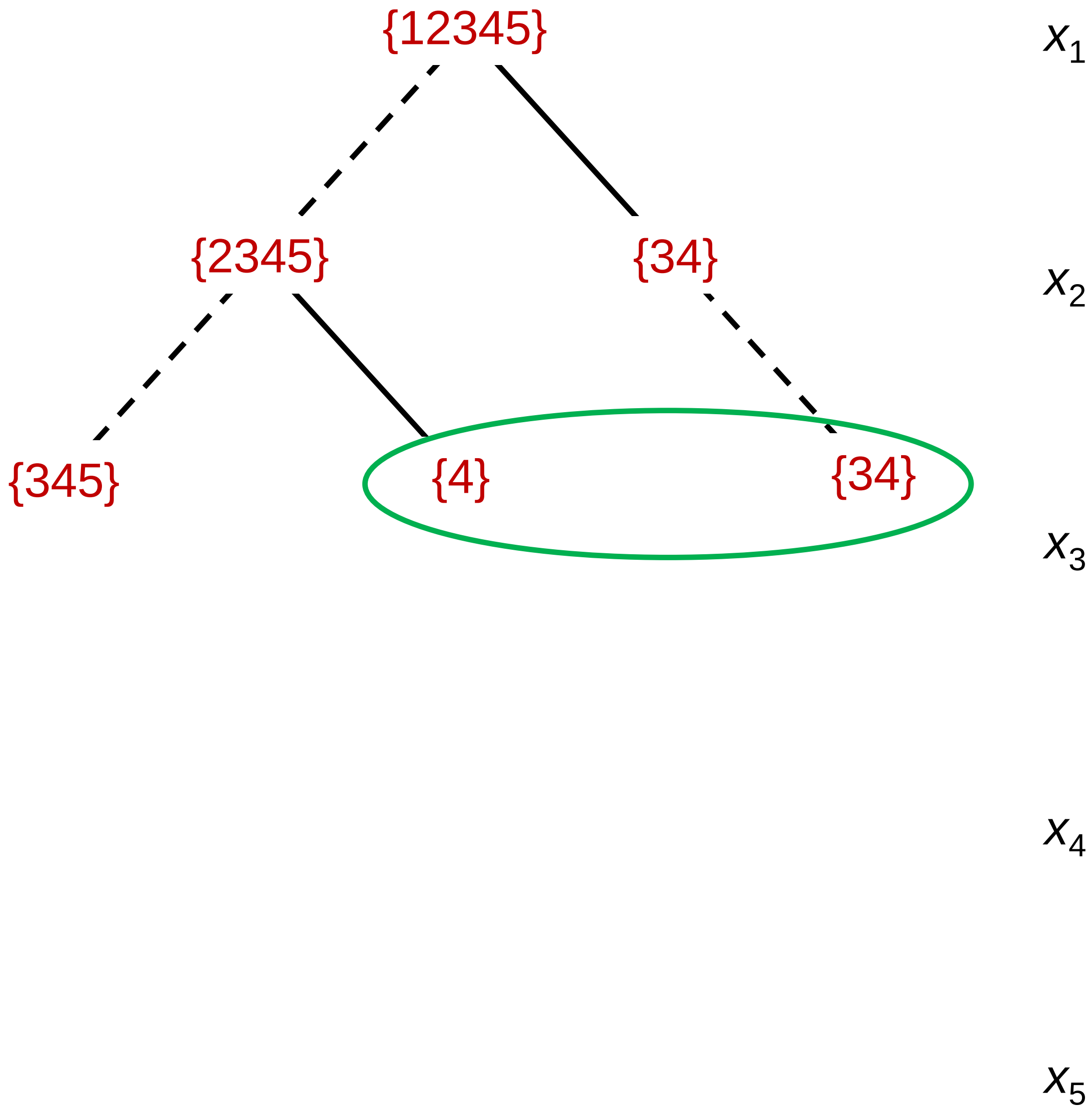
To build **relaxed** BDD, merge some additional nodes as we go along



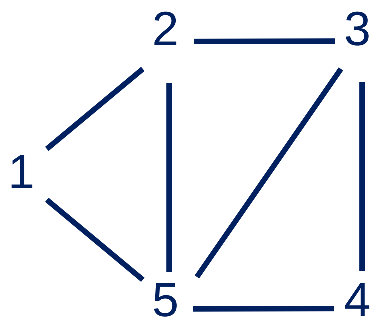


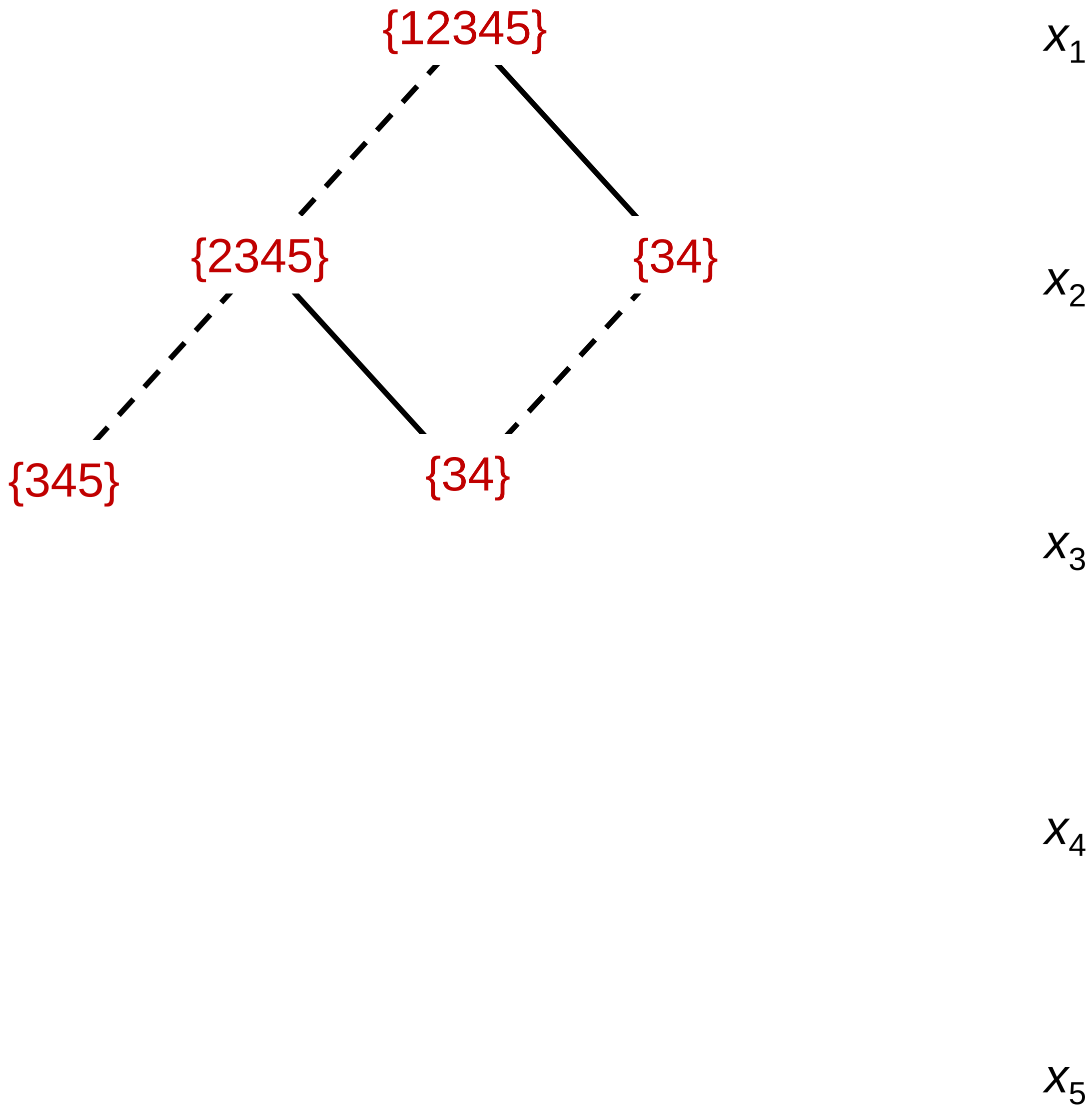
To build **relaxed** BDD, merge some additional nodes as we go along



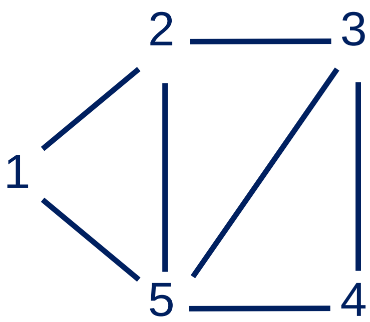


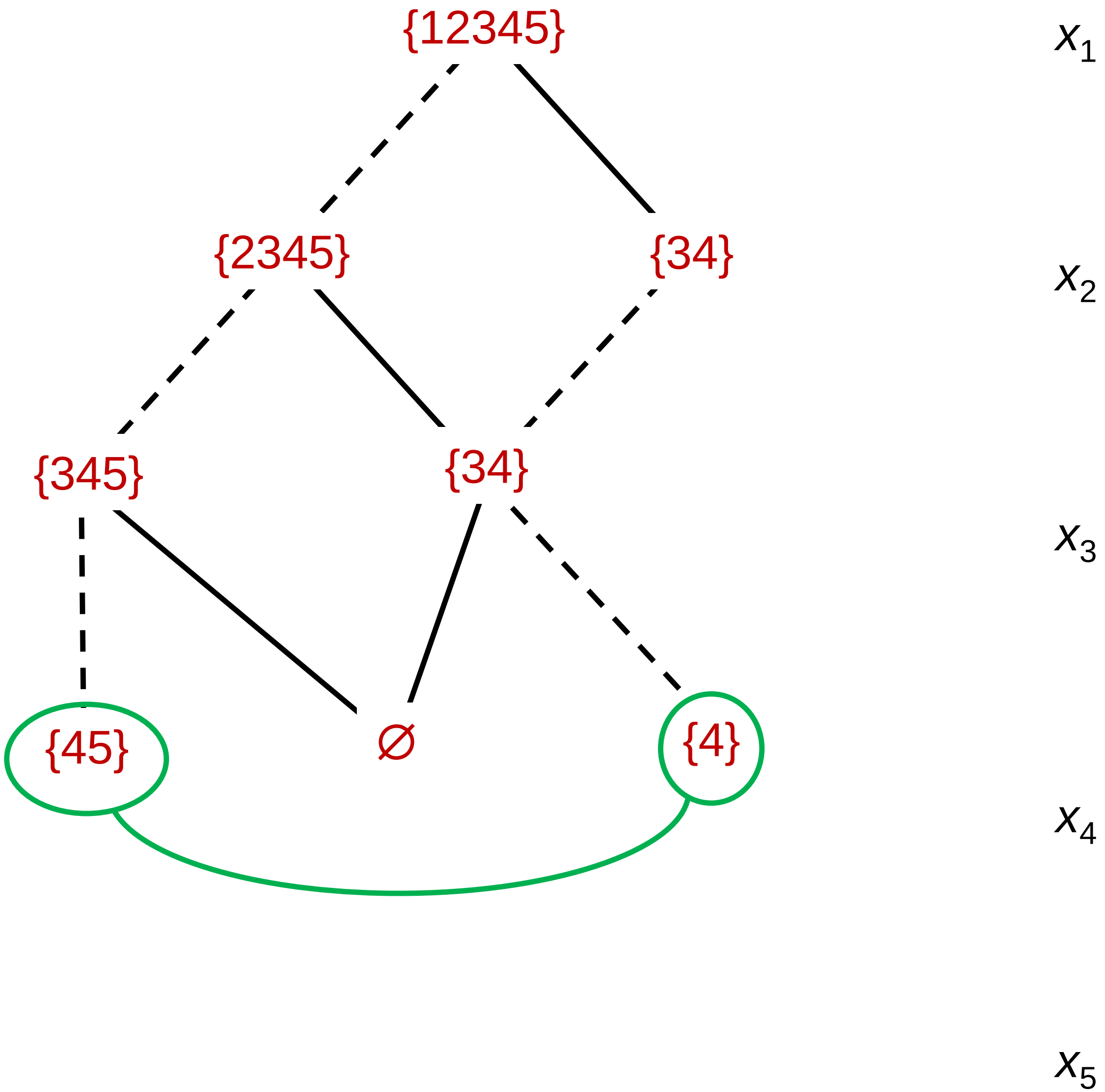
To build **relaxed** BDD, merge some additional nodes as we go along



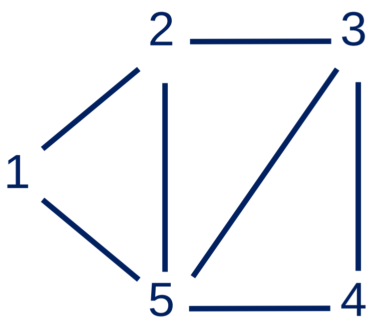


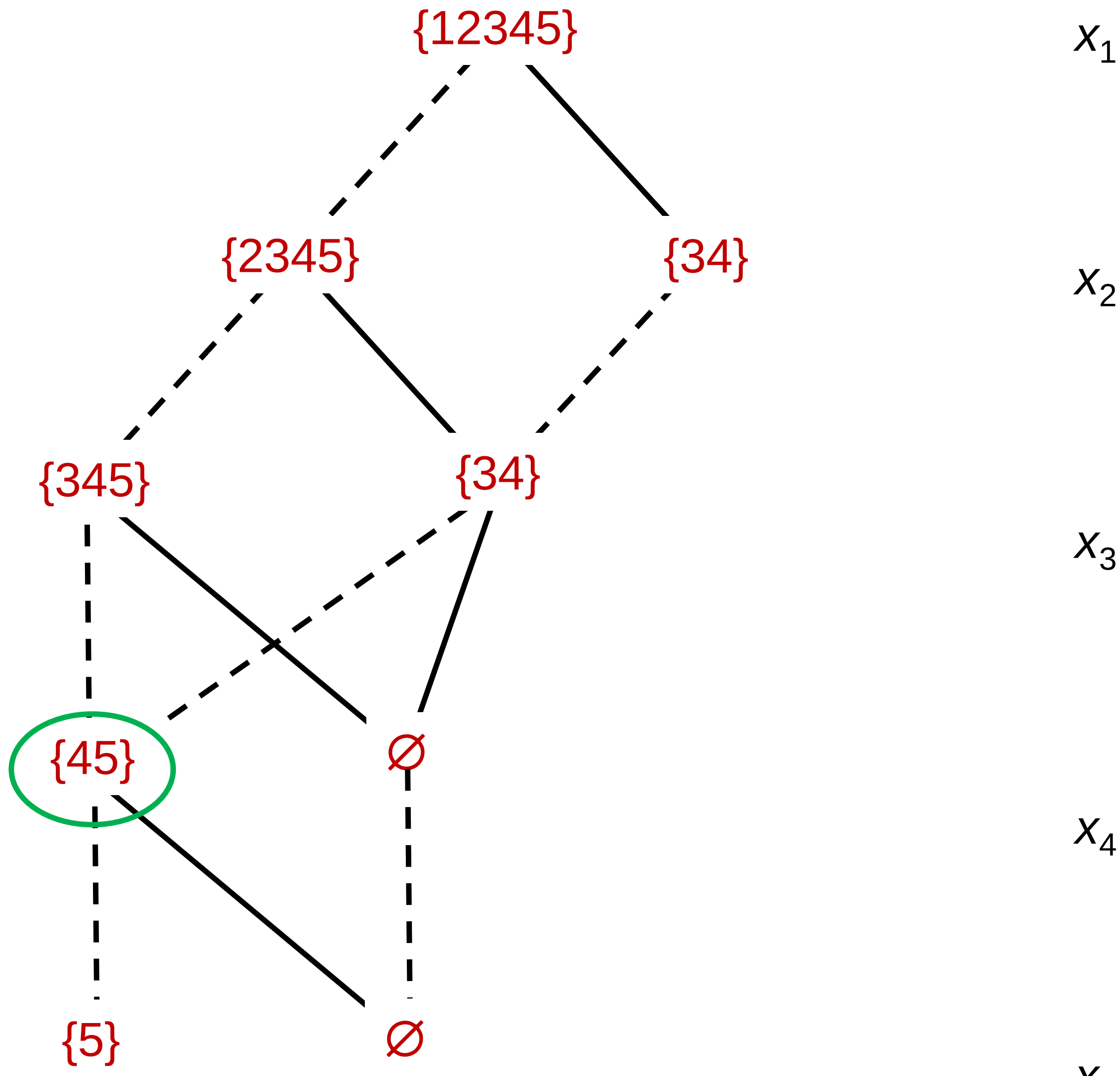
To build **relaxed** BDD, merge some additional nodes as we go along



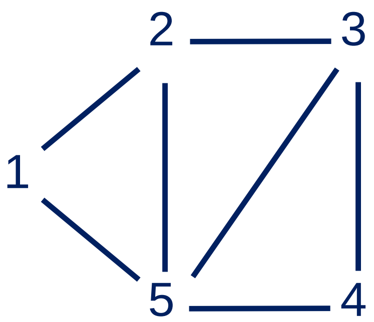


To build **relaxed** BDD, merge some additional nodes as we go along



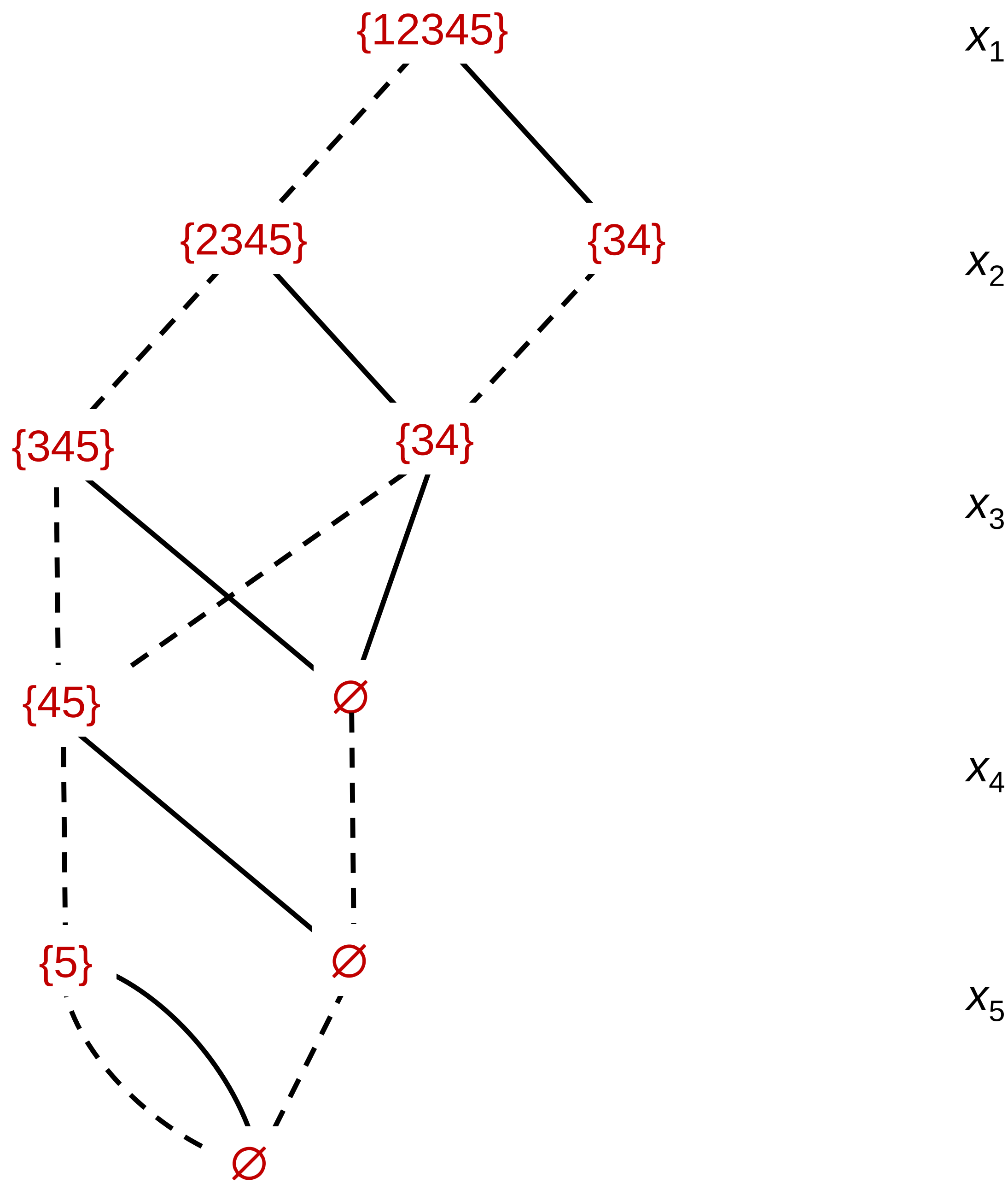


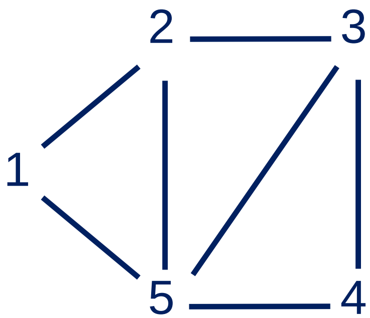
To build **relaxed** BDD, merge some additional nodes as we go along



Width = 2

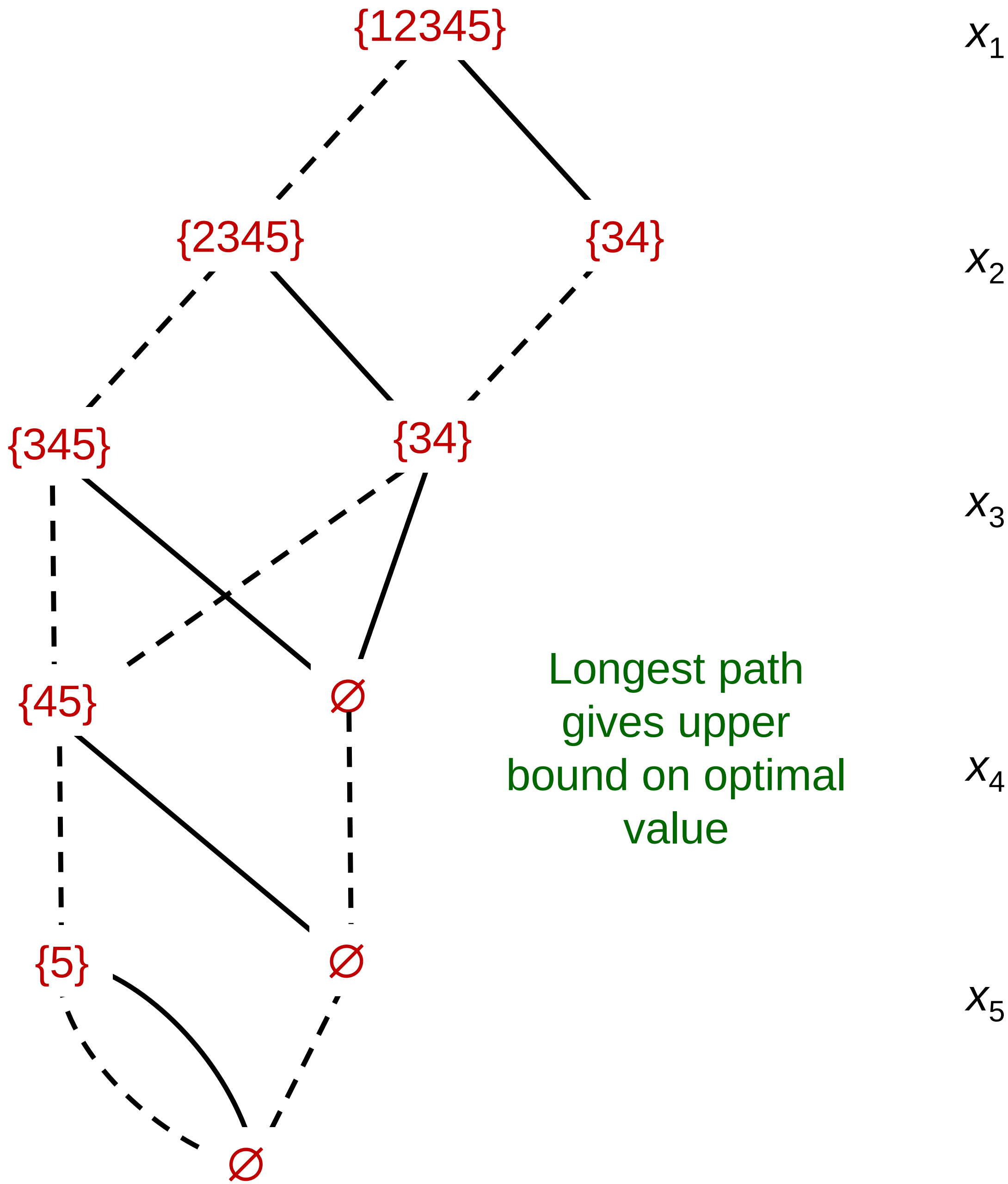
Represents 11 solutions, including 9 feasible solutions





Width = 2

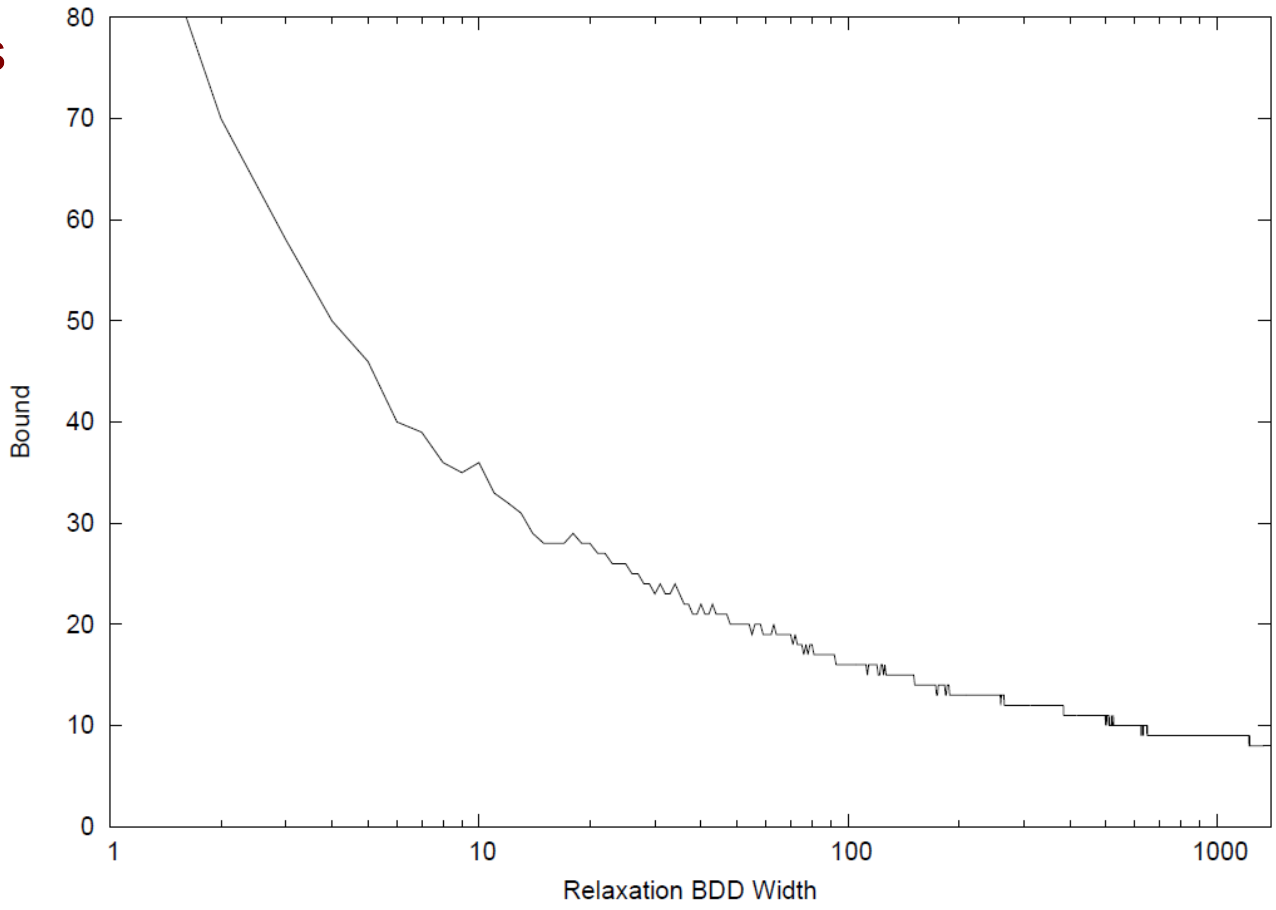
Represents 11 solutions, including 9 feasible solutions



Longest path gives upper bound on optimal value

Decision Diagrams

- Wider diagrams yield tighter bounds
  - But take longer to build.
  - Adjust width dynamically.



Decision Diagrams

- Recursive modeling
  - **Dynamic programming** model of problem, using **state variables**.
    - Using state variables.
  - Rule for **merging states** to create relaxed DD.
    - Analogous to adding valid inequalities in IP.
  - How about **curse of dimensionality**?
    - Solve by **branch and bound** in the relaxed diagram, rather than state space enumeration.

Decision Diagrams

- A novel **branch-and-bound** algorithm.
  - Branch on nodes in **last exact layer** of relaxed decision diagram.
    - …rather than branch on variables.
    - Create a new **relaxed DD rooted** at each branching node.
    - Prune search tree using bounds from relaxed DD.

Decision Diagrams

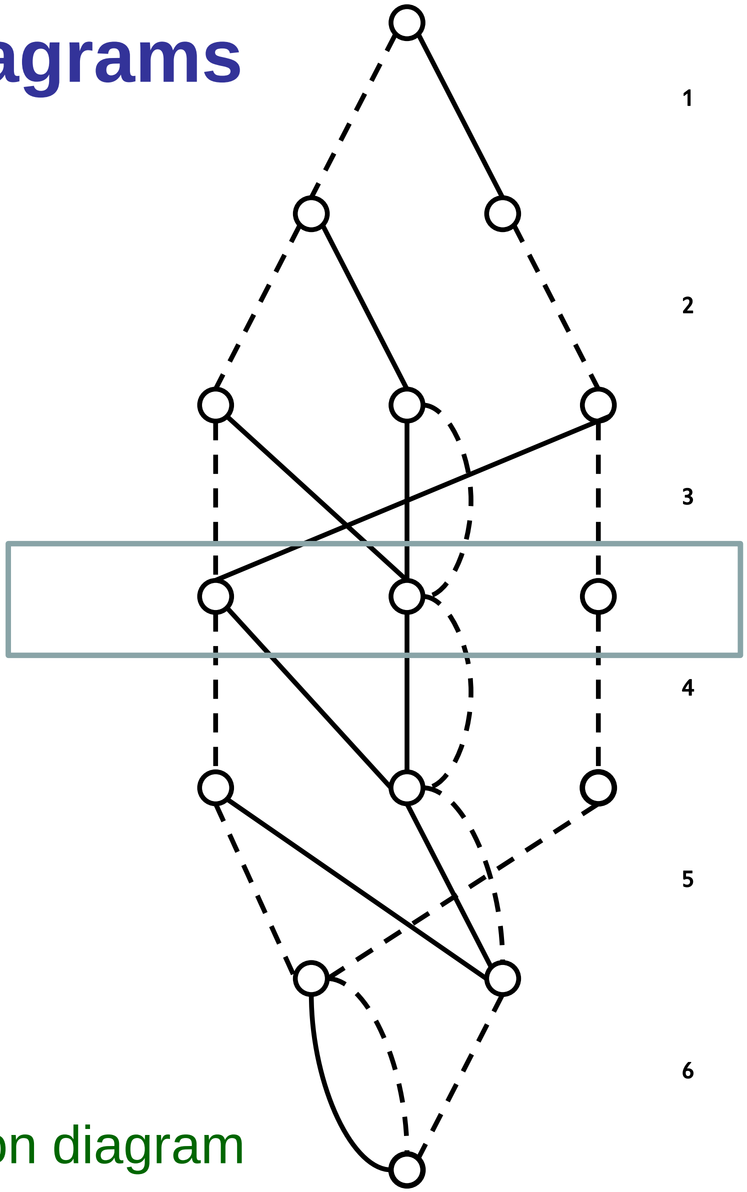
- A novel **branch-and-bound** algorithm.
  - Branch on nodes in **last exact layer** of relaxed decision diagram.
    - …rather than branch on variables.
    - Create a new **relaxed DD rooted** at each branching node.
    - Prune search tree using bounds from relaxed DD.
  - Advantage: a manageable number states may be reachable in first few layers.
    - …even if the state space is **exponential**.

Decision Diagrams

Branching in a relaxed decision diagram

Bergman, Ciré, van Hoeve, JH (2014)

Diagram is exact down to here



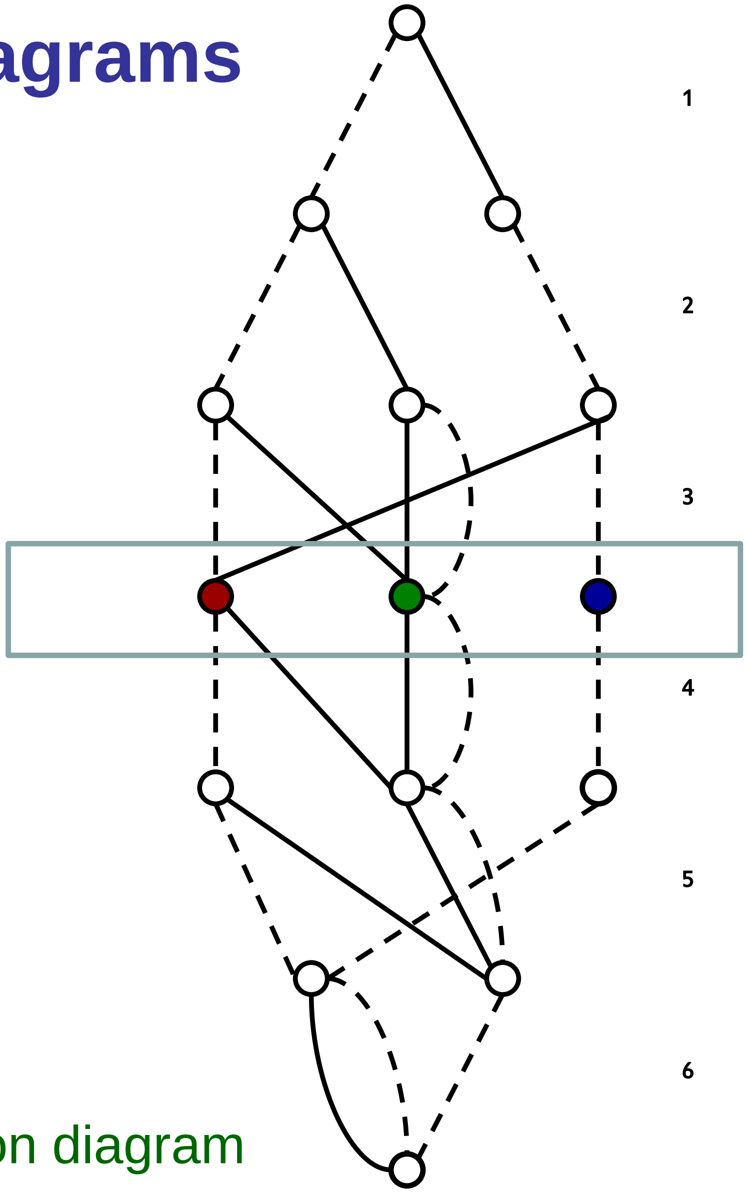
Relaxed decision diagram

Decision Diagrams

Branching in a relaxed decision diagram

Bergman, Ciré, van Hoeve, JH (2014)

Branch on nodes in this layer



Relaxed decision diagram

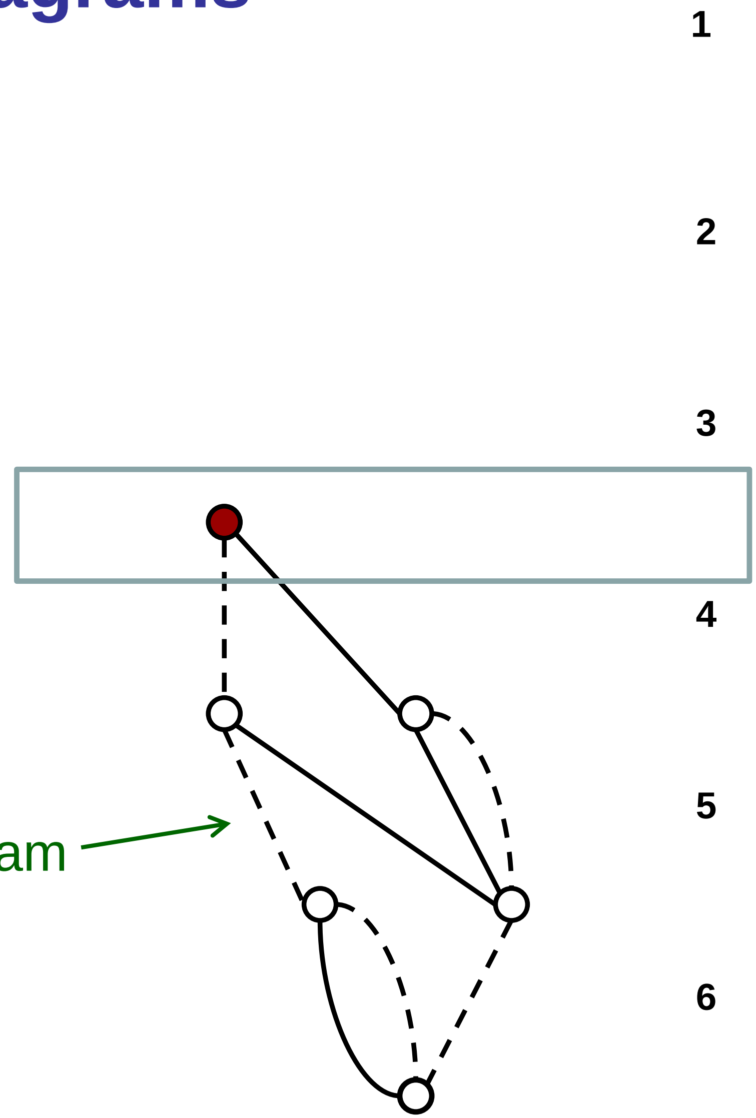
Decision Diagrams

Branching in a relaxed
decision diagram

Bergman, Ciré, van Hoeve, JH (2014)

First branch

New relaxed decision diagram

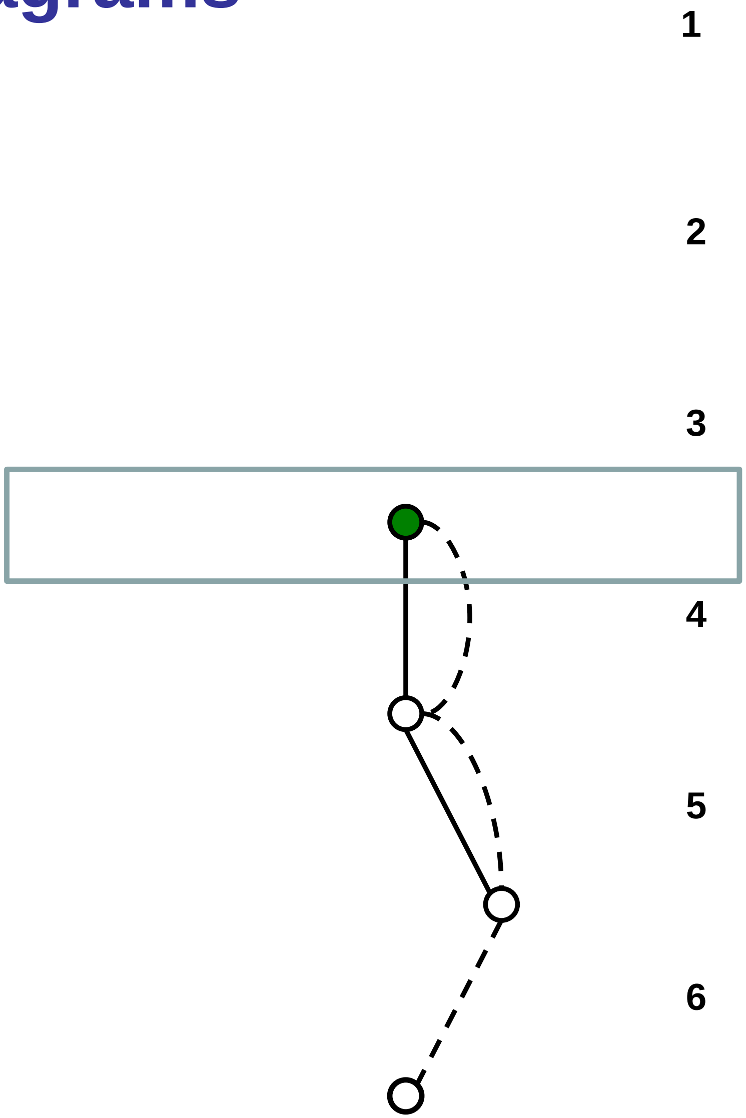


Decision Diagrams

Branching in a relaxed decision diagram

Bergman, Ciré, van Hoeve, JH (2014)

Second branch

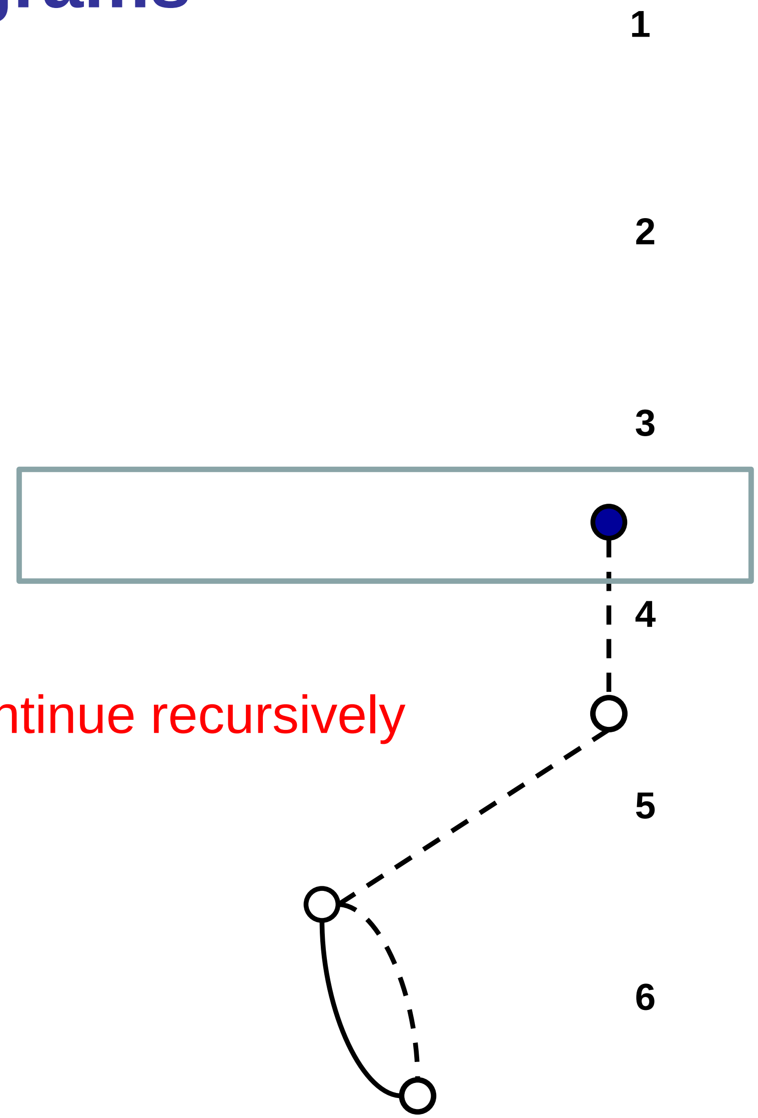


Decision Diagrams

Branching in a relaxed
decision diagram

Bergman, Ciré, van Hoeve, JH (2014)

Third branch



Decision Diagrams

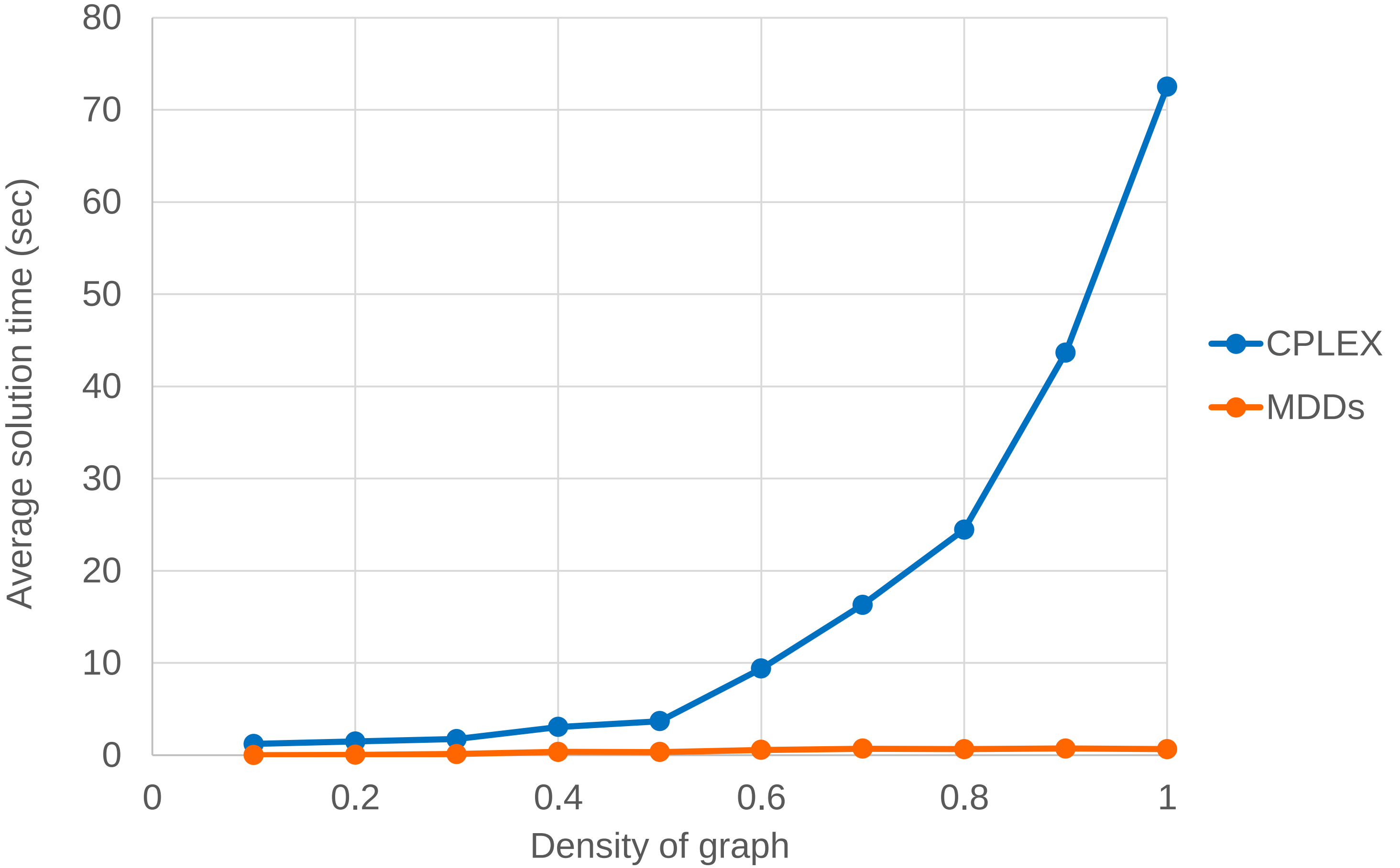
- Computational results for optimization.
  - Stable set, max cut, max 2-SAT.
    - Superior to commercial MIP solver (CPLEX) on most instances.
    - Even though the problems have **natural MIP models**.
    - Obtained best known solution on some max cut instances.
  - Slightly slower than MIP on stable set with precomputed clique cover model, but…

Decision Diagrams

Max cut
on a graph

Avg. solution time
vs
graph density

30 vertices

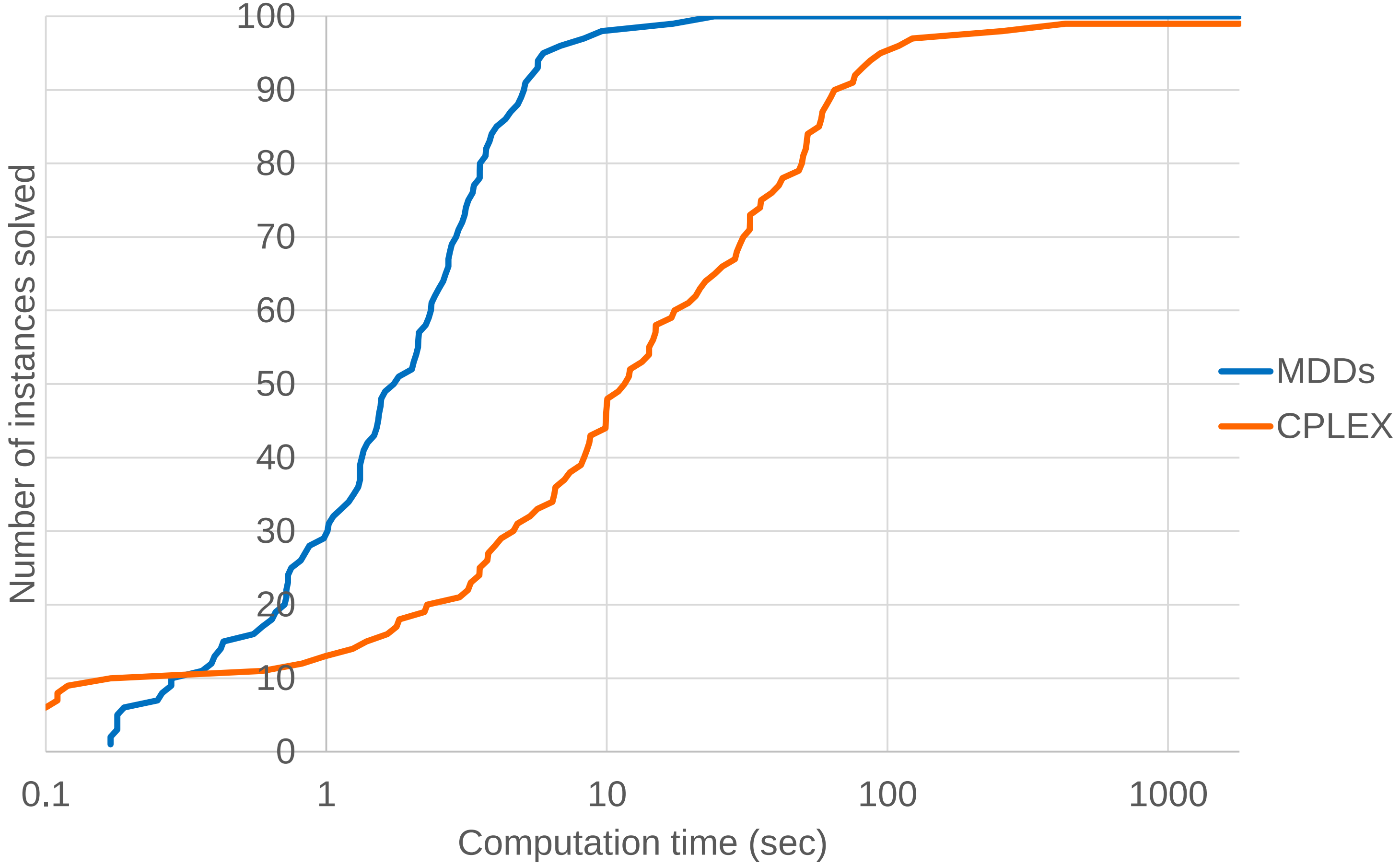


Decision Diagrams

Max 2-SAT

Performance profile

30 variables

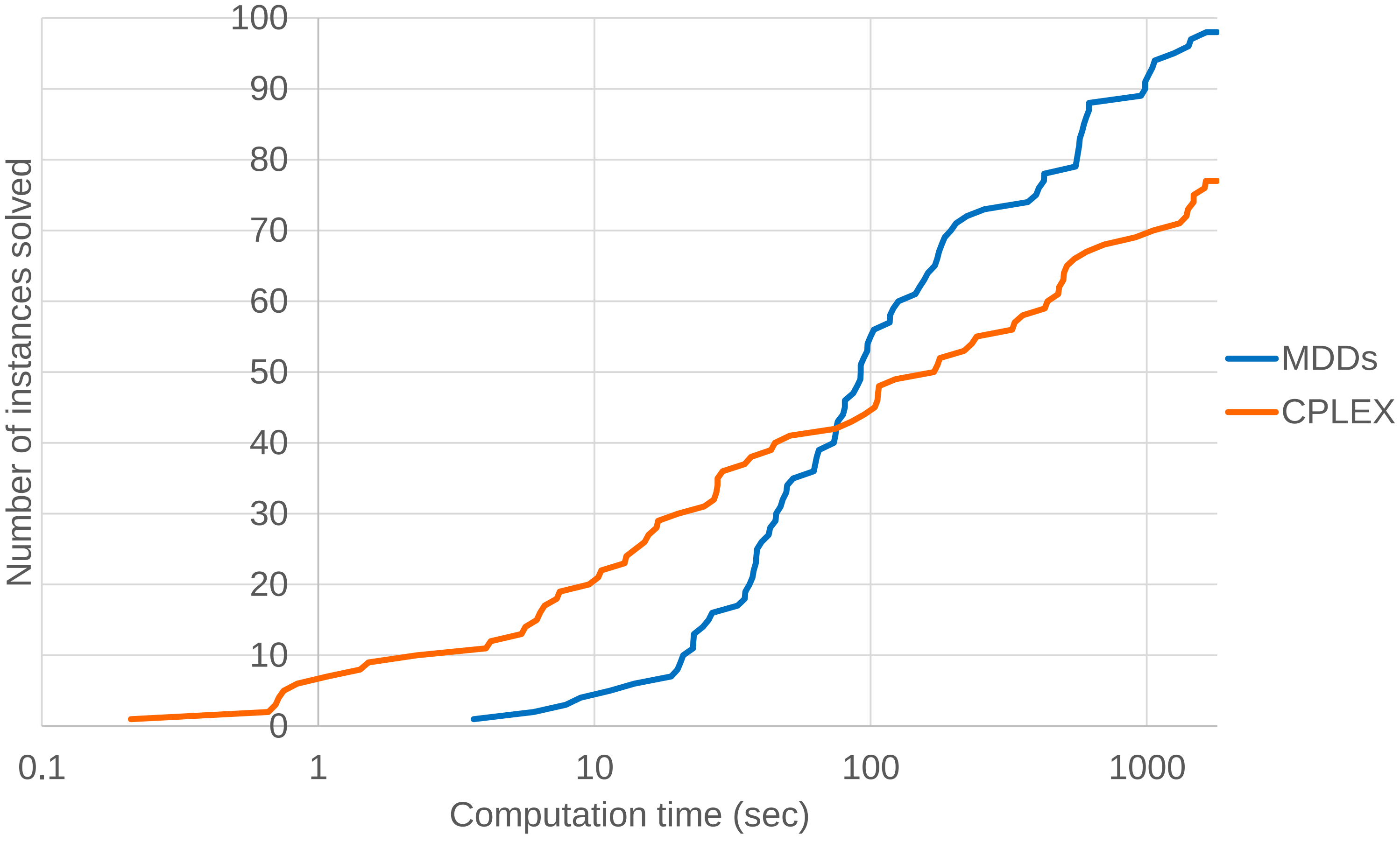


Decision Diagrams

Max 2-SAT

Performance profile

40 variables



Decision Diagrams

- Potential to scale up
  - No need to load large inequality model into solver.
  - Parallelizes very effectively
    - Near-linear speedup.
    - Much better than mixed integer programming.

Decision Diagrams

- Next steps
  - Decision diagram technology is ready for **large-scale** applications from industry
    - …which will also help develop the technology.

Decision Diagrams

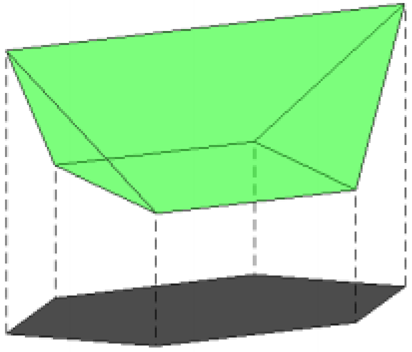
- Next steps
  - Decision diagram technology is ready for **large-scale** applications from industry
    - …which will also help develop the technology.
  - Give problems a **dynamic programming** model.
    - …natural for many applications.
    - But solve by **branch and bound**, not DP.
  - Extend to **stochastic** DP and optimal control.
    - Current research.

Logic-Based Benders

- **Logic-based Benders decomposition** is a generalization of classical Benders decomposition.

JH (1995) | JH and Yan (1995) | JH and Ottosson (2003)

- Solves a **projection** problem.
  - ...and therefore a **logical inference** problem.



Logic-Based Benders

- Collaborators…
  - André Ciré
  - Elvin Çoban
  - Aliza Heching
  - Greger Ottosson
  - Erlendur Thorsteinsson*
  - Hong Yan

***2001** *Best Paper Award*, CP conference

Logic-Based Benders

- Subproblem is an **arbitrary optimization** problem.
  - In classical Benders, subproblem must be linear or nonlinear programming problem.
    - LBBD replaces LP dual with **inference dual**.
  - Solves problems of the form

$$\begin{aligned}&\min f(x,y)\\&(x,y)\in S\\&x\in D\end{aligned}$$

Logic-Based Benders

- Decompose problem into master and subproblem.
  - Subproblem is obtained by fixing x to solution value in master problem.

Master problem

$$\min z$$
$$z \geq g_k(x) \quad \text{(Benders cuts)}$$
$$x \in D$$

Find x that minimizes cost z subject to Benders cuts obtained from solutions in previous iterations k.

Trial value $\bar{x}$ that solves master

Benders cut $z \geq g_k(x)$

Subproblem

$$\min f(\bar{x}, y)$$
$$(\bar{x}, y) \in S$$

Obtain proof of optimality (**inference dual**).
Use same proof to deduce cost bounds for other assignments, yielding **Benders cut**.

Logic-Based Benders

- Iterate until master problem value equals best subproblem value so far.
  - This yields optimal solution.

Master problem

$$\min z$$
$$z \ge g_k(x) \quad \text{(Benders cuts)}$$
$$x \in D$$

Find x that minimizes cost z subject to Benders cuts obtained from solutions in previous iterations k.

Trial value $\bar{x}$ that solves master

Benders cut $z \ge g_k(x)$

Subproblem

$$\min f(\bar{x}, y)$$
$$(\bar{x}, y) \in S$$

Obtain proof of optimality (**inference dual**).
Use same proof to deduce cost bounds for other assignments, yielding **Benders cut**.

Logic-Based Benders

- Substantial speedup for many applications.
  - Several orders of magnitude relative to state of the art.

Logic-Based Benders

- Substantial speedup for many applications.
  - Several orders of magnitude relative to state of the art.
- Some applications:
  - Circuit verification
  - Chemical batch processing (BASF, etc.)
  - Steel production scheduling
  - Auto assembly line management (Peugeot-Citroën)
  - Automated guided vehicles in flexible manufacturing
  - Allocation and scheduling of multicore processors (IBM, Toshiba, Sony)
  - Facility location-allocation
  - Stochastic facility location and fleet management
  - Capacity and distance-constrained plant location

Logic-Based Benders

- Some applications…
  - Transportation network design
  - Traffic diversion around blocked routes
  - Worker assignment in a queuing environment
  - Single- and multiple-machine allocation and scheduling
  - Permutation flow shop scheduling with time lags
  - Resource-constrained scheduling
  - Wireless local area network design
  - Service restoration in a network
  - Optimal control of dynamical systems
  - Sports scheduling

Logic-Based Benders

- Application to planning and scheduling.
  - Assign tasks in master, schedule them in subproblem.
    - Combine **mixed integer programming (MIP)** and **CP**

Master problem

Assign tasks to resources to minimize cost.

Solve by **MIP**.

Trial assignment $\bar{x}$

Benders cut $z \geq g_k(x)$

Subproblem

Schedule tasks on each resource, subject to time windows.

CP obtains proof of optimality (inference dual).

Use **same proof** to deduce cost for other assignments, yielding **Benders cut**.

Logic-Based Benders

- Objective function
  - Cost is based on **task assignment only**.

$$\text{cost } = \sum_{ij} c_{ij} x_{ij}, \quad x_{ij} = 1 \text{ if task } j \text{ assigned to resource } i$$

  - So cost appears only in the **master problem**.
  - Scheduling subproblem is a **feasibility problem**.

Logic-Based Benders

- Objective function
  - Cost is based on **task assignment only**.

    $$\text{cost} = \sum_{ij} c_{ij} x_{ij}, \quad x_{ij} = 1 \text{ if task } j \text{ assigned to resource } i$$

    - So cost appears only in the **master problem**.
    - Scheduling subproblem is a **feasibility problem**.

- Benders cuts
  - They have the form $\sum_{j \in J_i} (1 - x_{ij}) \geq 1, \text{ all } i$
    - where J_i is a set of tasks that create infeasibility when assigned to resource i.

Logic-Based Benders

- Resulting Benders decomposition:

Master problem

$$\min\ z$$
$$z = \sum_{ij} c_{ij} x_{ij}$$
Benders cuts

Trial assignment $\bar{x}$

Benders cuts

$$\sum_{j \in J_i} (1 - x_{ij}) \geq 1,$$

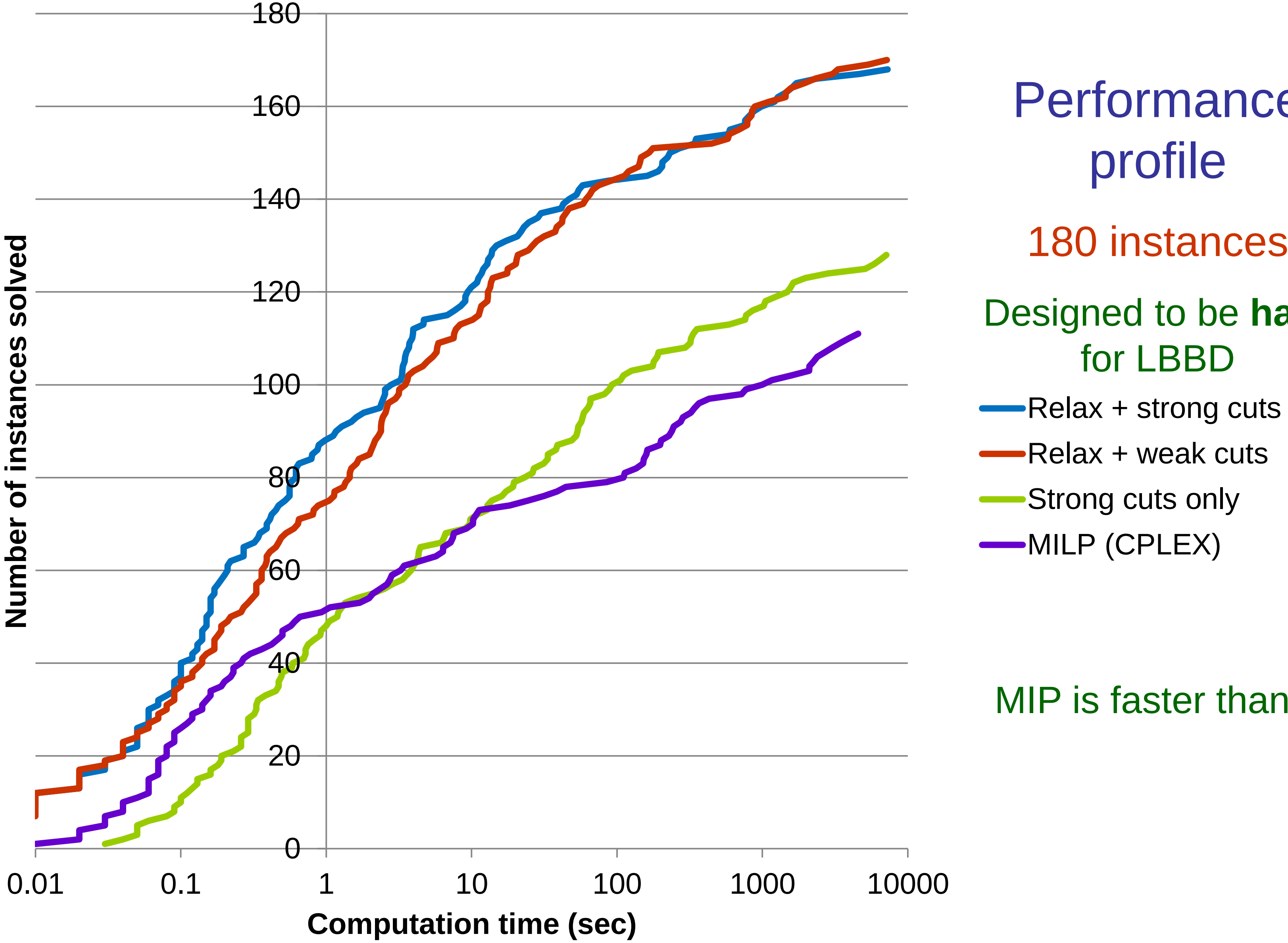
for infeasible resources *i*

Subproblem

Schedule jobs on each resource.

CP may obtain proof of infeasibility on some resources (inference dual).

Use **same proof** to deduce infeasibility for some other assignments, yielding **Benders cut**.



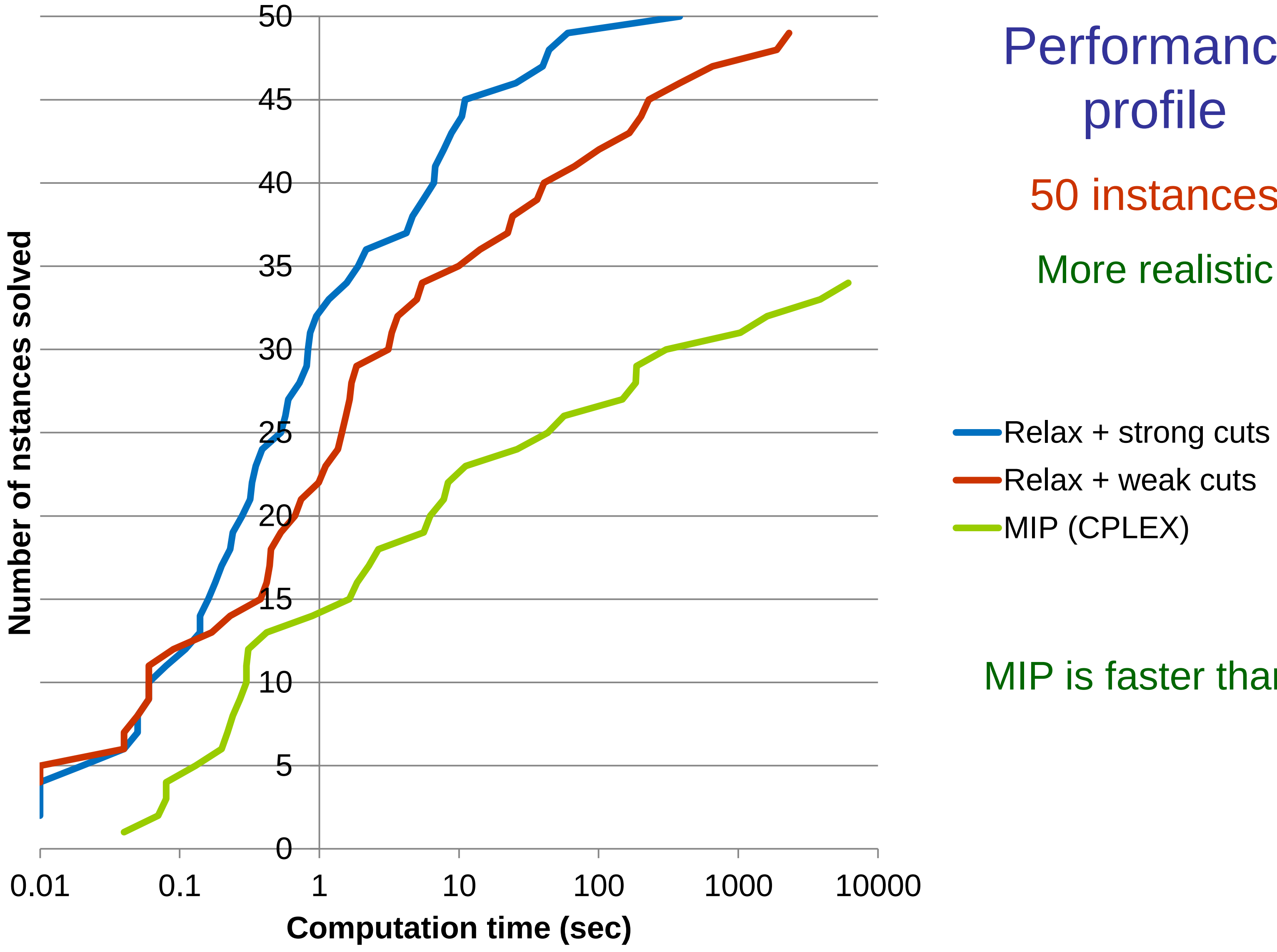
Performance profile

180 instances

Designed to be **hard** for LBBD

MIP is faster than CP

Ciré, Çoban, JH (2015)



Performance profile

50 instances

More realistic

MIP is faster than CP

Ciré, Çoban, JH (2015)

Logic-Based Benders

- Current research
  - Apply to **robust** scheduling.
  - Use a **decision diagram** in the master problem.
  - Apply to **logical inference** from large datasets.

Logic-Based Benders

- Robust scheduling
  - …with uncertainty sets.
    - Uncertainty subproblem becomes Benders subproblem.
  - IT service center scheduling.
    - Projects subject to uncertain delays
    - Give customers a reasonable worst-case completion date.

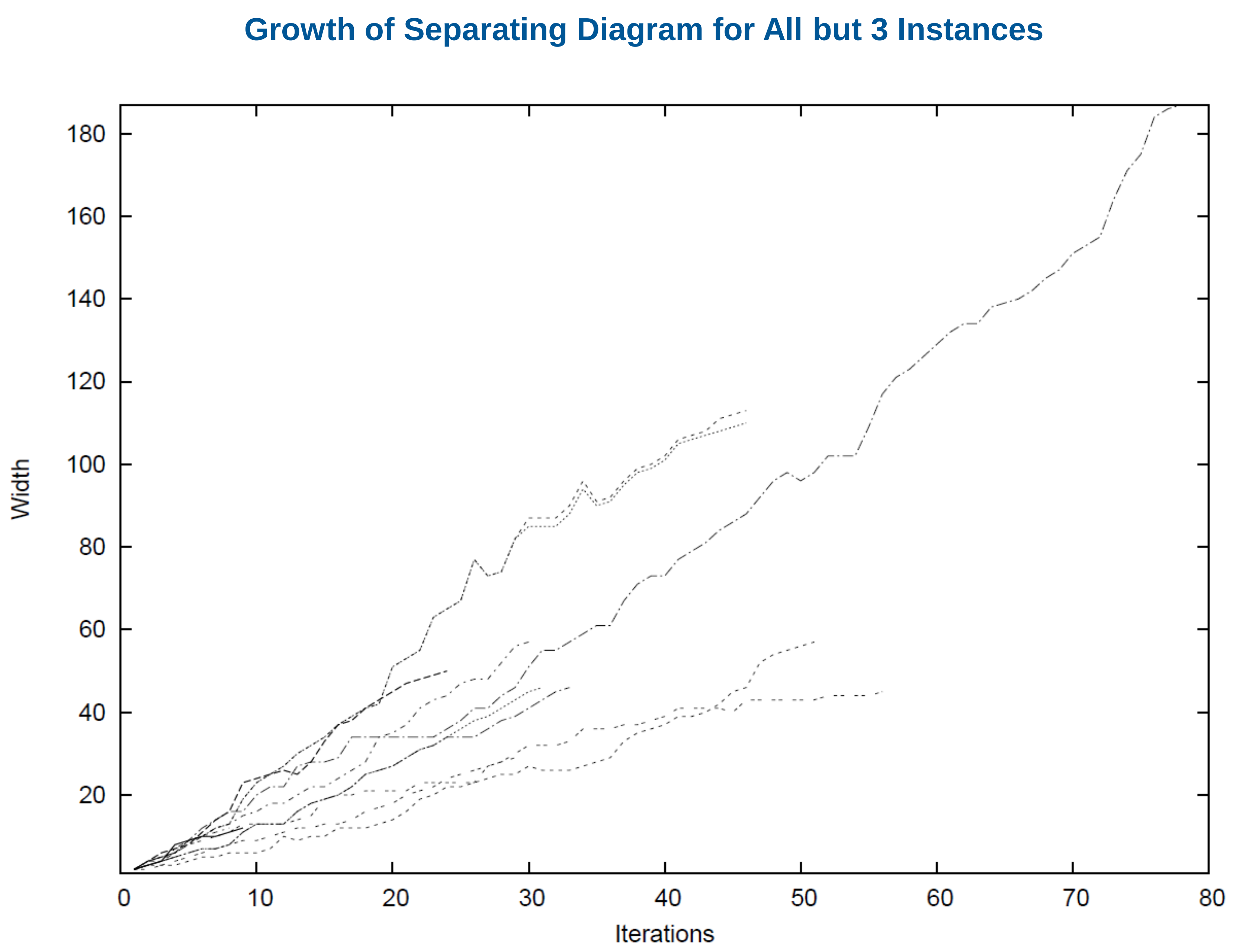
Çoban, Heching, JH (2015)

Logic-Based Benders

- **Decision diagram** in master problem.
  - Add Benders cuts by **modifying** the decision diagram.
    - Requires **separation algorithm** for decision diagrams
  - Home healthcare scheduling.
    - Master problem assigns healthcare aides to patients.
    - Subproblem schedules visits and routes the aides.
  - Key question:
    - How much will the decision diagram **grow** as Benders cuts are added?

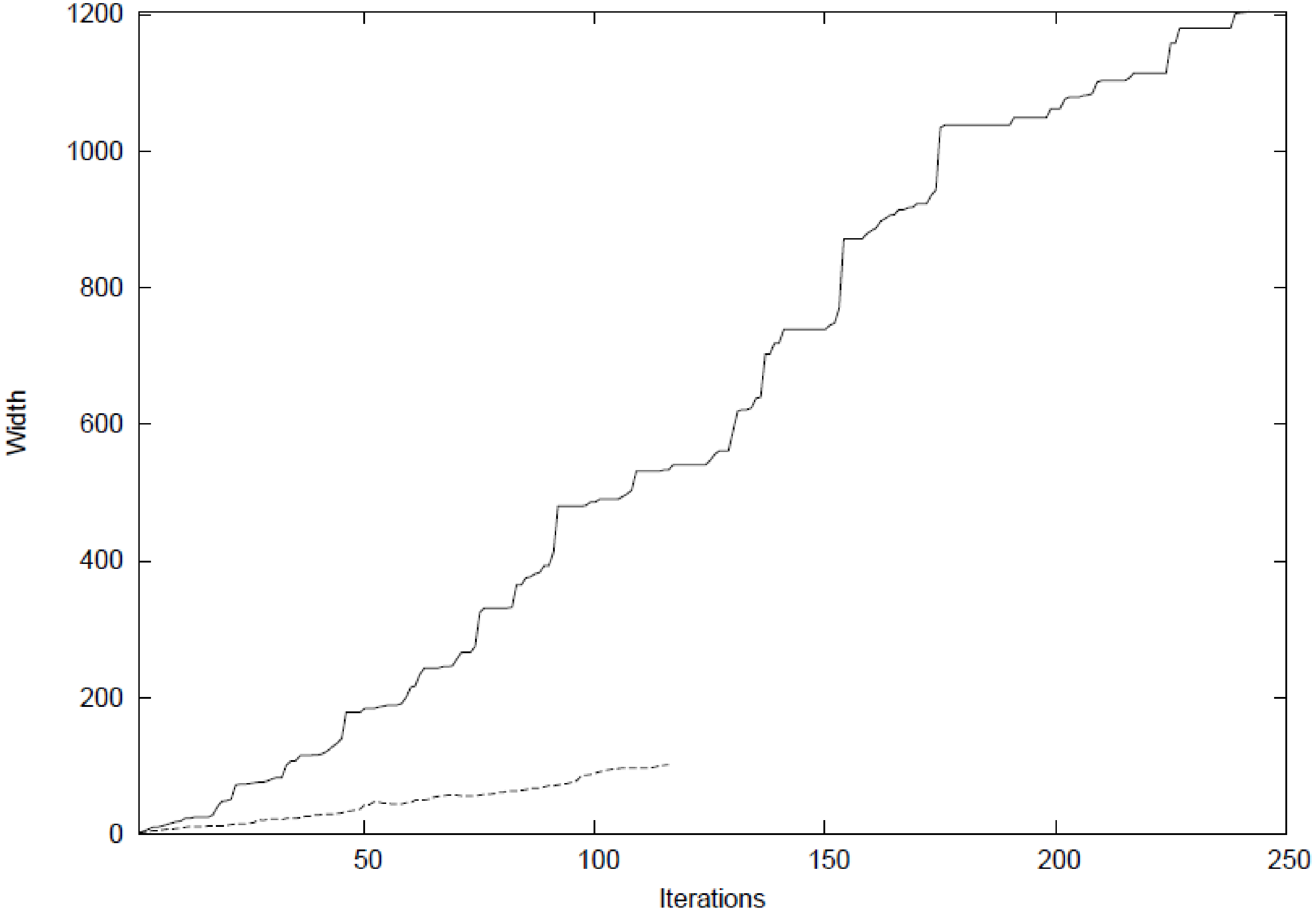
Ciré, JH (2014)

Heching, JH (2015)

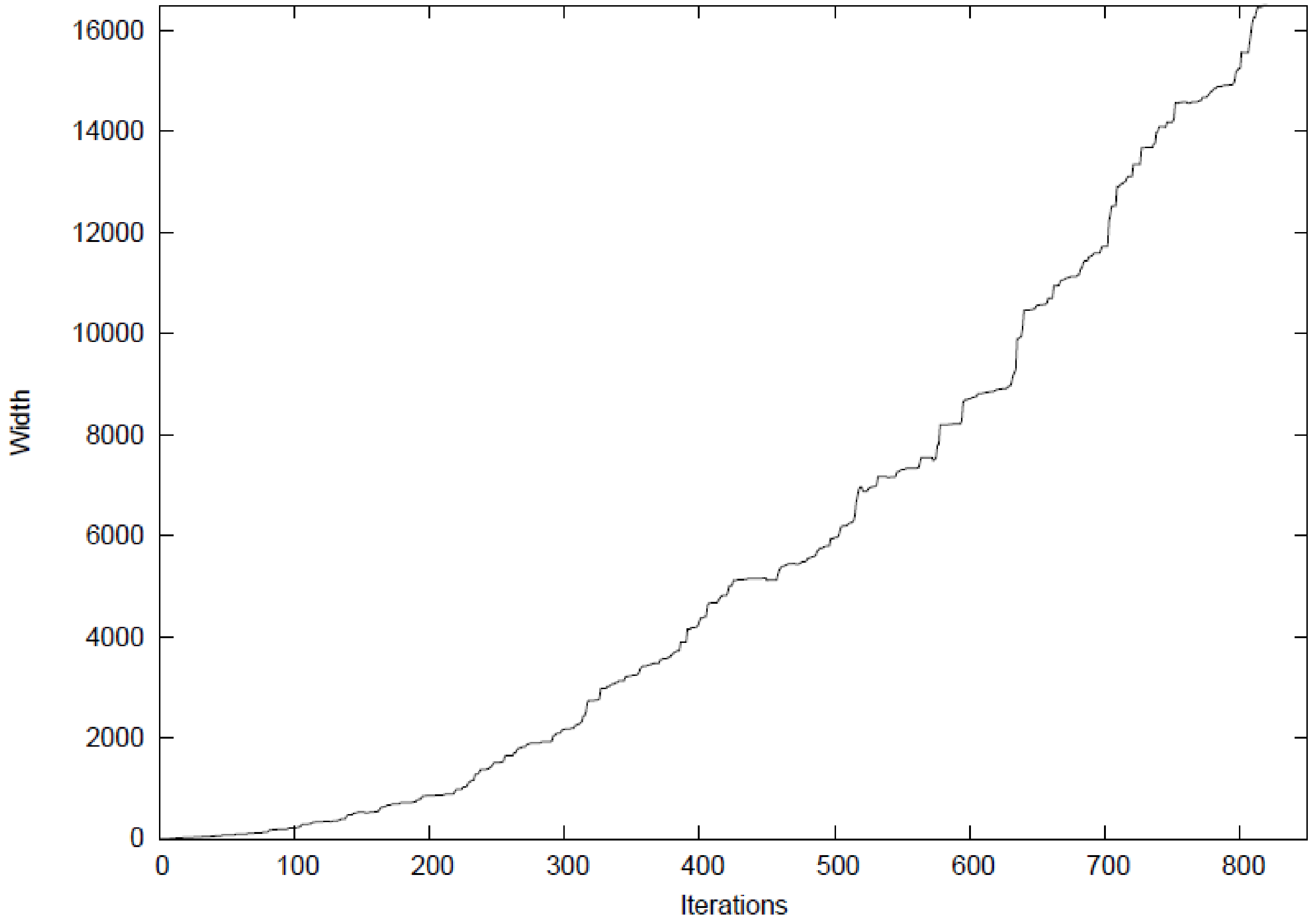


Growth of Separating Diagram for All but 3 Instances
Width
Iterations
0
10
20
30
40
50
60
70
80
20
40
60
80
100
120
140
160
180

Growth of Separating Diagram for 2 Harder Instances

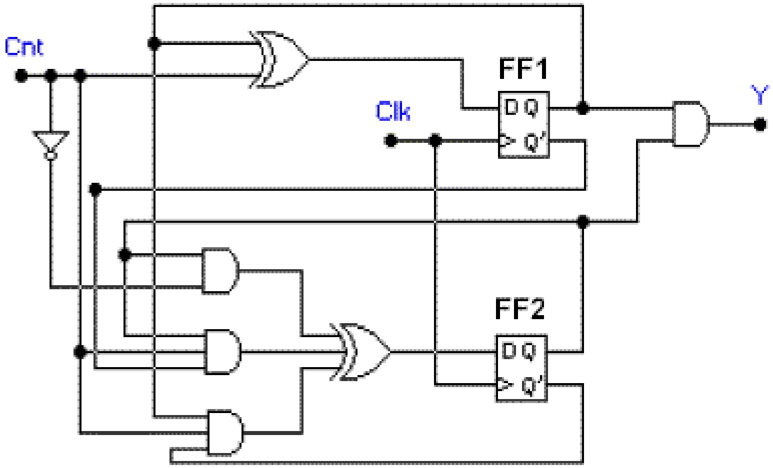


Growth of Separating Diagram for Hardest Instance



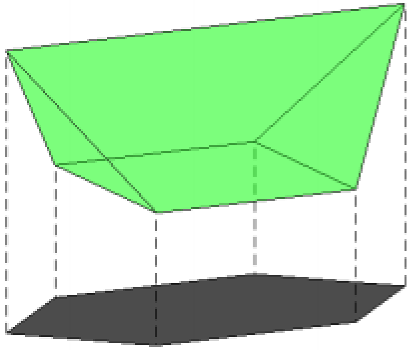
Logic-Based Benders

- Logical inference.
  - Benders cuts describe the **projection** of the feasible set onto the master problem
  - This is the fundamental **logical inference** problem.
    - Original application was to logic circuit verification.



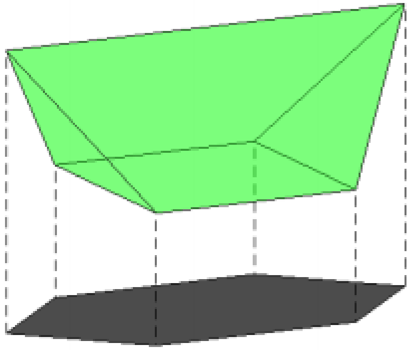
Logical Inference

- Connection between optimization and logic.
  - Both are **projection** problems.
  - Projection = extract information involving a subset of variables.



Logical Inference

- Connection between optimization and logic.
  - Both are **projection** problems.
  - Projection = extract information involving a subset of variables.
- Optimization…
  - **Project** feasible set onto a single variable representing the objective function.
- Inference…
  - **Project** knowledge base onto a subset of propositional variables.



Logical Inference

- Example:
  - Knowledge base consists of logical clauses.
  - Derive all implications involving x_1, x_2.
  - This is a projection problem.

$$x_1 \vee x_2$$
$$\neg x_1 \vee x_3$$
$$\neg x_1 \vee \neg x_2 \vee \neg x_3$$
$$x_1 \vee x_3 \vee x_4$$
$$x_2 \vee x_3 \vee \neg x_4$$

$$x_1 \vee x_2$$
$$\neg x_1 \vee \neg x_2$$

Projection

Logical Inference

- Example:
  - Knowledge base consists of logical clauses.
  - Derive all implications involving x_1, x_2.
  - This is a projection problem.
- Solved by logic-based Benders.
  - Clauses in the projection are Benders cuts
    - They are also **conflict clauses** containing x_1, x_2 from a SAT algorithm.

$$\boxed{\begin{array}{l} x_1 \vee x_2 \\ \neg x_1 \vee x_3 \\ \neg x_1 \vee \neg x_2 \vee \neg x_3 \\ x_1 \vee x_3 \vee x_4 \\ x_2 \vee x_3 \vee \neg x_4 \end{array}}$$

$$\boxed{\begin{array}{l} x_1 \vee x_2 \\ \neg x_1 \vee \neg x_2 \end{array}}$$

Projection

Logical Inference

- A central problem today:
  - Associate each inference with its **probability**, **relevance**, or **confidence**.
    - For example, IBM's **Watson** (Jeopardy player).

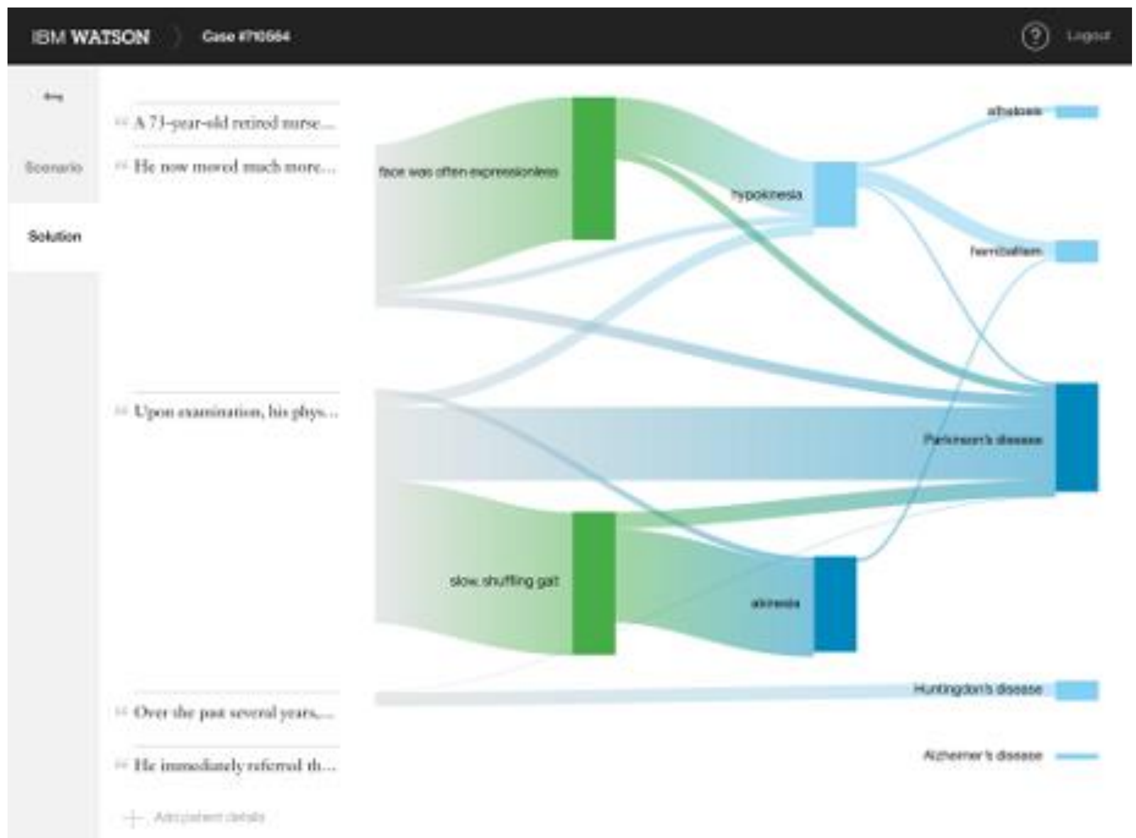
Logical Inference

Watson technology first applied to medicine (WatsonPaths).

Draws inferences from medical literature and clinical guidelines.

About 1 million articles listed per year in PubMed.

Probably 1.5-2 million overall.

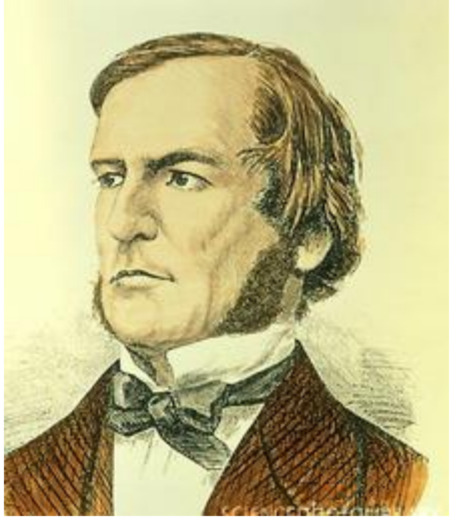


Logical Inference

- The problem was first posed for **probability**...

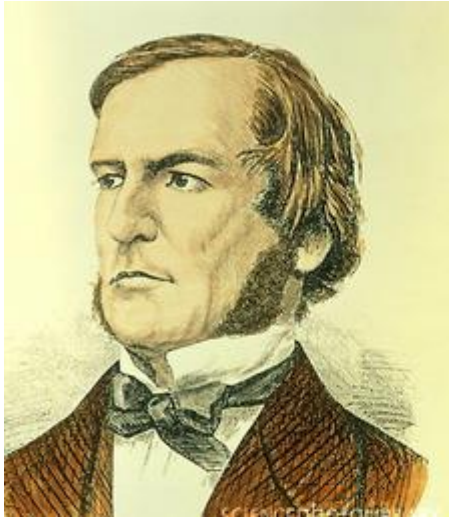
Logical Inference

- The problem was first posed for **probability**...
  - …by George Boole.
    - He viewed **probability logic** as a key contribution.



Logical Inference

- The problem was first posed for **probability**...
  - …by George Boole.
    - He viewed **probability logic** as his most important contribution.
    - Formulated inference problem as **linear programming**.
    - Can be solved by modern **column generation** methods.



Logical Inference

Example

Clause	Probability
x_1	0.9
$\bar{x}_1 \vee x_2$	0.8
$\bar{x}_2 \vee x_3$	0.4

Deduce probability range for x_3

Logical Inference

Example

Clause	Probability
x_1	0.9
$\bar{x}_1 \vee x_2$	0.8
$\bar{x}_2 \vee x_3$	0.4

Deduce probability range for x_3

Linear programming model

$$\min/\max \ \pi_0$$

$$\begin{bmatrix} 01010101 \\ 00001111 \\ 11110011 \\ 11011101 \\ 11111111 \end{bmatrix} \begin{bmatrix} p_{000} \\ p_{001} \\ p_{010} \\ \vdots \\ p_{111} \end{bmatrix} = \begin{bmatrix} \pi_0 \\ 0.9 \\ 0.8 \\ 0.4 \\ 1 \end{bmatrix}$$

p_{000} = probability that $(x_1,x_2,x_3) = (0,0,0)$

Logical Inference

Example

Clause	Probability
x_1	0.9
$\bar{x}_1 \vee x_2$	0.8
$\bar{x}_2 \vee x_3$	0.4

Deduce probability range for x_3

Linear programming model

$$\min/\max \ \pi_0$$

$$\begin{bmatrix} 01010101 \\ 00001111 \\ 11110011 \\ 11011101 \\ 11111111 \end{bmatrix} \begin{bmatrix} p_{000} \\ p_{001} \\ p_{010} \\ \vdots \\ p_{111} \end{bmatrix} = \begin{bmatrix} \pi_0 \\ 0.9 \\ 0.8 \\ 0.4 \\ 1 \end{bmatrix}$$

p_{000} = probability that $(x_1,x_2,x_3) = (0,0,0)$

Solution: $\pi_0 \in [0.1,0.4]$

Logical Inference

- Next steps
  - Apply **large-scale optimization** methods to logical inference.
    - They are substantially **underutilized** for this purpose.
  - In particular:
    - Use **logic-based Benders** for propositional logic, etc.
    - Use **column generation** for probability logic.
    - Use **linear programming** for belief logics, relevance, and variations of Dempster-Shafer theory.
    - Use **nonlinear programming** for Bayesian networks.
    - Use **decision diagrams** for queries and what-if analysis.

Andersen, JH (1992,1994,1996)

Chandru, JH (1999)