Alternative Methods for Obtaining Optimization Bounds

J. N. Hooker

Carnegie Mellon University

AFOSR Program Review, April 2012
Grant FA9550-11-1-0180

Integrating OR and CP/AI

- Early support by AFOSR
  - First conference (1995)
  - Now an annual conference (CPAIOR)

Integrating OR and CP/AI

- Early support by AFOSR
  - First conference (1995)
  - Now an annual conference (CPAIOR)
- Today…
  - Growing literature (4 books, many papers)
  - Moving into optimization software
    - OPL Studio, SIMPL, SCIP (constraint integer programming), Eclipse, Mosel, BARON, G12
  - Regular sessions at major conferences
    - INFORMS, ISMP, INFORMS Computing Society, INFORMS Optimization Society

Some Current Projects

- Bounds from finite domain cuts (OR + CP)*
  - Joint work with David Bergman
- Bounds from binary decision diagrams (OR + CS)*
  - Joint work with David Bergman, Andre Cire, & Willem van Hoeve
- BDD-based branching methods (OR + CS)
  - Joint work with D. Bergman, A. Cire, W. van Hoeve, & T. Yunes
- Semantic typing for optimization models (OR + AI)
  - Joint work with Andre Cire & Tallys Yunes

*Presented today

Bounds from Finite-Domain Cuts

Joint work with David Bergman

Finite Domain Formulations

- **0-1** variables often encode choices that can be represented with **finite domain** variables.
  - x_i = finite domain variable
    - Job assigned to worker *i*
    - Start time of job *i*
    - City visited after city *i*
  - y_{ij} = corresponding 0-1 variable
    - $y_{ij} = 1$ if $x_i = j$

Finite Domain Cuts

- **Finite-domain** variables are common in **constraint programming** formulations.
  - If the variables are numeric, the problem has **polyhedral structure**.
  - **Finite-domain cuts** can be mapped into the 0-1 model.
  - This may yield **new and stronger cuts** in the 0-1 model.

Finite Domain Cuts

- **Finite-domain** variables are common in **constraint programming** formulations.
  - If the variables are numeric, the problem has **polyhedral structure**.
  - **Finite-domain cuts** can be mapped into the 0-1 model.
  - This may yield **new and stronger cuts** in the 0-1 model.

- We apply this idea to **graph coloring**.
  - Has a natural CP formulation.

Motivation

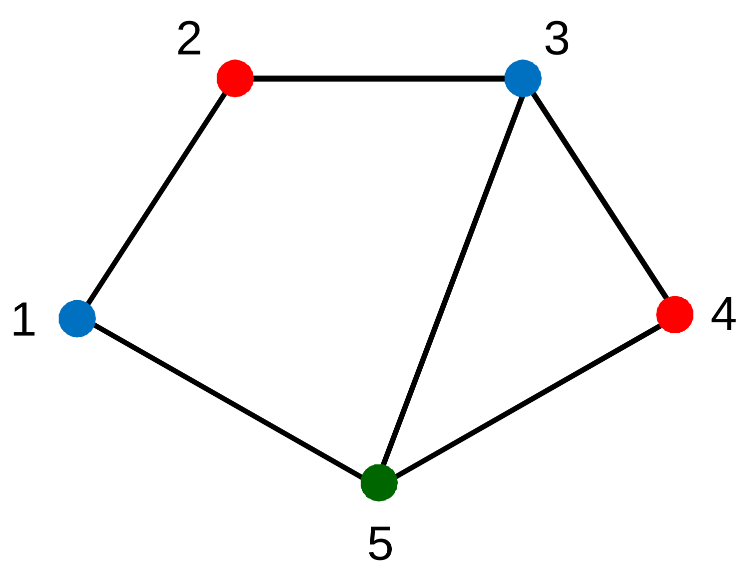
- We obtain two kinds of results:
  - If you find a structure (e.g., odd hole) that yields a known valid inequality in 0-1 space…
    - We will give you a stronger cut for **free**.
    - Use whatever separation algorithm you want.

Motivation

- We obtain two kinds of results:
  - If you find a structure (e.g., odd hole) that yields a known valid inequality in 0-1 space…
    - We will give you a stronger cut for **free**.
    - Use whatever separation algorithm you want.
  - We identify **additional** structures that yield valid inequalities.
    - They are much **stronger** than **known cuts**.
    - Many **fewer** are required.
    - We have separation algorithms (if needed)

Graph Coloring

- We focus on the **vertex coloring** problem.
  - Given a graph, assign colors to vertices so that no two adjacent vertices receive the same color.
  - Minimize the number of colors.



Graph Coloring

- 0-1 model

$$\min \sum_j w_j$$

= 1 if color j is used

$$\sum_j y_{ij} = 1, \quad \text{all vertices } i$$

$$y_{1j} + y_{2j} \le w_j, \quad \text{all colors } j$$

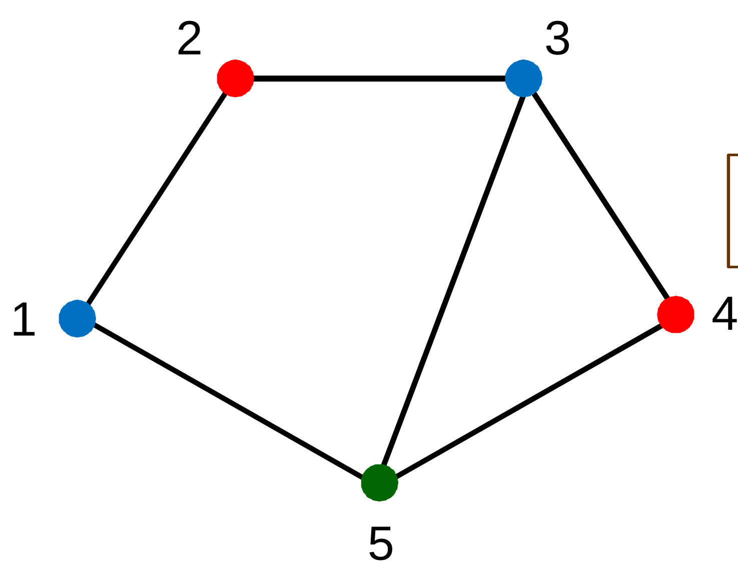
$$y_{1j} + y_{5j} \le w_j, \quad \text{all colors } j$$

$$y_{2j} + y_{3j} \le w_j, \quad \text{all colors } j$$

$$y_{3j} + y_{4j} + y_{5j} \le w_j, \quad \text{all colors } j$$

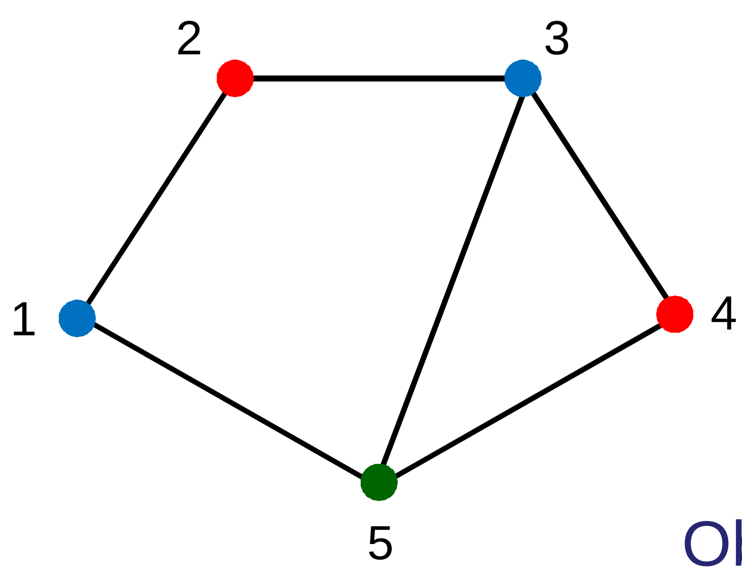
$$y_{ij} \in \{0,1\}$$

= 1 if vertex i receives color j



Alldiff Systems

- Use an **all-different** constraint for each clique.



$$
\begin{aligned}
&\min\ z \\
&z \geq x_i, \quad \text{all vertices } i \\
&\text{alldiff}(x_1, x_2), \quad \text{all colors } j \\
&\text{alldiff}(x_1, x_5), \quad \text{all colors } j \\
&\text{alldiff}(x_2, x_3), \quad \text{all colors } j \\
&\text{alldiff}(x_3, x_4, x_5), \quad \text{all colors } j \\
&\boxed{x_i} \in \{0, \ldots, 4\}
\end{aligned}
$$

x_i = color assigned to vertex i

Objective reduces **symmetry**

Alldiff Systems

- Applications:
  - Scheduling, timetabling.
    - Employee scheduling.
    - Course timetabling.
  - Latin squares.
    - Alldiff for each row, column.
    - Experimental design: orthogonal Latin squares.
    - Sudoku puzzles.
  - Graph coloring.
    - Many applications.

Related Work

- Convex hull of single alldiff.
  - Hooker (2000), Williams and Yan (2001).
- Convex hull of 2 alldiffs.
  - Appa, Magos and Mourtos (2004)
- Convex hull of alldiff systems with inclusion property.
  - Appa, Magos and Mourtos (2011).
  - Same facets as individual alldiffs.
- Some facets of systems without inclusion property.
  - Magos and Mourtos (2011).

Variable Mapping

- To write finite domain cuts in terms of 0-1 variables y_{ij}:
  - Substitute $x_i = \sum_j j y_{ij}$

Variable Mapping

- To write finite domain cuts in terms of 0-1 variables y_{ij}:
  - Substitute $x_i = \sum_j j y_{ij}$

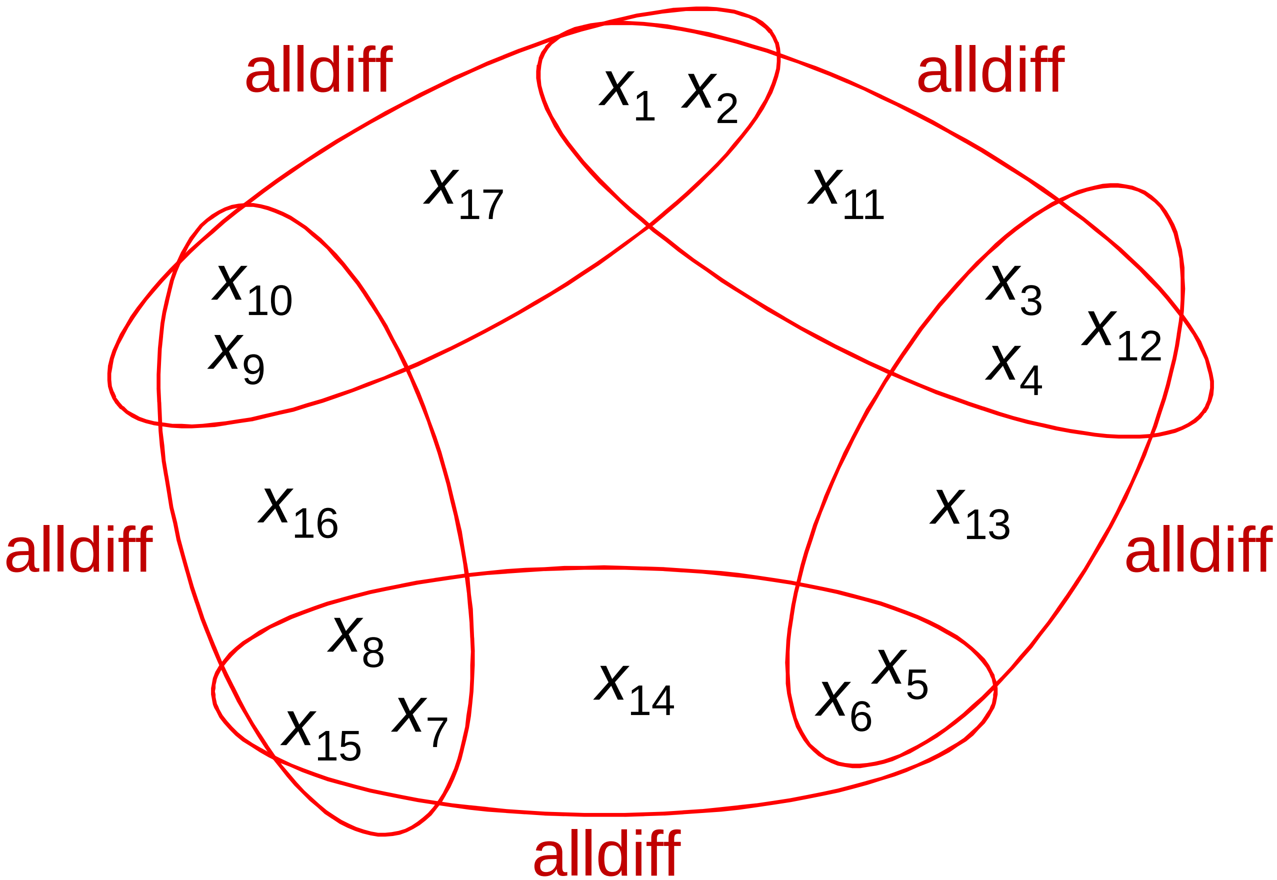
- In general, facet-defining finite-domain cuts don’t map to facet-defining 0-1 cuts.
  - They can nonetheless be more effective than known cuts.

Choice of Domain

- We will assume each x_i has domain $\{0, \ldots, n - 1\}$.
  - To simplify exposition.
- Most results can be generalized to an arbitrary numeric domain $\{v_0, \ldots, v_{n-1}\}$ with each $v_i \geq 0$.
  - Some results are valid for domain $D = \{0, \delta, \ldots, (n - 1)\delta\}$ with $\delta > 0$.

Odd Cycles

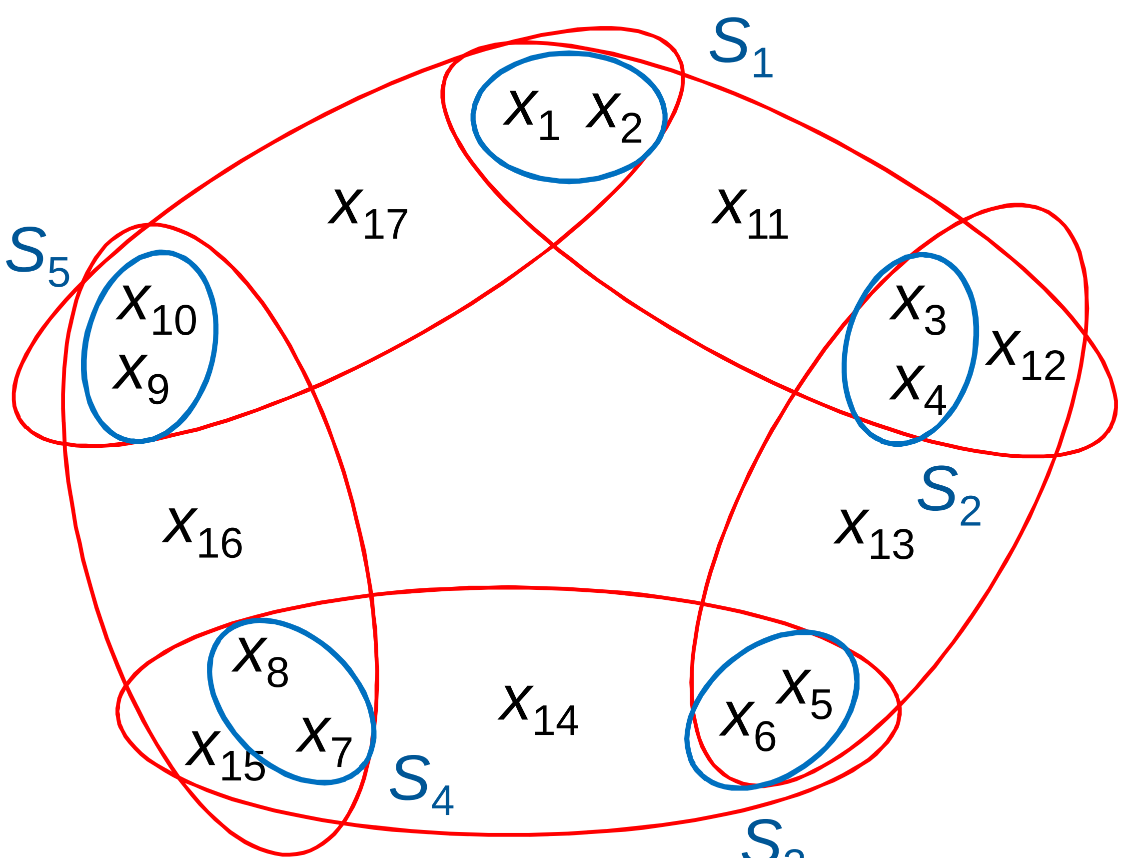
- A *q*-cycle consists of *q* alldiff constraints that look like this:



Odd Cycles

- Select any subset of s vertices in each overlap:

$s = 2$

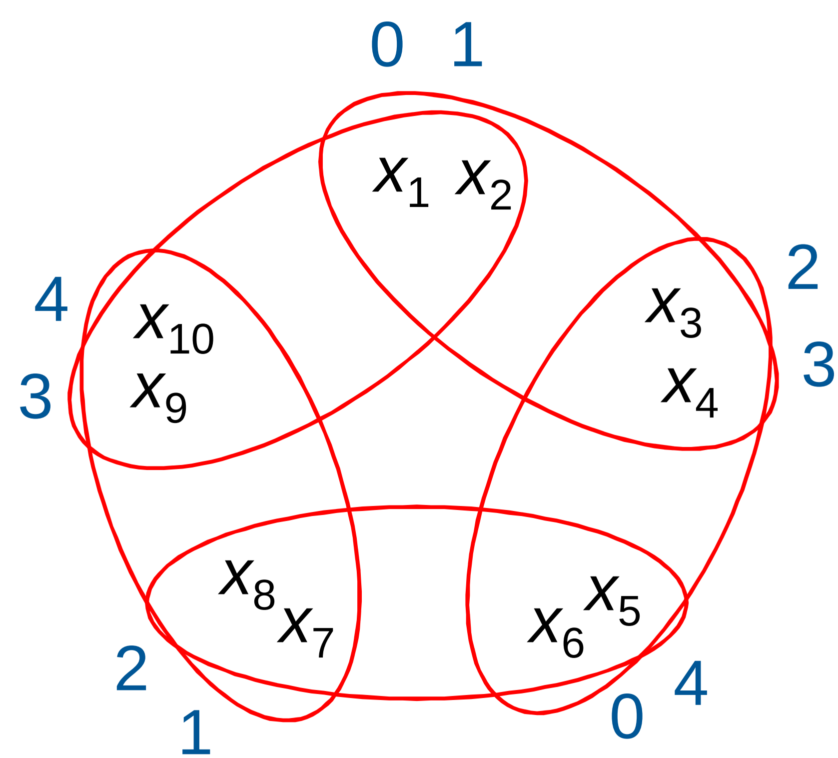


Odd Cycles

- We get a valid ***x*-cut**:

$$\sum_{i \in S} x_i \geq \left(sq - \frac{q-1}{4} L \right)(L-1) = 20$$

where $L = \left\lceil \frac{sq}{(q-1)/2} \right\rceil$



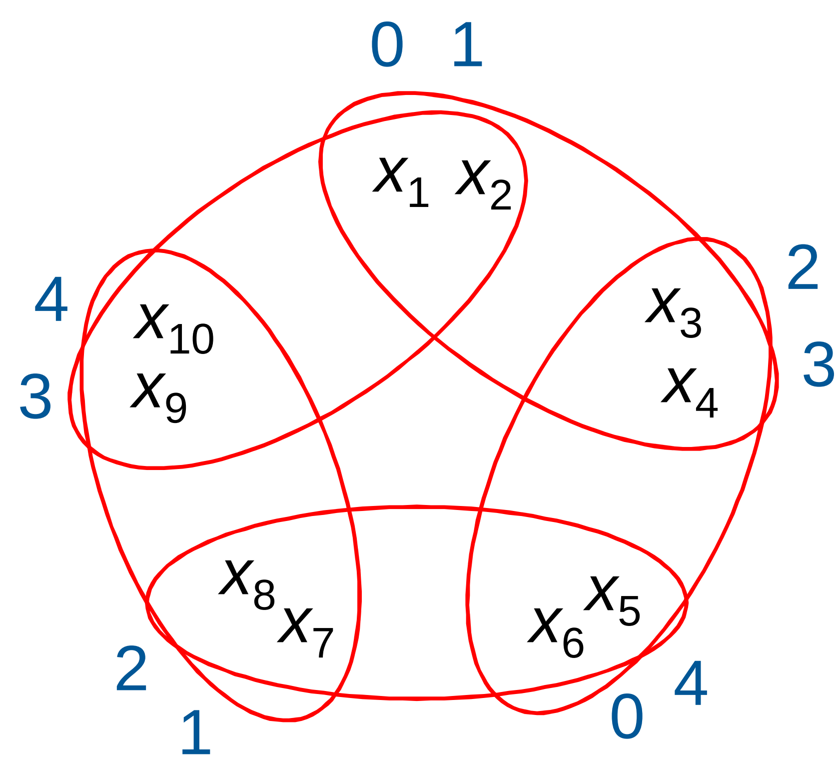
Odd Cycles

- We get a valid ***x*-cut**:

$$\sum_{i \in S} x_i \geq \left(sq - \frac{q-1}{4} L \right)(L-1) = 20$$

where $L = \left\lceil \frac{sq}{(q-1)/2} \right\rceil$

- The inequality is **facet-defining** if q is odd.



Odd Cycles

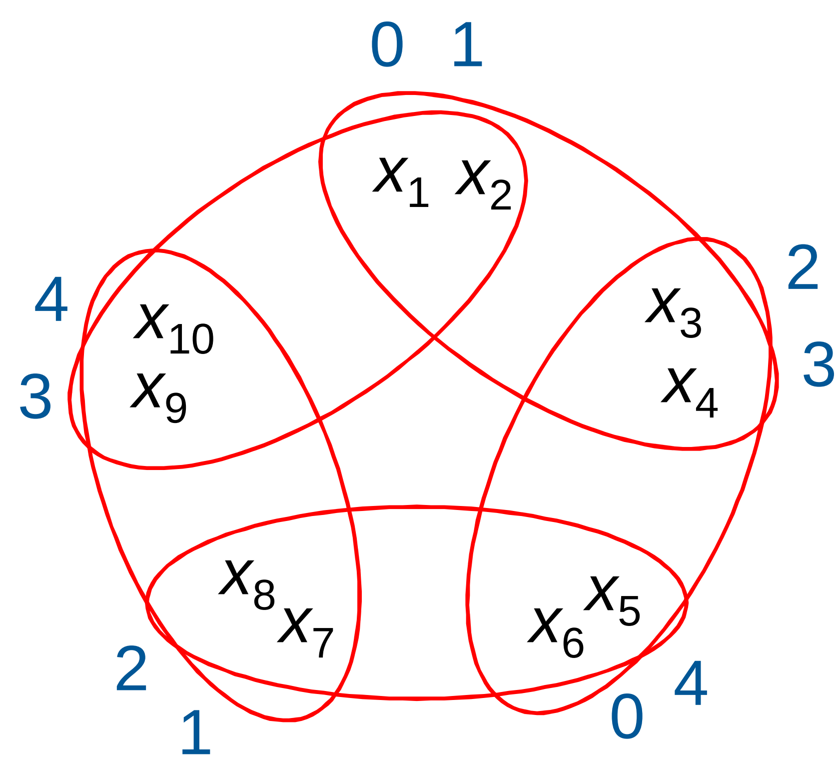
- We get a valid ***x*-cut**:

$$\sum_{i \in S} x_i \geq \left(sq - \frac{q-1}{4} L \right)(L-1) = 20$$

where $L = \left\lceil \frac{sq}{(q-1)/2} \right\rceil$

- The inequality is **facet-defining** if q is odd.
  - For $s = 1$ we get the odd hole cut

$$\sum_{i \in S} x_i \geq \frac{q+3}{2}$$



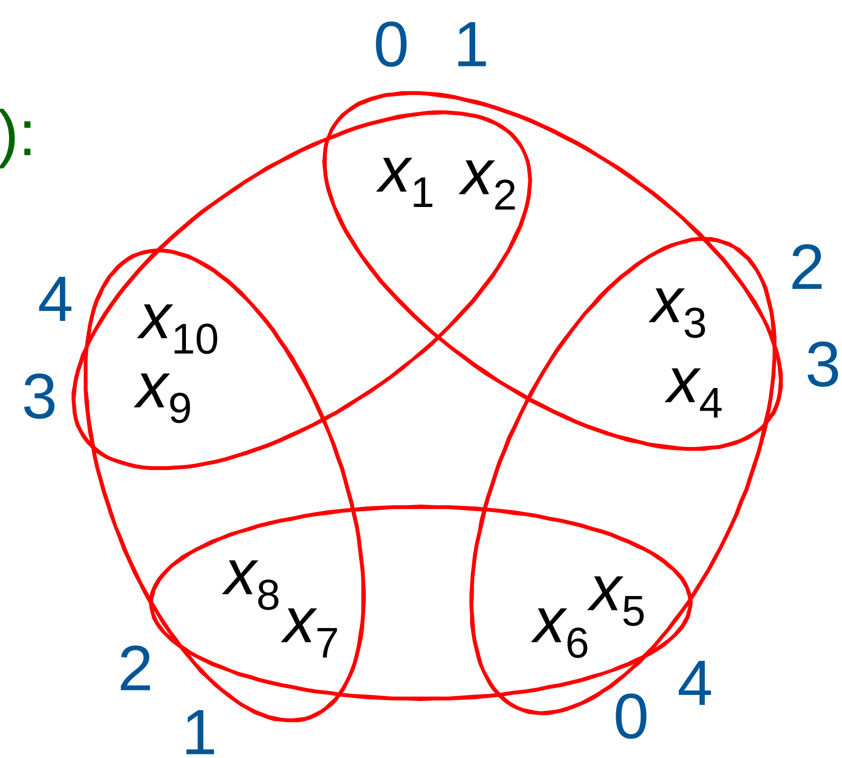
Odd Cycles

- We also get a valid ***z*-cut** (bound on number of colors z):

$$z \geq \frac{1}{qs}\sum_{i \in S} x_i + \left(1 - \frac{q-1}{4qs}L\right)(L-1)$$

$$= \frac{1}{10}\sum_{i \in S} x_i + 2$$

This is **facet defining.**



z-cuts in general

- In fact, facet-defining **x**-cuts for a graph coloring problem always give rise to facet-defining **z**-cuts:

  – **Theorem:** if $ax \geq b$ is facet defining for a coloring problem with domain $D = \{0, \delta, 2\delta, \ldots, (n-1)\delta\}$ for $\delta > 0$, then

  $$aez \geq ax + b$$

  is also facet defining, where $e = (1, \ldots, 1)$.

Mapping into 0-1 Space

- The finite-domain cuts map to

$$\sum_{i\in S}\sum_j j y_{ij} \geq \left(sq - \frac{q-1}{4}L\right)(L-1)$$

$$z \geq \frac{1}{qs}\sum_{i\in S}\sum_j j y_{ij} + \left(1 - \frac{q-1}{4qs}L\right)(L-1)$$

  – How do they compare with classical odd hole cuts?

Computed Bounds

Lower bound on number of colors in
0-1 model of 5-cycle

s **=**	**1**	**2**	**3**	**4**	**5**
All odd hole cuts*	2.5	4.0	6.0	8.0	10.0
x-cut only	2.0	4.0	6.0	8.0	10.0
z-cut only	2.3	4.5	6.77	9.0	10.0
x and **z**-cut only	2.6	5	7.53	10	12.52
Optimal	3	5	8	10	13
No. odd hole cuts	5	320	3645	20,480	78,125

* And clique inequalities

Computed Bounds

Lower bound on number of colors in
0-1 model of 7-cycle

***s* =**	**1**	**2**	**3**	**4**
All odd hole cuts*	2.33	4.0	6.0	8.0
x-cut only	2.0	4.0	6.0	8.0
z-cut only	2.21	4.36	6.5	8.68
x and **z**-cut only	2.43	4.71	7	9.36
Optimal	3	5	7	10
No. odd hole cuts	7	1792	45,927	458,752

* And clique inequalities

Computed Bounds

Lower bound on number of colors in
0-1 model of 9-cycle

s **=**	**1**	**2**	**3**
All odd hole cuts*	2.25	4.0	6.0
x-cut only	2.0	4.0	6.0
z-cut only	2.17	4.28	6.39
x and **z**-cut only	2.33	4.56	6.78
Optimal	3	5	7
No. odd hole cuts	9	9612	531,441

* And clique inequalities

Cuts in x-space

- Finite domain cuts can also be used in their original form.
  - This results in a much more compact relaxation.
    - $O(n)$ variables rather than $O(n^2)$ variables.
- Is the bound in the x-space as tight as in the 0-1 space?

Cuts in x-space

- Finite domain cuts can also be used in their original form.
  - This results in a much more compact relaxation.
    - $O(n)$ variables rather than $O(n^2)$ variables.
- Is the bound in the x-space as tight as in the 0-1 space?
  - Yes.

Computed Bounds

Lower bound on number of colors in
x-model of 5-cycle

s =	1	2	3	4	5
Clique cuts only	1.5	2.5	3.5	4.5	5.5
Plus **x**-cut	1.8	3.0	4.27	5.5	6.76
Plus **z**-cut	2.3	4.5	6.77	9.0	11.26
Plus **x** and **z**-cut	2.6	5	7.53	10	12.52
Optimal	3	5	8	10	13

Computed Bounds

Lower bound on number of colors in
x-model of 7-cycle

***s* =**	**1**	**2**	**3**	**4**
Clique cuts only	1.5	2.5	3.5	4.5
Plus **x**-cut	1.71	2.86	4.0	5.18
Plus **z**-cut	2.21	4.36	6.5	8.68
Plus **x** and **z**-cut	2.43	4.71	7	9.36
Optimal	3	5	7	10

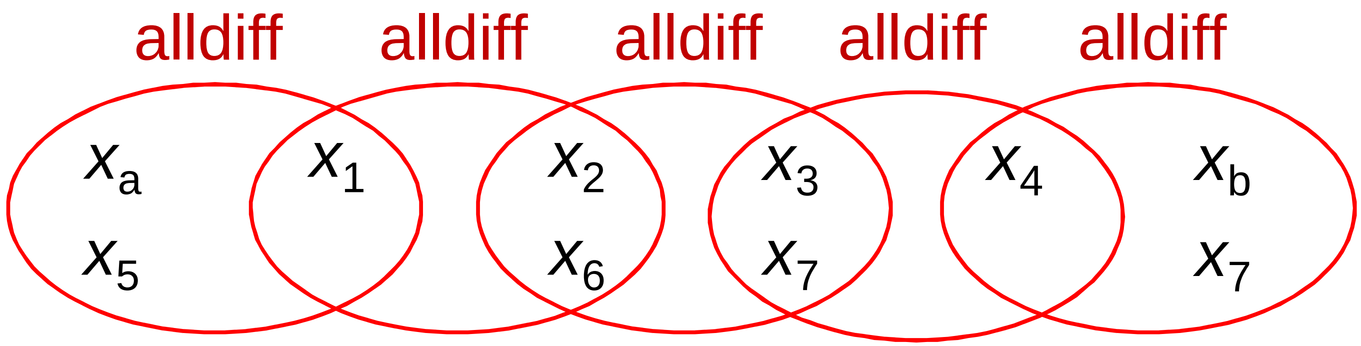
Computed Bounds

Lower bound on number of colors in *x*-model of 9-cycle

***s* =**	**1**	**2**	**3**
Clique cuts only	1.5	2.5	3.5
Plus **x**-cut	1.67	2.78	3.89
Plus **z**-cut	2.17	4.28	6.39
Plus **x** and **z**-cut	2.33	4.56	6.78
Optimal	3	5	7

Odd Paths

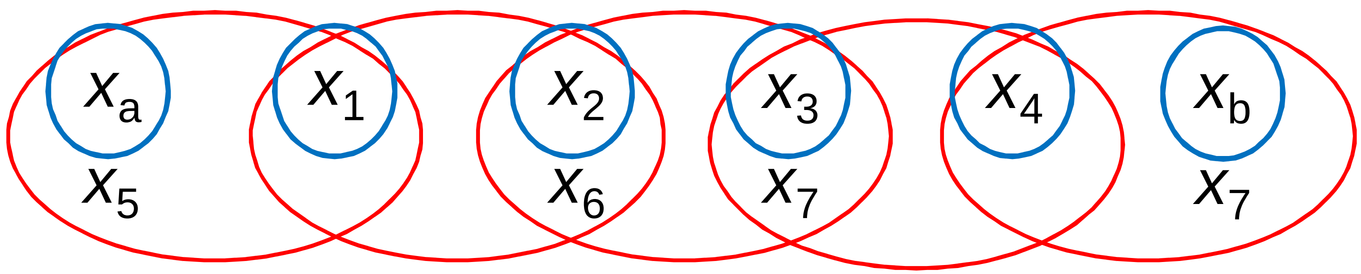
- A q-path looks like



$q = 5$

Odd Paths

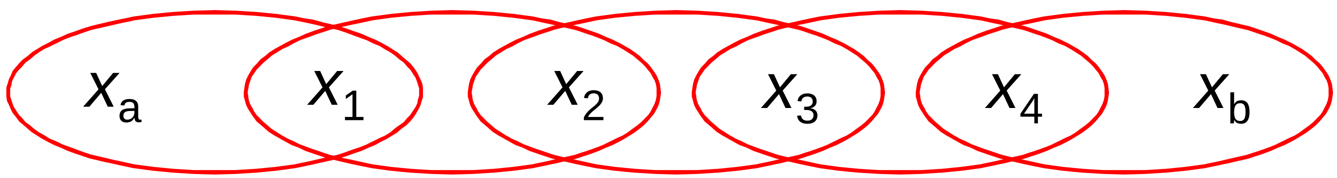
- Select q + 1 variables:



q = 5

Odd Paths

- This yields a valid inequality (**x**-cut)



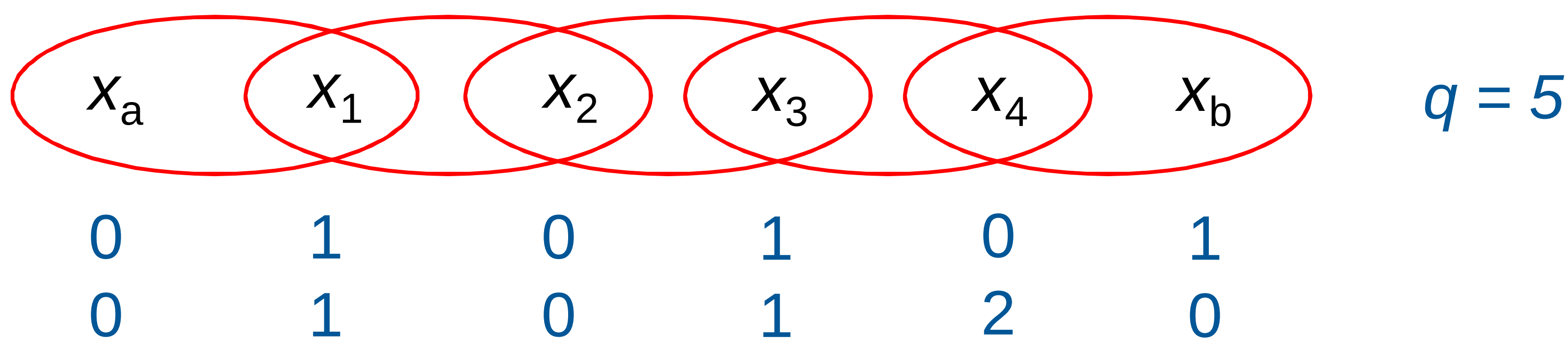
$q = 5$

0	1	0	1	0	1
0	1	0	1	2	0

$$2(x_a + x_b) + \sum_{i=1}^{q-1} x_i \geq \frac{q+3}{2} = 4$$

Odd Paths

- This yields a valid inequality (**x**-cut):

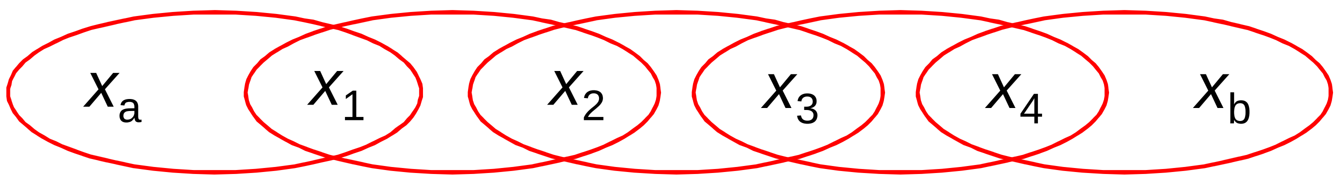


$$2(x_a + x_b) + \sum_{i=1}^{q-1} x_i \geq \frac{q+3}{2} = 4$$

- The inequality is **facet-defining** if q is odd.

Odd Paths

- We also have a **z**-cut



$q = 5$

0	1	0	1	0	1
2	0	1	2	1	2

$$z \geq \frac{1}{q+3}\left(2\left(x_a + x_b\right) + \sum_{i=1}^{q-1} x_i\right) + \frac{1}{2}$$

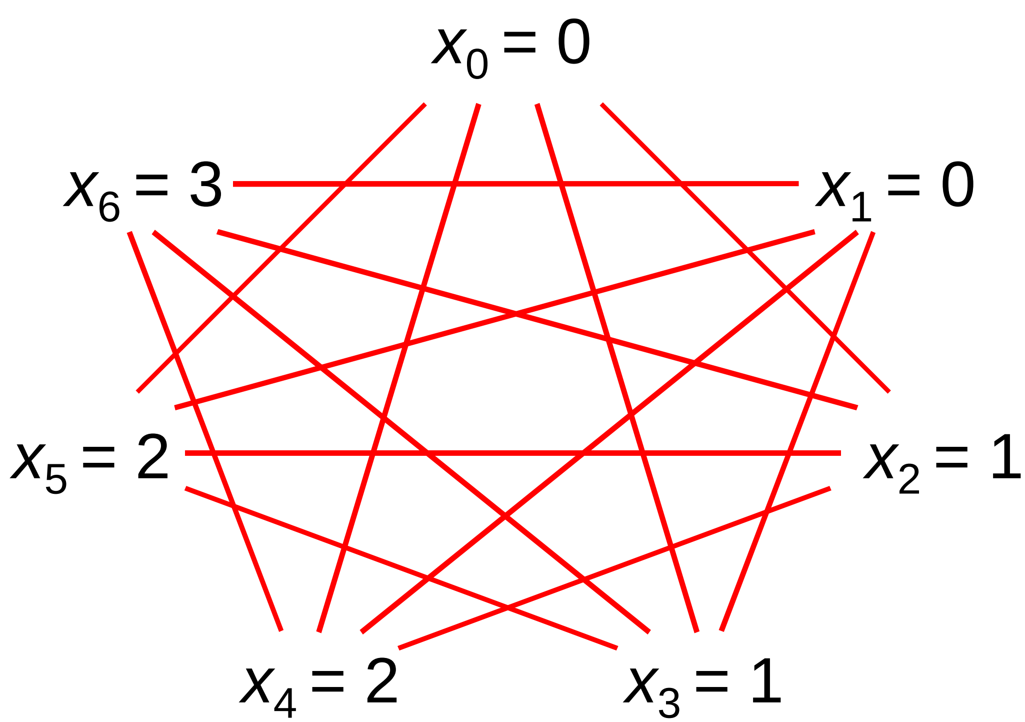
- This is also facet defining.

Mapping into 0-1 Space

- When mapped into 0-1 space, the finite domain cuts are **redundant** of the 0-1 model.
  - They don’t change the bound.
- However, the finite domain cuts provide a compact relaxation.

Webs

- A web $W(q,k)$ is a cycle of q vertices in which edges connect all vertices separated by distance at least k.
  - $W(q,2)$ is an anti-hole.



$W(7,2)$

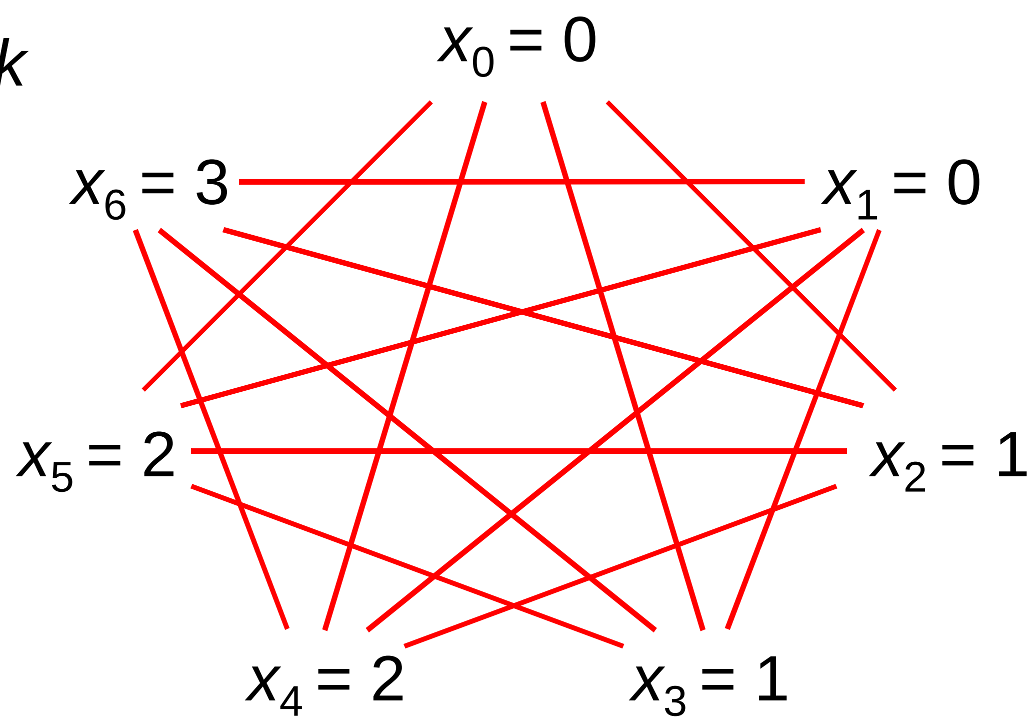
Webs

- If q and k are mutually prime,

$$\sum_i x_i \geq rq - \frac{1}{2}(r+1)rk$$

where $r = \left\lfloor \frac{q}{k} \right\rfloor$

is facet-defining.



$$\sum_i x_i \geq 9$$

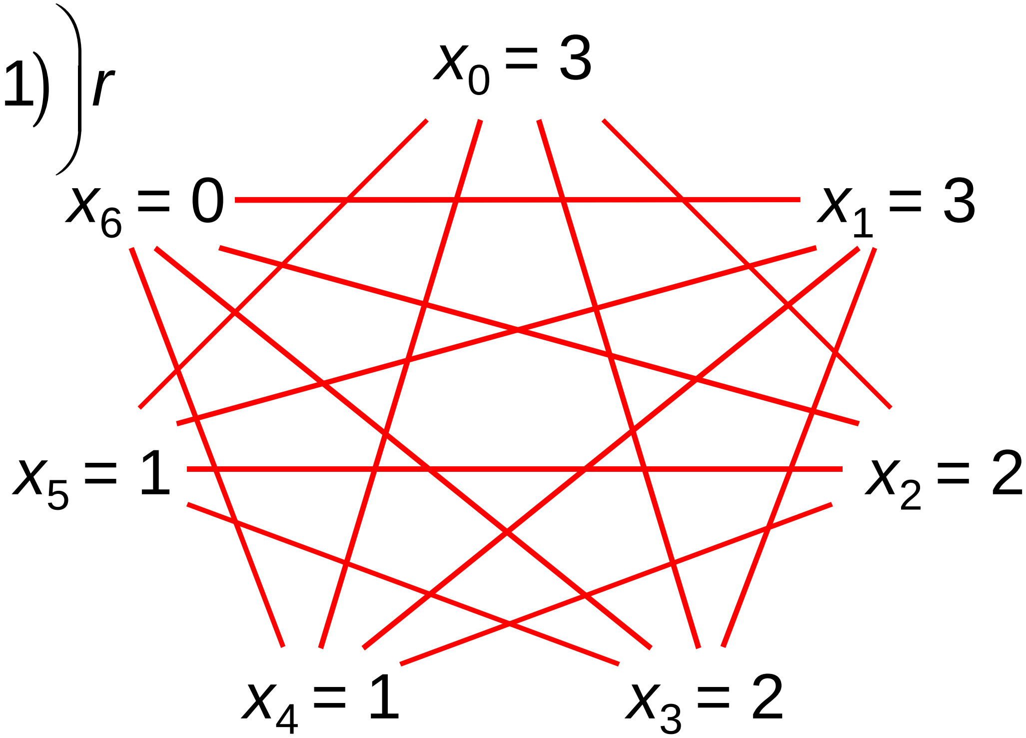
Webs

- If q and k are mutually prime,

$$z \geq \frac{1}{q}\sum_i x_i + \left(1 - \frac{k}{2q}(r+1)\right)r$$

where $r = \left\lfloor \frac{q}{k} \right\rfloor$

is facet-defining.



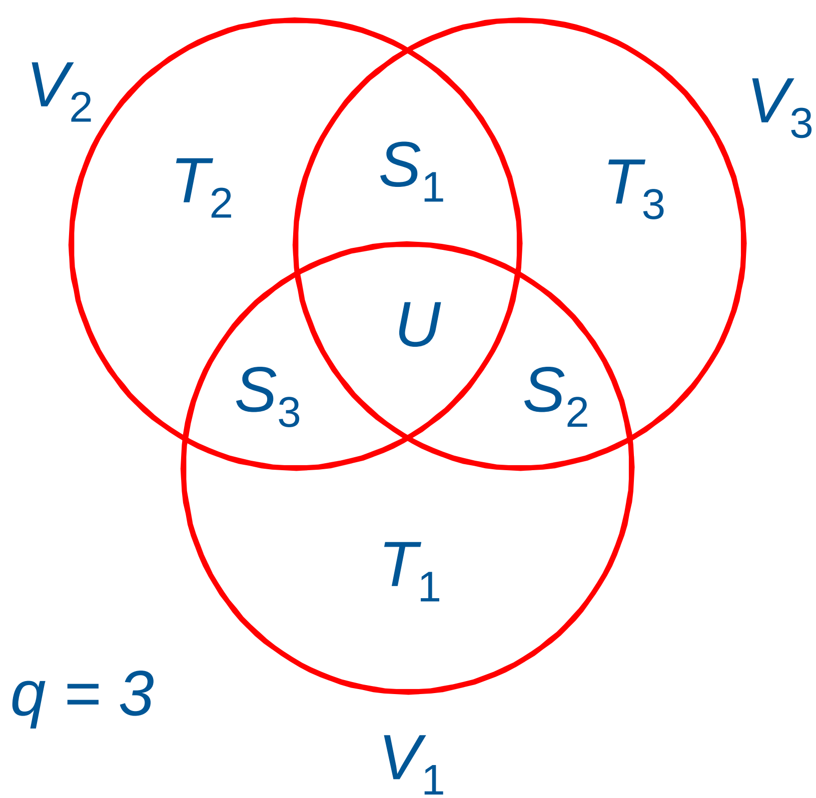
$$\sum_i x_i \geq 9$$

Mapping into 0-1 Space

- Finite-domain **web cuts** compare similarly with finite-domain **odd hole cuts**.

Intersecting Systems

- An intersecting system looks something like



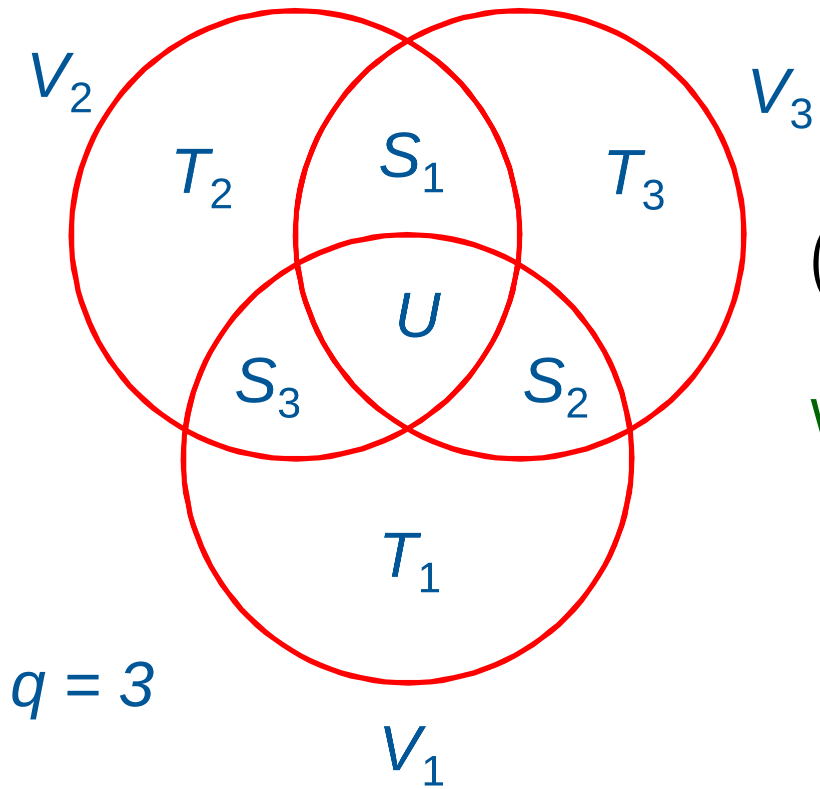
$$S_k = \bigcap_{\ell \neq k} V_\ell \setminus V_k$$

$$T_k = V_k \setminus \bigcup_{\ell \neq k} V_\ell$$

$$U = \bigcap_k V_k$$

Intersecting Systems

- Facet-defining inequality. Let $S = \bigcup_k S_k \quad T = \bigcup_k T_k \quad u = |U|$



A valid inequality is:

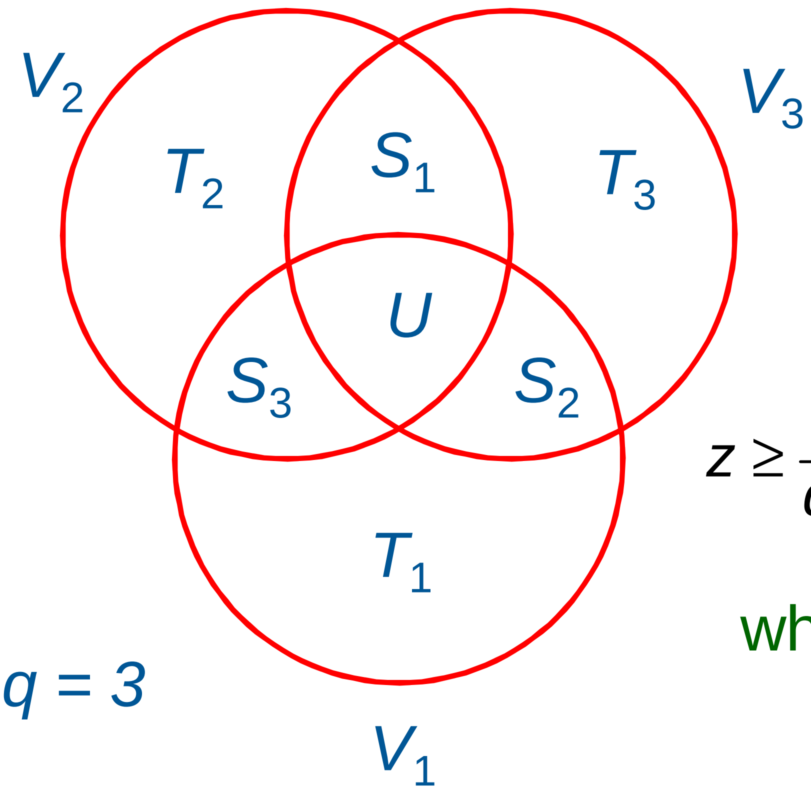
$$(qs+u)\sum_{i\in T} x_i + \frac{q(q-1)}{2}\sum_{i\in S\cup U} x_i \geq b$$

where

$$b = \frac{1}{2}q(q-1)(qs+u)(qs+u+1)$$

Intersecting Systems

- Facet-defining inequality. Let $S = \bigcup_k S_k \quad T = \bigcup_k T_k \quad u = |U|$



$q = 3$

A valid bound is:

$$z \geq \frac{2}{q(q+1)} \sum_{i \in T} x_i + \frac{q-1}{(q+1)(qs+u)} \sum_{i \in S \cup U} x_i + c$$

where

$$c = \tfrac{1}{2} \frac{q-1}{q+1} (qs + u + 1)$$

Benchmark Instances

with < 100 variables

Lower bound on number of colors in
0-1 model. Odd cycle cuts for $s = 1,2,3$

Instance	Odd hole	Odd cycle	Opt	Odd hole time	Odd cycle time
1-FullIns_3	2	2	3	0.4	0.4
1-FullIns_4	2	2	4	208	0.4
1-insertions_4	1.33	1.43	4	30.3	2.4
2-FullIns_3	2	2	4	0.9	0.7
2-insertions_3	1.25	1.33	3	2.9	0.2

Benchmark Instances

Lower bound on number of colors in
0-1 model. Odd cycle cuts for $s = 1,2,3$

Instance	Odd hole	Odd cycle	Opt	Odd hole time	Odd cycle time
3-FullIns_3	2	2	5	25.8	0.2
3-insertions_3	1.2	1.27	3	11.5	1.0
4-insertions	1.17	1.23	3	12.1	6.0
david	2	8	10	11.0	0.8
huck	2	8	10	7.2	0.3
jean	2	8	9	10.2	1.8

Benchmark Instances

Lower bound on number of colors in 0-1 model. Odd cycle cuts for $s = 1,2,3$

Instance	Odd hole	Odd cycle	Opt	Odd hole time	Odd cycle time
mug88_1	2	2	3	7.8	2.7
Mug88_25	2	2	3	5.3	1.7

Benchmark Instances

Lower bound on number of colors in
0-1 model. Odd cycle cuts for $s = 1,2,3$

Instance	Odd hole	Odd cycle	Opt	Odd hole time	Odd cycle time
myciel3	1.5	1.6	3	0.0	0.0
myciel4	1.5	1.6	4	0.6	0.0
myciel5	1.5	1.6	5	7.9	0.1
myciel6	1.5	1.6	3	1754	0.6

Benchmark Instances

Lower bound on number of colors in 0-1 model. Odd cycle cuts for $s = 1,2,3$

Instance	Odd hole	Odd cycle	Opt	Odd hole time	Odd cycle time
queen5_5	2	2	4	0.4	0.0
queen6_6	2	5	6	1.5	0.1
queen7_7	2	3.71	6	10.6	0.2
queen8_8	?	3.38	8	?	3.4
queen8_12	2	8	11	439.6	1.7
queen9_9	2	8	9	212.4	1.3

Future Work

- Conduct polyhedral for other finite-domain formulations.
  - Cumulative scheduling.
  - Circuit constraint (TSP).
  - Etc.

Bounds from Binary Decision Diagrams

Joint work with David Bergman, Andre Cire,
Willem van Hoeve

Binary Decision Diagrams

- **BDD**s historically used for circuit design and verification.
  - Lee 1959, Akers 1978, Bryant 1986.

Binary Decision Diagrams

- **BDD**s historically used for circuit design and verification.
  - Lee 1959, Akers 1978, Bryant 1986.
- **Compact** graphical representation of **boolean** function.
  - Can also represent **feasible set** of problem with binary variables.
  - Slight generalization (MDDs) represents finite domain variables.

Binary Decision Diagrams

- **BDD**s historically used for circuit design and verification.
  - Lee 1959, Akers 1978, Bryant 1986.
- **Compact** graphical representation of **boolean** function.
  - Can also represent **feasible set** of problem with binary variables.
  - Slight generalization (MDDs) represents finite domain variables.
- BDD is result of superimposing isomorphic subtrees in a search tree.
  - Unique reduced BDD for given variable ordering.

The 0-1 inequality

$$\begin{aligned} &300x_0 + 300x_1 + 285x_2 + 285x_3 + 265x_4 + 265x_5 + 230x_6 \\ &+230x_7 + 190x_8 + 200x_9 + 400x_{10} + 200x_{11} + 400x_{12} + 200x_{13} \\ &+400x_{14} + 200x_{15} + 400x_{16} + 200x_{17} + 400x_{18} \le 2700 \end{aligned}$$

has 117,520 minimal solutions

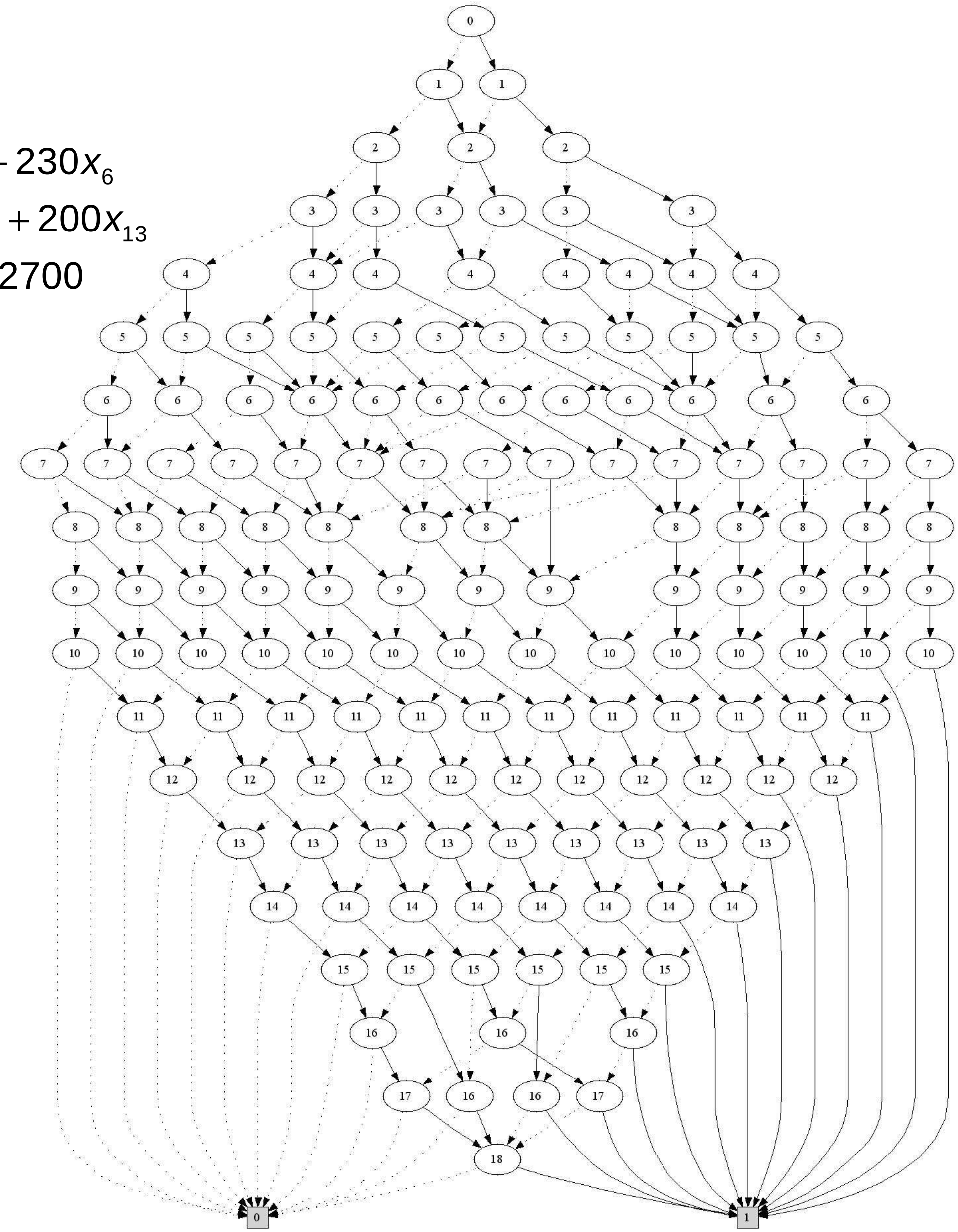
The 0-1 inequality

$$300x_0 + 300x_1 + 285x_2 + 285x_3 + 265x_4 + 265x_5 + 230x_6 + 230x_7 + 190x_8 + 200x_9 + 400x_{10} + 200x_{11} + 400x_{12} + 200x_{13} + 400x_{14} + 200x_{15} + 400x_{16} + 200x_{17} + 400x_{18} \le 2700$$

has 117,520 minimal solutions

The BDD has only 152 nodes.

Paths from top to bottom right correspond to feasible solutions



Binary Decision Diagrams

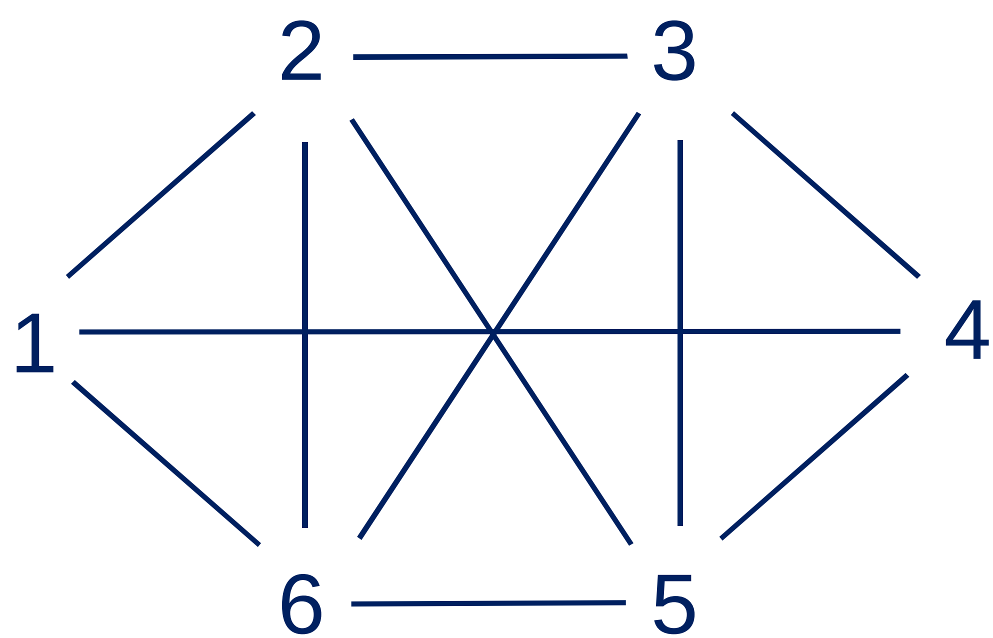
- BDD can grow exponentially with problem size.
  - So we use a smaller, **relaxed** BDD that represents **superset** of feasible set.
    - Andersen, Hadzic, Hooker, Tiedemann 2007.
    - For alldiff systems, reduced search tree from >1 million nodes to 1 node.
    - Subsequent papers with Hadzic, Hoda, van Hoeve, O'Sullivan.

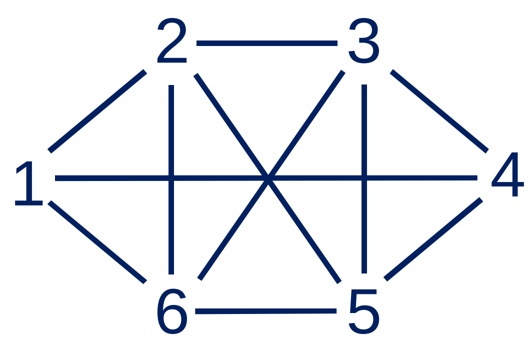
- We focus on **independent set problem** on a graph…

Independent Set Problem

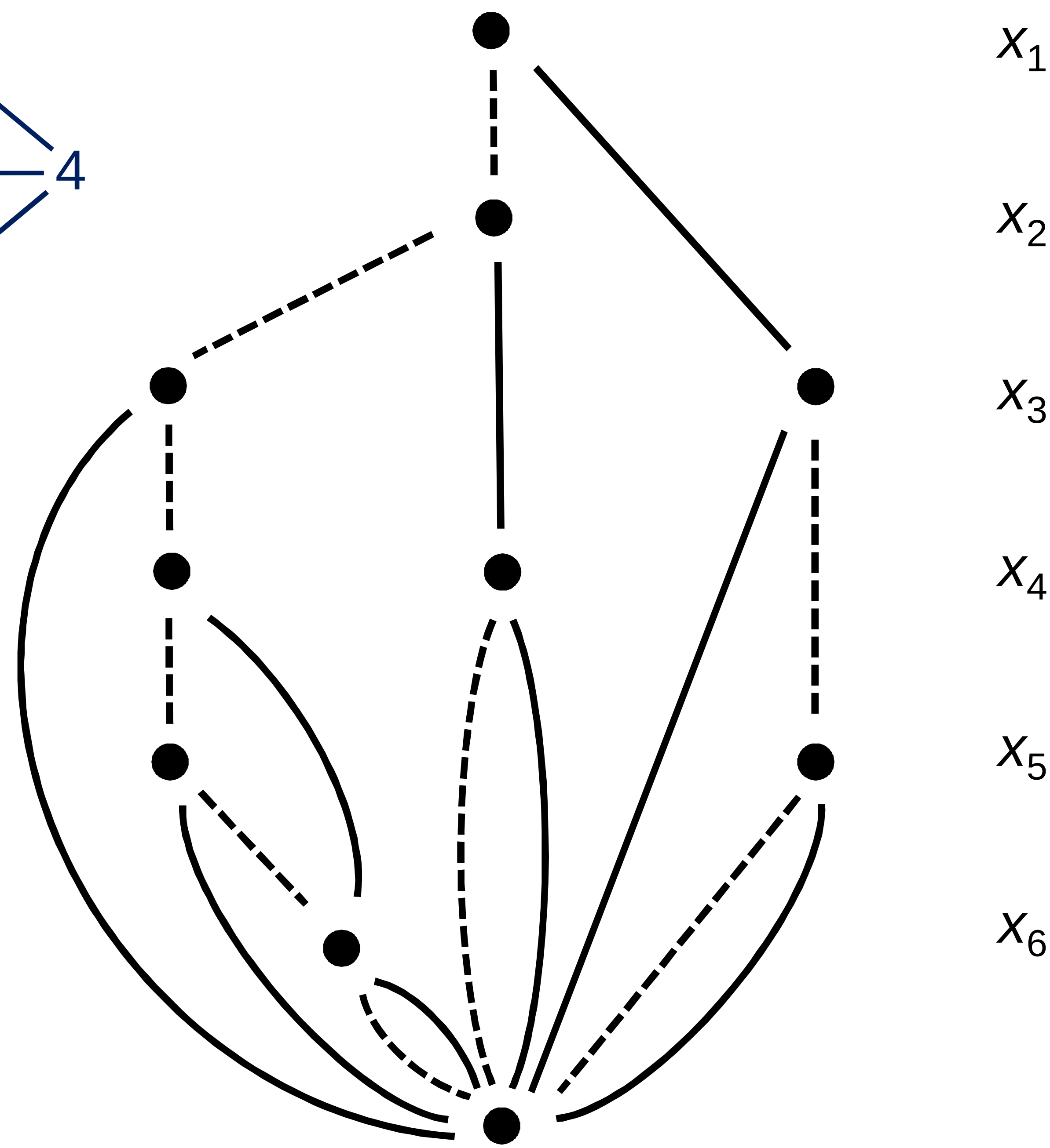
Let each vertex have weight w_i

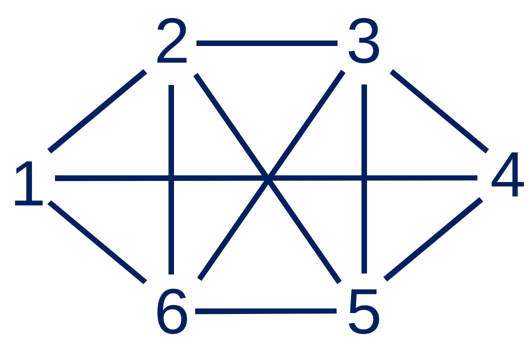
Select nonadjacent vertices to maximize $\sum_i w_i x_i$



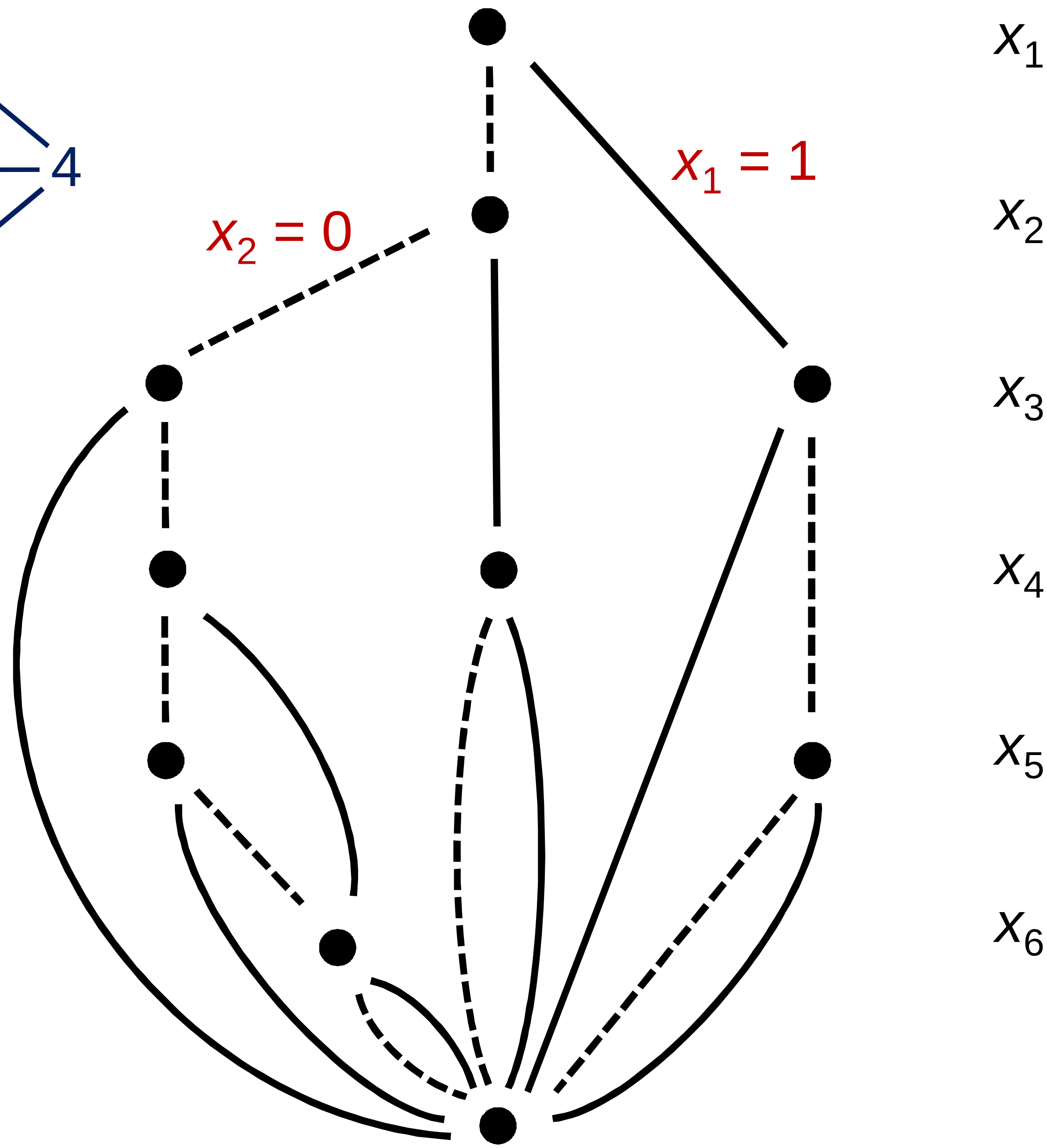


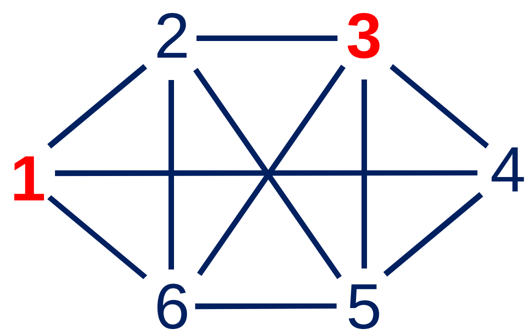
Exact BDD
for
independent
set problem



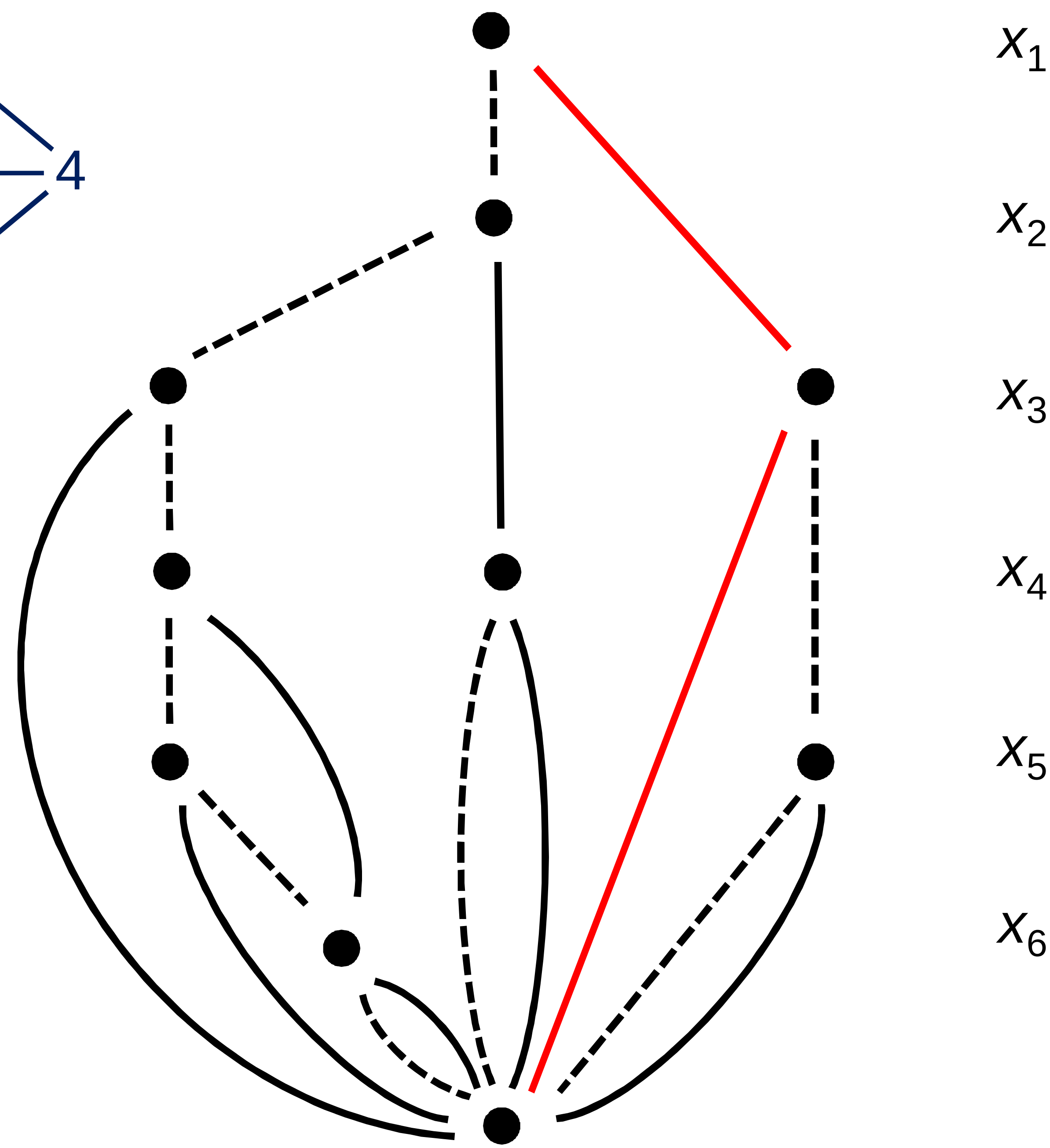


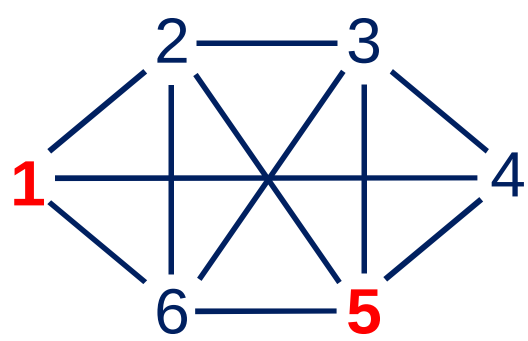
Exact BDD
for
independent
set problem



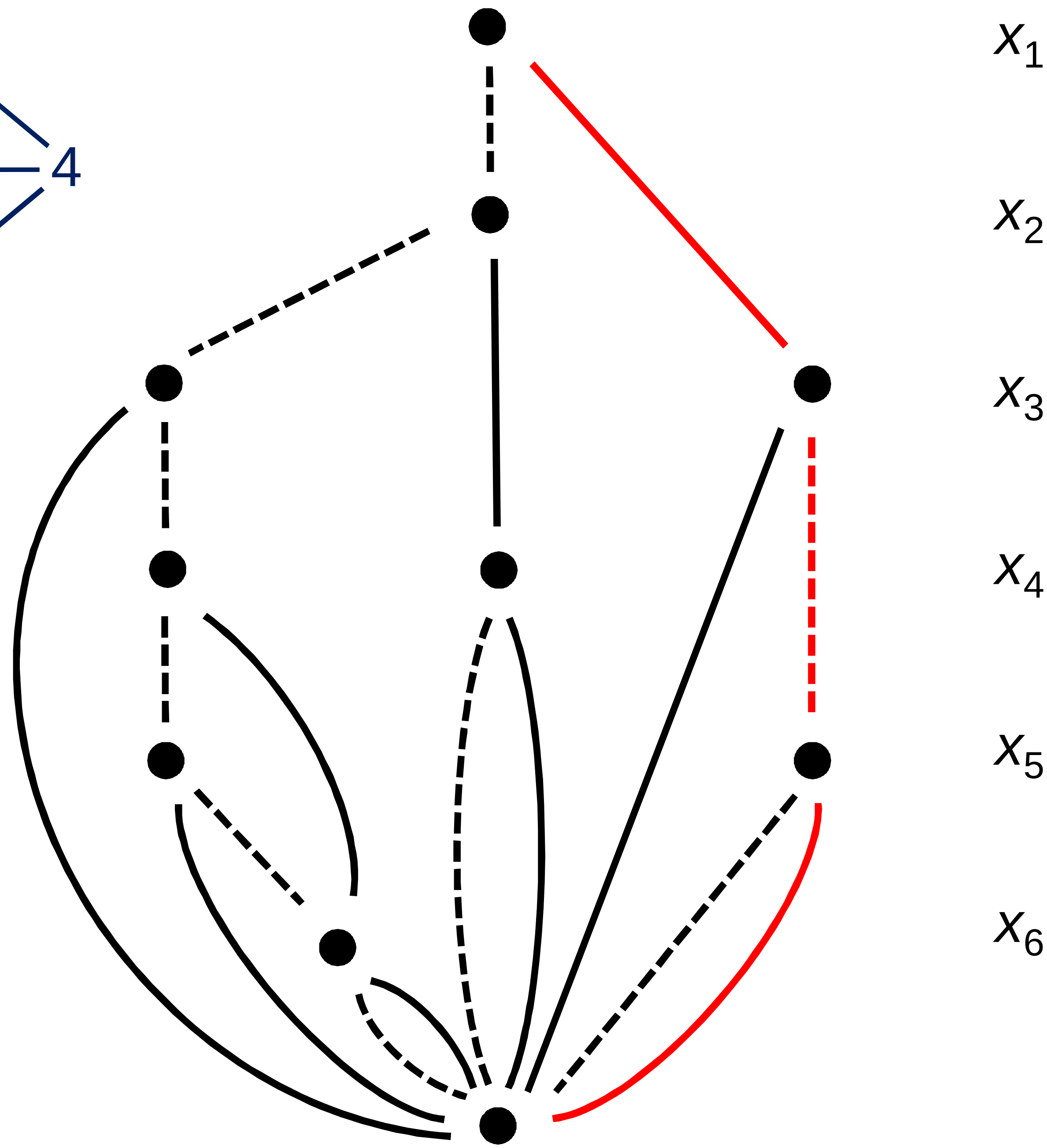


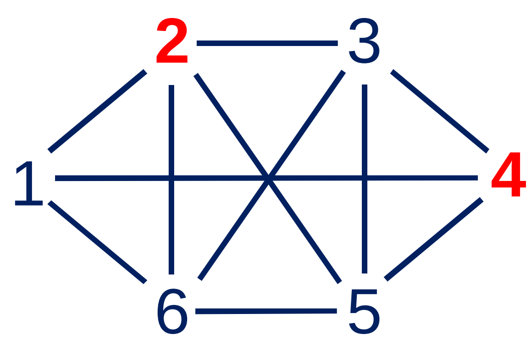
Paths from top to bottom correspond to the 11 feasible solutions

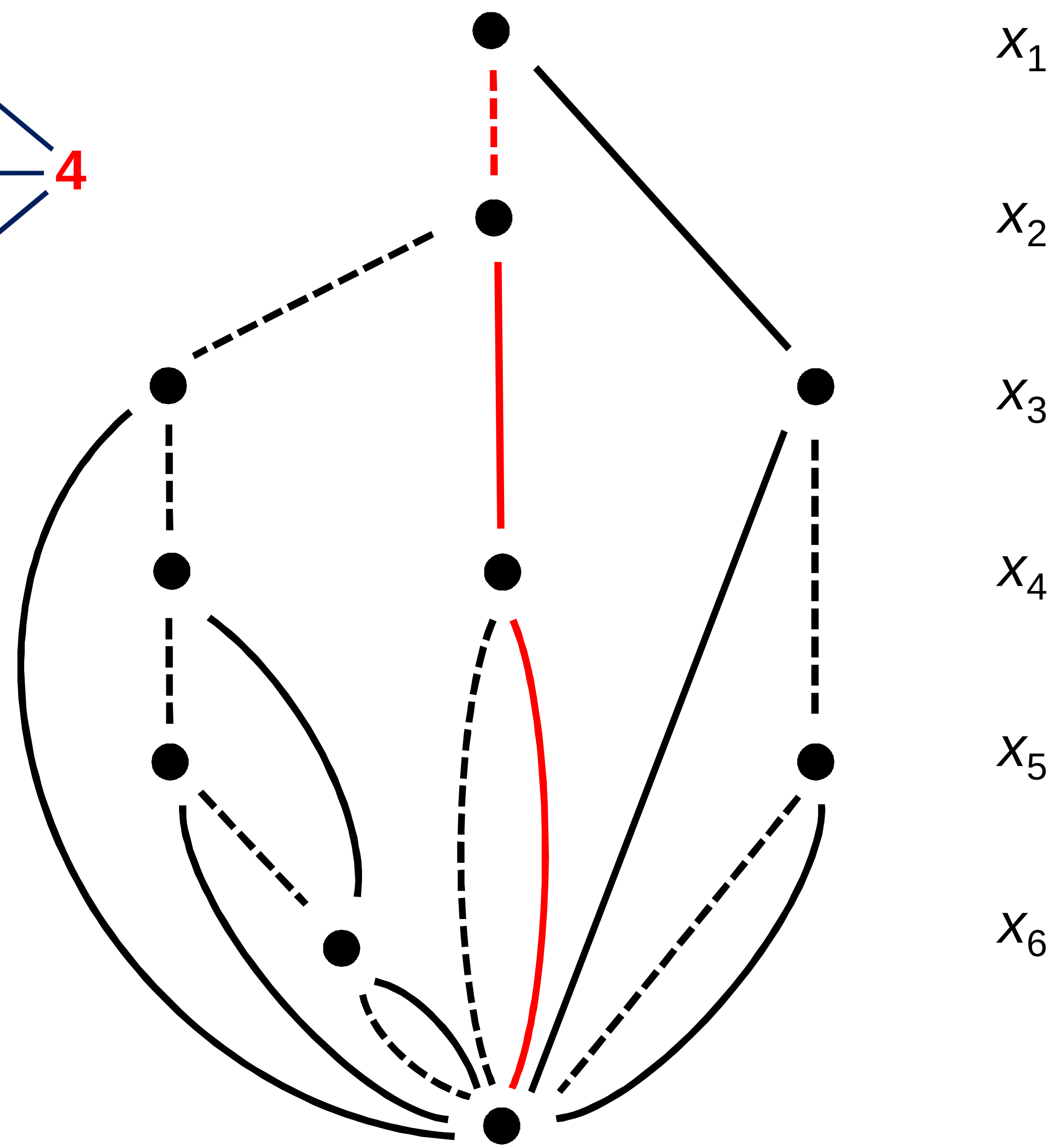




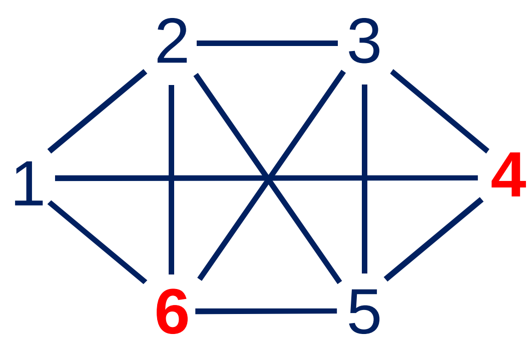
Paths from top to bottom correspond to the 11 feasible solutions



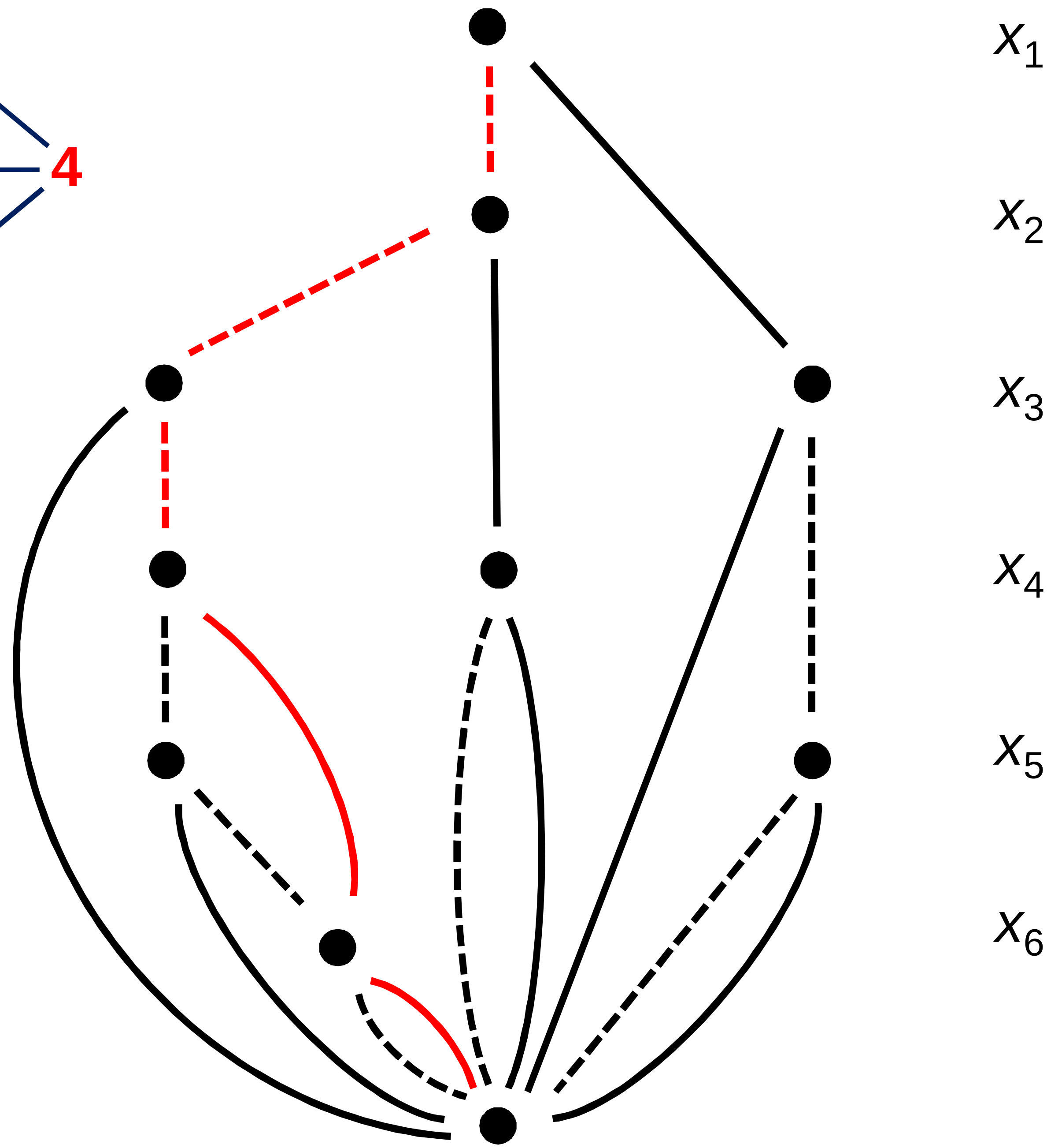


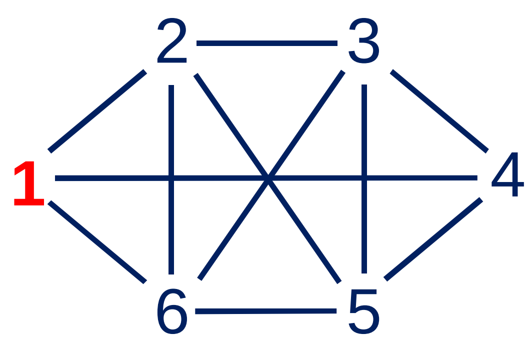


Paths from top to bottom correspond to the 11 feasible solutions



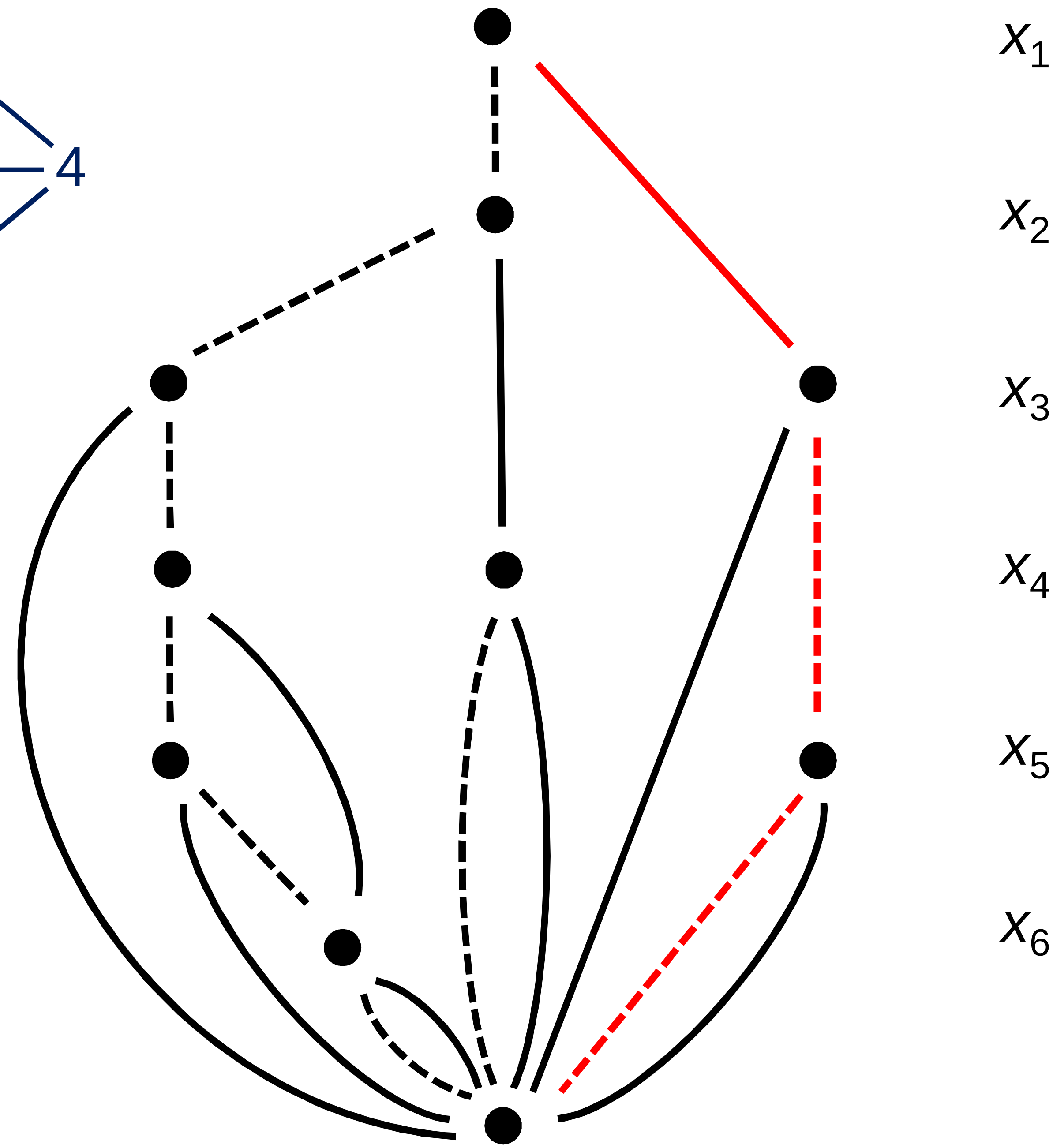
Paths from top to bottom correspond to the 11 feasible solutions

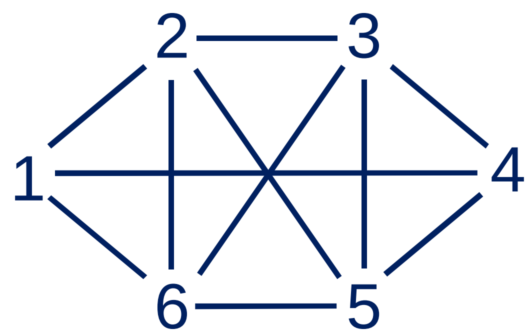




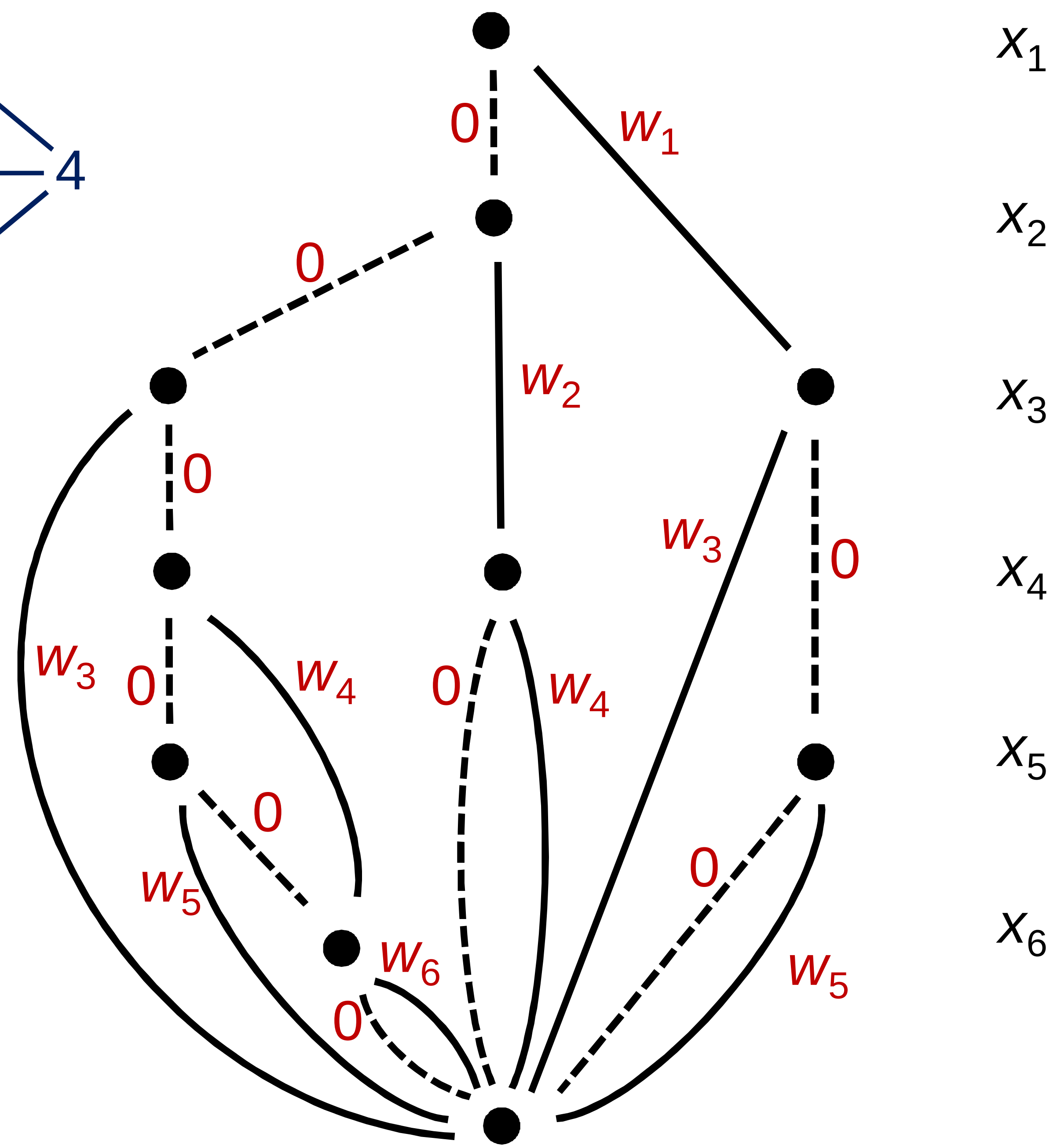
Paths from top to bottom correspond to the 11 feasible solutions

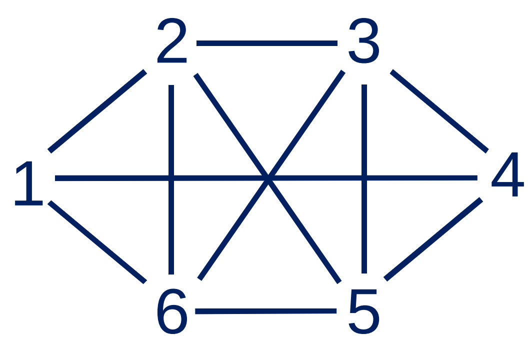
…and so forth



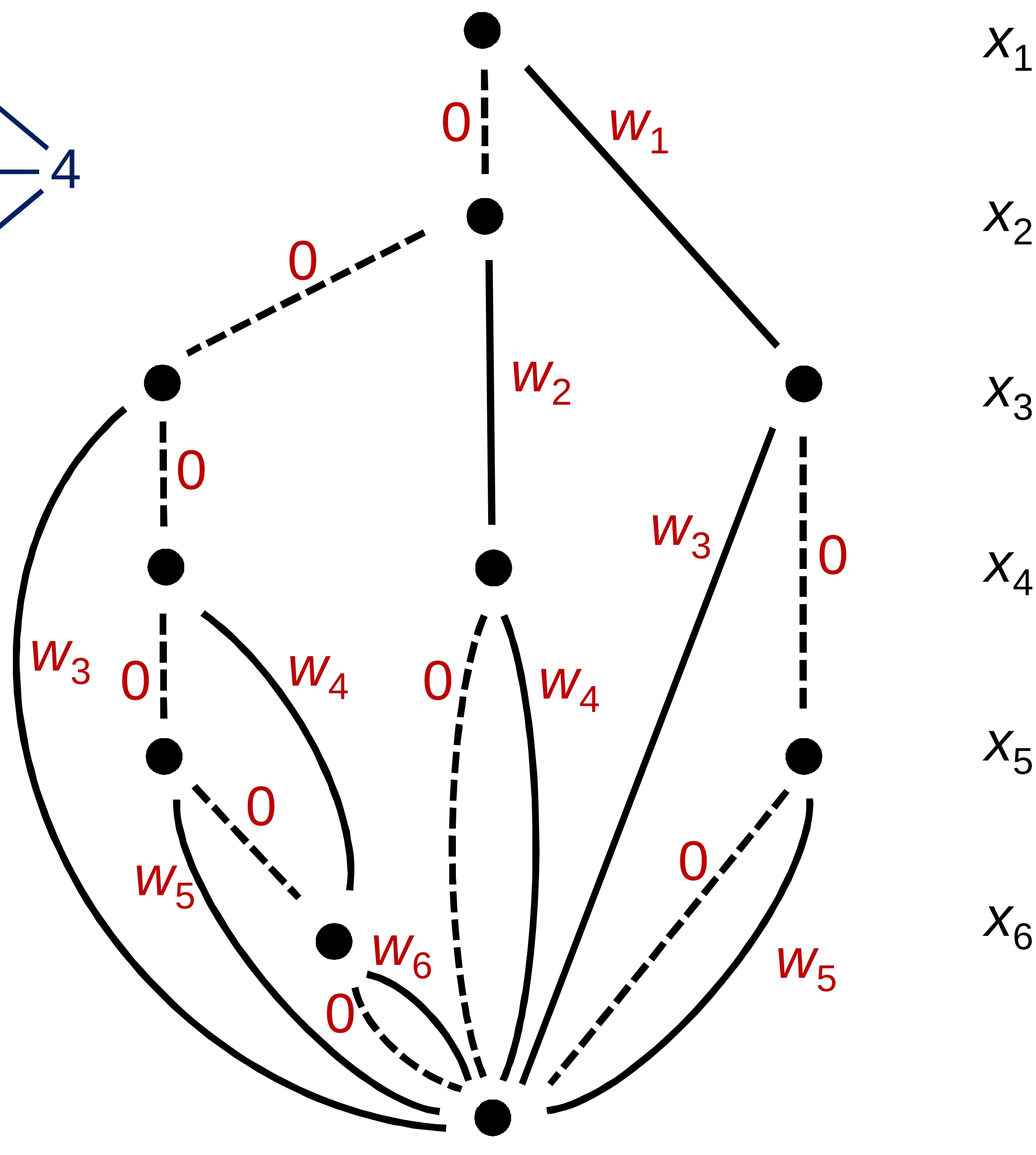


For objective function, associate weights with arcs





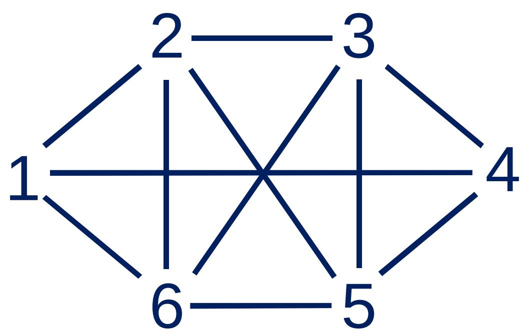
For objective function, associate weights with arcs

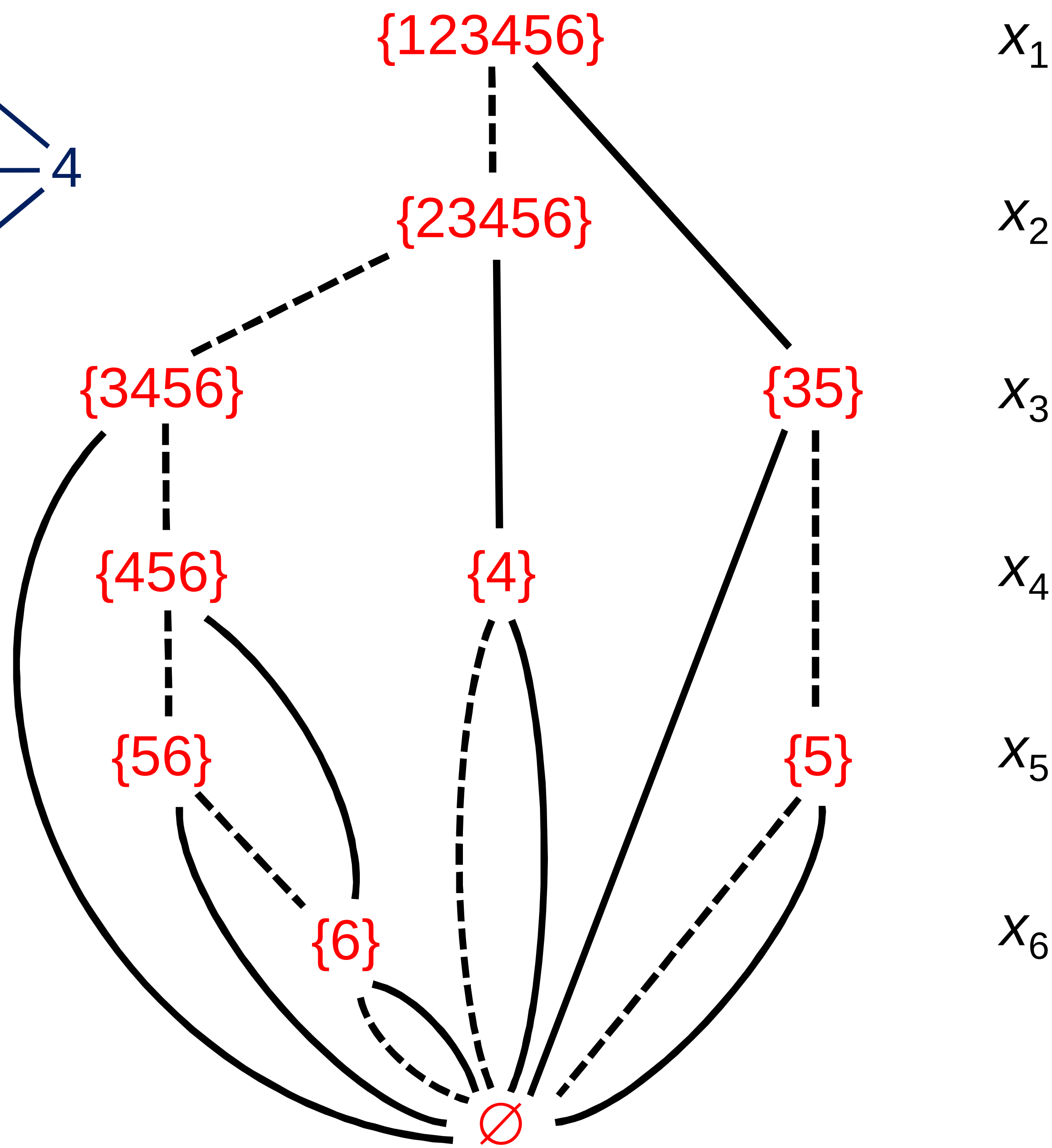


Optimal solution is **longest path**

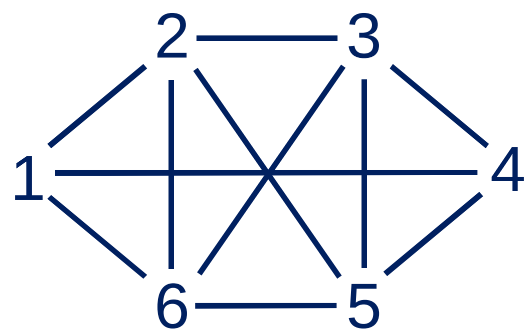
Objective Function

- In general, objective function can be any **separable function.**
  - Linear or nonlinear, convex or nonconvex.





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



{123456}

x_1

x_2

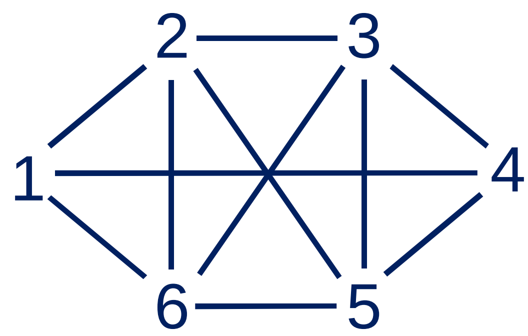
x_3

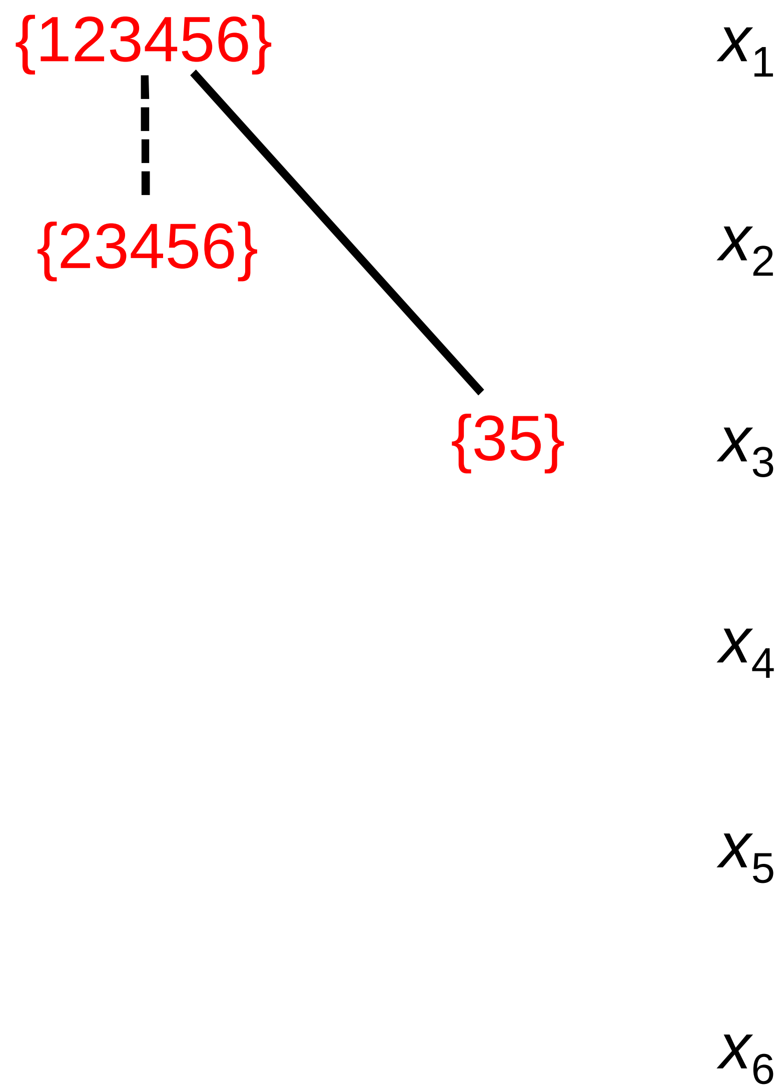
x_4

x_5

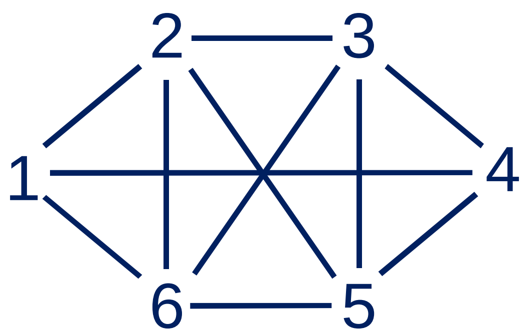
x_6

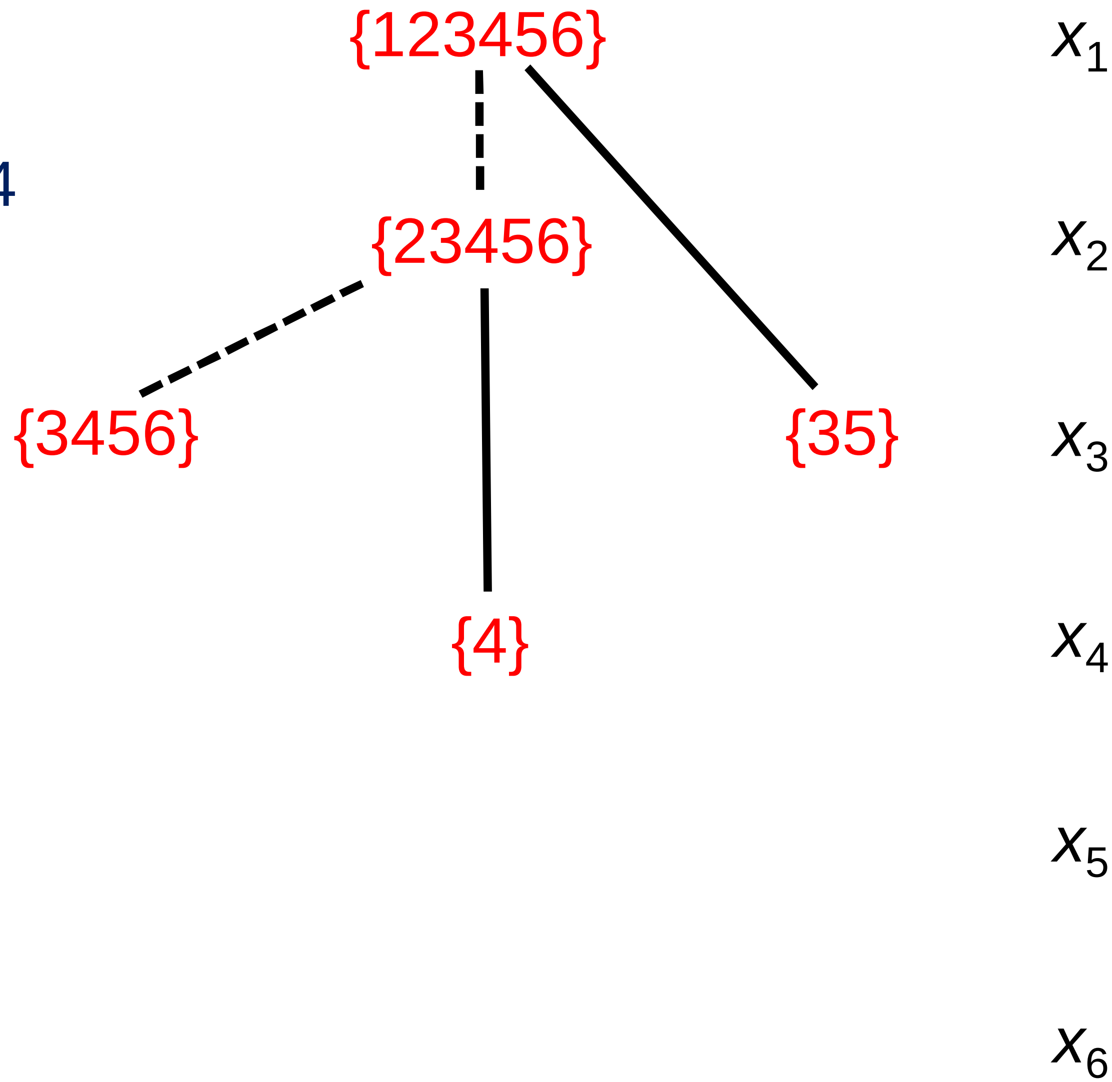
To build BDD, associate **state** with each node



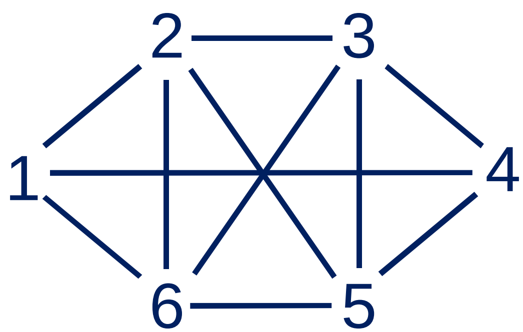


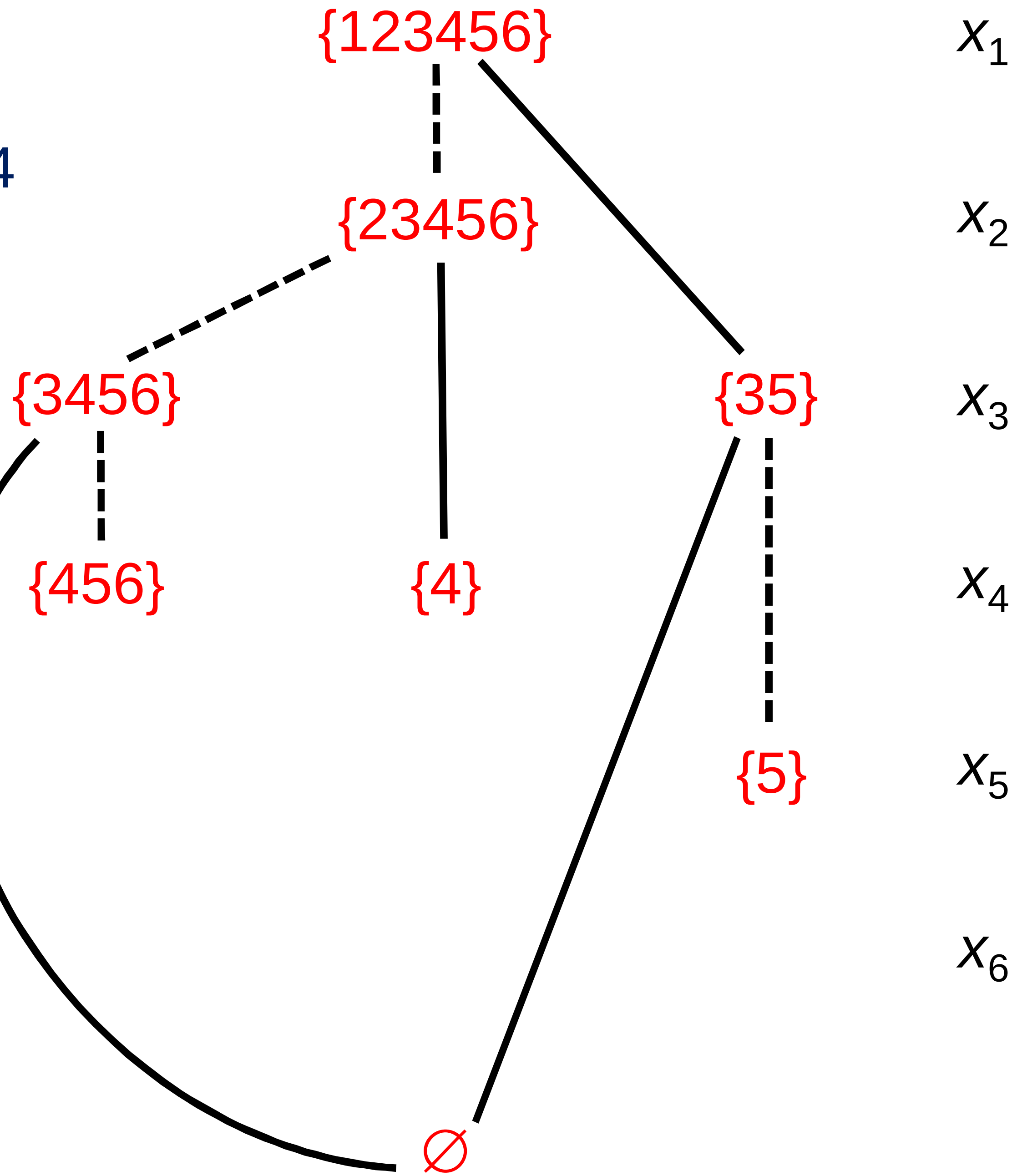
To build BDD, associate **state** with each node



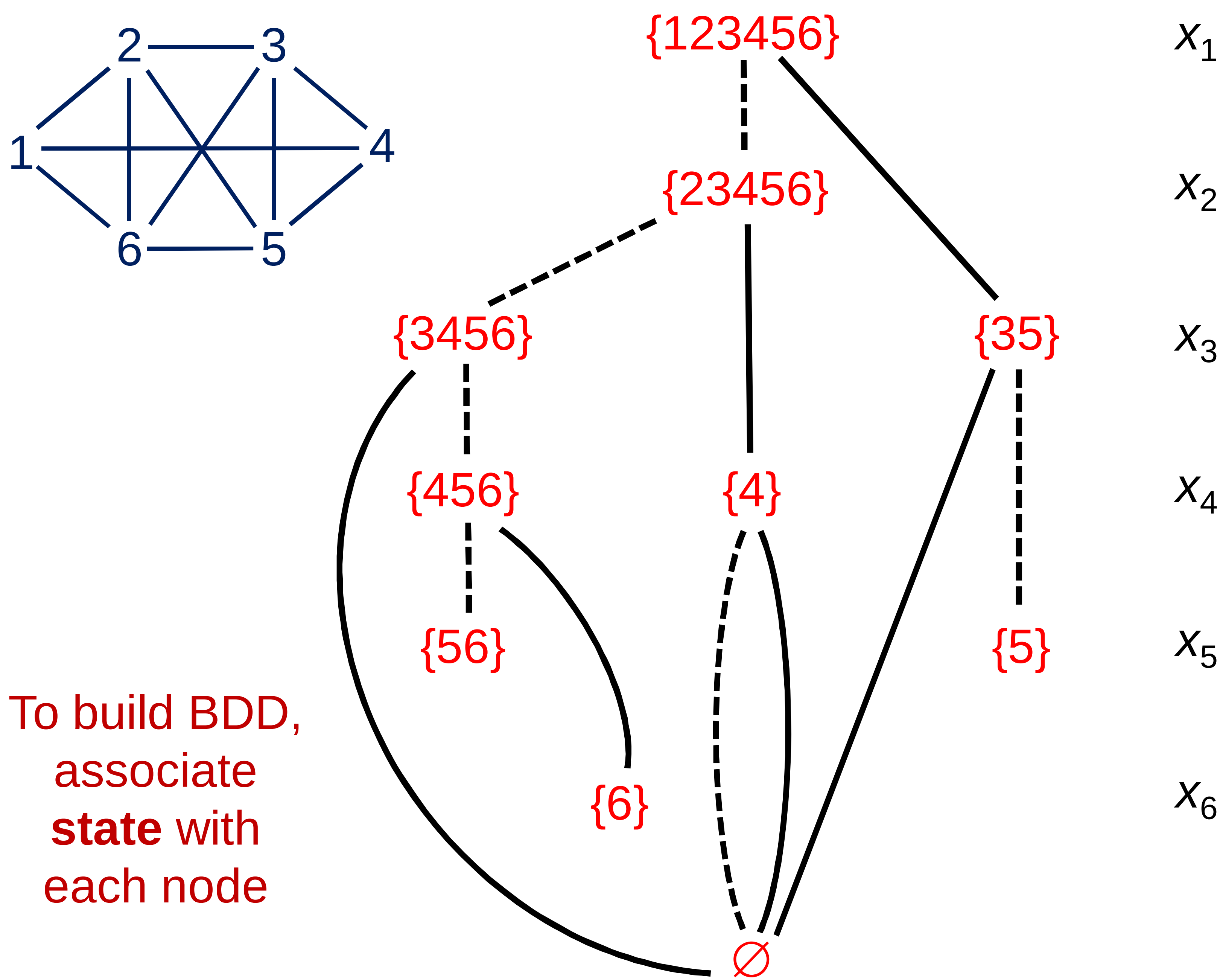


To build BDD, associate **state** with each node

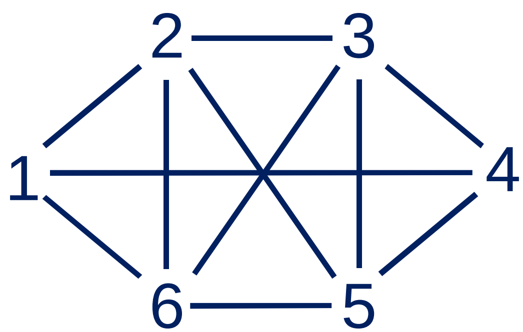


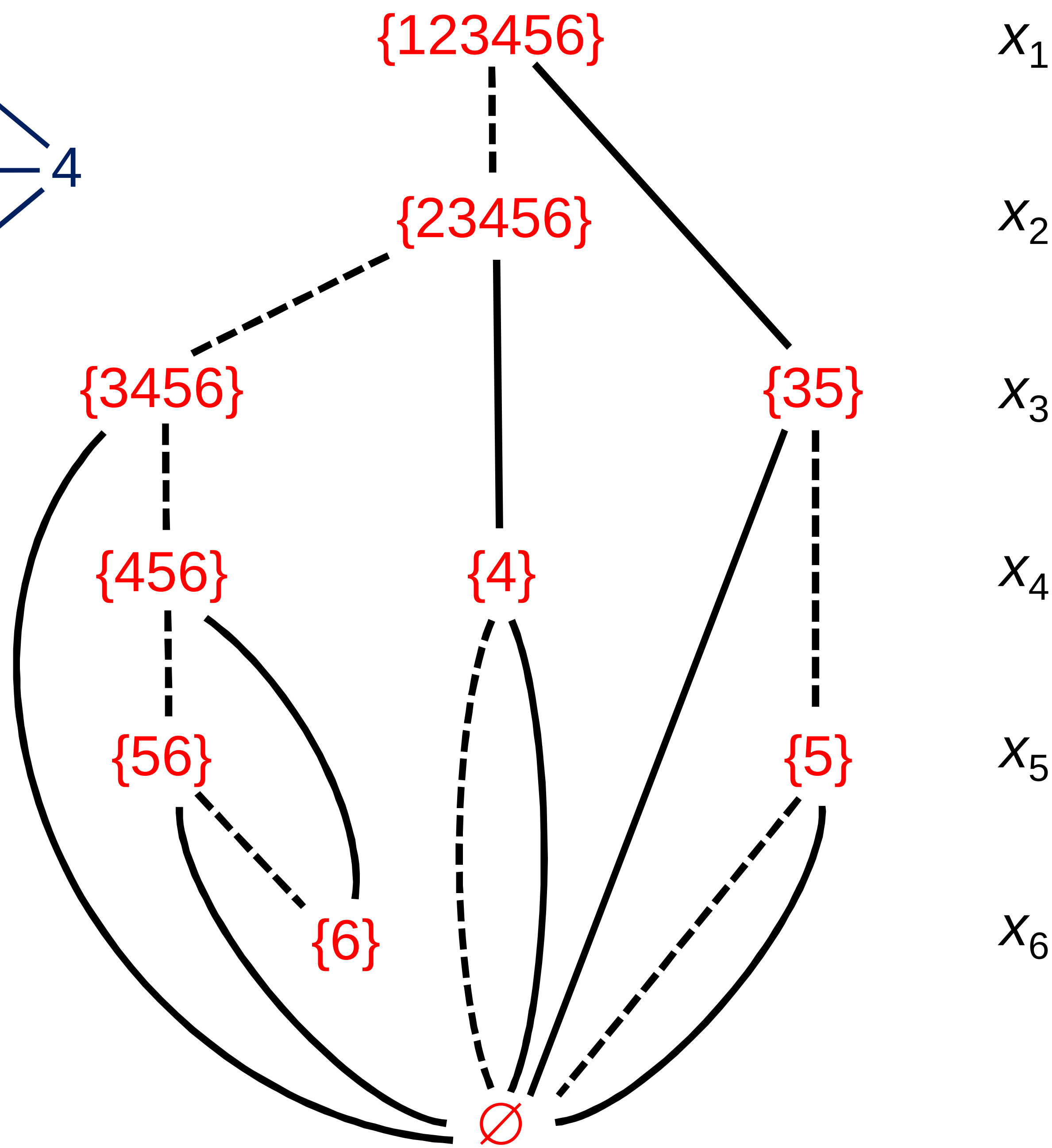


To build BDD, associate **state** with each node

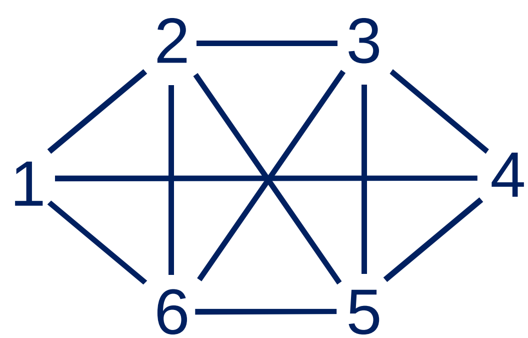


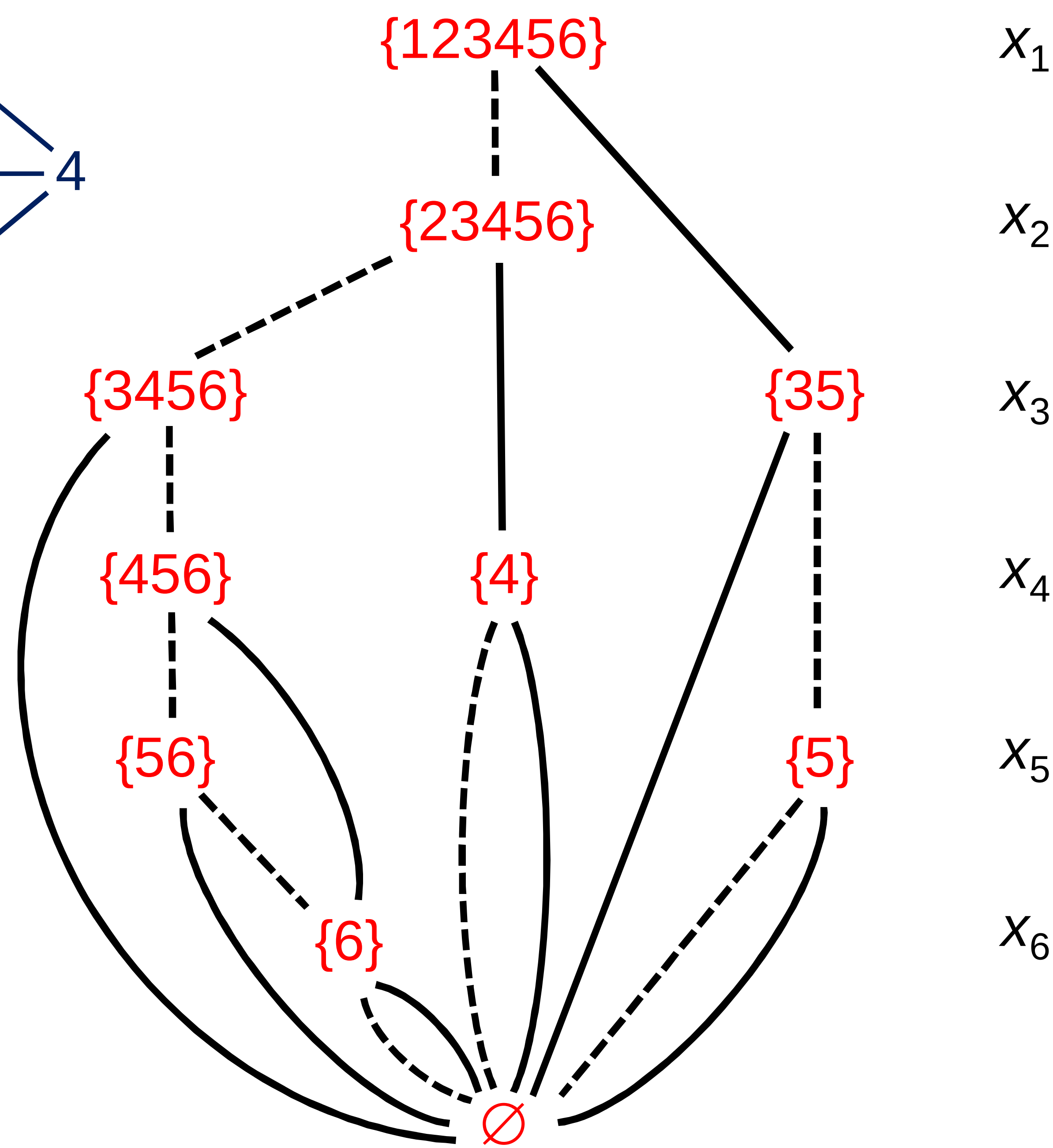
To build BDD, associate **state** with each node





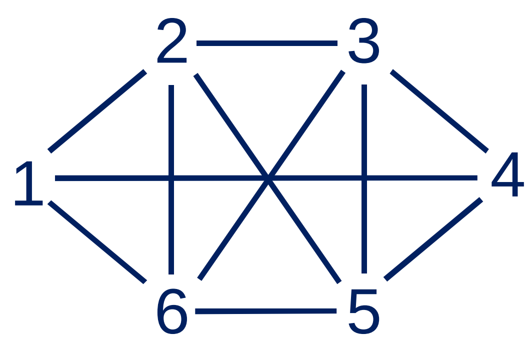
To build BDD, associate **state** with each node





Width = 2

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



{123456}

x_1

x_2

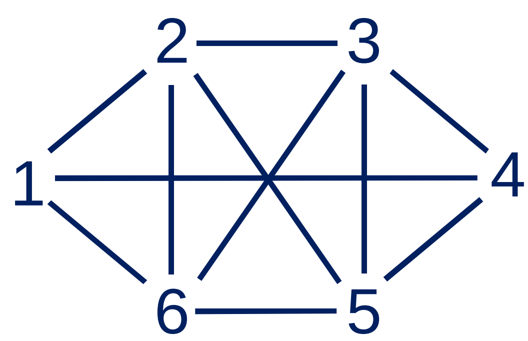
x_3

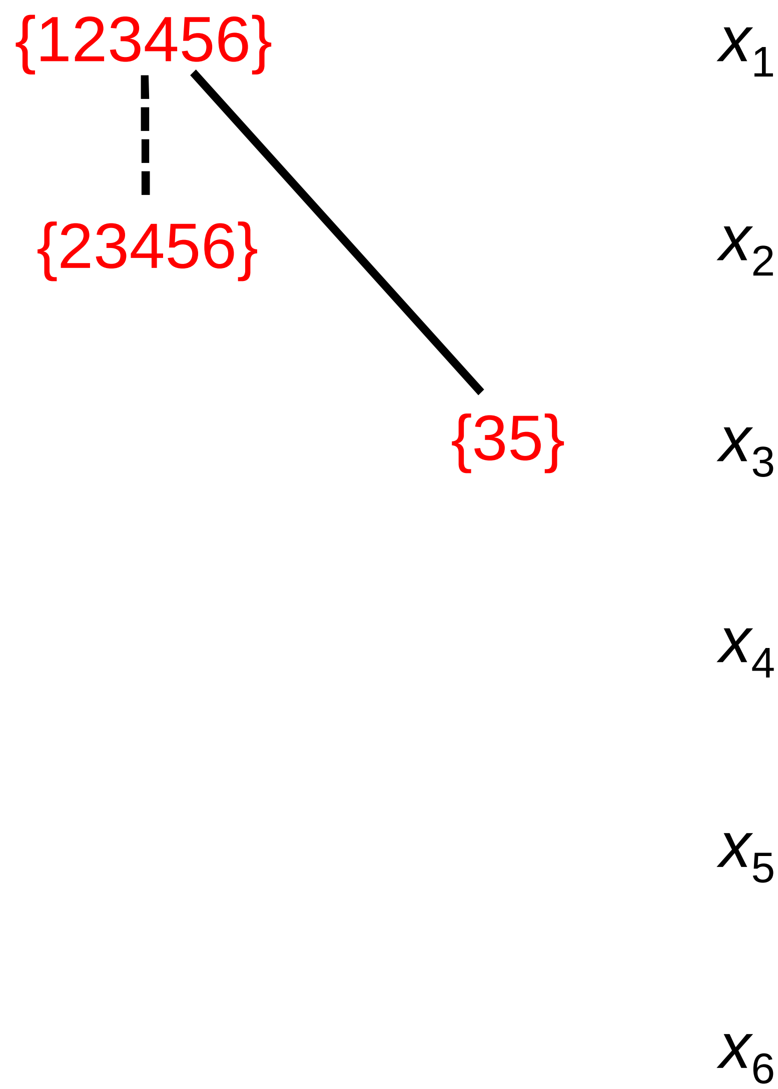
x_4

x_5

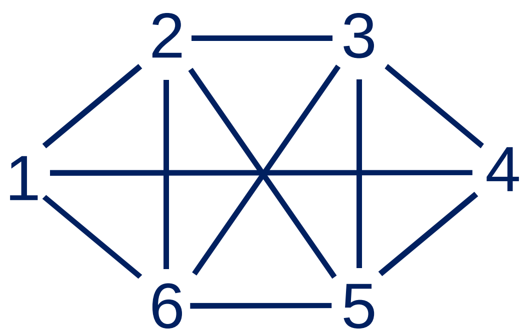
x_6

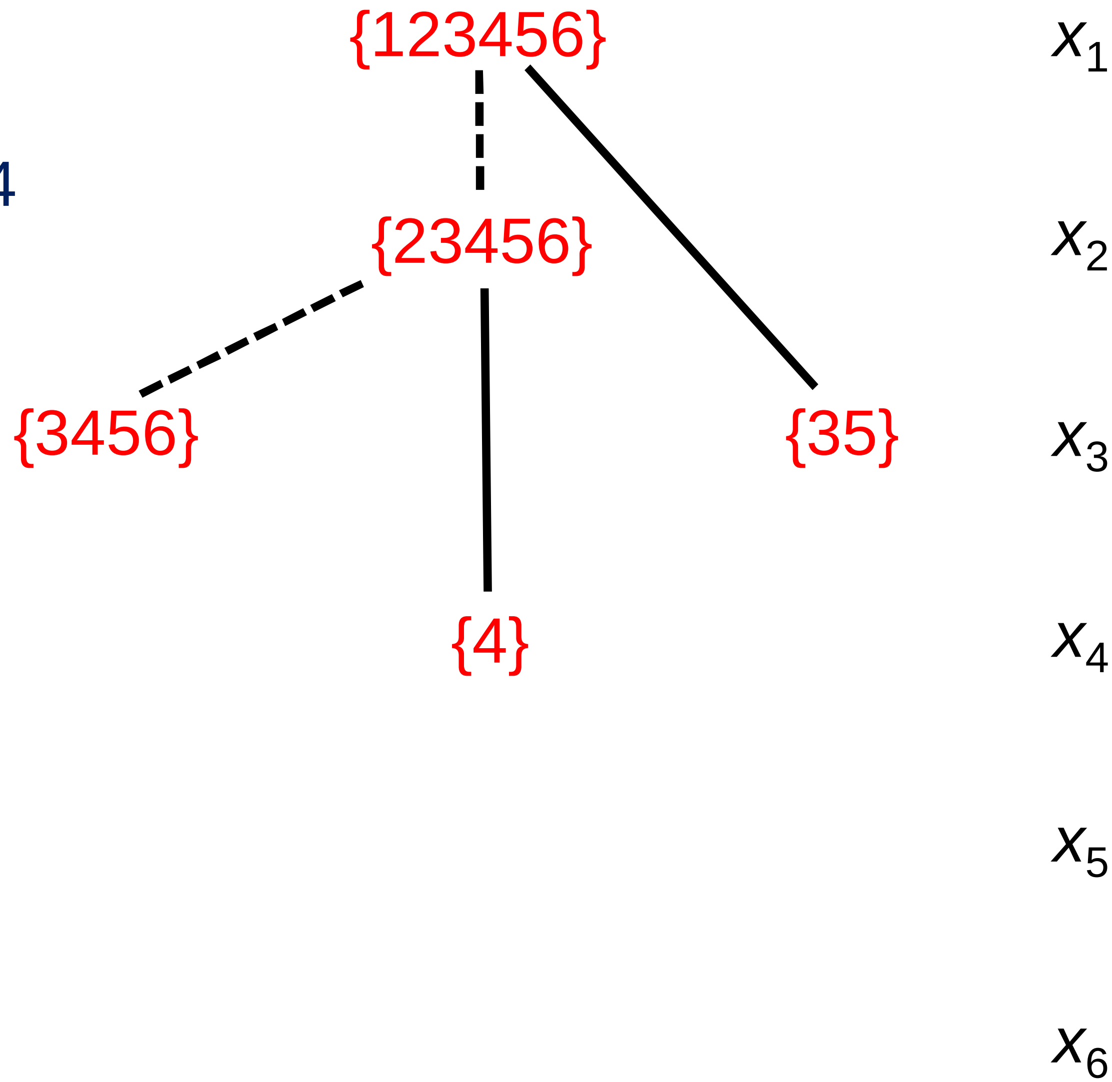
To build **relaxed** BDD, merge some nodes as we go along



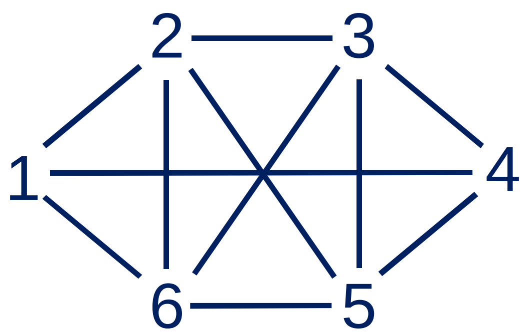


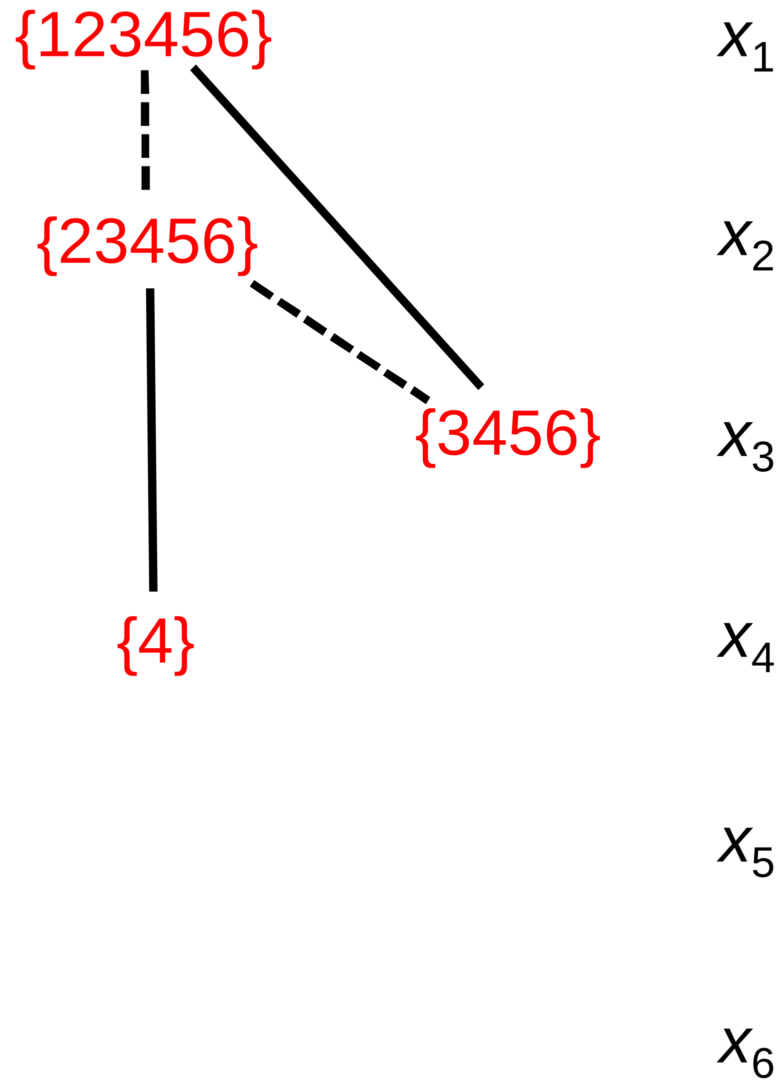
To build **relaxed** BDD, merge some nodes as we go along



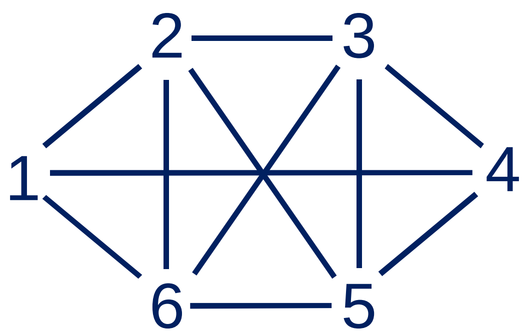


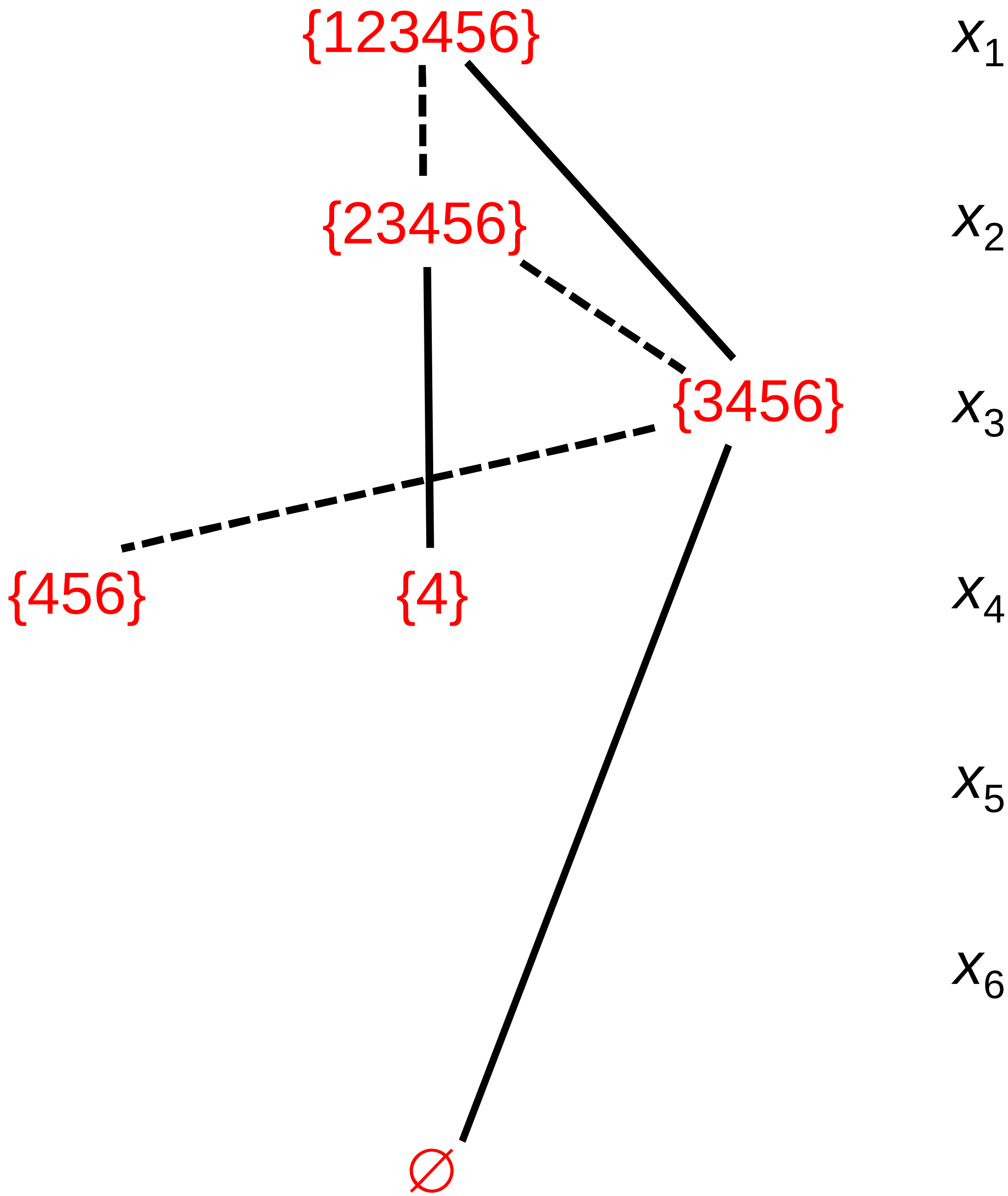
To build **relaxed** BDD, merge some nodes as we go along



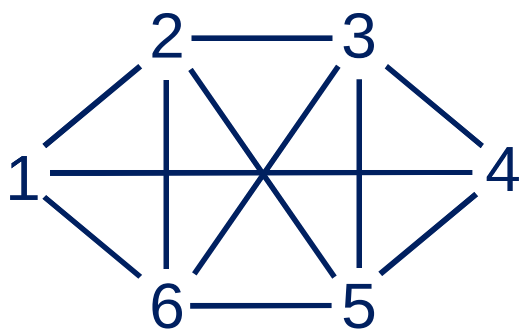


To build **relaxed** BDD, merge some nodes as we go along

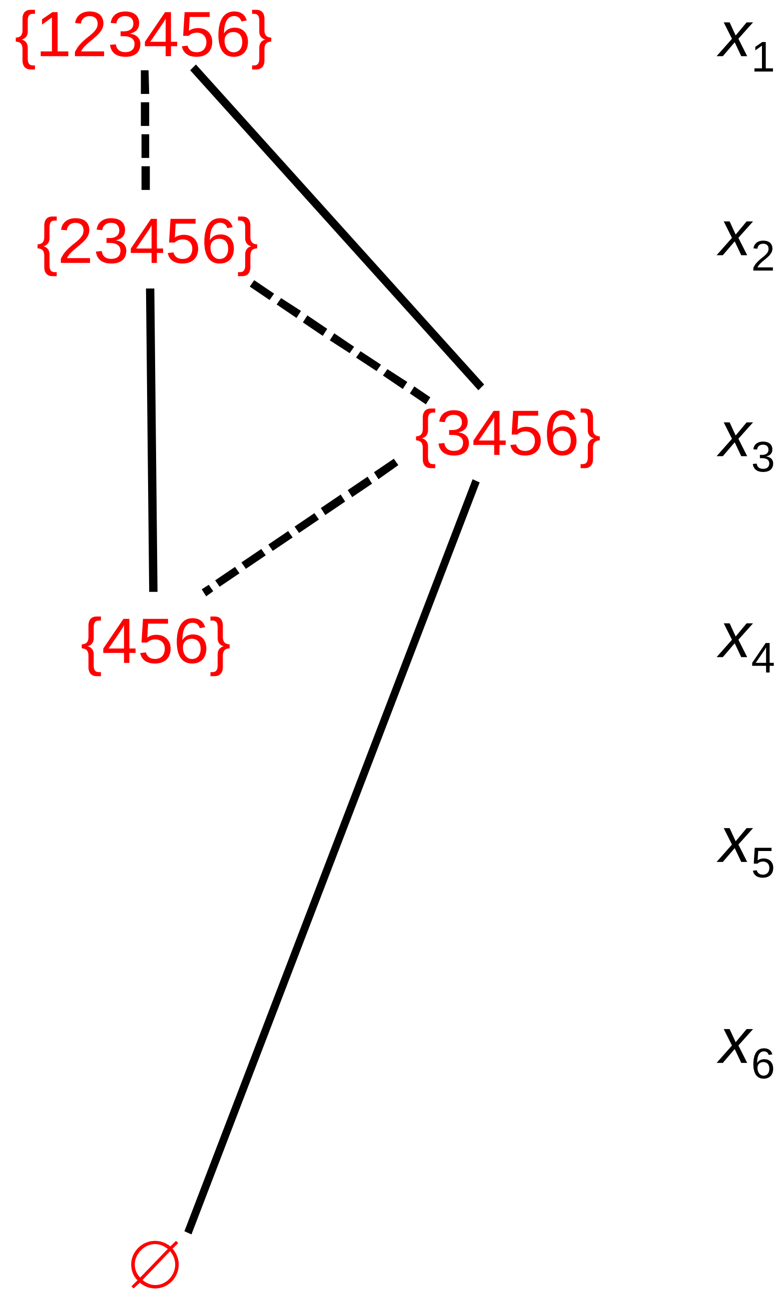


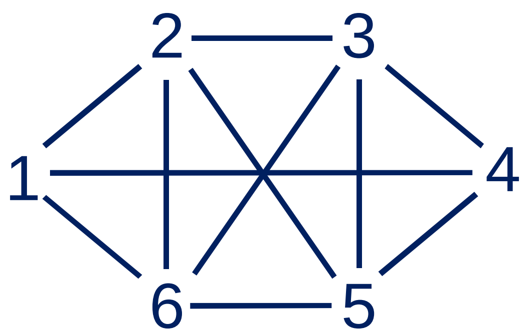


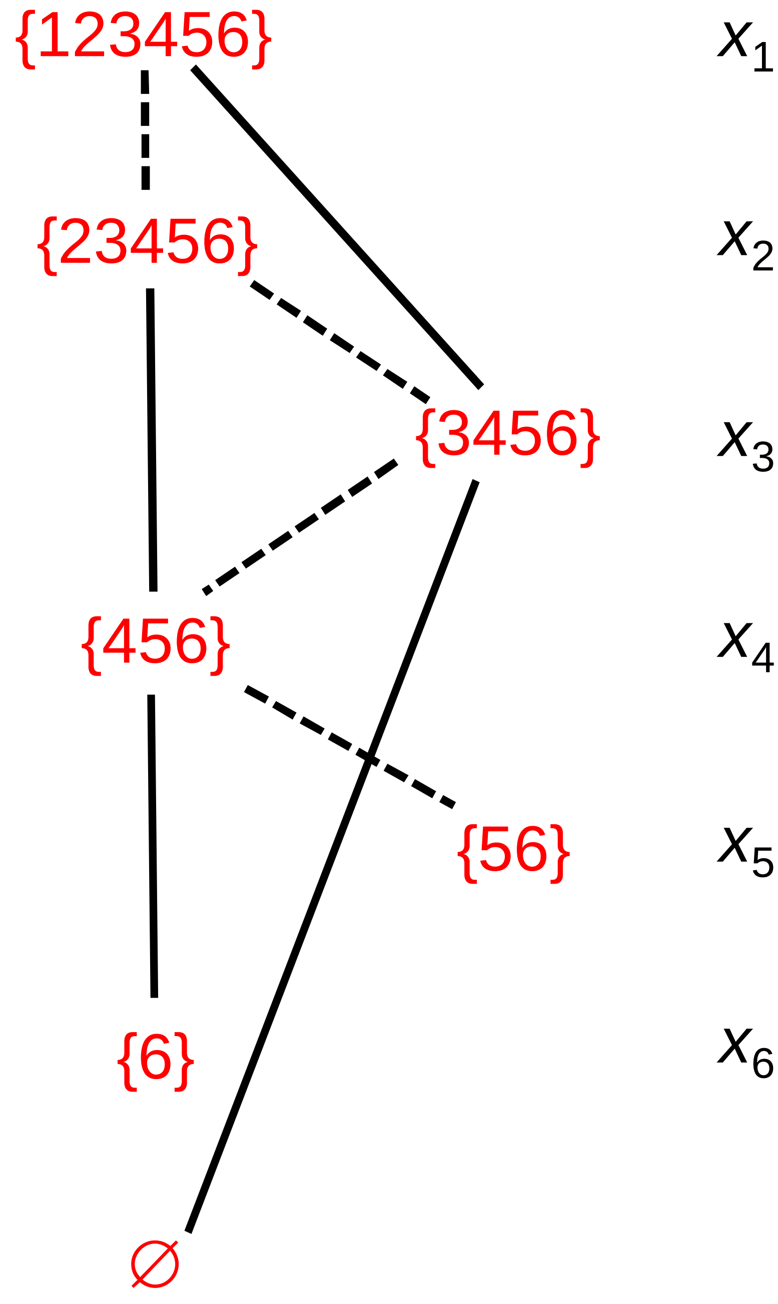
To build **relaxed** BDD, merge some nodes as we go along



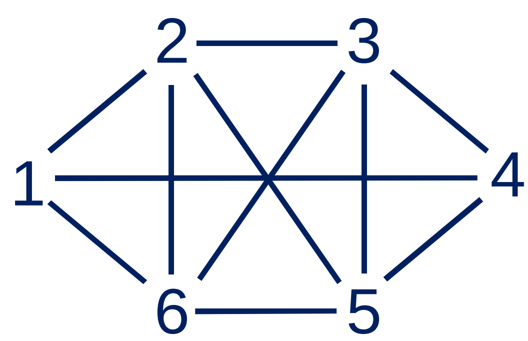
To build
relaxed
BDD, merge
some nodes
as we go
along

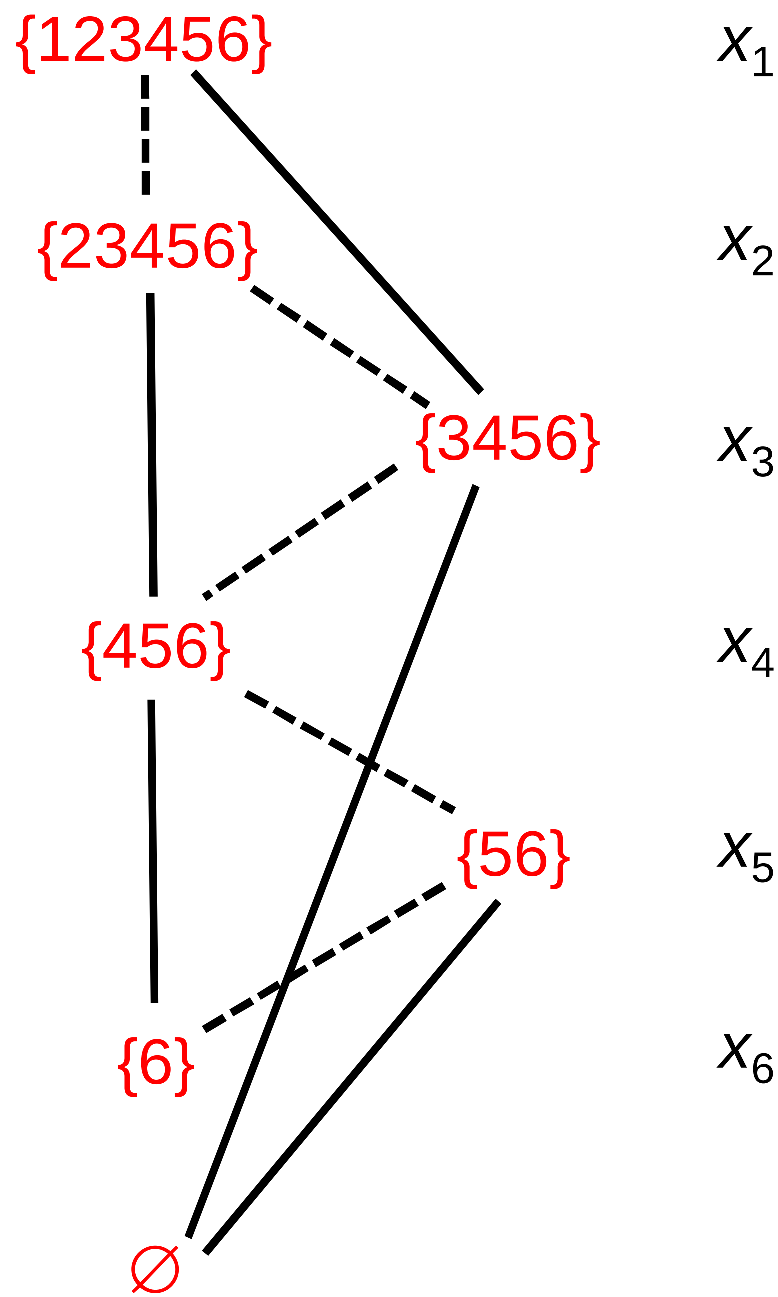




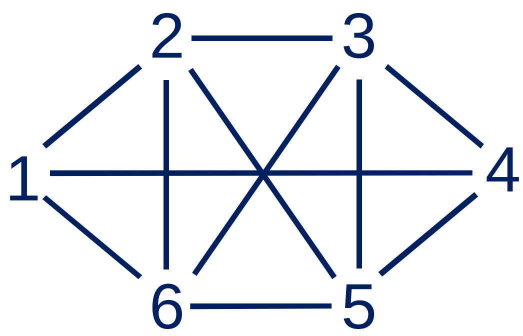


To build **relaxed** BDD, merge some nodes as we go along



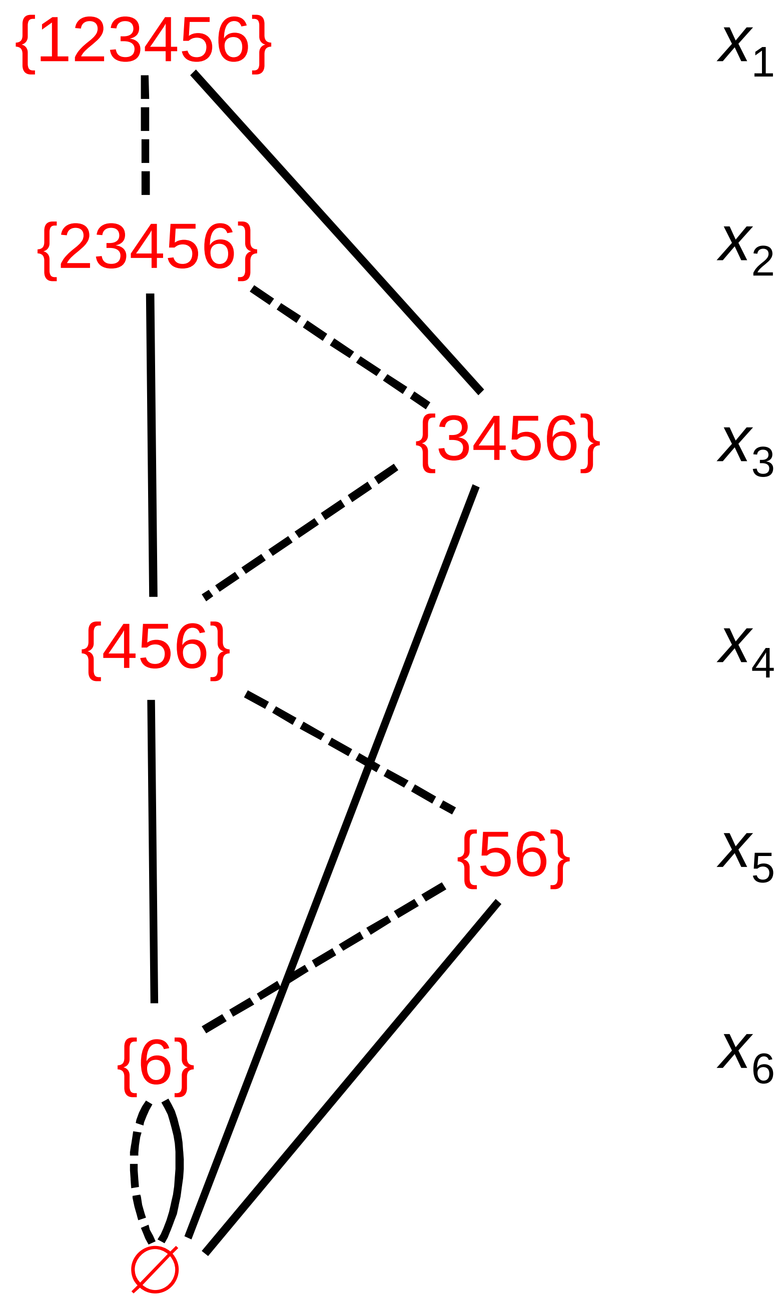


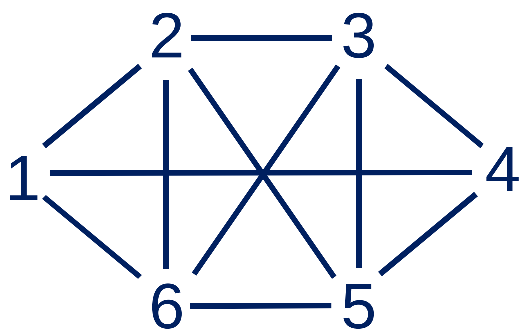
To build **relaxed** BDD, merge some nodes as we go along



Width = 1

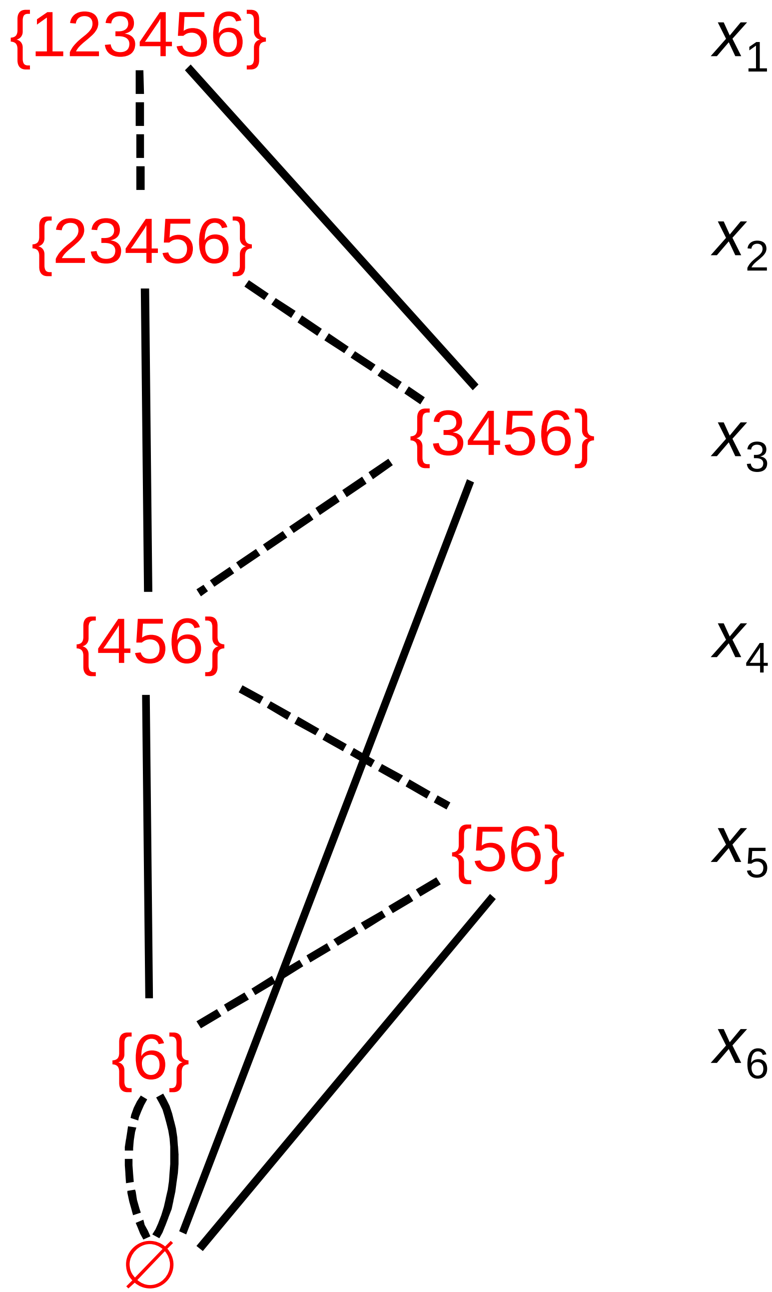
To build **relaxed** BDD, merge some nodes as we go along

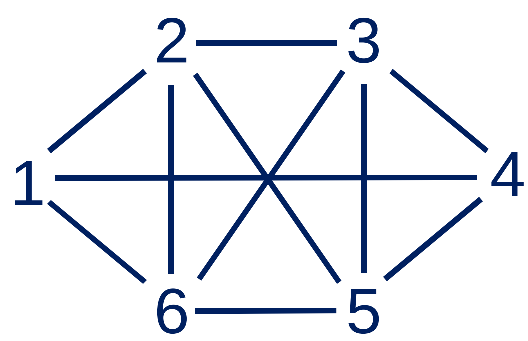




Width = 1

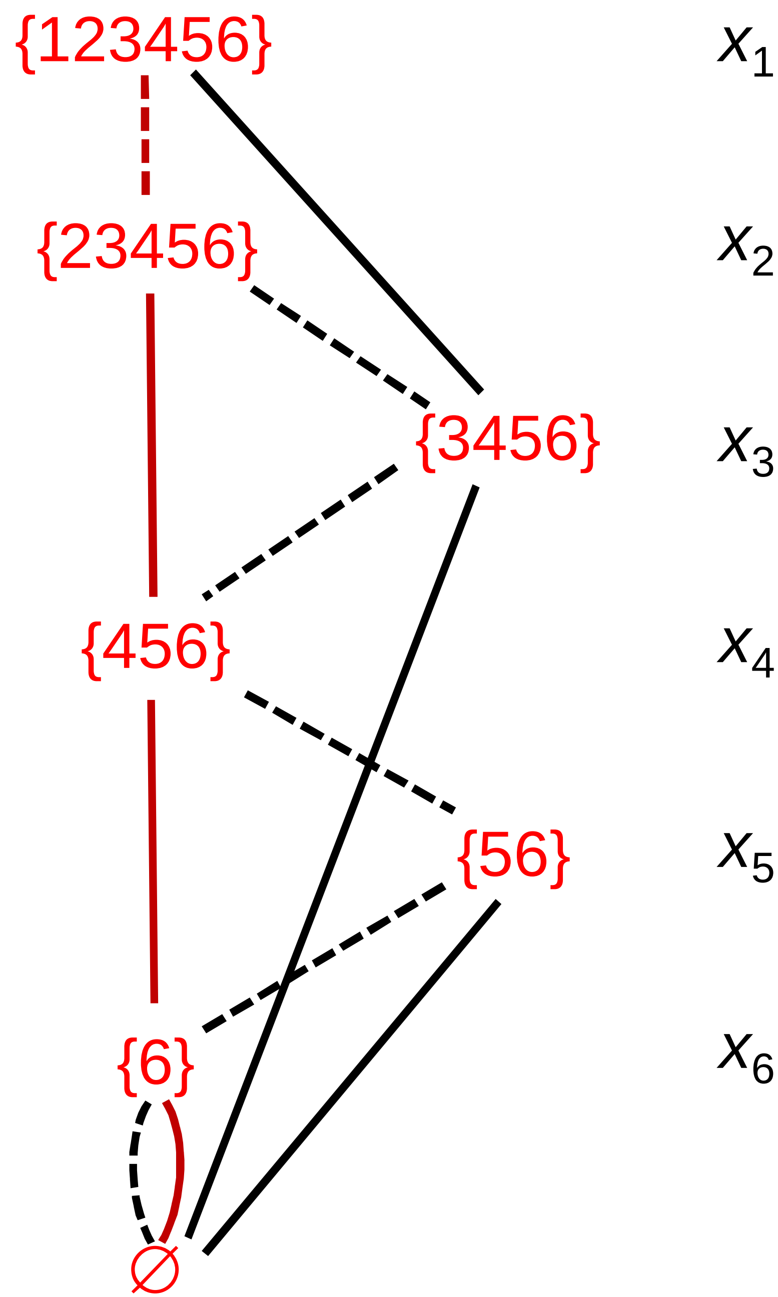
Represents 18 solutions, including 11 feasible solutions





Width = 1

Longest path gives bound of 3 on optimal value of 2



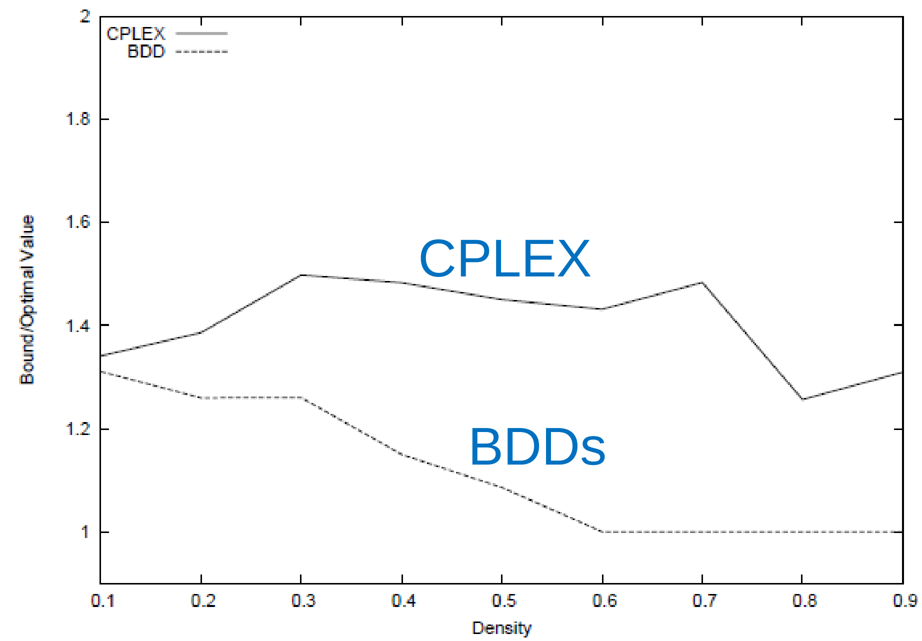
Comparison with LP Bound

- Random and benchmark instances
- Compare with LP bound at root node in CPLEX
  - Turn off presolve
    - We don’t use it.
    - It makes CPLEX bound worse or at most 1 better.
  - Use only clique cuts
    - Other cuts improve bound at most 1
    - And require orders of magnitude more time

S_2

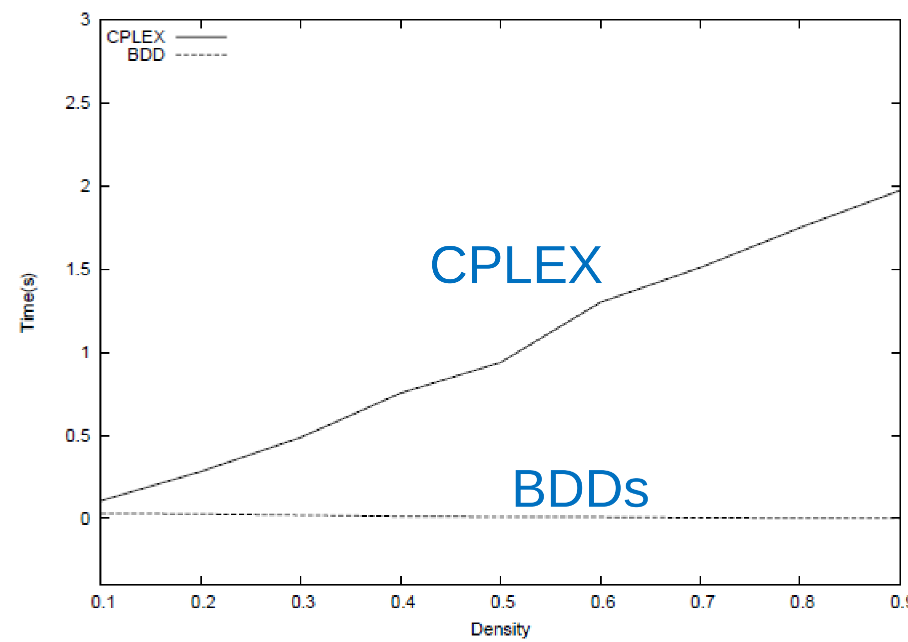
Random instances

Relative bound
vs. density



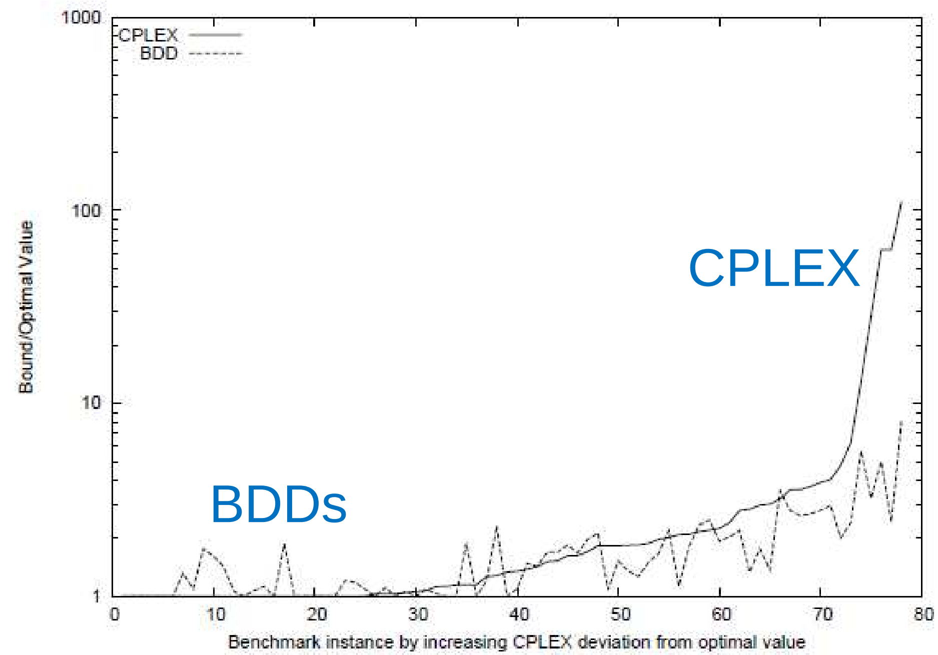
100 vertices
Max BDD width = 100

Time
vs. density



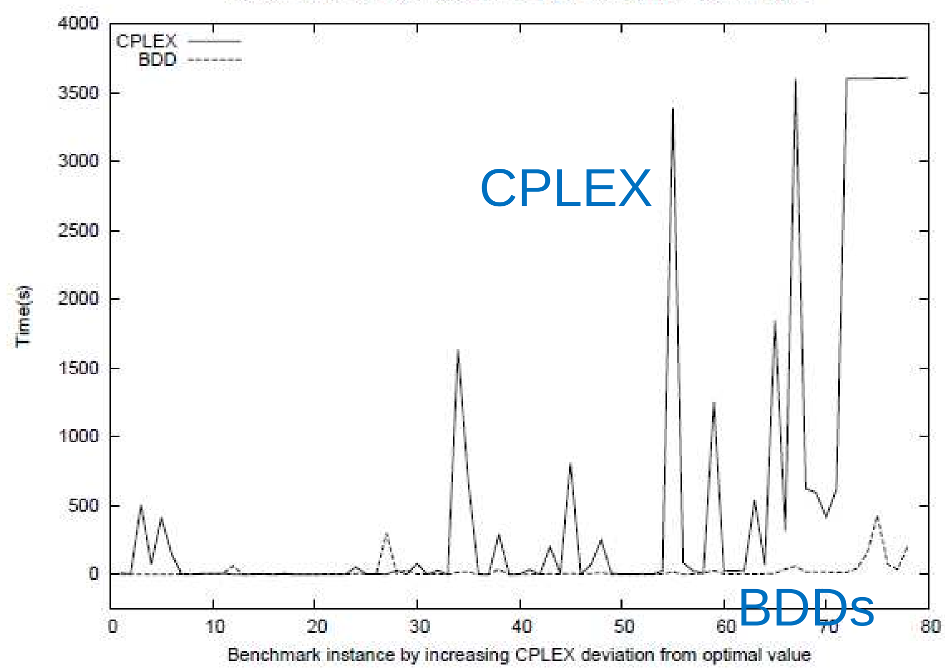
Benchmark instances
(DIMACS)

Relative bound



Max BDD width = 100

Time



Future Work

- More problems
  - Assembly line sequencing
  - Vehicle routing with time windows
- General BDD-based solver
  - Branch in the BDD
  - Combine with BDD-based propagation
  - BDD-based bounds, primal heuristic
    - No LP relaxation, cutting planes
  - Linearity, convexity irrelevant
    - But we need separable objective function