How to Relax

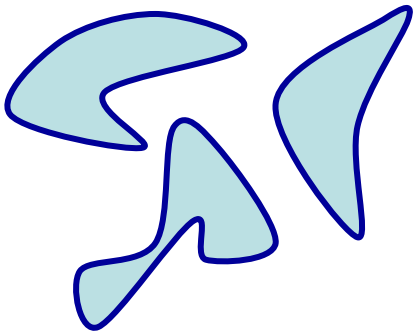
John Hooker
Carnegie Mellon University

September 2008

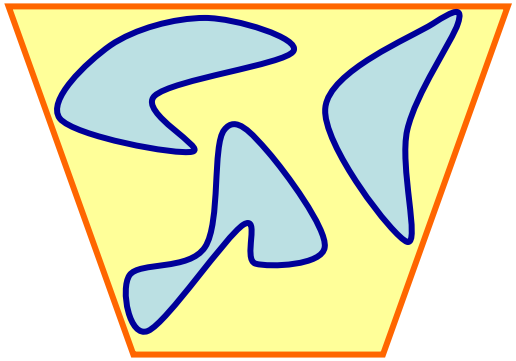
Two ways to relax

- Relax your mind and body.
- Relax your problem formulations.

Relaxing a problem



Feasible set of original problem



Feasible set of relaxed problem

Drop constraints to make the problem **easier to solve.**

Popular Relaxations

- Continuous relaxations
  - Linear relaxations are most popular.
  - Highly developed solution technology.

- Discrete relaxations
  - Already used in CP/AI, perhaps under other names.
  - For example, domain store.

Why relax a problem?

- Solution of relaxed problem **may solve original problem**.

Why relax a problem?

- Solution of relaxed problem **may solve original problem**.
- It may **guide the search** in a promising direction.

Why relax a problem?

- Solution of relaxed problem **may solve original problem**.
- It may **guide the search** in a promising direction.
- It may prune the search tree by **providing a bound** on the optimal value.

Why relax a problem?

- Solution of relaxed problem **may solve original problem**.
- It may **guide the search** in a promising direction.
- It may prune the search tree by **providing a bound** on the optimal value.
- It may help **filter domains** (e.g., with Lagrange multipliers).

Why relax a problem?

- Solution of relaxed problem **may solve original problem**.
- It may **guide the search** in a promising direction.
- It may prune the search tree by **providing a bound** on the optimal value.
- It may help **filter domains** (e.g., with Lagrange multipliers).
- It can provide a **more global view** of the problem, because a single relaxation can pool relaxations of several constraints.

When is Relaxation Useful?

- Relaxation already plays a central role in **constraint programming**.
  - The **domain store** is a relaxation.
  - Relaxations used in many other contexts.
  - It may pay to think about how to **strengthen** the relaxation.

When is Relaxation Useful?

- Relaxation is the workhorse of **optimization**.
  - Without it, problems would be intractable.
  - Particularly in **mathematical programming**.

When is Relaxation Useful?

- Relaxation is the workhorse of **optimization**.
  - Without it, problems would be intractable.
  - Particularly in **mathematical programming**.

- But it is **not** the mere existence of an **objective function** that makes relaxation important.

When is Relaxation Useful?

- Relaxation is the workhorse of **optimization**.
  - Without it, problems would be intractable.
  - Particularly in **mathematical programming**.
- But it is **not** the mere existence of an **objective function** that makes relaxation important.
- Often, it is the existence of constraints with **many variables**.
  - These constraints **do not propagate** well.
  - Such as **cost** constraints.
  - Relaxation is particularly valuable for such constraints.

Outline

- Continuous relaxations
  - Based on mixed integer modeling.
  - Based on analysis of global constraints.
- Relaxation duality
  - To strengthen continuous or discrete relaxations.
  - Example: Lagrangean duality
- Discrete relaxations.
  - Based on the dependency graph.
  - Based on multivalued decision diagrams.

Outline

- Continuous relaxations
  - Based on mixed integer modeling.
  - Based on analysis of global constraints.
- Relaxation duality
  - To strengthen continuous or discrete relaxations.
  - Example: Lagrangean duality
- Discrete relaxations.
  - Based on the dependency graph.
  - Based on multivalued decision diagrams.

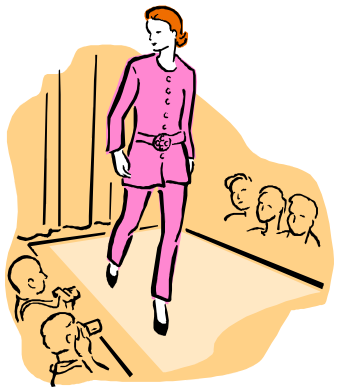
- Along the way…
  - Destinations where you can relax mind and body

Relax…

This is a survey. There is no need to follow everything in detail.

Great Barrier Reef, Australia

Cairns, Australia



Continuous Relaxation: Mixed Integer Modeling

Mixed Integer Models

Example: Facility Location

Example: Cumulative Scheduling

Cutting Planes

Mixed Integer Models

A **mixed integer/linear model** has the form

$$\min \; cx + dy$$
$$Ax + by \geq b$$
$$x, y \geq 0$$
$$y \text{ integer}$$

- First write a problem (or part of it) in this form.
- Then drop the integrality constraint to get a **continuous relaxation**.

Mixed Integer Models

A **mixed integer/linear model** has the form

$$\min \; cx + dy$$
$$Ax + by \geq b$$
$$x, y \geq 0$$
$$y \text{ integer}$$

- First write a problem (or part of it) in this form.
- Then drop the integrality constraint to get a **continuous relaxation**.
- Strengthen the relaxation with **cutting planes**.

Mixed Integer Models

A **mixed integer/linear model** has the form

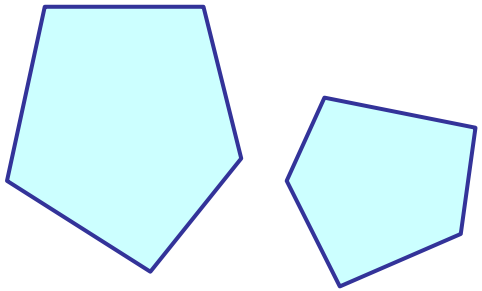
$$\min \; cx + dy$$
$$Ax + by \geq b$$
$$x, y \geq 0$$
$$y \text{ integer}$$

- First write a problem (or part of it) in this form.
- Then drop the integrality constraint to get a **continuous relaxation**.
- Strengthen the relaxation with **cutting planes**.

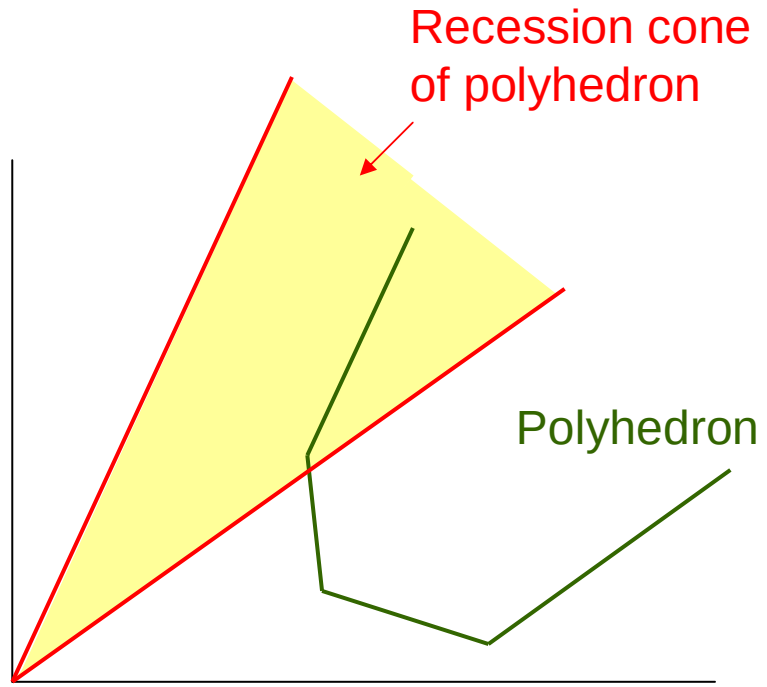
- Historical emphasis on inequality constraints is due to success of linear programming.

Mixed integer representability

Theorem. A problem can be given a mixed integer model if its feasible set is the union of finitely many **polyhedra** having the same recession cone.



Union of polyhedra with the same recession cone (in this case, the origin)



Modeling a union of polyhedra

Start with a disjunction of linear systems to represent the union of polyhedra.

The kth polyhedron is $\{ x \mid A^k x \geq b^k \}$

$$\min \; cx$$

$$\bigvee_k \left(A^k x \geq b^k \right)$$

Introduce a 0-1 variable y_k that is 1 when x is in polyhedron k.

Disaggregate x to create an x^k for each k.

$$\min \; cx$$

$$A^k x^k \geq b^k y_k, \text{ all } k$$

$$\sum_k y_k = 1$$

$$x = \sum_k x^k$$

$$y_k \in \{0,1\}$$

Convex hull relaxation

Good news: continuous relaxation of this model is a **convex hull relaxation** (tightest linear relaxation).

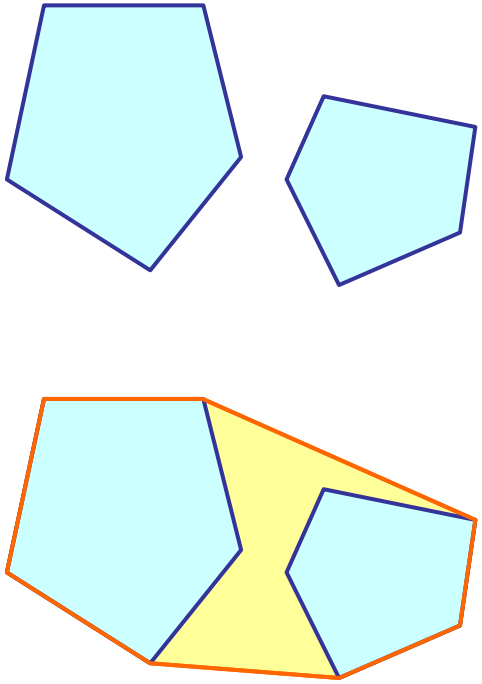
$$\min \ cx$$

$$A^k x^k \geq b^k y_k, \text{ all } k$$

$$\sum_k y_k = 1$$

$$x = \sum_k x^k$$

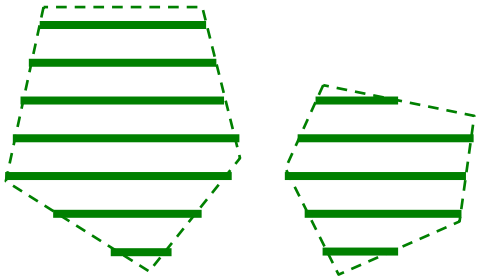
$$0 \leq y_k \leq 1$$



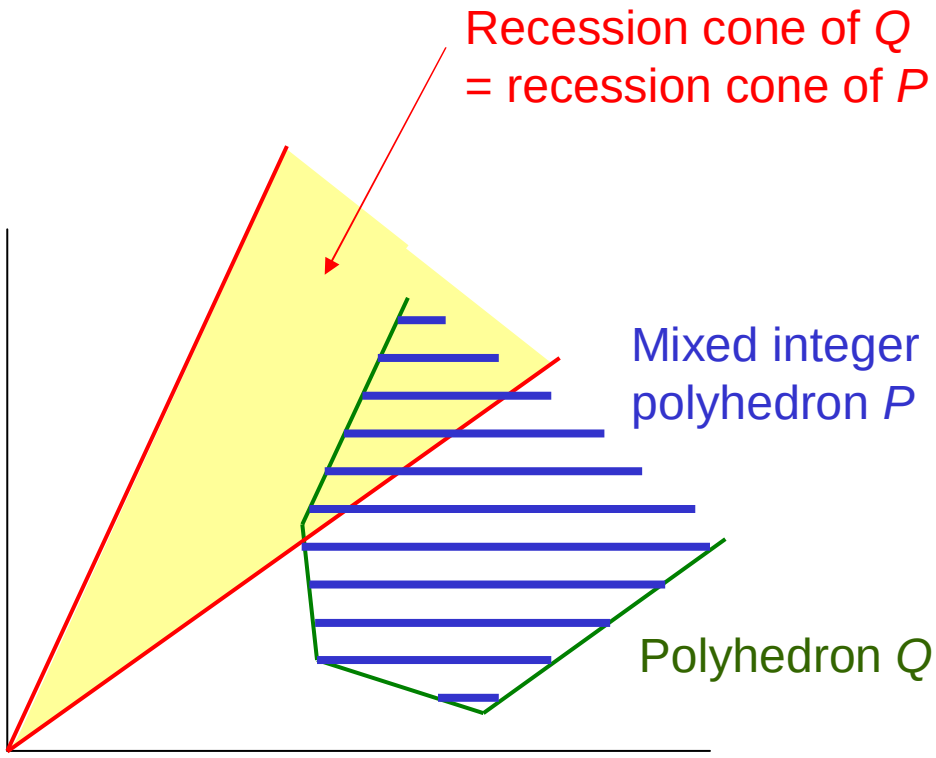
Convex hull

General representability theorem

Theorem. A problem can be given a mixed integer model **if and only if** it is the union of finitely many **mixed integer polyhedra** having the same recession cone.



Union of mixed integer polyhedra with the same recession cone (in this case, the origin)



JNH, A principled approach to mixed integer problem formulation, 2008

Convex hull relaxation

More good news: continuous relaxation of this model is a **convex hull relaxation**, if the polyhedra have convex hull models.

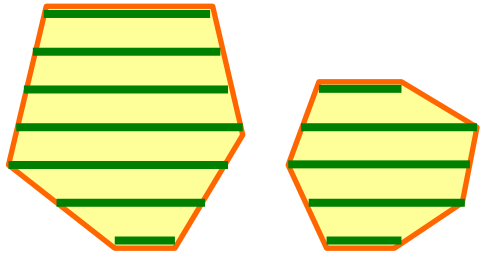
$$A^k x^k \geq b^k y_k, \text{ all } k$$

$$\sum_k y_k = 1$$

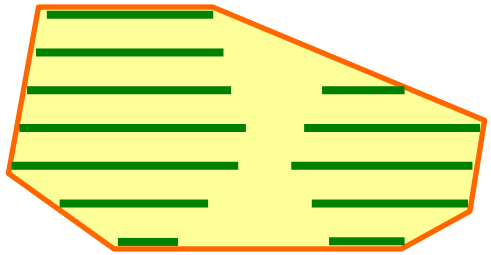
$$x = \sum_k x^k$$

$$x \in \mathbb{R}^n \times \mathbb{Z}^p$$

$$0 \leq y_k \leq 1$$

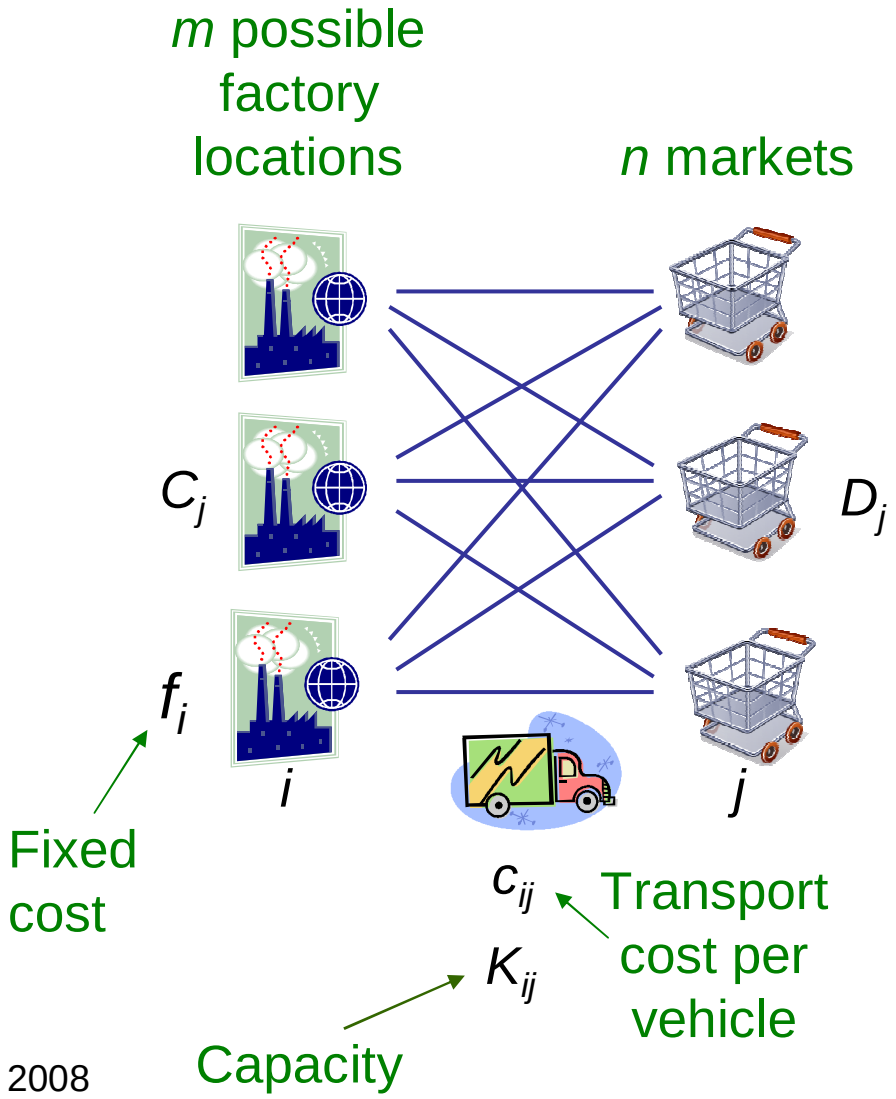


Union of mixed integer polyhedra with convex hull descriptions



Convex hull relaxation

Example: Facility location



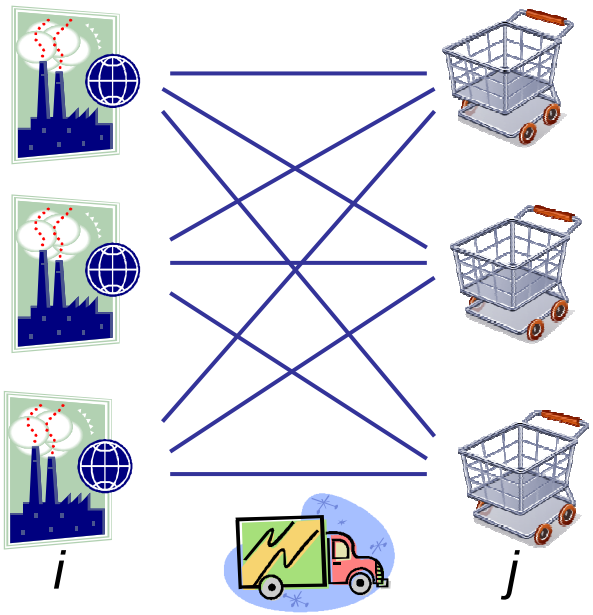
Locate factories to serve markets so as to minimize total factory cost and transport cost.

Fixed cost incurred for each vehicle used.

Facility location

m possible factory locations

n markets



Fixed cost

c_{ij} Transport cost per vehicle

K_{ij} Capacity

Disjunctive model:

$$\min \sum_i z_i + \sum_{ij} c_{ij} w_{ij}$$

$$\left[\begin{array}{c} \sum_j x_{ij} \le C_i \\ 0 \le x_{ij} \le K_{ij} w_{ij}, \text{ all } j \\ z_i = f \\ w_{ij} \in \mathbb{Z}, \text{ all } j \end{array}\right] \vee \left[\begin{array}{c} x_{ij} = 0, \text{ all } j \\ z_i = 0 \end{array}\right], \quad \text{all } i$$

$$\sum_i x_{ij} = D_j, \quad \text{all } j$$

Flow on route (*i*,*j*)

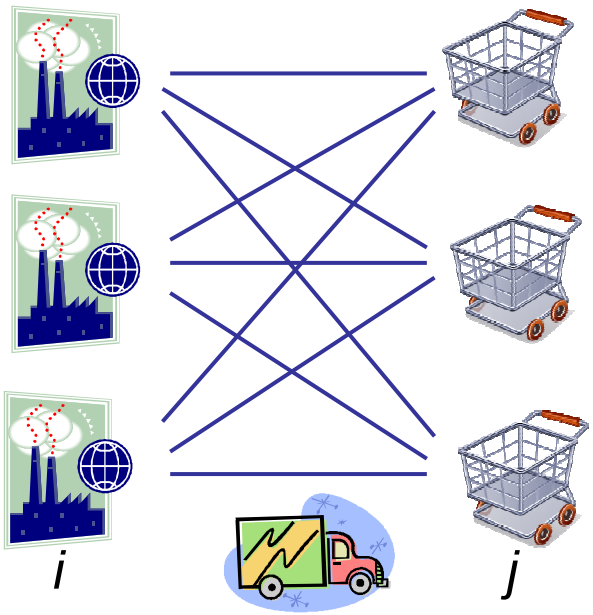
Factory at location *i*

No factory at location *i*

Facility location

m possible factory locations

n markets



C_j

f_i

i

j

Fixed cost

c_{ij} Transport cost per vehicle

K_{ij} Capacity

Number of vehicles from factory *i* to market *j*

Disjunctive model:

$$\min \sum_i z_i + \sum_{ij} c_{ij} w_{ij}$$

$$\begin{pmatrix} \sum_j x_{ij} \le C_i \\ 0 \le x_{ij} \le K_{ij} w_{ij}, \text{ all } j \\ z_i = f \\ w_{ij} \in \mathbb{Z}, \text{ all } j \end{pmatrix} \vee \begin{pmatrix} x_{ij} = 0, \text{ all } j \\ z_i = 0 \end{pmatrix}, \quad \text{all } i$$

$$\sum_i x_{ij} = D_j, \quad \text{all } j$$

Flow on route (*i*,*j*)

Factory at location *i*

No factory at location *i*

Facility location

Problem: Disjuncts don't have same recession cone

$$\min \sum_i z_i + \sum_{ij} c_{ij} w_{ij}$$

$$\begin{bmatrix} \sum_j x_{ij} \le C_i \\ 0 \le x_{ij} \le K_{ij} w_{ij}, \text{ all } j \\ z_i = f \\ w_{ij} \in \mathbb{Z}, \text{ all } j \end{bmatrix} \vee \begin{bmatrix} x_{ij} = 0, \text{ all } j \\ z_i = 0 \end{bmatrix}, \quad \text{all } i$$

$$\sum_i x_{ij} = D_j, \quad \text{all } j$$

Recession directions:

x_{ij}, z_i bounded

$w_{ij} \to +\infty$

x_{ij}, z_i bounded

$w_{ij} \to \pm\infty$

Facility location

Solution: Add innocuous bounds

$$\min \sum_i z_i + \sum_{ij} c_{ij} w_{ij}$$

$$\left(\begin{array}{c} \sum_j x_{ij} \le C_i \\ 0 \le x_{ij} \le K_{ij} w_{ij}, \text{ all } j \\ z_i = f \\ w_{ij} \in \mathbb{Z}, \text{ all } j \end{array} \right) \vee \left(\begin{array}{c} x_{ij} = 0, \text{ all } j \\ z_i = 0 \\ \boxed{w_{ij} \ge 0} \end{array} \right), \quad \text{all } i$$

$$\sum_i x_{ij} = D_j, \quad \text{all } j$$

Recession directions:

x_{ij}, z_i bounded

$w_{ij} \to +\infty$

x_{ij}, z_i bounded

$w_{ij} \to +\infty$

Facility location

$$\min \sum_i z_i + \sum_{ij} c_{ij} w_{ij}$$

Disjunctive model:

$$\begin{bmatrix} \sum_j x_{ij} \le C_i \\ 0 \le x_{ij} \le K_{ij} w_{ij}, \text{ all } j \\ z_i = f \\ w_{ij} \in \mathbb{Z}, \text{ all } j \end{bmatrix} \vee \begin{bmatrix} x_{ij} = 0, \text{ all } j \\ z_i = 0 \\ w_{ij} \ge 0 \end{bmatrix}, \quad \text{all } i$$

$$\sum_i x_{ij} = D_j, \quad \text{all } j$$

Mixed integer formulation:

$$\min \sum_i f_i y_i + \sum_{ij} c_{ij} w_{ij}$$

$$\sum_j x_{ij} \le C_i y_i, \text{ all } i$$

$$\sum_i x_{ij} = D_j, \text{ all } j$$

$$0 \le x_{ij} \le K_{ij} w_{ij}, \text{ all } i, j$$

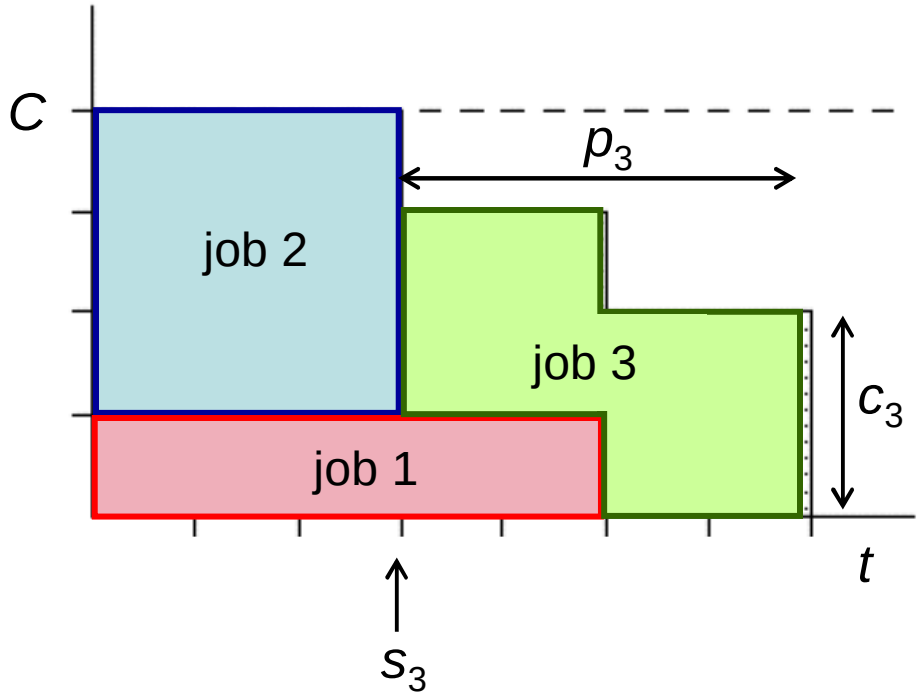
$$y_i \in \{0,1\}, \quad w_{ij} \in \mathbb{Z}, \text{ all } i, j$$

Example: Cumulative scheduling

Cumulative scheduling constraint:

$$\text{cumulative}\big((s_1,s_2,s_3),(p_1,p_2,p_3),(c_1,c_2,c_3),C\big)$$

A feasible solution:



Cumulative scheduling

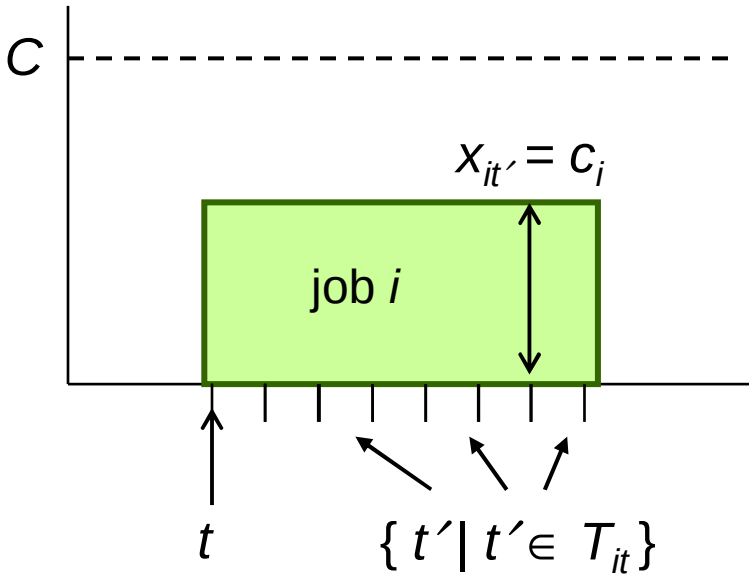
Job *i* must start at some time *t*

Disjunctive formulation:

$$\bigvee_t \begin{pmatrix} s_i = t \\ x_{it'} = c_i,\ t' \in T_{it} \end{pmatrix}, \quad \text{all } i$$

$$\sum_i \sum_{t:\, t' \in T_{it}} x_{it'} \le C, \ \text{all } t'$$

Set of times job *i* is running if it starts at *t*

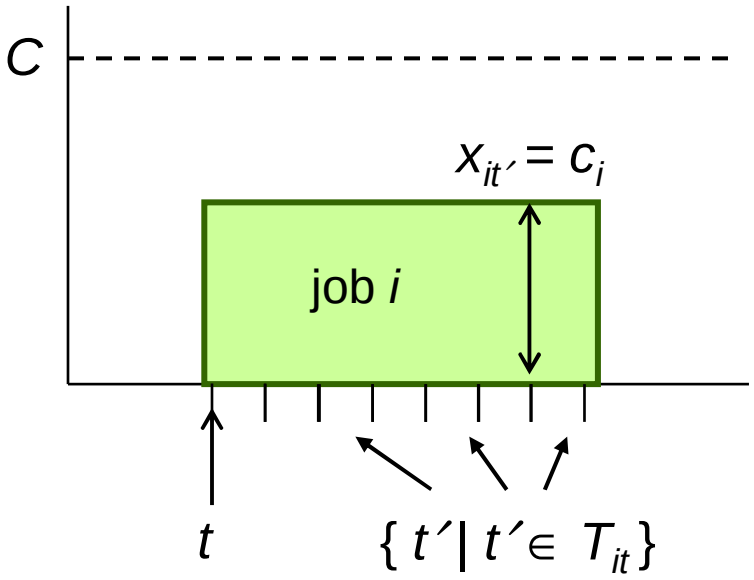


Cumulative scheduling

Disjunctive formulation:

$$\bigvee_t \begin{pmatrix} s_i = t \\ x_{it'} = c_i,\ t' \in T_{it} \end{pmatrix}, \quad \text{all } i$$

$$\sum_i \sum_{t:\, t' \in T_{it}} x_{it'} \le C, \quad \text{all } t'$$



Mixed integer model:

$$s_i^t = t y_{it}, \quad \text{all } i,t$$

$$x_{it'}^t = c_i y_{it}, \quad t' \in T_t, \text{ all } i,t$$

$$\sum_i \sum_{t:\, t' \in T_{it}} x_{it'} \le C, \quad \text{all } t'$$

$$s_i = \sum_t s_i^t, \quad \text{all } i$$

$$x_{it'} = \sum_t x_{it'}^t$$

$$\sum_t y_{it} = 1, \quad \text{all } i$$

$$y_{it} \in \{0,1\}, \quad \text{all } i,t$$

Cumulative scheduling

Disjunctive formulation:

$$\bigvee_t \begin{pmatrix} s_i = t \\ x_{it'} = c_i, \ t' \in T_{it} \end{pmatrix}, \ \text{all } i$$

$$\sum_i \sum_{t: t' \in T_{it}} x_{it'} \le C, \ \text{all } t'$$

Mixed integer model:

$$s_i = \sum_t t y_{it}, \ \text{all } i$$

$$\sum_i \sum_{t: t' \in T_{it}} c_i y_{it} \le C, \ \text{all } t'$$

$$\sum_t y_{it} = 1, \ \text{all } i$$

$$y_{it} \in \{0,1\}, \ \text{all } i,t$$

simplifies

$$s_i^t = t y_{it}, \ \text{all } i,t$$

$$x_{it'}^t = c_i y_{it}, \ t' \in T_t, \ \text{all } i,t$$

$$\sum_i \sum_{t: t' \in T_{it}} x_{it'} \le C, \ \text{all } t'$$

$$s_i = \sum_t s_i^t, \ \text{all } i$$

$$x_{it'} = \sum_t x_{it'}^t$$

$$\sum_t y_{it} = 1, \ \text{all } i$$

$$y_{it} \in \{0,1\}, \ \text{all } i,t$$

Cutting Planes

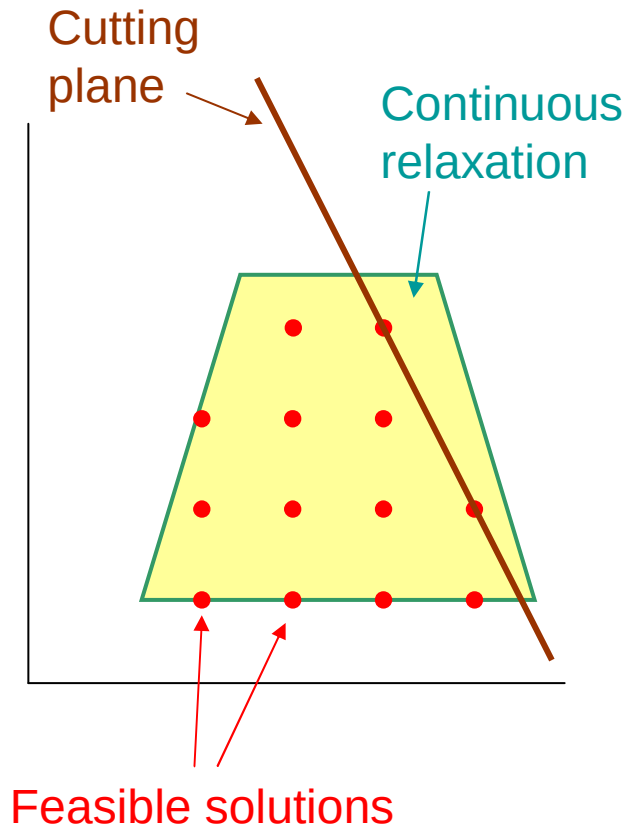
A **cutting plane** (cut, valid inequality) for a mixed integer model:

- …is **valid**

  - It is satisfied by all feasible solutions of the model.

- …**cuts off** solutions of the continuous relaxation.

  - This makes the relaxation tighter.



Some popular cutting planes

- **Knapsack cuts**
  - Generated for individual inequality constraints.
  - Sequential **lifting**.
  - **Sequence-independent** lifting.
- **Rounding cuts**
  - Generated for the entire model, they are widely used.
  - **Gomory cuts** for integer variables only.
  - **Mixed integer rounding cuts** for mixed integer models.
- **Special-purpose cuts**
  - For many problems.

Les Cevénnes, France

CP 2008
Slide 40

Le fromage roquefort, France

Continuous Relaxation for Global Constraints

Element

Alldiff

Circuit

Cumulative scheduling

Element

The element constraint implements a variable indexed by a variable: x_y

Example: Channeling constraints for employee scheduling

$$x_{y_k} = k$$

$$y_{x_i} = i$$

Element

The element constraint implements a variable indexed by a variable: x_y

Example: Channeling constraints for employee scheduling

$$x_{y_k} = k$$

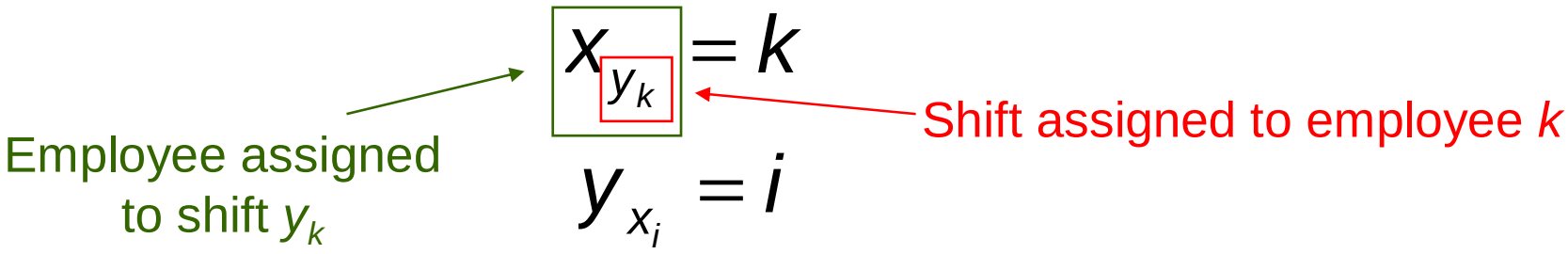
Shift assigned to employee *k*

$$y_{x_i} = i$$

Element

The element constraint implements a variable indexed by a variable: x_y

Example: Channeling constraints for employee scheduling



Element

The element constraint implements a variable indexed by a variable: x_y

Example: Channeling constraints for employee scheduling

$$x_{y_k} = k$$

$$y_{x_i} = i$$

Employee assigned to shift *i*

Element

The element constraint implements a variable indexed by a variable: x_y

Example: Channeling constraints for employee scheduling

$$x_{y_k} = k$$

$$y_{x_i} = i$$

Shift assigned to employee x_i

Employee assigned to shift i

Element

The element constraint implements a variable indexed by a variable: x_y

To implement x_y:

Replace x_y with z and add the constraint

$$\text{element}(y,(x_1,\ldots,x_n),z)$$

Assume domain of y is {1, …, n}.

Element

The element constraint implements a variable indexed by a variable: x_y

To relax $\text{element}(y,(x_1,\ldots,x_n),z)$

View it as a disjunction $\bigvee_i (x_i = z)$

Element

The element constraint implements a variable indexed by a variable: x_y

To relax $\text{element}(y,(x_1,\ldots,x_n),z)$

View it as a disjunction $\bigvee_i (x_i = z)$

and write the convex hull relaxation $z = \sum_i x_i^i$

$$Ly_i \le x^i \le Uy_i, \quad \text{all } i$$

$$x_i = \sum_k x_i^k, \quad \text{all } i$$

$$\sum_i y_i = 1, \quad y_i \ge 0, \quad \text{all } i$$

Assume bounds $L \le x \le U$

JNH, *Logic-based Methods for Optimization*, 2000

Element

The element constraint implements a variable indexed by a variable: x_y

If each x_i has the same bounds $L_0 \leq x_i \leq U_0$

the convex hull relaxation simplifies to

$$\sum_i x_i - (n-1)U_0 \leq z \leq \sum_i x_i - (n-1)L_0$$

$$L_0 \leq x_i \leq U_0, \quad \text{all } i$$

JNH, *Logic-based Methods for Optimization*, 2000

Element

A specially structured element constraint implements $a_y x$

Example: Assignment cost

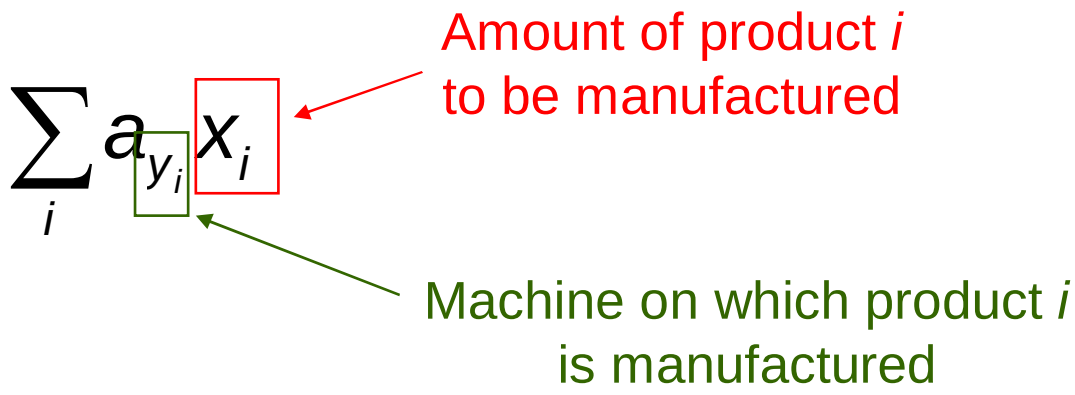
$$\sum_i a_{y_i} x_i$$

Amount of product *i* to be manufactured

Element

A specially structured element constraint implements $a_y x$

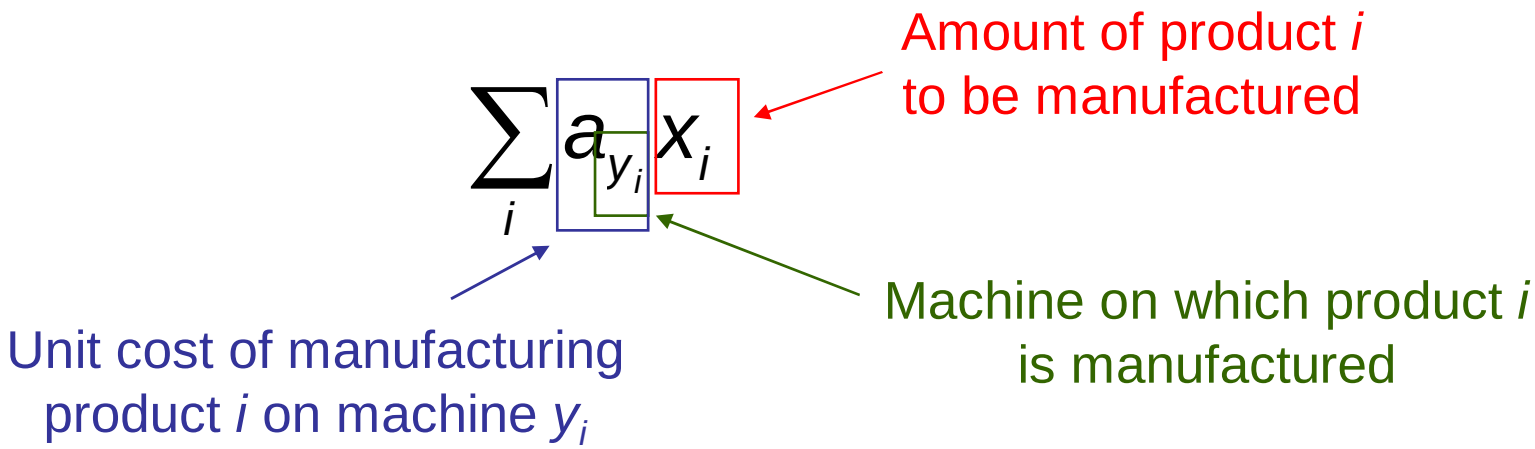
Example: Assignment cost



Element

A specially structured element constraint implements $a_y x$

Example: Assignment cost



Element

A specially structured element constraint implements $a_y x$

To implement $a_y x$:

Replace $a_y x$ with z and add the constraint

$$\text{element}(y,(a_1 x,\ldots,a_n x),z)$$

Element

A specially structured element constraint implements $a_y x$

To relax $\text{element}(y,(a_1 x,\ldots,a_n x),z)$

Element

A specially structured element constraint implements $a_y x$

To relax $\text{element}(y,(a_1 x,\ldots,a_n x),z)$

View it as a disjunction $\bigvee_i (a_i x = z)$

Element

A specially structured element constraint implements $a_y x$

To relax $\text{element}(y,(a_1x,\ldots,a_nx),z)$

View it as a disjunction $\bigvee_i (a_i x = z)$

and write the convex hull relaxation

$$\frac{U_{\min}-L_{\min}}{U-L}x+\frac{UL_{\min}-LU_{\min}}{U-L} \le z \le \frac{U_{\max}-L_{\max}}{U-L}x+\frac{UL_{\max}-LU_{\max}}{U-L}$$

where $L \le x \le U$, $L_{\min} = \min_i\{a_iL\}$ $U_{\min} = \min_i\{a_iU\}$

$L_{\max} = \max_i\{a_iL\}$ $U_{\max} = \max_i\{a_iU\}$

JNH, *Integrated Methods for Optimization*, 2007

Alldiff

The **alldiff polytope** (**permutation polytope**) is the convex hull of feasible solutions of alldiff.

The polytope is completely known.

The facet-defining inequalities provide a **convex hull relaxation**

Alldiff

The **alldiff polytope** (**permutation polytope**) is the convex hull of feasible solutions of alldiff.

The polytope is completely known.

The facet-defining inequalities provide a **convex hull relaxation**

For example, the convex hull of $\text{alldiff}(x_1, x_2, x_3)$

with domains $x_i \in \{1,3,6\}$ is completely described by

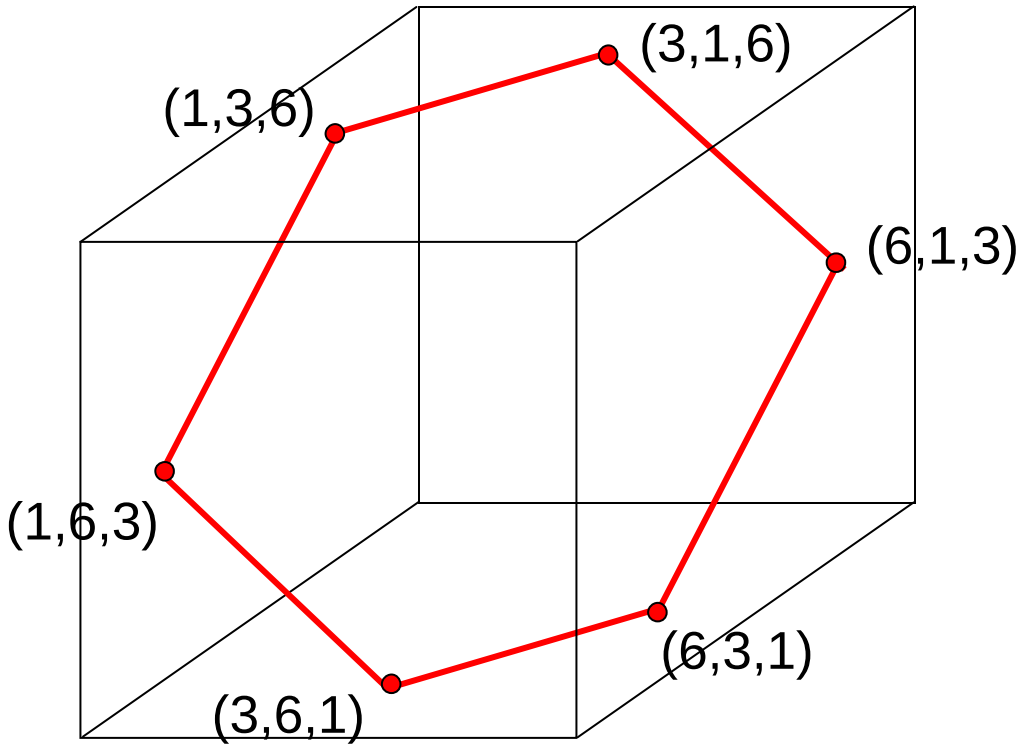
$$x_1 + x_2 + x_3 = 10$$
$$x_1 + x_2 \geq 4, \quad x_1 + x_3 \geq 4, \quad x_2 + x_3 \geq 4$$
$$x_1, x_2, x_3 \geq 1$$

JNH, *Logic-based Methods for Optimization*, 2000

Williams and Yan, Representations of the all-different predicate, 2001

Alldiff

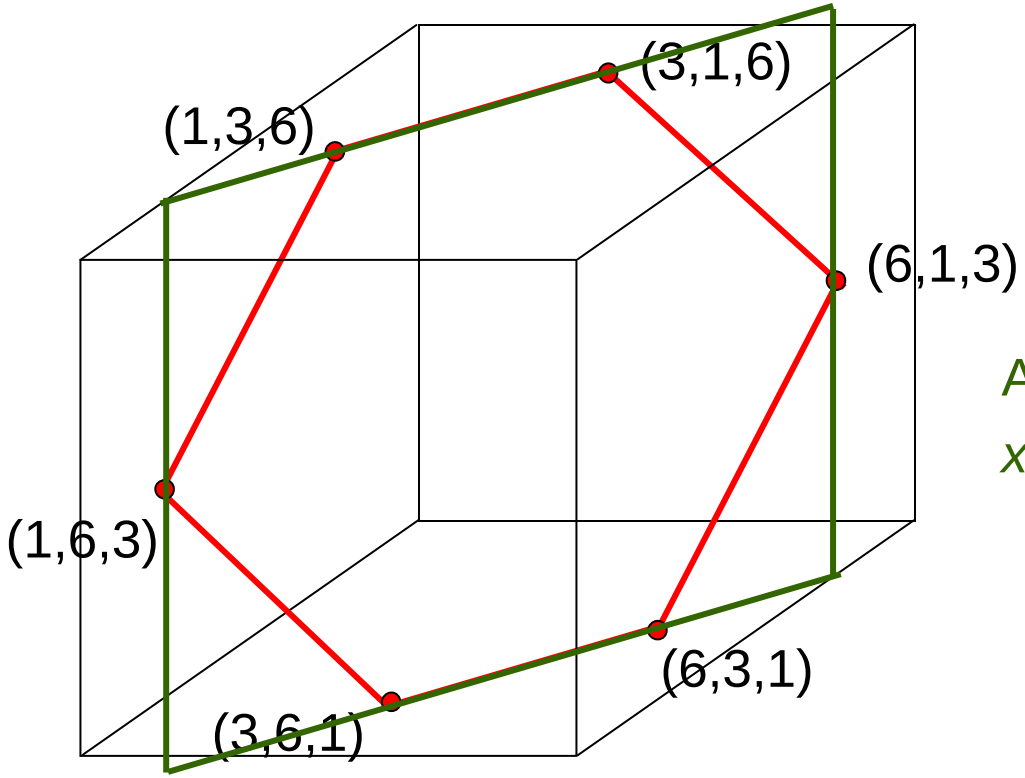
Permutation polytope for alldiff(x_1,x_2,x_3) with domain {1,3,6}



Dimension = 2

Alldiff

Permutation polytope for alldiff(x_1,x_2,x_3) with domain {1,3,6}



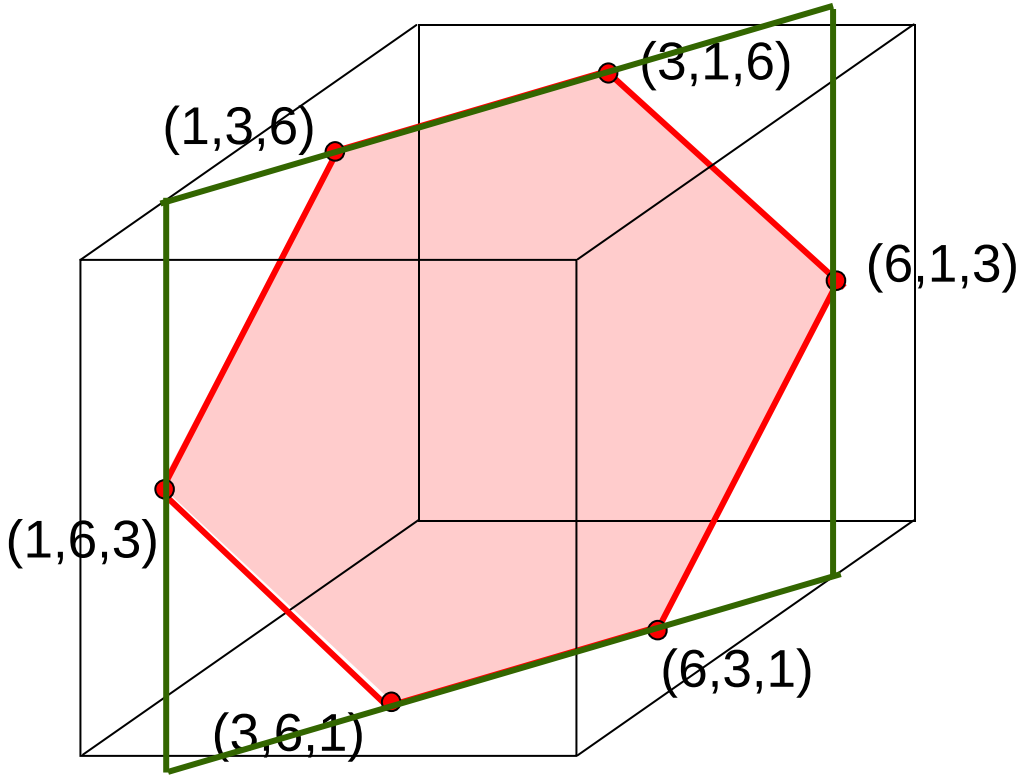
Affine hull

$x_1 + x_2 + x_3 = 10$

Dimension = 2

Alldiff

Unfortunately, convex hull relaxation of alldiff is weak.



Dimension = 2

Circuit

The **circuit** constraint has a tighter convex hull relaxation.

The polytope can be almost completely characterized.

Circuit

The **circuit** constraint has a tighter convex hull relaxation.

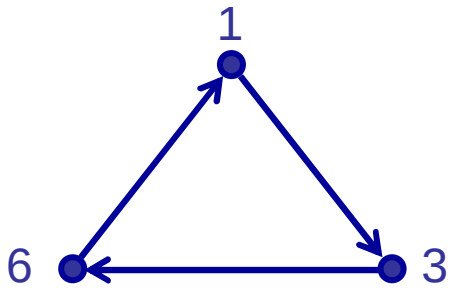
The polytope can be almost completely characterized.

For example, $\text{circuit}(x_1, x_3, x_6)$ with domains {1,3,6}

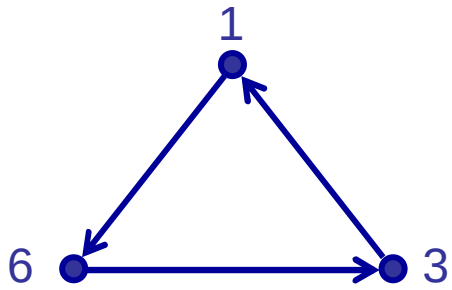
has solutions

Vertex after vertex 1

$(x_1,x_2,x_3) = (3,6,1)$



$(x_1,x_2,x_3) = (6,1,3)$



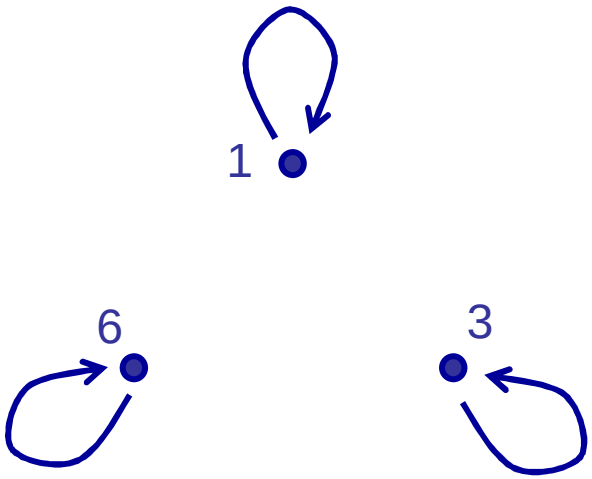
Circuit

The **circuit** constraint has a tighter convex hull relaxation.

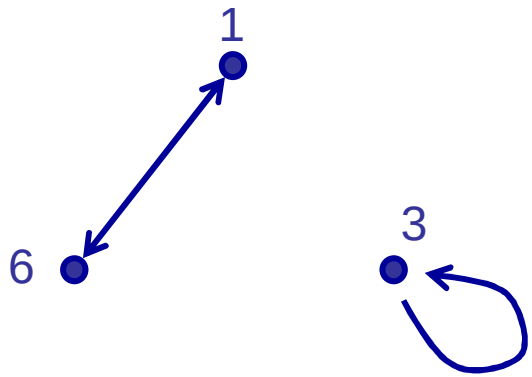
The polytope can be almost completely characterized.

Other permutations are not circuits, for example

$(x_1,x_2,x_3) = (1,3,6)$



$(x_1,x_2,x_3) = (6,1,3)$



Circuit

The **circuit** constraint has a tighter convex hull relaxation.

The polytope can be almost completely characterized.

Traveling salesman problem

using alldiff

$$\min \sum_i c_{x_i x_{i+1}}$$

$$\text{alldiff}(x_1, \boxed{x_2}, x_3)$$

2^{nd} city in tour

using circuit

$$\min \sum_i c_{i x_i}$$

$$\text{circuit}(x_1, \boxed{x_3}, x_6)$$

City after City 3 in tour

Circuit

The **circuit** constraint has a tighter convex hull relaxation.

The polytope can be almost completely characterized.

Traveling salesman problem

using alldiff

$$\min \sum_i c_{x_i x_{i+1}}$$

$$\text{alldiff}(x_1, x_2, x_3)$$

Distance from *i*th to (*i*+1)th city

using circuit

$$\min \sum_i c_{i x_i}$$

$$\text{circuit}(x_1, x_3, x_6)$$

Distance from City *i* to next city

Circuit

The **circuit** constraint has a tighter convex hull relaxation.

The polytope can be almost completely characterized.

Traveling salesman problem

using alldiff

$$\min \sum_i c_{x_i x_{i+1}}$$

$$\text{alldiff}(x_1, x_2, x_3)$$

Filtering is easy
but relaxation is weak

using circuit

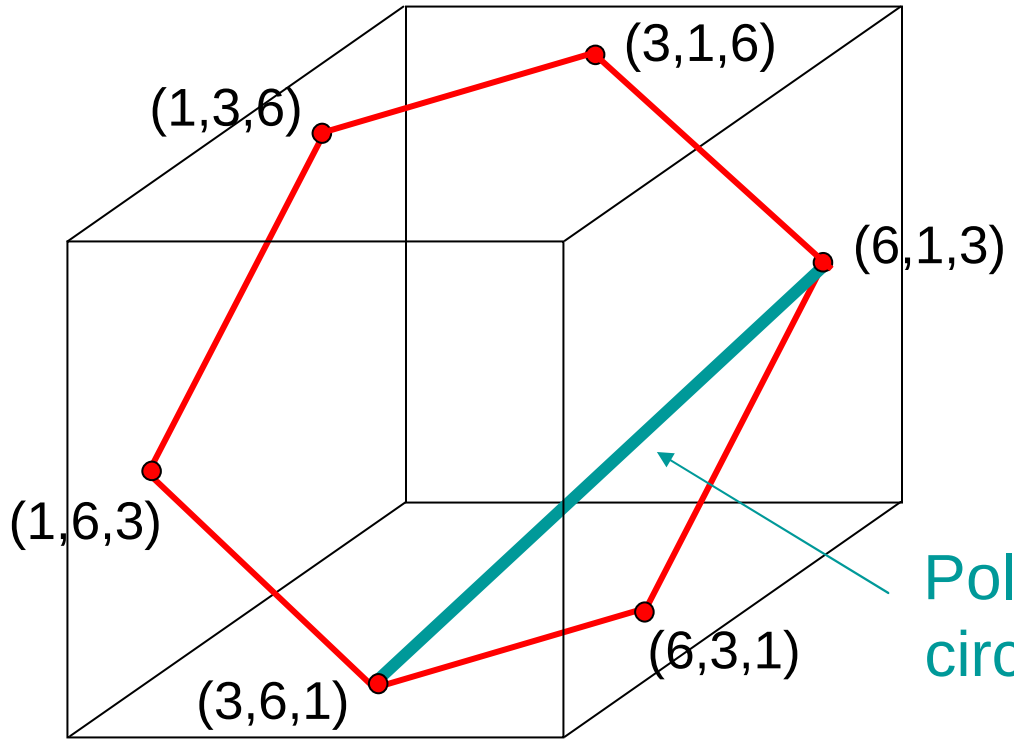
$$\min \sum_i c_{i x_i}$$

$$\text{circuit}(x_1, x_3, x_6)$$

Filtering is hard
but relaxation is tighter

L. Genc-Kaya and JNH, The circuit polytope, 2008

Circuit



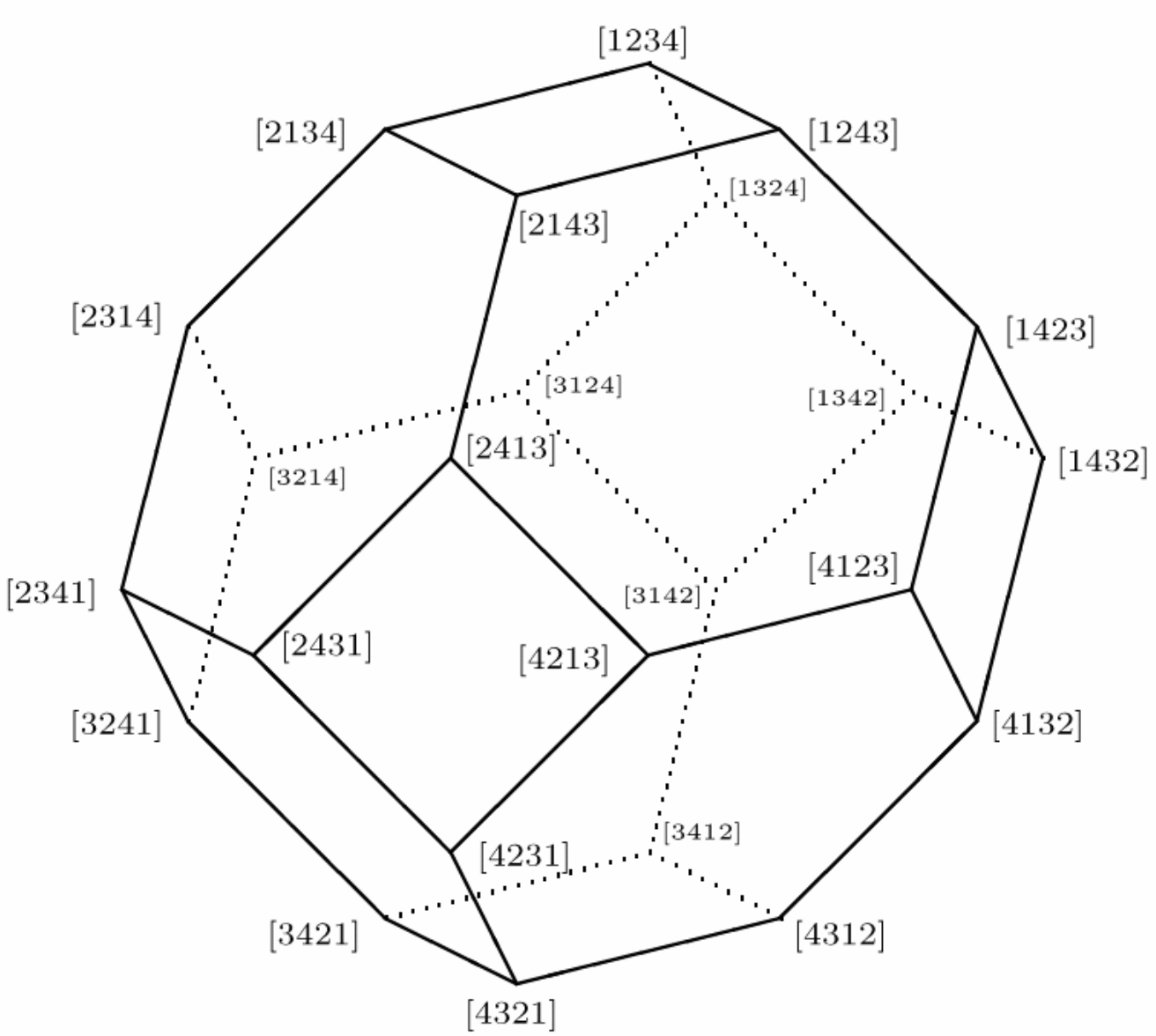
Polytope of circuit(x_1,x_3,x_6)

Dimension = 1

Permutation polytope for domain {1,2,3,4}

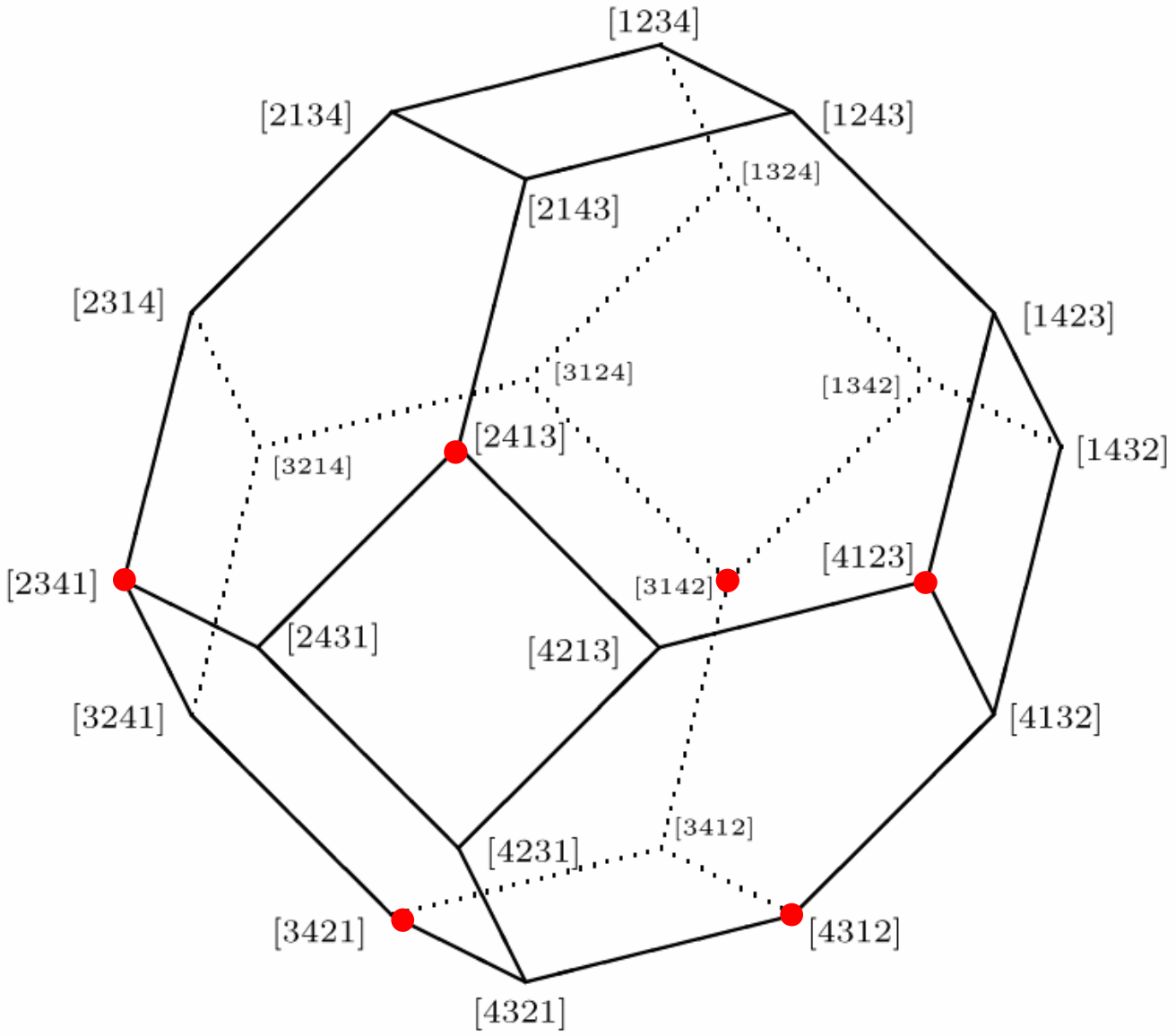
Projected onto 3-space

Dimension = 3



Solutions of circuit(x_1,x_2,x_3,x_4)

Dimension = 3



Circuit

To identify facet-defining inequalities containing variables $x_{j_1}, \ldots x_{j_k}$

- Solve the **combinatorial** problem of identifying **undominated** solution values of $x_{j_1}, \ldots x_{j_k}$
- Solve the **numerical** problem of checking which hyperplanes defined by undominated solutions correspond to valid inequalities.

Circuit

To identify facet-defining inequalities containing variables $x_{j_1}, \ldots x_{j_k}$

- Solve the **combinatorial** problem of identifying **undominated** solution values of $x_{j_1}, \ldots x_{j_k}$
- Solve the **numerical** problem of checking which hyperplanes defined by undominated solutions correspond to valid inequalities.

To simplify presentation, assume we seek facet-defining inequalities with **positive** coefficients.

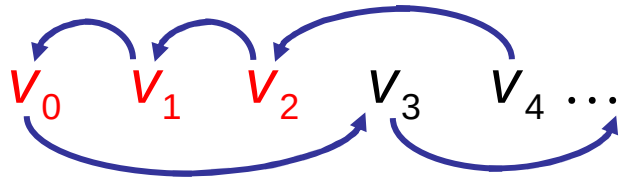
Any sign pattern can be accommodated.

Circuit

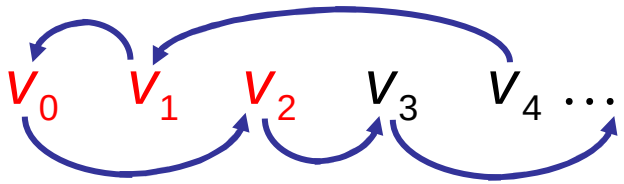
Generate facets of $\text{circuit}(x_{v_0}, \ldots x_{v_{n-1}})$
containing variables $x_{v_0}, x_{v_1}, x_{v_2}$

Circuit

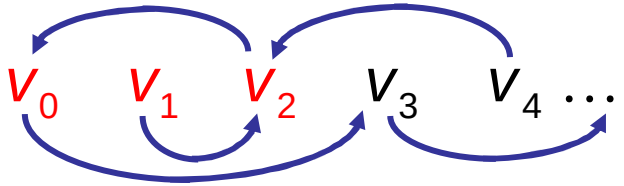
There are exactly 4 undominated solutions.



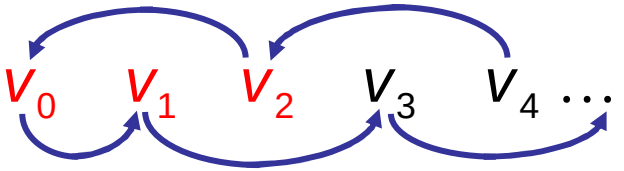
$$(x_{v_0}, x_{v_1}, x_{v_2}, x_{v_3}, x_{v_4} \ldots) = (v_3, v_0, v_1, v_5, v_2 \ldots)$$



$$(x_{v_0}, x_{v_1}, x_{v_2}, x_{v_3}, x_{v_4} \ldots) = (v_2, v_0, v_3, v_5, v_1 \ldots)$$



$$(x_{v_0}, x_{v_1}, x_{v_2}, x_{v_3}, x_{v_4} \ldots) = (v_3, v_2, v_0, v_5, v_2 \ldots)$$



$$(x_{v_0}, x_{v_1}, x_{v_2}, x_{v_3}, x_{v_4} \ldots) = (v_1, v_3, v_0, v_5, v_2 \ldots)$$

Circuit

Each set of 3 solutions determines a hyperplane and potential facet.

$$(X_{v_0}, X_{v_1}, X_{v_2}, X_{v_3}, X_{v_4} \ldots) = (v_3, v_0, v_1, v_5, v_2 \ldots)$$

$$(X_{v_0}, X_{v_1}, X_{v_2}, X_{v_3}, X_{v_4} \ldots) = (v_2, v_0, v_3, v_5, v_1 \ldots)$$

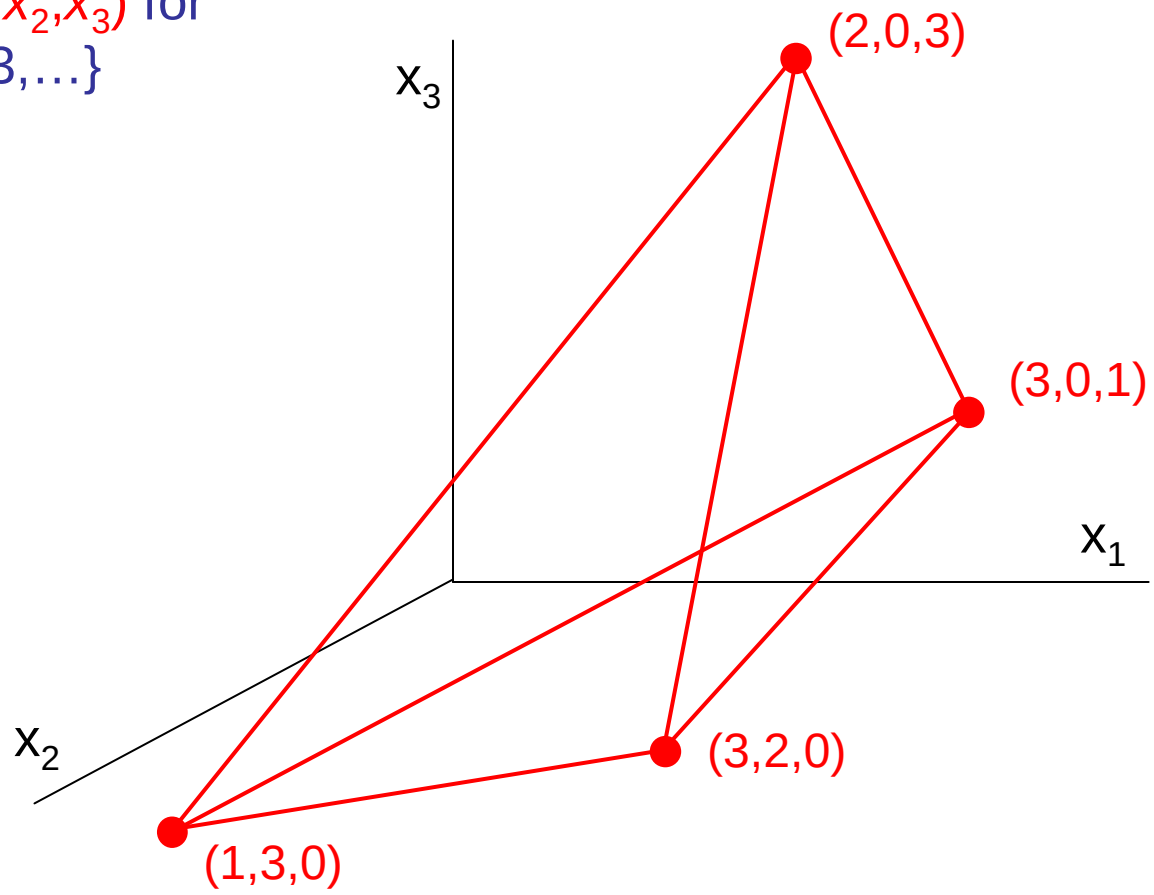
$$(X_{v_0}, X_{v_1}, X_{v_2}, X_{v_3}, X_{v_4} \ldots) = (v_3, v_2, v_0, v_5, v_2 \ldots)$$

$$(X_{v_0}, X_{v_1}, X_{v_2}, X_{v_3}, X_{v_4} \ldots) = (v_1, v_3, v_0, v_5, v_2 \ldots)$$

- The **valid** inequalities that result are **facet-defining**.
- Check each inequality to see if it is violated by the other solution.

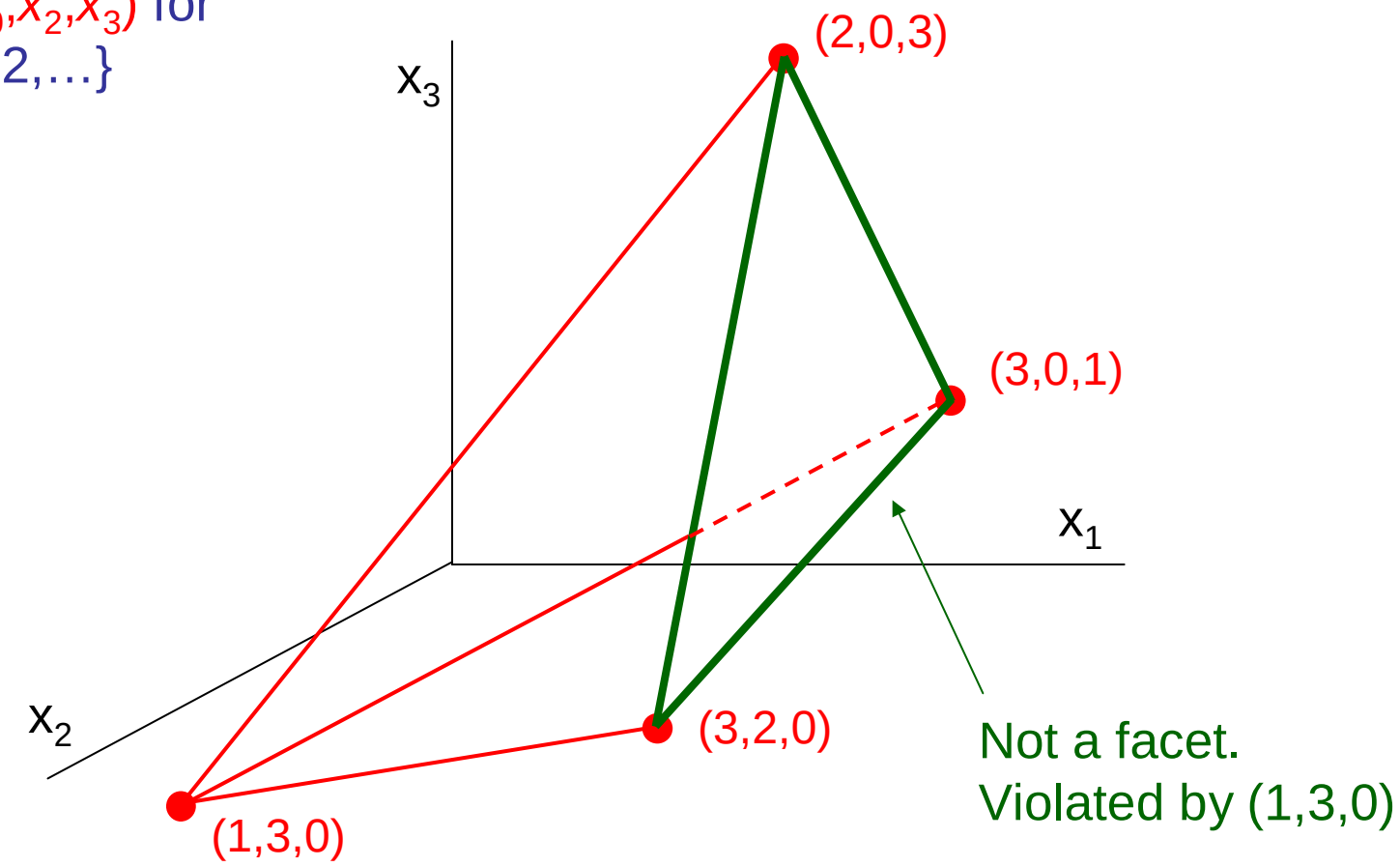
Circuit

Solutions (x_0,x_2,x_3) for domain {0,2,3,…}



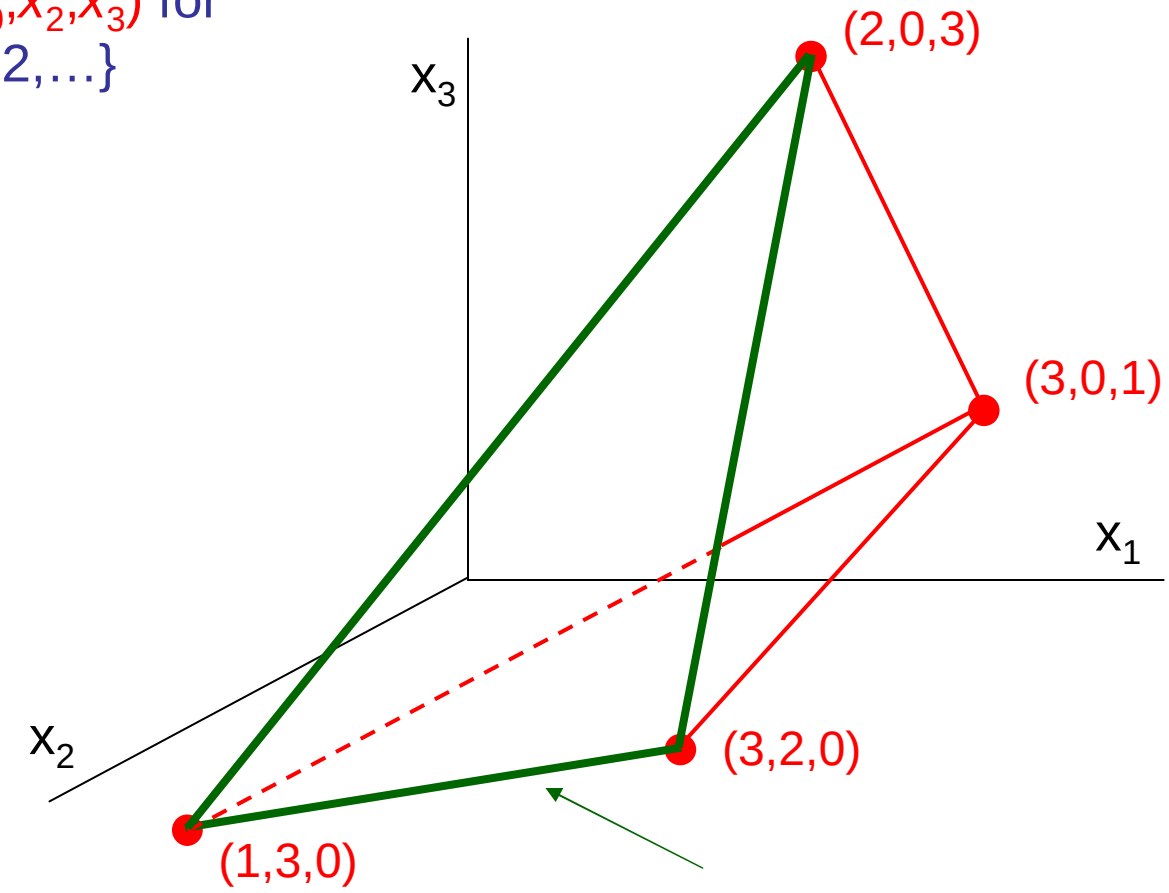
Circuit

Solutions (x_0,x_2,x_3) for domain {0,1,2,…}



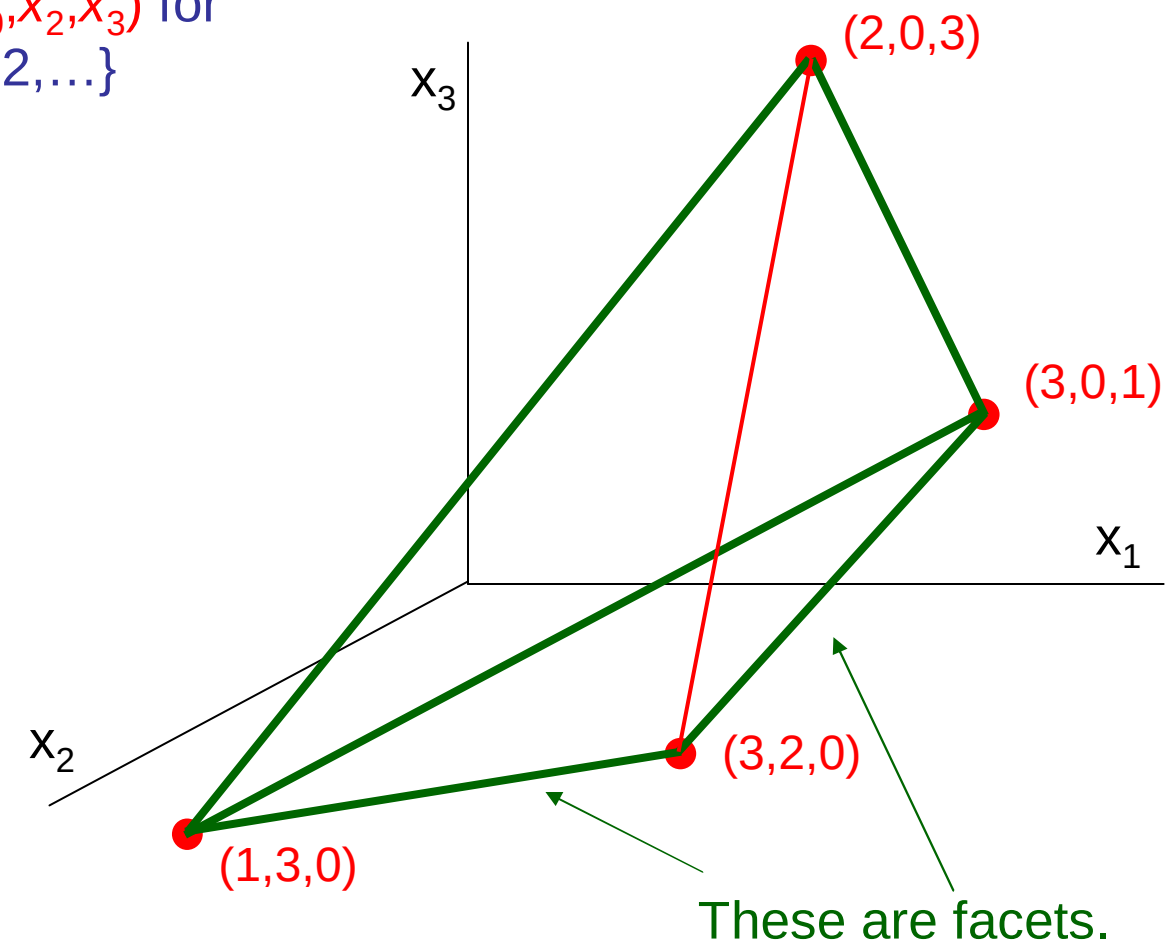
Circuit

Solutions (x_0,x_2,x_3) for domain {0,1,2,…}



Circuit

Solutions (x_0,x_2,x_3) for domain {0,1,2,…}



Circuit

Facets with positive coefficients and variables $X_{v_0}, X_{v_1}, X_{v_2}$

$$(v_3 - v_1)(v_3 - v_0)x_{v_0} + (v_1^2 + v_3^2 + v_0v_2 - v_0v_3 - v_1v_2 - v_1v_3)x_{v_1} + (v_3 - v_2)(v_3 - v_0)x_{v_2}$$
$$\geq -v_0^2(v_3 - v_2) - v_1^2v_0 + v_3^3 + v_0v_1v_3 - v_1v_2v_3$$

For domain {0,1,2,…}: $6x_0 + 5x_1 + 3x_2 \geq 21$

$$(v_3 - v_2)(v_1 - v_0)x_{v_0} + (v_1 - v_0)(v_3 - v_1)x_{v_1} + (v_2 - v_0)(v_3 - v_1)x_{v_2}$$
$$\geq -v_0^2(v_3 - v_1) - v_1^2v_2 + v_3^2(v_1 - v_0) + v_0v_2v_3$$

For domain {0,1,2,…}: $x_0 + 2x_1 + 4x_2 \geq 7$

Circuit

There are also fast **separation heuristics**.

Cumulative scheduling

We can derive **valid inequalities** for

$$\text{cumulative}\big((s_1,\ldots,s_n),(p_1,\ldots,p_n),(c_1,\ldots,c_n),C\big)$$

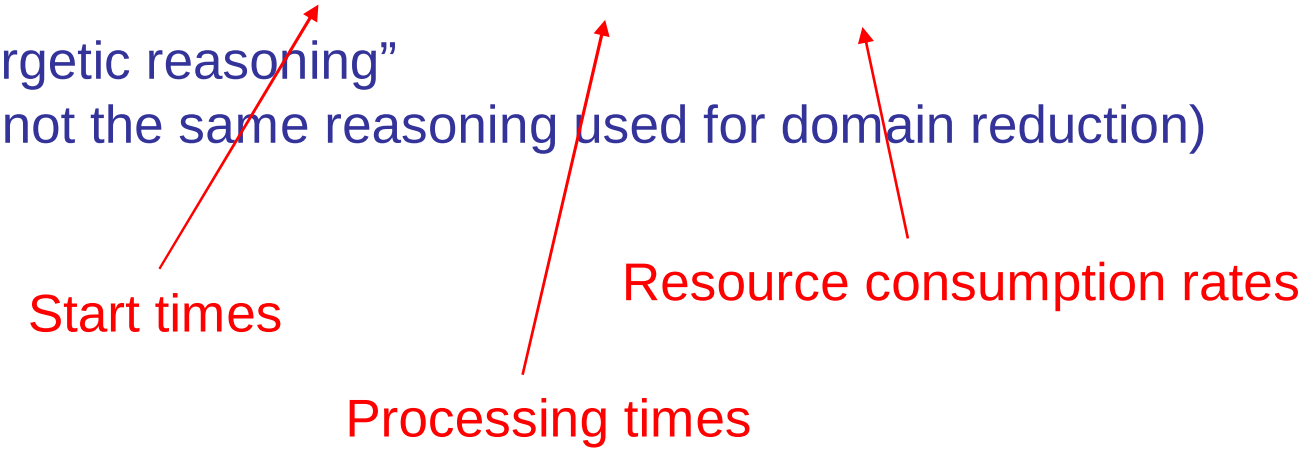
by "energetic reasoning"
(but not the same reasoning used for domain reduction)

Cumulative scheduling

We can derive **valid inequalities** for

$$\text{cumulative}\left((s_1,\ldots,s_n),(p_1,\ldots,p_n),(c_1,\ldots,c_n),C\right)$$

by "energetic reasoning"
(but not the same reasoning used for domain reduction)



Cumulative scheduling

We can derive **valid inequalities** for

$$\text{cumulative}\left((s_1,\ldots,s_n),(p_1,\ldots,p_n),(c_1,\ldots,c_n),C\right)$$

by "energetic reasoning"
(but not the same reasoning used for domain reduction)

Start times

Processing times

Resource consumption rates

Energy of job $j = p_j c_j$

Cumulative scheduling

Take any subset of jobs $J = \{ j_1, \ldots, j_k \}$.

Index jobs so that $p_{j_1} c_{j_1} \leq \cdots \leq p_{j_k} c_{j_k}$

Theorem. The following inequalities (for example) are valid:

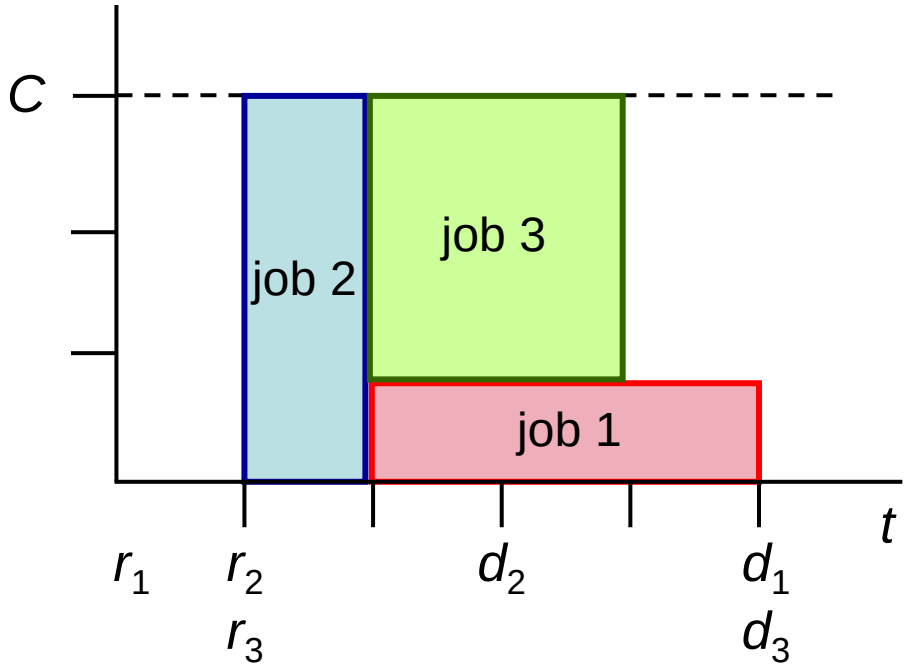
$$k r_J + \frac{1}{C} \sum_{i=1}^{k} (k-i+1) p_{j_i} c_{j_i} - \sum_{i=1}^{k} p_{j_i} \leq \sum_{j \in J} s_j \leq k d_J - \frac{1}{C} \sum_{i=1}^{k} (k-i+1) p_{j_i} c_{j_i}$$

Earliest release time in J

Latest deadline in J

Cumulative scheduling

Example



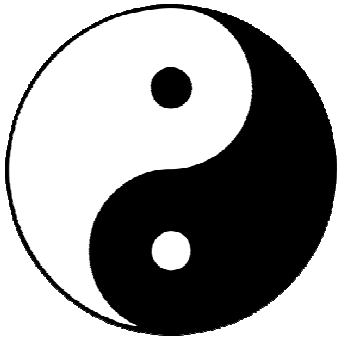
Valid inequalities: $\frac{1}{3} \leq s_1 + s_2 + s_3 \leq 8\frac{2}{3}$

$$2\frac{1}{3} \leq s_2 + s_3 \leq 6\frac{2}{3}$$

Cappadocia, Türkiye

CP 2008
Slide 89

Subterranean city, Derinkuyu, Türkiye



Relaxation Duality

Relaxation Dual

Lagrangean Dual

Example: Fast LP

Example: TSP with Time Windows

Relaxation dual

We often search over problem **restrictions** to find a better solutions.

- Branching methods
- Local search

Can we search over **relaxations** to find a better bound?

Relaxation dual

We often search over problem **restrictions** to find a better solutions.

- Branching methods
- Local search

Can we search over **relaxations** to find a better bound?

Yes, if we **parameterize** the relaxations.

- Then search over parameter values.

Relaxation dual

A **relaxation dual** seeks a relaxation that provides the tightest bound.

- The relaxation parameters are **dual variables**.
- Solve the dual by **searching** over values of the dual variables

Relaxation dual

A **relaxation dual** seeks a relaxation that provides the tightest bound.

- The relaxation parameters are **dual variables**.
- Solve the dual by **searching** over values of the dual variables

Some relaxation duals:

- Linear programming dual
- Lagrangean dual
- Surrogate dual
- Superadditive dual
- Roof dual, etc. etc.

Relaxation dual

Given the optimization problem

$$\min_{x \in D} \left\{ f(x) \mid C \right\}$$

Relaxation dual

Given the optimization problem

$$\min_{x \in D} \{ f(x) \mid C \}$$

Easy constraints

Hard constraints

Relaxation dual

Given the optimization problem

$$\min_{x \in D} \{ f(x) \mid C \}$$

Easy constraints

Hard constraints

A parameterized relaxation is

$$\min_{x \in D} \{ f(x, \lambda) \mid C(\lambda) \}$$

Lower bound on $f(x)$

Relaxation of C
(easier constraints)

Relaxation dual

Given the optimization problem

$$\min_{x \in D} \{ f(x) \mid C \}$$

Easy constraints

Hard constraints

A parameterized relaxation is

$$\min_{x \in D} \{ f(x, \lambda) \mid C(\lambda) \}$$

Lower bound on $f(x)$

Relaxation of C

vector of **dual variables**

Relaxation dual

Given the optimization problem

$$\min_{x \in D} \{ f(x) \mid C \}$$

Easy constraints

Hard constraints

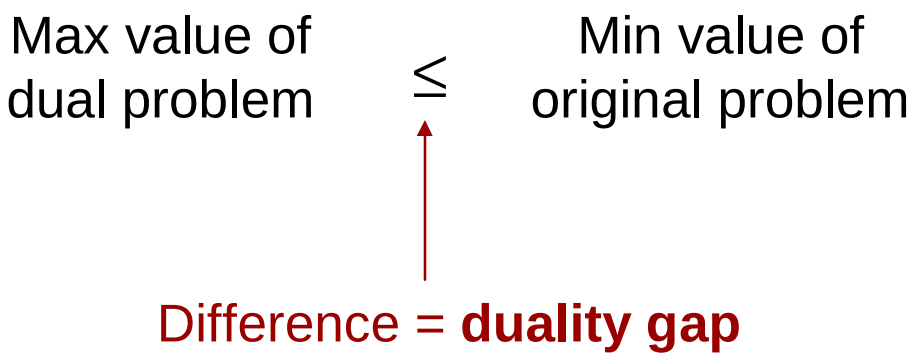
The **relaxation dual** is

$$\max_{\lambda} \left\{ \min_{x \in D} \{ f(x, \lambda) \mid C(\lambda) \} \right\}$$

Find relaxation that provides the largest lower bound.

Relaxation dual

Weak duality always holds:



Lagrangean dual

Used when the hard constraints are inequality constraints.

$$\min_{x \in D} \left\{ f(x) \mid g(x) \geq 0 \right\}$$

Lagrangean dual

Used when the hard constraints are inequality constraints.

$$\min_{x \in D} \{ f(x) \mid g(x) \geq 0 \}$$

The relaxation is parameterized by a vector of **Langrange multipliers** u.

$$\min_{x \in D} \{ f(x) - \lambda^T g(x) \mid \quad \}$$

Lower bound on $f(x)$

Hard constraints disappear completely

Lagrangean dual

Used when the hard constraints are inequality constraints.

$$\min_{x \in D} \{ f(x) \mid g(x) \geq 0 \}$$

The relaxation is parameterized by a vector of **Langrange multipliers** *u*.

$$\min_{x \in D} \{ f(x) - \lambda^T g(x) \mid \quad \}$$

Lower bound on $f(x)$

Hard constraints disappear completely

Dualized constraints

Lagrangean dual

Used when the hard constraints are inequality constraints.

$$\min_{x \in D} \left\{ f(x) \,\middle|\, g(x) \geq 0 \right\}$$

The **Lagrangean dual** is

$$\max_{u \geq 0} \left\{ \min_{x \in D} \left\{ f(x) - \lambda^T g(x) \right\} \right\}$$

Lagrangean dual

Can use **subgradient optimization** to solve the dual.

Exploit the fact that $\theta(\lambda) = \min_{x \in D} \{ f(x) \mid g(x) \geq 0 \}$

Is always **concave** (but not differentiable).

Lagrangean dual

Can use **subgradient optimization** to solve the dual.

Exploit the fact that $\theta(\lambda) = \min_{x \in D} \{ f(x) \mid g(x) \geq 0 \}$

Is always **concave** (but not differentiable).

A **subgradient** (direction of steepest ascent) of $\theta(\lambda)$ is $-g(x^*)$, where x^* solves $\min_{x \in D} \{ f(x) \mid g(x) \geq 0 \}$

Lagrangean dual

Can use **subgradient optimization** to solve the dual.

Exploit the fact that $\theta(\lambda) = \min_{x \in D} \{ f(x) \mid g(x) \geq 0 \}$

Is always **concave** (but not differentiable).

A **subgradient** (direction of steepest ascent) of $\theta(\lambda)$ is $-g(x^*)$, where x^* solves $\min_{x \in D} \{ f(x) \mid g(x) \geq 0 \}$

Step k of search is $\lambda^{k+1} = \lambda^k + \alpha_k g(x^*)$

Lagrangean dual

Can use **subgradient optimization** to solve the dual.

Exploit the fact that $\theta(\lambda) = \min_{x \in D} \{ f(x) \mid g(x) \geq 0 \}$

is always **concave** (but not differentiable).

A **subgradient** (direction of steepest ascent) of $\theta(\lambda)$ is $-g(x^*)$, where x^* solves $\min_{x \in D} \{ f(x) \mid g(x) \geq 0 \}$

Step k of search is $\lambda^{k+1} = \lambda^k + \alpha_k g(x^*)$

stepsize,
perhaps $\alpha_k = 1/k$

Example

$$
\begin{aligned}
\min \quad & 4x_1 + 5x_2 + 6x_3 \\
\text{subject to} \quad & 3x_1 + 4x_2 + 5x_3 \geq 47 \\
& x_1 + 2x_2 + 3x_3 \geq 24 \qquad \} \text{ hard} \\
& x_1 \geq 2 \\
& x_2 \geq 3 \qquad \} \text{ easy} \\
& x_3 \geq 4 \\
& x \text{ integer}
\end{aligned}
$$

Example

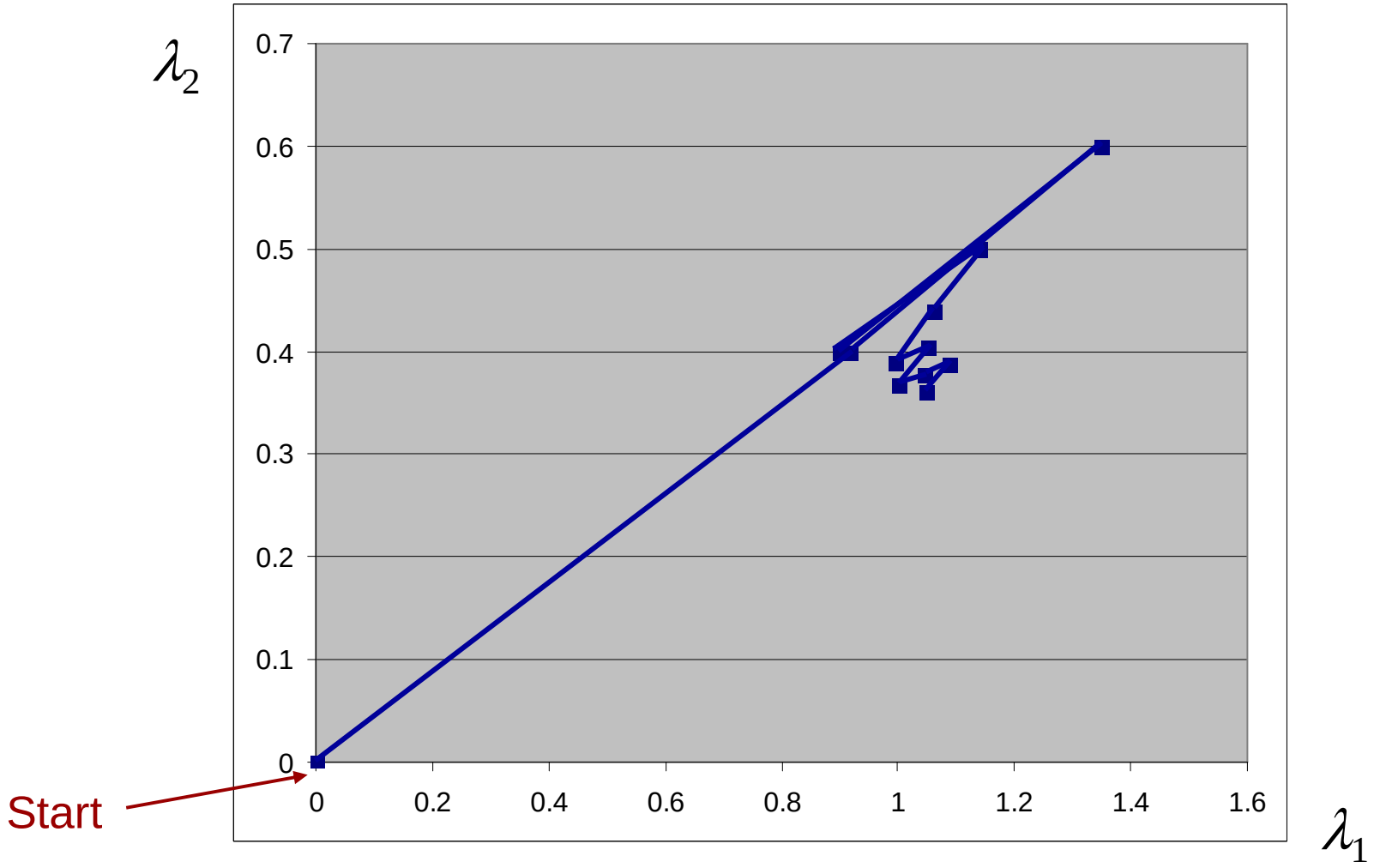
$$
\begin{aligned}
\min \quad & 4x_1 + 5x_2 + 6x_3 \\
\text{subject to} \quad & 3x_1 + 4x_2 + 5x_3 \geq 47 \\
& x_1 + 2x_2 + 3x_3 \geq 24 \qquad \} \text{ hard} \\
& x_1 \geq 2 \\
& x_2 \geq 3 \qquad \} \text{ easy} \\
& x_3 \geq 4 \\
& x \text{ integer}
\end{aligned}
$$

$$
\theta(\lambda) = \min_{\substack{x_1 \geq 2 \\ x_2 \geq 3 \\ x_3 \geq 4}} \left\{ \begin{array}{l} 4x_1 + 5x_2 + 6x_3 \\ +\lambda_1(47 - 3x_1 - 4x_2 - 5x_3) \\ +\lambda_2(24 - x_1 - 2x_2 - 3x_3) \end{array} \right\}
$$

Solve by inspection for fixed λ

$$
= \min_{\substack{x_1 \geq 2 \\ x_2 \geq 3 \\ x_3 \geq 4}} \left\{ \begin{array}{l} (4 - 3\lambda_1 - \lambda_2)x_1 + (5 - 4\lambda_1 - 2\lambda_2)x_2 \\ +(6 - 5\lambda_1 - 3\lambda_2)x_3 + 47\lambda_1 + 24\lambda_2 \end{array} \right\}
$$

Subgradient search in λ-space:



Value of Lagrangean dual = 57.6 < 58 = optimal value

Lagrangean dual

The Lagrangean dual of a **linear programming problem** is the classical linear programming dual.

Lagrangean dual

The Lagrangean dual of a **linear programming problem** is the classical linear programming dual.

Optimization duals can also be conceived as **inference duals**.

- As in Benders decomposition, sensitivity analysis, etc.

Example: Fast Linear Programming

In CP, it is best to process each node of the search tree very rapidly.

Lagrangean relaxation allows fast calculation of a lower bound on the LP value at each node.

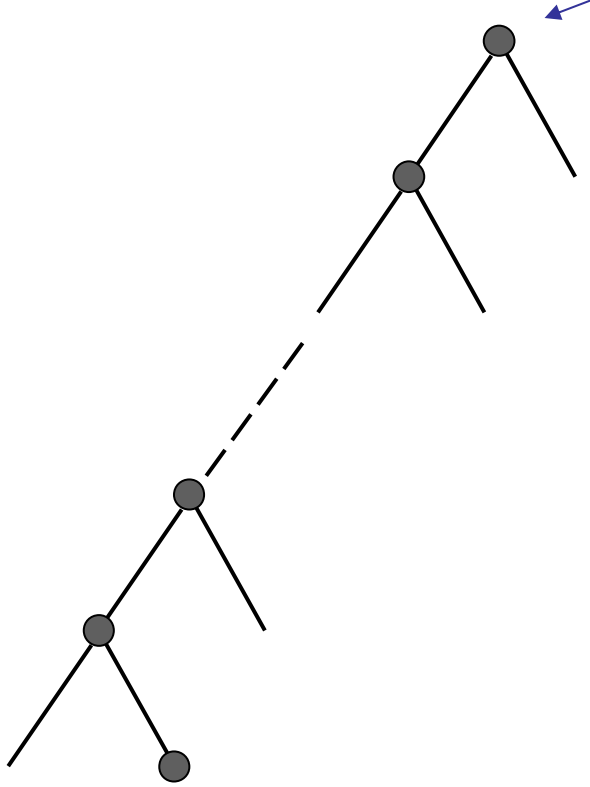
Example: Fast Linear Programming

In CP, it is best to process each node of the search tree very rapidly.

Lagrangean relaxation allows fast calculation of a lower bound on the LP value at each node.

Solve the Lagrangean dual at the root node (which is an LP) and use the **same Lagrange multipliers** to get an LP bound at other nodes.

Example: Fast LP



At root node, solve $\min\ cx$

Dualize → $Ax \geq b \quad (\lambda)$

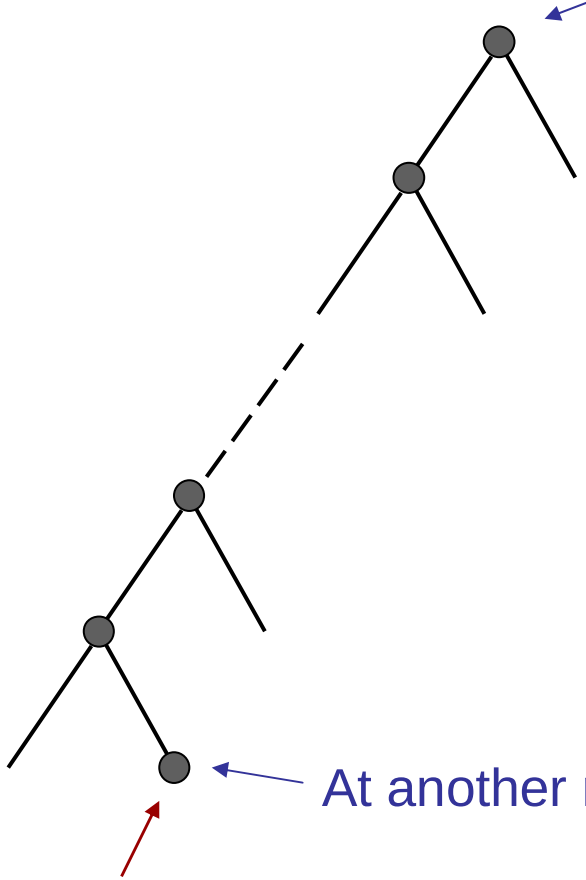
Special structure, e.g. variable bounds → $Dx \geq d$

$x \geq 0$

The (partial) LP dual solution λ^* solves the Lagrangean dual in which

$$\theta(\lambda) = \min_{\substack{Dx \geq d \\ x \geq 0}} \{cx - \lambda(Ax - b)\}$$

Example: Fast LP



At root node, solve $\min\ cx$

Dualize → $Ax \geq b \quad (u)$

Special structure, e.g. variable bounds → $Dx \geq d$

$x \geq 0$

The (partial) LP dual solution u^* solves the Lagrangean dual in which

$$\theta(u) = \min_{\substack{Dx \geq d \\ x \geq 0}} \left\{ cx - u^T (Ax - b) \right\}$$

At another node, the LP is

$\min\ cx$

$Ax \geq b \quad (u)$

$Dx \geq d$

$Hx \geq h$ ← Branching constraints, etc.

$x \geq 0$

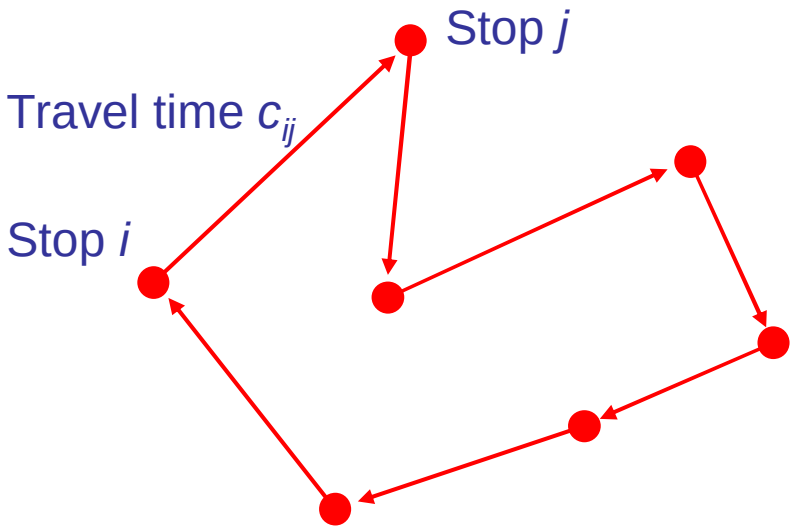
Lower bound $\theta(\lambda^*)$ is quickly calculated by solving a specially structured LP.

Example: TSP with Time Windows

A salesman must make several stops and return home, subject to time windows at each stop.

Objective: Minimize travel time.

Use Lagrange multipliers for **cost-based filtering**.



TSP with time windows

Assignment relaxation

$$\min \sum_{ij} c_{ij} x_{ij}$$

= 1 if stop *i* immediately precedes stop *j*

$$\sum_j x_{ij} = \sum_j x_{ji} = 1, \text{ all } i$$

Stop *i* is preceded and followed by exactly one stop.

$$x_{ij} \in \{0,1\}, \text{ all } i, j$$

TSP with time windows

Assignment relaxation

$$\min \sum_{ij} c_{ij} x_{ij}$$

$$\sum_j x_{ij} = \sum_j x_{ji} = 1, \text{ all } i$$

$$0 \le x_{ij} \le 1, \text{ all } i, j \quad (\lambda_{ij})$$

Because this problem is **totally unimodular**, it can be solved as an LP.

TSP with time windows

Assignment relaxation

$$\min \sum_{ij} c_{ij} x_{ij}$$

$$\sum_j x_{ij} = \sum_j x_{ji} = 1, \text{ all } i$$

$$0 \le x_{ij} \le 1, \text{ all } i, j \quad (\lambda_{ij})$$

Because this problem is **totally unimodular**, it can be solved as an LP.

The relaxation provides a **very weak lower bound** on the optimal value.

TSP with time windows

Assignment relaxation

$$\min \sum_{ij} c_{ij} x_{ij}$$

$$\sum_j x_{ij} = \sum_j x_{ji} = 1, \text{ all } i$$

$$0 \le x_{ij} \le 1, \text{ all } i, j \quad (\lambda_{ij})$$

Because this problem is **totally unimodular**, it can be solved as an LP.

The relaxation provides a **very weak lower bound** on the optimal value.

But **cost-based filtering** can be effective.

TSP with time windows

Assignment relaxation

$$\min \sum_{ij} c_{ij} x_{ij}$$

$$\sum_j x_{ij} = \sum_j x_{ji} = 1, \text{ all } i$$

$$0 \le x_{ij} \le 1, \text{ all } i, j \quad (\lambda_{ij})$$

We know $x_{ij} \le \dfrac{U - v^*}{\lambda_{ij}}$

TSP with time windows

Assignment relaxation

$$\min \sum_{ij} c_{ij} x_{ij}$$

$$\sum_j x_{ij} = \sum_j x_{ji} = 1, \text{ all } i$$

$$0 \le x_{ij} \le 1, \text{ all } i, j \quad (\lambda_{ij})$$

Known upper bound on shortest distance

Value of LP relaxation

We know $x_{ij} \le \dfrac{U - v^*}{\lambda_{ij}}$

TSP with time windows

Assignment relaxation

$$\min \sum_{ij} c_{ij} x_{ij}$$

$$\sum_j x_{ij} = \sum_j x_{ji} = 1, \text{ all } i$$

$$0 \le x_{ij} \le 1, \text{ all } i, j \quad (\lambda_{ij})$$

Known upper bound on shortest distance

Value of LP relaxation

We know $x_{ij} \le \dfrac{U - v^*}{\lambda_{ij}}$

Lagrange multiplier

TSP with time windows

Assignment relaxation

$$\min \sum_{ij} c_{ij} x_{ij}$$

$$\sum_j x_{ij} = \sum_j x_{ji} = 1, \quad \text{all } i$$

$$0 \le x_{ij} \le 1, \quad \text{all } i, j \qquad (\lambda_{ij})$$

Known upper bound on shortest distance

Value of LP relaxation

We know $x_{ij} \le \dfrac{U - v^*}{\lambda_{ij}}$

Lagrange multiplier

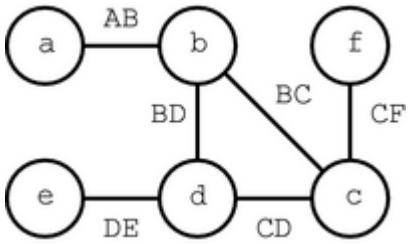
If this upper bound is < 1, we can **fix** x_{ij} to 0.

Known in OR as reduced-cost variable fixing

Focacci, Lodi & Milano, Solving TSP with time windows with constraints, 1999

Éméi Shān, Sichuan Province, China

Tài jí quán



Discrete Relaxation: Dependency Graph

Dependency Graph and Induced Width

Example: Relaxing the Constraint Dual

Dependency Graph and Induced Width

Complexity of solving a problem is at worst exponential in the **induced width** of the **dependency graph**

Dependency Graph and Induced Width

Complexity of solving a problem is at worst exponential in the **induced width** of the **dependency graph**

The idea has surfaced independently in several contexts.

- Nonserial dynamic programming (1972)
- Belief logics (1986)
- k-trees (1986)
- Bayesian networks (1988)
- Pseudo-boolean optimization (1990)
- Location analysis (1994)
- Bucket elimination (1996)

Dependency graph and induced width

Let's relax the problem by **thinning out** the dependency graph.

- Reduce the induced width while maintaining a valid relaxation.

Use **relaxation duality** to find the best relaxation.

Example: Relaxing the Constraint Dual

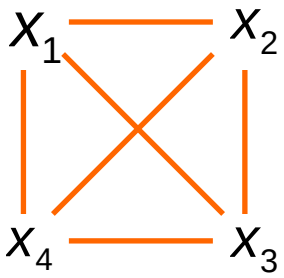
$$
\begin{aligned}
\min\ & x_1 + 2x_2 + 3x_3 + 4x_4 \\
& x_1 + x_2 + x_3 \geq 1 \\
& x_1 + x_2 + x_4 \geq 1 \\
& x_2 + x_3 + x_4 \geq 1 \\
& x_j \in \{0,1\}
\end{aligned}
$$

Solution: $(x_1,\ldots,x_4) = (0,1,0,0)$

Optimal value is 2.

Example: Relaxing the constraint dual

Dependency graph has induced width 3:



$$\min x_1 + 2x_2 + 3x_3 + 4x_4$$
$$x_1 + x_2 + x_3 \geq 1$$
$$x_1 + x_2 + x_4 \geq 1$$
$$x_2 + x_3 + x_4 \geq 1$$
$$x_j \in \{0,1\}$$

Example: Relaxing the constraint dual

Relax the problem by removing edges from dependency graph.

- and form "mini-buckets"

$$\begin{aligned}
\min\ & x_1 + 2x_2 + 3x_3 + 4x_4 \\
& x_1 + x_2 + x_3 \geq 1 \\
& x_1 + x_2 + x_4 \geq 1 \\
& x_2 + x_3 + x_4 \geq 1 \\
& x_j \in \{0,1\}
\end{aligned}$$

$$\min f_1(x_1,x_2,x_3) + f_2(x_1,x_2,x_4) + f_3(x_2,x_3,x_4)$$

$$f_1(x_1,x_2,x_3) = \begin{cases} x_1 + 2x_2 + 3x_3 & \text{if } x_1 + x_2 + x_3 \geq 1 \\ \infty & \text{otherwise} \end{cases}$$

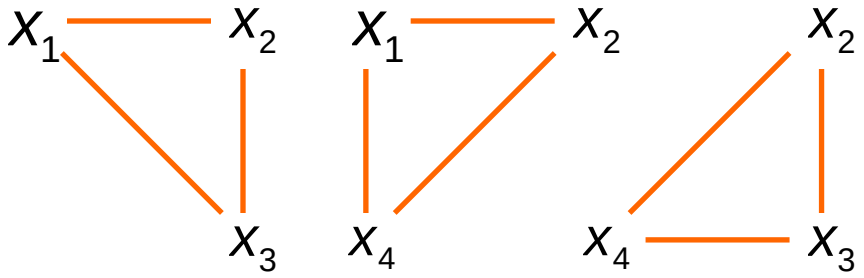
$$f_2(x_1,x_2,x_4) = \begin{cases} 0x_1 + 0x_2 + 4x_4 & \text{if } x_1 + x_2 + x_4 \geq 1 \\ \infty & \text{otherwise} \end{cases}$$

$$f_3(x_2,x_3,x_4) = \begin{cases} 0x_2 + 0x_3 + 0x_4 & \text{if } x_2 + x_3 + x_4 \geq 1 \\ \infty & \text{otherwise} \end{cases}$$

Dechter & Rish, Mini-buckets: A general scheme for approximating inference, 1998

Example: Relaxing the constraint dual

The problem separates into 3 problems with induced width 2:



$$\min f_1(x_1,x_2,x_3)+f_2(x_1,x_2,x_4)+f_3(x_2,x_3,x_4)$$

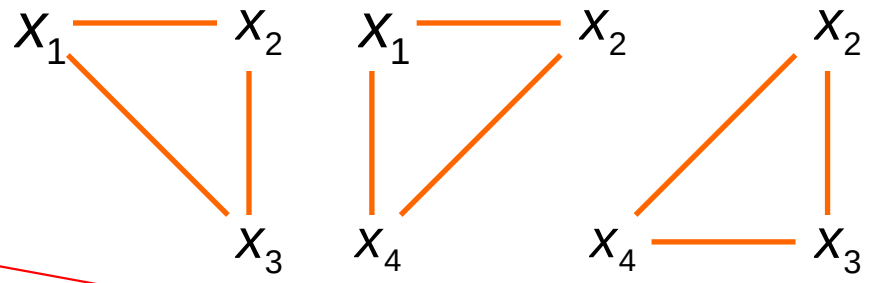
$$f_1(x_1,x_2,x_3)=\begin{cases} x_1+2x_2+3x_3 & \text{if } x_1+x_2+x_3\geq 1 \\ \infty & \text{otherwise} \end{cases}$$

$$f_2(x_1,x_2,x_4)=\begin{cases} 0x_1+0x_2+4x_4 & \text{if } x_1+x_2+x_4\geq 1 \\ \infty & \text{otherwise} \end{cases}$$

$$f_3(x_2,x_3,x_4)=\begin{cases} 0x_2+0x_3+0x_4 & \text{if } x_2+x_3+x_4\geq 1 \\ \infty & \text{otherwise} \end{cases}$$

Example: Relaxing the constraint dual

But the resulting bound is weak



$$\min f_1(x_1,x_2,x_3)+f_2(x_1,x_2,x_4)+f_3(x_2,x_3,x_4) = 1+0+0 = 1 < 2$$

$$f_1(x_1,x_2,x_3) = \begin{cases} x_1+2x_2+3x_3 & \text{if } x_1+x_2+x_3 \geq 1 \\ \infty & \text{otherwise} \end{cases}$$

$$f_2(x_1,x_2,x_4) = \begin{cases} 0x_1+0x_2+4x_4 & \text{if } x_1+x_2+x_4 \geq 1 \\ \infty & \text{otherwise} \end{cases}$$

$$f_3(x_2,x_3,x_4) = \begin{cases} 0x_2+0x_3+0x_4 & \text{if } x_2+x_3+x_4 \geq 1 \\ \infty & \text{otherwise} \end{cases}$$

Example: Relaxing the constraint dual

Let's apply relaxation duality to the constraint dual:

$$
\begin{aligned}
&x_1^1 = x_1^2 \\
&x_2^1 = x_2^2 = x_2^3 \\
&x_3^1 = x_3^3 \\
&x_4^1 = x_4^3 \\
&(x_1^1, x_2^1, x_3^1) \in D \\
&(x_1^2, x_2^2, x_4^2) \in D \\
&(x_2^3, x_3^3, x_4^3) \in D
\end{aligned}
$$

where $D = \{0,1\}^3 - \{(0,0,0)\}$

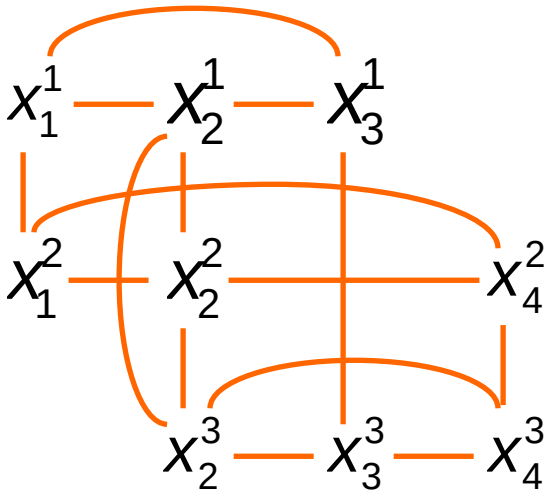
$$
\begin{aligned}
\min\ & x_1 + 2x_2 + 3x_3 + 4x_4 \\
& x_1 + x_2 + x_3 \geq 1 \\
& x_1 + x_2 + x_4 \geq 1 \\
& x_2 + x_3 + x_4 \geq 1 \\
& x_j \in \{0,1\}
\end{aligned}
$$

Standardize apart variables in different constraints and equate them in binary constraints.

Example: Relaxing the constraint dual

Dependency graph for constraint dual:

$$\min\ x_1 + 2x_2 + 3x_3 + 4x_4$$
$$x_1 + x_2 + x_3 \geq 1$$
$$x_1 + x_2 + x_4 \geq 1$$
$$x_2 + x_3 + x_4 \geq 1$$
$$x_j \in \{0,1\}$$



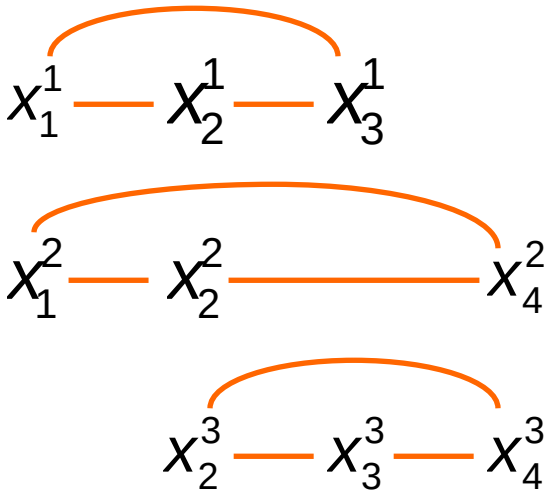
Induced width of dependency graph is 3.

Any deletion of vertical edges yields a valid relaxation.

Example: Relaxing the constraint dual

So let's delete vertical edges, which equate variables.

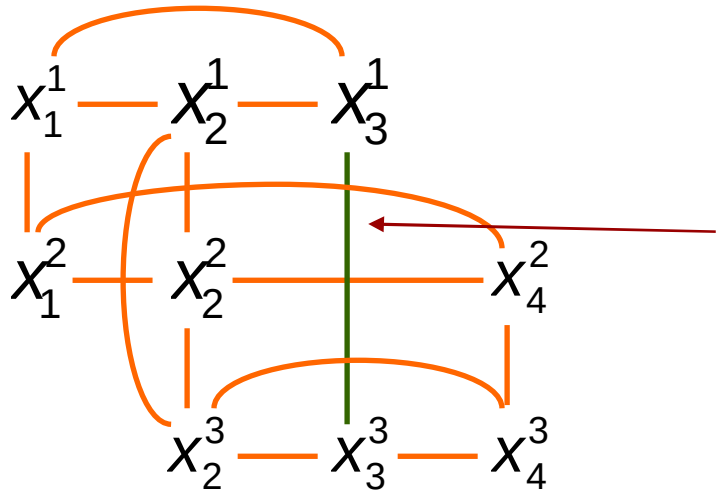
$$\min\ x_1 + 2x_2 + 3x_3 + 4x_4$$
$$x_1 + x_2 + x_3 \geq 1$$
$$x_1 + x_2 + x_4 \geq 1$$
$$x_2 + x_3 + x_4 \geq 1$$
$$x_j \in \{0,1\}$$



The previous mini-bucket scheme deletes **all** vertical edges.

Example: Relaxing the constraint dual

Let's search other deletion patterns to find a tighter relaxation (i.e., solve relaxation dual)

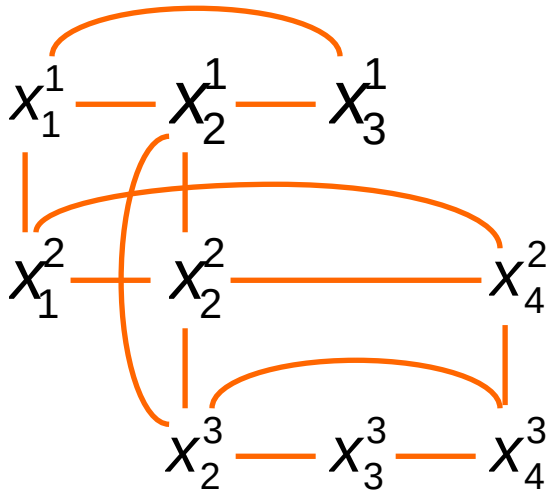


$$\min x_1 + 2x_2 + 3x_3 + 4x_4$$
$$x_1 + x_2 + x_3 \geq 1$$
$$x_1 + x_2 + x_4 \geq 1$$
$$x_2 + x_3 + x_4 \geq 1$$
$$x_j \in \{0,1\}$$

By deleting only **one** edge we can reduce induced width to 2

Example: Relaxing the constraint dual

Let's search other deletion patterns to find a tighter relaxation (i.e., solve relaxation dual).



$$\min\ x_1 + 2x_2 + 3x_3 + 4x_4$$
$$x_1 + x_2 + x_3 \geq 1$$
$$x_1 + x_2 + x_4 \geq 1$$
$$x_2 + x_3 + x_4 \geq 1$$
$$x_j \in \{0,1\}$$

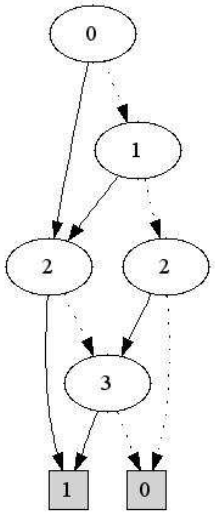
By deleting only **one** edge we can reduce induced width to 2.

Bound is now tight.

JNH, Duality in optimization and constraint satisfaction, 2006

Great Smoky Mountains, USA

800 million years old



Discrete Relaxation: MDDs

Domain Store Relaxation

Multivalued Decision Diagrams

MDD Relaxation

Propagation in Relaxed MDDs

Domain Store Relaxation

A **domain store** can be viewed as a (weak) relaxation.

We propagate constraints in the domain store by using them to **refine** it.

Domain Store Relaxation

A **domain store** can be viewed as a (weak) relaxation.

We propagate constraints in the domain store by using them to **refine** it.

Question: Can we propagate in a **tighter** relaxation?

Domain Store Relaxation

A **domain store** can be viewed as a (weak) relaxation.

We propagate constraints in the domain store by using them to **refine** it.

Question: Can we propagate in a **tighter** relaxation?

Proposal: Use a **relaxed multivalued decision diagram**.

This can **reduce branching** by propagating more information.

Domain Store Relaxation

A **domain store** can be viewed as a (weak) relaxation.

We propagate constraints in the domain store by using them to **refine** it.

Question: Can we propagate in a **tighter** relaxation?

Proposal: Use a **relaxed multivalued decision diagram**.

This can **reduce branching** by propagating more information.

Can also justify **more thorough processing** of each constraint.

Multivalued Decision Diagrams

A **reduced ordered MDD** is the result of superimposing all isomorphic subtrees in an enumeration tree and removing redundant edges.

Example:

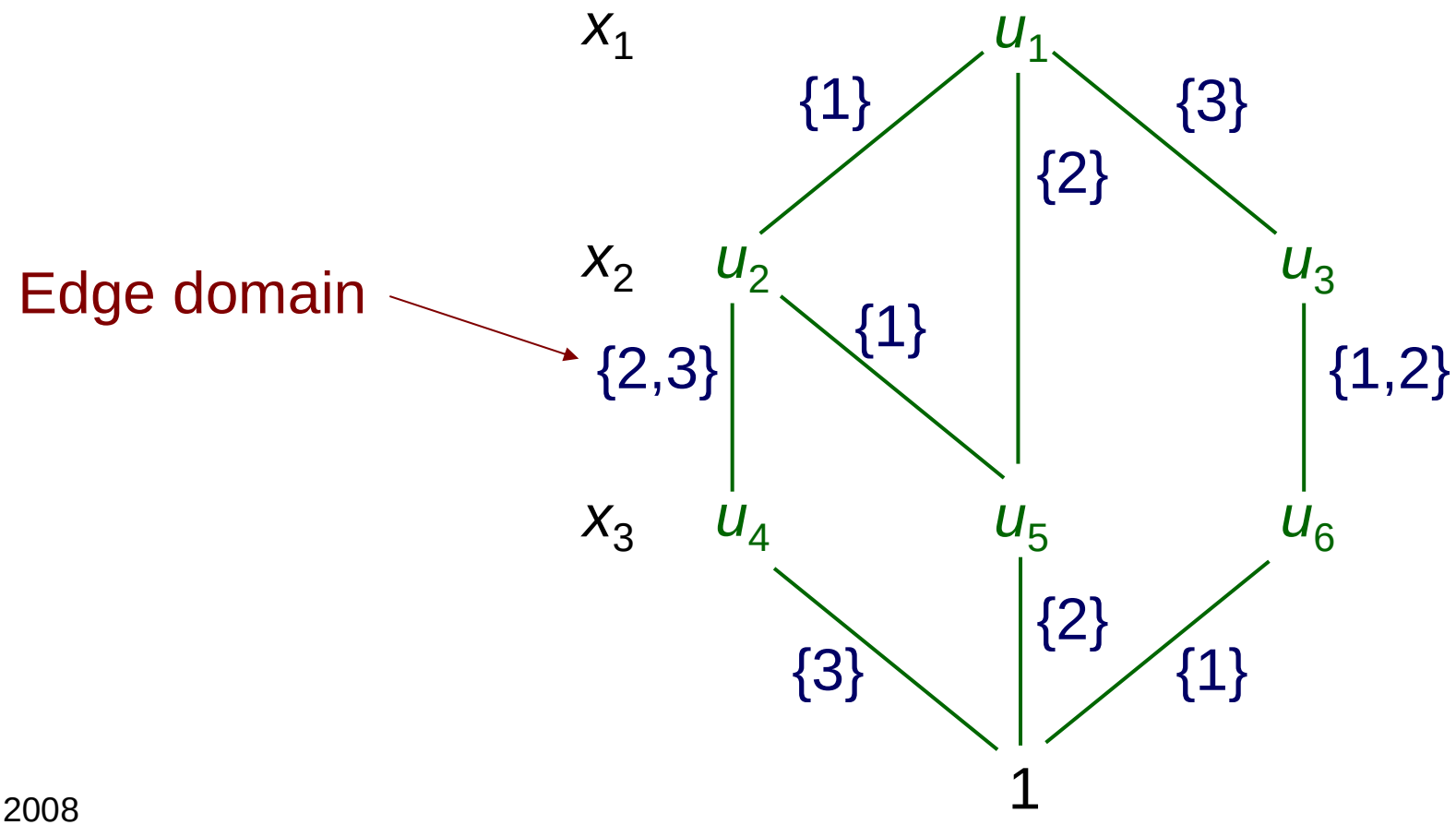
What is the reduced ordered MDD for

$$\begin{pmatrix} x_1 + x_3 = 4 \\ 3 \le x_1 + x_2 \le 5 \end{pmatrix} \vee \ (x_1, x_2, x_3) = (1,1,2)$$

$$x_j \in \{1,2,3\}$$

Multivalued Decision Diagrams

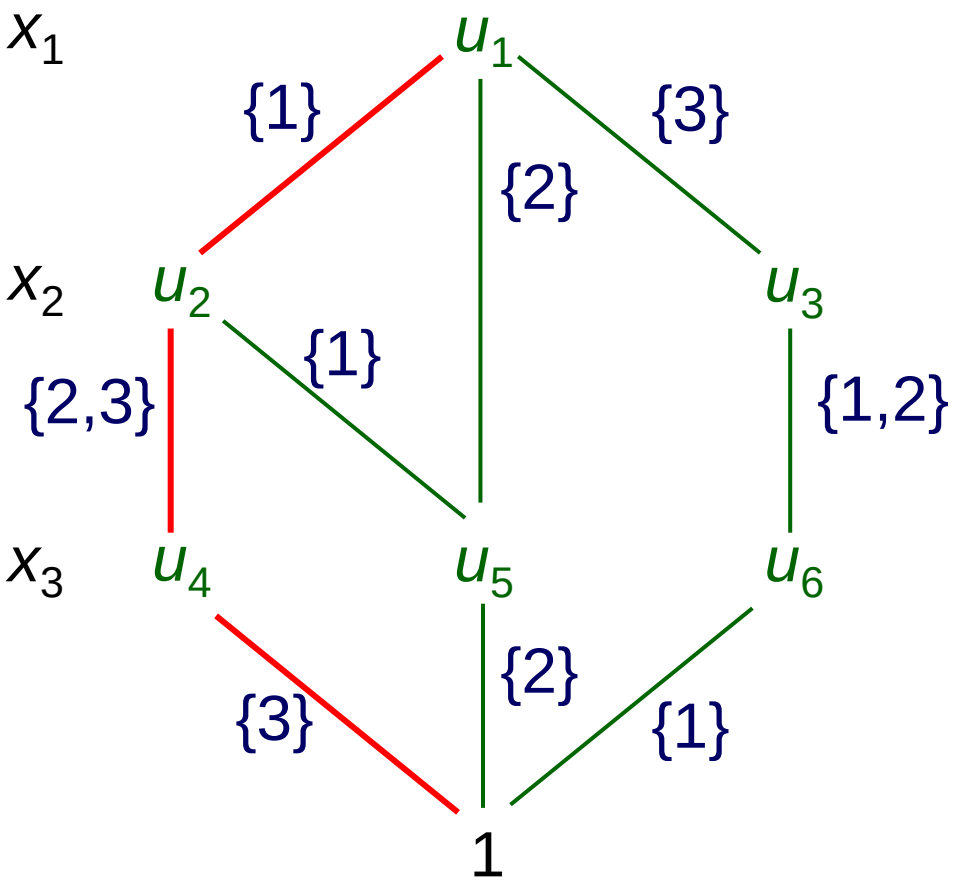
$$\begin{pmatrix} x_1 + x_3 = 4 \\ 3 \le x_1 + x_2 \le 5 \end{pmatrix} \vee \; (x_1, x_2, x_3) = (1,1,2)$$



Multivalued Decision Diagrams

Each path represents a cartesian product of solutions

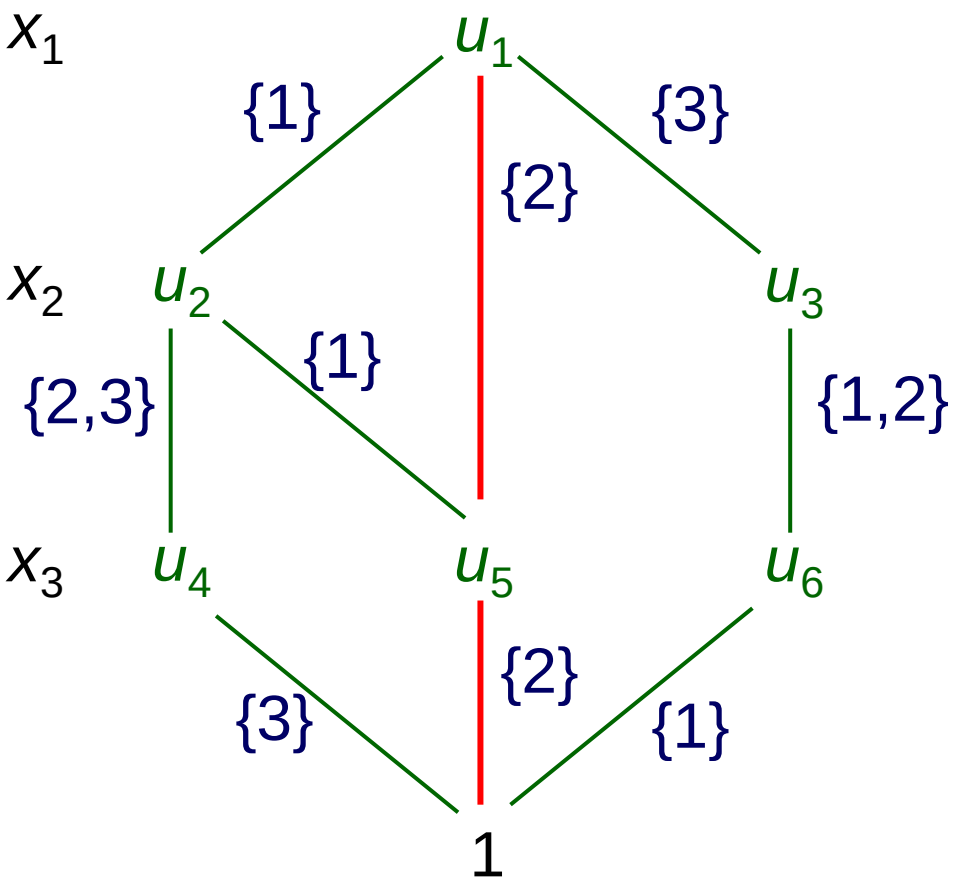
$\{1\}\times\{2,3\}\times\{3\}$



Multivalued Decision Diagrams

Each path represents a cartesian product of solutions

$\{2\}\times\{1,2,3\}\times\{2\}$

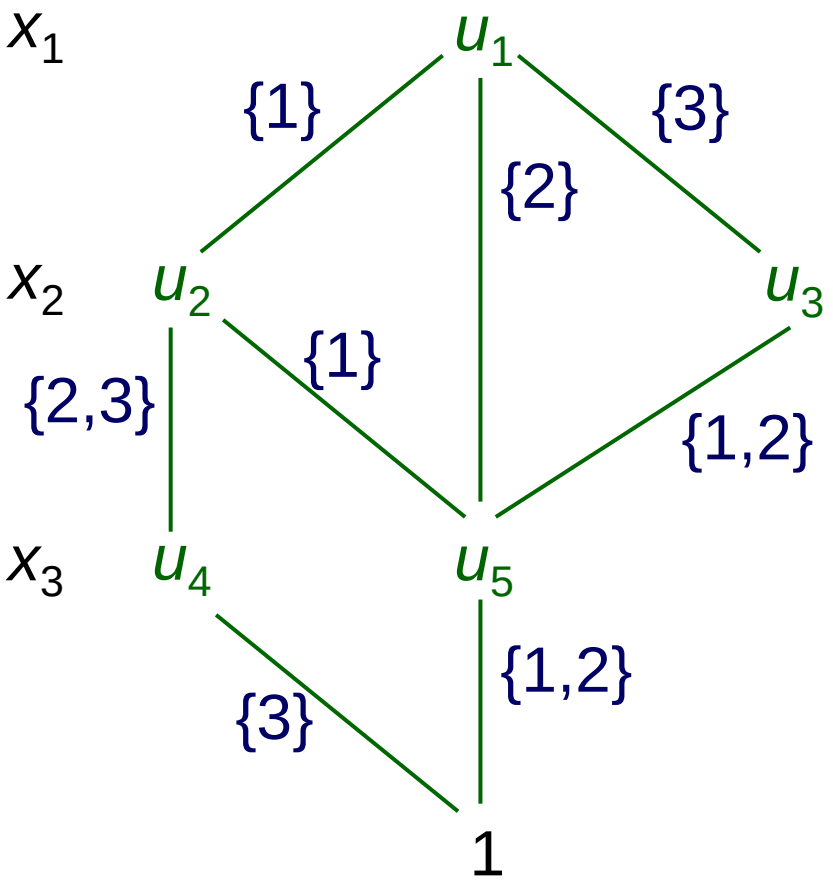


MDD Relaxation

Let's use relaxed MDD
with max width of 2

Full MDD:
8 solutions

Relaxed MDD:
14 solutions



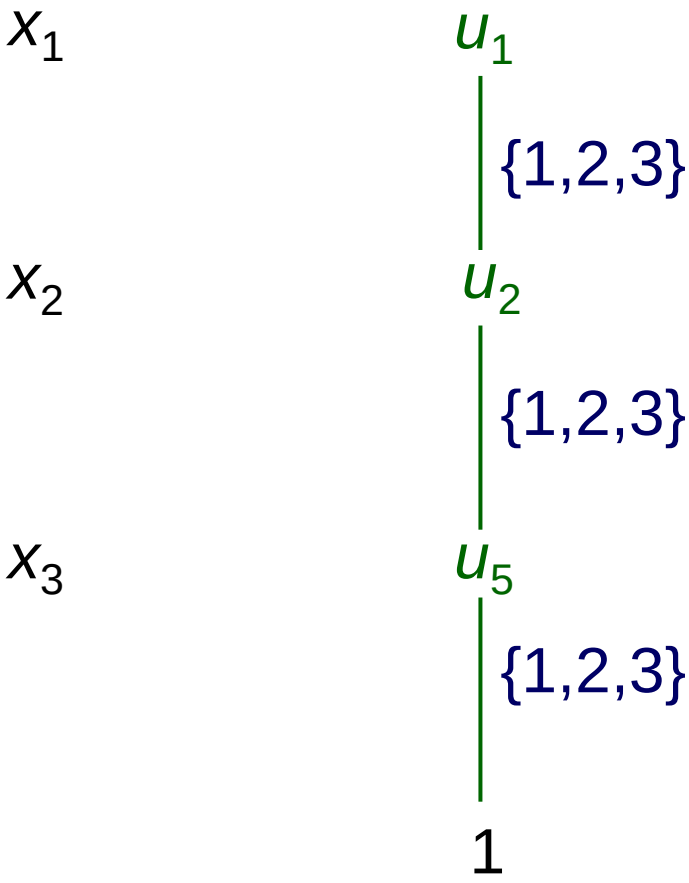
MDD Relaxation

MDD relaxation with **width 1** is just the **domain store**.

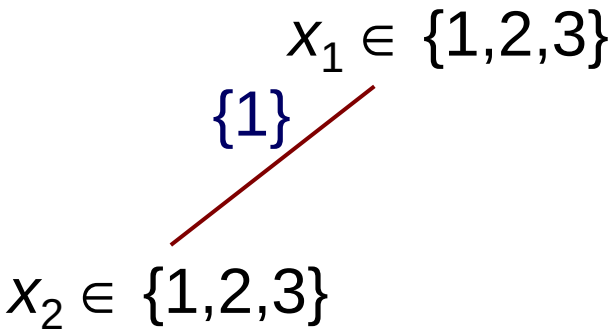
Full MDD:
8 solutions

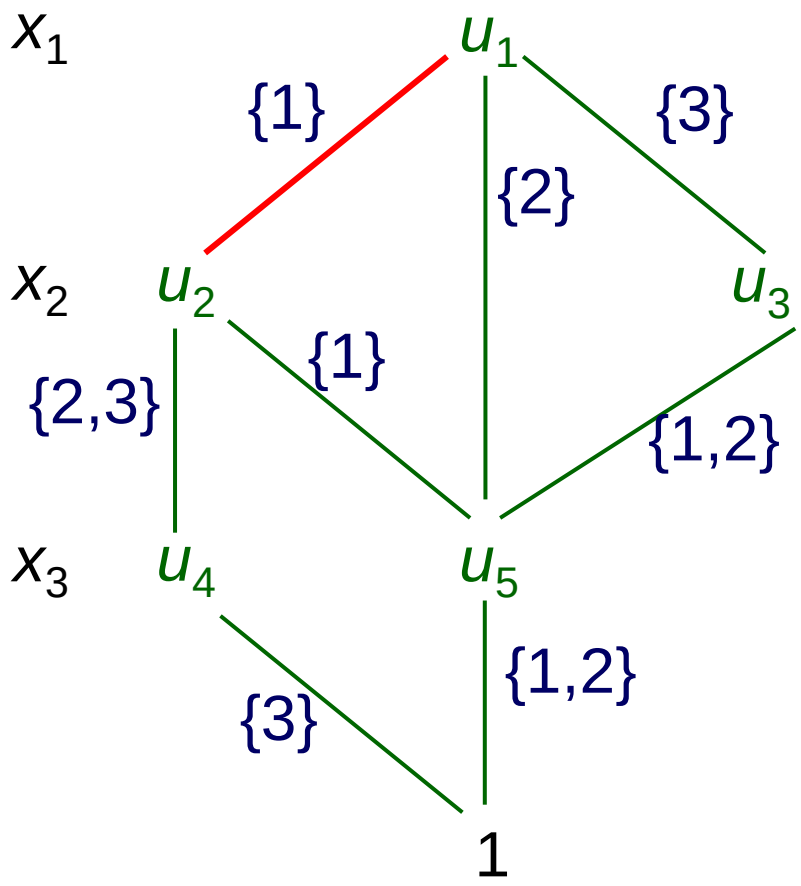
Relaxed MDD:
14 solutions

Domain store:
3×3×3 = 27 solutions

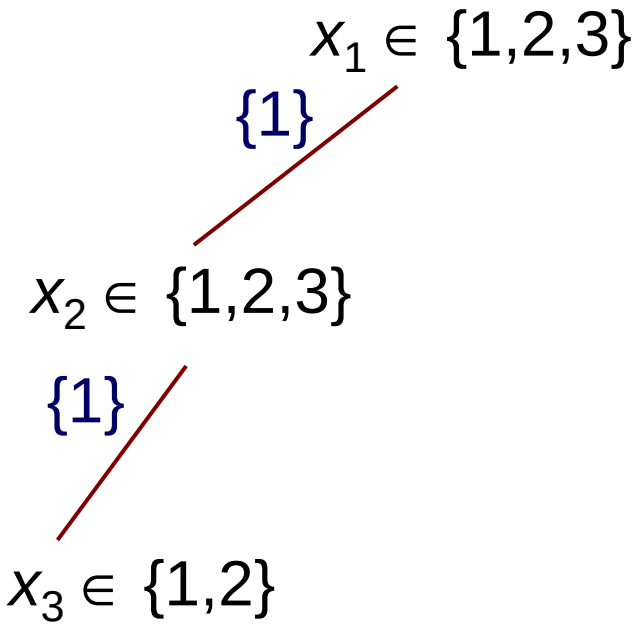


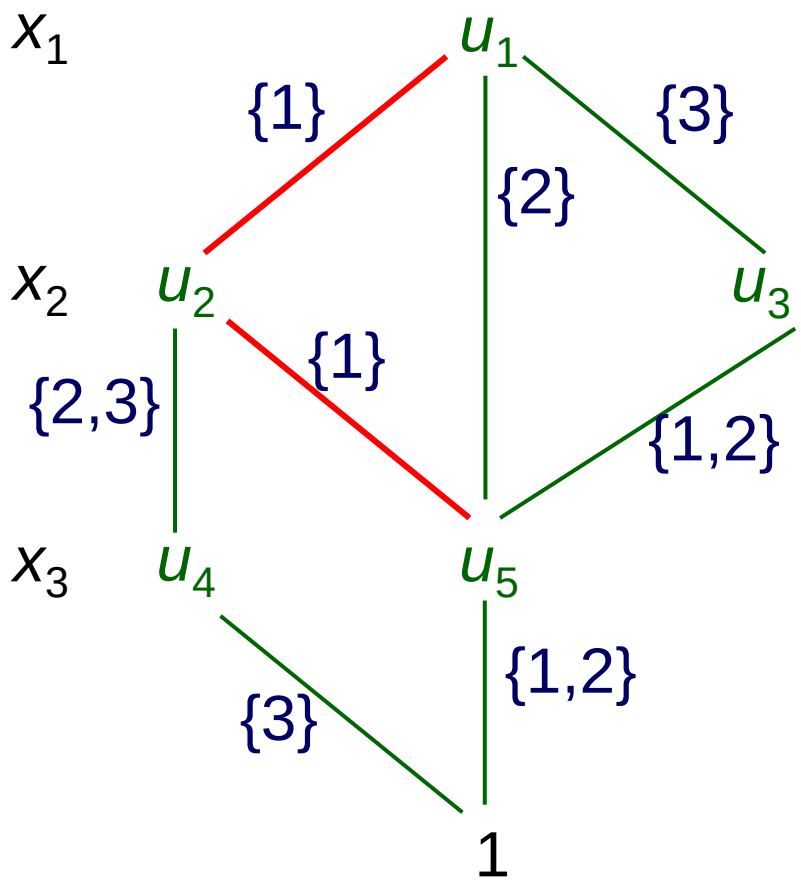
Branching search using relaxed MDD



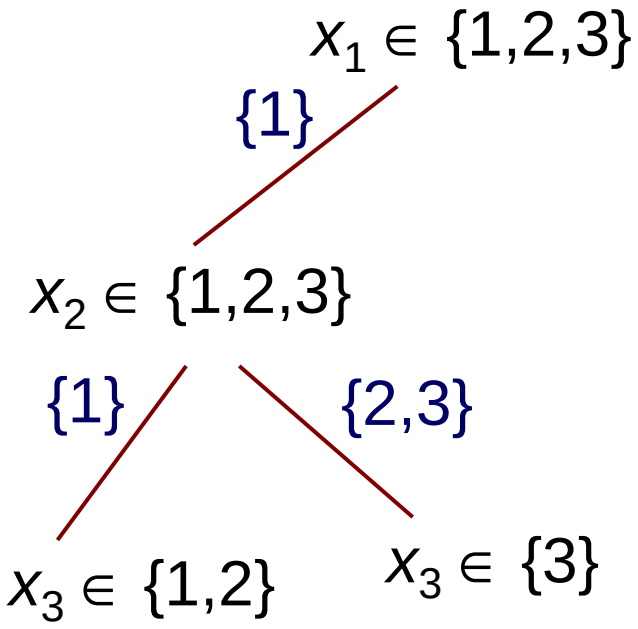


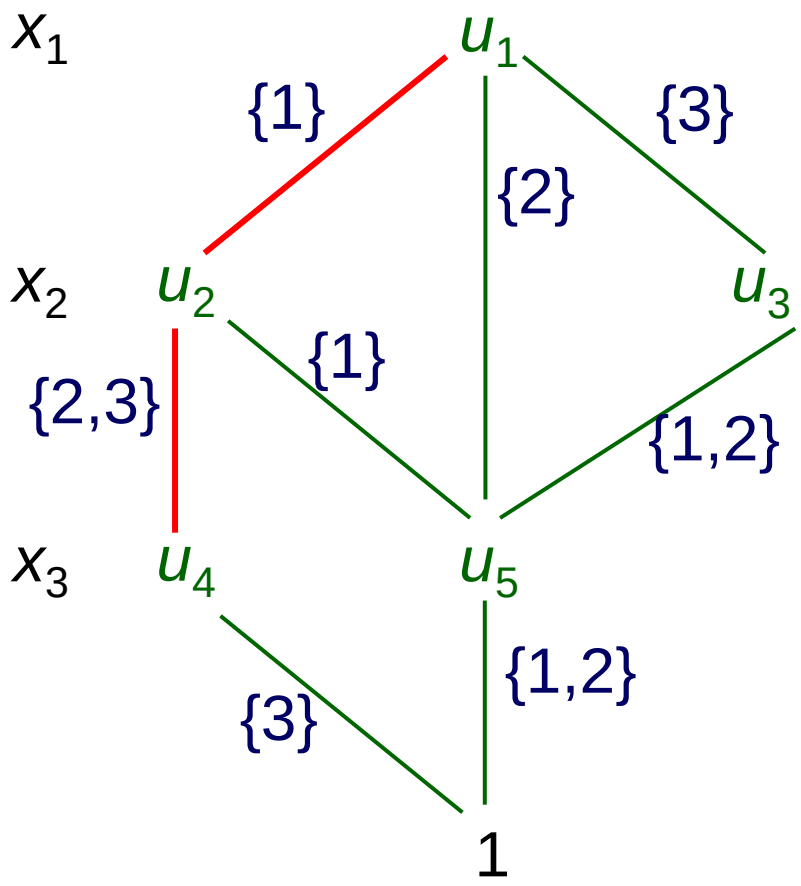
Branching search using relaxed MDD



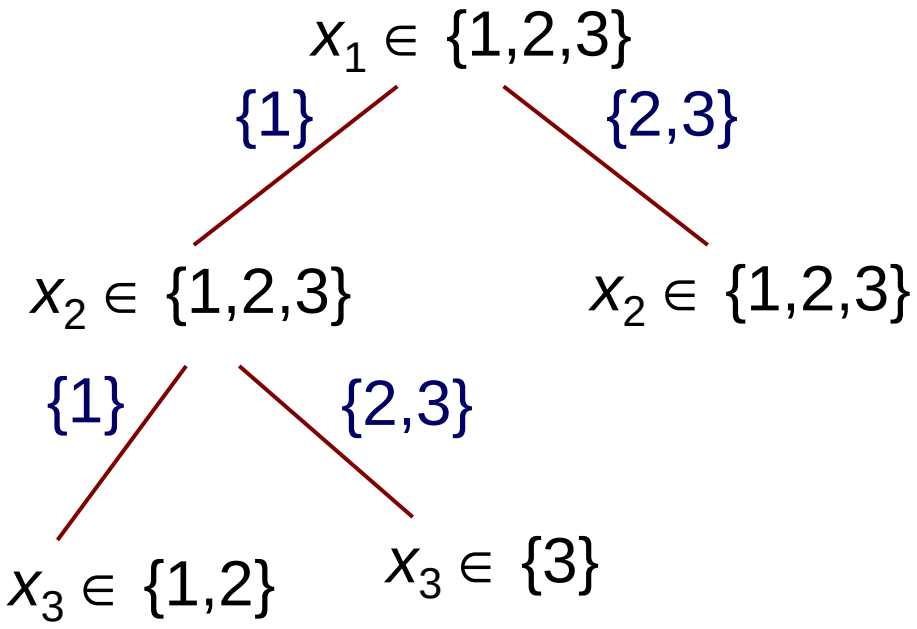


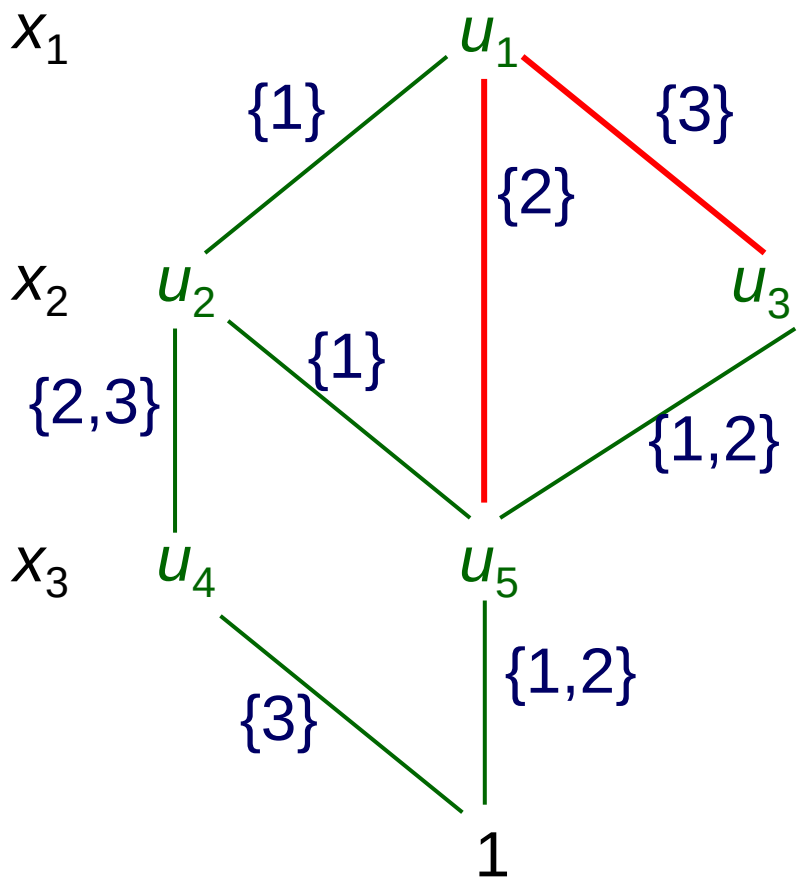
Branching search using relaxed MDD





Branching search using relaxed MDD





And so forth.

Less branching than with domain store.

Propagation in MDDs

We can **propagate** a constraint in an MDD relaxation by **edge domain filtering** and **refinement** (node splitting).

We take care not to exceed max width.

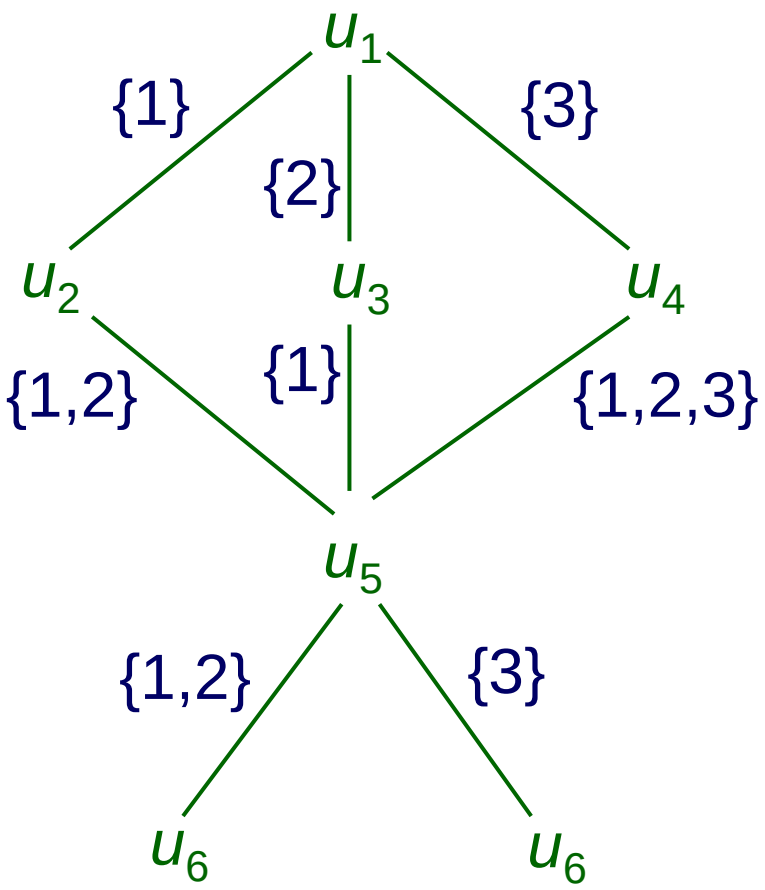
Example:

We will propagate **alldiff** in an MDD relaxation of width 3.

Andersen, Hadzic, Hooker & Tiedemann, A constraint store based on multivalued decision diagrams, 2007

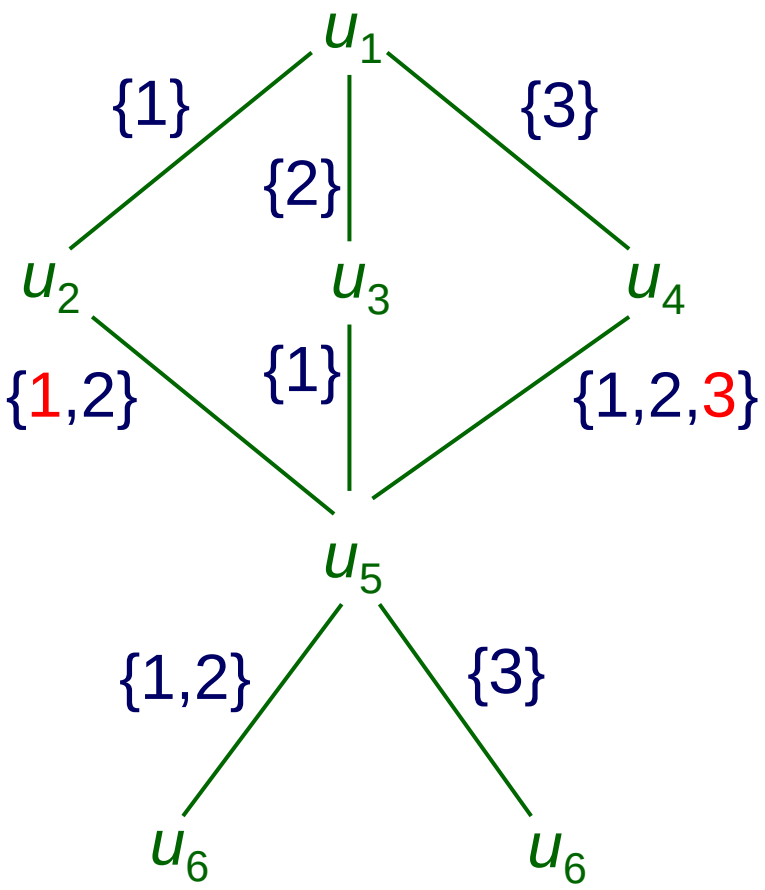
Propagation in MDDs

Current MDD relaxation



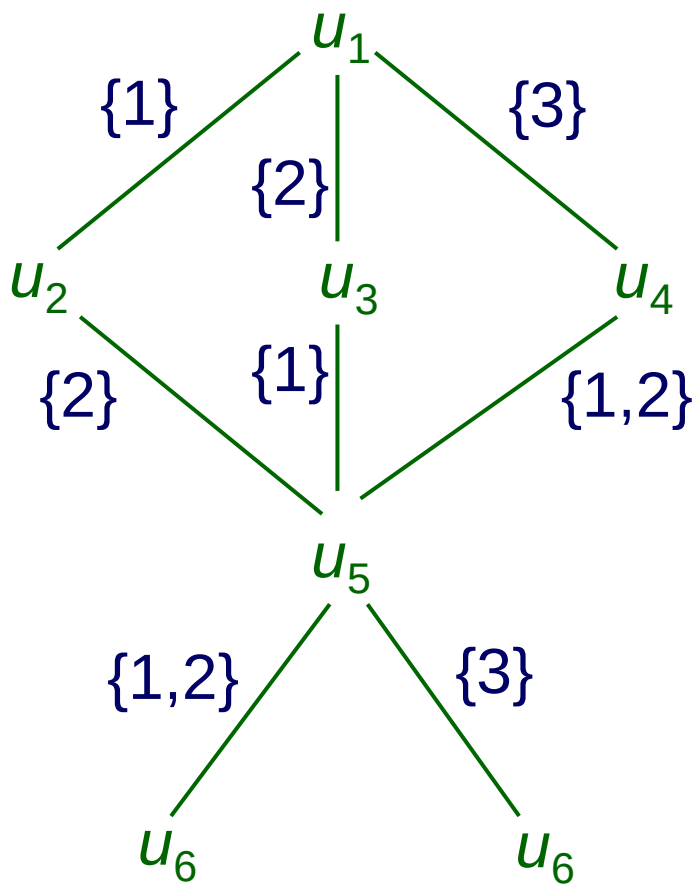
Propagation in MDDs

First filter edge domains using alldiff



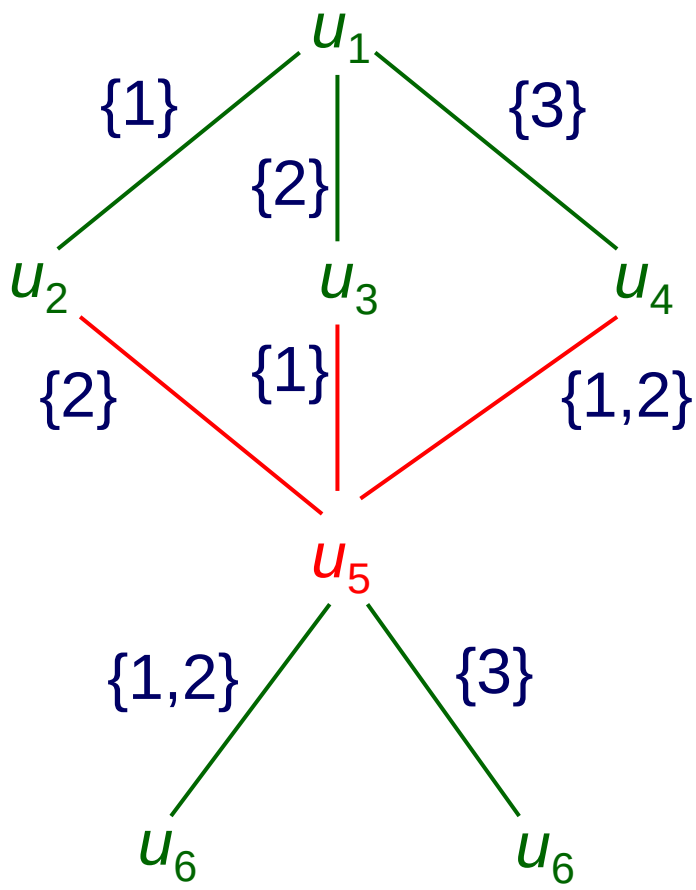
Propagation in MDDs

First filter edge domains using alldiff



Propagation in MDDs

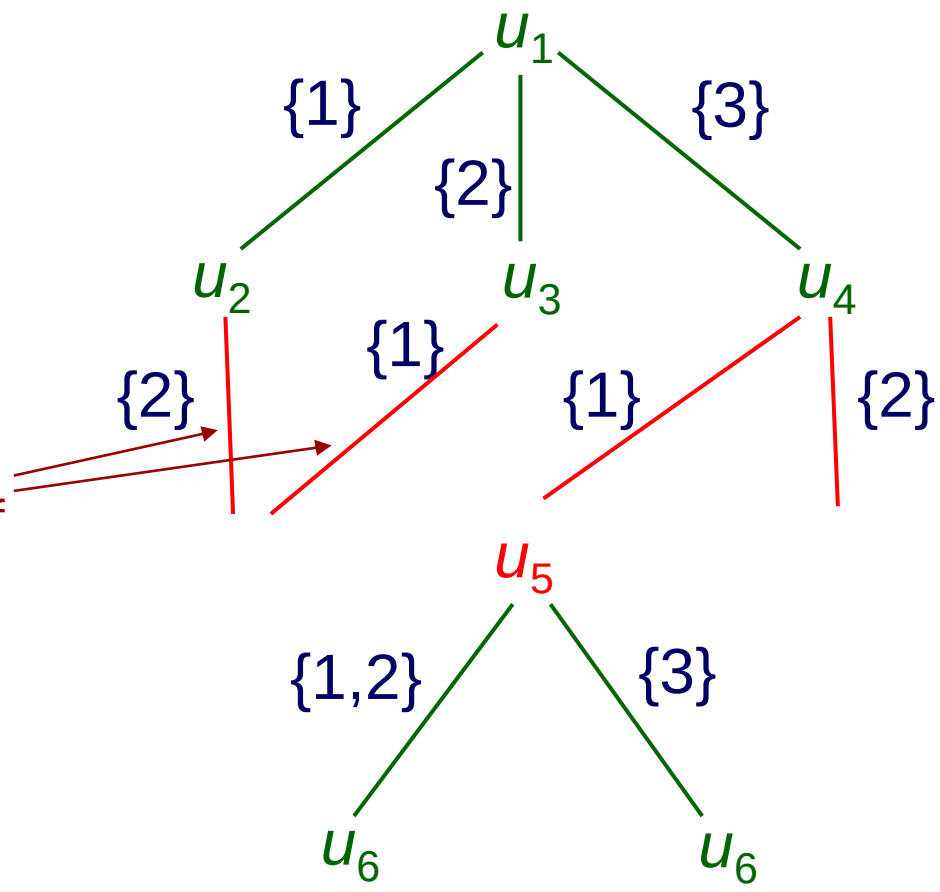
To split u_5: Identify equivalence classes of incoming edges



Propagation in MDDs

To split u_5: Identify equivalence classes of incoming edges

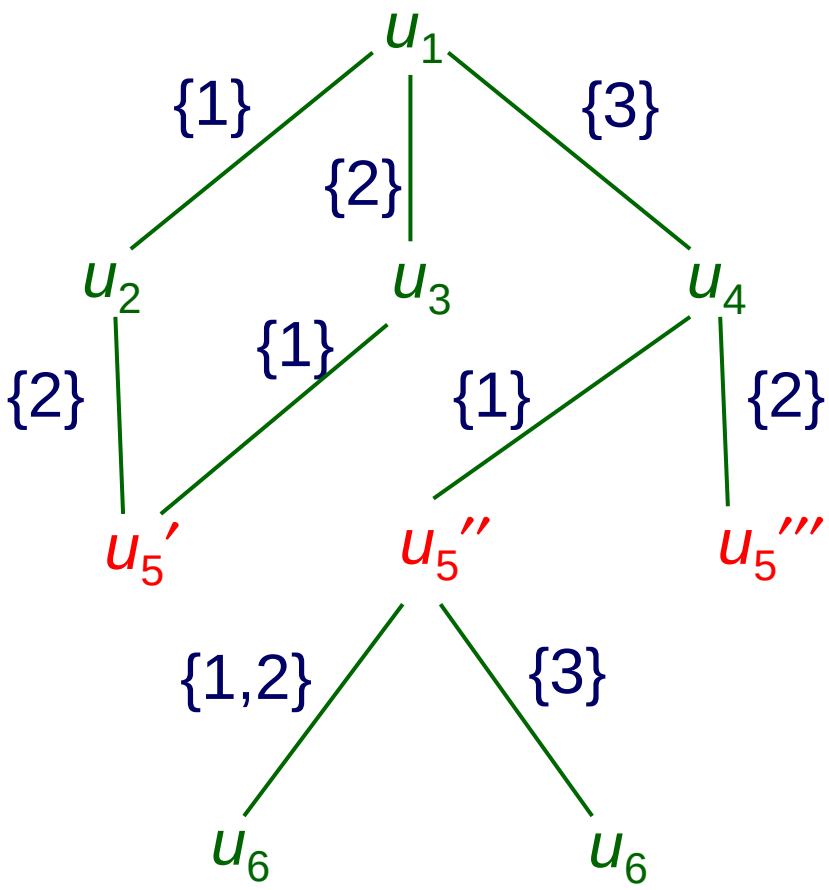
These are equivalent for alldiff because they lead to the same set of feasible paths.



Propagation in MDDs

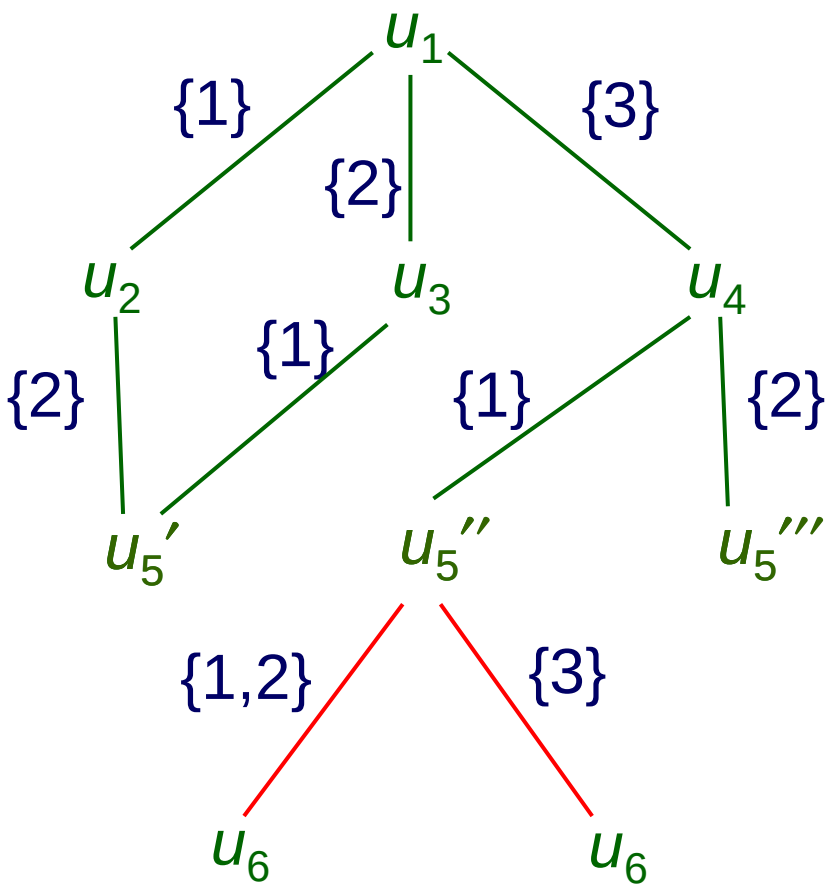
To split u_5: Identify equivalence classes of incoming edges.

Split u_5 to receive ≤ 3 equivalence classes.



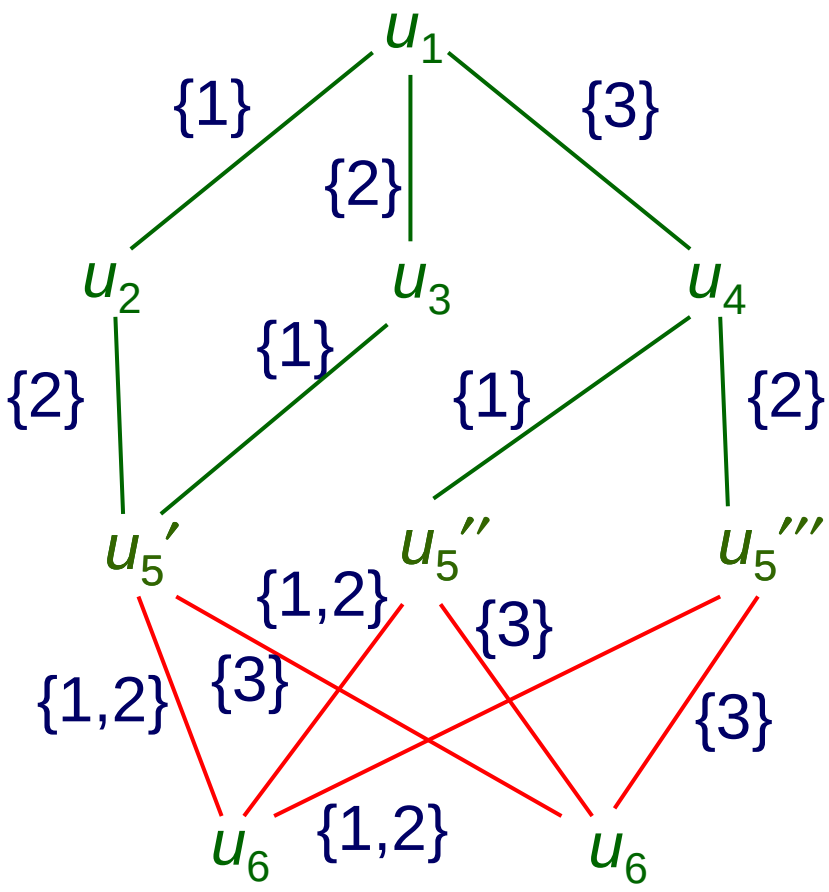
Propagation in MDDs

Duplicate outgoing edges.



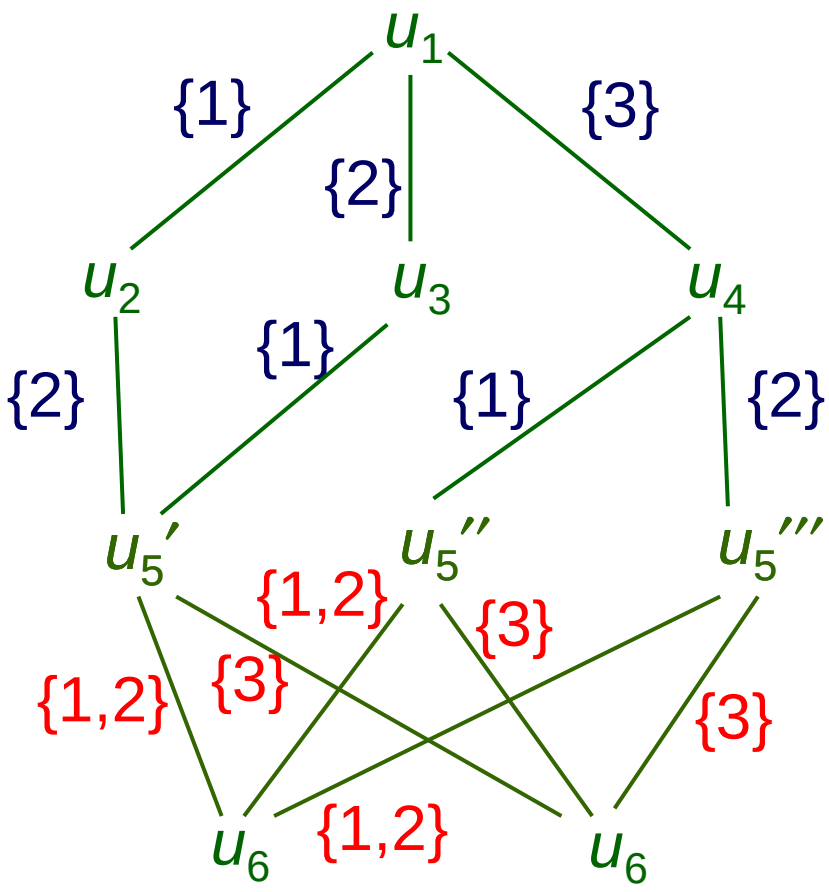
Propagation in MDDs

Duplicate outgoing edges.



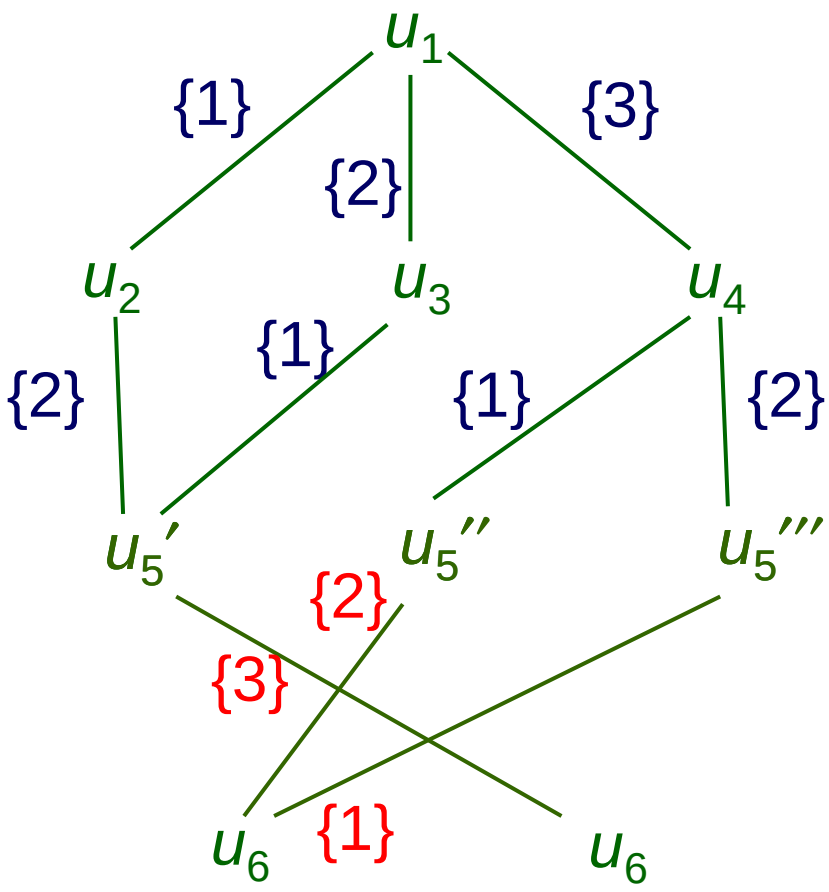
Propagation in MDDs

Filter domains.



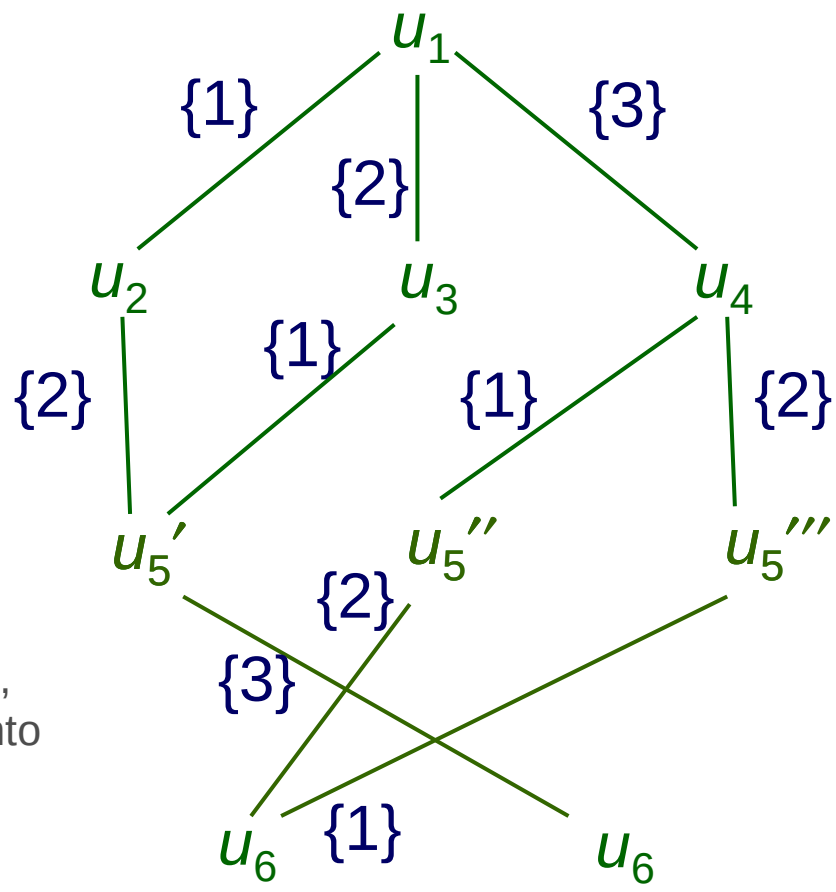
Propagation in MDDs

Filter domains.



Propagation in MDDs

Alldiff has now been propagated.



Hadzic, Hooker, O'Sullivan & Tiedemann, Approximate compilation of constraints into multivalued decision diagrams, 2008

That’s it.
Have a relaxing day!