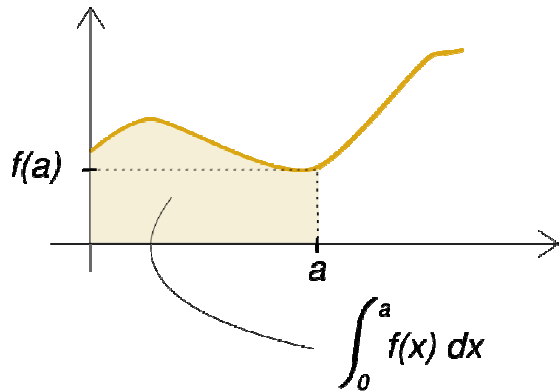


Tutorial: Operations Research in Constraint Programming

John Hooker
Carnegie Mellon University

May 2009
Revised June 2009



Why Integrate OR and CP?

Complementary strengths
Computational advantages
Outline of the Tutorial

Complementary Strengths

- CP:
 - Inference methods
 - Modeling
 - Exploits local structure
- OR:
 - Relaxation methods
 - Duality theory
 - Exploits global structure

Let's bring them
together!



Computational Advantage of Integrating CP and OR

Using CP + relaxation from MILP

	<i>Problem</i>	<i>Speedup</i>
Focacci, Lodi, Milano (1999)	Lesson timetabling	2 to 50 times faster than CP
Refalo (1999)	Piecewise linear costs	2 to 200 times faster than MILP
Hooker & Osorio (1999)	Flow shop scheduling, etc.	4 to 150 times faster than MILP.
Thorsteinsson & Ottosson (2001)	Product configuration	30 to 40 times faster than CP, MILP

Computational Advantage of Integrating CP and MILP

Using CP + relaxation from MILP

	<i>Problem</i>	<i>Speedup</i>
Sellmann & Fahle (2001)	Automatic recording	1 to 10 times faster than CP, MILP
Van Hoesve (2001)	Stable set problem	Better than CP in less time
Bollapragada, Ghattas & Hooker (2001)	Structural design (nonlinear)	Up to 600 times faster than MILP. 2 problems: <6 min vs >20 hrs for MILP
Beck & Refalo (2003)	Scheduling with earliness & tardiness costs	Solved 67 of 90, CP solved only 12

Computational Advantage of Integrating CP and MILP

Using CP-based Branch and Price

	<i>Problem</i>	<i>Speedup</i>
Yunes, Moura & de Souza (1999)	Urban transit crew scheduling	Optimal schedule for 210 trips, vs. 120 for traditional branch and price
Easton, Nemhauser & Trick (2002)	Traveling tournament scheduling	First to solve 8-team instance

Computational Advantage of Integrating CP and MILP

Using CP/MILP Benders methods

	<i>Problem</i>	<i>Speedup</i>
Jain & Grossmann (2001)	Min-cost planning & scheduling	20 to 1000 times faster than CP, MILP
Thorsteinsson (2001)	Min-cost planning & scheduling	10 times faster than Jain & Grossmann
Timpe (2002)	Polypropylene batch scheduling at BASF	Solved previously insoluble problem in 10 min

Computational Advantage of Integrating CP and MILP

Using CP/MILP Benders methods

	<i>Problem</i>	<i>Speedup</i>
Benoist, Gaudin, Rottembourg (2002)	Call center scheduling	Solved twice as many instances as traditional Benders
Hooker (2004)	Min-cost, min-makespan planning & cumulative scheduling	100-1000 times faster than CP, MILP
Hooker (2005)	Min tardiness planning & cumulative scheduling	10-1000 times faster than CP, MILP

Software for Integrated Methods

- ECLiPSe
 - Exchanges information between ECLiPSEe solver, Xpress-MP
- OPL Studio
 - Combines CPLEX and ILOG CP Optimizer with script language
- Mosel
 - Combines Xpress-MP, Xpress-Kalis with low-level modeling
- BARON
 - Global optimization with relaxation + domain reduction
- SIMPL
 - Full integration with high-level modeling (prototype)
- SCIP
 - Combines MILP and CP-based propagation

Outline of the Tutorial

- Why Integrate OR and CP?
- A Glimpse at CP
- Initial Example: Integrated Methods
- CP Concepts
- CP Filtering Algorithms
- Linear Relaxation and CP
- Mixed Integer/Linear Modeling
- Cutting Planes
- Lagrangean Relaxation and CP
- Dynamic Programming in CP
- CP-based Branch and Price
- CP-based Benders Decomposition

Detailed Outline

- Why Integrate OR and CP?
 - Complementary strengths
 - Computational advantages
 - Outline of the tutorial
- A Glimpse at CP
 - Early successes
 - Advantages and disadvantages
- Initial Example: Integrated Methods
 - Freight Transfer
 - Bounds Propagation
 - Cutting Planes
 - Branch-infer-and-relax Tree

Detailed Outline

- CP Concepts
 - Consistency
 - Hyperarc Consistency
 - Modeling Examples
- CP Filtering Algorithms
 - Element
 - Alldiff
 - Disjunctive Scheduling
 - Cumulative Scheduling
- Linear Relaxation and CP
 - Why relax?
 - Algebraic Analysis of LP
 - Linear Programming Duality
 - LP-Based Domain Filtering
 - Example: Single-Vehicle Routing
 - Disjunctions of Linear Systems

Detailed Outline

- Mixed Integer/Linear Modeling
 - MILP Representability
 - 4.2 Disjunctive Modeling
 - 4.3 Knapsack Modeling
- Cutting Planes
 - 0-1 Knapsack Cuts
 - Gomory Cuts
 - Mixed Integer Rounding Cuts
 - Example: Product Configuration
- Lagrangean Relaxation and CP
 - Lagrangean Duality
 - Properties of the Lagrangean Dual
 - Example: Fast Linear Programming
 - Domain Filtering
 - Example: Continuous Global Optimization

Detailed Outline

- Dynamic Programming in CP
 - Example: Capital Budgeting
 - Domain Filtering
 - Recursive Optimization
- CP-based Branch and Price
 - Basic Idea
 - Example: Airline Crew Scheduling
- CP-based Benders Decomposition
 - Benders Decomposition in the Abstract
 - Classical Benders Decomposition
 - Example: Machine Scheduling

Background Reading



This tutorial is based on:

- J. N. Hooker, *Integrated Methods for Optimization*, Springer (2007). Contains 295 exercises.
- J. N. Hooker, Operations research methods in constraint programming, in F. Rossi, P. van Beek and T. Walsh, eds., *Handbook of Constraint Programming*, Elsevier (2006), pp. 527-570.



A Glimpse at Constraint Programming

Early Successes

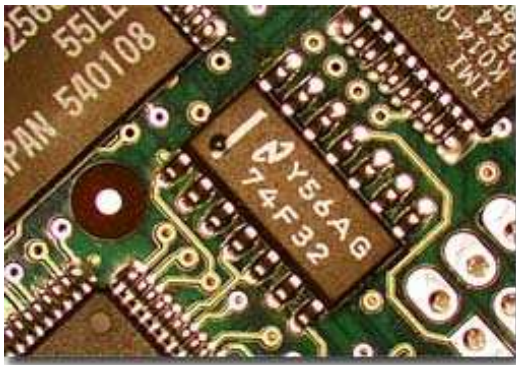
Advantages and Disadvantages

What is constraint programming?

- It is a relatively new technology developed in the computer science and artificial intelligence communities.
- It has found an important role in scheduling, logistics and supply chain management.

Early commercial successes

- Circuit design (Siemens)



- Real-time control (Siemens, Xerox)



- Container port scheduling (Hong Kong and Singapore)



Applications

- Job shop scheduling
- Assembly line smoothing and balancing
- Cellular frequency assignment
- Nurse scheduling
- Shift planning
- Maintenance planning
- Airline crew rostering and scheduling
- Airport gate allocation and stand planning



Applications

- Production scheduling
 - chemicals
 - aviation
 - oil refining
 - steel
 - lumber
 - photographic plates
 - tires
- Transport scheduling (food, nuclear fuel)
- Warehouse management
- Course timetabling



Advantages and Disadvantages

CP vs. Mathematical Programming

MP	CP
Numerical calculation	Logic processing
Relaxation	Inference (filtering, constraint propagation)
Atomistic modeling (linear inequalities)	High-level modeling (global constraints)
Branching	Branching
Independence of model and algorithm	Constraint-based processing

Programming \neq programming

- In **constraint programming**:
 - *programming* = a form of computer programming (constraint-based processing)
- In **mathematical programming**:
 - *programming* = logistics planning (historically)

CP vs. MP

- In **mathematical programming**, equations (constraints) describe the problem but don't tell how to solve it.
- In **constraint programming**, each constraint invokes a procedure that screens out unacceptable solutions.
 - Much as each line of a computer program invokes an operation.

Advantages of CP

- Better at sequencing and scheduling
 - ...where MP methods have weak relaxations.
- Adding messy constraints makes the problem easier.
 - The more constraints, the better.
- More powerful modeling language.
 - Global constraints lead to succinct models.
 - Constraints convey problem structure to the solver.
- “Better at highly-constrained problems”
 - Misleading – better when constraints propagate well, or when constraints have few variables.

Disdvantages of CP

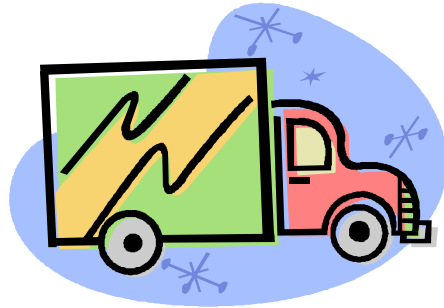
- Weaker for continuous variables.
 - Due to lack of numerical techniques
- May fail when constraints contain many variables.
 - These constraints don't propagate well.
- Often not good for finding optimal solutions.
 - Due to lack of relaxation technology.
- May not scale up
 - Discrete combinatorial methods
- Software is not robust
 - Younger field

Obvious solution...

- Integrate CP and MP.

Trends

- CP is better known in continental Europe, Asia.
 - Less known in North America, seen as threat to OR.
- CP/MP integration is growing
 - Eclipse, Mozart, OPL Studio, SIMPL, SCIP, BARON
- Heuristic methods increasingly important in CP
 - Discrete combinatorial methods
- MP/CP/heuristics may become a single technology.



Initial Example: Integrated Methods

Freight Transfer
Bounds Propagation
Cutting Planes
Branch-infer-and-relax Tree

Example: Freight Transfer

- Transport 42 tons of freight using 8 trucks, which come in 4 sizes...



Truck size	Number available	Capacity (tons)	Cost per truck
1	3	7	90
2	3	5	60
3	3	4	50
4	3	3	40



Number of trucks of type 1

$$\min 90x_1 + 60x_2 + 50x_3 + 40x_4$$

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

Knapsack
covering
constraint

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

Knapsack
packing
constraint

$$x_i \in \{0, 1, 2, 3\}$$

Truck type	Number available	Capacity (tons)	Cost per truck
1	3	7	90
2	3	5	60
3	3	4	50
4	3	3	40

Bounds propagation



$$\min 90x_1 + 60x_2 + 50x_3 + 40x_4$$

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

$$x_i \in \{0, 1, 2, 3\}$$

$$x_1 \geq \left\lceil \frac{42 - 5 \cdot 3 - 4 \cdot 3 - 3 \cdot 3}{7} \right\rceil = 1$$

Bounds propagation



$$\min 90x_1 + 60x_2 + 50x_3 + 40x_4$$

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

$$x_1 \in \{1, 2, 3\}, \quad x_2, x_3, x_4 \in \{0, 1, 2, 3\}$$

Reduced
domain

$$x_1 \geq \left\lceil \frac{42 - 5 \cdot 3 - 4 \cdot 3 - 3 \cdot 3}{7} \right\rceil = 1$$

Bounds consistency

- Let $\{L_j, \dots, U_j\}$ be the domain of x_j
- A constraint set is **bounds consistent** if for each j :
 - $x_j = L_j$ in some feasible solution and
 - $x_j = U_j$ in some feasible solution.
- Bounds consistency \Rightarrow we will not set x_j to any infeasible values during branching.
- Bounds propagation achieves bounds consistency for a **single inequality**.
 - $7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$ is bounds consistent when the domains are $x_1 \in \{1,2,3\}$ and $x_2, x_3, x_4 \in \{0,1,2,3\}$.
- But not necessarily for a **set** of inequalities.

Bounds consistency

- Bounds propagation may not achieve bounds consistency for a set of constraints.

- Consider set of inequalities
$$x_1 + x_2 \geq 1$$
$$x_1 - x_2 \geq 0$$

with domains $x_1, x_2 \in \{0,1\}$, solutions $(x_1, x_2) = (1,0), (1,1)$.

- Bounds propagation has no effect on the domains.
- But constraint set is not bounds consistent because $x_1 = 0$ in no feasible solution.

Cutting Planes



Begin with continuous relaxation

$$\min 90x_1 + 60x_2 + 50x_3 + 40x_4$$

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

$$0 \leq x_i \leq 3, \quad x_1 \geq 1$$

← Replace domains
with bounds

This is a linear programming problem, which is easy to solve.

Its optimal value provides a lower bound on optimal value of original problem.

Cutting planes (valid inequalities)



$$\min 90x_1 + 60x_2 + 50x_3 + 40x_4$$

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

$$0 \leq x_i \leq 3, \quad x_1 \geq 1$$

We can create a **tighter** relaxation (larger minimum value) with the addition of **cutting planes**.

Cutting planes (valid inequalities)



$$\min 90x_1 + 60x_2 + 50x_3 + 40x_4$$

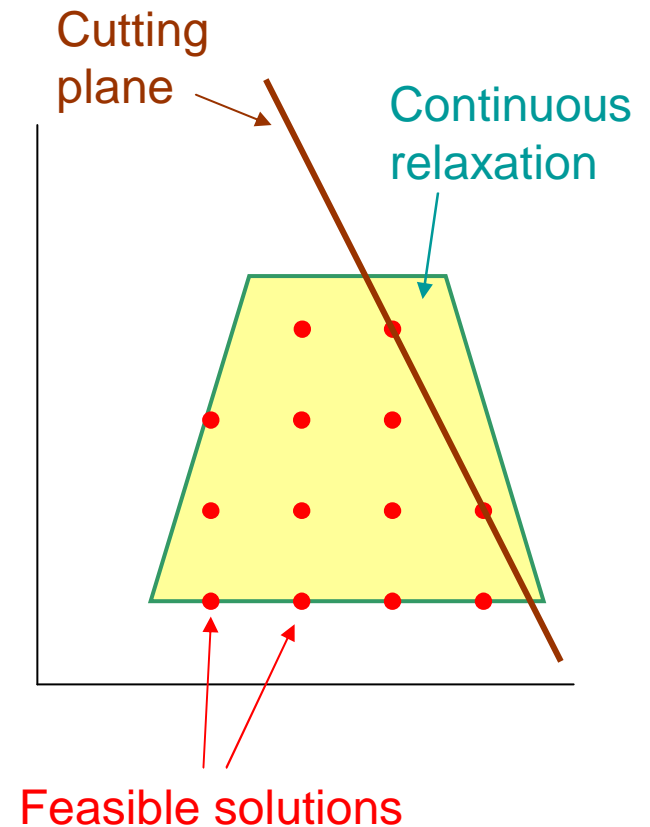
$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

$$0 \leq x_i \leq 3, \quad x_1 \geq 1$$

All feasible solutions of the original problem satisfy a cutting plane (i.e., it is **valid**).

But a cutting plane may exclude (“**cut off**”) solutions of the continuous relaxation.



Cutting planes (valid inequalities)



$$\min 90x_1 + 60x_2 + 50x_3 + 40x_4$$

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

$$0 \leq x_i \leq 3, \quad x_1 \geq 1$$

$\{1,2\}$ is a **packing**

...because $7x_1 + 5x_2$ alone cannot satisfy the inequality, even with $x_1 = x_2 = 3$.

Cutting planes (valid inequalities)



$$\min 90x_1 + 60x_2 + 50x_3 + 40x_4$$

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

$$0 \leq x_i \leq 3, \quad x_1 \geq 1$$

$\{1,2\}$ is a **packing**

So, $4x_3 + 3x_4 \geq 42 - (7 \cdot 3 + 5 \cdot 3)$

which implies

$$x_3 + x_4 \geq \left\lceil \frac{42 - (7 \cdot 3 + 5 \cdot 3)}{\max\{4, 3\}} \right\rceil = 2$$

Knapsack cut

Cutting planes (valid inequalities)



Let x_i have domain $[L_i, U_i]$ and let $a \geq 0$.

In general, a **packing** P for $ax \geq a_0$ satisfies

$$\sum_{i \notin P} a_i x_i \geq a_0 - \sum_{i \in P} a_i U_i$$

and generates a **knapsack cut**

$$\sum_{i \notin P} x_i \geq \left\lceil \frac{a_0 - \sum_{i \in P} a_i U_i}{\max_{i \notin P} \{a_i\}} \right\rceil$$

Cutting planes (valid inequalities)



$$\min 90x_1 + 60x_2 + 50x_3 + 40x_4$$

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

$$0 \leq x_i \leq 3, \quad x_1 \geq 1$$

Maximal Packings	Knapsack cuts
$\{1,2\}$	$x_3 + x_4 \geq 2$
$\{1,3\}$	$x_2 + x_4 \geq 2$
$\{1,4\}$	$x_2 + x_3 \geq 3$

Knapsack cuts corresponding to nonmaximal packings can be nonredundant.

Continuous relaxation with cuts



$$\min 90x_1 + 60x_2 + 50x_3 + 40x_4$$

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

$$0 \leq x_i \leq 3, \quad x_1 \geq 1$$

$$x_3 + x_4 \geq 2$$

$$x_2 + x_4 \geq 2$$

$$x_2 + x_3 \geq 3$$

Knapsack cuts

Optimal value of 523.3 is a lower bound on optimal value of original problem.

Branch- infer-and- relax tree

Propagate bounds
and solve
relaxation of
original problem.

$$\begin{aligned}x_1 &\in \{123\}\\x_2 &\in \{0123\}\\x_3 &\in \{0123\}\\x_4 &\in \{0123\}\\x &= (2\frac{1}{3}, 3, 2\frac{2}{3}, 0)\\ \text{value} &= 523\frac{1}{3}\end{aligned}$$



Branch-infer- and-relax tree

Branch on a
variable with
nonintegral value
in the relaxation.

$$\begin{aligned}x_1 &\in \{1, 2, 3\} \\x_2 &\in \{0, 1, 2, 3\} \\x_3 &\in \{0, 1, 2, 3\} \\x_4 &\in \{0, 1, 2, 3\} \\x &= (2\frac{1}{3}, 3, 2\frac{2}{3}, 0) \\ \text{value} &= 523\frac{1}{3}\end{aligned}$$

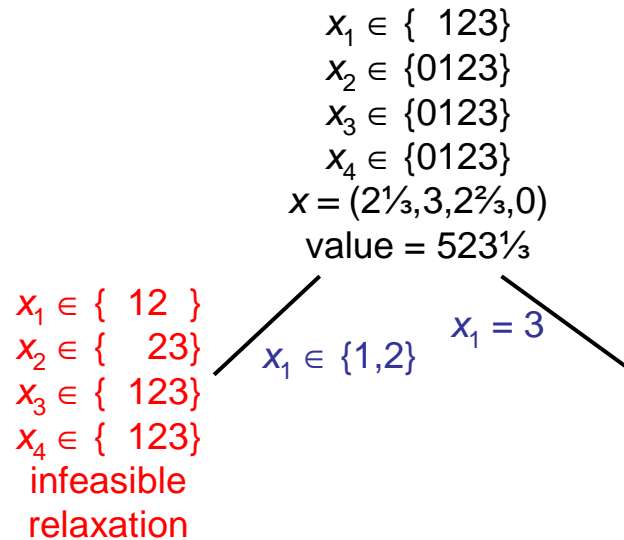
$$\begin{aligned}x_1 &\in \{1, 2\} & x_1 &= 3\end{aligned}$$



Branch-infer- and-relax tree

Propagate bounds
and solve
relaxation.

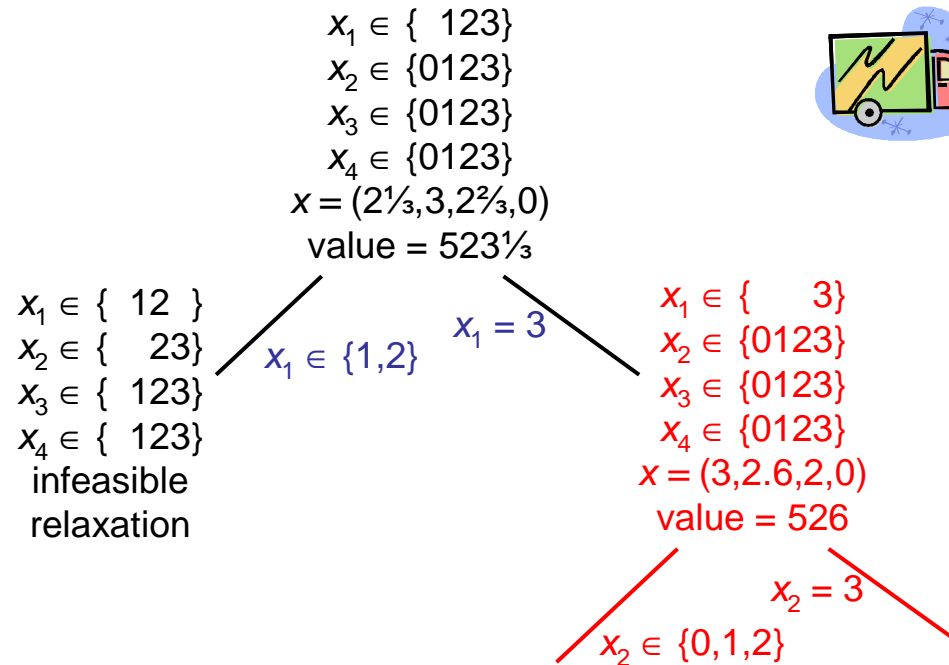
Since relaxation
is infeasible,
backtrack.



Branch-infer- and-relax tree

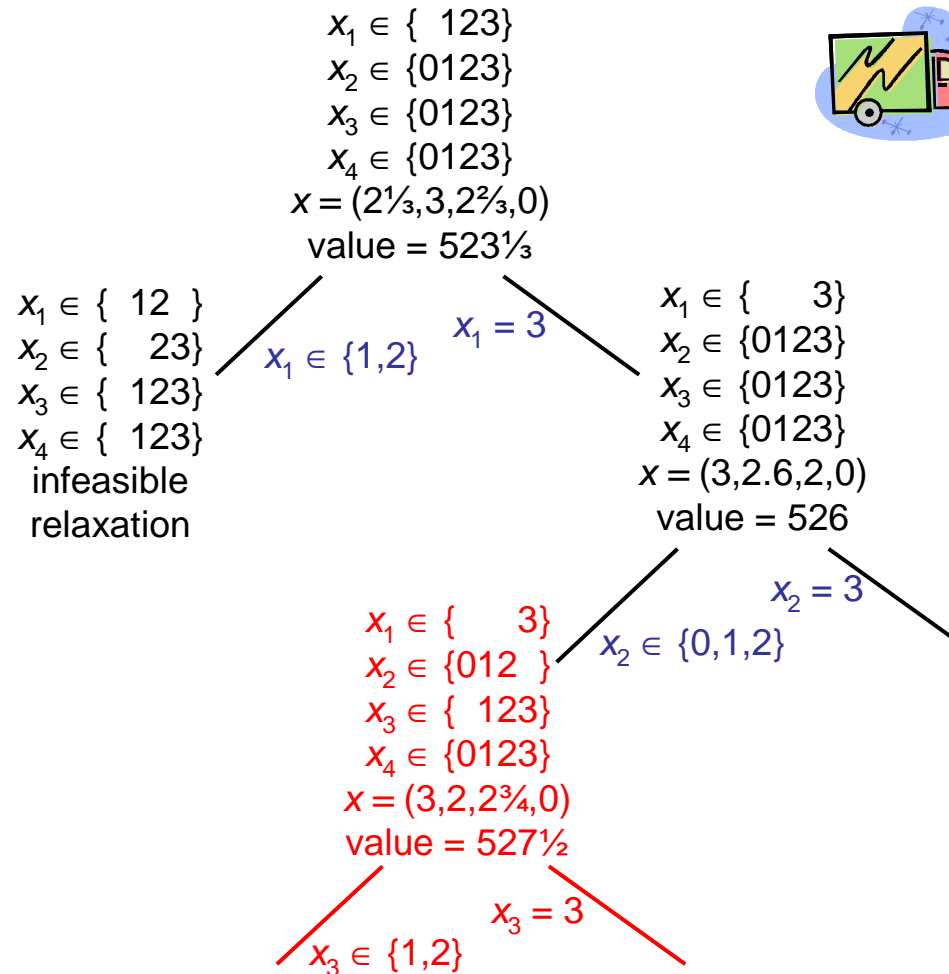
Propagate bounds
and solve
relaxation.

Branch on
nonintegral
variable.



Branch-infer- and-relax tree

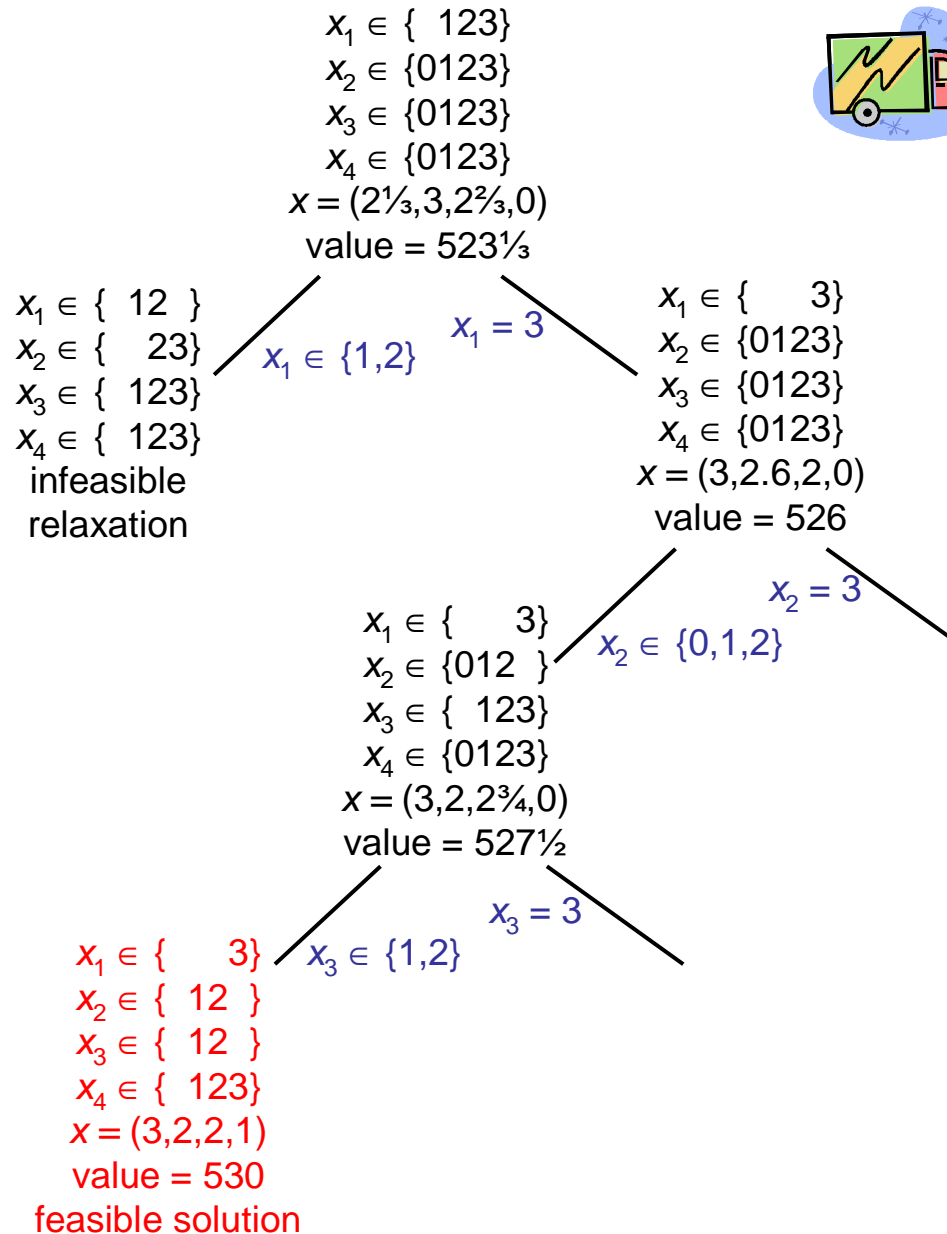
Branch again.



Branch-infer- and-relax tree

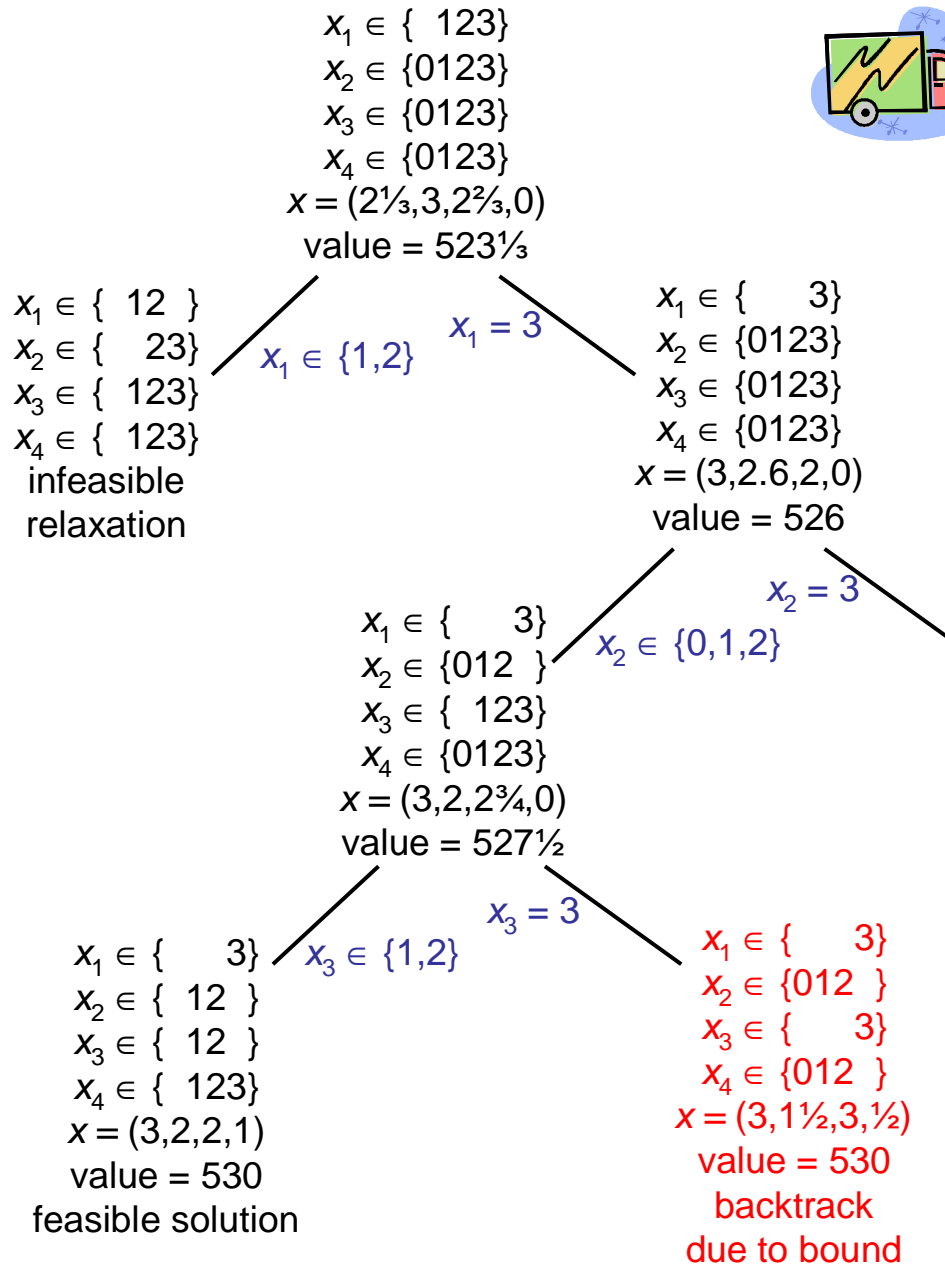
Solution of
relaxation
is integral and
therefore feasible
in the original
problem.

This becomes the
incumbent
solution.



Branch-infer- and-relax tree

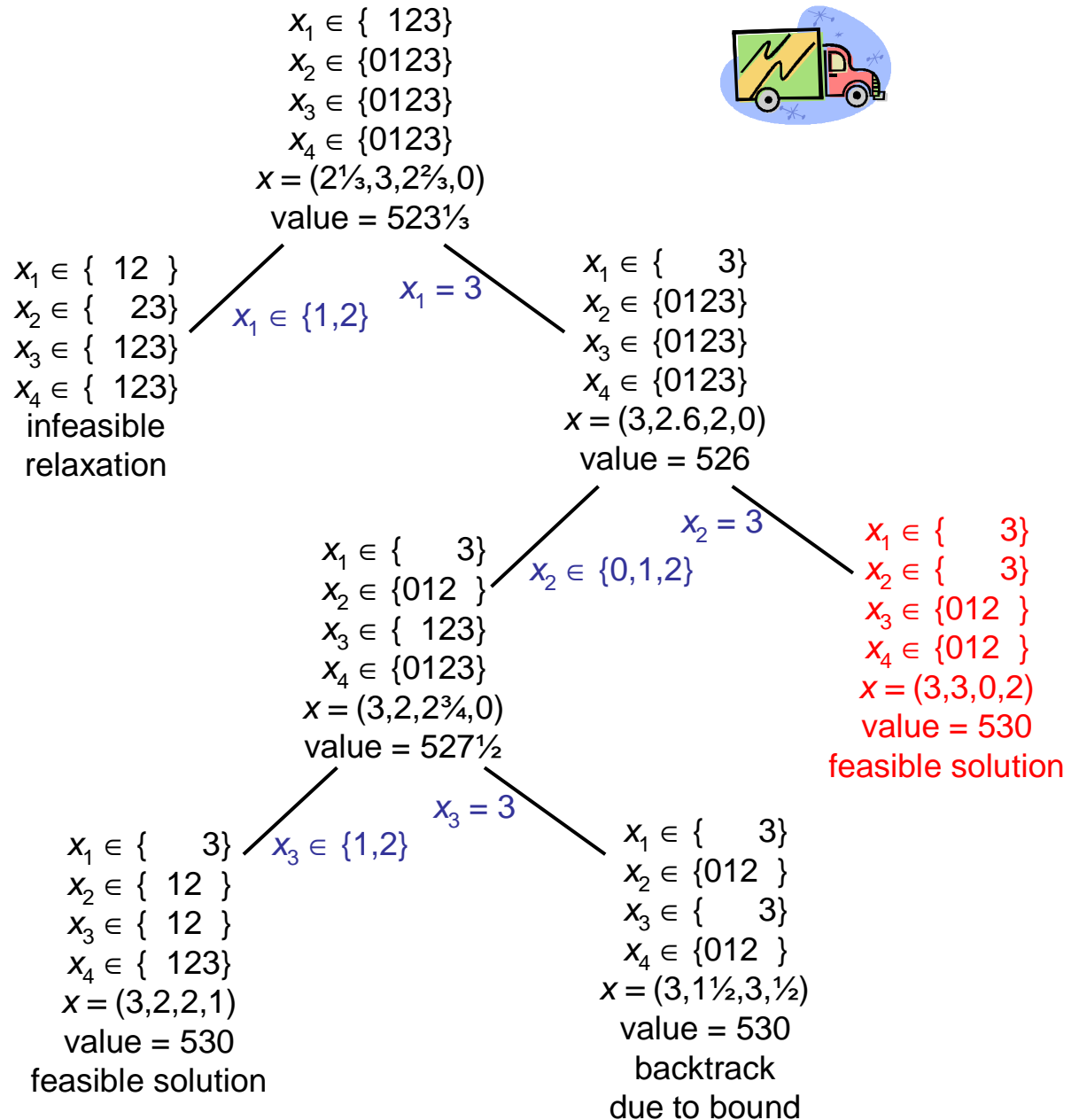
Solution is
nonintegral, but
we can backtrack
because value of
relaxation is
no better than
incumbent solution.



Branch-infer-and-relax tree

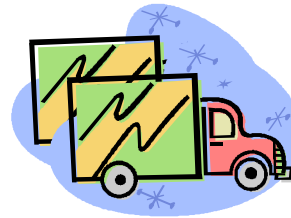
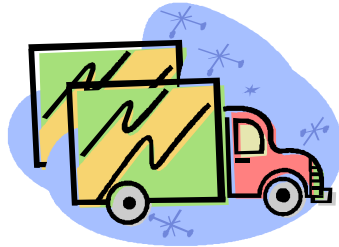
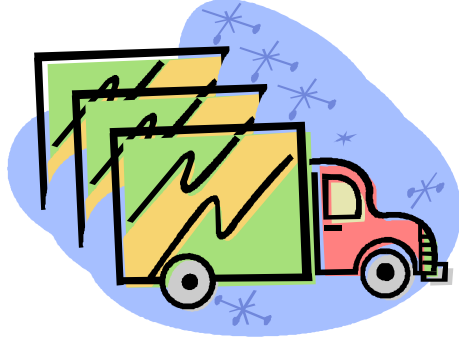
Another feasible solution found.

No better than incumbent solution, which is optimal because search has finished.

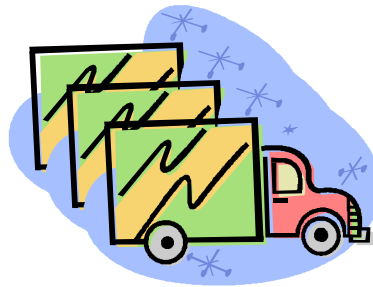
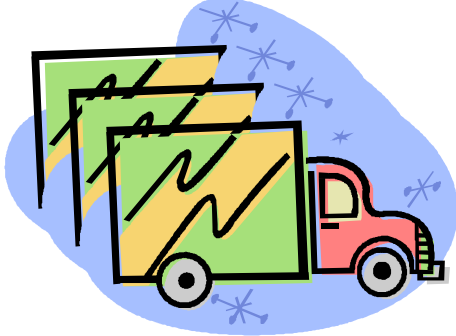


Two optimal solutions...

$$x = (3, 2, 2, 1)$$



$$x = (3, 3, 0, 2)$$





Constraint Programming Concepts

Consistency

Generalized Arc Consistency

Modeling Examples

Consistency

- A constraint set is **consistent** if every partial assignment to the variables that violates no constraint is feasible.
 - i.e., can be extended to a feasible solution.
- Consistency \neq feasibility
 - Consistency means that any infeasible partial assignment is explicitly ruled out by a constraint.
- Fully consistent constraint sets can be solved **without backtracking**.

Consistency

Consider the constraint set

$$x_1 + x_{100} \geq 1$$

$$x_1 - x_{100} \geq 0$$

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

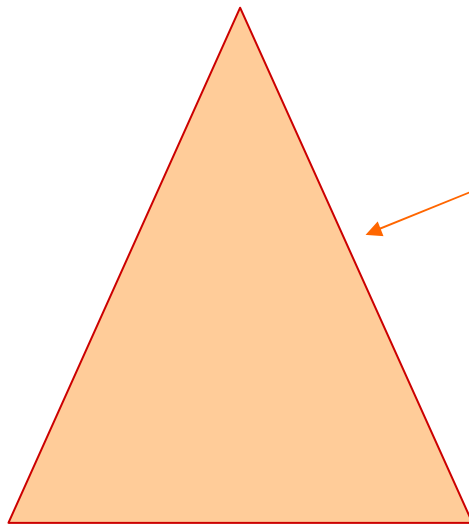
It is not consistent, because $x_1 = 0$ violates no constraint and yet is infeasible (no solution has $x_1 = 0$).

Adding the constraint $x_1 = 1$ makes the set consistent.

$x_1 + x_{100} \geq 1$
 $x_1 - x_{100} \geq 1$
other constraints
 $x_j \in \{0,1\}$

$x_1 = 0$

$x_1 = 1$



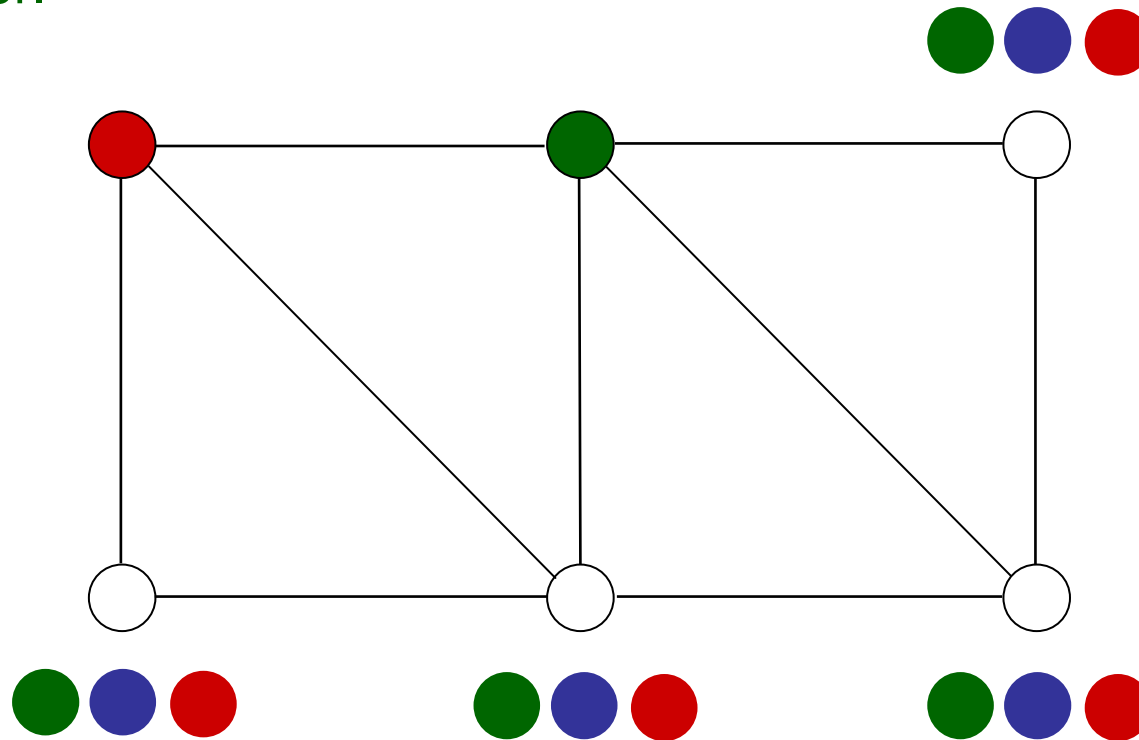
subtree with 2^{99} nodes
but no feasible solution

By adding the constraint
 $x_1 = 1$, the left subtree is
eliminated

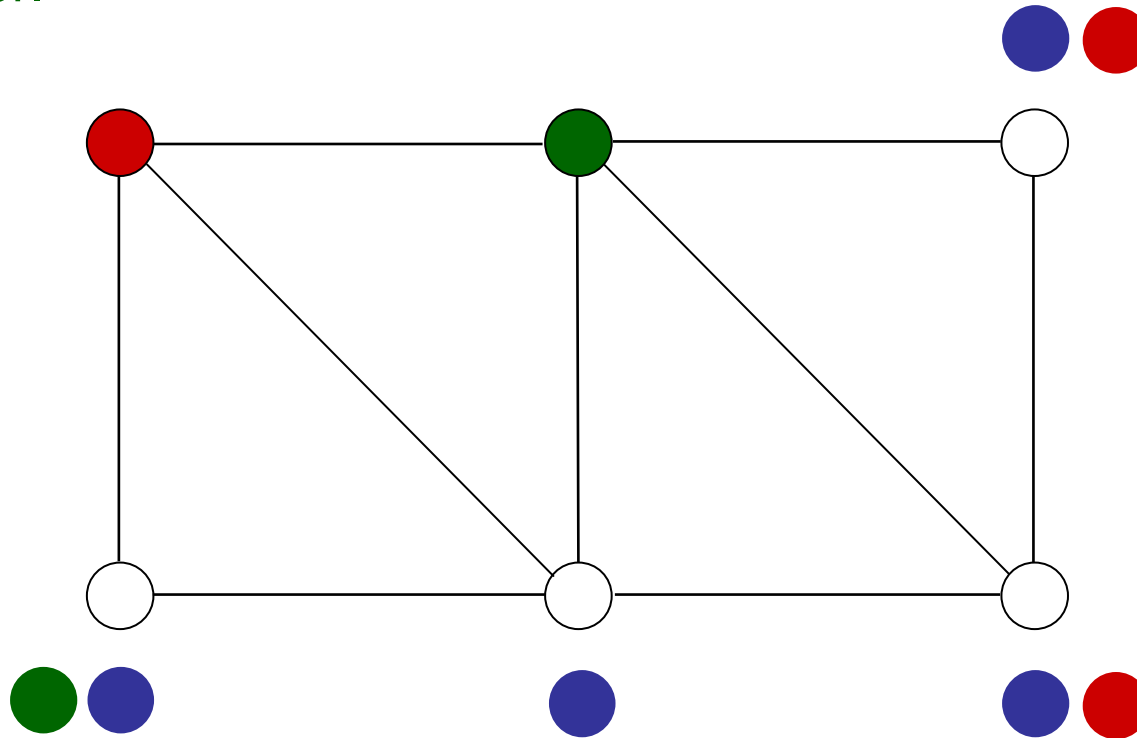
Generalized Arc Consistency (GAC)

- Also known as **hyperarc consistency**.
- A constraint set is **GAC** if every value in every variable domain is part of some feasible solution.
 - That is, the domains are reduced as much as possible.
 - If all constraints are “binary” (contain 2 variables), GAC = arc consistency.
 - Domain reduction is CP’s biggest engine.

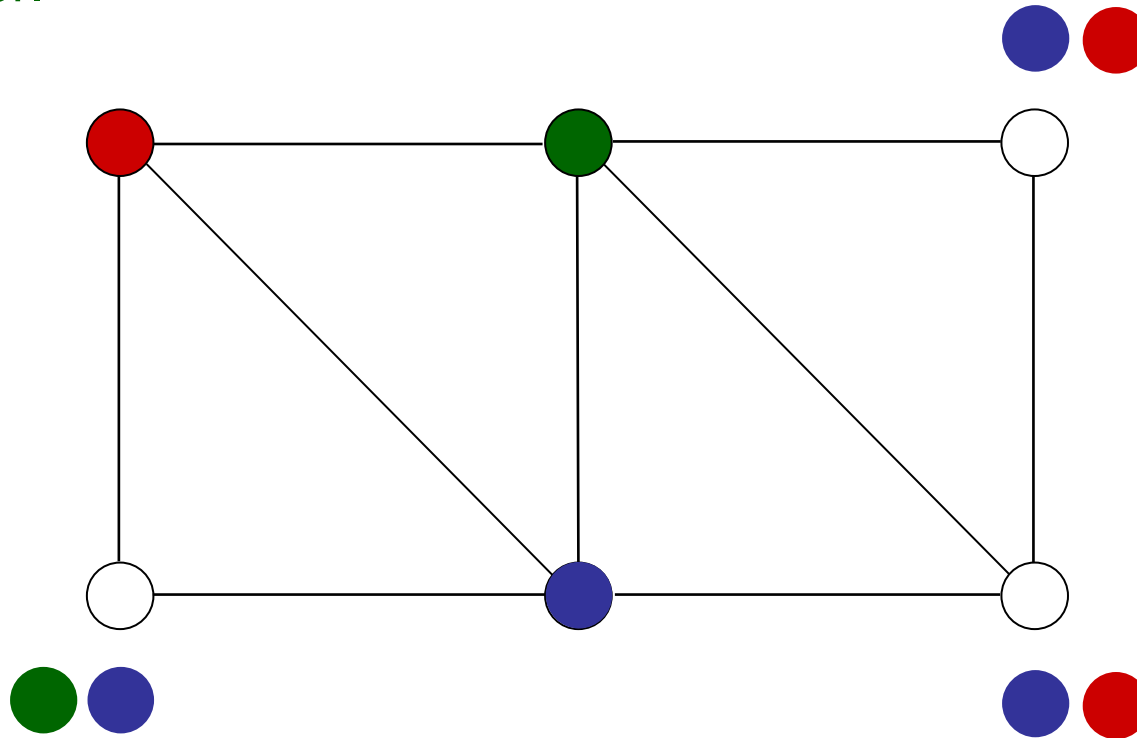
Graph coloring problem that can be solved by arc consistency maintenance alone. Color nodes with red, green, blue with no two adjacent nodes having the same color.



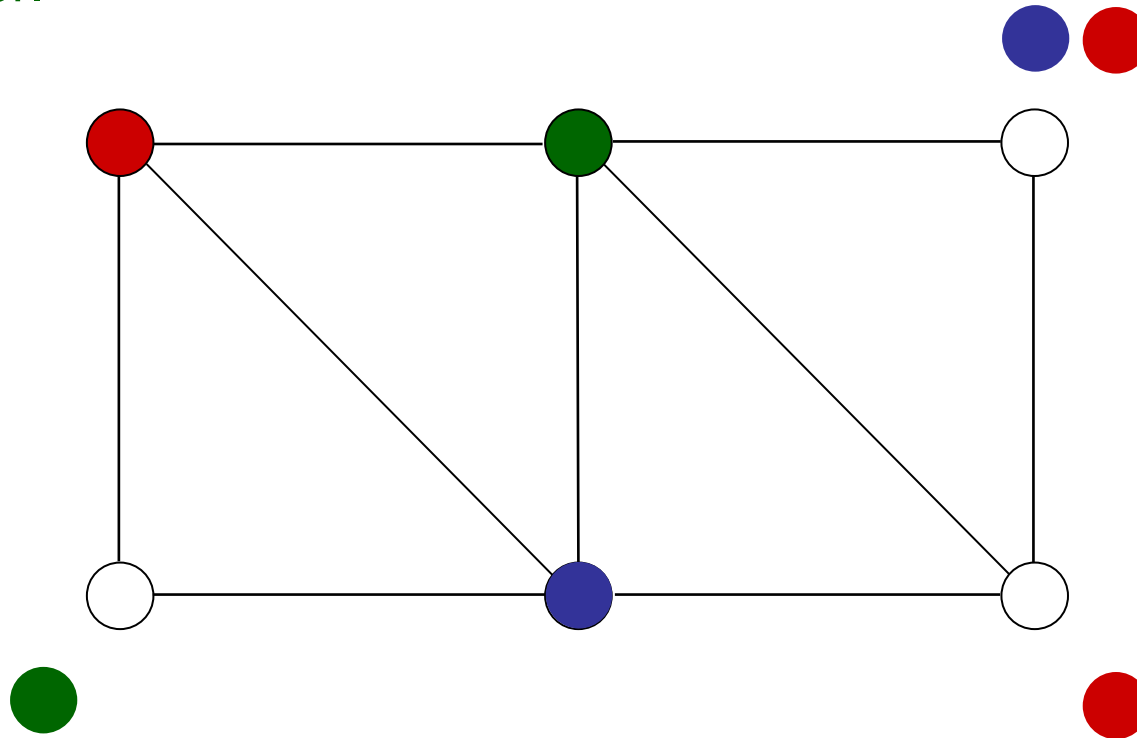
Graph coloring problem that can be solved by arc consistency maintenance alone. Color nodes with red, green, blue with no two adjacent nodes having the same color.



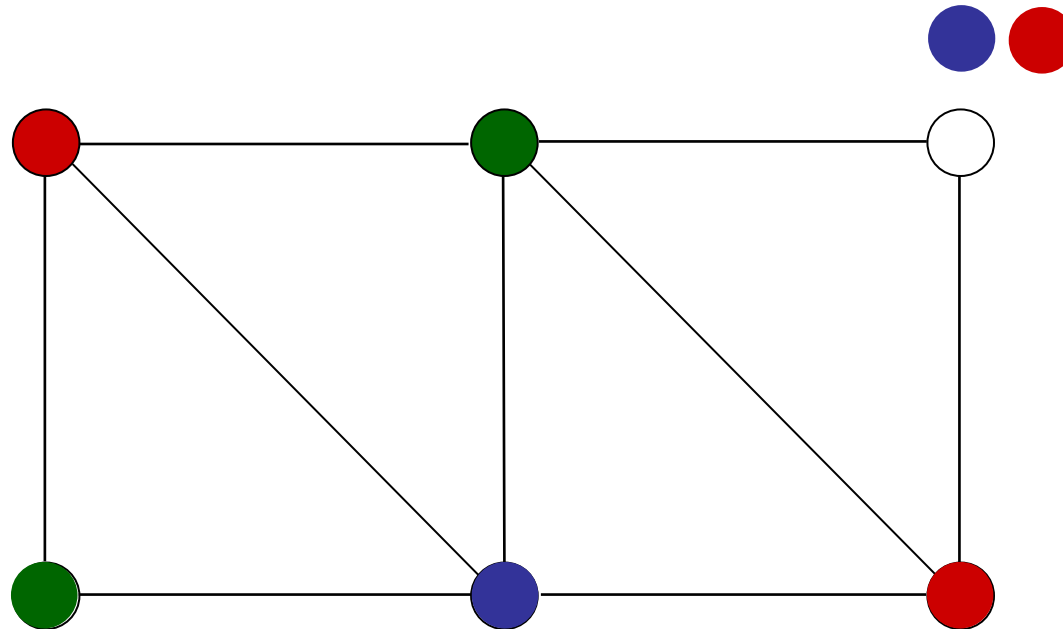
Graph coloring problem that can be solved by arc consistency maintenance alone. Color nodes with red, green, blue with no two adjacent nodes having the same color.



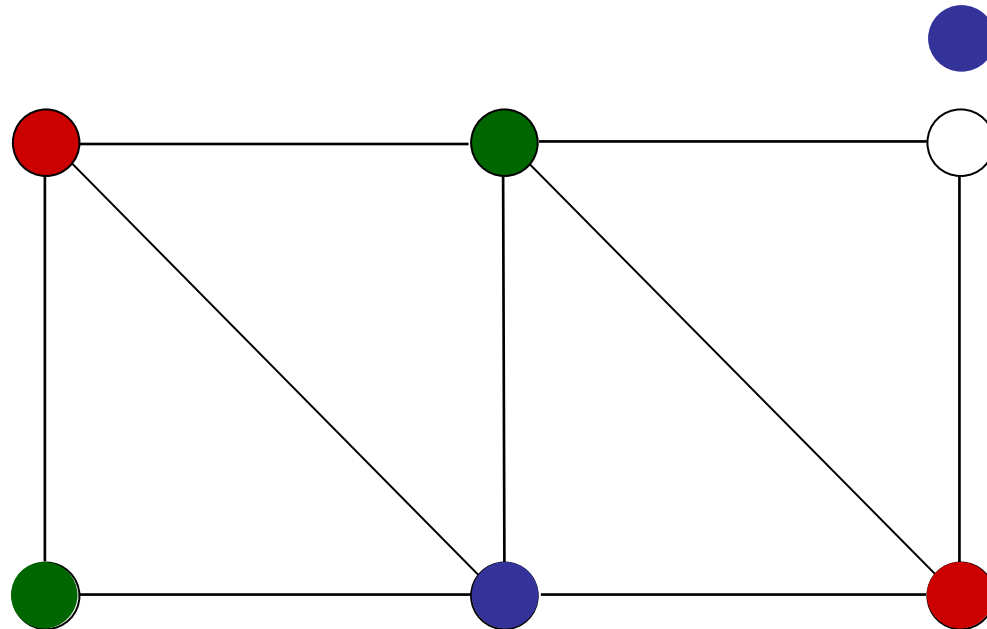
Graph coloring problem that can be solved by arc consistency maintenance alone. Color nodes with red, green, blue with no two adjacent nodes having the same color.



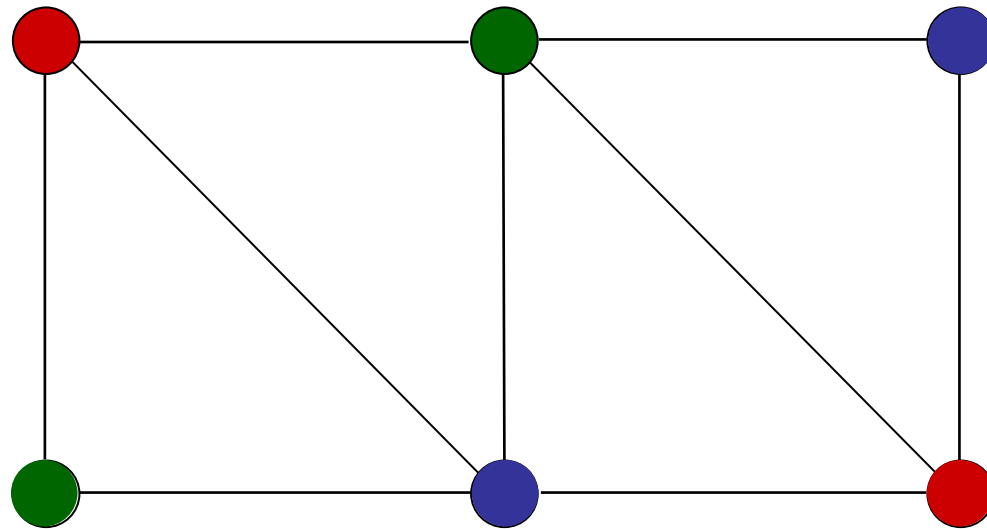
Graph coloring problem that can be solved by arc consistency maintenance alone. Color nodes with red, green, blue with no two adjacent nodes having the same color.



Graph coloring problem that can be solved by arc consistency maintenance alone. Color nodes with red, green, blue with no two adjacent nodes having the same color.



Graph coloring problem that can be solved by arc consistency maintenance alone. Color nodes with red, green, blue with no two adjacent nodes having the same color.



Modeling Examples with Global Constraints

Traveling Salesman

Traveling salesman problem:

Let c_{ij} = distance from city i to city j .


Find the shortest route that visits each of n cities exactly once.

Popular 0-1 model

Let $x_{ij} = 1$ if city i immediately precedes city j , 0 otherwise

$$\begin{array}{ll}\min & \sum_{ij} c_{ij} x_{ij} \\ \text{s.t.} & \sum_i x_{ij} = 1, \text{ all } j \\ & \sum_j x_{ij} = 1, \text{ all } i \\ & \sum_{i \in V} \sum_{j \in W} x_{ij} \geq 1, \text{ all disjoint } V, W \subset \{1, \dots, n\} \\ & x_{ij} \in \{0, 1\}\end{array}$$

Subtour elimination constraints



A CP model

Let y_k = the k th city visited.

The model would be written in a specific constraint programming language but would essentially say:

$$\begin{array}{ll}\min & \sum_k c_{y_k y_{k+1}} \\ \text{s.t.} & \text{alldiff}(y_1, \dots, y_n) \\ & y_k \in \{1, \dots, n\}\end{array}$$

Variable indices

“Global” constraint

An alternate CP model

Let y_k = the city visited after city k .

$$\min \sum_k c_{ky_k}$$

$$\text{s.t. } \text{circuit}(y_1, \dots, y_n)$$

$$y_k \in \{1, \dots, n\}$$




Hamiltonian circuit
constraint

Element constraint


The constraint $c_y \leq 5$ can be implemented:

$$z \leq 5$$

$\text{element}(y, (c_1, \dots, c_n), z)$  Assign z the y th value in the list

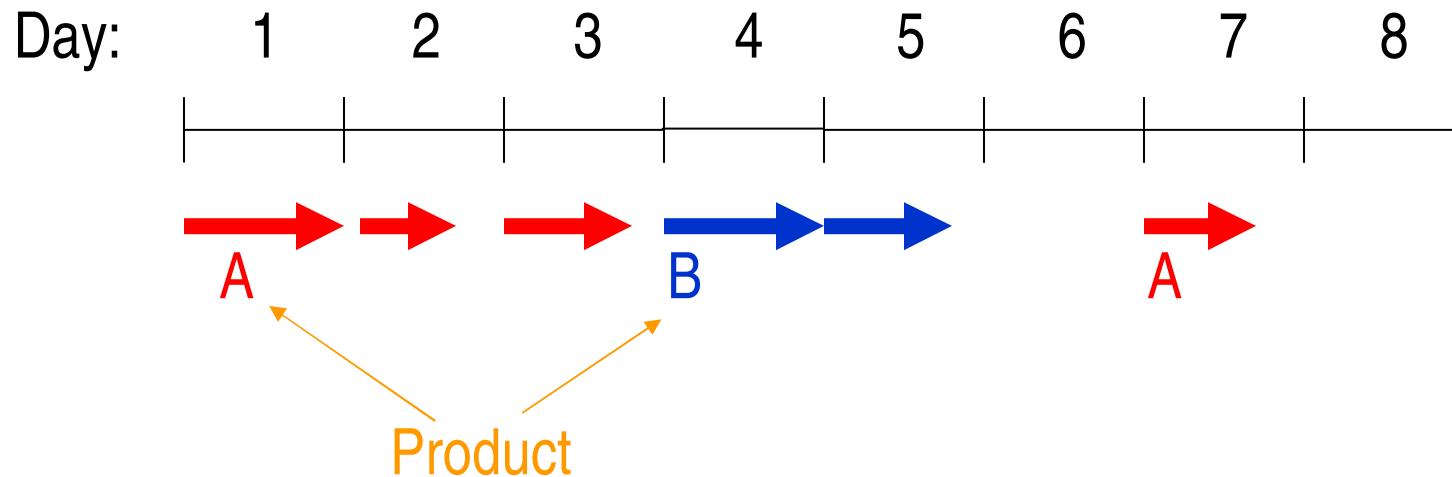
The constraint $x_y \leq 5$ can be implemented

$$z \leq 5$$

$\text{element}(y, (x_1, \dots, x_n), z)$  Add the constraint $z = x_y$

(this is a slightly different constraint)

Modeling example: Lot sizing and scheduling



- At most one product manufactured on each day.
- Demands for each product on each day.
- Minimize setup + holding cost.

Integer
programming
model
(Wolsey)

$$\min \sum_{t,i} \left(h_{it} s_{it} + \sum_{j \neq t} q_{ij} \delta_{ijt} \right) \leftarrow \text{Many variables}$$

$$\begin{aligned} \text{s.t.} \quad & s_{i,t-1} + x_{it} = d_{it} + s_{it}, \quad \text{all } i, t \\ & z_{it} \geq y_{it} - y_{i,t-1}, \quad \text{all } i, t \\ & z_{it} \leq y_{it}, \quad \text{all } i, t \\ & z_{it} \leq 1 - y_{i,t-1}, \quad \text{all } i, t \\ & \delta_{ijt} \geq y_{i,t-1} + y_{jt} - 1, \quad \text{all } i, j, t \\ & \delta_{ijt} \geq y_{i,t-1}, \quad \text{all } i, j, t \\ & \delta_{ijt} \geq y_{jt}, \quad \text{all } i, j, t \\ & x_{it} \leq C y_{it}, \quad \text{all } i, t \\ & \sum_i y_{it} = 1, \quad \text{all } t \\ & y_{it}, z_{it}, \delta_{ijt} \in \{0, 1\} \\ & x_{it}, s_{it} \geq 0 \end{aligned}$$

CP model

Minimize holding and setup costs

$$\min \sum_t \left(q_{y_{t-1}y_t} + \sum_i h_i s_{it} \right)$$

Inventory balance

$$\text{s.t. } s_{i,t-1} + x_{it} = d_{it} + s_{it}, \quad \text{all } i, t$$

Production capacity

$$0 \leq x_{it} \leq C, \quad s_{it} \geq 0, \quad \text{all } i, t$$

$$(y_t \neq i) \rightarrow (x_{it} = 0), \quad \text{all } i, t$$

CP model

Minimize holding and setup costs

Variable indices

$$\min \sum_t \left(q_{y_{t-1}y_t} + \sum_i h_i s_{it} \right)$$

Inventory balance

Production capacity

Product manufactured in period t

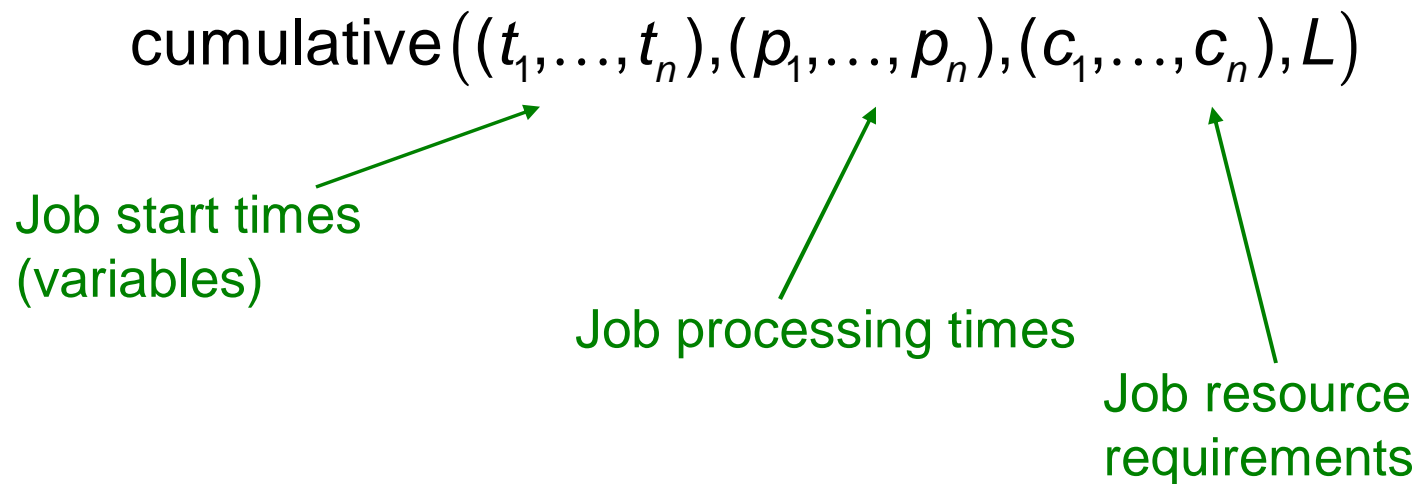
Production level of product i in period t

s.t. $s_{i,t-1} + x_{it} = d_{it} + s_{it}, \text{ all } i, t$

$$0 \leq x_{it} \leq C, \quad s_{it} \geq 0, \quad \text{all } i, t$$
$$(y_t \neq i) \rightarrow (x_{it} = 0), \quad \text{all } i, t$$

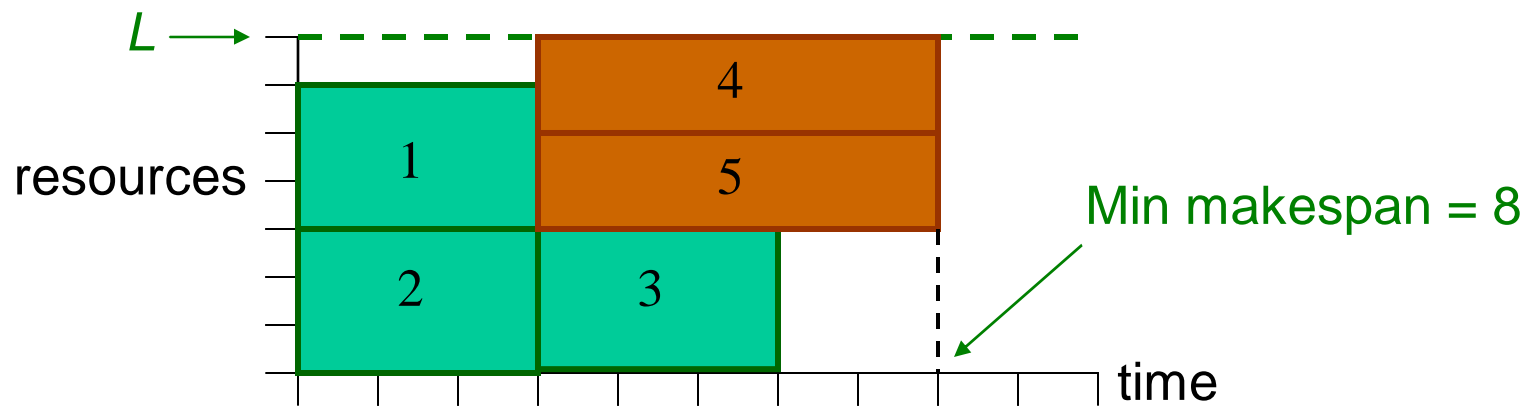
Cumulative scheduling constraint

- Used for resource-constrained scheduling.
- Total resources consumed by jobs at any one time must not exceed L .



Cumulative scheduling constraint

Minimize makespan (no deadlines, all release times = 0):



$$\begin{aligned}
 \min \quad & z \\
 \text{s.t.} \quad & \text{cumulative}((t_1, \dots, t_5), (3, 3, 3, 5, 5), (3, 3, 3, 2, 2), 7) \\
 & z \geq t_1 + 3 \\
 & \vdots \\
 & z \geq t_5 + 2
 \end{aligned}$$

L →
 Resources used
 Processing times
 Job start times

Modeling example: Ship loading

- Will use ILOG's OPL Studio modeling language.
 - Example is from OPL manual.
- The problem
 - Load 34 items on the ship in minimum time (min makespan)
 - Each item requires a certain time and certain number of workers.
 - Total of 8 workers available.

Item	Dura- tion	Labor
1	3	4
2	4	4
3	4	3
4	6	4
5	5	5
6	2	5
7	3	4
8	4	3
9	3	4
10	2	8
11	3	4
12	2	5
13	1	4
14	5	3
15	2	3
16	3	3
17	2	6

Item	Dura- tion	Labor
18	2	7
19	1	4
20	1	4
21	1	4
22	2	4
23	4	7
24	5	8
25	2	8
26	1	3
27	1	3
28	2	6
29	1	8
30	3	3
31	2	3
32	1	3
33	2	3
34	2	3

Problem data

Precedence constraints

1 → 2,4

2 → 3

3 → 5,7

4 → 5

5 → 6

6 → 8

7 → 8

8 → 9

9 → 10

9 → 14

10 → 11

10 → 12

11 → 13

12 → 13

13 → 15,16

14 → 15

15 → 18

16 → 17

17 → 18

18 → 19

18 → 20,21

19 → 23

20 → 23

21 → 22

22 → 23

23 → 24

24 → 25

25 → 26,30,31,32

26 → 27

27 → 28

28 → 29

30 → 28

31 → 28

32 → 33

33 → 34

Use the cumulative scheduling constraint.

min z

s.t. $z \geq t_1 + 3, \quad z \geq t_2 + 4, \quad \text{etc.}$

cumulative $((t_1, \dots, t_{34}), (3, 4, \dots, 2), (4, 4, \dots, 3), 8)$

$t_2 \geq t_1 + 3, \quad t_4 \geq t_1 + 3, \quad \text{etc.}$

OPL model

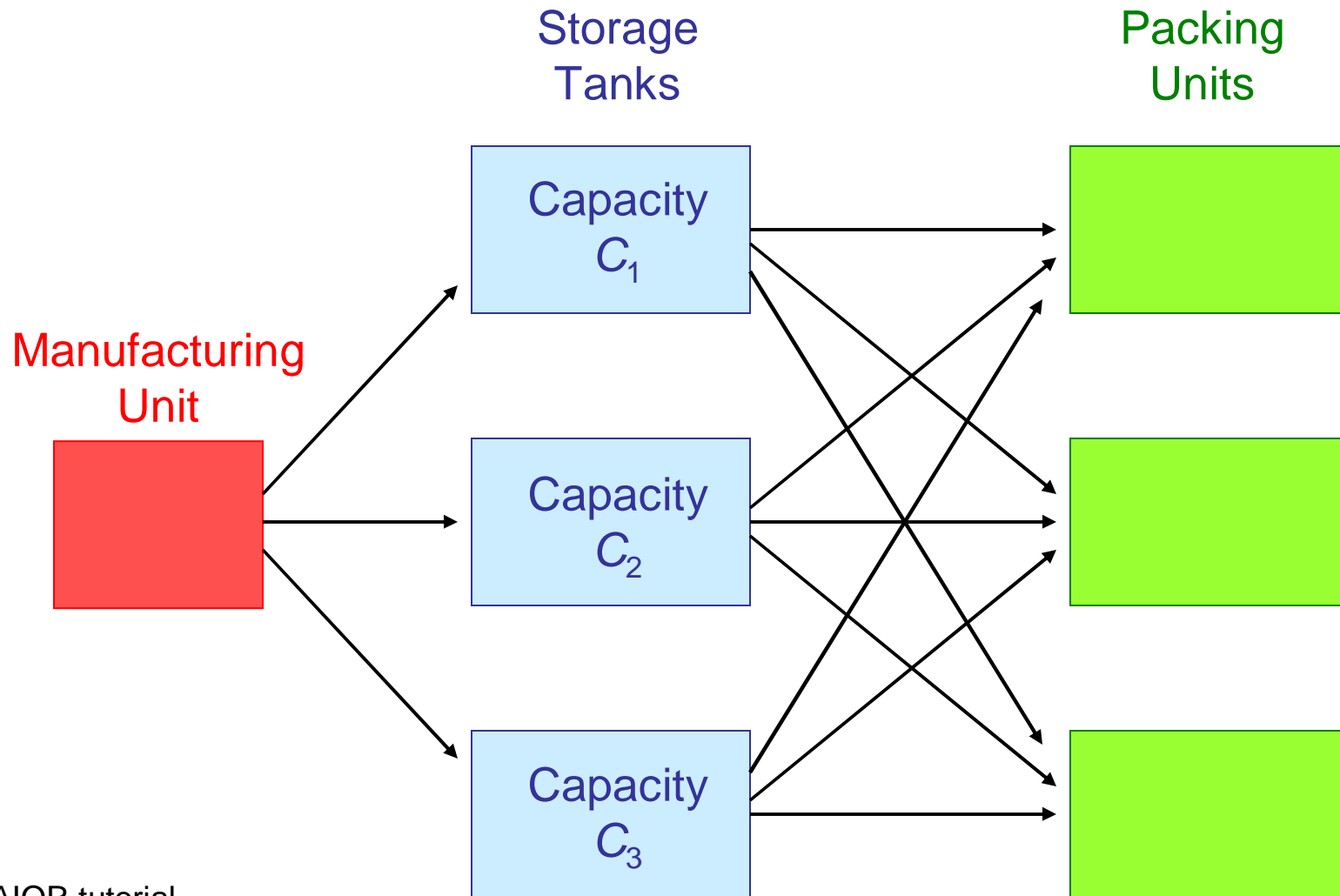
```
int capacity = 8;
int nbTasks = 34;
range Tasks 1..nbTasks;
int duration[Tasks] = [3,4,4,6,...,2];
int totalDuration =
    sum(t in Tasks) duration[t];
int demand[Tasks] = [4,4,3,4,...,3];
struct Precedences {
    int before;
    int after;
}
{Precedences} setOfPrecedences = {
    <1,2>, <1,4>, ..., <33,34> };
```

```

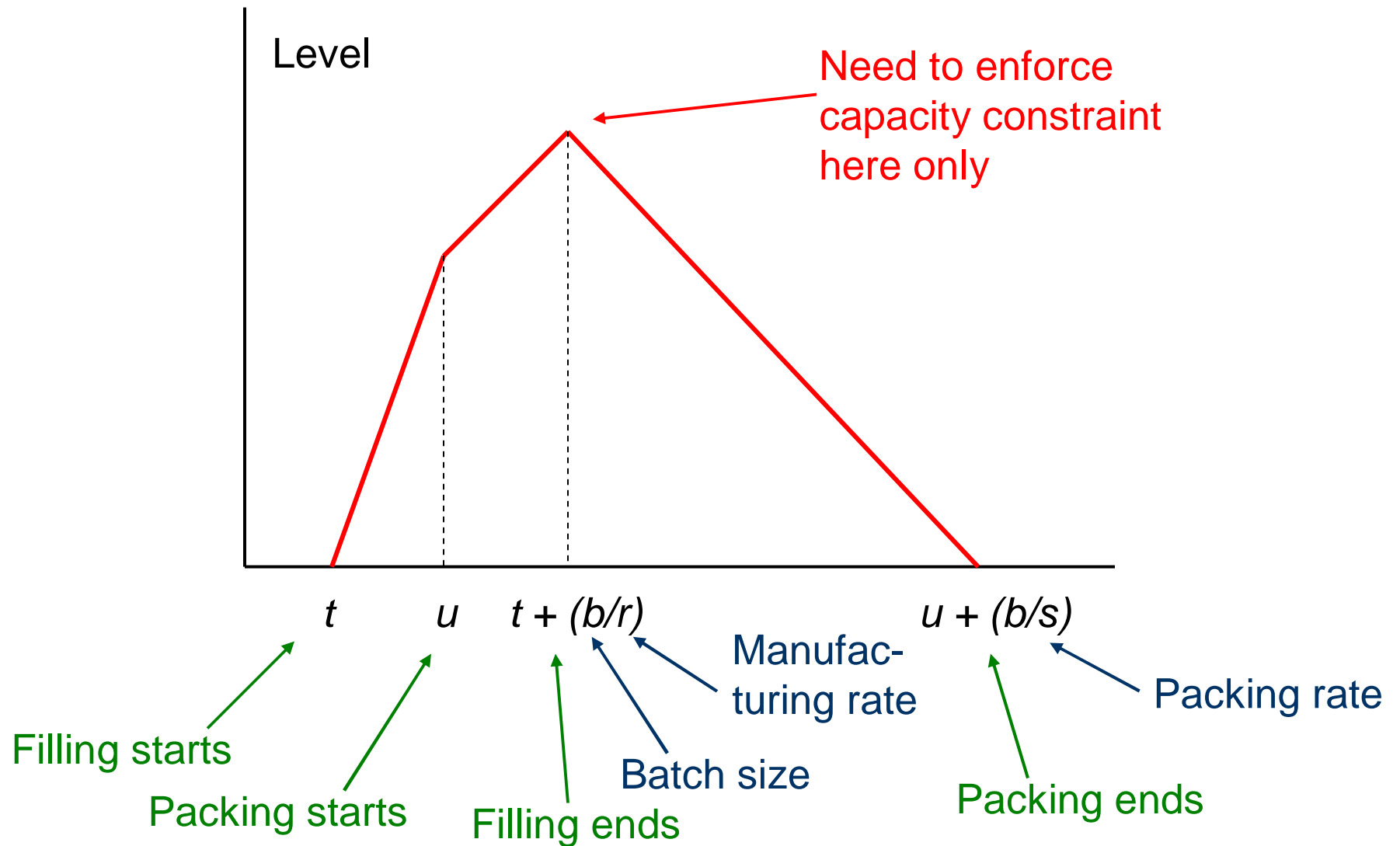
scheduleHorizon = totalDuration;
Activity a[t in Tasks](duration[t]);
DiscreteResource res(8);
Activity makespan(0);
minimize
    makespan.end
subject to
    forall(t in Tasks)
        a[t] precedes makespan;
    forall(p in setOfPrecedences)
        a[p.before] precedes a[p.after];
    forall(t in Tasks)
        a[t] requires(demand[t]) res;
};

```


Modeling example: Production scheduling with intermediate storage




Filling of storage tank





min T  Makespan


s.t. $T \geq u_j + \frac{b_j}{s_j}, \text{ all } j$

$t_j \geq R_j, \text{ all } j$  Job release time

cumulative(t, v, e, m)  m storage tanks

$v_i = u_i + \frac{b_i}{s_i} - t_i, \text{ all } i$  Job duration

$b_i \left(1 - \frac{s_i}{r_i} \right) + s_i u_i \leq C_i, \text{ all } i$  Tank capacity

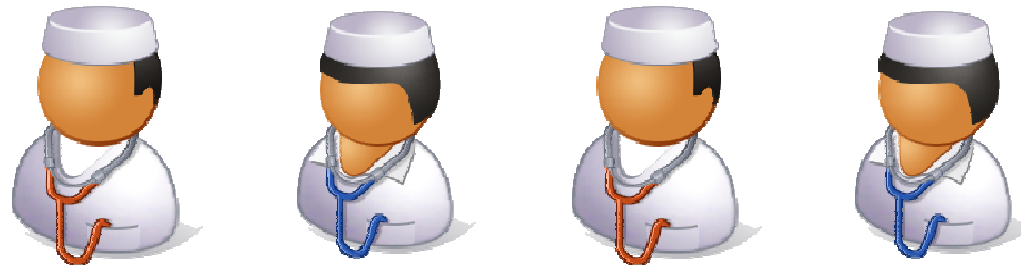
cumulative $\left(u, \left(\frac{b_1}{s_1}, \dots, \frac{b_n}{s_n} \right), e, p \right)$  p packing units

$u_j \geq t_j \geq 0$

$e = (1, \dots, 1)$

Modeling example: Employee scheduling

- Schedule four nurses in 8-hour shifts.
- A nurse works at most one shift a day, at least 5 days a week.
- Same schedule every week.
- No shift staffed by more than two different nurses in a week.
- A nurse cannot work different shifts on two consecutive days.
- A nurse who works shift 2 or 3 must do so at least two days in a row.



Two ways to view the problem

Assign nurses to shifts

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
Shift 1	A	B	A	A	A	A	A
Shift 2	C	C	C	B	B	B	B
Shift 3	D	D	D	D	C	C	D

Assign shifts to nurses

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
Nurse A	1	0	1	1	1	1	1
Nurse B	0	1	0	2	2	2	2
Nurse C	2	2	2	0	3	3	0
Nurse D	3	3	3	3	0	0	3

Use **both** formulations in the same model!

First, assign nurses to shifts.

Let w_{sd} = nurse assigned to shift s on day d

$\text{alldiff}(w_{1d}, w_{2d}, w_{3d}), \text{ all } d$

← The variables w_{1d} , w_{2d} , w_{3d} take different values

That is, schedule 3 different nurses on each day


Use **both** formulations in the same model!

First, assign nurses to shifts.

Let w_{sd} = nurse assigned to shift s on day d

$\text{alldiff}(w_{1d}, w_{2d}, w_{3d}), \text{ all } d$

$\text{cardinality}(w \mid (A, B, C, D), (5, 5, 5, 5), (6, 6, 6, 6))$



A occurs at least 5 and at most 6 times in the array w , and similarly for B, C, D .

That is, each nurse works at least 5 and at most 6 days a week

Use **both** formulations in the same model!

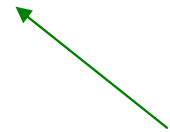
First, assign nurses to shifts.

Let w_{sd} = nurse assigned to shift s on day d

$\text{alldiff}(w_{1d}, w_{2d}, w_{3d}), \text{ all } d$

$\text{cardinality}(w \mid (A, B, C, D), (5, 5, 5, 5), (6, 6, 6, 6))$

$\text{nvalues}(w_{s,\text{Sun}}, \dots, w_{s,\text{Sat}} \mid 1, 2), \text{ all } s$



The variables $w_{s,\text{Sun}}, \dots, w_{s,\text{Sat}}$ take at least 1 and at most 2 different values.

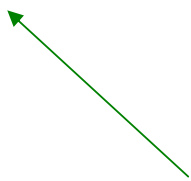
That is, at least 1 and at most 2 nurses work any given shift.

Remaining constraints are not easily expressed in this notation.

So, assign shifts to nurses.

Let y_{id} = shift assigned to nurse i on day d

$\text{alldiff}(y_{1d}, y_{2d}, y_{3d}), \text{ all } d$



Assign a different nurse to each shift on each day.

This constraint is redundant of previous constraints, but redundant constraints speed solution.

Remaining constraints are not easily expressed in this notation.

So, assign shifts to nurses.

Let y_{id} = shift assigned to nurse i on day d

$\text{alldiff}(y_{1d}, y_{2d}, y_{3d}), \text{ all } d$

$\text{stretch}(y_{i,\text{Sun}}, \dots, y_{i,\text{Sat}} \mid (2,3), (2,2), (6,6), P), \text{ all } i$

Every stretch of 2's has length between 2 and 6.

Every stretch of 3's has length between 2 and 6.

So a nurse who works shift 2 or 3 must do so at least two days in a row.

Remaining constraints are not easily expressed in this notation.

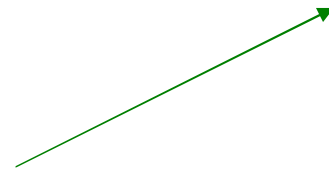
So, assign shifts to nurses.

Let y_{id} = shift assigned to nurse i on day d

$\text{alldiff}(y_{1d}, y_{2d}, y_{3d}), \text{ all } d$

$\text{stretch}(y_{i,\text{Sun}}, \dots, y_{i,\text{Sat}} \mid (2,3), (2,2), (6,6), P), \text{ all } i$

Here $P = \{(s,0), (0,s) \mid s = 1,2,3\}$



Whenever a stretch of a 's immediately precedes a stretch of b 's, (a,b) must be one of the pairs in P .

So a nurse cannot switch shifts without taking at least one day off.

Now we must connect the w_{sd} variables to the y_{id} variables.

Use **channeling constraints**:

$$w_{y_{id}d} = i, \text{ all } i, d$$

$$y_{w_{sd}d} = s, \text{ all } s, d$$

Channeling constraints increase propagation and make the problem easier to solve.

The complete model is:

$\text{alldiff}(w_{1d}, w_{2d}, w_{3d}), \text{ all } d$

$\text{cardinality}(w \mid (A, B, C, D), (5, 5, 5, 5), (6, 6, 6, 6))$

$\text{nvalues}(w_{s,\text{Sun}}, \dots, w_{s,\text{Sat}} \mid 1, 2), \text{ all } s$

$\text{alldiff}(y_{1d}, y_{2d}, y_{3d}), \text{ all } d$

$\text{stretch}(y_{i,\text{Sun}}, \dots, y_{i,\text{Sat}} \mid (2, 3), (2, 2), (6, 6), P), \text{ all } i$

$w_{y_{id}d} = i, \text{ all } i, d$

$y_{w_{sd}d} = s, \text{ all } s, d$



CP Filtering Algorithms

Element

Alldiff

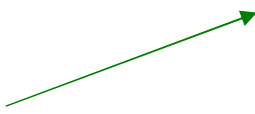
Disjunctive Scheduling

Cumulative Scheduling

Filtering for element

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

Variable domains can be easily filtered to maintain GAC.

Domain of z 

$$D_z \leftarrow D_z \cap \bigcup_{j \in D_y} D_{x_j}$$
$$D_y \leftarrow D_y \cap \{j \mid D_z \cap D_{x_j} \neq \emptyset\}$$
$$D_{x_j} \leftarrow \begin{cases} D_z & \text{if } D_y = \{j\} \\ D_{x_j} & \text{otherwise} \end{cases}$$

Filtering for element

Example... $\text{element}(y, (x_1, x_2, x_3, x_4), z)$

The initial domains are:

$$D_z = \{20, 30, 60, 80, 90\}$$

$$D_y = \{1, 3, 4\}$$

$$D_{x_1} = \{10, 50\}$$

$$D_{x_2} = \{10, 20\}$$

$$D_{x_3} = \{40, 50, 80, 90\}$$

$$D_{x_4} = \{40, 50, 70\}$$

The reduced domains are:

$$D_z = \{80, 90\}$$

$$D_y = \{3\}$$

$$D_{x_1} = \{10, 50\}$$

$$D_{x_2} = \{10, 20\}$$

$$D_{x_3} = \{80, 90\}$$

$$D_{x_4} = \{40, 50, 70\}$$

Filtering for alldiff

$$\text{alldiff}(y_1, \dots, y_n)$$

Domains can be filtered with an algorithm based on maximum cardinality bipartite matching and a theorem of Berge.

It is a special case of optimality conditions for max flow.

Filtering for alldiff

Consider the domains

$$y_1 \in \{1\}$$

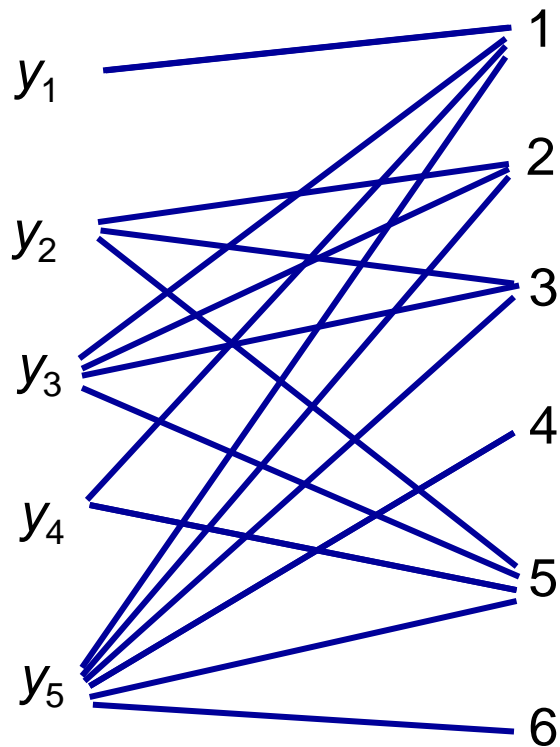
$$y_2 \in \{2, 3, 5\}$$

$$y_3 \in \{1, 2, 3, 5\}$$

$$y_4 \in \{1, 5\}$$

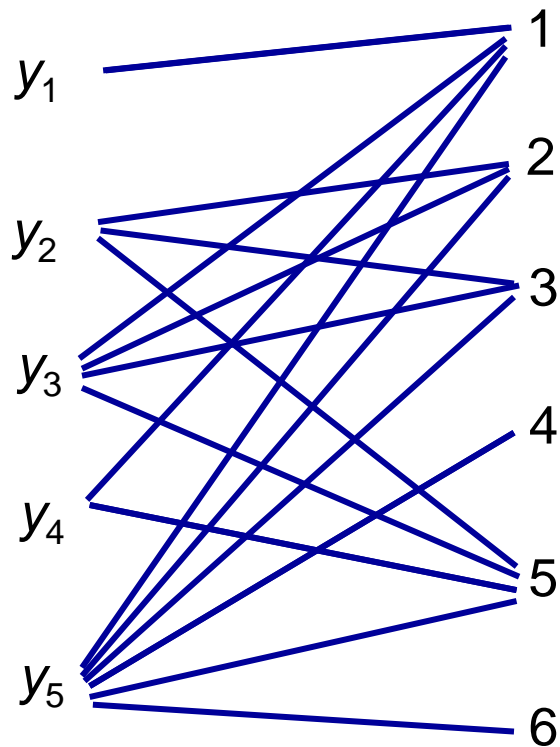
$$y_5 \in \{1, 2, 3, 4, 5, 6\}$$

Indicate domains with edges



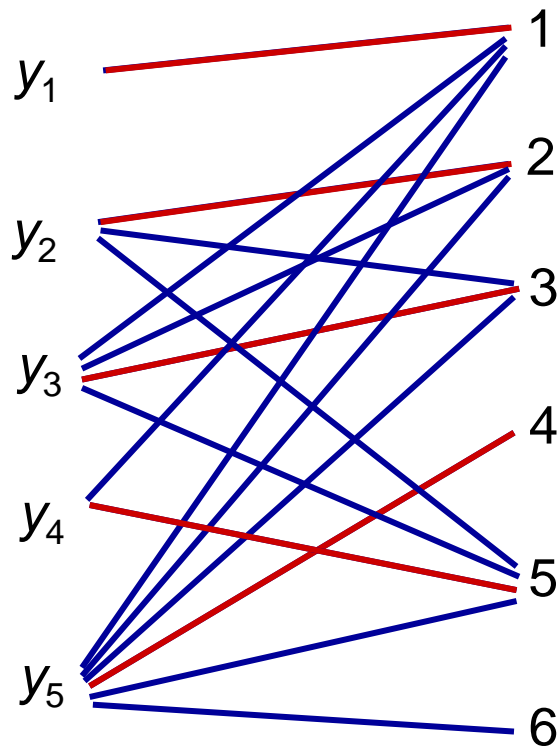
Indicate domains with edges

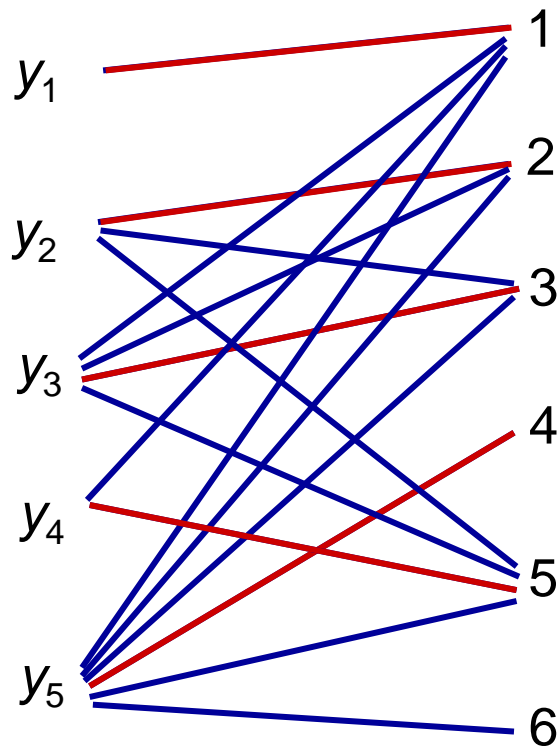
Find maximum cardinality bipartite matching.



Indicate domains with edges

Find maximum cardinality bipartite matching.

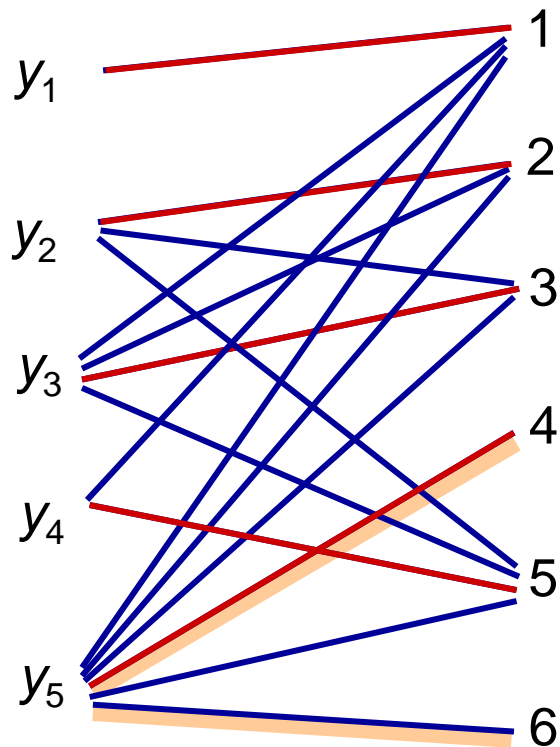




Indicate domains with edges

Find maximum cardinality bipartite matching.

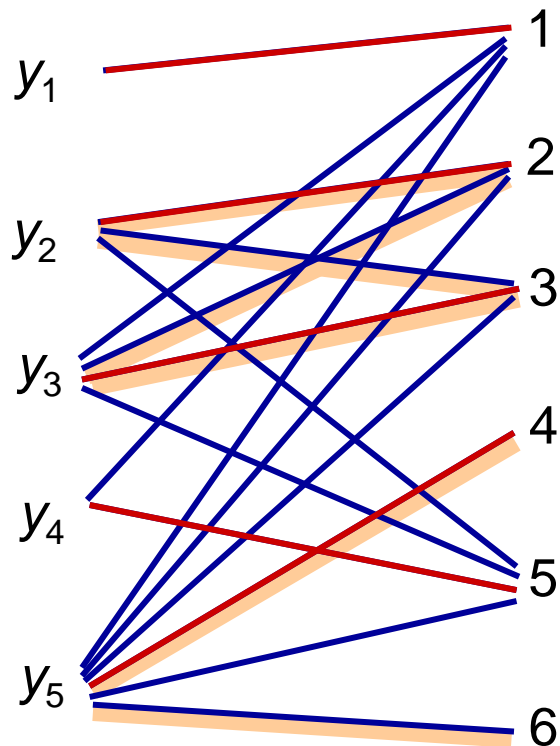
Mark edges in alternating paths that start at an uncovered vertex.



Indicate domains with edges

Find maximum cardinality bipartite matching.

Mark edges in alternating paths that start at an uncovered vertex.

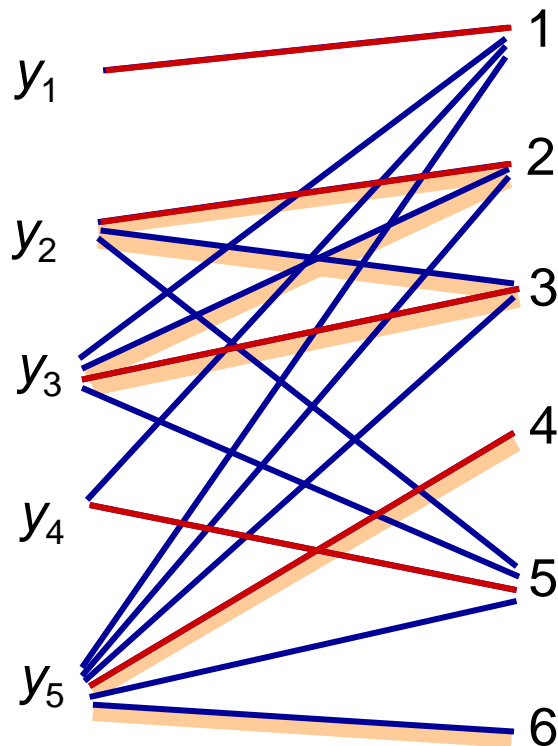


Indicate domains with edges

Find maximum cardinality bipartite matching.

Mark edges in alternating paths that start at an uncovered vertex.

Mark edges in alternating cycles.



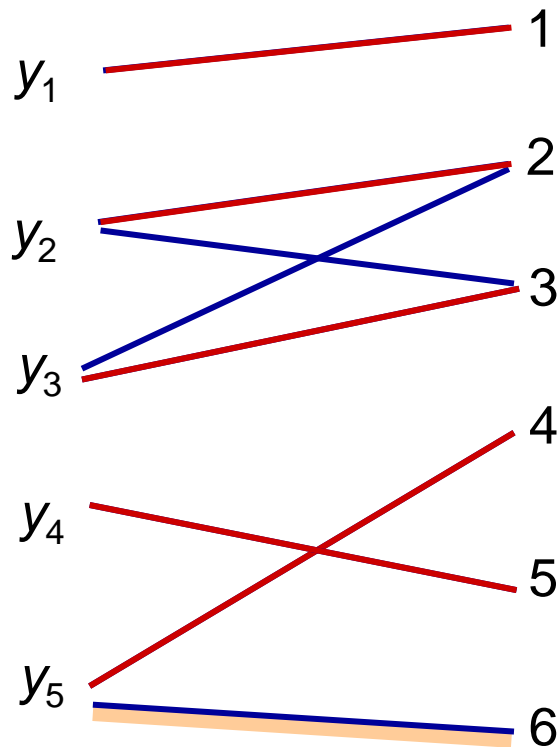
Indicate domains with edges

Find maximum cardinality bipartite matching.

Mark edges in alternating paths that start at an uncovered vertex.

Mark edges in alternating cycles.

Remove unmarked edges not in matching.



Indicate domains with edges

Find maximum cardinality bipartite matching.

Mark edges in alternating paths that start at an uncovered vertex.

Mark edges in alternating cycles.

Remove unmarked edges not in matching.

Filtering for alldiff

Domains have been filtered:

$$\begin{array}{lcl} y_1 \in \{1\} & & y_1 \in \{1\} \\ y_2 \in \{2,3,5\} & & y_2 \in \{2,3\} \\ y_3 \in \{1,2,3,5\} & \longrightarrow & y_3 \in \{2,3\} \\ y_4 \in \{1,5\} & & y_4 \in \{5\} \\ y_5 \in \{1,2,3,4,5,6\} & & y_5 \in \{4,6\} \end{array}$$

GAC achieved.

Disjunctive scheduling

Consider a disjunctive scheduling constraint:

$$\text{disjunctive}((s_1, s_2, s_3, s_5), (p_1, p_2, p_3, p_5))$$

Start time variables

<i>Job</i> <i>j</i>	<i>Release</i> <i>time</i> r_j	<i>Dead-</i> <i>line</i> d_j	<i>Processing</i> <i>time</i> pA_j pB_j	
1	0	10	1	5
2	0	10	3	6
3	2	7	3	7
4	2	10	4	6
5	4	7	2	5

Edge finding for disjunctive scheduling

Consider a disjunctive scheduling constraint:

$$\text{disjunctive}((s_1, s_2, s_3, s_5), (p_1, p_2, p_3, p_5))$$

<i>Job</i> <i>j</i>	<i>Release</i> <i>time</i> r_j	<i>Dead-</i> <i>line</i> d_j	<i>Processing</i> <i>time</i>	
			pA_j	pB_j
1	0	10	1	5
2	0	10	3	6
3	2	7	3	7
4	2	10	4	6
5	4	7	2	5

Processing times

Edge finding for disjunctive scheduling

Consider a disjunctive scheduling constraint:

$$\text{disjunctive}((s_1, s_2, s_3, s_5), (p_1, p_2, p_3, p_5))$$

<i>Job</i> <i>j</i>	<i>Release</i> <i>time</i> r_j	<i>Dead-</i> <i>line</i> d_j	<i>Processing</i> <i>time</i>	
			pA_j	pB_j
1	0	10	1	5
2	0	10	3	6
3	2	7	3	7
4	2	10	4	6
5	4	7	2	5

Variable domains defined by time windows and processing times

$$s_1 \in [0, 10 - 1]$$

$$s_2 \in [0, 10 - 3]$$

$$s_3 \in [2, 7 - 3]$$

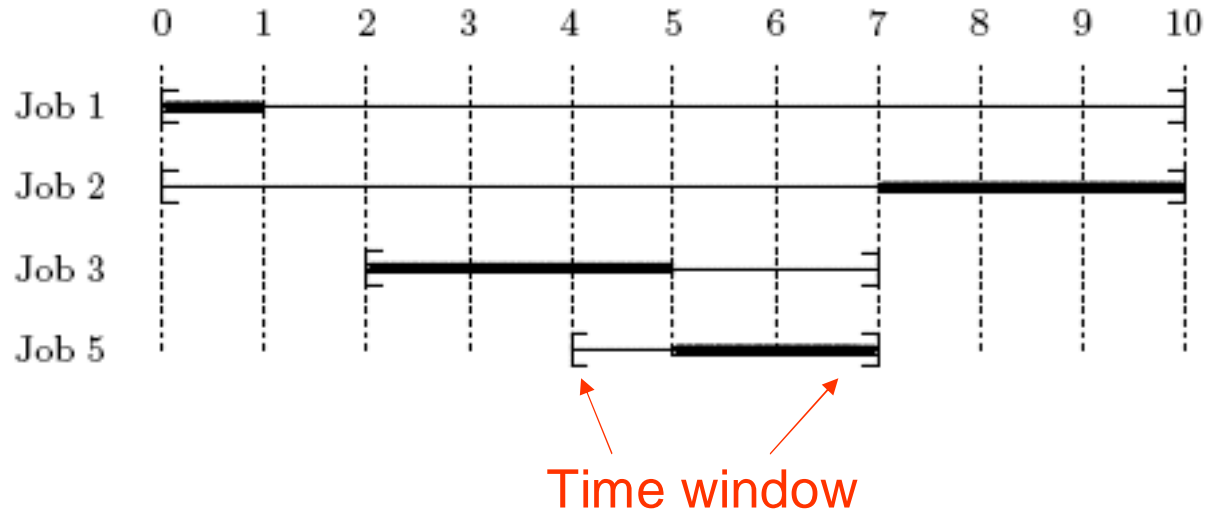
$$s_5 \in [4, 7 - 2]$$

Edge finding for disjunctive scheduling

Consider a disjunctive scheduling constraint:

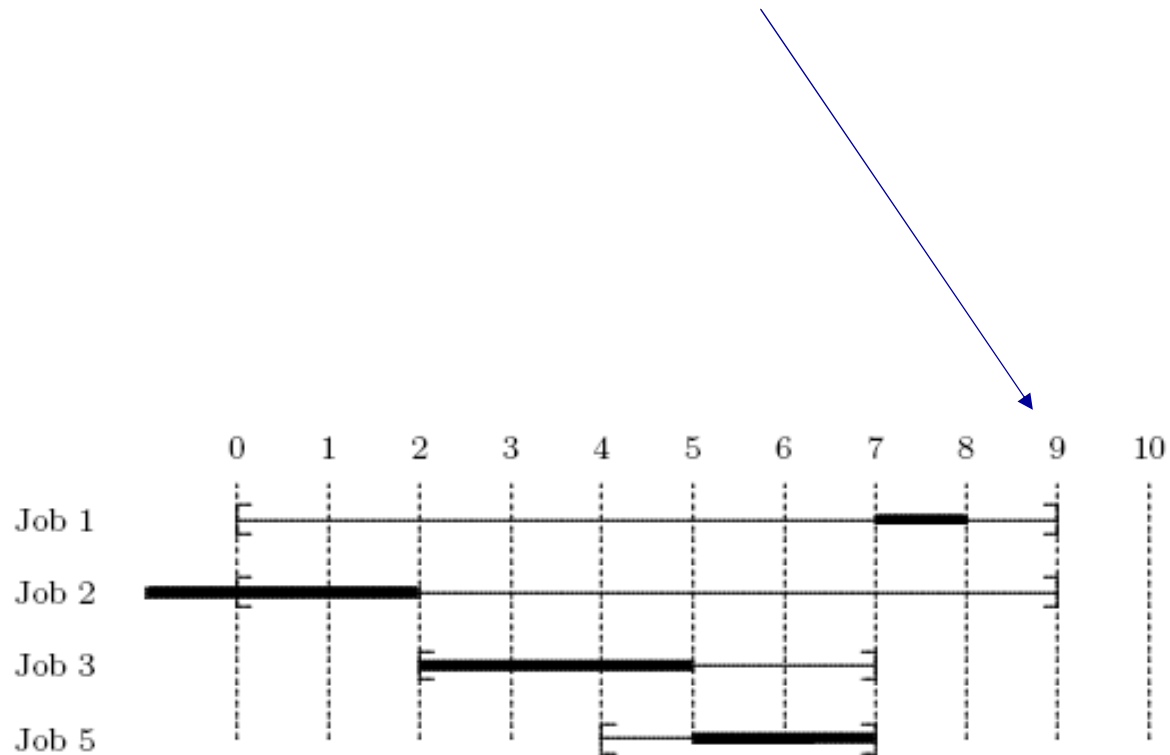
$$\text{disjunctive}((s_1, s_2, s_3, s_5), (p_1, p_2, p_3, p_5))$$

A feasible (min makespan) solution:



Edge finding for disjunctive scheduling

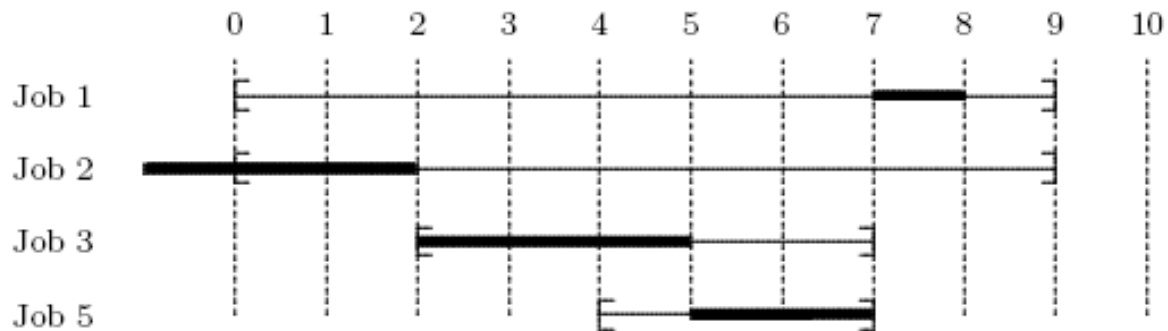
But let's reduce 2 of the deadlines to 9:



Edge finding for disjunctive scheduling

But let's reduce 2 of the deadlines to 9:

We will use edge finding
to prove that there is no
feasible schedule.

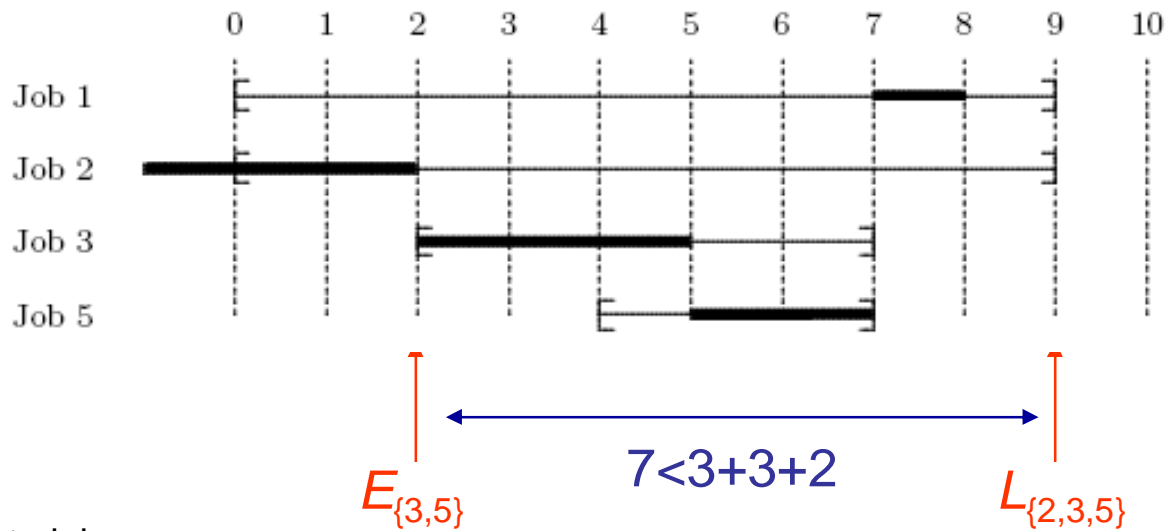


Edge finding for disjunctive scheduling

We can deduce that job 2 must precede jobs 3 and 5: $2 \ll \{3,5\}$

Because if job 2 is not first, there is not enough time for all 3 jobs within the time windows:

$$L_{\{2,3,5\}} - E_{\{3,5\}} < p_{\{2,3,5\}}$$



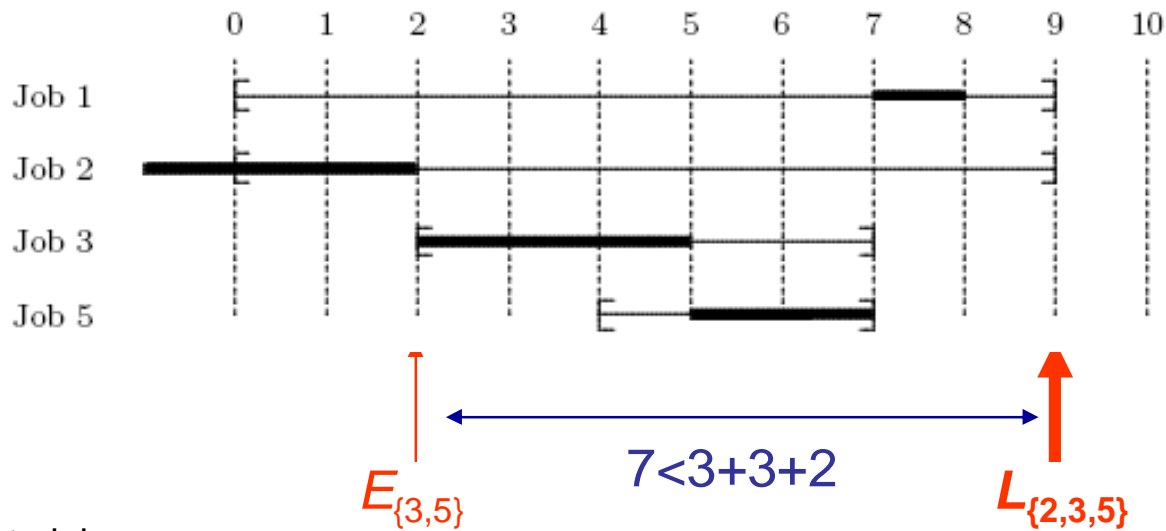
Edge finding for disjunctive scheduling

We can deduce that job 2 must precede jobs 3 and 5: $2 \ll \{3,5\}$

Because if job 2 is not first, there is not enough time for all 3 jobs within the time windows:

$$L_{\{2,3,5\}} - E_{\{3,5\}} < p_{\{2,3,5\}}$$

Latest deadline



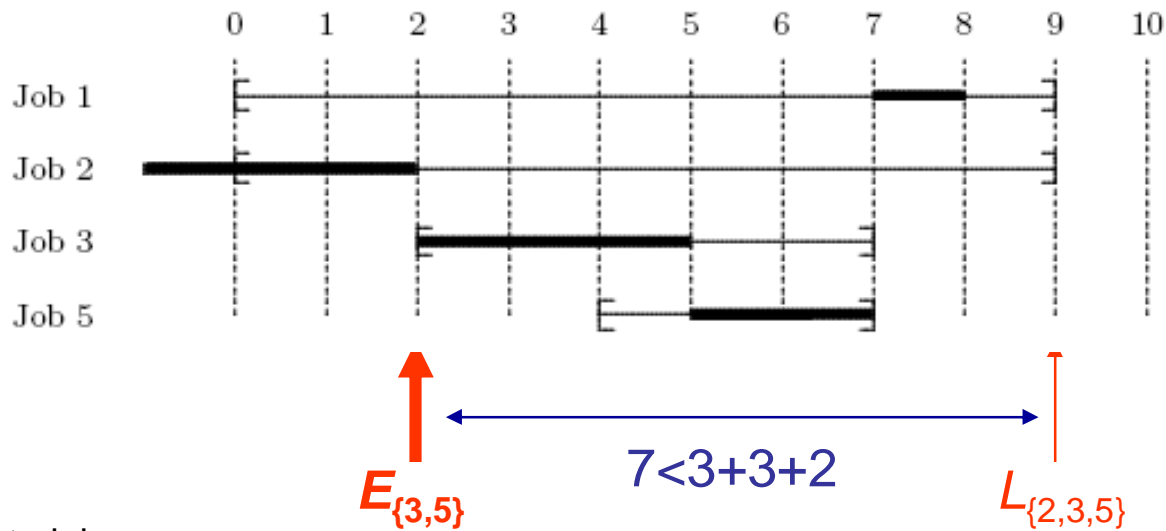
Edge finding for disjunctive scheduling

We can deduce that job 2 must precede jobs 3 and 5: $2 \ll \{3,5\}$

Because if job 2 is not first, there is not enough time for all 3 jobs within the time windows:

$$L_{\{2,3,5\}} - E_{\{3,5\}} < p_{\{2,3,5\}}$$

Earliest release time



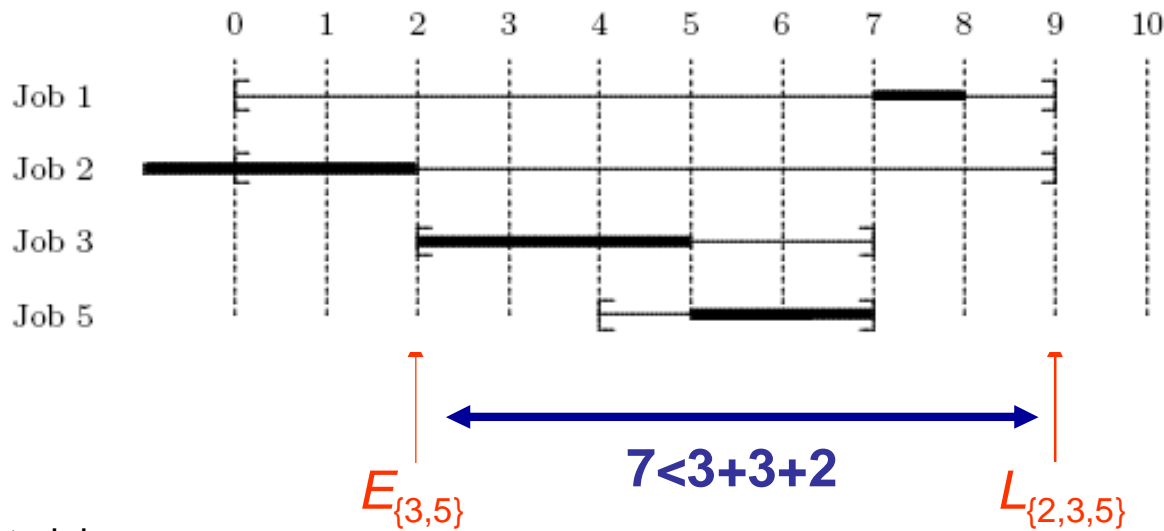
Edge finding for disjunctive scheduling

We can deduce that job 2 must precede jobs 3 and 5: $2 \ll \{3,5\}$

Because if job 2 is not first, there is not enough time for all 3 jobs within the time windows:

$$L_{\{2,3,5\}} - E_{\{3,5\}} < p_{\{2,3,5\}}$$

Total processing time



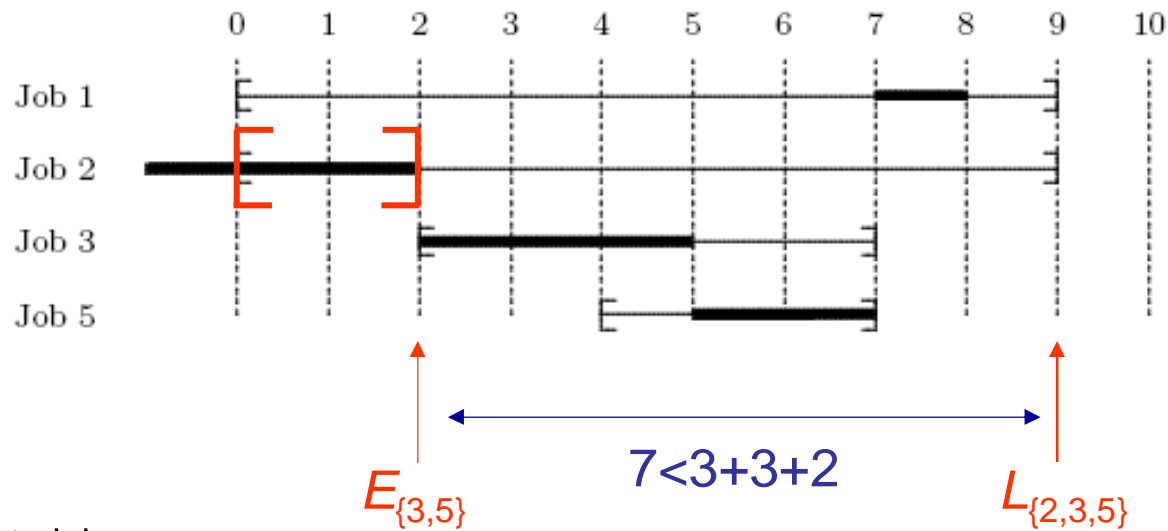
Edge finding for disjunctive scheduling

We can deduce that job 2 must precede jobs 3 and 5: $2 \ll \{3,5\}$

So we can tighten deadline of job 2 to minimum of

$$L_{\{3\}} - p_{\{3\}} = 4 \qquad L_{\{5\}} - p_{\{5\}} = 5 \qquad L_{\{3,5\}} - p_{\{3,5\}} = 2$$

Since time window of job 2 is now too narrow, there is no feasible schedule.



Edge finding for disjunctive scheduling

In general, we can deduce that job k must precede all the jobs in set J : $k \ll J$

If there is not enough time for all the jobs after the earliest release time of the jobs in J

$$L_{J \cup \{k\}} - E_J < p_{J \cup \{k\}} \qquad L_{\{2,3,5\}} - E_{\{3,5\}} < p_{\{2,3,5\}}$$

Edge finding for disjunctive scheduling

In general, we can deduce that job k must precede all the jobs in set J : $k \ll J$

If there is not enough time for all the jobs after the earliest release time of the jobs in J

$$L_{J \cup \{k\}} - E_J < p_{J \cup \{k\}} \qquad L_{\{2,3,5\}} - E_{\{3,5\}} < p_{\{2,3,5\}}$$

Now we can tighten the deadline for job k to:

$$\min_{J' \subset J} \{L_{J'} - p_{J'}\} \qquad L_{\{3,5\}} - p_{\{3,5\}} = 2$$

Edge finding for disjunctive scheduling

There is a symmetric rule: $k \gg J$

If there is not enough time for all the jobs before the latest deadline of the jobs in J :

$$L_J - E_{J \cup \{k\}} < p_{J \cup \{k\}}$$

Now we can tighten the release date for job k to:

$$\max_{J' \subset J} \{E_{J'} + p_{J'}\}$$

Edge finding for disjunctive scheduling

Problem: how can we avoid enumerating all subsets J of jobs to find edges?

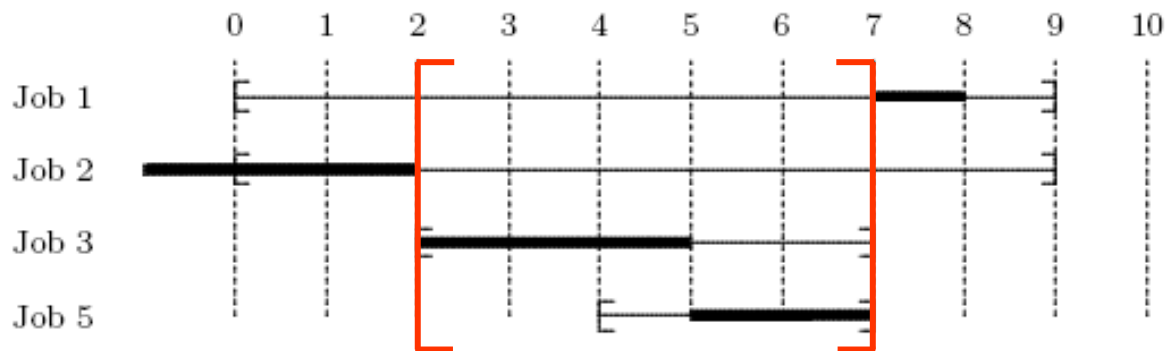
$$L_{J \cup \{k\}} - E_J < p_{J \cup \{k\}}$$

...and all subsets J' of J to tighten the bounds?

$$\min_{J' \subset J} \{L_{J'} - p_{J'}\}$$

Edge finding for disjunctive scheduling

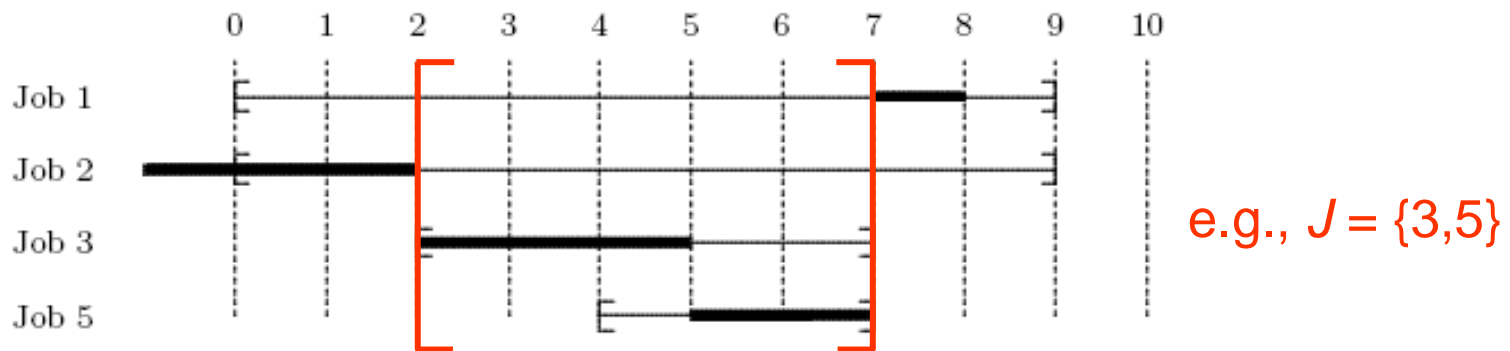
Key result: We only have to consider sets J whose time windows lie within some interval.



e.g., $J = \{3, 5\}$

Edge finding for disjunctive scheduling

Key result: We only have to consider sets J whose time windows lie within some interval.



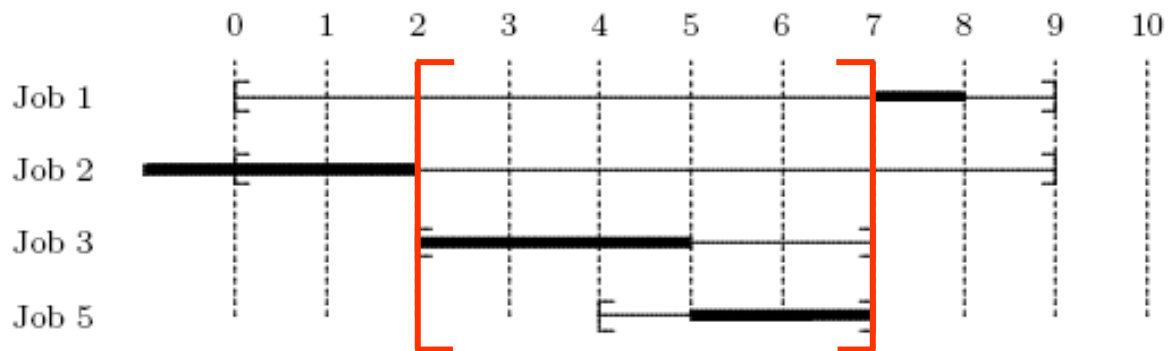
Removing a job from those within an interval only weakens the test

$$L_{J \cup \{k\}} - E_J < p_{J \cup \{k\}}$$

There are a polynomial number of intervals defined by release times and deadlines.

Edge finding for disjunctive scheduling

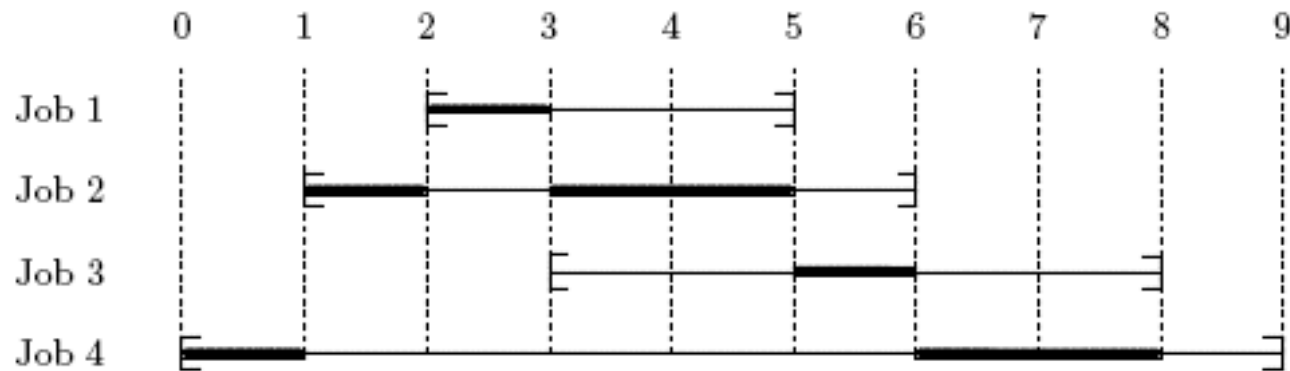
Key result: We only have to consider sets J whose time windows lie within some interval.



Note: Edge finding does not achieve bounds consistency, which is an NP-hard problem.

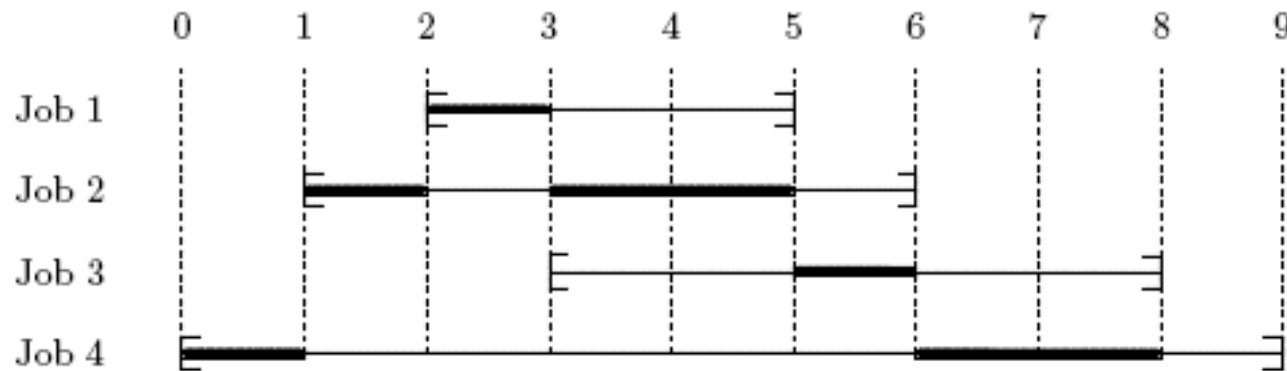
Edge finding for disjunctive scheduling

One $O(n^2)$ algorithm is based on the Jackson pre-emptive schedule (JPS). Using a different example, the JPS is:



Edge finding for disjunctive scheduling

One $O(n^2)$ algorithm is based on the Jackson pre-emptive schedule (JPS). Using a different example, the JPS is:



- For each job i
- Jobs unfinished at time E_i in JPS
 - Scan jobs $k \in J_i$ in decreasing order of L_k
 - Select first k for which $L_k - E_i < p_i + \bar{p}_{J_{ik}}$
 - Total remaining processing time in JPS of jobs in J_{ik}
 - Conclude that $i \gg J_{ik}$
 - Jobs $j \neq i$ in J_i with $L_j \leq L_k$
 - Update E_i to $\text{JPS}(i, k)$
 - Latest completion time in JPS of jobs in J_{ik}

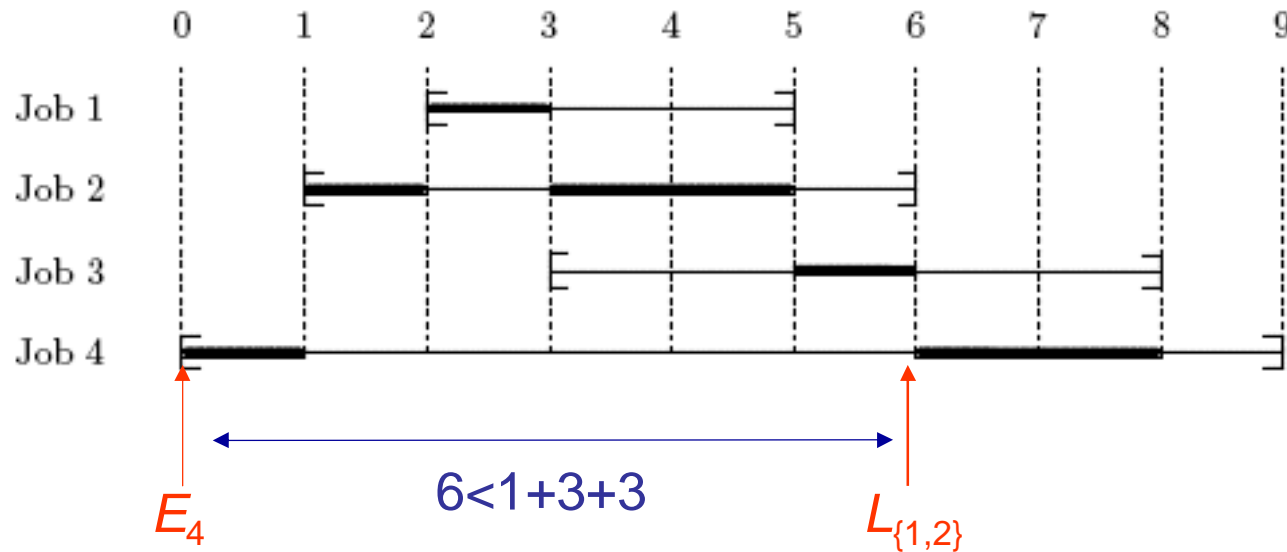
Not-first/not-last rules

We can deduce that job 4 cannot precede jobs 1 and 2:

$$\neg(4 \ll \{1,2\})$$

Because if job 4 is first, there is too little time to complete the jobs before the later deadline of jobs 1 and 2:

$$L_{\{1,2\}} - E_4 < p_1 + p_2 + p_4$$



Not-first/not-last rules

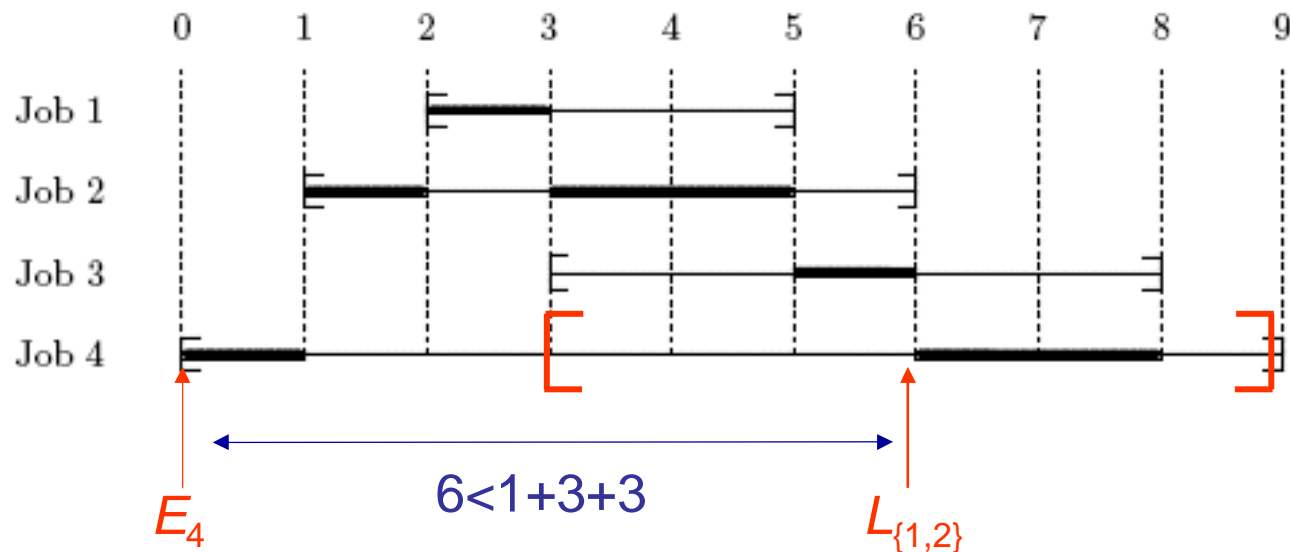
We can deduce that job 4 cannot precede jobs 1 and 2:

$$\neg(4 \ll \{1,2\})$$

Now we can tighten the release time of job 4 to minimum of:

$$E_1 + p_1 = 3$$

$$E_2 + p_2 = 4$$



Not-first/not-last rules

In general, we can deduce that job k cannot precede all the jobs in J :

$$\neg(k \ll J)$$

if there is too little time after release time of job k to complete all jobs before the latest deadline in J :

$$L_J - E_k < p_J$$

Now we can update E_i to

$$\min_{j \in J} \{E_j + p_j\}$$

Not-first/not-last rules

In general, we can deduce that job k cannot precede all the jobs in J :

$$\neg(k \ll J)$$

if there is too little time after release time of job k to complete all jobs before the latest deadline in J :

$$L_J - E_k < p_J$$

Now we can update E_i to

$$\min_{j \in J} \{E_j + p_j\}$$

There is a symmetric not-last rule.

The rules can be applied in polynomial time, although an efficient algorithm is quite complicated.

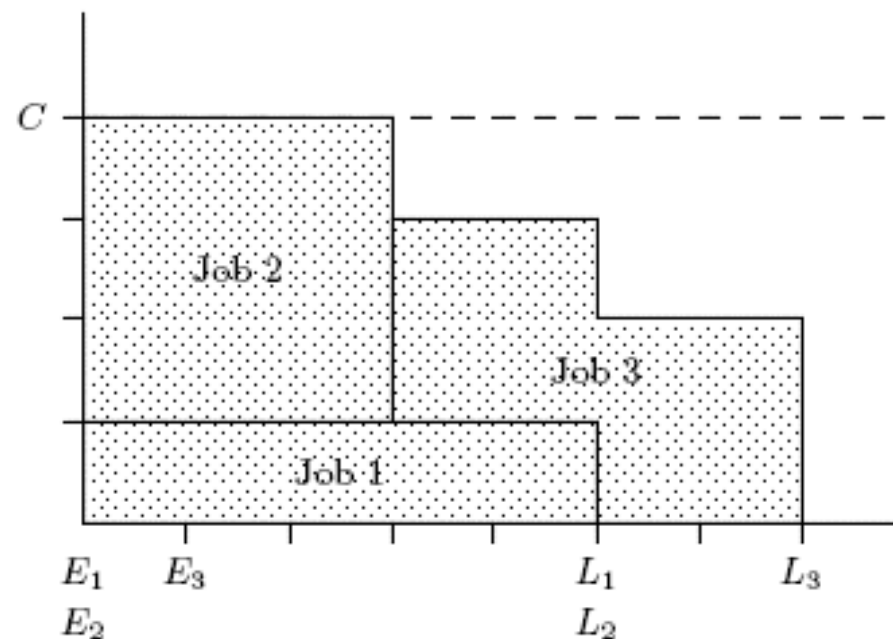
Cumulative scheduling

Consider a cumulative scheduling constraint:

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

j	p_j	c_j	E_j	L_j
1	5	1	0	5
2	3	3	0	5
3	4	2	1	7

A feasible solution:

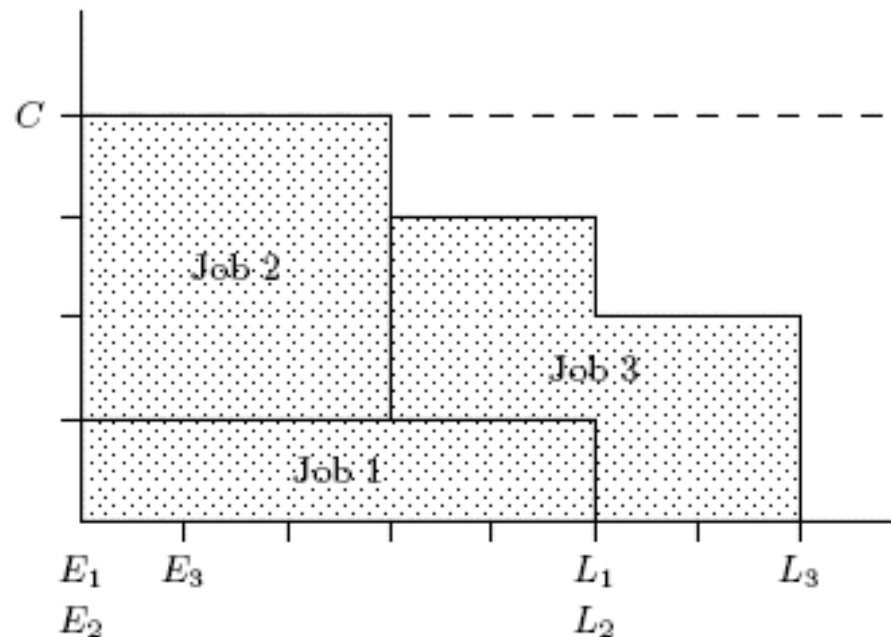


Edge finding for cumulative scheduling

We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

Because the total **energy** required exceeds the area between the earliest release time and the later deadline of jobs 1,2:

$$e_3 + e_{\{1,2\}} > C \cdot (L_{\{1,2\}} - E_{\{1,2,3\}})$$



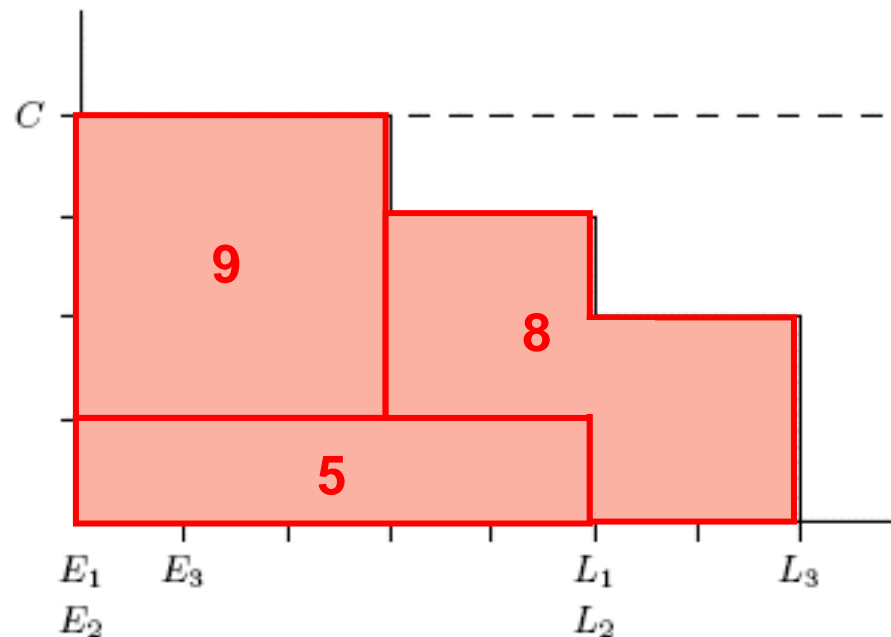
Edge finding for cumulative scheduling

We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

Because the total **energy** required exceeds the area between the earliest release time and the later deadline of jobs 1,2:

$$e_3 + e_{\{1,2\}} > C \cdot (L_{\{1,2\}} - E_{\{1,2,3\}})$$

Total energy
required = 22



Edge finding for cumulative scheduling

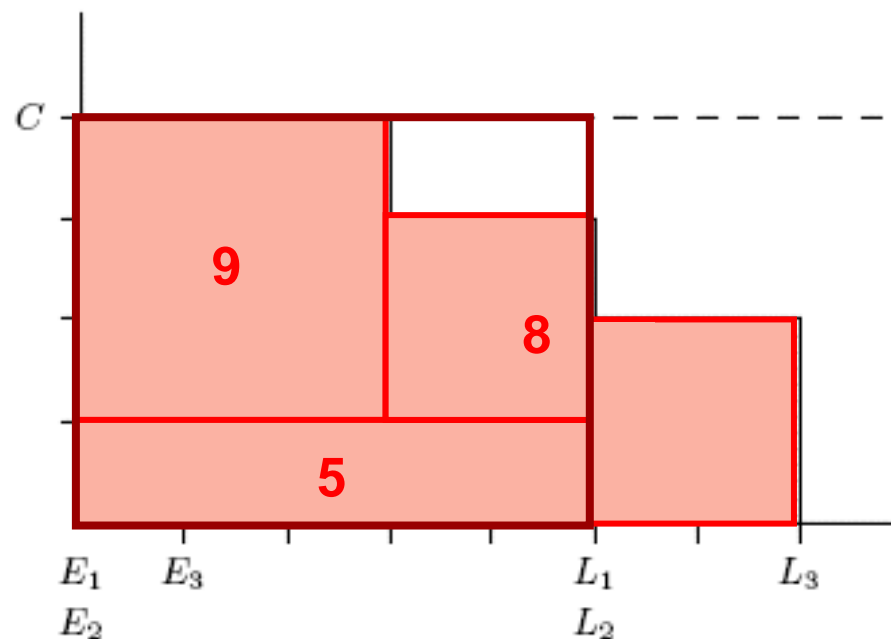
We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

Because the total **energy** required exceeds the area between the earliest release time and the later deadline of jobs 1,2:

$$e_3 + e_{\{1,2\}} > C \cdot (L_{\{1,2\}} - E_{\{1,2,3\}})$$

Total energy
required = 22

Area available
= 20



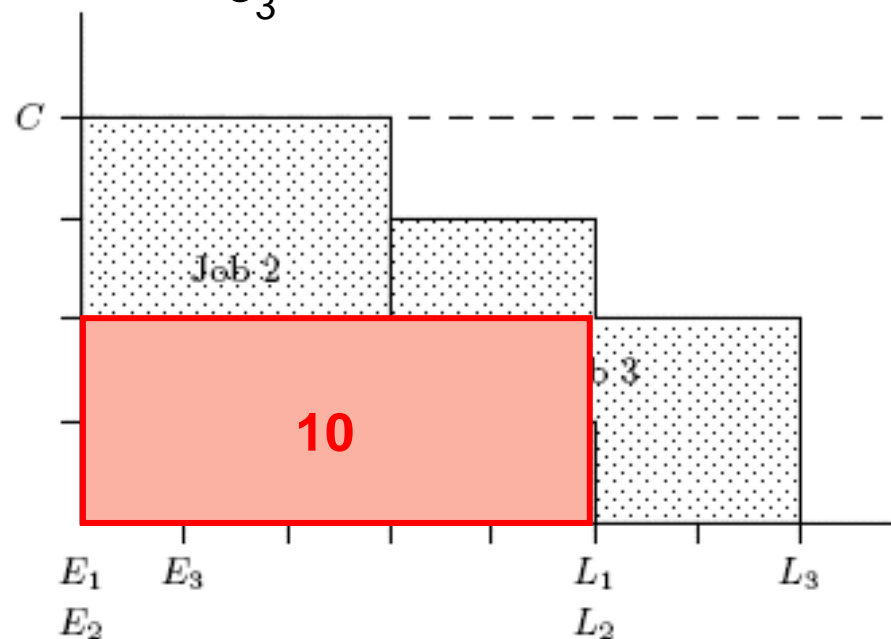
Edge finding for cumulative scheduling

We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

We can update the release time of job 3 to

$$E_{\{1,2\}} + \frac{e_J - (C - c_3)(L_{\{1,2\}} - E_{\{1,2\}})}{c_3}$$

Energy available
for jobs 1,2 if
space is left for job
3 to start anytime
= 10



Edge finding for cumulative scheduling

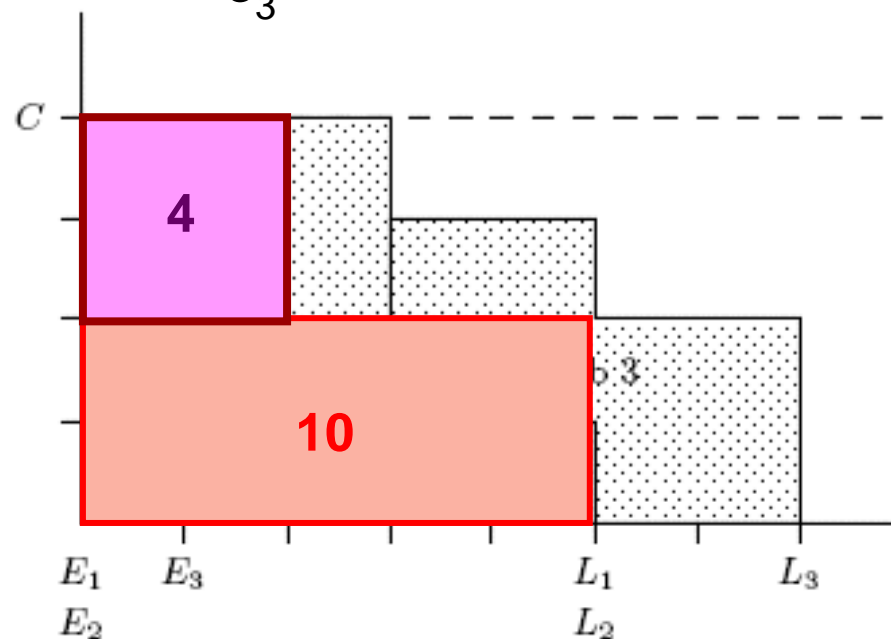
We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

We can update the release time of job 3 to

$$E_{\{1,2\}} + \frac{e_J - (C - c_3)(L_{\{1,2\}} - E_{\{1,2\}})}{c_3}$$

Energy available
for jobs 1,2 if
space is left for job
3 to start anytime
= 10

Excess energy
required by jobs
1,2 = 4



Edge finding for cumulative scheduling

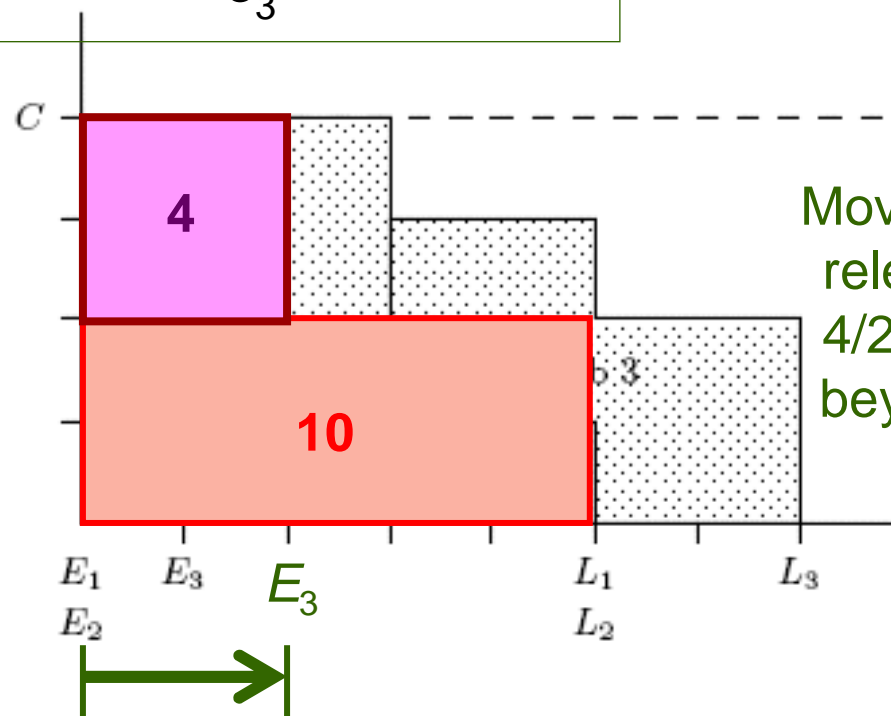
We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

We can update the release time of job 3 to

$$E_{\{1,2\}} + \frac{e_j - (C - c_3)(L_{\{1,2\}} - E_{\{1,2\}})}{c_3}$$

Energy available
for jobs 1,2 if
space is left for job
3 to start anytime
= 10

Excess energy
required by jobs
1,2 = 4



Move up job 3
release time
 $4/2 = 2$ units
beyond $E_{\{1,2\}}$

Edge finding for cumulative scheduling

In general, if $e_{J \cup \{k\}} > C \cdot (L_J - E_{J \cup \{k\}})$

then $k > J$, and update E_k to

$$\max_{\substack{J' \subset J \\ e_{J'} - (C - c_k)(L_{J'} - E_{J'}) > 0}} \left\{ E_{J'} + \frac{e_{J'} - (C - c_k)(L_{J'} - E_{J'})}{c_k} \right\}$$

In general, if $e_{J \cup \{k\}} > C \cdot (L_{J \cup \{k\}} - E_J)$

then $k < J$, and update L_k to

$$\min_{\substack{J' \subset J \\ e_{J'} - (C - c_k)(L_{J'} - E_{J'}) > 0}} \left\{ L_{J'} - \frac{e_{J'} - (C - c_k)(L_{J'} - E_{J'})}{c_k} \right\}$$

Edge finding for cumulative scheduling

There is an $O(n^2)$ algorithm that finds all applications of the edge finding rules.

Other propagation rules for cumulative scheduling

- Extended edge finding.
- Timetabling.
- Not-first/not-last rules.
- Energetic reasoning.



Linear Relaxation

Why Relax?

Algebraic Analysis of LP

Linear Programming Duality

LP-Based Domain Filtering

Example: Single-Vehicle Routing

Disjunctions of Linear Systems

Why Relax?

Solving a relaxation of a problem can:

- Tighten variable bounds.
- Possibly solve original problem.
- Guide the search in a promising direction.
- Filter domains using reduced costs or Lagrange multipliers.
- Prune the search tree using a bound on the optimal value.
- Provide a more global view, because a single OR relaxation can pool relaxations of several constraints.

Some OR models that can provide relaxations:

- Linear programming (LP).
- Mixed integer linear programming (MILP)
 - Can itself be relaxed as an LP.
 - LP relaxation can be strengthened with cutting planes.
- Lagrangean relaxation.
- Specialized relaxations.
 - For particular problem classes.
 - For global constraints.

Motivation

- **Linear programming** is remarkably versatile for representing real-world problems.
- LP is by far the most widely used tool for **relaxation**.
- LP relaxations can be strengthened by **cutting planes**.
 - Based on polyhedral analysis.
- LP has an elegant and powerful **duality theory**.
 - Useful for domain filtering, and much else.
- The LP problem is **extremely well solved**.

Algebraic Analysis of LP

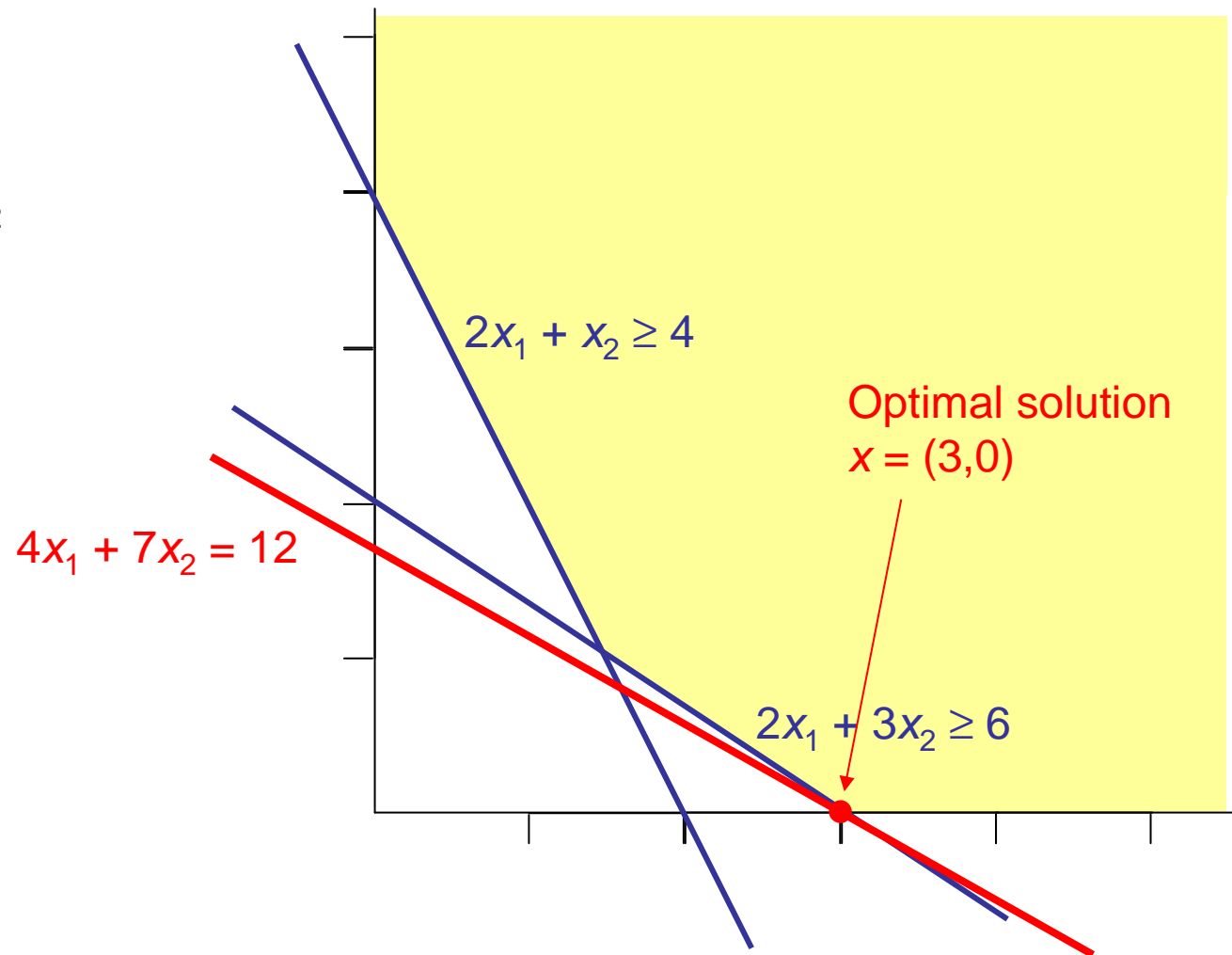
An example...

$$\min 4x_1 + 7x_2$$

$$2x_1 + 3x_2 \geq 6$$

$$2x_1 + x_2 \geq 4$$

$$x_1, x_2 \geq 0$$



Algebraic Analysis of LP

Rewrite

as

$$\min 4x_1 + 7x_2$$

$$2x_1 + 3x_2 \geq 6$$

$$2x_1 + x_2 \geq 4$$

$$x_1, x_2 \geq 0$$

$$\min 4x_1 + 7x_2$$

$$2x_1 + 3x_2 - x_3 = 6$$

$$2x_1 + x_2 - x_4 = 4$$

$$x_1, x_2, x_3, x_4 \geq 0$$

In general an LP has the form $\min cx$

$$Ax = b$$

$$x \geq 0$$

Algebraic analysis of LP

Write $\min cx$
 $\boxed{A}x = b$
 $x \geq 0$

$m \times n$ matrix

as $\min c_B x_B + c_N x_N$
 $Bx_B + Nx_N = b$
 $\boxed{x_B}, \boxed{x_N} \geq 0$

Basic
variables

Nonbasic
variables

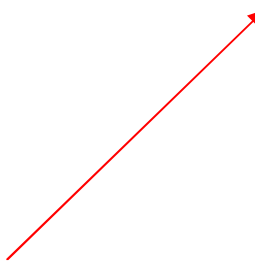
where
 $A = [\boxed{B} \ N]$

Any set of
 m linearly
independent
columns of A .

These form a
basis for the
space spanned
by the columns.

Algebraic analysis of LP

Write $\min cx$ as $\min c_B x_B + c_N x_N$ where
 $Ax = b$ $Bx_B + Nx_N = b$ $A = [B \ N]$
 $x \geq 0$ $x_B, x_N \geq 0$



Solve constraint equation for x_B : $x_B = B^{-1}b - B^{-1}Nx_N$



All solutions can be obtained by setting x_N to some value.

The solution is **basic** if $x_N = 0$.

It is a **basic feasible solution** if $x_N = 0$ and $x_B \geq 0$.

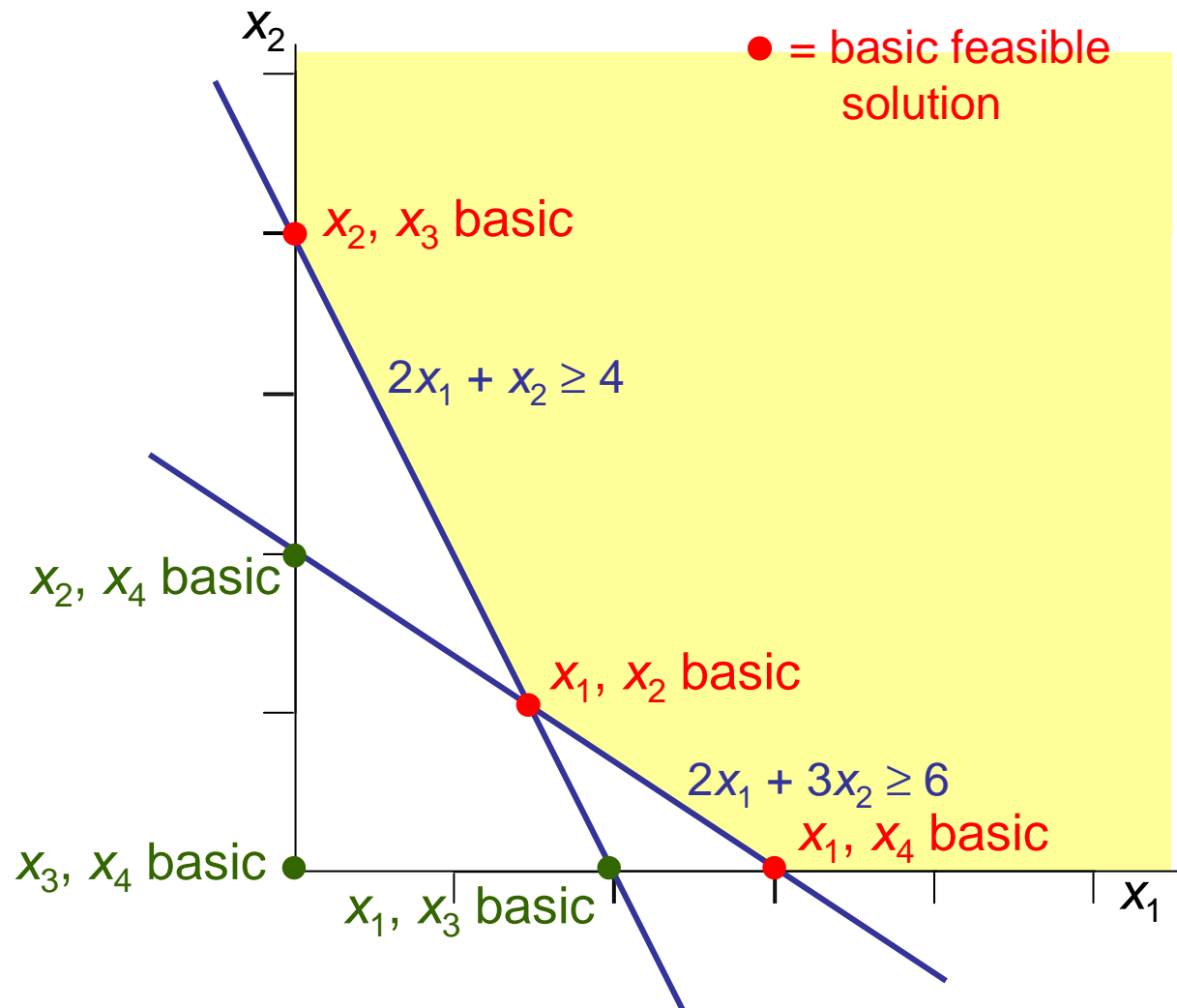
Example...

$$\min 4x_1 + 7x_2$$

$$2x_1 + 3x_2 - x_3 = 6$$

$$2x_1 + x_2 - x_4 = 4$$

$$x_1, x_2, x_3, x_4 \geq 0$$



Algebraic analysis of LP

Write $\min cx$ as $\min \boxed{c_B x_B + c_N x_N}$ where

$$Ax = b \quad Bx_B + Nx_N = b \quad A = [B \ N]$$
$$x \geq 0 \quad x_B, x_N \geq 0$$

Solve constraint equation for x_B : $x_B = B^{-1}b - B^{-1}Nx_N$

Express cost in terms of nonbasic variables:

$$c_B B^{-1}b + \boxed{(c_N - c_B B^{-1}N)} x_N$$

Vector of reduced costs

Since $x_N \geq 0$,
basic solution $(x_B, 0)$
is optimal if
reduced costs are
nonnegative.

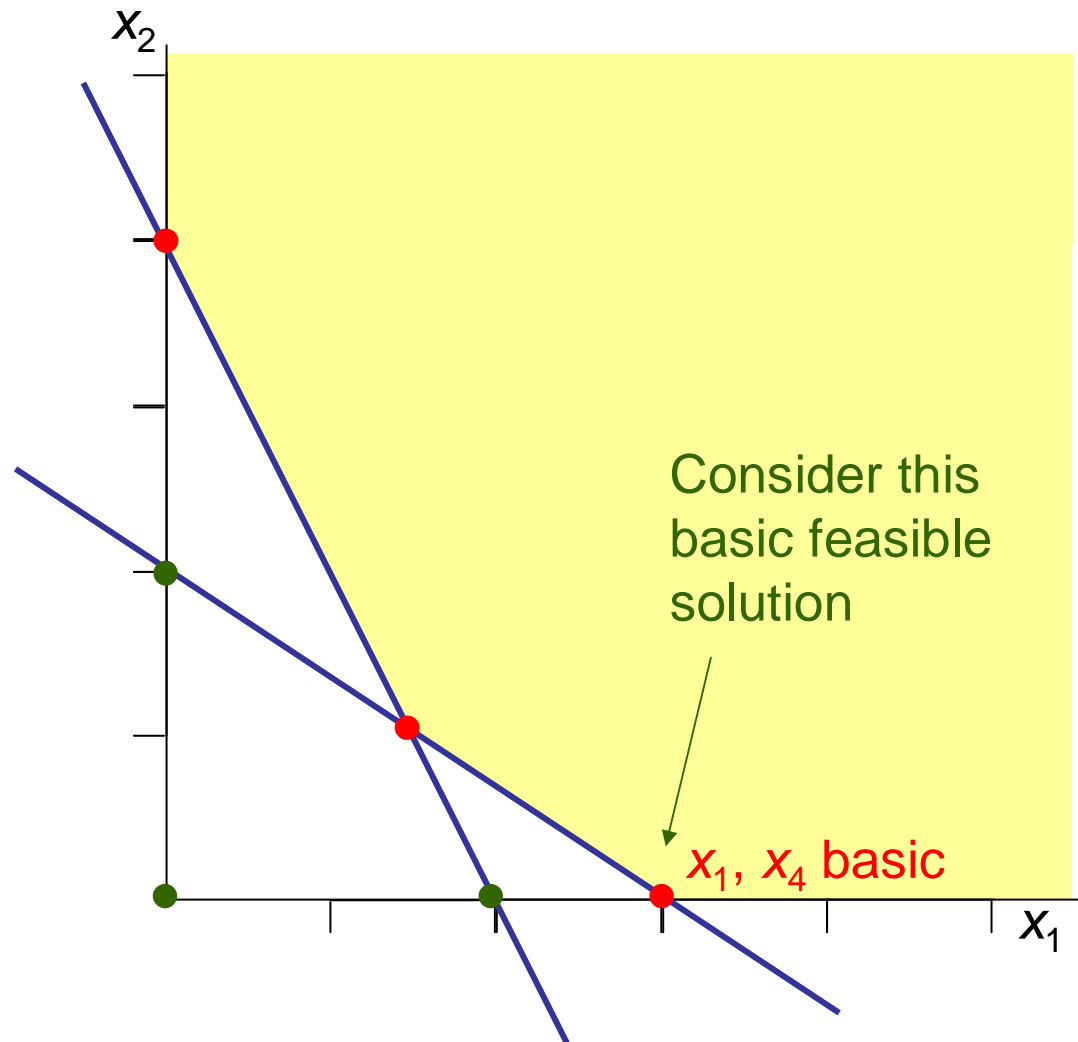
Example...

$$\min 4x_1 + 7x_2$$

$$2x_1 + 3x_2 - x_3 = 6$$

$$2x_1 + x_2 - x_4 = 4$$

$$x_1, x_2, x_3, x_4 \geq 0$$



Example...

Write...

$$\min \quad 4x_1 + 7x_2$$

$$2x_1 + 3x_2 - x_3 = 6$$

$$2x_1 + x_2 - x_4 = 4$$

$$x_1, x_2, x_3, x_4 \geq 0$$

as...

$$\begin{aligned} \min \quad & C_B x_B + C_N x_N \\ & \begin{bmatrix} 4 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_4 \end{bmatrix} + \begin{bmatrix} 7 & 0 \end{bmatrix} \begin{bmatrix} x_2 \\ x_3 \end{bmatrix} \\ & Bx_B + Nx_N = b \\ & \begin{bmatrix} 2 & 0 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_4 \end{bmatrix} + \begin{bmatrix} 3 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \end{bmatrix} \\ & \begin{bmatrix} x_1 \\ x_4 \end{bmatrix}, \begin{bmatrix} x_2 \\ x_3 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{aligned}$$

Example...

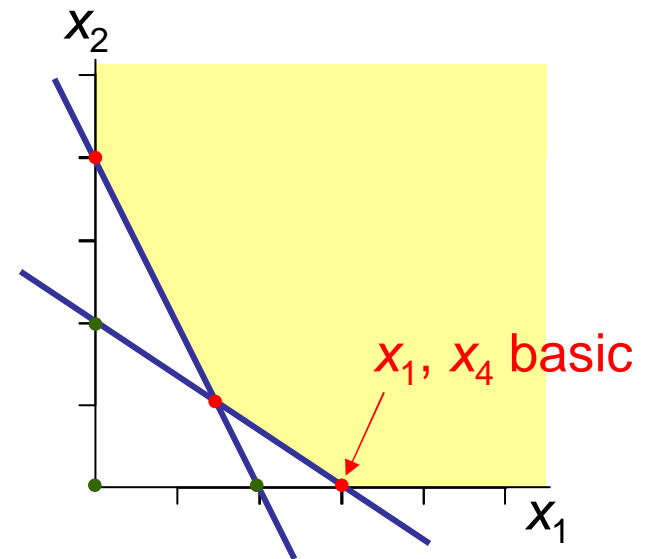
$$\begin{aligned}
 & \min \quad \overbrace{\begin{bmatrix} 4 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_4 \end{bmatrix}}^{C_B X_B} + \overbrace{\begin{bmatrix} 7 & 0 \end{bmatrix} \begin{bmatrix} x_2 \\ x_3 \end{bmatrix}}^{C_N X_N} \\
 & \underbrace{\begin{bmatrix} 2 & 0 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_4 \end{bmatrix}}_{B X_B} + \underbrace{\begin{bmatrix} 3 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_4 \end{bmatrix}}_{N X_N} = \underbrace{\begin{bmatrix} 6 \\ 4 \end{bmatrix}}_b \\
 & \begin{bmatrix} x_1 \\ x_4 \end{bmatrix}, \begin{bmatrix} x_1 \\ x_4 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix}
 \end{aligned}$$

Example...

$$\begin{aligned} \min \quad & \overset{C_B X_B}{\boxed{\begin{bmatrix} 4 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_4 \end{bmatrix}}} + \overset{C_N X_N}{\boxed{\begin{bmatrix} 7 & 0 \end{bmatrix} \begin{bmatrix} x_2 \\ x_3 \end{bmatrix}}} \\ \overset{B X_B}{\boxed{\begin{bmatrix} 2 & 0 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_4 \end{bmatrix}}} + \overset{N X_N}{\boxed{\begin{bmatrix} 3 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_2 \\ x_3 \end{bmatrix}}} &= \underset{b}{\boxed{\begin{bmatrix} 6 \\ 4 \end{bmatrix}}} \\ \begin{bmatrix} x_1 \\ x_4 \end{bmatrix}, \begin{bmatrix} x_2 \\ x_3 \end{bmatrix} &\geq \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{aligned}$$

Basic solution is

$$\begin{aligned} x_B &= B^{-1}b - B^{-1}Nx_N = B^{-1}b \\ = \begin{bmatrix} x_1 \\ x_4 \end{bmatrix} &= \begin{bmatrix} 1/2 & 0 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 6 \\ 4 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \end{aligned}$$



Example...

$$\begin{aligned} \min \quad & \overset{c_B x_B}{\boxed{[4 \ 0] \begin{bmatrix} x_1 \\ x_4 \end{bmatrix}}} + \overset{c_N x_N}{\boxed{[7 \ 0] \begin{bmatrix} x_2 \\ x_3 \end{bmatrix}}} \\ & \overset{Bx_B}{\boxed{\begin{bmatrix} 2 & 0 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_4 \end{bmatrix}}} + \overset{Nx_N}{\boxed{\begin{bmatrix} 3 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_4 \end{bmatrix}}} = \boxed{\begin{bmatrix} 6 \\ 4 \end{bmatrix}} \\ & \begin{bmatrix} x_1 \\ x_4 \end{bmatrix}, \begin{bmatrix} x_1 \\ x_4 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{aligned}$$

Basic solution is

$$\begin{aligned} x_B &= B^{-1}b - B^{-1}Nx_N = B^{-1}b \\ &= \begin{bmatrix} x_1 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1/2 & 0 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 6 \\ 4 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \end{aligned}$$

Reduced costs are

$$\begin{aligned} & c_N - c_B B^{-1}N \\ &= [7 \ 0] - [4 \ 0] \begin{bmatrix} 1/2 & 0 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 3 & -1 \\ 1 & 0 \end{bmatrix} \\ &= [1 \ 2] \geq [0 \ 0] \end{aligned}$$

Solution is optimal

Linear Programming Duality

An LP can be viewed as an inference problem...

$$\begin{array}{l} \min \quad cx \\ Ax \geq b \\ x \geq 0 \end{array} = \begin{array}{l} \max \quad v \\ Ax \geq b \overset{x \geq 0}{\Rightarrow} cx \geq v \\ \text{implies} \end{array}$$

Dual problem: Find the tightest lower bound on the objective function that is implied by the constraints.

An LP can be viewed as an inference problem...

$$\min cx = \max v$$

$$Ax \geq b$$

$$x \geq 0$$

$$Ax \geq b \stackrel{x \geq 0}{\Rightarrow} cx \geq v$$

That is, some **surrogate**
(nonnegative linear
combination) of
 $Ax \geq b$ dominates $cx \geq v$

From Farkas Lemma: If $Ax \geq b, x \geq 0$ is feasible,

$$Ax \geq b \stackrel{x \geq 0}{\Rightarrow} cx \geq v \quad \text{iff} \quad \lambda Ax \geq \lambda b \quad \boxed{\text{dominates}} \quad cx \geq v$$

for some $\lambda \geq 0$

$$\lambda A \leq c \quad \text{and} \quad \lambda b \geq v$$

An LP can be viewed as an inference problem...

$$\begin{array}{llll}
 \min \quad cx & = & \max \quad v & = & \max \quad \lambda b \\
 Ax \geq b & & Ax \geq b \stackrel{x \geq 0}{\Rightarrow} cx \geq v & & \lambda A \leq c \\
 x \geq 0 & & & & \lambda \geq 0
 \end{array}$$

This is the **classical LP dual**

From Farkas Lemma: If $Ax \geq b, x \geq 0$ is feasible,

$$Ax \geq b \stackrel{x \geq 0}{\Rightarrow} cx \geq v \quad \text{iff} \quad \lambda Ax \geq \lambda b \quad \boxed{\text{dominates}} \quad cx \geq v$$

for some $\lambda \geq 0$

$$\lambda A \leq c \quad \text{and} \quad \lambda b \geq v$$

This equality is called **strong duality**.

$$\begin{array}{ll} \min & cx \\ & Ax \geq b \\ & x \geq 0 \end{array} = \begin{array}{ll} \max & \lambda b \\ & \lambda A \leq c \\ & \lambda \geq 0 \end{array}$$

This is the **classical LP dual**

If $Ax \geq b$, $x \geq 0$ is feasible

Note that the dual of the dual is the **primal** (i.e., the original LP).

Example

Primal

$$\min 4x_1 + 7x_2 =$$

$$2x_1 + 3x_2 \geq 6 \quad (\lambda_1)$$

$$2x_1 + x_2 \geq 4 \quad (\lambda_1)$$

$$x_1, x_2 \geq 0$$

Dual

$$\max 6\lambda_1 + 4\lambda_2 = 12$$

$$2\lambda_1 + 2\lambda_2 \leq 4 \quad (x_1)$$

$$3\lambda_1 + \lambda_2 \leq 7 \quad (x_2)$$

$$\lambda_1, \lambda_2 \geq 0$$

A dual solution is $(\lambda_1, \lambda_2) = (2, 0)$

$$2x_1 + 3x_2 \geq 6 \quad \cdot (\lambda_1 = 2)$$

$$2x_1 + x_2 \geq 4 \quad \cdot (\lambda_2 = 0)$$

← Dual multipliers

$$4x_1 + 6x_2 \geq 12 \quad \leftarrow \text{Surrogate}$$

↓ dominates

$$4x_1 + 7x_2 \geq 12 \quad \leftarrow \text{Tightest bound on cost}$$

Weak Duality

If x^* is feasible in the primal problem

$$\begin{aligned} \min \quad & cx \\ \text{subject to} \quad & Ax \geq b \\ & x \geq 0 \end{aligned}$$

and λ^* is feasible in the dual problem

$$\begin{aligned} \max \quad & \lambda b \\ \text{subject to} \quad & \lambda A \leq c \\ & \lambda \geq 0 \end{aligned}$$

then $cx^* \geq \lambda^* b$.

This is because

$$cx^* \geq \lambda^* Ax^* \geq \lambda^* b$$

λ^* is dual
feasible
and $x^* \geq 0$

x^* is primal
feasible
and $\lambda^* \geq 0$

Dual multipliers as marginal costs

Suppose we perturb the RHS of an LP (i.e., change the requirement levels):

$$\begin{aligned} \min \quad & cx \\ & Ax \geq b + \Delta b \\ & x \geq 0 \end{aligned}$$

The dual of the perturbed LP has the same constraints at the original LP:

$$\begin{aligned} \max \quad & \lambda(b + \Delta b) \\ & \lambda A \leq c \\ & \lambda \geq 0 \end{aligned}$$

So an optimal solution λ^* of the original dual is feasible in the perturbed dual.

Dual multipliers as marginal costs

Suppose we perturb the RHS of an LP (i.e., change the requirement levels):

$$\begin{aligned} \min \quad & cx \\ Ax \geq & b + \Delta b \\ x \geq & 0 \end{aligned}$$

By weak duality, the optimal value of the perturbed LP is at least $\lambda^*(b + \Delta b) = \boxed{\lambda^*b} + \lambda^*\Delta b$.

Optimal value of original LP, by strong duality.

So λ_i^* is a lower bound on the marginal cost of increasing the i -th requirement by one unit ($\Delta b_i = 1$).

If $\lambda_i^* > 0$, the i -th constraint must be tight (**complementary slackness**).

Dual of an LP in equality form

Primal

$$\min c_B x_B + c_N x_N$$

$$Bx_B + Nx_N = b \quad (\lambda)$$

$$x_B, x_N \geq 0$$

Dual

$$\max \lambda b$$

$$\lambda B \leq c_B \quad (x_B)$$

$$\lambda N \leq c_N \quad (x_B)$$

λ unrestricted

Dual of an LP in equality form

Primal

$$\min c_B x_B + c_N x_N$$

$$Bx_B + Nx_N = b \quad (\lambda)$$

$$x_B, x_N \geq 0$$

Dual

$$\max \lambda b$$

$$\lambda B \leq c_B \quad (x_B)$$

$$\lambda N \leq c_N \quad (x_B)$$

λ unrestricted

Recall that reduced cost vector is $c_N - \boxed{c_B B^{-1} N} = c_N - \lambda N$

λ

this solves the dual
if $(x_B, 0)$ solves the primal

Dual of an LP in equality form

Primal

$$\min c_B x_B + c_N x_N$$

$$Bx_B + Nx_N = b \quad (\lambda)$$

$$x_B, x_N \geq 0$$

Dual

$$\max \lambda b$$

$$\lambda B \leq c_B \quad (x_B)$$

$$\lambda N \leq c_N \quad (x_B)$$

λ unrestricted

Recall that reduced cost vector is $c_N - \boxed{c_B B^{-1}} N = c_N - \lambda N$

Check: $\lambda B = c_B B^{-1} B = c_B$

$$\lambda N = c_B B^{-1} N \leq c_N$$

λ

this solves the dual
if $(x_B, 0)$ solves the primal

Because reduced cost is nonnegative
at optimal solution $(x_B, 0)$.

Dual of an LP in equality form

Primal

$$\min c_B x_B + c_N x_N$$

$$Bx_B + Nx_N = b \quad (\lambda)$$

$$x_B, x_N \geq 0$$

Dual

$$\max \lambda b$$

$$\lambda B \leq c_B \quad (x_B)$$

$$\lambda N \leq c_N \quad (x_B)$$

λ unrestricted

Recall that reduced cost vector is $c_N - \boxed{c_B B^{-1}} N = c_N - \lambda N$

λ

this solves the dual
if $(x_B, 0)$ solves the primal

In the example,

$$\lambda = c_B B^{-1} = \begin{bmatrix} 4 & 0 \end{bmatrix} \begin{bmatrix} 1/2 & 0 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \end{bmatrix}$$

Dual of an LP in equality form

Primal

$$\min c_B x_B + c_N x_N$$

$$Bx_B + Nx_N = b \quad (\lambda)$$

$$x_B, x_N \geq 0$$

Dual

$$\max \lambda b$$

$$\lambda B \leq c_B \quad (x_B)$$

$$\lambda N \leq c_N \quad (x_B)$$

λ unrestricted

Recall that reduced cost vector is $c_N - \underbrace{c_B B^{-1}}_{\lambda} N = c_N - \lambda N$

Note that the reduced cost of an individual variable x_j is $r_j = c_j - \lambda \underbrace{A_j}_{\text{Column } j \text{ of } A}$

LP-based Domain Filtering

$$\min \quad cx$$

Let $Ax \geq b$ be an LP relaxation of a CP problem.
 $x \geq 0$

- One way to filter the domain of x_j is to minimize and maximize x_j subject to $Ax \geq b, x \geq 0$.
 - This is time consuming.
- A faster method is to use **dual multipliers** to derive valid inequalities.
 - A special case of this method uses **reduced costs** to bound or fix variables.
 - **Reduced-cost variable fixing** is a widely used technique in OR.

Suppose:

$\min \quad cx$ has optimal solution x^* , optimal value v^* , and
 $Ax \geq b$ optimal dual solution λ^* .
 $x \geq 0$

...and $\lambda_i^* > 0$, which means the i -th constraint is tight
(complementary slackness);

...and the LP is a relaxation of a CP problem;

...and we have a feasible solution of the CP problem with value
 U , so that U is an upper bound on the optimal value.

Supposing $\begin{array}{l} \min \quad cx \\ Ax \geq b \\ x \geq 0 \end{array}$ has optimal solution x^* , optimal value v^* , and optimal dual solution λ^* :

If x were to change to a value other than x^* , the LHS of i -th constraint $A_i x \geq b_i$ would change by some amount Δb_i .

Since the constraint is tight, this would increase the optimal value as much as changing the constraint to $A_i x \geq b_i + \Delta b_i$.

So it would increase the optimal value at least $\lambda_i^* \Delta b_i$.

Supposing $\begin{array}{l} \min \quad cx \\ Ax \geq b \\ x \geq 0 \end{array}$ has optimal solution x^* , optimal value v^* , and optimal dual solution λ^* :

We have found: a change in x that changes $A^i x$ by Δb_i increases the optimal value of LP at least $\lambda_i^* \Delta b_i$.

Since optimal value of the LP \leq optimal value of the CP $\leq U$,
we have $\lambda_i^* \Delta b_i \leq U - v^*$, or
$$\Delta b_i \leq \frac{U - v^*}{\lambda_i^*}$$

Supposing $\begin{array}{l} \min \quad cx \\ Ax \geq b \\ x \geq 0 \end{array}$ has optimal solution x^* , optimal value v^* , and optimal dual solution λ^* :

We have found: a change in x that changes $A^i x$ by Δb_i increases the optimal value of LP at least $\lambda_i^* \Delta b_i$.

Since optimal value of the LP \leq optimal value of the CP $\leq U$,
we have $\lambda_i^* \Delta b_i \leq U - v^*$, or
$$\Delta b_i \leq \frac{U - v^*}{\lambda_i^*}$$

Since $\Delta b_i = A^i x - A^i x^* = A^i x - b_i$, this implies the inequality

$$A^i x \leq b_i + \frac{U - v^*}{\lambda_i^*}$$

...which can be propagated.

Example

$$\min 4x_1 + 7x_2$$

$$2x_1 + 3x_2 \geq 6 \quad (\lambda_1 = 2)$$

$$2x_1 + x_2 \geq 4 \quad (\lambda_1 = 0)$$

$$x_1, x_2 \geq 0$$

Suppose we have a feasible solution of the original CP with value $U = 13$.

Since the first constraint is tight, we can propagate the inequality

$$A^1 x \leq b_1 + \frac{U - v^*}{\lambda_1^*}$$

$$\text{or} \quad 2x_1 + 3x_2 \leq 6 + \frac{13 - 12}{2} = 6.5$$

Reduced-cost domain filtering

Suppose $x_j^* = 0$, which means the constraint $x_j \geq 0$ is tight.

The inequality $A^i x \leq b_i + \frac{U - v^*}{\lambda_i^*}$ becomes $x_j \leq \frac{U - v^*}{\boxed{r_j}}$

The dual multiplier for $x_j \geq 0$ is the reduced cost r_j of x_j , because increasing x_j (currently 0) by 1 increases optimal cost by r_j .

Similar reasoning can bound a variable below when it is at its upper bound.

Example

$$\min 4x_1 + 7x_2$$

$$2x_1 + 3x_2 \geq 6 \quad (\lambda_1 = 2)$$

$$2x_1 + x_2 \geq 4 \quad (\lambda_1 = 0)$$

$$x_1, x_2 \geq 0$$

Suppose we have a feasible solution of the original CP with value $U = 13$.

$$\text{Since } x_2^* = 0, \text{ we have } x_2 \leq \frac{U - v^*}{r_2}$$

$$\text{or } x_2 \leq \frac{13 - 12}{2} = 0.5$$

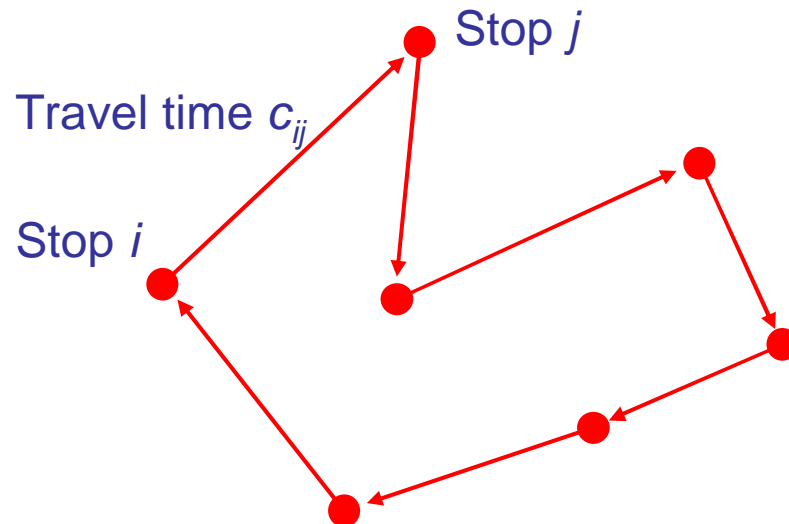
If x_2 is required to be integer, we can fix it to zero.
This is **reduced-cost variable fixing**.

Example: Single-Vehicle Routing

A vehicle must make several stops and return home, perhaps subject to time windows.

The objective is to find the order of stops that minimizes travel time.

This is also known as the **traveling salesman problem** (with time windows).



Assignment Relaxation



$$\begin{aligned} \min \sum_{ij} c_{ij} x_{ij} & \quad \leftarrow = 1 \text{ if stop } i \text{ immediately precedes stop } j \\ \sum_j x_{ij} = \sum_j x_{ji} = 1, \text{ all } i & \quad \leftarrow \text{Stop } i \text{ is preceded and followed by exactly one stop.} \\ x_{ij} \in \{0, 1\}, \text{ all } i, j \end{aligned}$$

Assignment Relaxation



$$\begin{aligned} \min \quad & \sum_{ij} c_{ij} x_{ij} \quad \leftarrow = 1 \text{ if stop } i \text{ immediately precedes stop } j \\ \sum_j x_{ij} = \sum_j x_{ji} = 1, \quad & \text{all } i \quad \leftarrow \text{Stop } i \text{ is preceded and followed by exactly one stop.} \\ 0 \leq x_{ij} \leq 1, \quad & \text{all } i, j \end{aligned}$$

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 **reduced-cost variable fixing** can be very useful in a CP context.

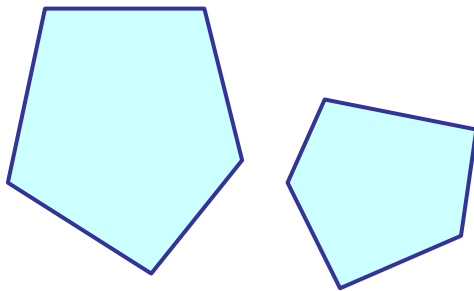
Disjunctions of linear systems

Disjunctions of linear systems often occur naturally in problems and can be given a convex hull relaxation.

A disjunction of linear systems represents a union of polyhedra.

$$\min \quad cx$$

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



Relaxing a disjunction of linear systems

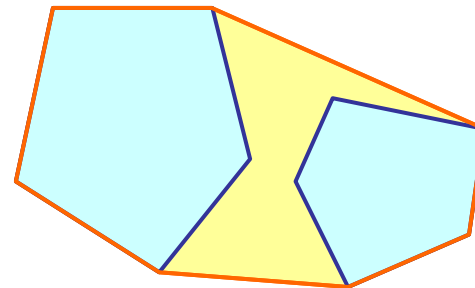
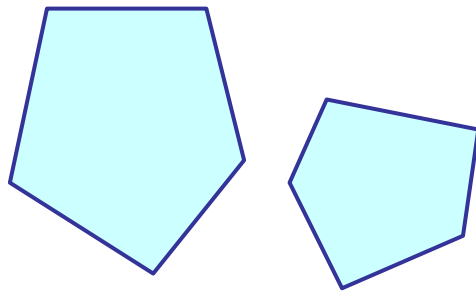
Disjunctions of linear systems often occur naturally in problems and can be given a convex hull relaxation.

A disjunction of linear systems represents a union of polyhedra.

We want a convex hull relaxation (tightest linear relaxation).

$$\min \quad cx$$

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



Relaxing a disjunction of linear systems

Disjunctions of linear systems often occur naturally in problems and can be given a convex hull relaxation.

The closure of the convex hull of

$$\min \quad cx$$

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

...is described by

$$\min \quad cx$$

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

$$\sum_k y_k = 1$$

$$x = \sum_k x^k$$

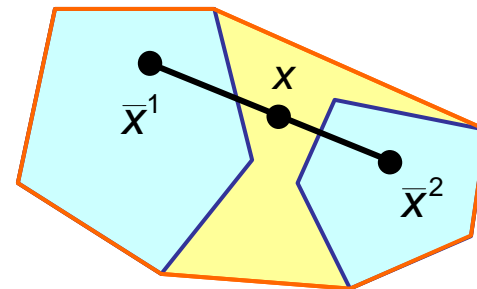
$$0 \leq y_k \leq 1$$

Why?

To derive convex hull relaxation of a disjunction...

Write each solution as a convex combination of points in the polyhedron

$$\begin{aligned} \min \quad & cx \\ & A^k \bar{x}^k \geq b^k, \text{ all } k \\ & \sum_k y_k = 1 \\ & x = \sum_k y_k \bar{x}^k \\ & 0 \leq y_k \leq 1 \end{aligned}$$



Convex hull relaxation
(tightest linear relaxation)

Why?

To derive convex hull relaxation of a disjunction...

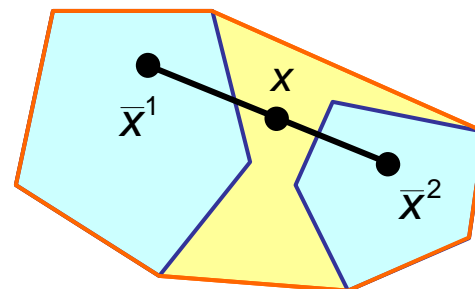
Write each solution as a convex combination of points in the polyhedron

$$\begin{aligned} \min \quad & cx \\ & A^k \bar{x}^k \geq b^k, \text{ all } k \\ & \sum_k y_k = 1 \\ & x = \sum_k y_k \bar{x}^k \\ & 0 \leq y_k \leq 1 \end{aligned}$$

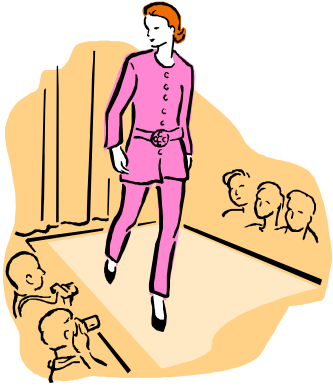
Change of variable

$$x = y_k \bar{x}^k$$

$$\begin{aligned} \min \quad & cx \\ & A^k x^k \geq b^k y_k, \text{ all } k \\ & \sum_k y_k = 1 \\ & x = \sum_k x^k \\ & 0 \leq y_k \leq 1 \end{aligned}$$



Convex hull relaxation
(tightest linear relaxation)



Mixed Integer/Linear Modeling

MILP Representability
Disjunctive Modeling
Knapsack Modeling

Motivation

A **mixed integer/linear programming** (MILP) problem has the form

$$\min \quad cx + dy$$

$$Ax + by \geq b$$

$$x, y \geq 0$$

$$y \text{ integer}$$

- We can **relax** a CP problem by modeling some constraints with an MILP.
- If desired, we can then **relax the MILP** by dropping the integrality constraint, to obtain an LP.
- The LP relaxation can be strengthened with **cutting planes**.
- The first step is to learn **how to write** MILP models.

MILP Representability

A subset S of \mathbb{R}^n is **MILP representable** if it is the projection onto x of some MILP constraint set of the form

$$Ax + Bu + Dy \geq b$$

$$x, y \geq 0$$

$$x \in \mathbb{R}^n, \quad u \in \mathbb{R}^m, \quad y_k \in \{0, 1\}$$

MILP Representability

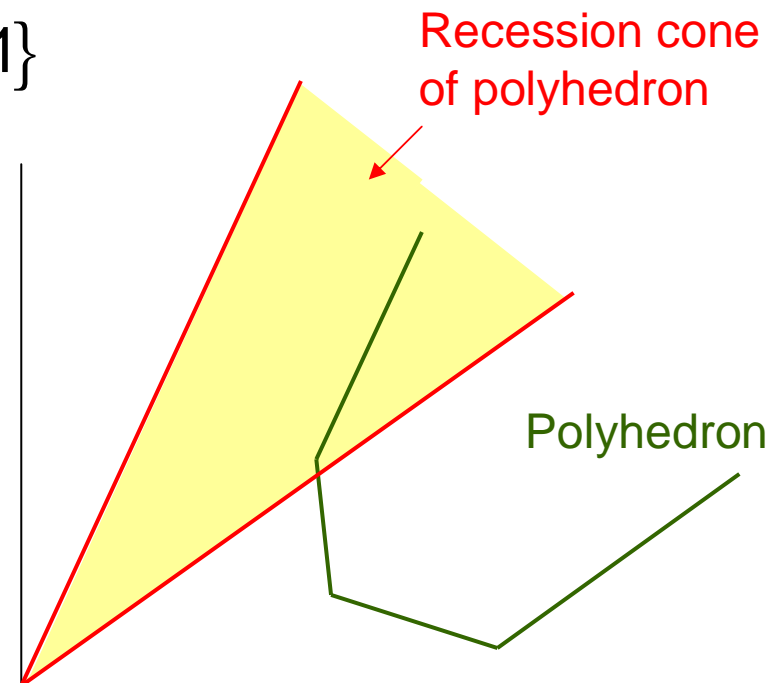
A subset S of \mathbb{R}^n is **MILP representable** if it is the projection onto x of some MILP constraint set of the form

$$Ax + Bu + Dy \geq b$$

$$x, y \geq 0$$

$$x \in \mathbb{R}^n, u \in \mathbb{R}^m, y_k \in \{0,1\}$$

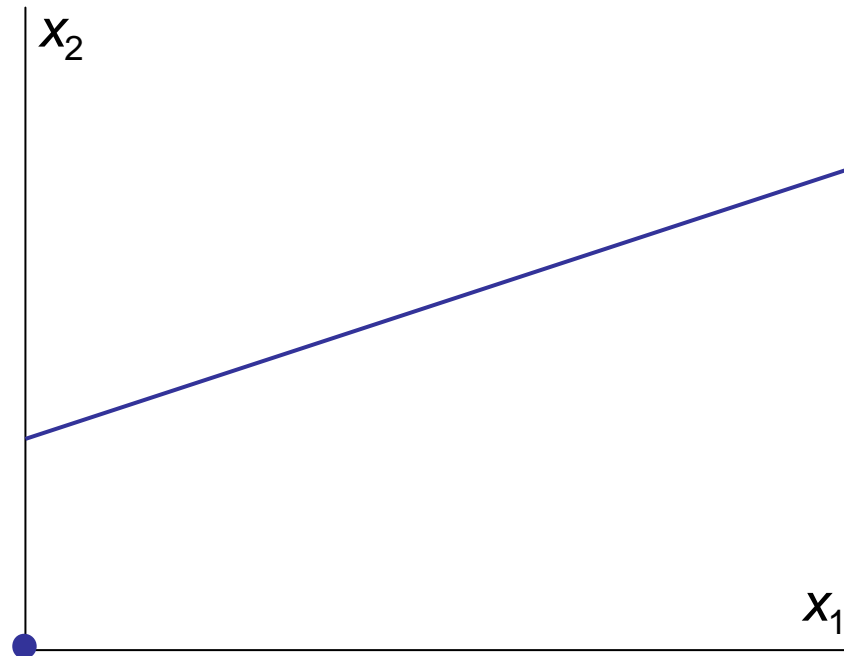
Theorem. $S \subset \mathbb{R}^n$ is MILP representable if and only if S is the union of finitely many polyhedra having the same recession cone.



Example: Fixed charge function

Minimize a fixed charge function:

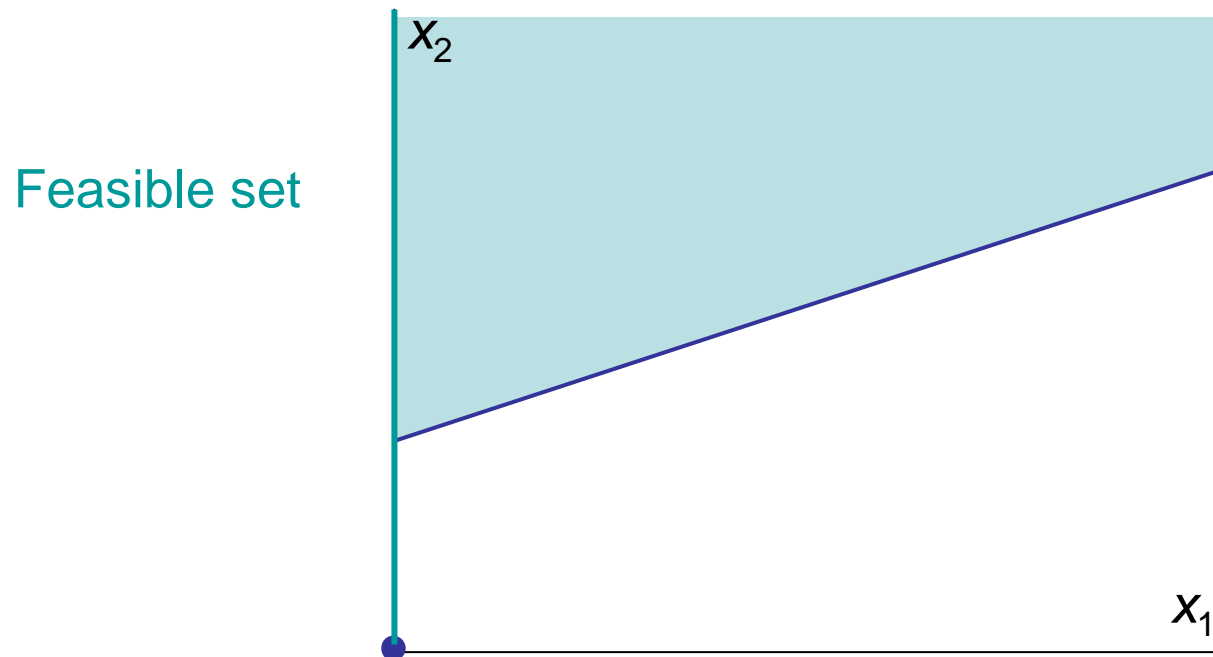
$$\begin{aligned} \min \quad & x_2 \\ x_2 \geq & \begin{cases} 0 & \text{if } x_1 = 0 \\ f + cx_1 & \text{if } x_1 > 0 \end{cases} \\ x_1 \geq & 0 \end{aligned}$$



Example

Minimize a fixed charge function:

$$\begin{aligned} \min \quad & x_2 \\ x_2 \geq \quad & \begin{cases} 0 & \text{if } x_1 = 0 \\ f + cx_1 & \text{if } x_1 > 0 \end{cases} \\ x_1 \geq \quad & 0 \end{aligned}$$

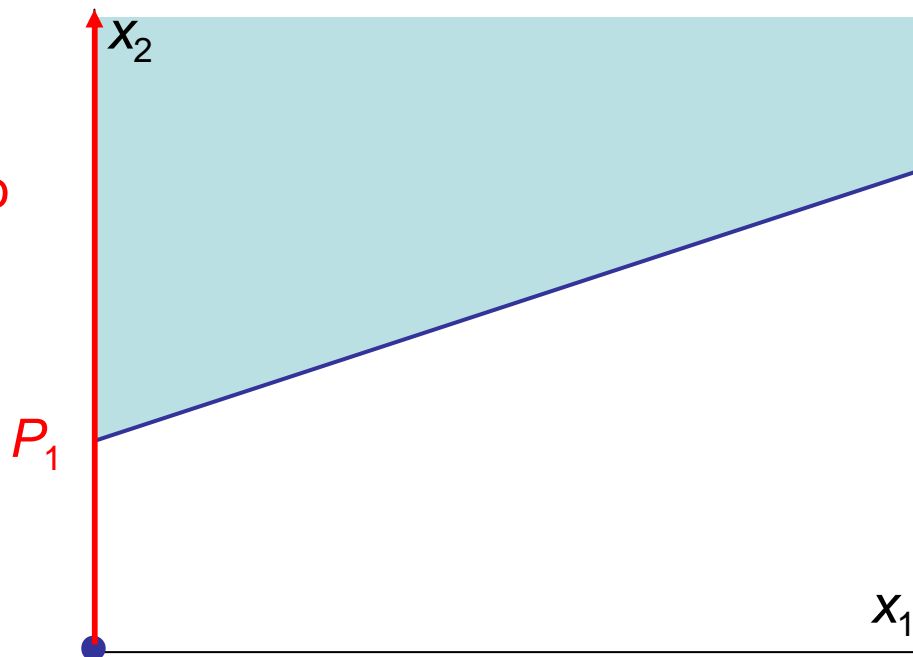


Example

Minimize a fixed charge function:

$$\begin{aligned} \min \quad & x_2 \\ & x_2 \geq \begin{cases} 0 & \text{if } x_1 = 0 \\ f + cx_1 & \text{if } x_1 > 0 \end{cases} \\ & x_1 \geq 0 \end{aligned}$$

Union of two
polyhedra
 P_1, P_2

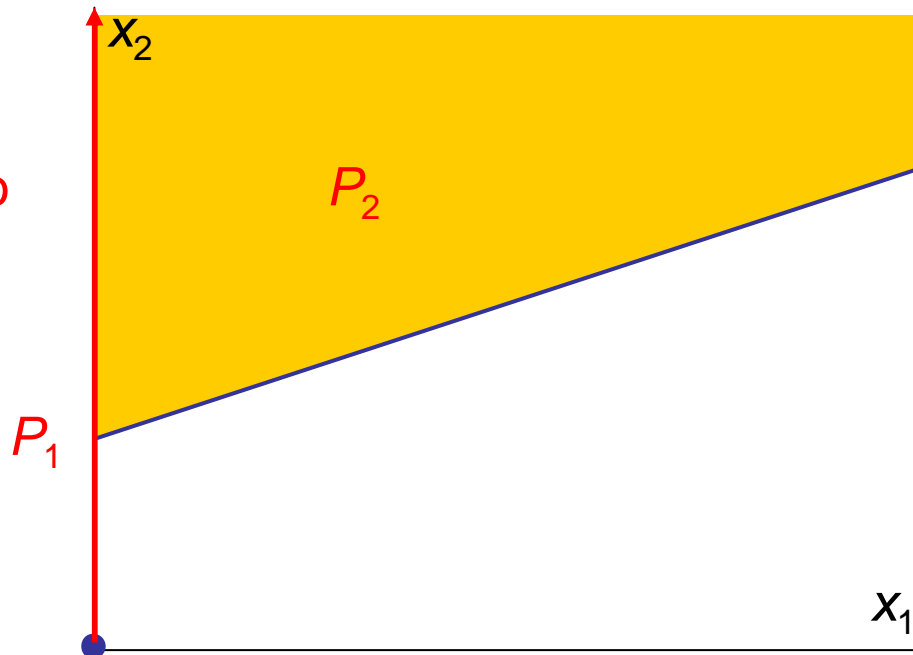


Example

Minimize a fixed charge function:

$$\begin{aligned} \min \quad & x_2 \\ & x_2 \geq \begin{cases} 0 & \text{if } x_1 = 0 \\ f + cx_1 & \text{if } x_1 > 0 \end{cases} \\ & x_1 \geq 0 \end{aligned}$$

Union of two
polyhedra
 P_1, P_2

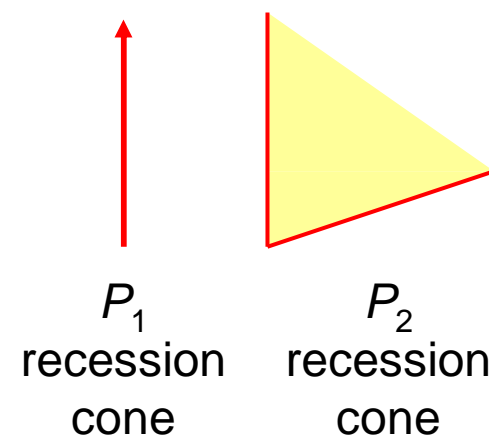
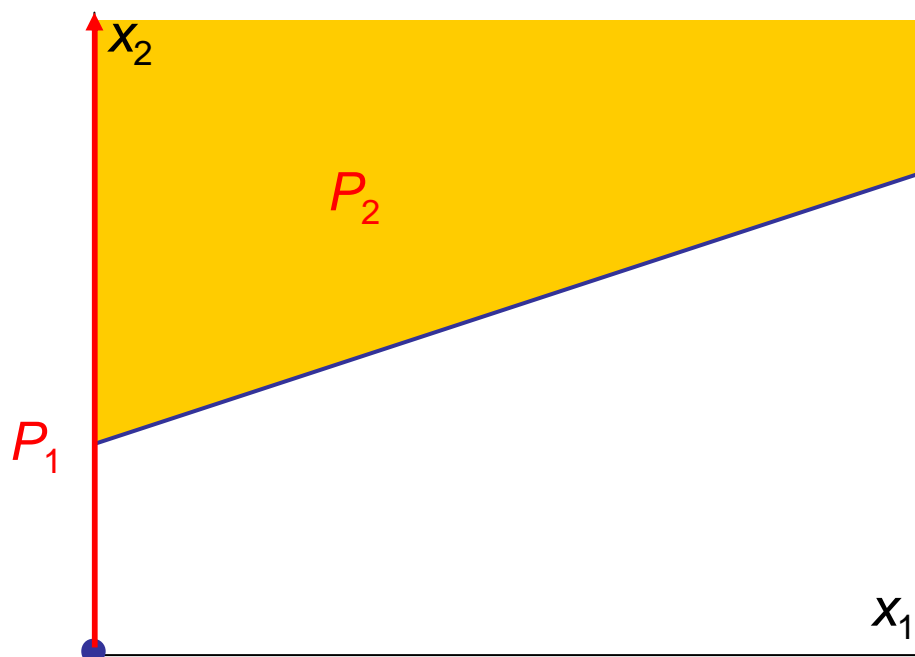


Example

Minimize a fixed charge function:

$$\begin{aligned} \min \quad & x_2 \\ x_2 \geq \quad & \begin{cases} 0 & \text{if } x_1 = 0 \\ f + cx_1 & \text{if } x_1 > 0 \end{cases} \\ x_1 \geq \quad & 0 \end{aligned}$$

The polyhedra have different recession cones.



Example

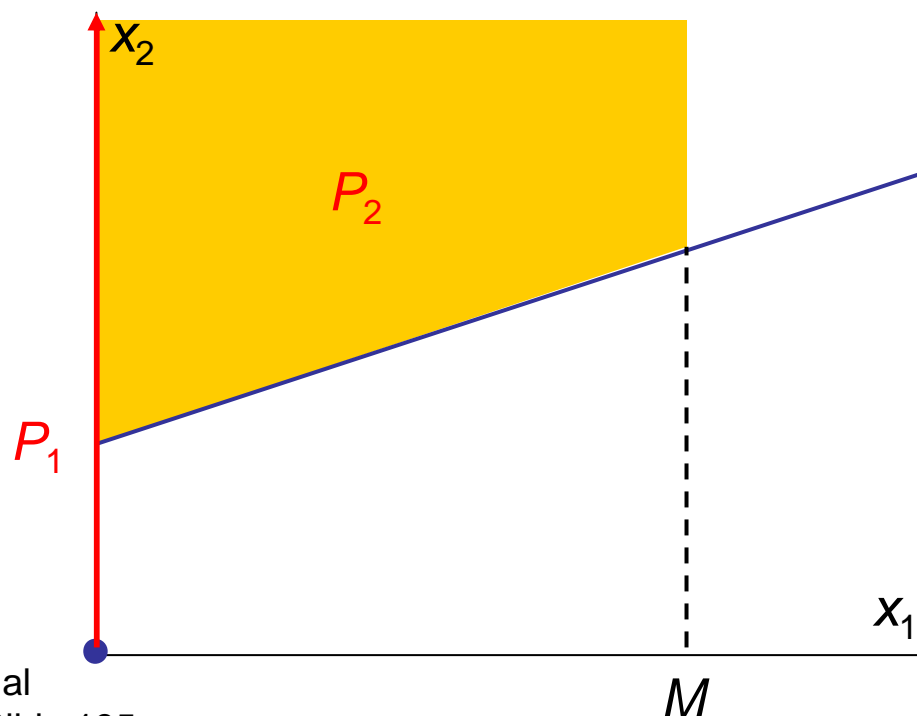
Minimize a fixed charge function:

Add an upper bound on x_1

$$\begin{aligned} \min \quad & x_2 \\ \text{s.t.} \quad & x_2 \geq \begin{cases} 0 & \text{if } x_1 = 0 \\ f + cx_1 & \text{if } x_1 > 0 \end{cases} \end{aligned}$$

$$0 \leq x_1 \leq M$$

The polyhedra have the same recession cone.



P_1
recession
cone

P_2
recession
cone

Modeling a union of polyhedra

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

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

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 .

$$\min \quad cx$$

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

$$\min \quad cx$$

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

$$\sum_k y_k = 1$$

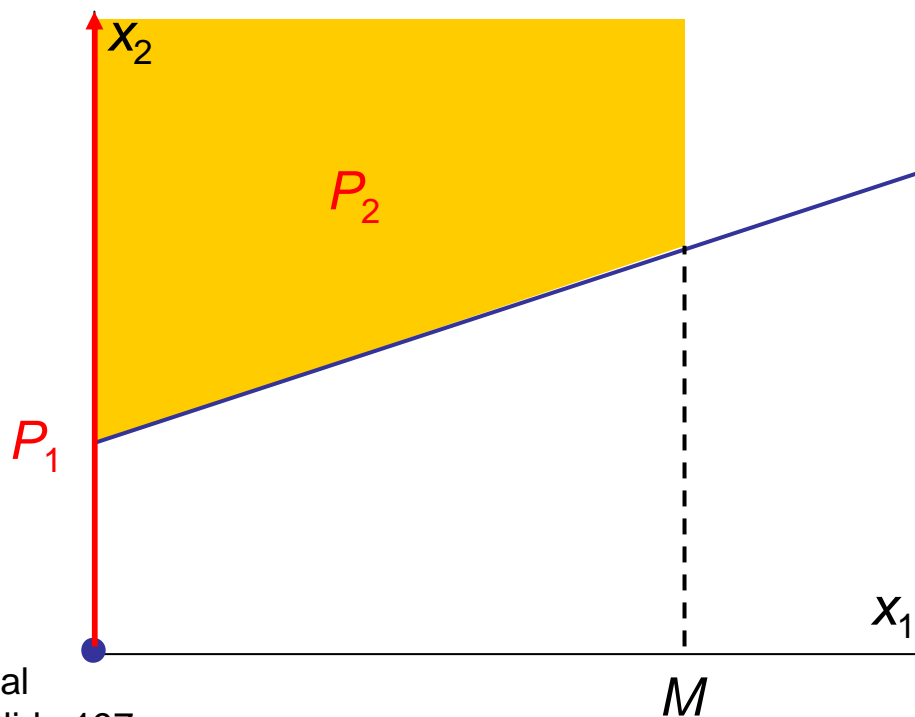
$$x = \sum_k x^k$$

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

Example

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

$$\min x_2$$
$$\left(\begin{array}{l} x_1 = 0 \\ x_2 \geq 0 \end{array} \right) \vee \left(\begin{array}{l} 0 \leq x_1 \leq M \\ x_2 \geq f + cx_1 \end{array} \right)$$



Example

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

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 .

$$\min x_2$$
$$\left(\begin{array}{l} x_1 = 0 \\ x_2 \geq 0 \end{array} \right) \vee \left(\begin{array}{l} 0 \leq x_1 \leq M \\ x_2 \geq f + cx_1 \end{array} \right)$$

$$\min cx$$
$$x_1^1 = 0, \quad x_2^1 \geq 0$$
$$0 \leq x_1^2 \leq My_2, \quad -cx_1^2 + x_2^2 \geq fy_2$$
$$y_1 + y_2 = 1, \quad y_k \in \{0, 1\}$$
$$x = x^1 + x^2$$

Example

To simplify:

Replace x_1^2 with x_1 .

Replace x_2^2 with x_2 .

Replace y_2 with y .

$$\min x_2$$

$$x_1^1 = 0, \quad x_2^1 \geq 0$$

$$0 \leq x_1^2 \leq My_2, \quad -cx_1^2 + x_2^2 \geq fy_2$$

$$y_1 + y_2 = 1, \quad y_k \in \{0,1\}$$

$$x = x^1 + x^2$$

This yields

$$\min x_2$$

$$0 \leq x_1 \leq My$$

$$x_2 \geq fy + cx_1$$

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

or

$$\min fy + cx$$

$$0 \leq x \leq My$$

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

“Big M”

Disjunctive Modeling

Disjunctions often occur naturally in problems and can be given an MILP model.

Recall that a disjunction of linear systems (representing polyhedra with the same recession cone)

$$\min \quad cx$$

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

...has the MILP model

$$\min \quad cx$$

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

$$\sum_k y_k = 1$$

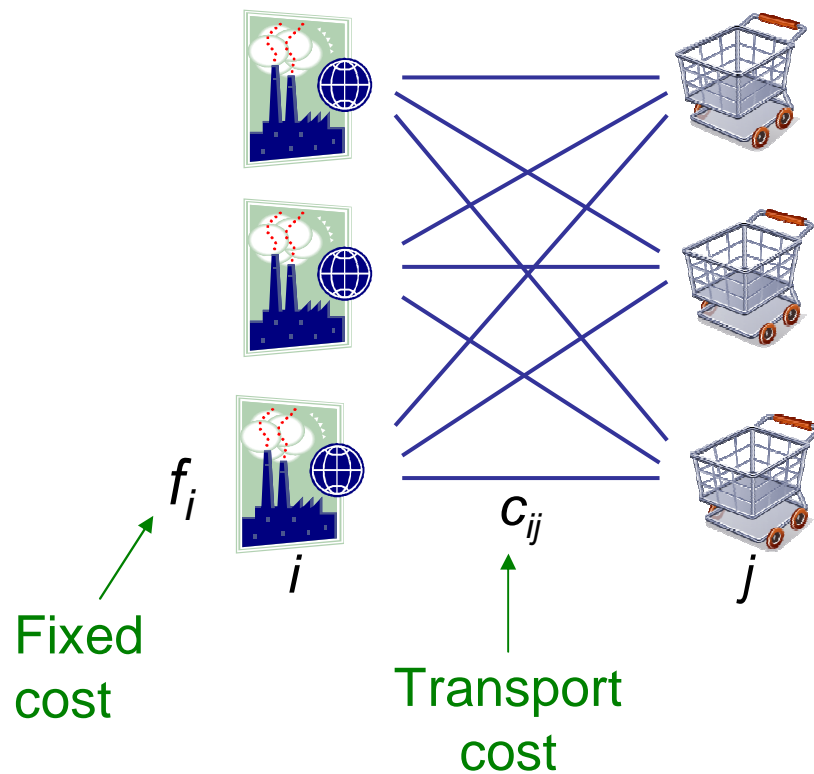
$$x = \sum_k x^k$$

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

Example: Uncapacitated facility location

m possible
factory
locations

n markets



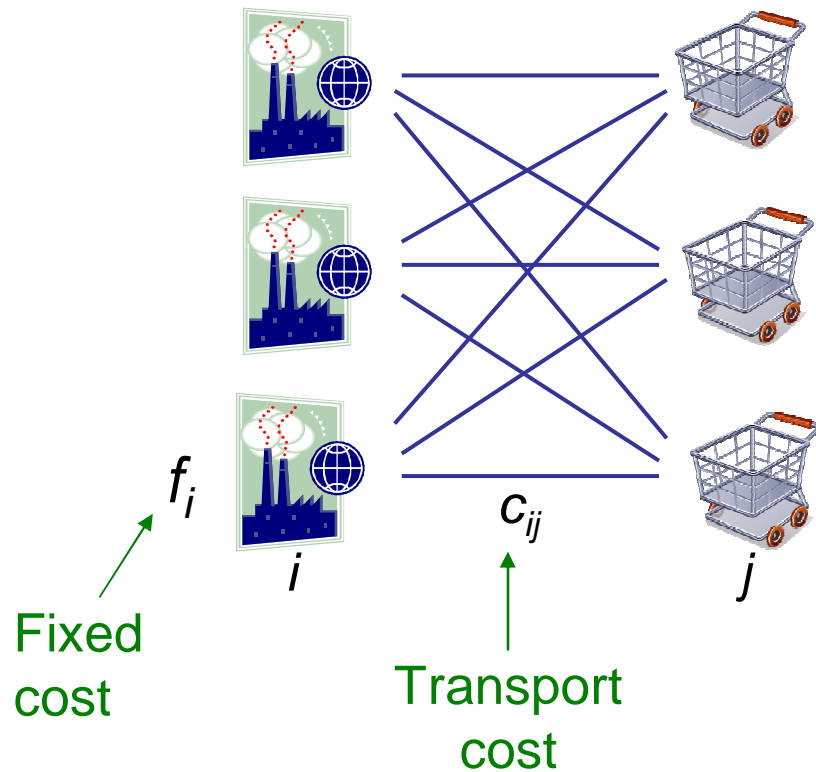
Locate factories to serve markets so as to minimize total fixed cost and transport cost.

No limit on production capacity of each factory.

Uncapacitated facility location

m possible
factory
locations

n markets



Fraction of
market j 's demand
satisfied from
location i

Disjunctive model:

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

$$\left(\begin{array}{l} x_{ij} = 0, \text{ all } j \\ z_i = 0 \end{array} \right) \vee \left(\begin{array}{l} 0 \leq x_{ij} \leq 1, \text{ all } j \\ z_i \geq f_i \end{array} \right), \text{ all } i$$

$$\sum_i x_{ij} = 1, \text{ all } j$$

No factory
at location i

Factory
at location i

Uncapacitated facility location



MILP formulation:

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

$$0 \leq x_{ij} \leq y_i, \quad \text{all } i, j$$

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

Disjunctive model:

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

$$\left(\begin{array}{l} x_{ij} = 0, \text{ all } j \\ z_i = 0 \end{array} \right) \vee \left(\begin{array}{l} 0 \leq x_{ij} \leq 1, \text{ all } j \\ z_i \geq f_i \end{array} \right), \quad \text{all } i$$

$$\sum_i x_{ij} = 1, \quad \text{all } j$$

No factory
at location i

Factory
at location i

Uncapacitated facility location



MILP formulation:

$$\begin{aligned} \min \quad & \sum_i f_i y_i + \sum_{ij} c_{ij} x_{ij} \\ 0 \leq x_{ij} \leq y_i, \quad & \text{all } i, j \\ y_i \in \{0, 1\} \end{aligned}$$

Beginner's model:

$$\begin{aligned} \min \quad & \sum_i f_i y_i + \sum_{ij} c_{ij} x_{ij} \\ \sum_j x_{ij} \leq ny_i, \quad & \text{all } i, j \\ y_i \in \{0, 1\} \end{aligned}$$

Maximum output
from location i

Based on capacitated location model.

It has a **weaker continuous relaxation**
(obtained by replacing $y_i \in \{0, 1\}$ with $0 \leq y_i \leq 1$).

This beginner's mistake can be avoided by
starting with disjunctive formulation.

Knapsack Modeling

- Knapsack models consist of **knapsack covering** and **knapsack packing** constraints.
- The freight transfer model presented earlier is an example.
- We will consider a similar example that combines disjunctive and knapsack modeling.
- Most OR professionals are unlikely to write a model as good as the one presented here.

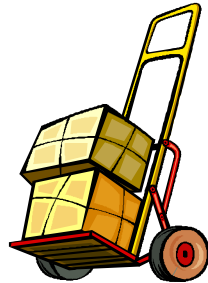


Note on tightness of knapsack models

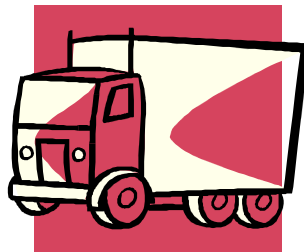
- The continuous relaxation of a knapsack model is not in general a convex hull relaxation.
 - A disjunctive formulation would provide a convex hull relaxation, but there are exponentially many disjuncts.
- Knapsack cuts can significantly tighten the relaxation.

Example: Package transport

Each package j
has size a_j



Each truck i has
capacity Q_i and
costs c_i to
operate



Disjunctive model

Knapsack
constraints

$$\min \sum_i z_i$$

$$\sum_i Q_i y_i \geq \sum_j a_j; \quad \sum_i x_{ij} = 1, \text{ all } j$$

$$\left(\begin{array}{l} y_i = 1 \\ z_i = c_i \\ \sum_j a_j x_{ij} \leq Q_i \\ 0 \leq x_{ij} \leq 1, \text{ all } j \end{array} \right) \vee \left(\begin{array}{l} y_i = 0 \\ z_i = 0 \\ x_{ij} = 0 \end{array} \right), \text{ all } i$$

Truck i used

Truck i not used

$$x_{ij}, y_i \in \{0, 1\}$$

1 if truck i carries
package j

1 if truck i is used

Example: Package transport



MILP model

$$\begin{aligned} \min \quad & \sum_i c_i y_i \\ \sum_i Q_i y_i & \geq \sum_j a_j; \quad \sum_i x_{ij} = 1, \text{ all } j \\ \sum_j a_j x_{ij} & \leq Q_i y_i, \text{ all } i \\ x_{ij} & \leq y_i, \text{ all } i, j \\ x_{ij}, y_i & \in \{0, 1\} \end{aligned}$$

Disjunctive model

$$\begin{aligned} \min \quad & \sum_i z_i \\ \sum_i Q_i y_i & \geq \sum_j a_j; \quad \sum_i x_{ij} = 1, \text{ all } j \\ \left(\begin{array}{l} y_i = 1 \\ z_i = c_i \\ \sum_j a_j x_{ij} \leq Q_i \\ 0 \leq x_{ij} \leq 1, \text{ all } j \end{array} \right) & \vee \left(\begin{array}{l} y_i = 0 \\ z_i = 0 \\ x_{ij} = 0 \end{array} \right), \text{ all } i \\ x_{ij}, y_i & \in \{0, 1\} \end{aligned}$$

Example: Package transport



MILP model

$$\min \sum_i c_i y_i$$

$$\sum_i Q_i y_i \geq \sum_j a_j; \quad \sum_i x_{ij} = 1, \text{ all } j$$

$$\sum_j a_j x_{ij} \leq Q_i y_i, \text{ all } i$$

$$x_{ij} \leq y_i, \text{ all } i, j$$

$$x_{ij}, y_i \in \{0, 1\}$$

Most OR professionals would omit this constraint, since it is the sum over i of the next constraint. But it generates very effective knapsack cuts.

Modeling trick;
unobvious without
disjunctive approach



Cutting Planes

0-1 Knapsack Cuts

Gomory Cuts

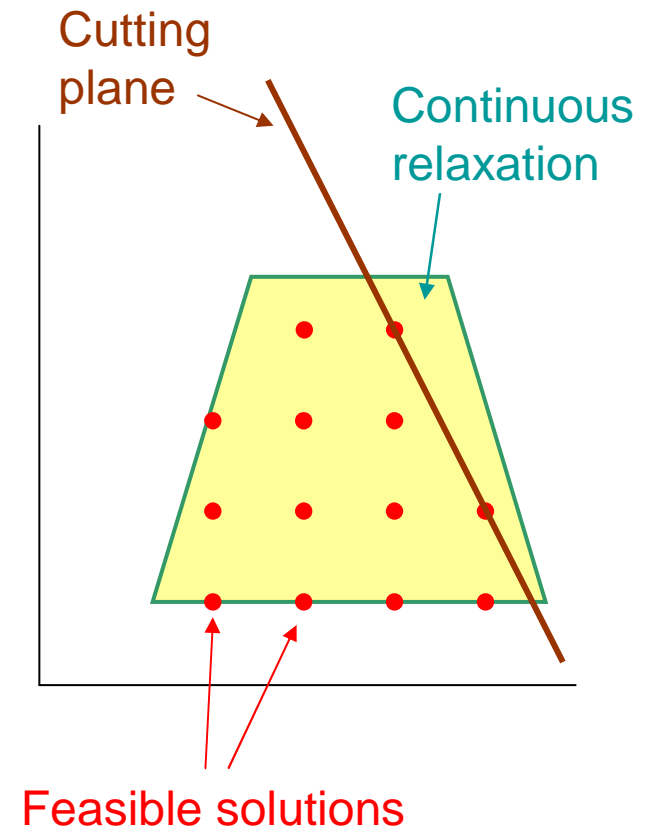
Mixed Integer Rounding Cuts

Example: Product Configuration

To review...

A **cutting plane** (cut, valid inequality) for an MILP 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.



Motivation

- **Cutting planes** (cuts) tighten the continuous relaxation of an MILP model.
- **Knapsack cuts**
 - Generated for individual knapsack constraints.
 - We saw **general integer knapsack cuts** earlier.
 - **0-1 knapsack cuts** and **lifting** techniques are well studied and widely used.
- **Rounding cuts**
 - Generated for the entire MILP, they are widely used.
 - **Gomory cuts** for integer variables only.
 - **Mixed integer rounding cuts** for any MILP.

0-1 Knapsack Cuts

0-1 knapsack cuts are designed for knapsack constraints with 0-1 variables.

The analysis is different from that of general knapsack constraints, to exploit the special structure of 0-1 inequalities.

0-1 Knapsack Cuts

0-1 knapsack cuts are designed for knapsack constraints with 0-1 variables.

The analysis is different from that of general knapsack constraints, to exploit the special structure of 0-1 inequalities.

Consider a 0-1 knapsack packing constraint $ax \leq a_0$. (Knapsack covering constraints are similarly analyzed.)

Index set J is a **cover** if $\sum_{j \in J} a_j > a_0$

The **cover inequality** $\sum_{j \in J} x_j \leq |J| - 1$ is a **0-1 knapsack cut** for $ax \leq a_0$

Only **minimal** covers need be considered.

Example

$J = \{1, 2, 3, 4\}$ is a cover for

$$6x_1 + 5x_2 + 5x_3 + 5x_4 + 8x_5 + 3x_6 \leq 17$$

This gives rise to the cover inequality

$$x_1 + x_2 + x_3 + x_4 \leq 3$$

Index set J is a **cover** if $\sum_{j \in J} a_j > a_0$

The **cover inequality** $\sum_{j \in J} x_j \leq |J| - 1$ is a **0-1 knapsack cut** for $ax \leq a_0$

Only **minimal** covers need be considered.

Sequential lifting

- A cover inequality can often be strengthened by **lifting** it into a higher dimensional space.
 - That is, by adding variables.
- **Sequential lifting** adds one variable at a time.
- **Sequence-independent lifting** adds several variables at once.

Sequential lifting

To lift a cover inequality $\sum_{j \in J} x_j \leq |J| - 1$

add a term to the left-hand side $\sum_{j \in J} x_j + \pi_k x_k \leq |J| - 1$

where π_k is the largest coefficient for which the inequality is still valid.

$$\text{So, } \pi_k = |J| - 1 - \max_{\substack{x_j \in \{0,1\} \\ \text{for } j \in J}} \left\{ \sum_{j \in J} x_j \mid \sum_{j \in J} a_j x_j \leq a_0 - a_k \right\}$$

This can be done repeatedly (by dynamic programming).

Example

Given $6x_1 + 5x_2 + 5x_3 + 5x_4 + 8x_5 + 3x_6 \leq 17$

To lift $x_1 + x_2 + x_3 + x_4 \leq 3$

add a term to the left-hand side $x_1 + x_2 + x_3 + x_4 + \pi_5 x_5 \leq 3$

where

$$\pi_5 = 3 - \max_{\substack{x_j \in \{0,1\} \\ \text{for } j \in \{1,2,3,4\}}} \{x_1 + x_2 + x_3 + x_4 \mid 6x_1 + 5x_2 + 5x_3 + 5x_4 \leq 17 - 8\}$$

This yields $x_1 + x_2 + x_3 + x_4 + 2x_5 \leq 3$

Further lifting leaves the cut unchanged.

But if the variables are added in the order x_6, x_5 , the result is different:

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \leq 3$$

Sequence-independent lifting

- Sequence-independent lifting usually yields a weaker cut than sequential lifting.
 - But it adds all the variables at once and is much faster.
 - Commonly used in commercial MILP solvers.

Sequence-independent lifting

To lift a cover inequality $\sum_{j \in J} x_j \leq |J| - 1$

add terms to the left-hand side $\sum_{j \in J} x_j + \sum_{j \notin J} \rho(a_j) x_k \leq |J| - 1$

where $\rho(u) = \begin{cases} j & \text{if } A_j \leq u \leq A_{j+1} - \Delta \text{ and } j \in \{0, \dots, p-1\} \\ j + (u - A_j) / \Delta & \text{if } A_j - \Delta \leq u < A_j - \Delta \text{ and } j \in \{1, \dots, p-1\} \\ p + (u - A_p) / \Delta & \text{if } A_p - \Delta \leq u \end{cases}$

with $\Delta = \sum_{j \in J} a_j - a_0$ $A_j = \sum_{k=1}^j a_k$

$J = \{1, \dots, p\}$ $A_0 = 0$

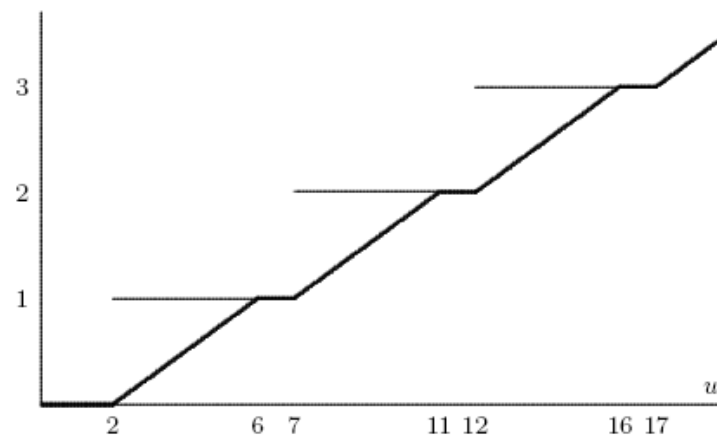
Example

Given $6x_1 + 5x_2 + 5x_3 + 5x_4 + 8x_5 + 3x_6 \leq 17$

To lift $x_1 + x_2 + x_3 + x_4 \leq 3$

Add terms $x_1 + x_2 + x_3 + x_4 + \rho(8)x_5 + \rho(3)x_6 \leq 3$

where $\rho(u)$ is given by



This yields the lifted cut

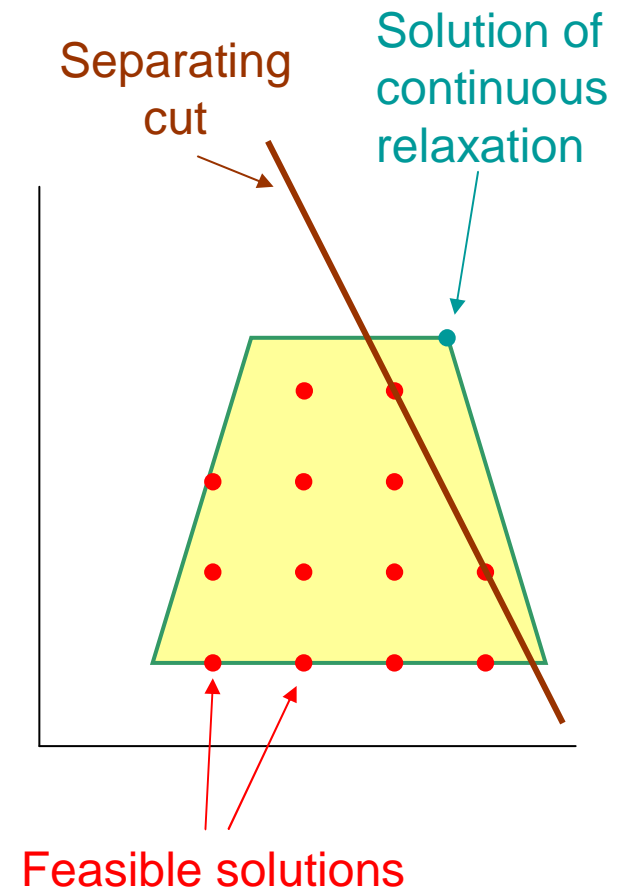
$$x_1 + x_2 + x_3 + x_4 + (5/4)x_5 + (1/4)x_6 \leq 3$$

Gomory Cuts

- When an integer programming problem has a nonintegral solution, we can generate at least one **Gomory cut** to cut off that solution.

- This is a special case of a **separating cut**, because it separates the current solution of the relaxation from the feasible set.

- Gomory cuts are widely used and very effective in MILP solvers.



Gomory cuts

Given an integer programming problem

$$\min cx$$

$$Ax = b$$

$$x \geq 0 \text{ and integral}$$

Let $(x_B, 0)$ be an optimal solution of the continuous relaxation, where

$$x_B = \hat{b} - \hat{N}x_N$$

$$\hat{b} = B^{-1}b, \quad \hat{N} = B^{-1}N$$

Then if x_i is nonintegral in this solution, the following **Gomory cut** is violated by $(x_B, 0)$:

$$x_i + \lfloor \hat{N}_i \rfloor x_N \leq \lfloor \hat{b}_i \rfloor$$

Example

$$\min 2x_1 + 3x_2$$

$$x_1 + 3x_2 \geq 3$$

$$4x_1 + 3x_2 \geq 6$$

$$x_1, x_2 \geq 0 \text{ and integral}$$

or

$$\min 2x_1 + 3x_2$$

$$x_1 + 3x_2 - x_3 = 3$$

$$4x_1 + 3x_2 - x_4 = 6$$

$$x_j \geq 0 \text{ and integral}$$

Optimal solution of the continuous relaxation has

$$x_B = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2/3 \end{bmatrix}$$

$$\hat{N} = \begin{bmatrix} 1/3 & -1/3 \\ -4/9 & 1/9 \end{bmatrix}$$

$$\hat{b} = \begin{bmatrix} 1 \\ 2/3 \end{bmatrix}$$

Example

$$\min 2x_1 + 3x_2$$

$$x_1 + 3x_2 \geq 3$$

$$4x_1 + 3x_2 \geq 6$$

$$x_1, x_2 \geq 0 \text{ and integral}$$

or

$$\min 2x_1 + 3x_2$$

$$x_1 + 3x_2 - x_3 = 3$$

$$4x_1 + 3x_2 - x_4 = 6$$

$$x_j \geq 0 \text{ and integral}$$

Optimal solution of the continuous relaxation has

$$x_B = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2/3 \end{bmatrix}$$

$$\hat{N} = \begin{bmatrix} 1/3 & -1/3 \\ -4/9 & 1/9 \end{bmatrix}$$

$$\hat{b} = \begin{bmatrix} 1 \\ 2/3 \end{bmatrix}$$

The Gomory cut $x_i + \lfloor \hat{N}_i \rfloor x_N \leq \lfloor \hat{b}_i \rfloor$

is $x_2 + \lfloor [-4/9 \quad 1/9] \rfloor \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} \leq \lfloor 2/3 \rfloor$

or $x_2 - x_3 \leq 0$

In x_1, x_2 space this is $x_1 + 2x_2 \geq 3$

Example

$$\min 2x_1 + 3x_2$$

$$x_1 + 3x_2 \geq 3$$

$$4x_1 + 3x_2 \geq 6$$

$$x_1, x_2 \geq 0 \text{ and integral}$$

or

$$\min 2x_1 + 3x_2$$

$$x_1 + 3x_2 - x_3 = 3$$

$$4x_1 + 3x_2 - x_4 = 6$$

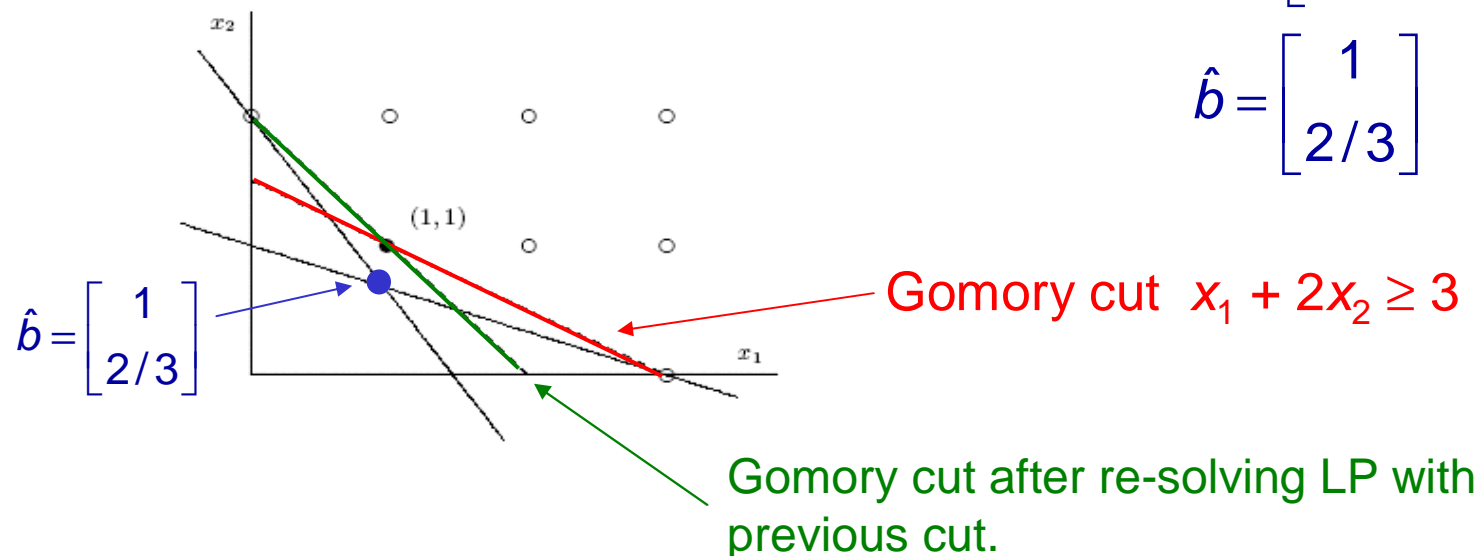
$$x_j \geq 0 \text{ and integral}$$

Optimal solution of the continuous relaxation has

$$x_B = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2/3 \end{bmatrix}$$

$$\hat{N} = \begin{bmatrix} 1/3 & -1/3 \\ -4/9 & 1/9 \end{bmatrix}$$

$$\hat{b} = \begin{bmatrix} 1 \\ 2/3 \end{bmatrix}$$



Mixed Integer Rounding Cuts

- **Mixed integer rounding (MIR) cuts** can be generated for solutions of any relaxed MILP in which one or more integer variables has a fractional value.
 - Like Gomory cuts, they are separating cuts.
 - MIR cuts are widely used in commercial solvers.

MIR cuts

Given an MILP problem

$$\min cx + dy$$

$$Ax + Dy = b$$

$x, y \geq 0$ and y integral

In an optimal solution of the continuous relaxation, let

$$J = \{ j \mid y_j \text{ is nonbasic} \}$$

$$K = \{ j \mid x_j \text{ is nonbasic} \}$$

$$N = \text{nonbasic cols of } [A \ D]$$

Then if y_i is nonintegral in this solution, the following **MIR cut** is violated by the solution of the relaxation:

$$y_i + \sum_{j \in J_1} \lceil \hat{N}_{ij} \rceil y_j + \sum_{j \in J_2} \left(\lfloor \hat{N}_{ij} \rfloor + \frac{\text{frac}(\hat{N}_{ij})}{\text{frac}(\hat{b}_i)} \right) + \frac{1}{\text{frac}(\hat{b}_i)} \sum_{j \in K} \hat{N}_{ij}^+ x_j \geq \hat{N}_{ij} \lceil \hat{b}_i \rceil$$

$$\text{where } J_1 = \{ j \in J \mid \text{frac}(\hat{N}_{ij}) \geq \text{frac}(\hat{b}_i) \} \quad J_2 = J \setminus J_1$$

Example

$$3x_1 + 4x_2 - 6y_1 - 4y_2 = 1$$

$$x_1 + 2x_2 - y_1 - y_2 = 3$$

$$x_j, y_j \geq 0, \quad y_j \text{ integer}$$

Take basic solution $(x_1, y_1) = (8/3, 17/3)$.

$$\text{Then } \hat{N} = \begin{bmatrix} 1/3 & 2/3 \\ -2/3 & 8/3 \end{bmatrix} \quad \hat{b} = \begin{bmatrix} 8/3 \\ 17/3 \end{bmatrix}$$

$$J = \{2\}, \quad K = \{2\}, \quad J_1 = \emptyset, \quad J_2 = \{2\}$$

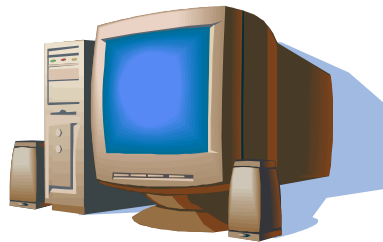
$$\text{The MIR cut is } y_1 + \left(\lfloor 1/3 \rfloor + \frac{1/3}{2/3} \right) y_2 + \frac{1}{2/3} (2/3)^+ x_2 \geq \lceil 8/3 \rceil$$

$$\text{or } y_1 + (1/2)y_2 + x_2 \geq 3$$

Example: Product Configuration

This example illustrates:

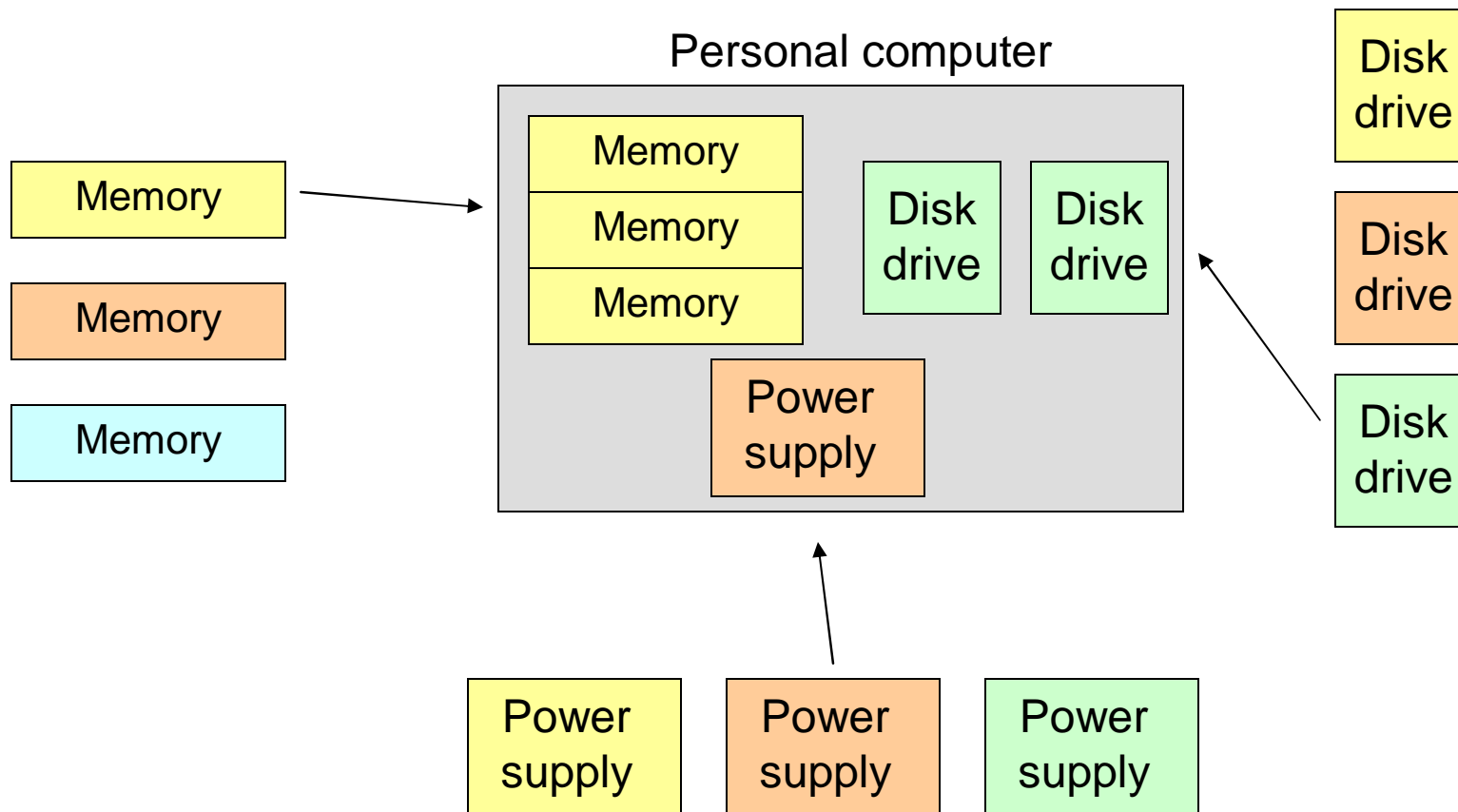
- Combination of propagation and relaxation.
- Processing of variable indices.
- Continuous relaxation of element constraint.



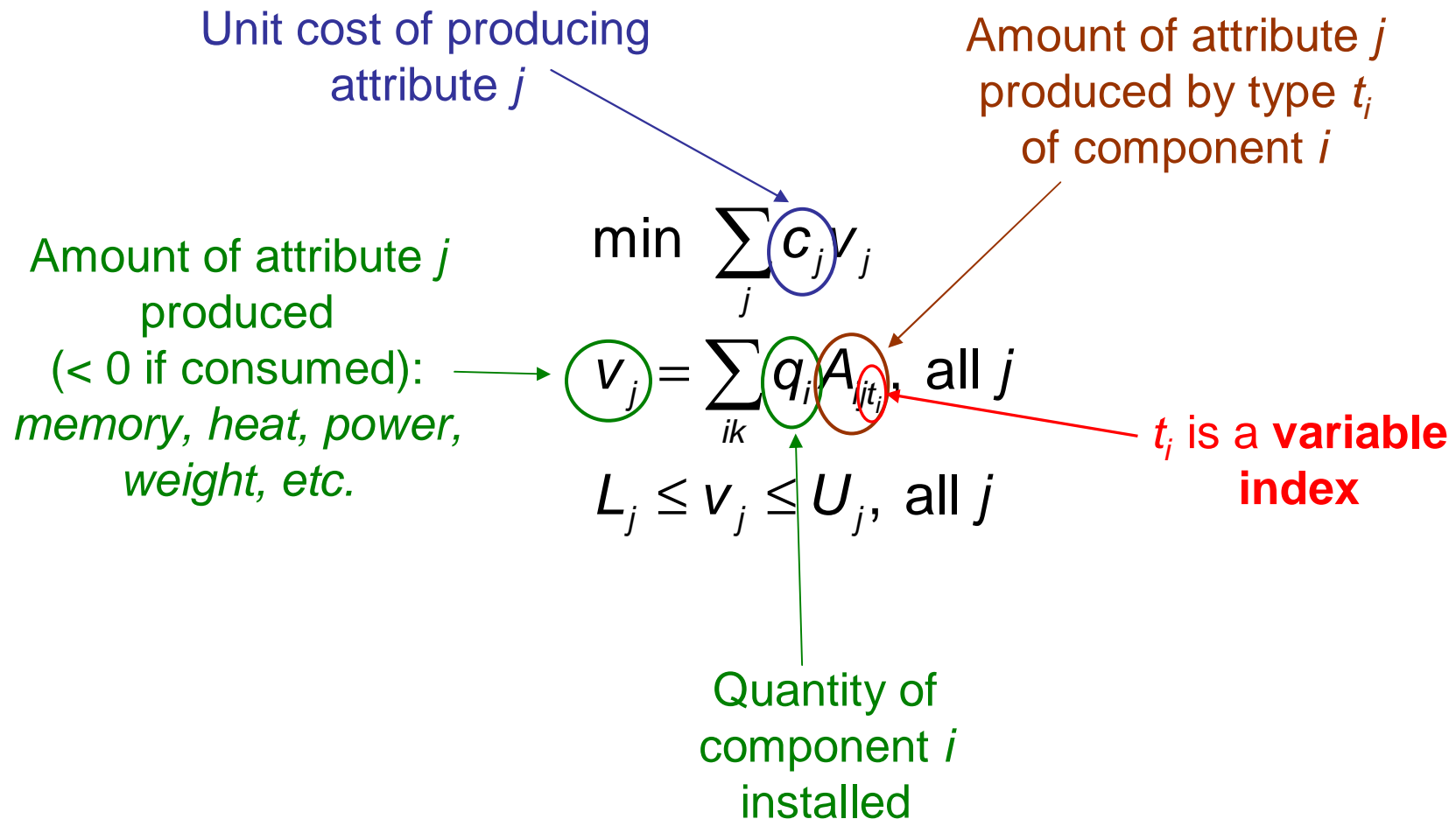
The problem



Choose what type of each component, and how many



Model of the problem

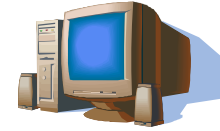


To solve it:



- **Branch** on domains of t_i and q_i .
- **Propagate** *element* constraints and bounds on v_j .
 - **Variable index** is converted to specially structured *element* constraint.
 - Valid **knapsack** cuts are derived and propagated.
- Use linear continuous **relaxations**.
 - Special purpose **MILP** relaxation for *element*.

Propagation



$$\min \sum_j c_j v_j$$

$$v_j = \sum_{ik} q_i A_{ij t_i}, \text{ all } j$$

$$L_j \leq v_j \leq U_j, \text{ all } j$$

← *This is propagated
in the usual way*

Propagation



$$v_j = \sum_i z_i, \text{ all } j$$

$$\text{element}(t_i, (q_i, A_{ij1}, \dots, q_i A_{ijn}), z_i), \text{ all } i, j$$

$$\min \sum_j c_j v_j$$

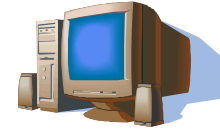
$$v_j = \sum_{ik} q_i A_{ij t_i}, \text{ all } j$$

This is rewritten as

$$L_j \leq v_j \leq U_j, \text{ all } j$$

*This is propagated
in the usual way*

Propagation



$$v_j = \sum_i z_i, \text{ all } j$$

$$\text{element}(t_i, (q_i, A_{ij1}, \dots, q_i A_{ijn}), z_i), \text{ all } i, j$$

This can be propagated by
(a) using specialized **filters** for *element* constraints of this form...

Propagation



$$v_j = \sum_i z_i, \text{ all } j$$

$$\text{element}(t_i, (q_i, A_{ij1}, \dots, q_i A_{ijn}), z_i), \text{ all } i, j$$

This is propagated by

- (a) using specialized **filters** for *element* constraints of this form,
- (b) adding **knapsack cuts** for the valid inequalities:

$$\sum_i \max_{k \in D_{t_i}} \{A_{ijk}\} q_i \geq \underline{v}_j, \text{ all } j$$

$$\sum_i \min_{k \in D_{t_i}} \{A_{ijk}\} q_i \leq \bar{v}_j, \text{ all } j$$

and (c) propagating the knapsack cuts.

$[\underline{v}_j, \bar{v}_j]$ is current
domain of v_j

Relaxation



$$\min \sum_j c_j v_j$$

$$v_j = \sum_{ik} q_i A_{ij t_i}, \text{ all } j$$

$$L_j \leq v_j \leq U_j, \text{ all } j$$

This is relaxed as

$$\underline{v}_j \leq v_j \leq \bar{v}_j$$

Relaxation



$$v_j = \sum_i z_i, \text{ all } j$$

$$\text{element}(t_i, (q_i, A_{ij1}, \dots, q_i A_{ijn}), z_i), \text{ all } i, j$$

$$\min \sum_j c_j v_j$$

$$v_j = \sum_{ik} q_i A_{ij t_i}, \text{ all } j$$

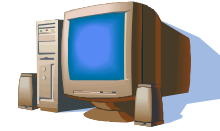
$$L_j \leq v_j \leq U_j, \text{ all } j$$

This is relaxed by relaxing this and adding the knapsack cuts.

This is relaxed as

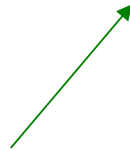
$$\underline{v}_j \leq v_j \leq \bar{v}_j$$

Relaxation



$$v_j = \sum_i z_i, \text{ all } j$$

$$\text{element}(t_i, (q_i, A_{ij1}, \dots, q_i A_{ijn}), z_i), \text{ all } i, j$$



*This is relaxed by replacing each *element* constraint with a disjunctive **convex hull** relaxation:*

$$z_i = \sum_{k \in D_{t_j}} A_{ijk} q_{ik}, \quad q_i = \sum_{k \in D_{t_j}} q_{ik}$$

Relaxation



So the following LP relaxation is solved at each node of the search tree to obtain a lower bound:

$$\min \sum_j c_j v_j$$

$$v_j = \sum_i \sum_{k \in D_{t_j}} A_{ijk} q_{ik}, \text{ all } j$$

$$q_j = \sum_{k \in D_{t_j}} q_{ik}, \text{ all } i$$

$$\underline{v}_j \leq v_j \leq \bar{v}_j, \text{ all } j$$

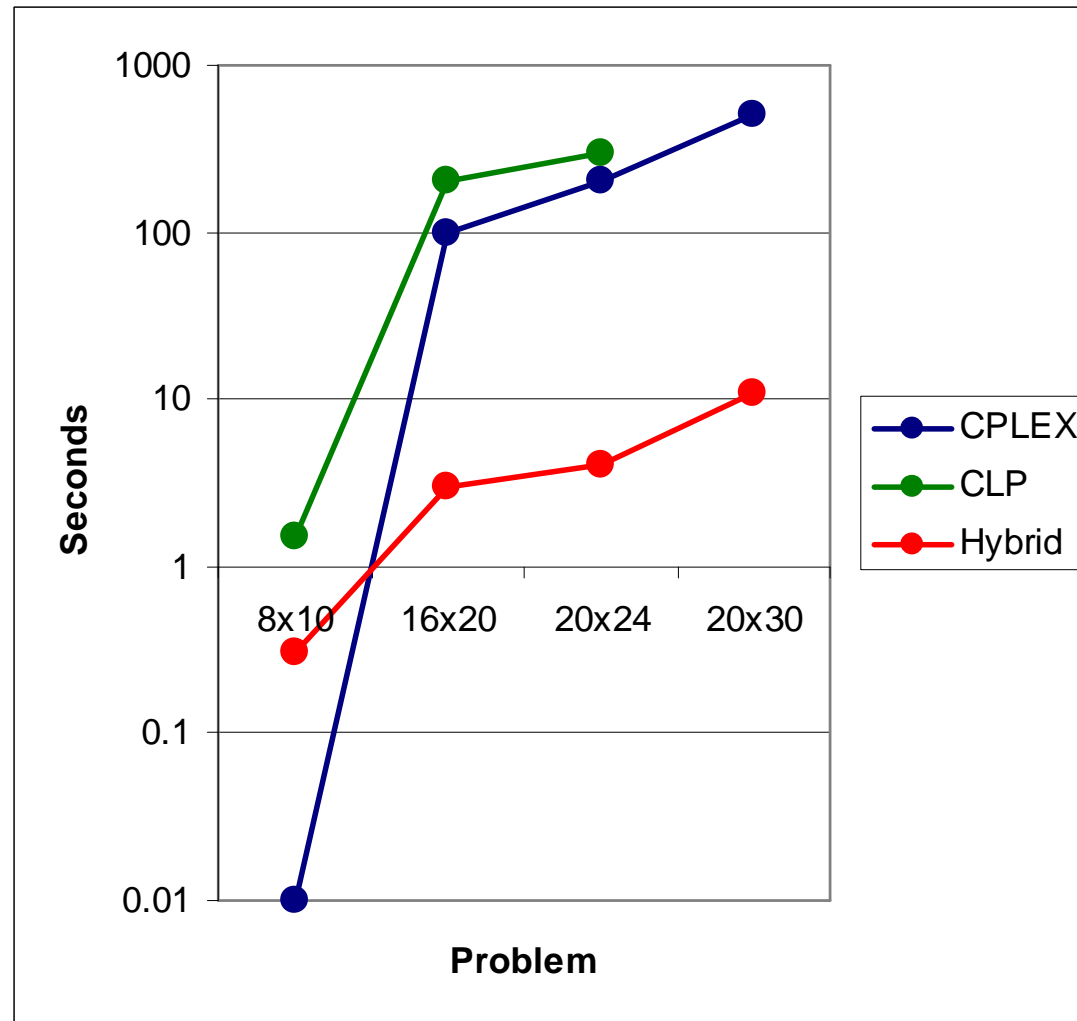
$$\underline{q}_i \leq q_i \leq \bar{q}_i, \text{ all } i$$

$$\text{knapsack cuts for } \sum_i \max_{k \in D_{t_j}} \{A_{ijk}\} q_i \geq \underline{v}_j, \text{ all } j$$

$$\text{knapsack cuts for } \sum_i \min_{k \in D_{t_j}} \{A_{ijk}\} q_i \leq \bar{v}_j, \text{ all } j$$

$$q_{ik} \geq 0, \text{ all } i, k$$

Computational Results





Lagrangian Relaxation

Lagrangian Duality

Properties of the Lagrangian Dual

Example: Fast Linear Programming

Domain Filtering

Example: Continuous Global Optimization

Motivation

- **Lagrangian relaxation** can provide better bounds than LP relaxation.
- The **Lagrangian dual** generalizes LP duality.
- It provides **domain filtering** analogous to that based on LP duality.
 - This is a key technique in **continuous global optimization**.
- Lagrangian relaxation gets rid of troublesome constraints by **dualizing** them.
 - That is, moving them into the objective function.
 - The Lagrangian relaxation may **decouple**.

Lagrangean Duality

Consider an
inequality-constrained
problem

$$\min f(x)$$

$$g(x) \geq 0$$

$$x \in S$$

Hard constraints

Easy constraints

The object is to get rid of (**dualize**) the hard constraints by moving them into the objective function.

Lagrangean Duality

Consider an
inequality-constrained
problem

$$\min f(x)$$

$$g(x) \geq 0$$

$$x \in S$$

It is related to an
inference problem

$$\max v$$

$$g(x) \geq b \overset{s \in S}{\Rightarrow} f(x) \geq v$$

implies

Lagrangian Dual problem: Find the tightest lower bound
on the objective function that is implied by the constraints.

Primal

$$\min f(x)$$

$$g(x) \geq 0$$

$$x \in S$$

Dual

$$\max v$$

$$g(x) \geq 0 \stackrel{x \in S}{\Rightarrow} f(x) \geq v$$

Surrogate

Let us say that

$$g(x) \geq 0 \stackrel{x \in S}{\Rightarrow} f(x) \geq v \quad \text{iff} \quad \boxed{\lambda g(x) \geq 0} \quad \boxed{\text{dominates}} \quad f(x) - v \geq 0$$

for some $\lambda \geq 0$

$$\lambda g(x) \leq f(x) - v \quad \text{for all } x \in S$$

$$\text{That is, } v \leq f(x) - \lambda g(x) \quad \text{for all } x \in S$$

Primal

$$\min f(x)$$

$$g(x) \geq 0$$

$$x \in S$$

Dual

$$\max v$$

$$g(x) \geq 0 \stackrel{x \in S}{\Rightarrow} f(x) \geq v$$

Surrogate

Let us say that

$$g(x) \geq 0 \stackrel{x \in S}{\Rightarrow} f(x) \geq v \quad \text{iff} \quad \boxed{\lambda g(x) \geq 0} \quad \boxed{\text{dominates}} \quad f(x) - v \geq 0$$

for some $\lambda \geq 0$

$$\lambda g(x) \leq f(x) - v \quad \text{for all } x \in S$$

$$\text{That is, } v \leq f(x) - \lambda g(x) \quad \text{for all } x \in S$$

$$\text{Or } v \leq \min_{x \in S} \{f(x) - \lambda g(x)\}$$

Primal

$$\min f(x)$$

$$g(x) \geq 0$$

$$x \in S$$

Dual

$$\max v$$

$$g(x) \geq b \stackrel{s \in S}{\Rightarrow} f(x) \geq v$$

Surrogate

Let us say that

$$g(x) \geq 0 \stackrel{x \in S}{\Rightarrow} f(x) \geq v \quad \text{iff} \quad \boxed{\lambda g(x) \geq 0} \quad \boxed{\text{dominates}} \quad f(x) - v \geq 0$$

for some $\lambda \geq 0$

$$\lambda g(x) \leq f(x) - v \quad \text{for all } x \in S$$

$$\text{That is, } v \leq f(x) - \lambda g(x) \quad \text{for all } x \in S$$

$$\text{Or } v \leq \min_{x \in S} \{f(x) - \lambda g(x)\}$$

So the dual becomes

$$\max v$$

$$v \leq \min_{x \in S} \{f(x) - \lambda g(x)\} \quad \text{for some } \lambda \geq 0$$

Now we have...

Primal

$$\min f(x)$$

$$g(x) \geq 0$$

$$x \in S$$

These constraints
are **dualized**

Dual

$$\max v$$

$$v \leq \min_{x \in S} \{f(x) - \lambda g(x)\} \text{ for some } \lambda \geq 0$$

or

$$\max_{\lambda \geq 0} \theta(\lambda)$$

where

$$\theta(\lambda) = \min_{x \in S} \{f(x) - \lambda g(x)\}$$

Lagrangean
relaxation

Vector of
Lagrange
multipliers

The Lagrangean dual can be viewed as the problem of finding the Lagrangean relaxation that gives the tightest bound.

Example

$$\min 3x_1 + 4x_2$$

$$-x_1 + 3x_2 \geq 0$$

$$2x_1 + x_2 - 5 \geq 0$$

$$x_1, x_2 \in \{0, 1, 2, 3\}$$

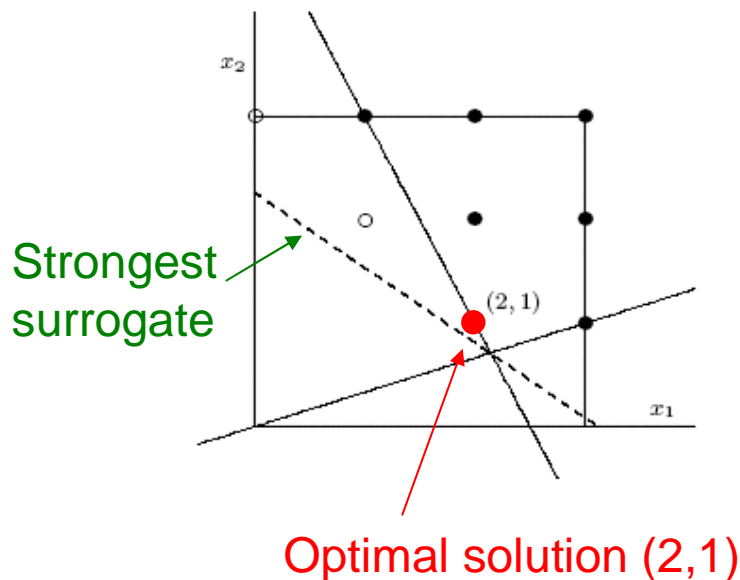
The Lagrangean relaxation is

$$\begin{aligned}\theta(\lambda_1, \lambda_2) &= \min_{x_j \in \{0, \dots, 3\}} \{3x_1 + 4x_2 - \lambda_1(-x_1 + 3x_2) - \lambda_2(2x_1 + x_2 - 5)\} \\ &= \min_{x_j \in \{0, \dots, 3\}} \{(3 + \lambda_1 - 2\lambda_2)x_1 + (4 - 3\lambda_1 - \lambda_2)x_2 + 5\lambda_2\}\end{aligned}$$

The Lagrangean relaxation is easy to solve for any given λ_1, λ_2 :

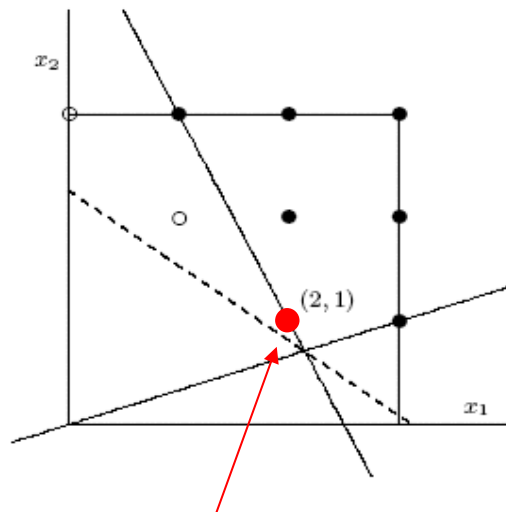
$$x_1 = \begin{cases} 0 & \text{if } 3 + \lambda_1 - 2\lambda_2 \geq 0 \\ 3 & \text{otherwise} \end{cases}$$

$$x_2 = \begin{cases} 0 & \text{if } 4 - 3\lambda_1 - \lambda_2 \geq 0 \\ 3 & \text{otherwise} \end{cases}$$



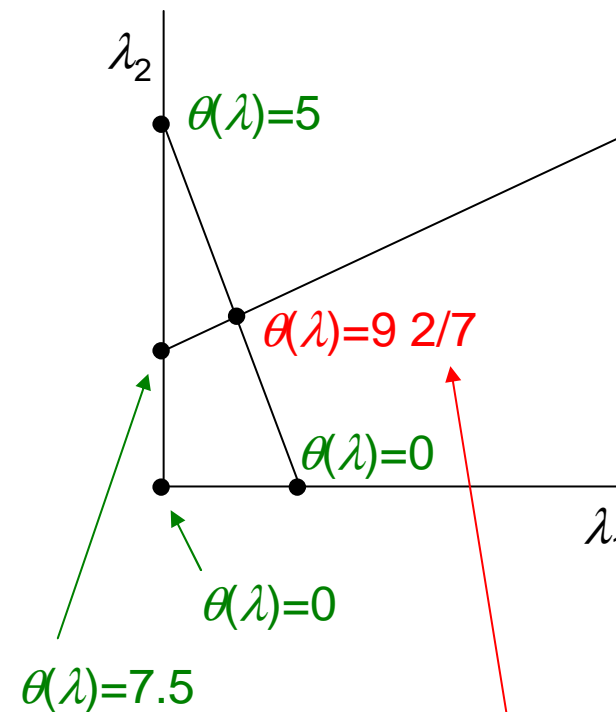
Example

$$\begin{aligned} \min \quad & 3x_1 + 4x_2 \\ -x_1 + 3x_2 \geq & 0 \\ 2x_1 + x_2 - 5 \geq & 0 \\ x_1, x_2 \in \{0, 1, 2, 3\} \end{aligned}$$



Optimal solution (2,1)
Value = 10

$\theta(\lambda_1, \lambda_2)$ is piecewise linear and concave.



Solution of Lagrangean dual:

$$(\lambda_1, \lambda_2) = (5/7, 13/7), \quad \theta(\lambda) = 9 \frac{2}{7}$$

Note **duality gap** between 10 and $9 \frac{2}{7}$
(no strong duality).

Example

$$\begin{aligned} \min \quad & 3x_1 + 4x_2 \\ & -x_1 + 3x_2 \geq 0 \\ & 2x_1 + x_2 - 5 \geq 0 \\ & x_1, x_2 \in \{0, 1, 2, 3\} \end{aligned}$$

Note: in this example, the Lagrangean dual provides the same bound (9 2/7) as the continuous relaxation of the IP.

This is because the Lagrangean relaxation can be solved as an LP:

$$\begin{aligned} \theta(\lambda_1, \lambda_2) &= \min_{x_j \in \{0, \dots, 3\}} \{(3 + \lambda_1 - 2\lambda_2)x_1 + (4 - 3\lambda_1 - \lambda_2)x_2 + 5\lambda_2\} \\ &= \min_{0 \leq x_j \leq 3} \{(3 + \lambda_1 - 2\lambda_2)x_1 + (4 - 3\lambda_1 - \lambda_2)x_2 + 5\lambda_2\} \end{aligned}$$

Lagrangean duality is useful when the Lagrangean relaxation is tighter than an LP but nonetheless easy to solve.

Properties of the Lagrangean dual

Weak duality: For any feasible x^* and any $\lambda^* \geq 0$, $f(x^*) \geq \theta(\lambda^*)$.

In particular, $\min_{\substack{x \in S \\ g(x) \geq 0}} f(x) \geq \max_{\lambda \geq 0} \theta(\lambda)$

Concavity: $\theta(\lambda)$ is concave. It can therefore be maximized by local search methods.

Complementary slackness: If x^* and λ^* are optimal, and there is no duality gap, then $\lambda^* g(x^*) = 0$.

Solving the Lagrangean dual

Let λ^k be the k th iterate, and let $\lambda^{k+1} = \lambda^k + \alpha_k \xi^k$

Subgradient of $\theta(\lambda)$ at $\lambda = \lambda^k$

If x^k solves the Lagrangean relaxation for $\lambda = \lambda^k$, then $\xi^k = g(x^k)$.

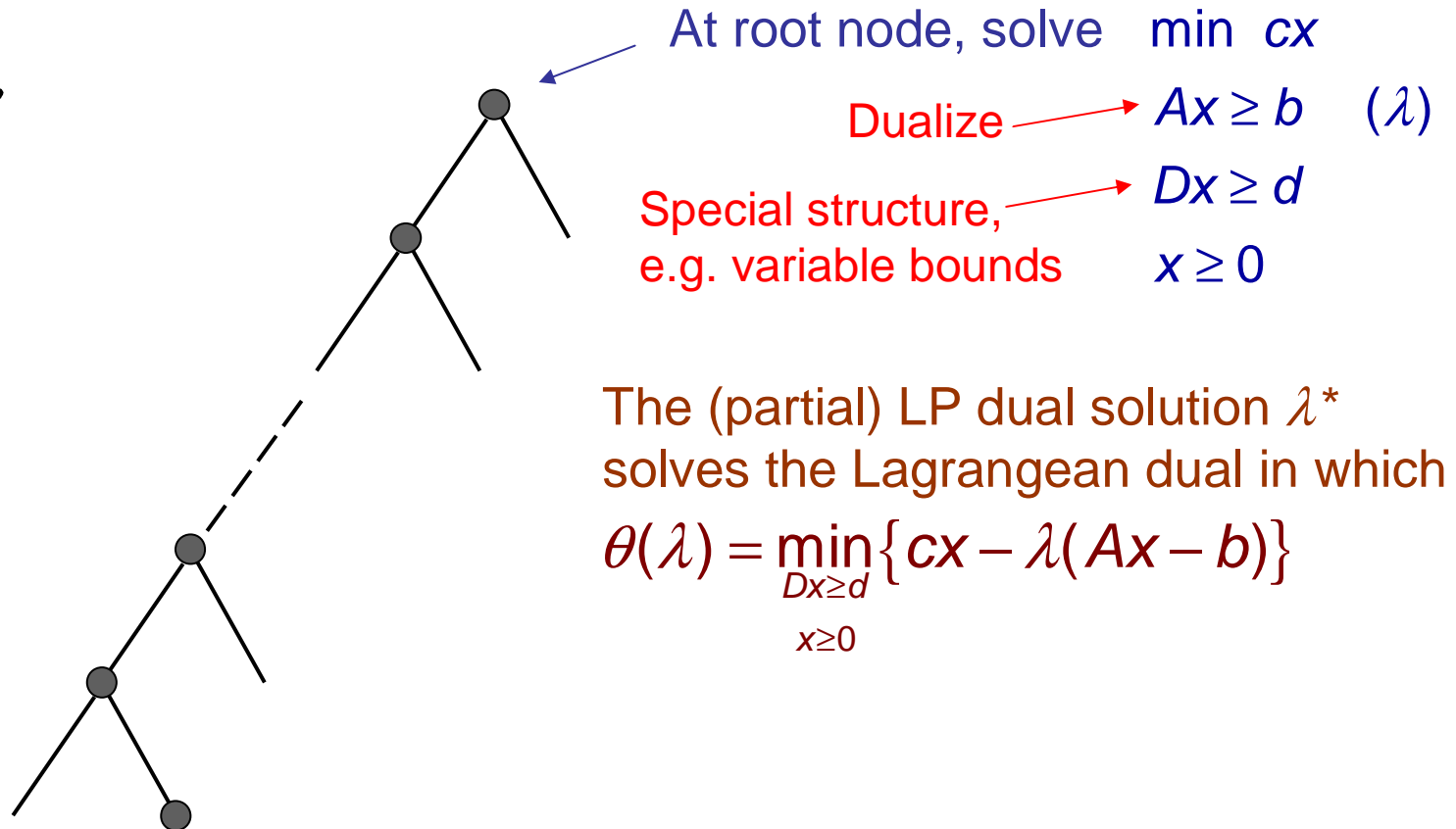
This is because $\theta(\lambda) = f(x^k) + \lambda g(x^k)$ at $\lambda = \lambda^k$.

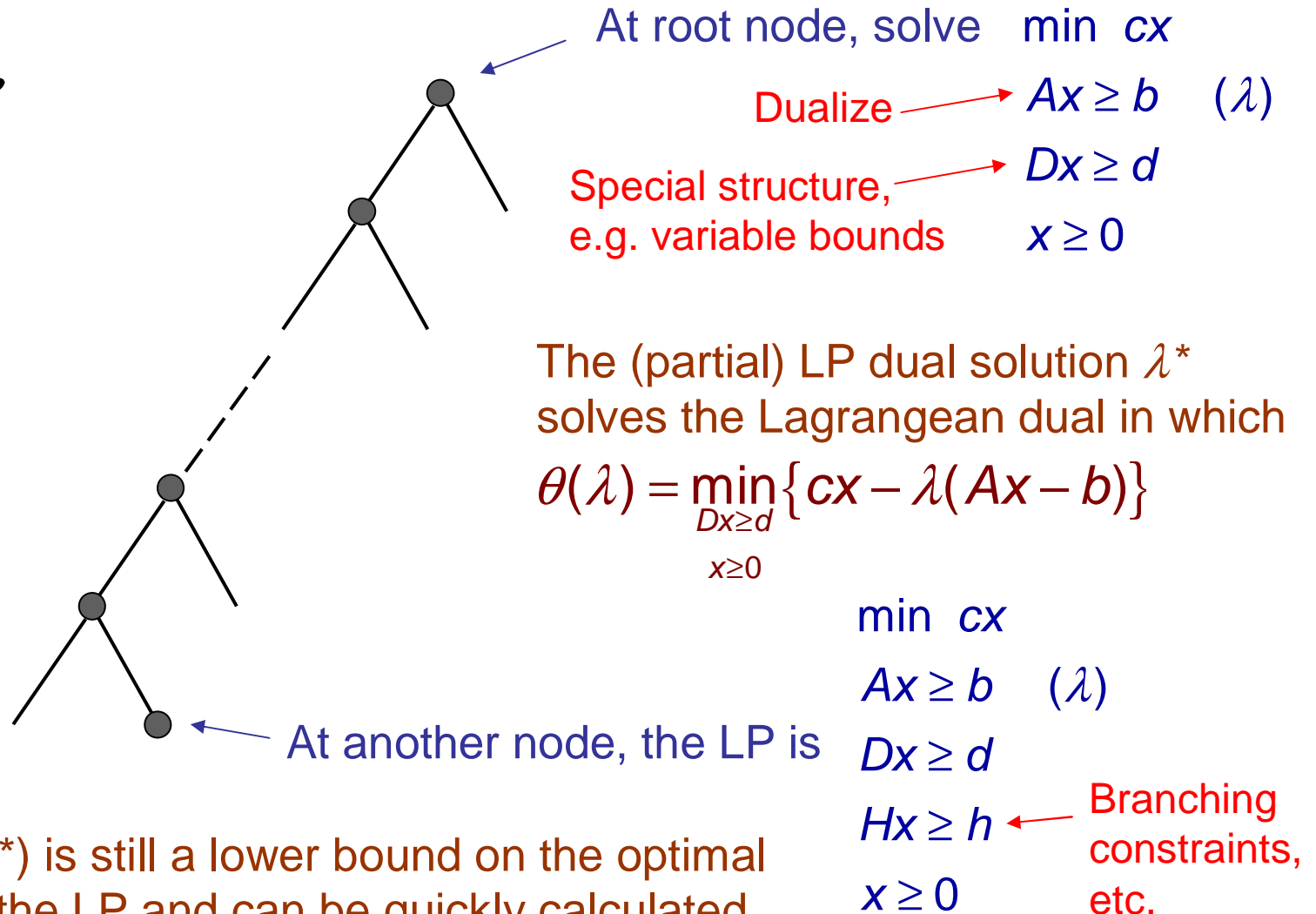
The stepsize α_k must be adjusted so that the sequence converges but not before reaching a maximum.

Example: Fast Linear Programming

- In CP contexts, it is best to process each node of the search tree very rapidly.
- Lagrangean relaxation may allow very fast calculation of a lower bound on the optimal value of the LP relaxation at each node.
- The idea is to 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.







Here $\theta(\lambda^*)$ is still a lower bound on the optimal value of the LP and can be quickly calculated by solving a specially structured LP.

Domain Filtering

Suppose:

$\min_{x \in S} f(x)$
 $g(x) \geq 0$ has optimal solution x^* , optimal value v^* , and
optimal Lagrangean dual solution λ^* .

...and $\lambda_i^* > 0$, which means the i -th constraint is tight
(complementary slackness);

...and the problem is a relaxation of a CP problem;

...and we have a feasible solution of the CP problem with value
 U , so that U is an upper bound on the optimal value.

Supposing $\min_{x \in S} f(x)$ has optimal solution x^* , optimal value v^* , and optimal Lagrangean dual solution λ^* :

If x were to change to a value other than x^* , the LHS of i -th constraint $g_i(x) \geq 0$ would change by some amount Δ_i .

Since the constraint is tight, this would increase the optimal value as much as changing the constraint to $g_i(x) - \Delta_i \geq 0$.

So it would increase the optimal value at least $\lambda_i^* \Delta_i$.

(It is easily shown that Lagrange multipliers are marginal costs. Dual multipliers for LP are a special case of Lagrange multipliers.)

Supposing $\min_{x \in S} f(x)$ has optimal solution x^* , optimal value v^* , and optimal Lagrangean dual solution λ^* :

We have found: a change in x that changes $g_i(x)$ by Δ_i increases the optimal value at least $\lambda_i^* \Delta_i$.

Since optimal value of this problem \leq optimal value of the CP $\leq U$, we have $\lambda_i^* \Delta_i \leq U - v^*$, or

$$\Delta_i \leq \frac{U - v^*}{\lambda_i^*}$$

Supposing $\min_{x \in S} f(x)$ has optimal solution x^* , optimal value v^* , and optimal Lagrangean dual solution λ^* :

We have found: a change in x that changes $g_i(x)$ by Δ_i increases the optimal value at least $\lambda_i^* \Delta_i$.

Since optimal value of this problem \leq optimal value of the CP $\leq U$, we have $\lambda_i^* \Delta_i \leq U - v^*$, or

$$\Delta_i \leq \frac{U - v^*}{\lambda_i^*}$$

Since $\Delta_i = g_i(x) - g_i(x^*) = g_i(x)$, this implies the inequality

$$g_i(x) \leq \frac{U - v^*}{\lambda_i^*}$$

...which can be propagated.

Example: Continuous Global Optimization

- Some of the best continuous global solvers (e.g., BARON) combine OR-style relaxation with CP-style interval arithmetic and domain filtering.
- These methods can be combined with domain filtering based on Lagrange multipliers.



Continuous Global Optimization

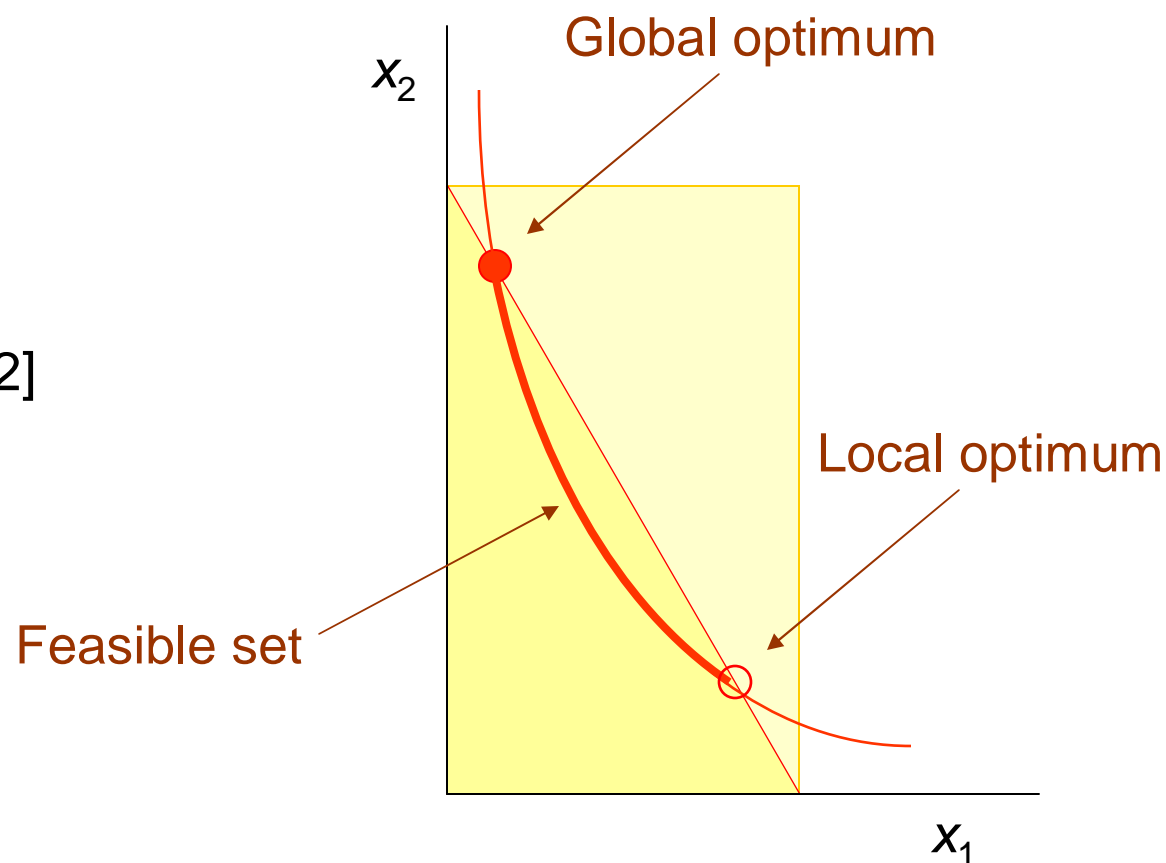


$$\max x_1 + x_2$$

$$4x_1x_2 = 1$$

$$2x_1 + x_2 \leq 2$$

$$x_1 \in [0, 1], \quad x_2 \in [0, 2]$$





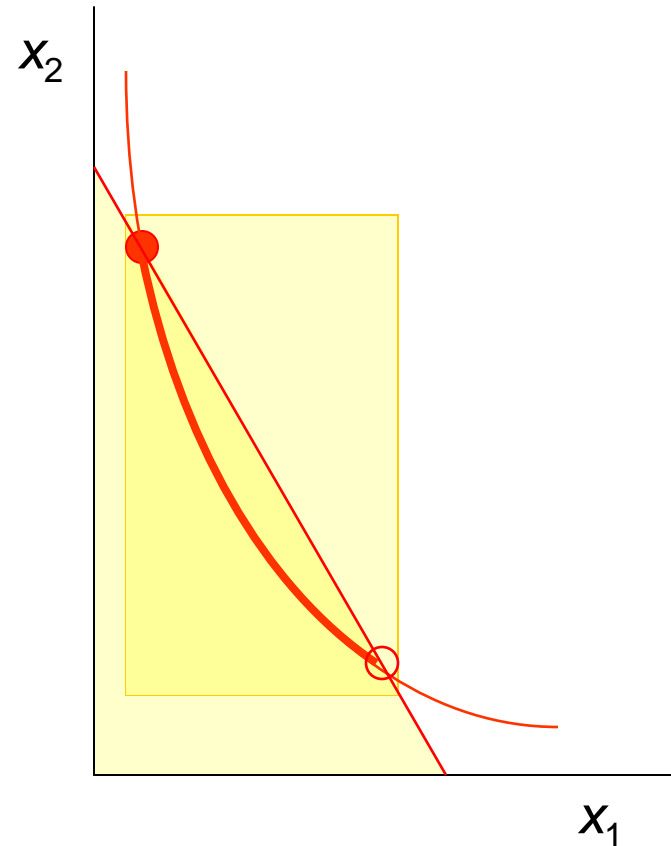
To solve it:

- **Search:** split interval domains of x_1, x_2 .
 - Each **node** of search tree is a problem restriction.
- **Propagation:** Interval propagation, domain filtering.
 - Use **Lagrange multipliers** to infer valid inequality for propagation.
 - **Reduced-cost variable** fixing is a special case.
- **Relaxation:** Use function **factorization** to obtain linear continuous relaxation.

Interval propagation



Propagate intervals
 $[0,1]$, $[0,2]$
through constraints
to obtain
 $[1/8, 7/8]$, $[1/4, 7/4]$



Relaxation (function factorization)



Factor complex functions into elementary functions that have known linear relaxations.

Write $4x_1x_2 = 1$ as $4y = 1$ where $y = x_1x_2$.

This factors $4x_1x_2$ into linear function $4y$ and bilinear function x_1x_2 .

Linear function $4y$ is its own linear relaxation.

Relaxation (function factorization)



Factor complex functions into elementary functions that have known linear relaxations.

Write $4x_1x_2 = 1$ as $4y = 1$ where $y = x_1x_2$.

This factors $4x_1x_2$ into linear function $4y$ and bilinear function x_1x_2 .

Linear function $4y$ is its own linear relaxation.

Bilinear function $y = x_1x_2$ has relaxation:

$$\underline{x}_2\overline{x}_1 + \underline{x}_1\overline{x}_2 - \underline{x}_1\underline{x}_2 \leq y \leq \underline{x}_2\overline{x}_1 + \overline{x}_1\overline{x}_2 - \overline{x}_1\underline{x}_2$$

$$\overline{x}_2\overline{x}_1 + \overline{x}_1\overline{x}_2 - \overline{x}_1\overline{x}_2 \leq y \leq \overline{x}_2\overline{x}_1 + \underline{x}_1\overline{x}_2 - \underline{x}_1\overline{x}_2$$

where domain of x_j is $[\underline{x}_j, \overline{x}_j]$

Relaxation (function factorization)



The linear relaxation becomes:

$$\min x_1 + x_2$$

$$4y = 1$$

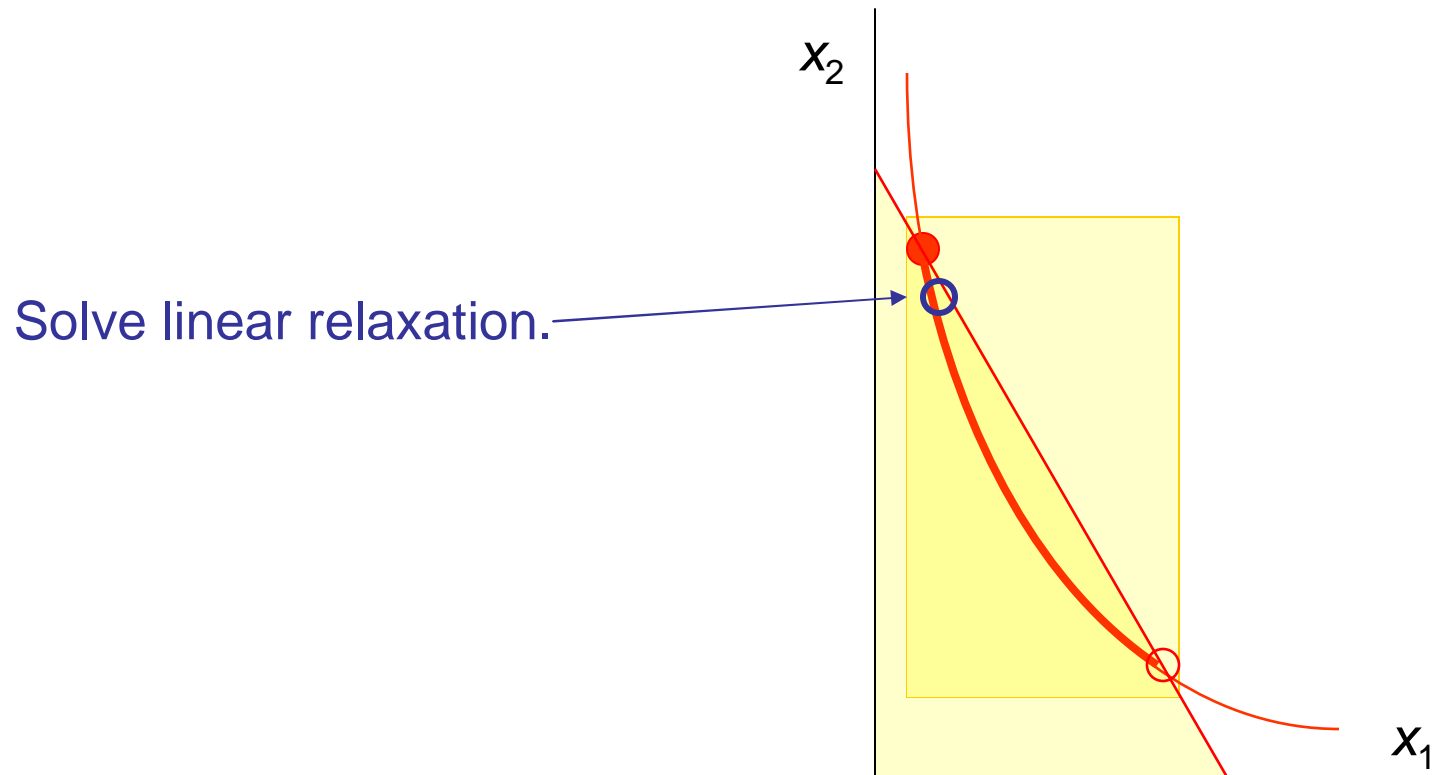
$$2x_1 + x_2 \leq 2$$

$$\underline{x}_2 \underline{x}_1 + \underline{x}_1 \underline{x}_2 - \underline{x}_1 \underline{x}_2 \leq y \leq \underline{x}_2 \underline{x}_1 + \bar{x}_1 \underline{x}_2 - \bar{x}_1 \underline{x}_2$$

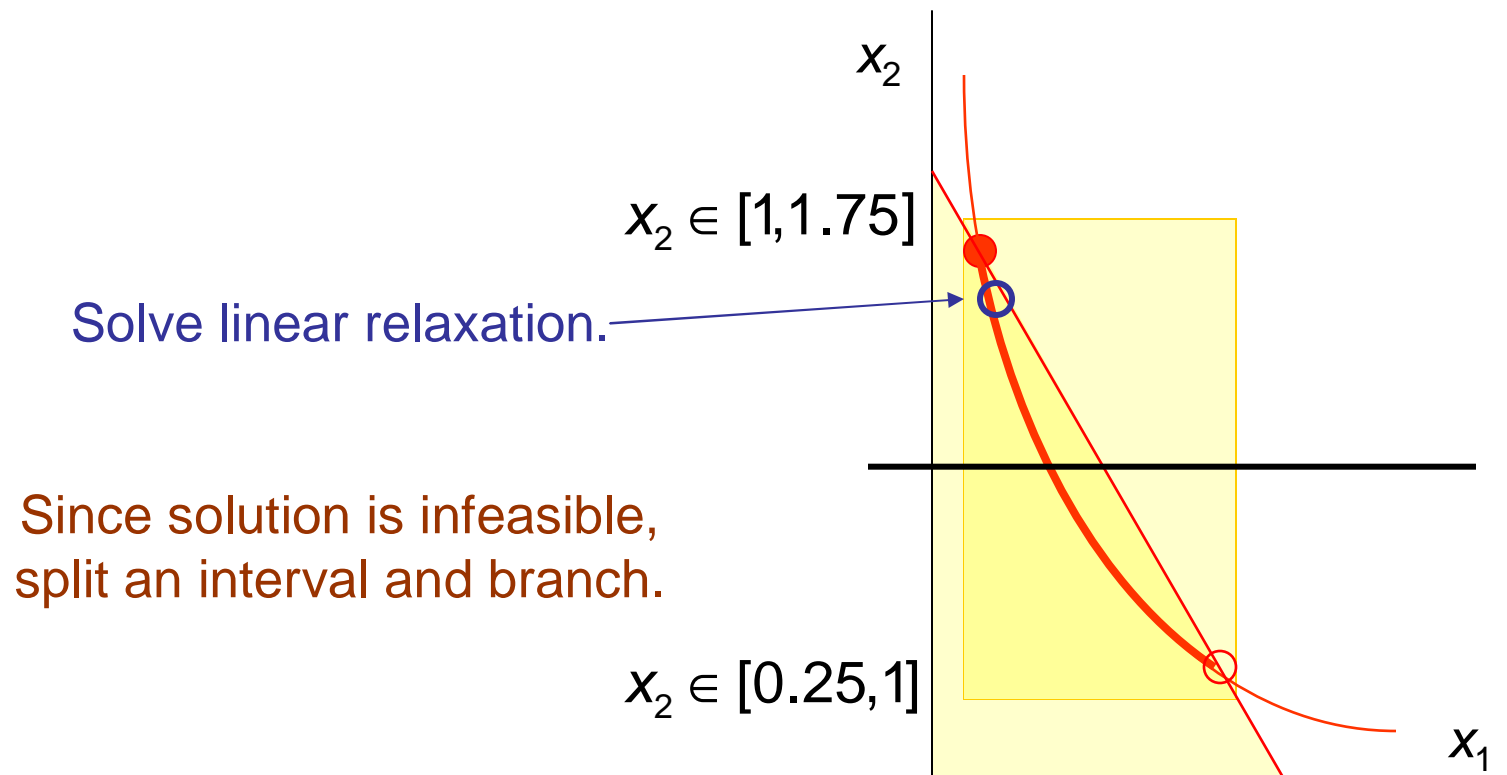
$$\bar{x}_2 \underline{x}_1 + \bar{x}_1 \underline{x}_2 - \bar{x}_1 \bar{x}_2 \leq y \leq \bar{x}_2 \underline{x}_1 + \underline{x}_1 \underline{x}_2 - \underline{x}_1 \bar{x}_2$$

$$\underline{x}_j \leq x_j \leq \bar{x}_j, \quad j = 1, 2$$

Relaxation (function factorization)

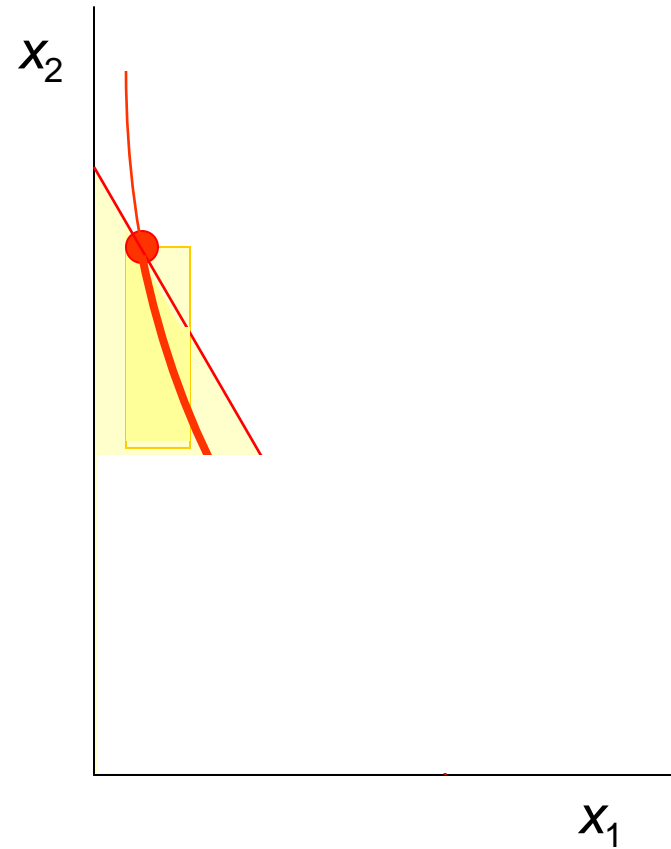
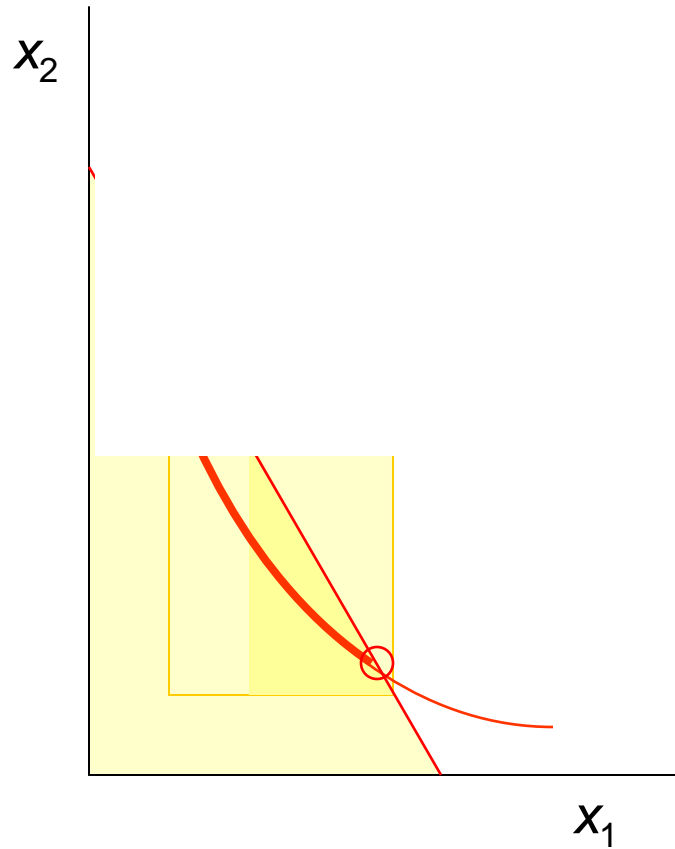


Relaxation (function factorization)



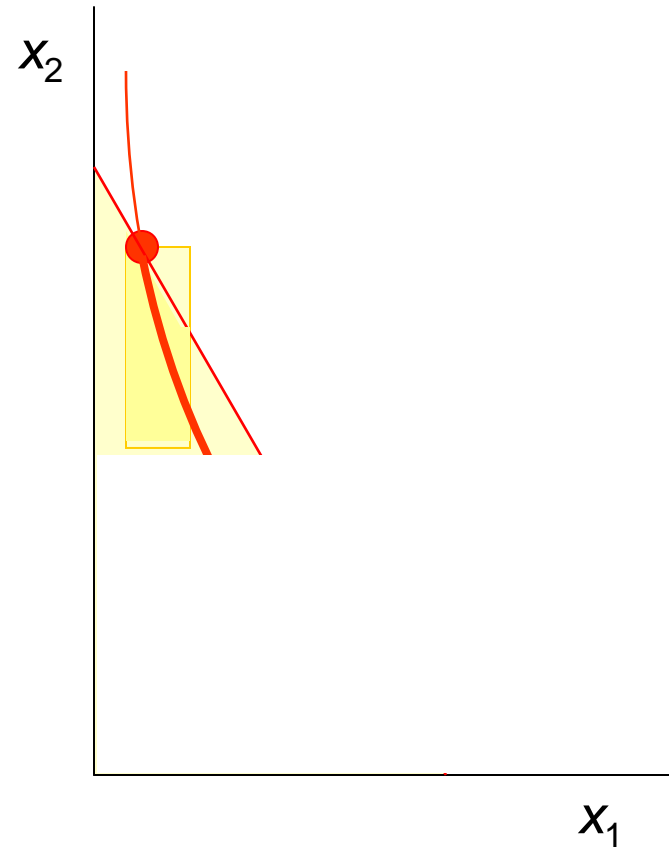
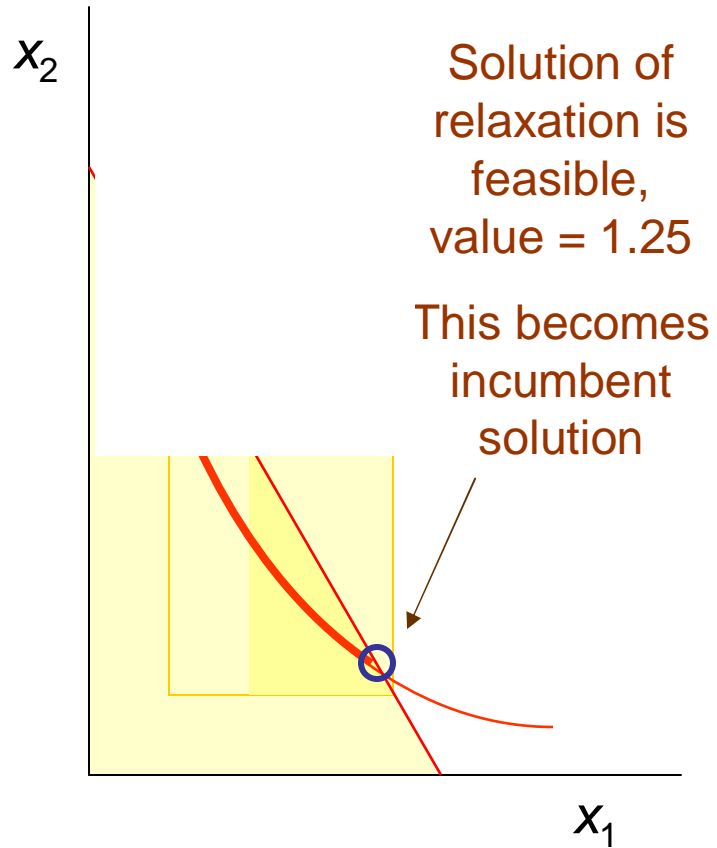
$$x_2 \in [1, 1.75]$$

$$x_2 \in [0.25, 1]$$



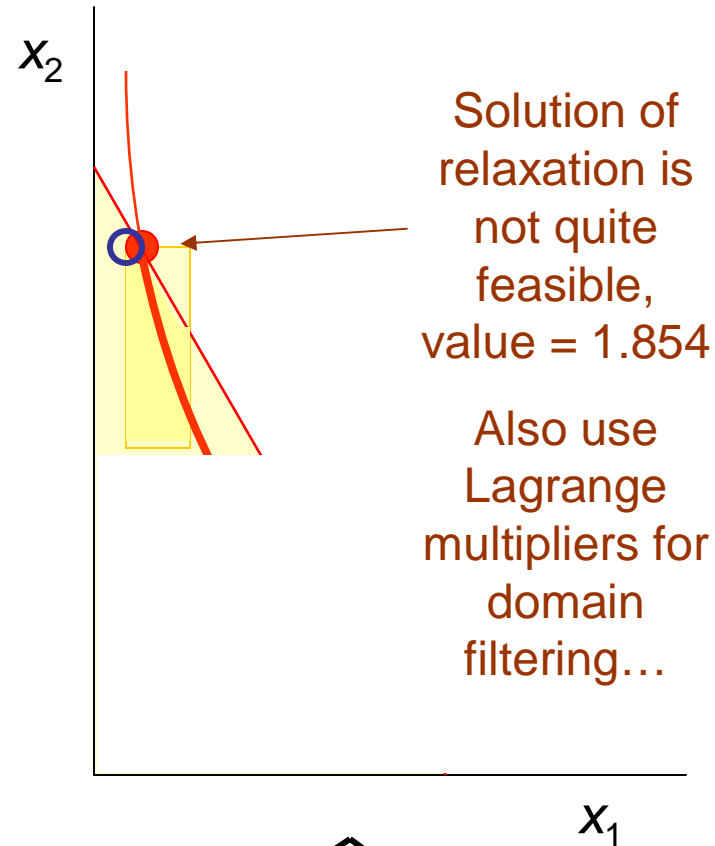
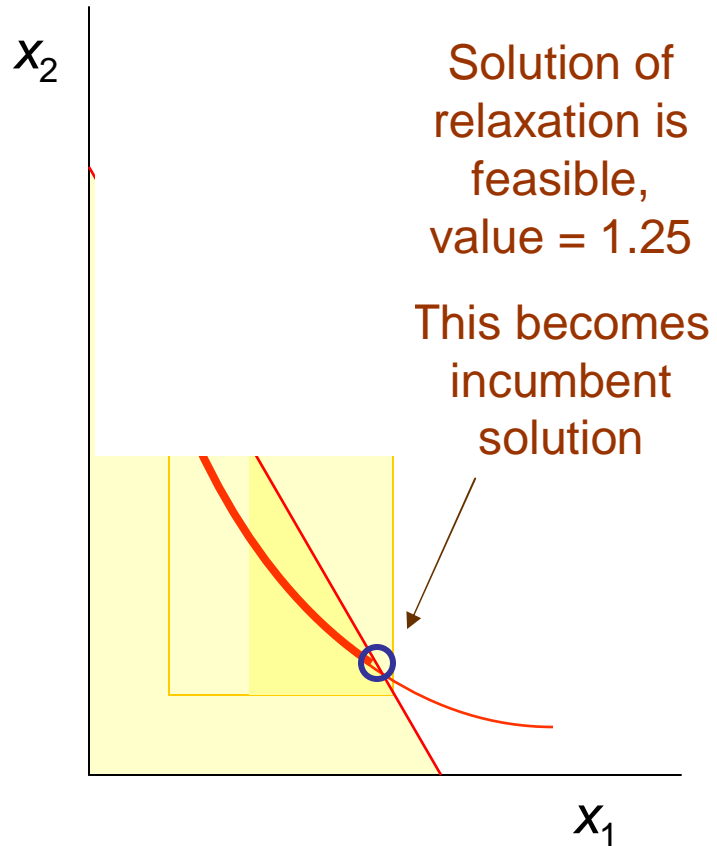
$$x_2 \in [1, 1.75]$$

$$x_2 \in [0.25, 1]$$



$$x_2 \in [1, 1.75]$$

$$x_2 \in [0.25, 1]$$



Relaxation (function factorization)



$$\min x_1 + x_2$$

$$4y = 1$$

$$2x_1 + x_2 \leq 2$$

Associated Lagrange multiplier in solution of relaxation is $\lambda_2 = 1.1$

$$\underline{x}_2 x_1 + \underline{x}_1 x_2 - \underline{x}_1 \underline{x}_2 \leq y \leq \underline{x}_2 x_1 + \bar{x}_1 x_2 - \bar{x}_1 \underline{x}_2$$

$$\bar{x}_2 x_1 + \bar{x}_1 x_2 - \bar{x}_1 \bar{x}_2 \leq y \leq \bar{x}_2 x_1 + \underline{x}_1 x_2 - \underline{x}_1 \bar{x}_2$$

$$\underline{x}_j \leq x_j \leq \bar{x}_j, \quad j = 1, 2$$

Relaxation (function factorization)



$$\min x_1 + x_2$$

$$4y = 1$$

$$2x_1 + x_2 \leq 2$$

Associated Lagrange multiplier in solution of relaxation is $\lambda_2 = 1.1$

$$\underline{x}_2 \underline{x}_1 + \underline{x}_1 \underline{x}_2 - \underline{x}_1 \underline{x}_2 \leq y \leq \underline{x}_2 \underline{x}_1 + \bar{x}_1 \underline{x}_2 - \bar{x}_1 \underline{x}_2$$

$$\bar{x}_2 \underline{x}_1 + \bar{x}_1 \underline{x}_2 - \bar{x}_1 \bar{x}_2 \leq y \leq \bar{x}_2 \underline{x}_1 + \underline{x}_1 \underline{x}_2 - \underline{x}_1 \bar{x}_2$$

$$\underline{x}_j \leq x_j \leq \bar{x}_j, \quad j = 1, 2$$

This yields a valid inequality for propagation:

$$2x_1 + x_2 \geq 2 - \frac{1.854 - 1.25}{1.1} = 1.451$$

Value of relaxation

Lagrange multiplier

Value of incumbent solution



Dynamic Programming in CP

Example: Capital Budgeting
Domain Filtering
Recursive Optimization

Motivation

- **Dynamic programming** (DP) is a highly versatile technique that can exploit recursive structure in a problem.
- **Domain filtering** is straightforward for problems modeled as a DP.
- DP is also important in designing **filters** for some global constraints, such as the *stretch* constraint (employee scheduling).
- **Nonserial DP** is related to bucket elimination in CP and exploits the structure of the primal graph.
- DP modeling is the **art** of keeping the state space small while maintaining a Markovian property.
- We will examine only **one simple example** of serial DP.

Example: Capital Budgeting

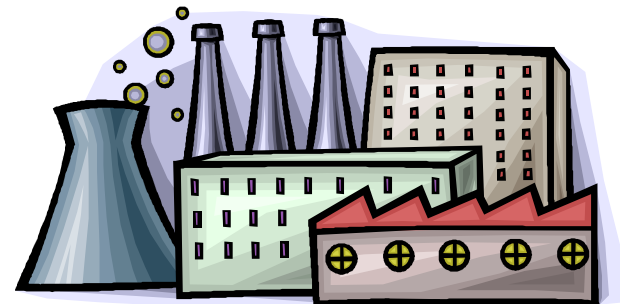
We wish to build power plants with a total cost of at most 12 million Euros.

There are three types of plants, costing 4, 2 or 3 million Euros each. We must build one or two of each type.

The problem has a simple knapsack packing model:

$$4x_1 + 2x_2 + 3x_3 \leq 12$$

Number of
factories of type j \rightarrow $x_j \in \{1, 2\}$



Example: Capital Budgeting

$$4x_1 + 2x_2 + 3x_3 \leq 12$$

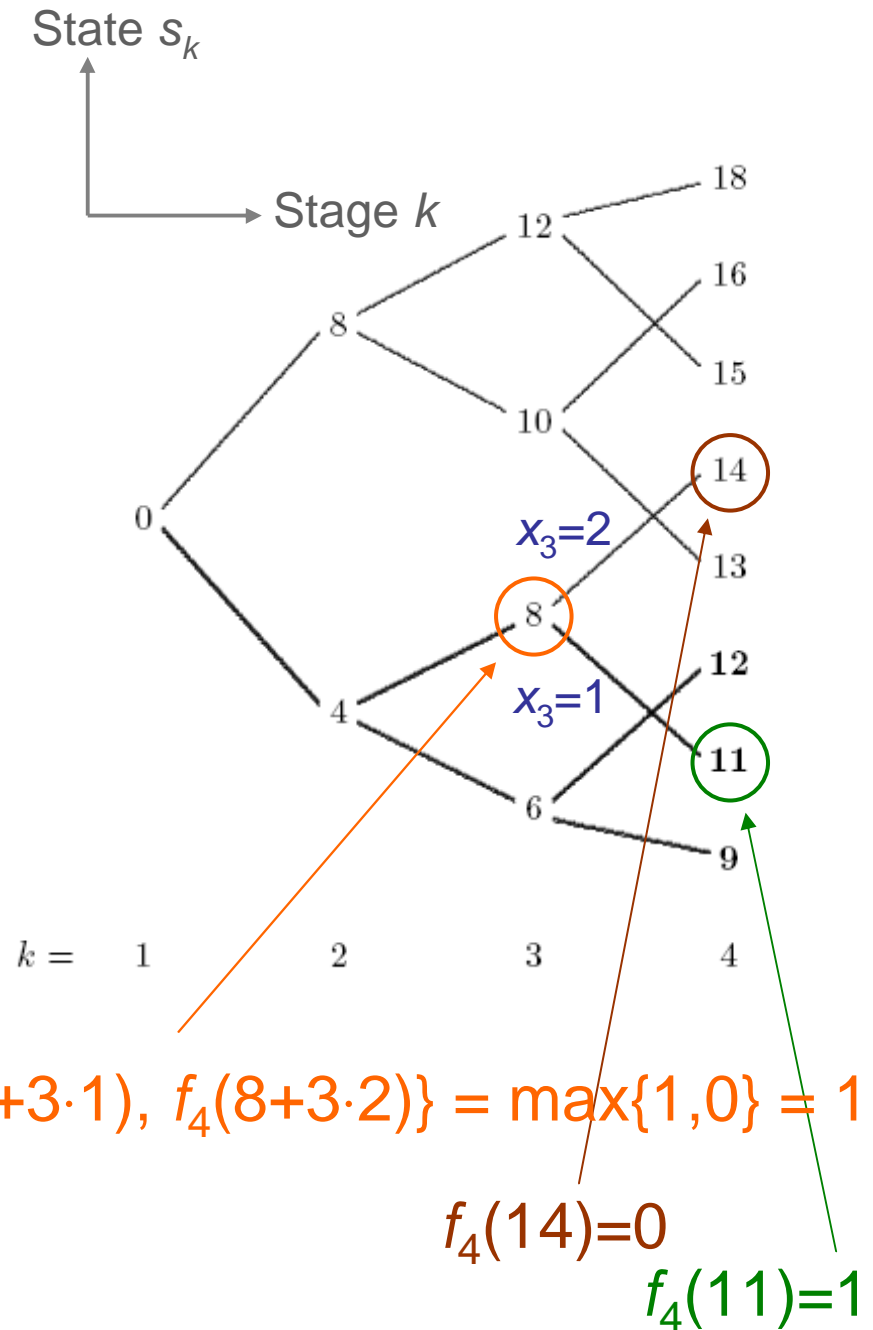
$$x_j \in \{1, 2\}$$

In general the recursion for $ax \leq b$ is

$$f_k(s_k) = \max_{x_k \in D_{x_k}} \{f_{k+1}(s_k + a_k x_k)\}$$

= 1 if there is
a path from
state s_k to a
feasible
solution,
0 otherwise

State is sum
of first k terms
of ax



Example: Capital Budgeting

$$4x_1 + 2x_2 + 3x_3 \leq 12$$

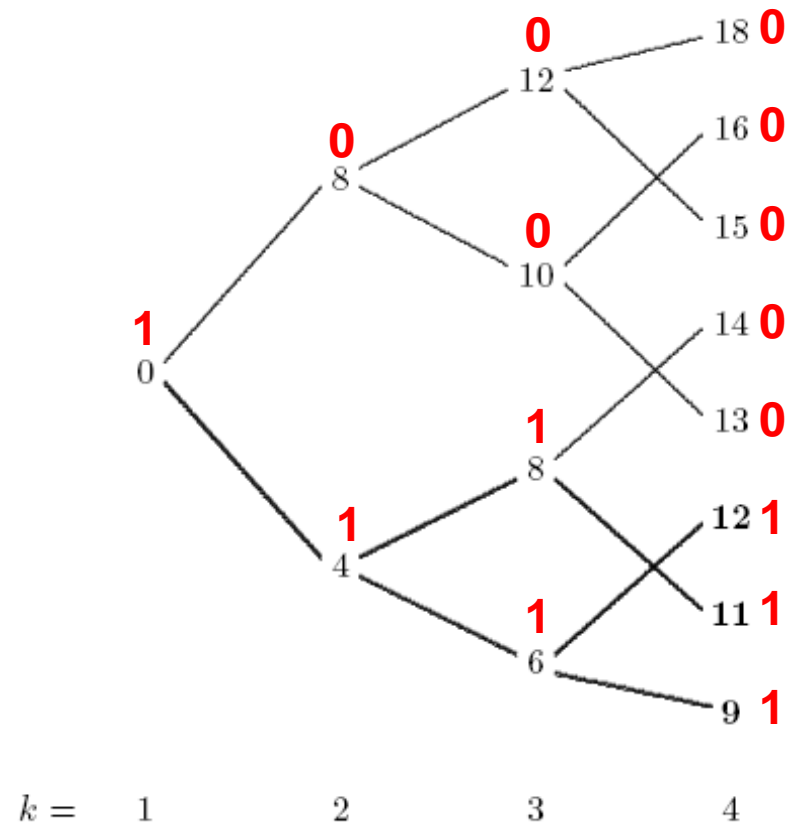
$$x_j \in \{1, 2\}$$

In general the recursion for $ax \leq b$ is

$$f_k(s_k) = \max_{x_k \in D_{x_k}} \{f_{k+1}(s_k + a_k x_k)\}$$

Boundary condition:

$$f_{n+1}(s_{n+1}) = \begin{cases} 1 & \text{if } s_{n+1} \leq b \\ 0 & \text{otherwise} \end{cases}$$



$f_k(s_k)$ for each state s_k

Example: Capital Budgeting

$$4x_1 + 2x_2 + 3x_3 \leq 12$$

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

The problem is feasible.

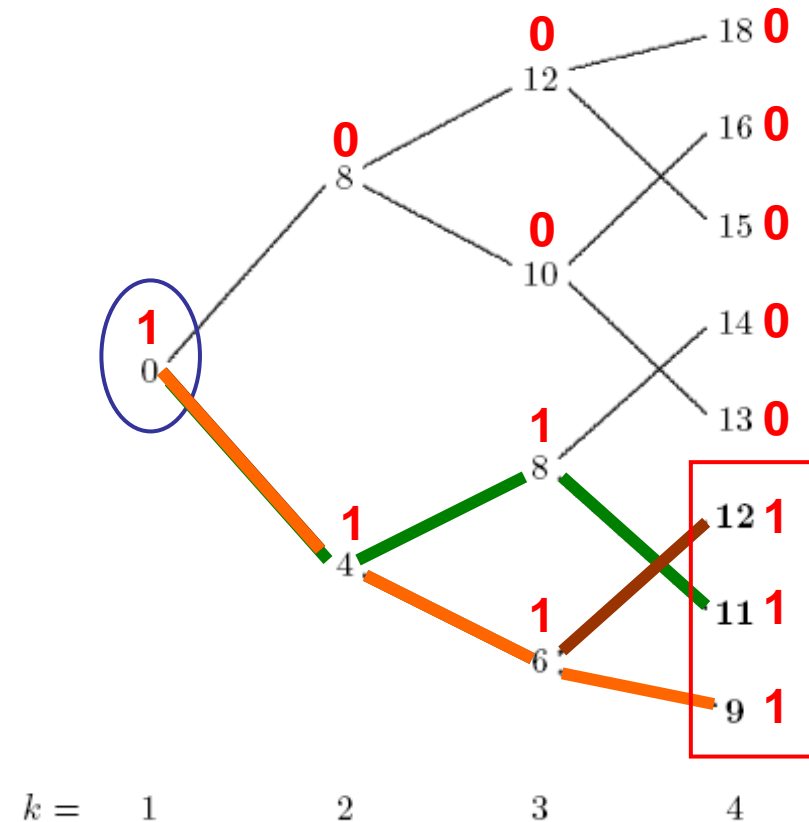
Each path to 1 is a feasible solution.

Path 1: $x = (1, 2, 1)$

Path 2: $x = (1, 1, 2)$

Path 3: $x = (1, 1, 1)$

Possible costs are 9, 11, 12.



$f_k(s_k)$ for each state s_k

Domain Filtering

$$4x_1 + 2x_2 + 3x_3 \leq 12$$

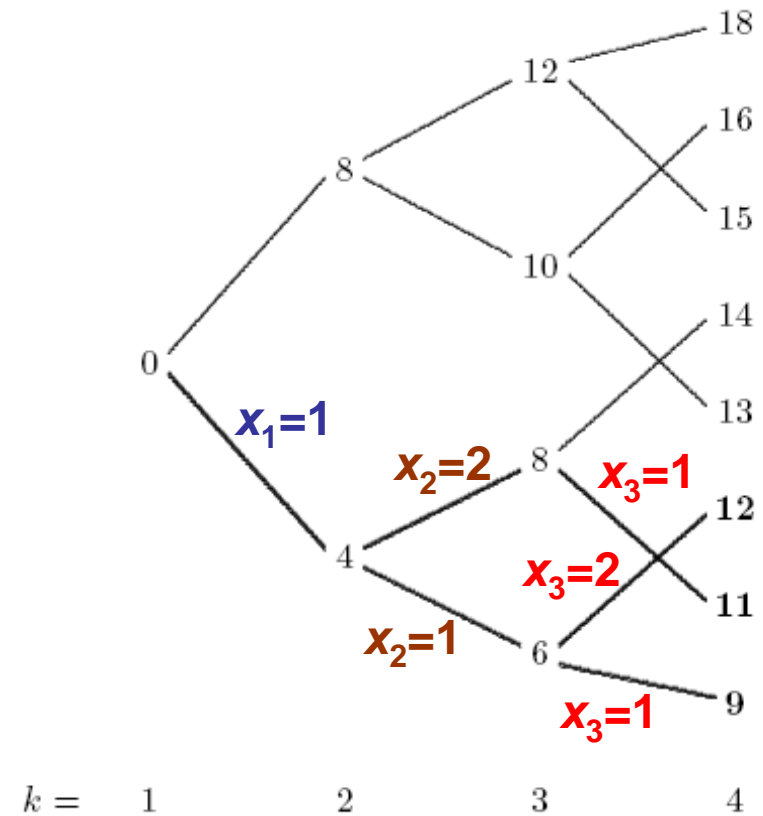
$$x_j \in \{1, 2\}$$

To filter domains: observe what values of x_k occur on feasible paths.

$$D_{x_3} = \{1, 2\}$$

$$D_{x_2} = \{1, 2\}$$

$$D_{x_1} = \{1\}$$



Recursive Optimization



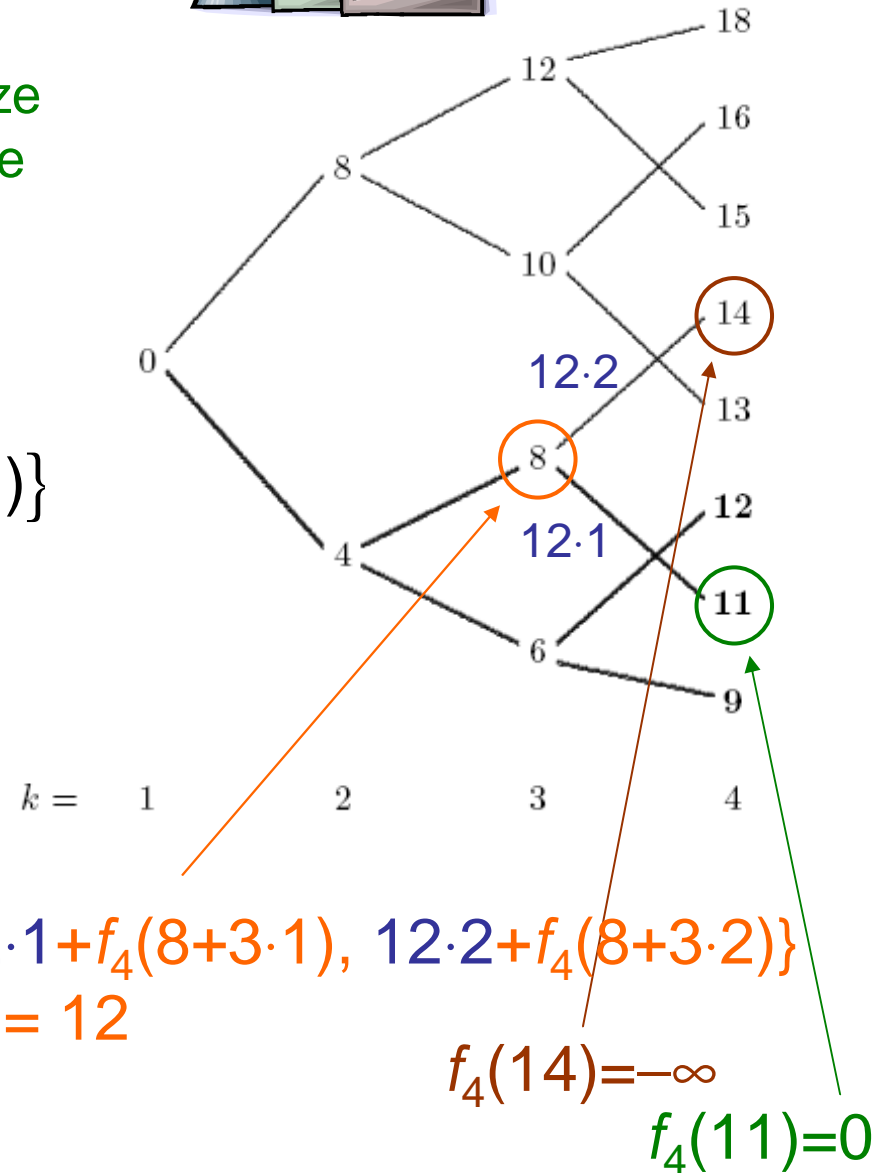
$$\begin{aligned} \max \quad & 15x_1 + 10x_2 + 12x_3 \quad \leftarrow \text{Maximize revenue} \\ & 4x_1 + 2x_2 + 3x_3 \leq 12 \\ & x_j \in \{1, 2\} \end{aligned}$$

The recursion includes arc values:

$$f_k(s_k) = \max_{x_k \in D_{x_k}} \{c_k x_k + f_{k+1}(s_k + a_k x_k)\}$$

= value on max
value path from
 s_k to final stage
(value to go)

Arc value



Recursive optimization

$$\max 15x_1 + 10x_2 + 12x_3$$

$$4x_1 + 2x_2 + 3x_3 \leq 12$$

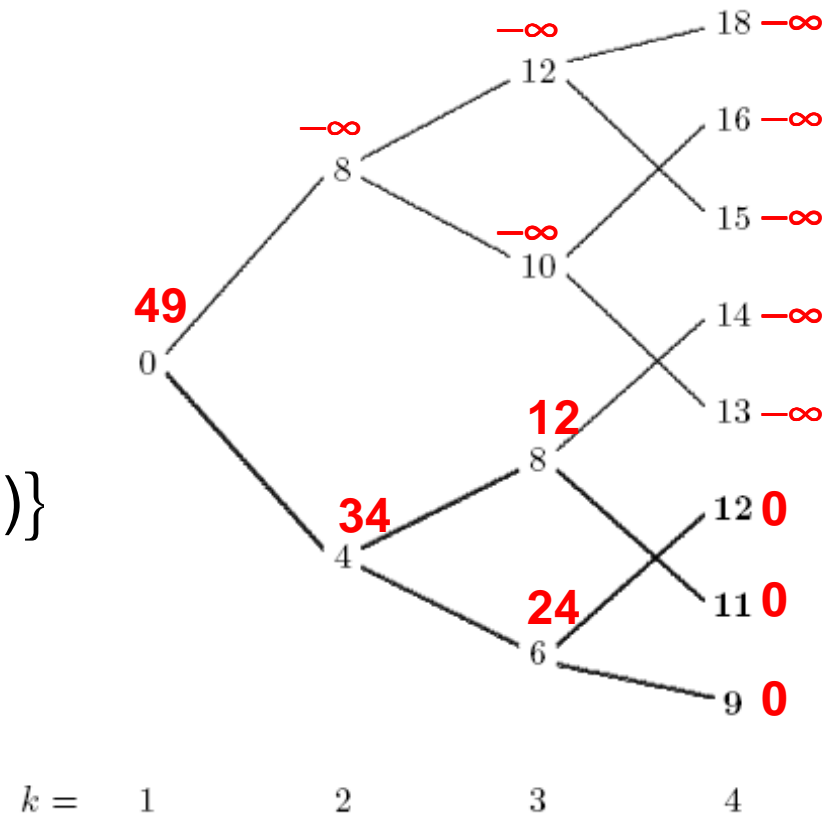
$$x_j \in \{1, 2\}$$

The recursion includes arc values:

$$f_k(s_k) = \max \{c_k x_k + f_{k+1}(s_k + a_k x_k)\}$$

Boundary condition:

$$f_{n+1}(s_{n+1}) = \begin{cases} 0 & \text{if } s_{n+1} \leq b \\ -\infty & \text{otherwise} \end{cases}$$



$f_k(s_k)$ for each state s_k

Recursive optimization

$$\max 15x_1 + 10x_2 + 12x_3$$

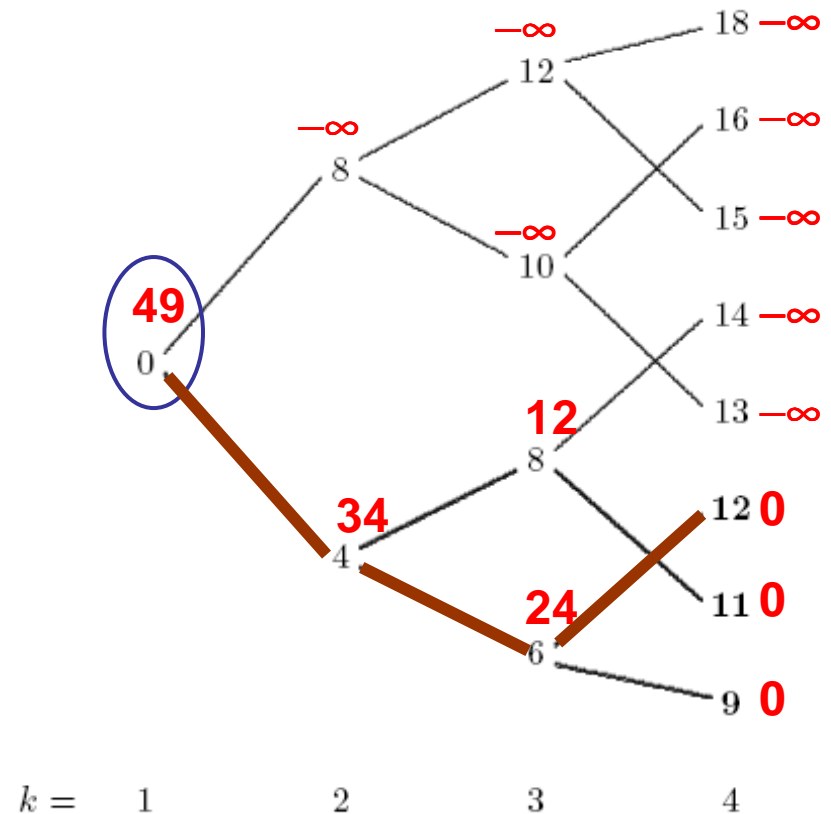
$$4x_1 + 2x_2 + 3x_3 \leq 12$$

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

The maximum revenue is 49.

The optimal path is easy to retrace.

$$(x_1, x_2, x_3) = (1, 1, 2)$$



$f_k(s_k)$ for each state s_k



CP-based Branch and Price

Basic Idea

Example: Airline Crew Scheduling

Motivation

- **Branch and price** allows solution of integer programming problems with a huge number of variables.
- The problem is solved by a branch-and-relax method. The difference lies in how the LP relaxation is solved.
- Variables are added to the LP relaxation only as needed.
- Variables are **priced** to find which ones should be added.
- **CP** is useful for solving the pricing problem, particularly when constraints are complex.
- **CP-based branch and price** has been successfully applied to airline crew scheduling, transit scheduling, and other transportation-related problems.

Basic Idea

Suppose the LP relaxation of an integer programming problem has a huge number of variables:

$$\begin{aligned} \min \quad & cx \\ & Ax = b \\ & x \geq 0 \end{aligned}$$

We will solve a **restricted master problem**, which has a small subset of the variables:

$$\begin{aligned} \min \quad & \sum_{j \in J} c_j x_j \\ & \sum_{j \in J} A_j x_j = b \quad (\lambda) \\ & x_j \geq 0 \end{aligned}$$

Column j of A



Adding x_k to the problem would improve the solution if x_k has a negative reduced cost:

$$r_k = c_k - \lambda A_k < 0$$

Basic Idea

Adding x_k to the problem would improve the solution if x_k has a negative reduced cost:

$$r_k = c_k - \lambda A_k < 0$$

Computing the reduced cost of x_k is known as **pricing** x_k .

So we solve the pricing problem: $\min c_y - \lambda y$
 y is a column of A



If the solution y^* satisfies $c_{y^*} - \lambda y^* < 0$, then we can add column y to the restricted master problem.

Basic Idea

The pricing problem $\max \lambda y$
 y is a column of A

need not be solved to optimality, so long as we find a column with negative reduced cost.

However, when we can no longer find an improving column, we solved the pricing problem to optimality to make sure we have the optimal solution of the LP.

If we can state constraints that the columns of A must satisfy, CP may be a good way to solve the pricing problem.

Example: Airline Crew Scheduling

We want to assign crew members to flights to minimize cost while covering the flights and observing complex work rules.



Flight data

j	s_j	f_j
1	0	3
2	1	3
3	5	8
4	6	9
5	10	12
6	14	16

Start time Finish time

A **roster** is the sequence of flights assigned to a single crew member.

The gap between two consecutive flights in a roster must be from 2 to 3 hours. Total flight time for a roster must be between 6 and 10 hours.

For example,

flight 1 cannot immediately precede 6

flight 4 cannot immediately precede 5.

The possible rosters are:

(1,3,5), (1,4,6), (2,3,5), (2,4,6)

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1 2 3 4
 (1,3,5), (1,4,6), (2,3,5), (2,4,6)



The LP relaxation of the problem is:

Cost of assigning crew member 1 to roster 2

$$\min z$$

10	12	7	13	9	11	6	12	$\begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \end{bmatrix}$	$=$	z	$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$
1	1	1	1	0	0	0	0	x_{11}	$=$	z	1
0	0	0	0	1	1	1	1	x_{12}	$=$	1	1
1	1	0	0	1	1	0	0	x_{13}	$=$	1	1
0	0	1	1	0	0	1	1	x_{14}	\geq	1	1
1	0	1	0	1	0	1	0	x_{21}	\geq	1	1
0	1	0	1	0	1	0	1	x_{22}	\geq	1	1
1	0	1	0	1	0	1	0	x_{23}	\geq	1	1
0	1	0	1	0	1	0	1	x_{24}	\geq	1	1

$x_{ik} \geq 0, \text{ all } i, k$

← Cost of assigning crew member 1 to roster 2

← = 1 if we assign crew member 1 to roster 2, = 0 otherwise.

← Each crew member is assigned to exactly 1 roster.

← Each flight is assigned at least 1 crew member.

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1 2 3 4
 (1,3,5), (1,4,6), (2,3,5), (2,4,6)



The LP relaxation of the problem is:

min z

Cost of assigning crew member 1 to roster 2

$$\begin{bmatrix}
 10 & 12 & 7 & 13 & 9 & 11 & 6 & 12 \\
 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
 \end{bmatrix}
 \begin{bmatrix}
 x_{11} \\
 x_{12} \\
 x_{13} \\
 x_{14} \\
 x_{21} \\
 x_{22} \\
 x_{23} \\
 x_{24}
 \end{bmatrix}
 =
 \begin{bmatrix}
 z \\
 1 \\
 1 \\
 1 \\
 1 \\
 1 \\
 1 \\
 1
 \end{bmatrix}$$

$x_{ik} \geq 0$, all i, k

= 1 if we assign crew member 1 to roster 2, = 0 otherwise.

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

Rosters that cover flight 1.

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1 2 3 4
 (1,3,5), (1,4,6), (2,3,5), (2,4,6)



The LP relaxation of the problem is:

min z

Cost of assigning crew member 1 to roster 2

$$\begin{bmatrix}
 10 & 12 & 7 & 13 & 9 & 11 & 6 & 12 \\
 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
 \end{bmatrix}
 \begin{bmatrix}
 x_{11} \\
 x_{12} \\
 x_{13} \\
 x_{14} \\
 x_{21} \\
 x_{22} \\
 x_{23} \\
 x_{24}
 \end{bmatrix}
 =
 \begin{bmatrix}
 z \\
 1 \\
 1 \\
 1 \\
 1 \\
 1 \\
 1 \\
 1
 \end{bmatrix}$$

$x_{ik} \geq 0$, all i, k

= 1 if we assign crew member 1 to roster 2, = 0 otherwise.

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

Rosters that cover flight 2.

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1 2 3 4
 (1,3,5), (1,4,6), (2,3,5), (2,4,6)



The LP relaxation of the problem is:

min z

Cost of assigning crew member 1 to roster 2

$$\begin{bmatrix}
 10 & 12 & 7 & 13 & 9 & 11 & 6 & 12 \\
 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
 \end{bmatrix}
 \begin{bmatrix}
 x_{11} \\
 x_{12} \\
 x_{13} \\
 x_{14} \\
 x_{21} \\
 x_{22} \\
 x_{23} \\
 x_{24}
 \end{bmatrix}
 =
 \begin{bmatrix}
 z \\
 1 \\
 1 \\
 1 \\
 1 \\
 1 \\
 1 \\
 1
 \end{bmatrix}$$

$x_{ik} \geq 0$, all i, k

= 1 if we assign crew member 1 to roster 2, = 0 otherwise.

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

Rosters that cover flight 3.

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1 2 3 4
 (1,3,5), (1,4,6), (2,3,5), (2,4,6)



The LP relaxation of the problem is:

min z

Cost of assigning crew member 1 to roster 2

$$\begin{bmatrix}
 10 & 12 & 7 & 13 & 9 & 11 & 6 & 12 \\
 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
 \end{bmatrix}
 \begin{bmatrix}
 x_{11} \\
 x_{12} \\
 x_{13} \\
 x_{14} \\
 x_{21} \\
 x_{22} \\
 x_{23} \\
 x_{24}
 \end{bmatrix}
 =
 \begin{bmatrix}
 z \\
 1 \\
 1 \\
 1 \\
 1 \\
 1 \\
 1 \\
 1
 \end{bmatrix}$$

$x_{ik} \geq 0$, all i, k

= 1 if we assign crew member 1 to roster 2, = 0 otherwise.

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

Rosters that cover flight 4.

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1 2 3 4
 (1,3,5), (1,4,6), (2,3,5), (2,4,6)



The LP relaxation of the problem is:

min z

Cost of assigning crew member 1 to roster 2

$$\begin{bmatrix}
 10 & 12 & 7 & 13 & 9 & 11 & 6 & 12 \\
 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
 \end{bmatrix}
 \begin{bmatrix}
 x_{11} \\
 x_{12} \\
 x_{13} \\
 x_{14} \\
 x_{21} \\
 x_{22} \\
 x_{23} \\
 x_{24}
 \end{bmatrix}
 =
 \begin{bmatrix}
 z \\
 1 \\
 1 \\
 1 \\
 1 \\
 1 \\
 1 \\
 1
 \end{bmatrix}$$

$x_{ik} \geq 0$, all i, k

= 1 if we assign crew member 1 to roster 2, = 0 otherwise.

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

Rosters that cover flight 5.

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1 2 3 4
 (1,3,5), (1,4,6), (2,3,5), (2,4,6)



The LP relaxation of the problem is:

Cost of assigning crew member 1 to roster 2

$$\min z$$

10	12	7	13	9	11	6	12	x_{11}	=	z
1	1	1	1	0	0	0	0	12	=	1
0	0	0	0	1	1	1	1	x_{13}	=	1
1	1	0	0	1	1	0	0	x_{14}	\geq	1
0	0	1	1	0	0	1	1	x_{21}	\geq	1
1	0	1	0	1	0	1	0	x_{22}	\geq	1
0	1	0	1	0	1	0	1	x_{23}	\geq	1
1	0	1	0	1	0	1	0	x_{24}	\geq	1
0	1	0	1	0	1	0	1		\geq	1

$x_{ik} \geq 0, \text{ all } i, k$

= 1 if we assign crew member 1 to roster 2, = 0 otherwise.

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

Rosters that cover flight 6.

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1 2 3 4
 (1,3,5), (1,4,6), (2,3,5), (2,4,6)



The LP relaxation of the problem is:

min z

Cost c_{12} of assigning crew member 1 to roster 2

$$\begin{bmatrix} 10 & 12 & 7 & 13 & 9 & 11 & 6 & 12 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \end{bmatrix} = \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$x_{ik} \geq 0$, all i, k

= 1 if we assign crew member 1 to roster 2, = 0 otherwise.

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

In a real problem, there can be **millions** of rosters.

Airline Crew Scheduling

We start by solving the problem with a subset of the columns:

$\min z$

$$\begin{bmatrix} 10 & 13 & 9 & 12 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{14} \\ x_{21} \\ x_{24} \end{bmatrix} = \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$x_{ik} \geq 0, \text{ all } i, k$

Optimal
dual
solution

$$\begin{bmatrix} (10) \\ (9) \\ (0) \\ (0) \\ (0) \\ (0) \\ (0) \\ (0) \\ (3) \end{bmatrix} \begin{matrix} u_1 \\ u_2 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{matrix}$$



Airline Crew Scheduling

We start by solving the problem with a subset of the columns:



min z

$$\begin{bmatrix} 10 & 13 & 9 & 12 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{14} \\ x_{21} \\ x_{24} \end{bmatrix} = \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$x_{ik} \geq 0$, all i, k

Dual
variables

$$\begin{array}{l} (10) \quad u_1 \\ (9) \quad u_2 \\ (0) \quad v_1 \\ (0) \quad v_2 \\ (0) \quad v_3 \\ (0) \quad v_4 \\ (0) \quad v_5 \\ (3) \quad v_6 \end{array}$$

Airline Crew Scheduling

We start by solving the problem with a subset of the columns:



min z

$$\begin{bmatrix} 10 & 13 & 9 & 12 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{14} \\ x_{21} \\ x_{24} \end{bmatrix} = \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$x_{ik} \geq 0$, all i, k

Dual
variables

$$\begin{array}{l} (10) \quad u_1 \\ (9) \quad u_2 \\ (0) \quad v_1 \\ (0) \quad v_2 \\ (0) \quad v_3 \\ (0) \quad v_4 \\ (0) \quad v_5 \\ (3) \quad v_6 \end{array}$$

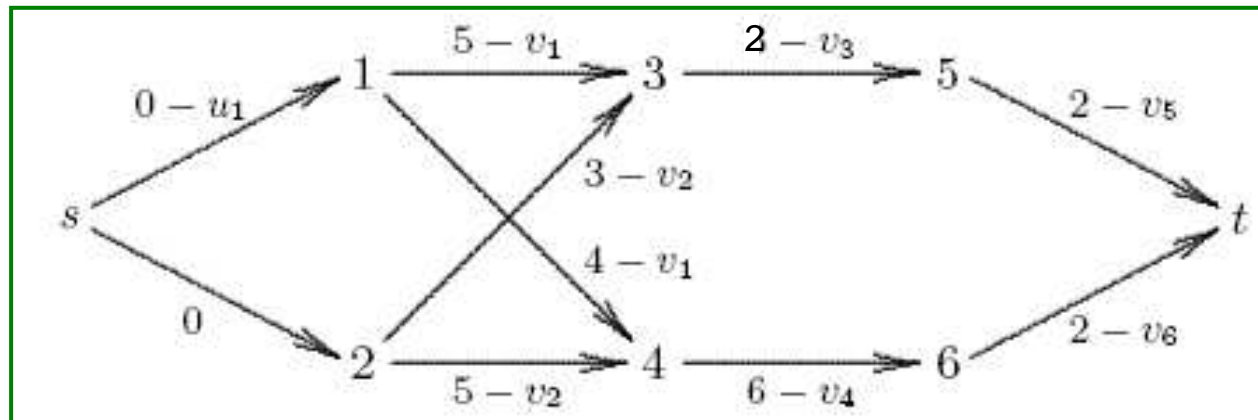
The reduced cost of an excluded roster k for crew member i is

$$c_{ik} - u_i - \sum_{j \text{ in roster } k} v_j$$

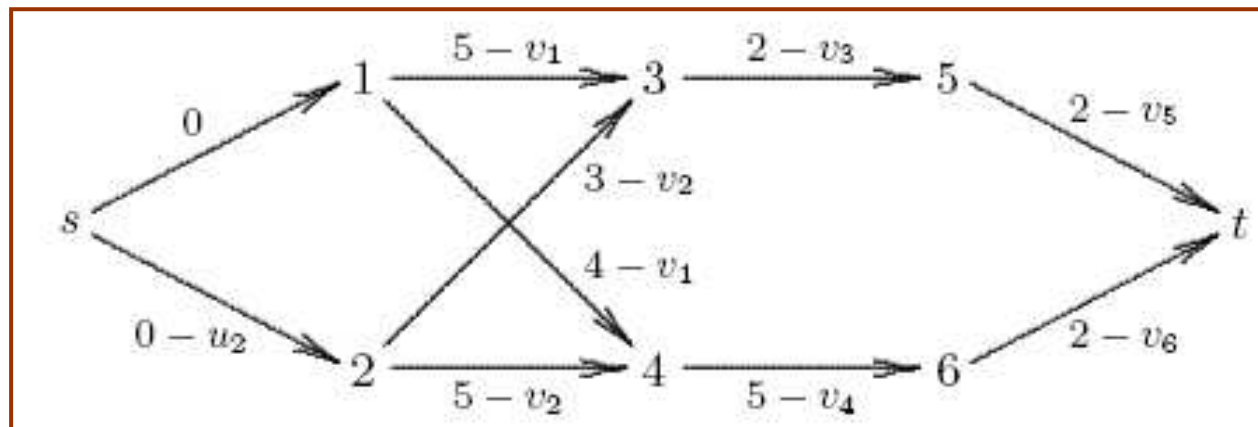
We will formulate the pricing problem as a shortest path problem.

Pricing problem

Crew
member 1



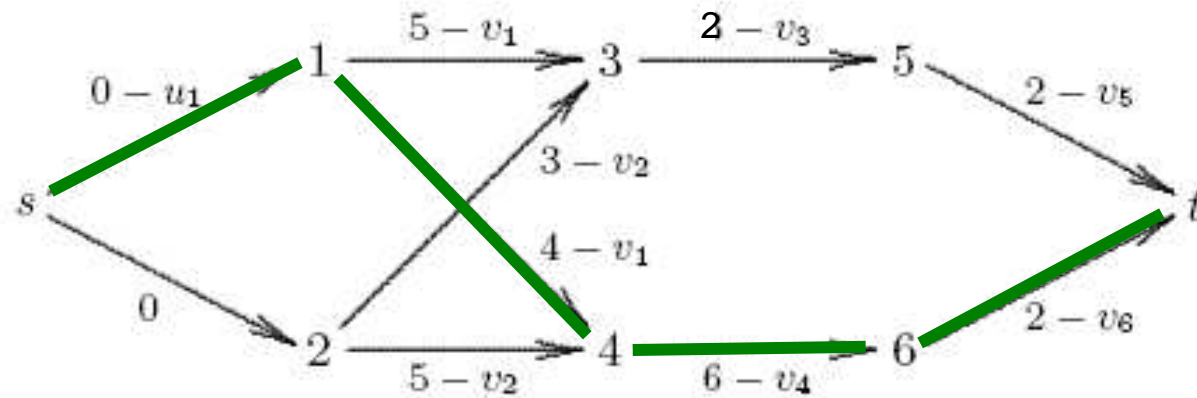
Crew
member 2



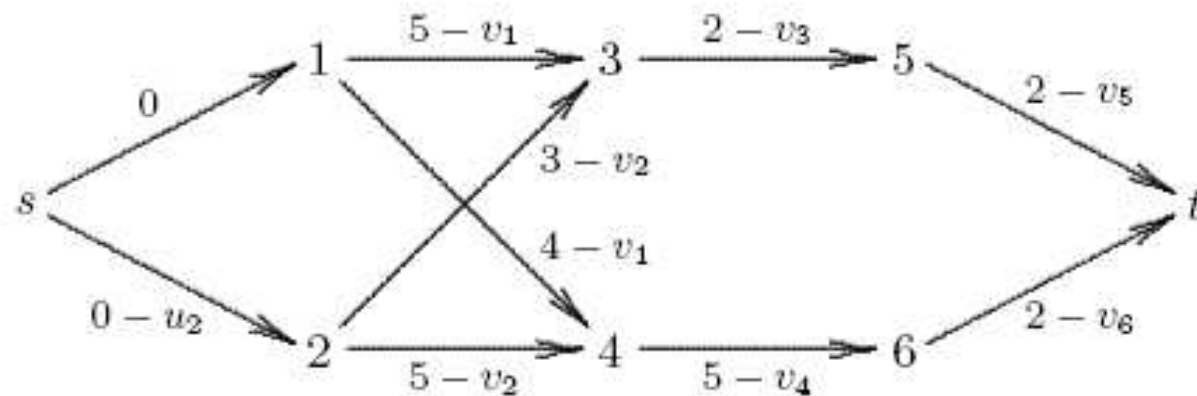
Pricing problem

Each s-t path corresponds to a roster, provided the flight time is within bounds.

Crew member 1



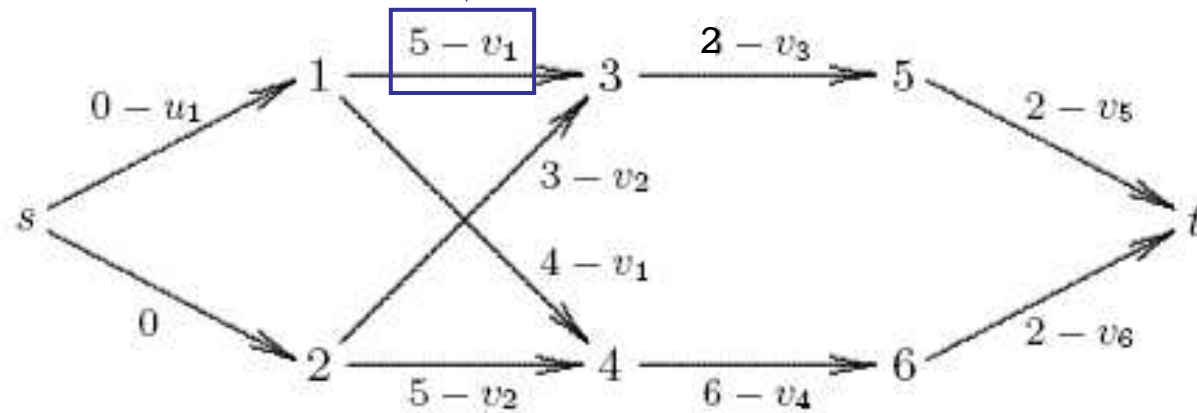
Crew member 2



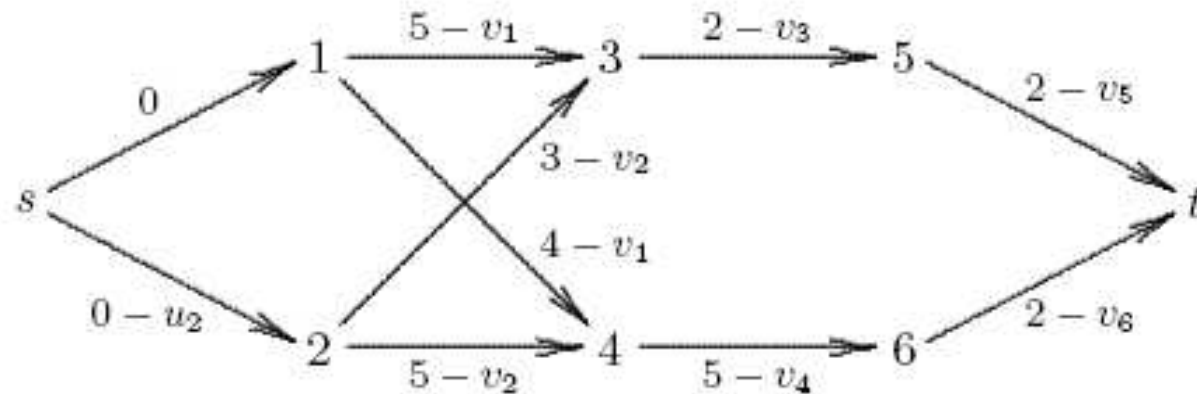
Pricing problem

Cost of flight 3 if it immediately follows flight 1, offset by dual multiplier for flight 1

Crew member 1



Crew member 2

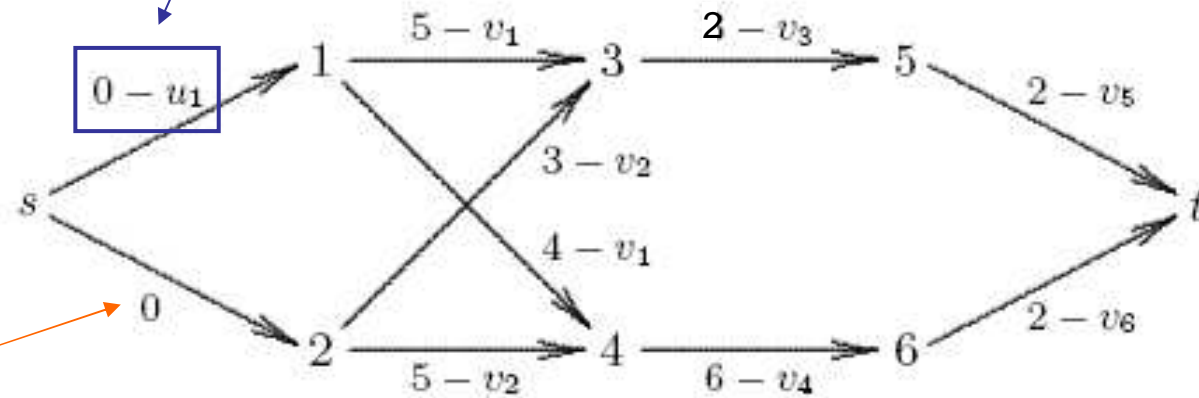


Pricing problem

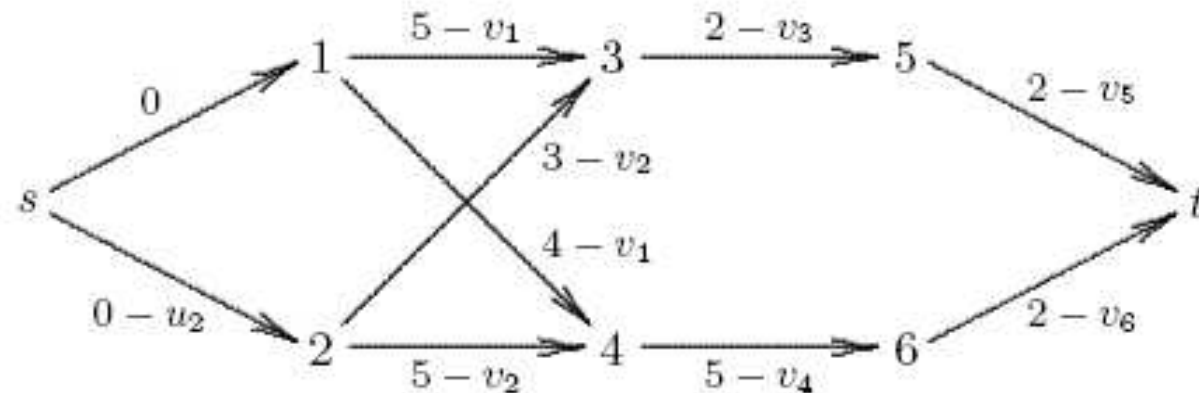
Cost of transferring from home to flight 1,
offset by dual multiplier for crew member 1

Crew
member 1

Dual multiplier
omitted to break
symmetry



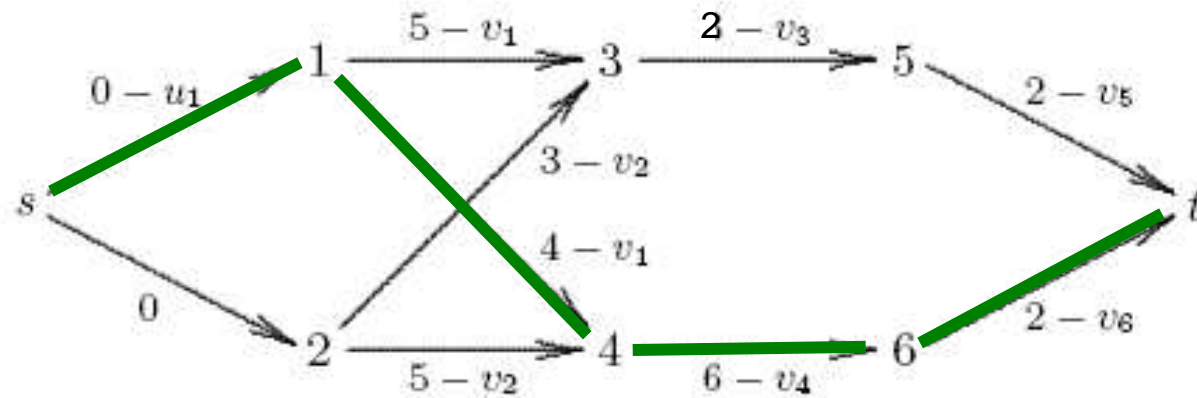
Crew
member 2



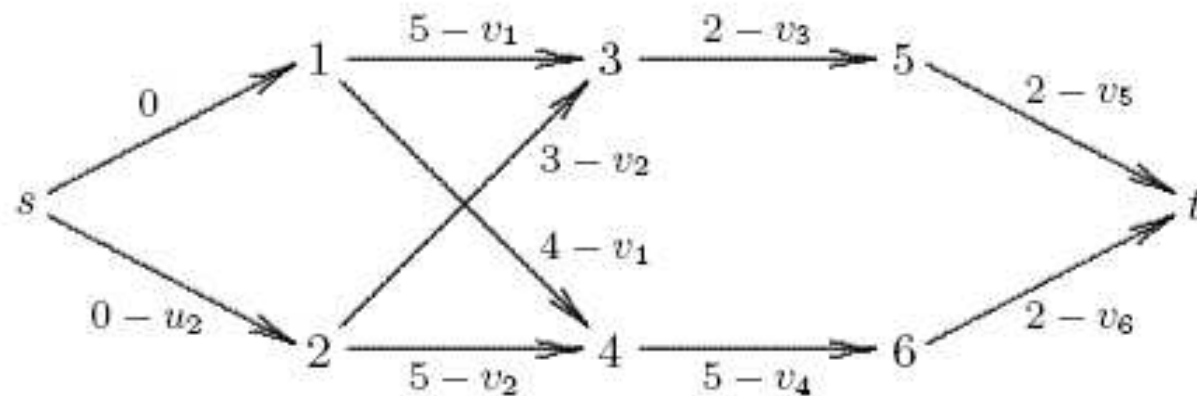
Pricing problem

Length of a path is reduced cost of the corresponding roster.

Crew
member 1



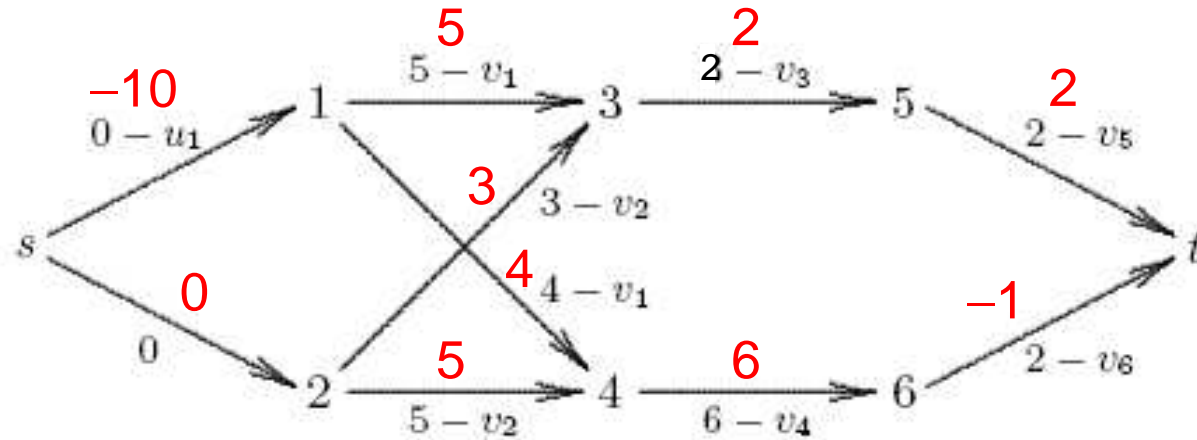
Crew
member 2



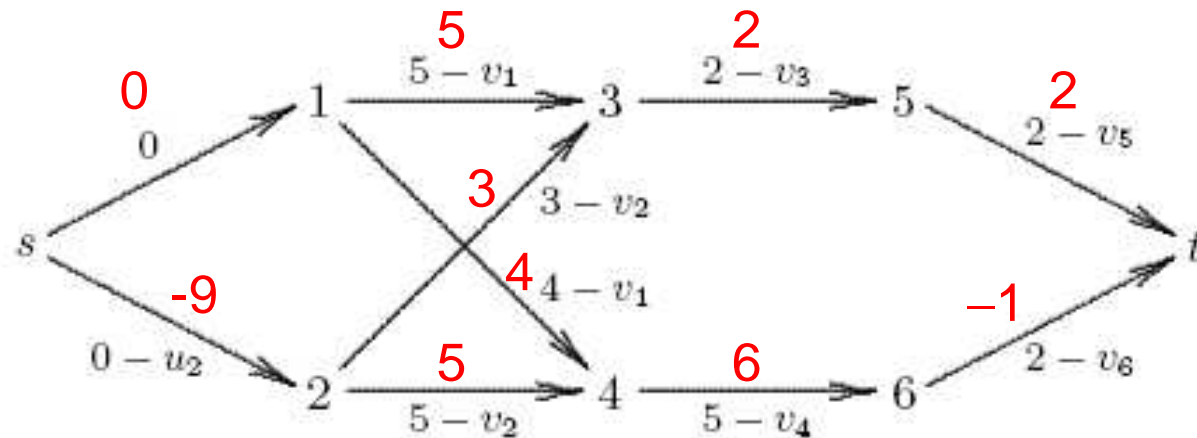
Pricing problem

Arc lengths using dual solution of LP relaxation

Crew member 1



Crew member 2

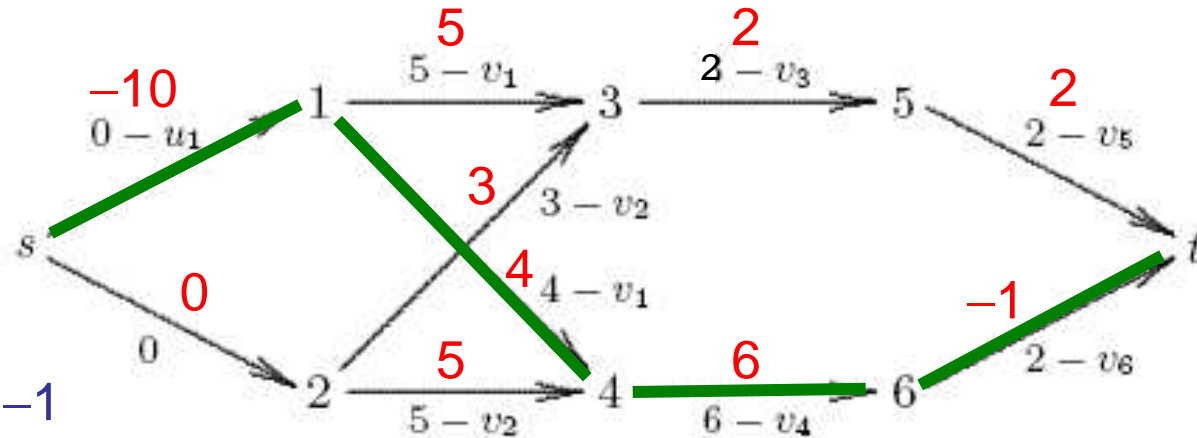


Pricing problem

Solution of shortest path problems

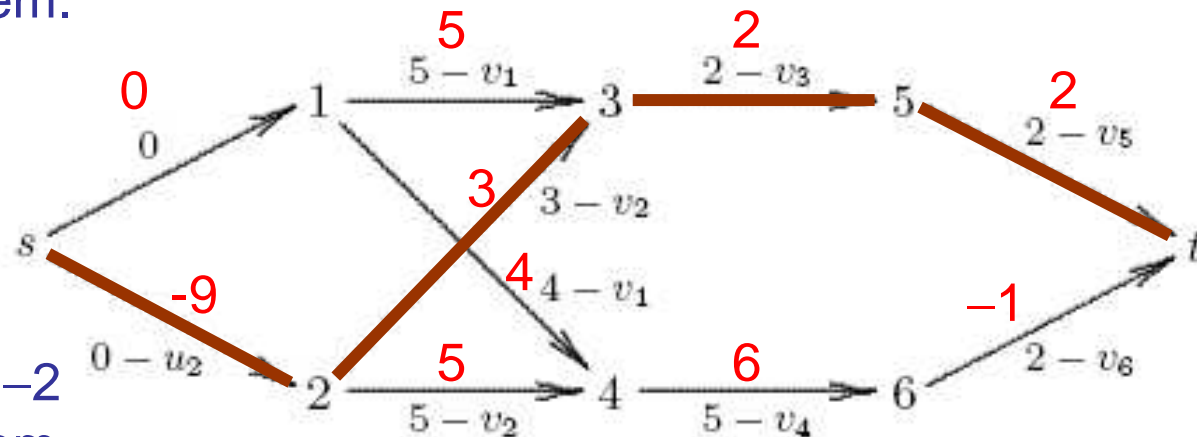
Crew
member 1

Reduced cost = -1
Add x_{12} to problem.



Crew
member 2

Reduced cost = -2
Add x_{23} to problem.

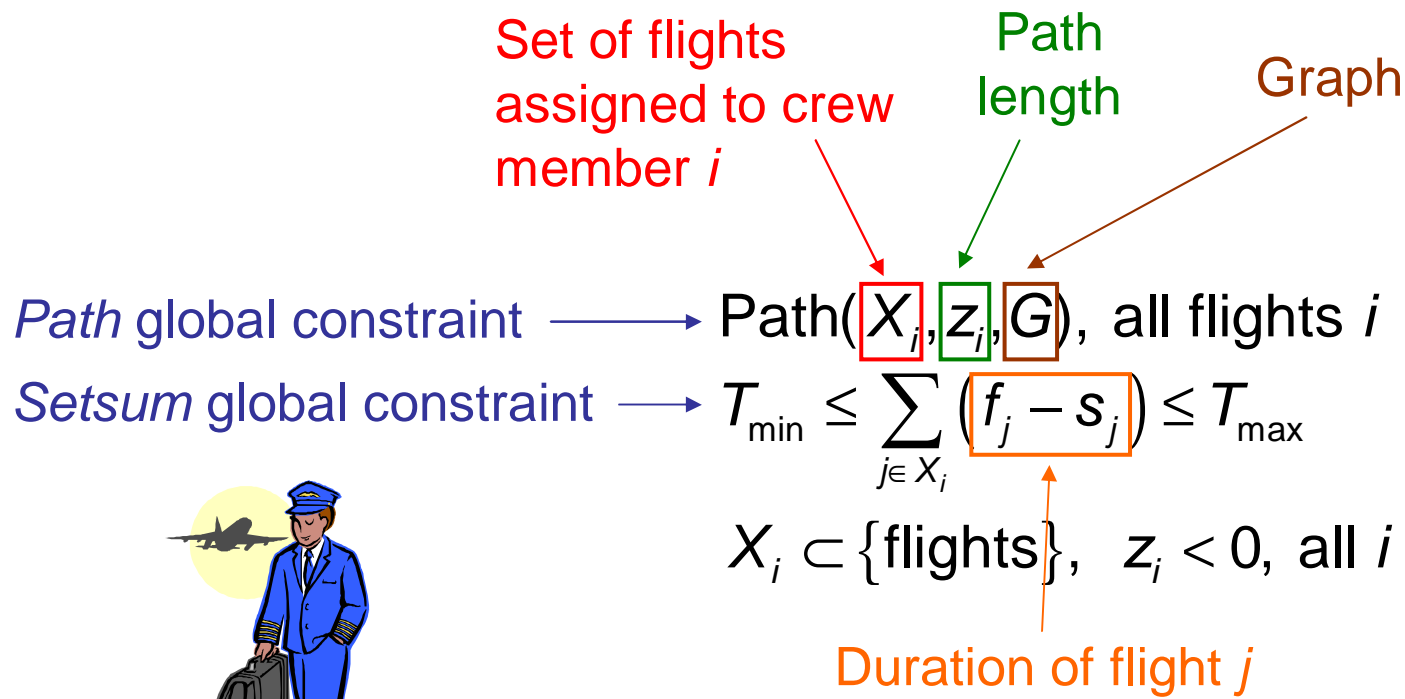


After x_{12} and x_{23} are added to the problem, no remaining variable has negative reduced cost.

Pricing problem

The shortest path problem cannot be solved by traditional shortest path algorithms, due to the bounds on total duration of flights.

It **can** be solved by CP:





CP-based Benders Decomposition

Benders Decomposition in the Abstract

Classical Benders Decomposition

Example: Machine Scheduling

Motivation

- **Benders decomposition** allows us to apply CP and OR to different parts of the problem.
- It searches over values of certain variables that, when fixed, result in a much simpler **subproblem**.
- The search learns from past experience by accumulating **Benders cuts** (a form of nogood).
- The technique can be **generalized** far beyond the original OR conception.
- Generalized Benders methods have resulted in the **greatest speedups** achieved by combining CP and OR.

Benders Decomposition in the Abstract

Benders decomposition
can be applied to
problems of the form

$$\min f(x, y)$$

$$S(x, y)$$

$$x \in D_x, y \in D_y$$

When x is fixed to some
value, the resulting
subproblem is much
easier:

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

$$S(\bar{x}, y)$$

$$y \in D_y$$

...perhaps
because it
decouples into
smaller problems.

For example, suppose x assigns jobs to machines, and y schedules the jobs on the machines.

When x is fixed, the problem decouples into a separate scheduling subproblem for each machine.

Benders Decomposition

We will search over assignments to x . This is the **master problem**.

In iteration k we assume $x = x^k$ and solve the subproblem

$$\min_{y \in D_y} f(x^k, y)$$

and get optimal value v_k

We generate a **Benders cut** (a type of nogood) $\boxed{v} \geq B_{k+1}(x)$

that satisfies $B_{k+1}(x^k) = v_k$.

Cost in the original problem

The Benders cut says that if we set $x = x^k$ again, the resulting cost v will be at least v_k . To do better than v_k , we must try something else.

It also says that any other x will result in a cost of at least $B_{k+1}(x)$, perhaps due to some similarity between x and x^k .

Benders Decomposition

We will search over assignments to x . This is the **master problem**.

In iteration k we assume $x = x^k$ and solve the subproblem

$$\min_{y \in D_y} \begin{matrix} f(x^k, y) \\ S(x^k, y) \end{matrix}$$

and get optimal value v_k

We generate a **Benders cut** (a type of nogood) $\boxed{v} \geq B_{k+1}(x)$

that satisfies $B_{k+1}(x) = v_k$.

Cost in the original problem

We add the Benders cut to the master problem, which becomes

$$\min v$$

$$v \geq B_i(x), \quad i = 1, \dots, k+1$$

$$x \in D_x$$

Benders cuts
generated so far

Benders Decomposition

We now solve the master problem

$$\begin{array}{ll} \min & v \\ & v \geq B_i(x), \quad i = 1, \dots, k+1 \\ & x \in D_x \end{array}$$

to get the next trial value x^{k+1} .

The master problem is a relaxation of the original problem, and its optimal value is a **lower bound** on the optimal value of the original problem.

The subproblem is a restriction, and its optimal value is an **upper bound**.

The process continues until the bounds meet.

The Benders cuts partially define the **projection** of the feasible set onto x . We hope not too many cuts are needed to find the optimum.

Classical Benders Decomposition

The classical method
applies to problems
of the form

$$\begin{aligned} \min \quad & f(x) + cy \\ & g(x) + Ay \geq b \\ & x \in D_x, \quad y \geq 0 \end{aligned}$$

and the subproblem
is an LP

$$\begin{aligned} \min \quad & f(x^k) + cy \\ & Ay \geq b - g(x^k) \quad (\lambda) \\ & y \geq 0 \end{aligned}$$

whose dual is

$$\begin{aligned} \max \quad & f(x^k) + \lambda(b - g(x^k)) \\ & \lambda A \leq c \\ & \lambda \geq 0 \end{aligned}$$

Let λ^k solve the dual.

By strong duality, $B_{k+1}(x) = f(x) + \lambda^k(b - g(x))$ is the tightest lower bound on the optimal value v of the original problem when $x = x^k$.

Even for other values of x , λ^k **remains feasible in the dual**. So by weak duality, $B_{k+1}(x)$ remains a lower bound on v .

Classical Benders

So the master problem

$$\min v$$

$$v \geq B_i(x), \quad i = 1, \dots, k+1$$

$$x \in D_x$$

becomes

$$\min v$$

$$v \geq f(x) + \lambda^i(b - g(x)), \quad i = 1, \dots, k+1$$

$$x \in D_x$$

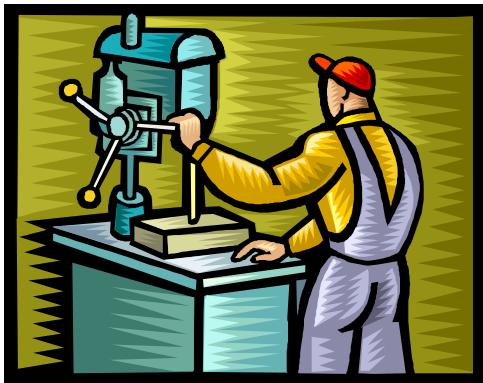
In most applications the master problem is

- an MILP
- a nonlinear programming problem (NLP), or
- a mixed integer/nonlinear programming problem (MINLP).

Example: Machine Scheduling

- Assign 5 jobs to 2 machines (A and B), and schedule the machines assigned to each machine within time windows.
- The objective is to minimize **makespan**.

Time lapse between
start of first job and
end of last job.



- Assign the jobs in the **master problem**, to be solved by **MILP**.
- Schedule the jobs in the **subproblem**, to be solved by **CP**.

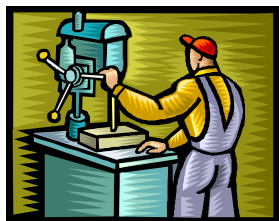
Machine Scheduling

Job Data

<i>Job j</i>	<i>Release time</i>	<i>Dead- line</i>	<i>Processing time</i>	
	r_j	d_j	p_{Aj}	p_{Bj}
1	0	10	1	5
2	0	10	3	6
3	2	7	3	7
4	2	10	4	6
5	4	7	2	5

Once jobs are assigned, we can minimize overall makespan by minimizing makespan on each machine individually.

So the subproblem decouples.



Machine A

Machine B

Machine Scheduling

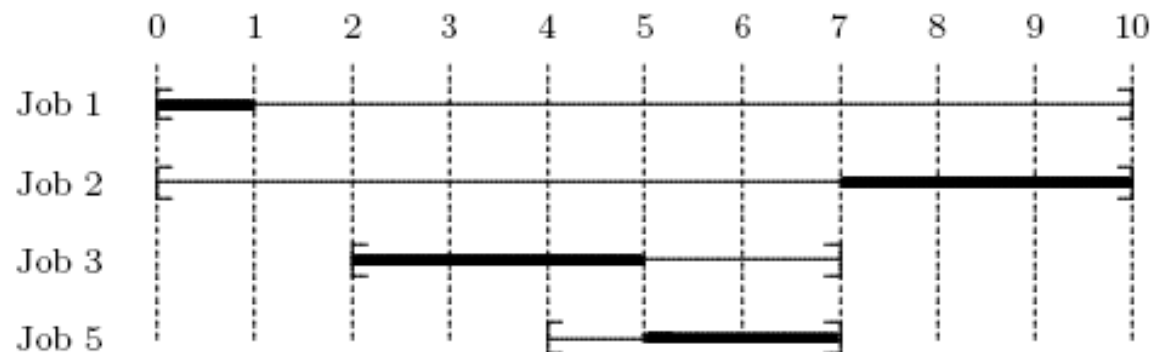
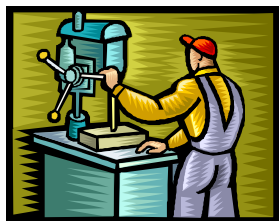
Job Data

<i>Job j</i>	<i>Release time</i>	<i>Dead- line</i>	<i>Processing time</i>	
	r_j	d_j	p_{Aj}	p_{Bj}
1	0	10	1	5
2	0	10	3	6
3	2	7	3	7
4	2	10	4	6
5	4	7	2	5

Once jobs are assigned, we can minimize overall makespan by minimizing makespan on each machine individually.

So the subproblem decouples.

Minimum makespan
schedule for jobs 1, 2, 3, 5
on machine A



Machine Scheduling

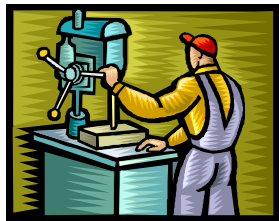
The problem is

$$\begin{aligned} \min \quad & M \\ M \geq & \boxed{s_j} + p_{x_j j}, \text{ all } j \\ r_j \leq & s_j \leq d_j - p_{x_j j}, \text{ all } j \\ \text{disjunctive} \quad & ((s_j | x_j = i), (p_{ij} | x_j = i)), \text{ all } i \end{aligned}$$

Start time of job j

Time windows

Jobs cannot overlap



Machine Scheduling

The problem is

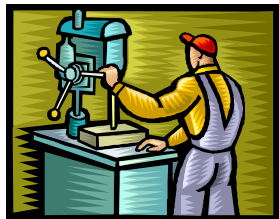
$$\begin{aligned}
 &\min M \\
 &M \geq \boxed{s_j} + p_{x_j j}, \text{ all } j \\
 &r_j \leq s_j \leq d_j - p_{x_j j}, \text{ all } j \\
 &\text{disjunctive}((s_j | x_j = i), (p_{ij} | x_j = i)), \text{ all } i
 \end{aligned}$$

Start time of job j

Time windows

Jobs cannot overlap

For a fixed assignment \bar{x} the subproblem on each machine i is

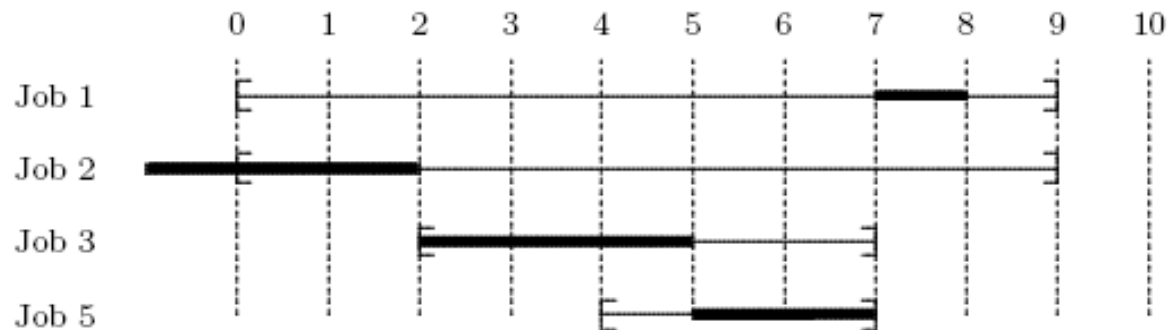


$$\begin{aligned}
 &\min M \\
 &M \geq s_j + p_{\bar{x}_j j}, \text{ all } j \text{ with } \bar{x}_j = i \\
 &r_j \leq s_j \leq d_j - p_{\bar{x}_j j}, \text{ all } j \text{ with } \bar{x}_j = i \\
 &\text{disjunctive}((s_j | \bar{x}_j = i), (p_{ij} | \bar{x}_j = i))
 \end{aligned}$$

Benders cuts

Suppose we assign jobs 1,2,3,5 to machine A in iteration k .

We can prove that 10 is the optimal makespan by proving that the schedule is infeasible with makespan 9.



Edge finding derives infeasibility by reasoning only with jobs 2,3,5. So these jobs alone create a minimum makespan of 10.

So we have a Benders cut

$$v \geq B_{k+1}(x) = \begin{cases} 10 & \text{if } x_2 = x_3 = x_4 = A \\ 0 & \text{otherwise} \end{cases}$$

Benders cuts

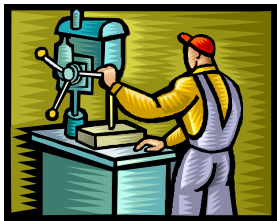
We want the master problem to be an MILP, which is good for assignment problems.

So we write the Benders cut

$$v \geq B_{k+1}(x) = \begin{cases} 10 & \text{if } x_2 = x_3 = x_4 = A \\ 0 & \text{otherwise} \end{cases}$$

Using 0-1 variables: $v \geq 10(x_{A2} + x_{A3} + x_{A5} - 2)$
 $v \geq 0$

x_{A5} = 1 if job 5 is assigned to machine A



Master problem

The master problem is an MILP:

$$\min v$$

$$\sum_{j=1}^5 p_{Aj} x_{Aj} \leq 10, \text{ etc.}$$

$$\sum_{j=1}^5 p_{Bj} x_{Bj} \leq 10, \text{ etc.}$$

$$v \geq \sum_{j=1}^5 p_{ij} x_{ij}, \quad v \geq 2 + \sum_{j=3}^5 p_{ij} x_{ij}, \text{ etc.}, \quad i = A, B$$

$$v \geq 10(x_{A2} + x_{A3} + x_{A5} - 2)$$

$$v \geq 8x_{B4}$$

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

Constraints derived from time windows

Constraints derived from release times

Benders cut from machine A

Benders cut from machine B

Stronger Benders cuts

If all release times are the same, we can strengthen the Benders cuts.

We are now using the cut

$$v \geq M_{ik} \left(\sum_{j \in J_{ik}} x_{ij} - |J_{ik}| + 1 \right)$$

Min makespan
on machine i
in iteration k

Set of jobs
assigned to
machine i in
iteration k

A stronger cut provides a useful bound even if only some of the jobs in J_{ik} are assigned to machine i :

$$v \geq M_{ik} - \sum_{j \in J_{ik}} (1 - x_{ij}) p_{ij}$$

These results can be generalized to cumulative scheduling.

