Benders Decomposition

John Hooker
Carnegie Mellon University

CP Summer School
Cork, Ireland, June 2016

Outline

- Essence of Benders decomposition
  - Simple example
- Logic-based Benders
- Inference dual
- Classical LP dual
- Classical Benders
- Examples…

Outline

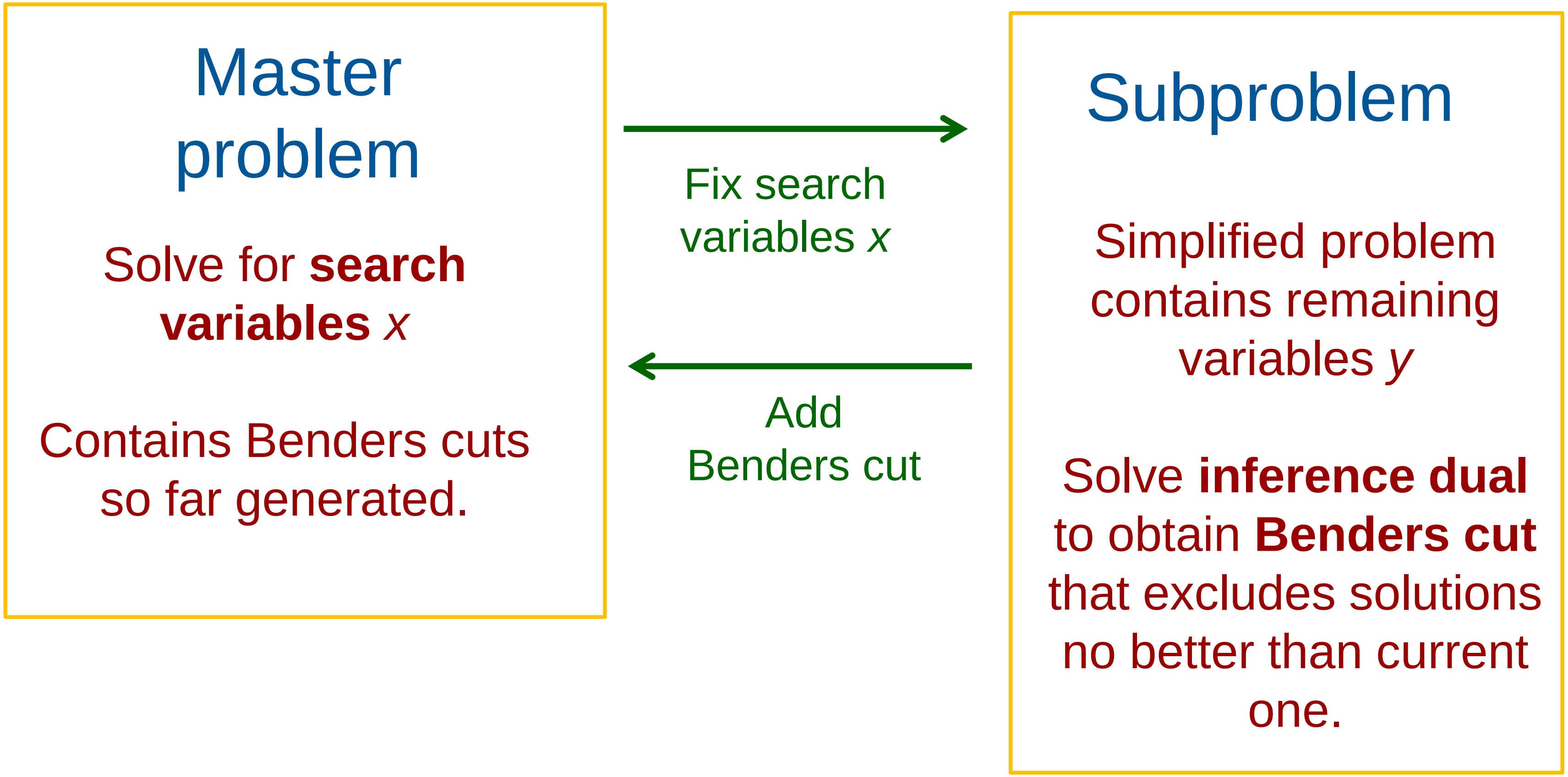
- Examples
  - Logic circuit verification
  - Planning and disjunctive scheduling
  - Planning and cumulative scheduling
    - Min cost
    - Min makespan
    - Min number of late tasks
    - Min total tardilness
  - Single-resource scheduling
  - Home hospice care
- Branch and check
  - Inference as projection

Essence of Benders Decomposition

- The clever idea behind classical Benders works in a **much more general setting**.
  - For problems that **simplify** when certain variables are **fixed**.
  - Use classical Benders if the resulting **subproblem** is a linear programming (LP) problem.*
  - Same idea can be extended to **any** subproblem by generalizing LP duality to **inference duality**.

* Generalized Benders allows a nonlinear programming subproblem

Essence of Benders Decomposition



Master problem

Solve for **search variables** *x*

Contains Benders cuts so far generated.

Fix search variables *x*

Add Benders cut

Subproblem

Simplified problem contains remaining variables *y*

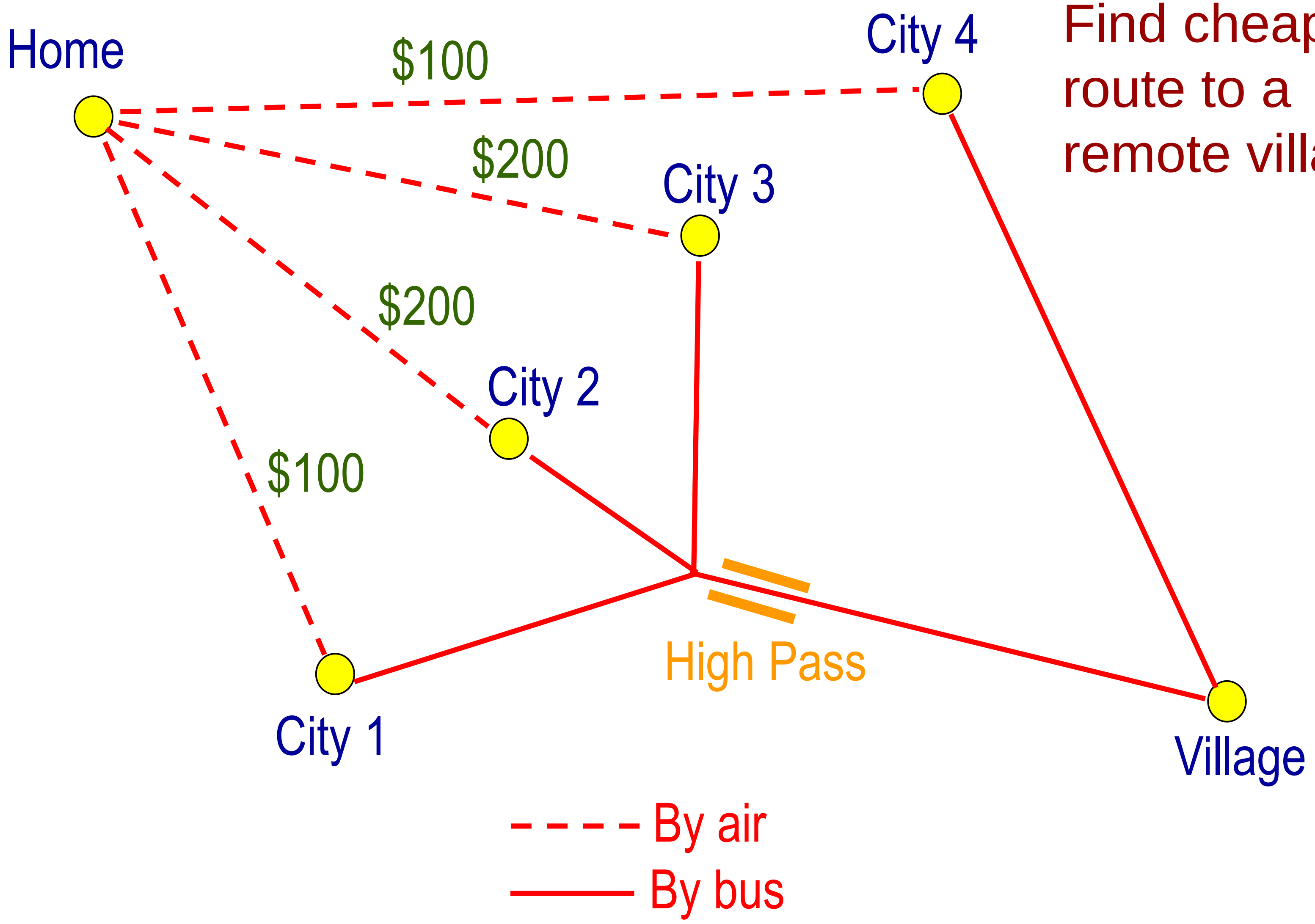
Solve **inference dual** to obtain **Benders cut** that excludes solutions no better than current one.

Essence of Benders Decomposition

- The key to generalizing Benders is generalizing the **dual**.
  - A solution of the **inference dual** is a **proof of optimality** (or infeasibility).
    - It proves a bound on the optimal value…
    - Given the values of search variables as premises.
  - It is an **explanation** of why the solution is optimal.
  - The **same proof** may yield a bound for **other values** of the search values.
    - This is key to obtaining Benders cuts.

Simple Example

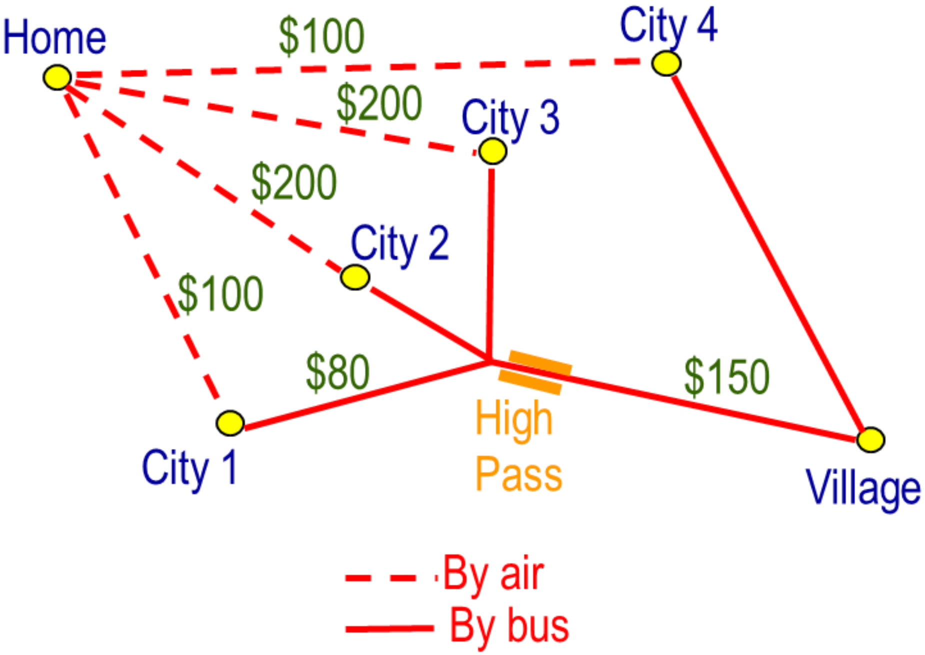
Find cheapest route to a remote village



Simple Example

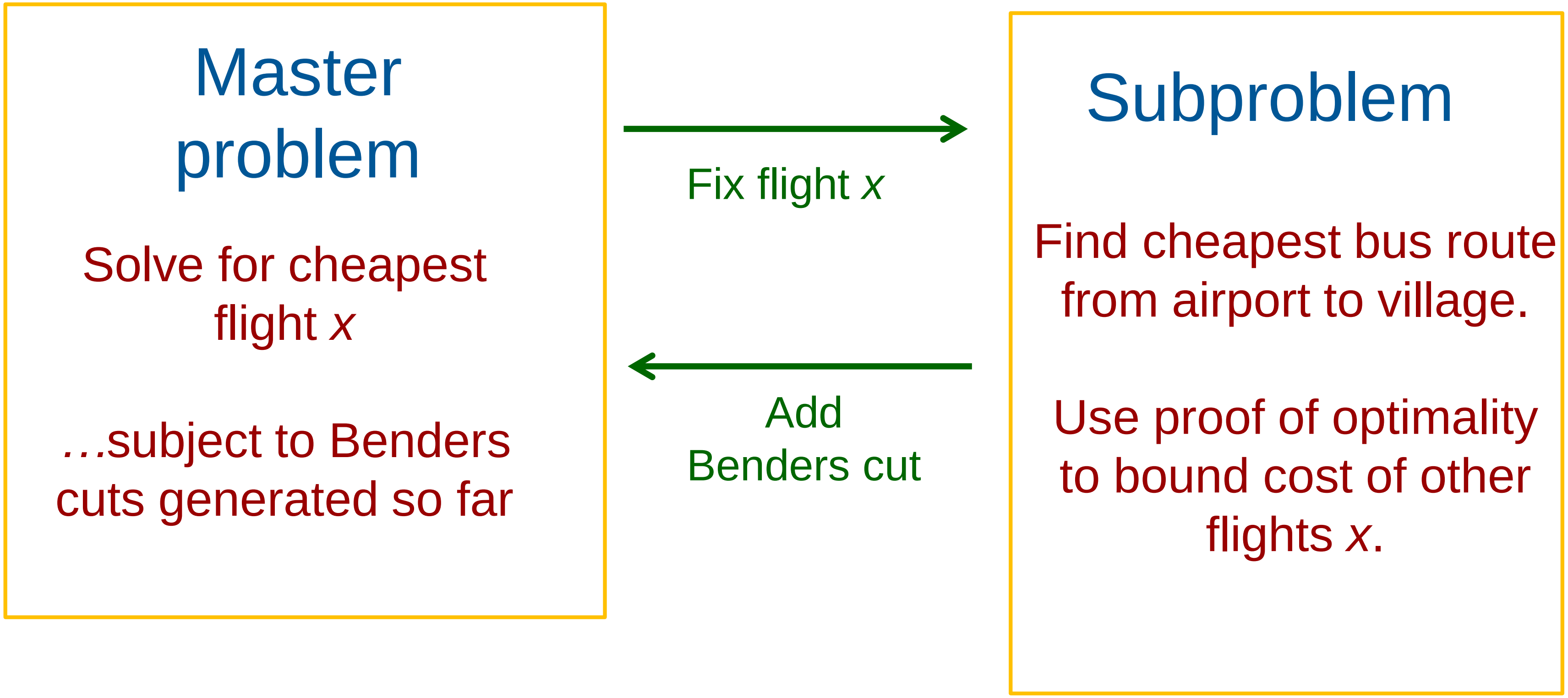
Let *x* = flight destination
y = bus route

Find cheapest route *(x,y)*



Let x = flight destination
y = bus route

Find cheapest route (x,y)



Master problem

Solve for cheapest flight x

…subject to Benders cuts generated so far

Fix flight x

Add Benders cut

Subproblem

Find cheapest bus route from airport to village.

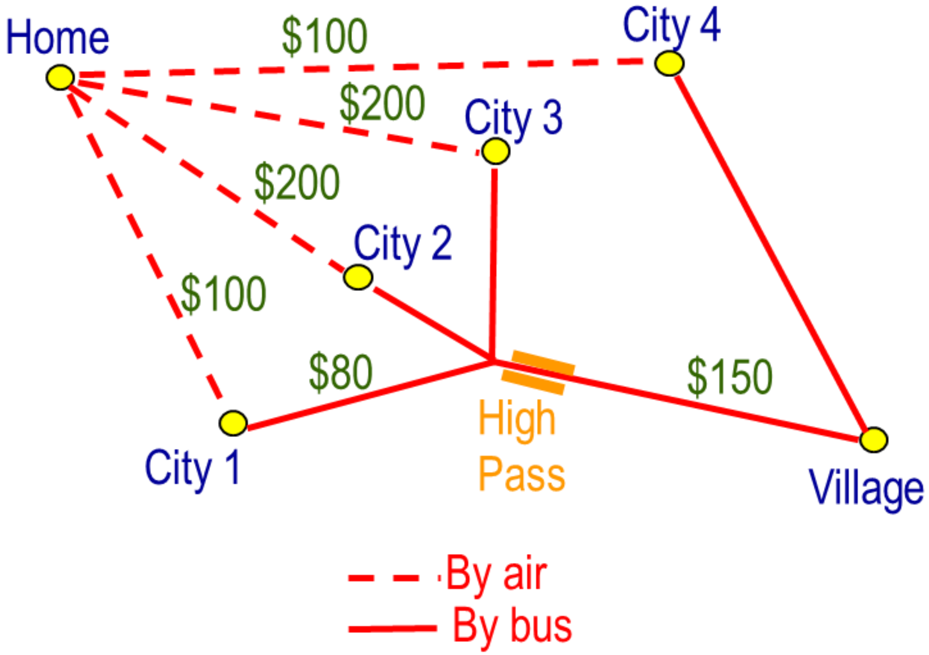
Use proof of optimality to bound cost of other flights x.

Let x = flight destination
y = bus route

Find cheapest route (x,y)

Begin with x = City 1 and pose the subproblem:

Find the cheapest route given that x = City 1.
Optimal cost is \$100 + 80 + 150 = \$330.

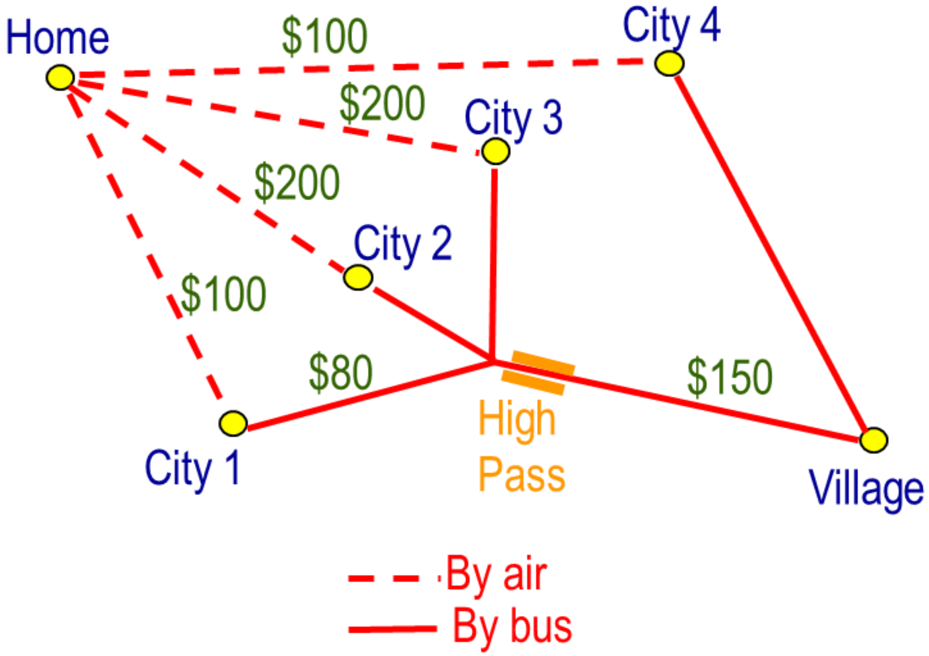


The **dual** problem of finding the optimal route is to prove optimality.

The **proof** is that the route from City 1 to the village must go through High Pass. So

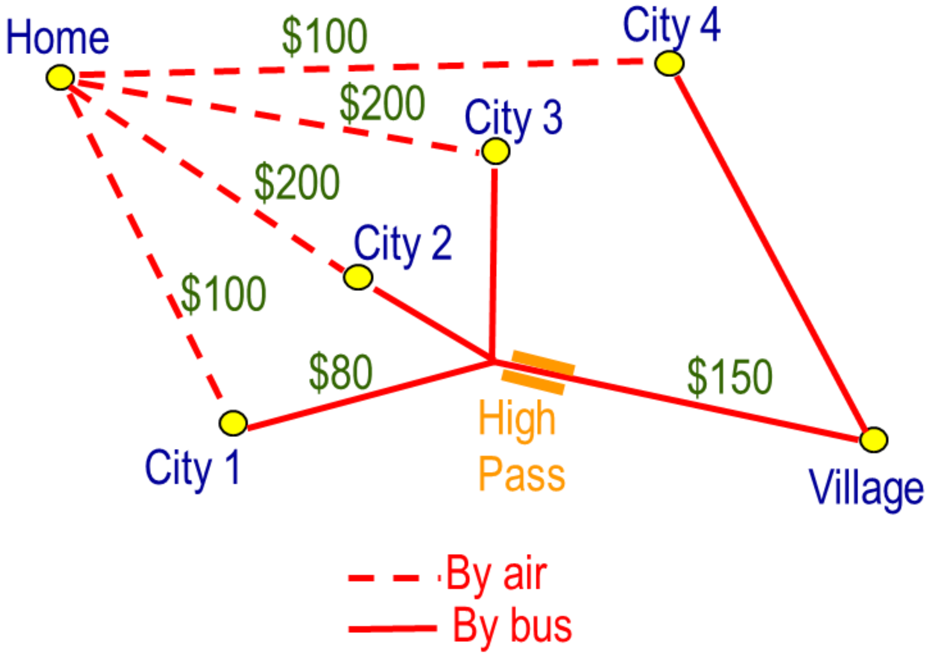
cost $\geq$ airfare + bus from city to High Pass + \$150

But **this same argument** applies to City 1, 2 or 3. This gives us the above **Benders cut**.



Specifically the Benders cut is

$$\text{cost} \geq B_{\text{City}\,1}(x) = \begin{Bmatrix} \$100+80+150 & \text{if } x = \text{City}\,1 \\ \$200+150 & \text{if } x = \text{City}\,2,3 \\ \$100 & \text{if } x = \text{City}\,4 \end{Bmatrix}$$



Now solve the **master problem**:

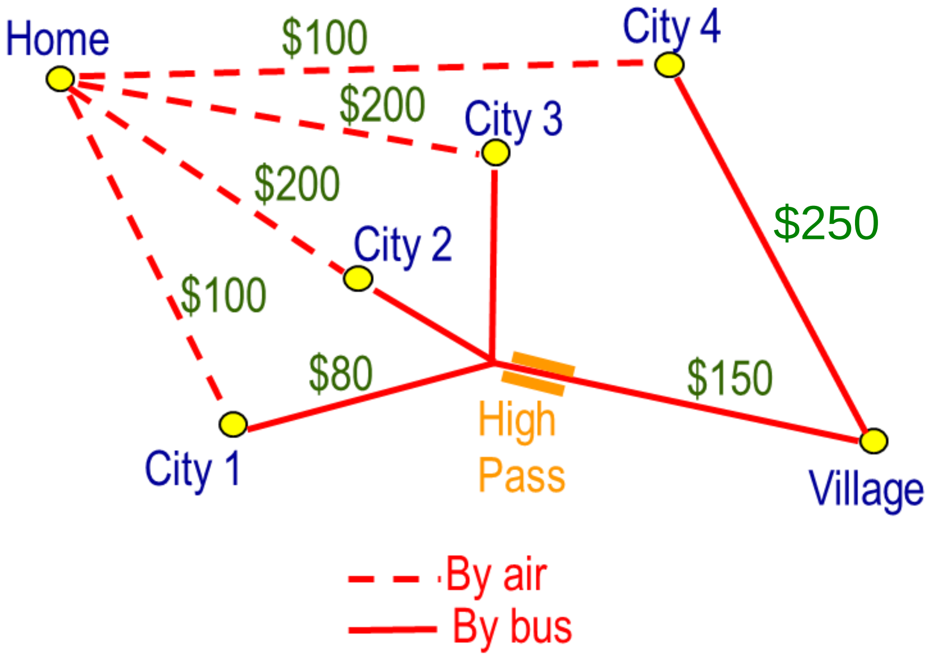
Pick the city *x* to minimize cost subject to

$$\text{cost} \geq B_{\text{City 1}}(x) = \left\{ \begin{array}{ll} \$100 + 80 + 150 & \text{if } x = \text{City 1} \\ \$200 + 150 & \text{if } x = \text{City 2,3} \\ \$100 & \text{if } x = \text{City 4} \end{array} \right\}$$

Clearly the solution is *x* = City 4, with cost \$100.

Now let *x* = City 4 and pose the **subproblem**:

Find the cheapest route given that *x* = City 4.
Optimal cost is $100 + 250 = $350.



Again solve the master problem:

Pick the city *x* to minimize cost subject to

$$\text{cost} \geq B_{\text{City }1}(x) = \begin{cases} \$100 + 80 + 150 & \text{if } x = \text{City }1 \\ \$200 + 150 & \text{if } x = \text{City }2,3 \\ \$100 & \text{if } x = \text{City }4 \end{cases}$$

$$\text{cost} \geq B_{\text{City }4}(x) = \begin{cases} \$350 & \text{if } x = \text{City }1 \\ \$0 & \text{otherwise} \end{cases}$$

The solution is *x* = City 1, with cost \$330.

Because this is equal to the value of a previous subproblem, we are done.

Logic-Based Benders

- Solve problem of the form $$\begin{aligned}&\min f(x,y)\\&(x,y)\in S\end{aligned}$$

Iteration k :

Master problem

$$\begin{aligned}&\min z\\&z \geq B_{x^i}(x),\ i \leq k-1\end{aligned}$$

Minimize cost z subject to Benders cuts

Trial value x^k that solves master

Benders cut $z \geq B_{x^k}(x)$

Subproblem

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

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

Logic-Based Benders

- In any iteration,

  master value ≤ optimal value ≤ smallest subproblem value so far

  - Continue until equality is obtained.

Master problem

$$\min z$$
$$z \geq B_{x^i}(x),\ i \leq k-1$$

Minimize cost z subject to Benders cuts

Trial value x^k that solves master

Benders cut
$$z \geq B_{x^k}(x)$$

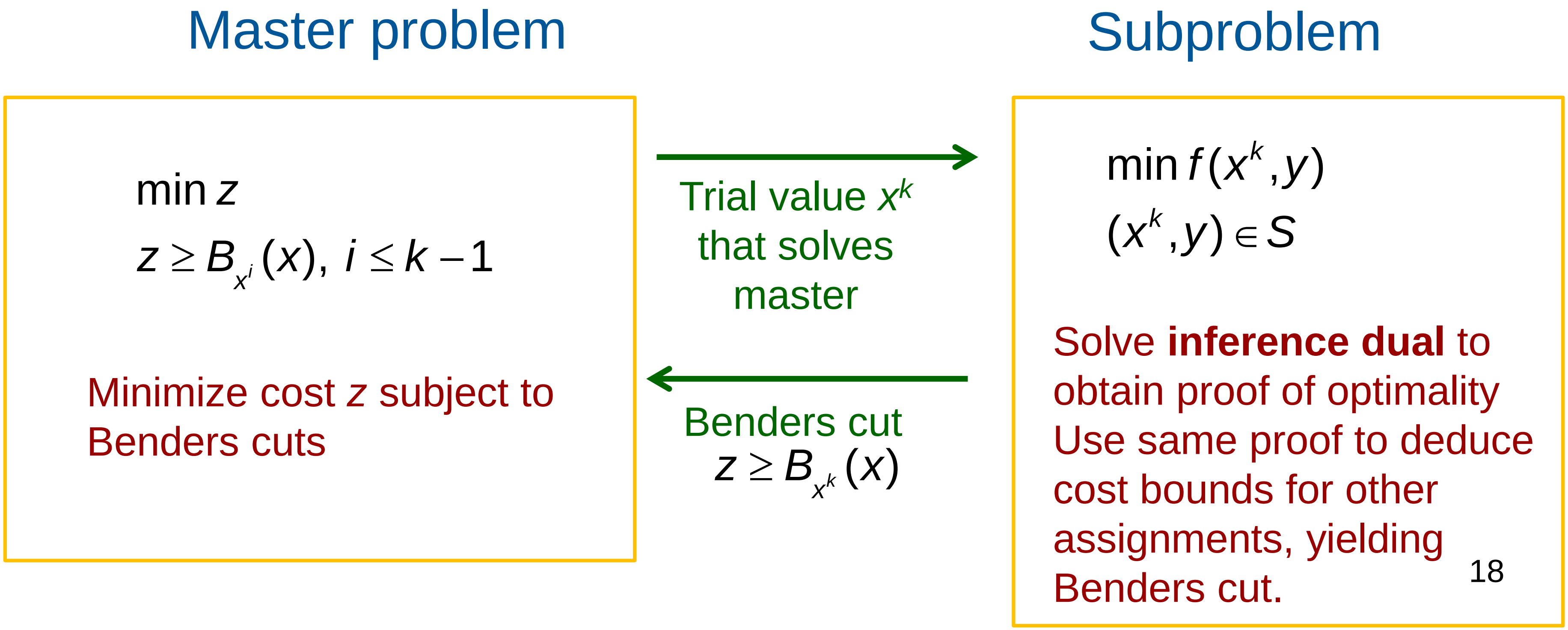
Subproblem

$$\min f(x^k, y)$$
$$(x^k, y) \in S$$

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

Logic-Based Benders

- Benders cuts describe **projection** of feasible set onto *x*
  - …if all cuts are generated.



Master problem

$$\min z$$
$$z \geq B_{x^i}(x),\ i \leq k-1$$

Minimize cost *z* subject to Benders cuts

Trial value x^k that solves master

Benders cut

$$z \geq B_{x^k}(x)$$

Subproblem

$$\min f(x^k, y)$$
$$(x^k, y) \in S$$

Solve **inference dual** to obtain proof of optimality

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

Logic-Based Benders

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

Logic-Based Benders

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

Logic-Based Benders

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

Inference Dual

- An **optimization problem** minimizes an objective function subject to constraints.
  - It is solved by searching over **values of the variables**.

- The **inference dual** finds the tightest lower bound on the objective function that is implied by the constraints.
  - It is solved by searching over **proofs**.

Inference Dual

Primal problem: optimization

$$\min f(x)$$
$$x \in S$$

Find **best** feasible solution by searching over **values of *x***.

Dual problem: Inference

$$\max v$$
$$x \in S \overset{P}{\Rightarrow} f(x) \geq v$$
$$P \in \mathcal{P}$$

Find a proof of optimal value *v** by searching over **proofs *P***.

Inference Dual

- **Weak duality** always holds:

Min value of primal problem ≥ Max value of dual problem

Difference = duality gap

Inference Dual

- **Strong duality** sometimes holds:

Min value of primal problem = Max value of dual problem

$\mathcal{P}$ is a complete proof family ⇒ Strong duality

“Complete” means that the family contains a proof for anything that is implied by the constraint set.

Classical LP Dual

Primal problem

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

Inference dual

$$\begin{aligned} &\max v \\ &\begin{pmatrix} Ax \geq b \\ x \geq 0 \end{pmatrix} \overset{P}{\Rightarrow} cx \geq v \\ &P \in \mathcal{P} \end{aligned}$$

Classical LP Dual

Primal problem

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

Inference dual

$$\begin{aligned} &\max v \\ &\begin{pmatrix} Ax \geq b \\ x \geq 0 \end{pmatrix} \overset{P}{\Rightarrow} cx \geq v \\ &P \in \mathcal{P} \end{aligned}$$

Proof family $\mathcal{P}$:

$$\begin{pmatrix} Ax \geq b \\ x \geq 0 \end{pmatrix} \overset{P}{\Rightarrow} cx \geq v$$ when $uAx \geq ub$ dominates $cx \geq v$ for some $u \geq 0$

Assuming $Ax \geq b$, $x \geq 0$ is feasible.

Classical LP Dual

Primal problem

$$\begin{aligned} &\min cx \\ &Ax \ge b \\ &x \ge 0 \end{aligned}$$

Inference dual

$$\begin{aligned} &\max v \\ &\begin{pmatrix} Ax \ge b \\ x \ge 0 \end{pmatrix} \overset{P}{\Rightarrow} cx \ge v \\ &P \in \mathcal{P} \end{aligned}$$

Proof family $\mathcal{P}$:

$$\begin{pmatrix} Ax \ge b \\ x \ge 0 \end{pmatrix} \overset{P}{\Rightarrow} cx \ge v$$

when $uAx \ge ub$ dominates $cx \ge v$ for some $u \ge 0$

$\uparrow$

$uA \le c$

$ub \ge v$

Assuming $Ax \ge b$, $x \ge 0$ is feasible.

Classical LP Dual

Primal problem

$$\begin{aligned} &\min cx \\ &Ax \ge b \\ &x \ge 0 \end{aligned}$$

Inference dual

$$\begin{aligned} &\max v \\ &\begin{pmatrix} Ax \ge b \\ x \ge 0 \end{pmatrix} \overset{P}{\Rightarrow} \; cx \ge v \\ &P \in \mathcal{P} \end{aligned}$$

Proof family $\mathcal{P}$:

$$\begin{pmatrix} Ax \ge b \\ x \ge 0 \end{pmatrix} \overset{P}{\Rightarrow} \; cx \ge v$$

when $uAx \ge ub$ dominates $cx \ge v$ for some $u \ge 0$

$$uA \le c$$

$$ub \ge v$$

Assuming $Ax \ge b$, $x \ge 0$ is feasible.

This is a **complete** inference method (due to Farkas Lemma)

Classical LP Dual

Primal problem

$$\begin{array}{l} \min cx \\ Ax \ge b \\ x \ge 0 \end{array}$$

Inference dual

$$\begin{array}{l} \max v \\ \begin{pmatrix} Ax \ge b \\ x \ge 0 \end{pmatrix} \overset{P}{\Rightarrow} cx \ge v \\ P \in \mathcal{P} \end{array} = \begin{array}{l} \max v \\ uA \le c \\ ub \ge v \\ u \ge 0 \end{array}$$

Proof family $\mathcal{P}$:

$$\begin{pmatrix} Ax \ge b \\ x \ge 0 \end{pmatrix} \overset{P}{\Rightarrow} cx \ge v$$

when $uAx \ge ub$ dominates $cx \ge v$ for some $u \ge 0$

$$\begin{array}{l} uA \le c \\ ub \ge v \end{array}$$

Assuming $Ax \ge b$, $x \ge 0$ is feasible.

This is a **complete** inference method (due to Farkas Lemma)

Classical LP Dual

Primal problem

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

Inference dual

$$\begin{aligned} &\max ub \\ &uA \leq c \\ &u \geq 0 \end{aligned} \quad = \quad \begin{aligned} &\max v \\ &uA \leq c \\ &ub \geq v \\ &u \geq 0 \end{aligned}$$

Proof family $\mathcal{P}$:

$$\begin{pmatrix} Ax \geq b \\ x \geq 0 \end{pmatrix} \overset{P}{\Rightarrow} cx \geq v$$

when

$uAx \geq ub$ dominates $cx \geq v$

for some $u \geq 0$

↑

$uA \leq c$

$ub \geq v$

Assuming $Ax \geq b$, $x \geq 0$ is feasible.

This is a **complete** inference method (due to Farkas Lemma)

Classical LP Dual

Primal problem

$$\begin{array}{l} \min cx \\ Ax \geq b \\ x \geq 0 \end{array} \quad = \quad \begin{array}{l} \max ub \\ uA \leq c \\ u \geq 0 \end{array}$$

Classical LP dual

A **strong dual** due to Farkas Lemma

…assuming $Ax \geq b$, $x \geq 0$ is feasible

Proof family $\mathcal{P}$:

$$\begin{pmatrix} Ax \geq b \\ x \geq 0 \end{pmatrix} \overset{P}{\Rightarrow} cx \geq v$$

when $uAx \geq ub$ dominates $cx \geq v$ for some $u \geq 0$

$uA \leq c$

$ub \geq v$

Assuming $Ax \geq b$, $x \geq 0$ is feasible.

This is a **complete** inference method (due to Farkas Lemma)

Inference Duals

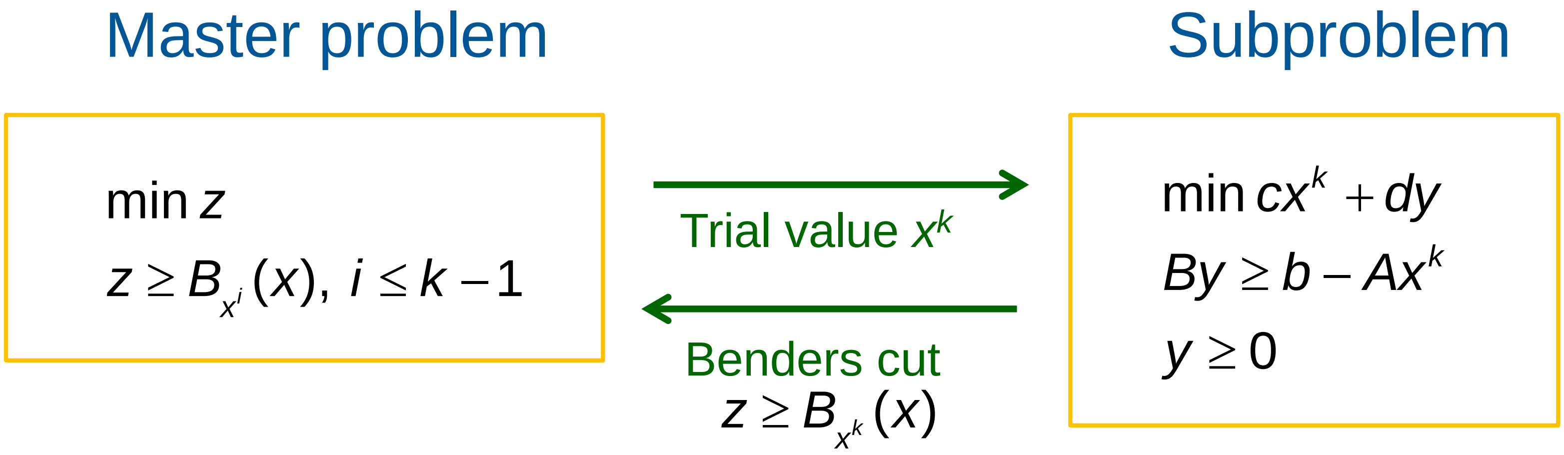
Problem	**Inference Method**	**Inference dual**
Linear programming	Linear combination + domination	Classical LP dual (strong)
Inequality constrained optimization	Linear combination + implication	Surrogate dual
Inequality constrained optimization	Linear combination + domination	Lagrangean dual
Integer programming	Chvátal-Gomory cuts	Subadditive dual (strong)

Classical Benders

- Solve problem of the form

$$\begin{aligned}&\min cx + dy\\ &Ax + By \ge b\\ &x, y \ge 0\end{aligned}$$

Iteration k :



Master problem

$$\begin{aligned}&\min z\\ &z \ge B_{x^i}(x),\ i \le k-1\end{aligned}$$

Trial value x^k

Benders cut

$$z \ge B_{x^k}(x)$$

Subproblem

$$\begin{aligned}&\min cx^k + dy\\ &By \ge b - Ax^k\\ &y \ge 0\end{aligned}$$

Classical Benders

- Solve problem of the form

$$\begin{aligned} &\min cx + dy \\ &Ax + By \geq b \\ &x, y \geq 0 \end{aligned}$$

Iteration k :

Master problem

$$\begin{aligned} &\min z \\ &z \geq B_{x^i}(x),\ i \leq k-1 \end{aligned}$$

Trial value x^k

Benders cut

$$z \geq B_{x^k}(x)$$

Subproblem

$$\begin{aligned} &\min cx^k + dy \\ &By \geq b - Ax^k \\ &y \geq 0 \end{aligned}$$

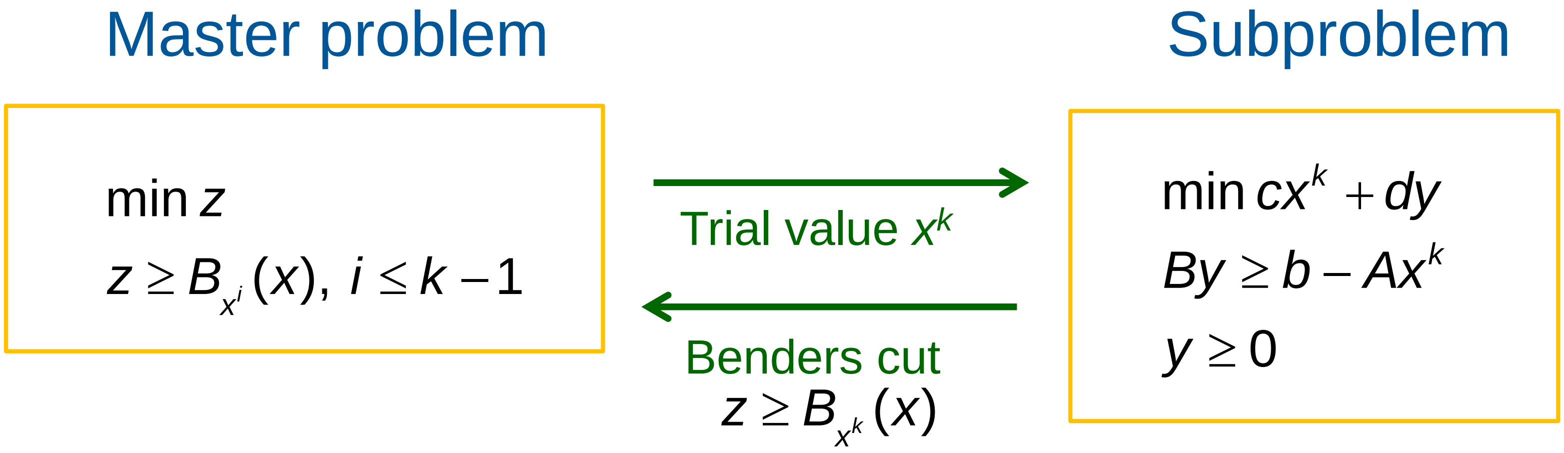
Dual solution u^k proves optimality: $u^k By \geq u^k(b - Ax^k)$ dominates $dy \geq v^*$

Classical Benders

- Solve problem of the form

$$\begin{aligned} &\min cx + dy \\ &Ax + By \geq b \\ &x, y \geq 0 \end{aligned}$$

Iteration *k* :



Master problem

$$\begin{aligned} &\min z \\ &z \geq B_{x^i}(x),\ i \leq k-1 \end{aligned}$$

Trial value x^k

Benders cut $z \geq B_{x^k}(x)$

Subproblem

$$\begin{aligned} &\min cx^k + dy \\ &By \geq b - Ax^k \\ &y \geq 0 \end{aligned}$$

Dual solution u^k proves optimality: $u^k By \geq u^k(b - Ax^k)$ dominates $dy \geq v^*$

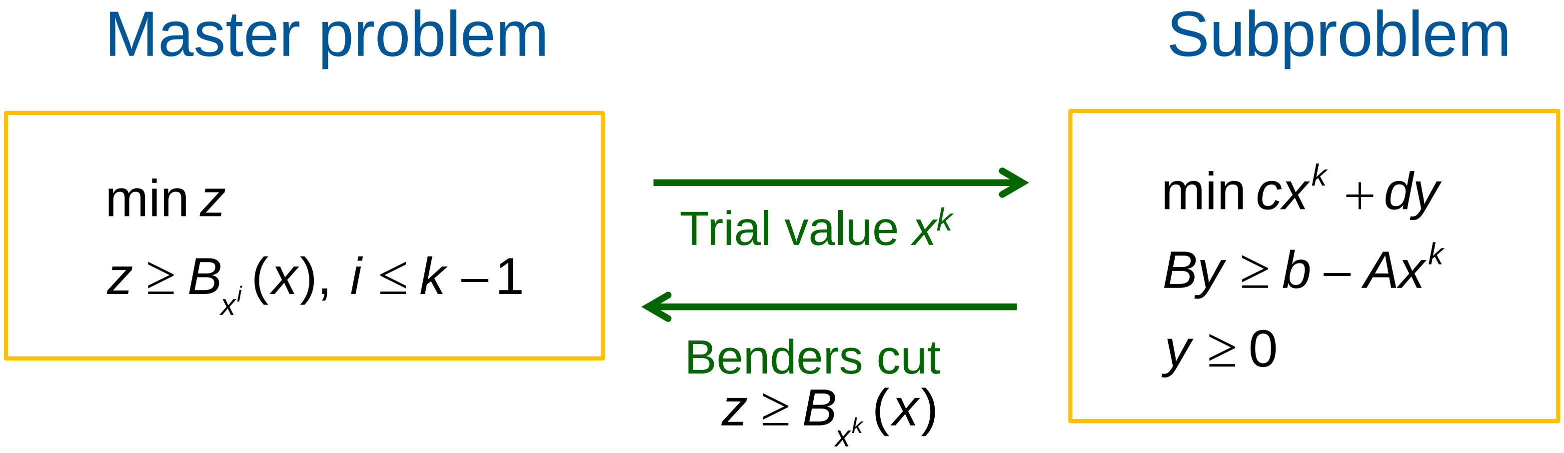
So $u^k B \leq d$ and $u^k(b - Ax^k) = v^*$

Classical Benders

- Solve problem of the form

$$\begin{aligned} &\min cx + dy \\ &Ax + By \geq b \\ &x, y \geq 0 \end{aligned}$$

Iteration k :



Master problem

$$\begin{aligned} &\min z \\ &z \geq B_{x^i}(x),\ i \leq k-1 \end{aligned}$$

Trial value x^k

Benders cut

$$z \geq B_{x^k}(x)$$

Subproblem

$$\begin{aligned} &\min cx^k + dy \\ &By \geq b - Ax^k \\ &y \geq 0 \end{aligned}$$

Dual solution u^k proves optimality: $u^k By \geq u^k(b - Ax^k)$ dominates $dy \geq v^*$

So $u^k B \leq d$ and $u^k(b - Ax^k) = v^*$

But u^k remains dual feasible for any x, so by weak duality $u^k(b - Ax) \leq v$

Classical Benders

- Solve problem of the form

$$\begin{aligned} &\min cx + dy \\ &Ax + By \geq b \\ &x,y \geq 0 \end{aligned}$$

Iteration k :

Master problem

$$\begin{aligned} &\min z \\ &z \geq B_{x^i}(x),\ i \leq k-1 \end{aligned}$$

Trial value x^k

Benders cut

$$z \geq B_{x^k}(x)$$

Subproblem

$$\begin{aligned} &\min cx^k + dy \\ &By \geq b - Ax^k \\ &y \geq 0 \end{aligned}$$

Dual solution u^k proves optimality: $u^k By \geq u^k(b - Ax^k)$ dominates $dy \geq v^*$

So $u^k B \leq d$ and $u^k(b - Ax^k) = v^*$

But u^k remains dual feasible for any x, so by weak duality $u^k(b - Ax) \leq v$

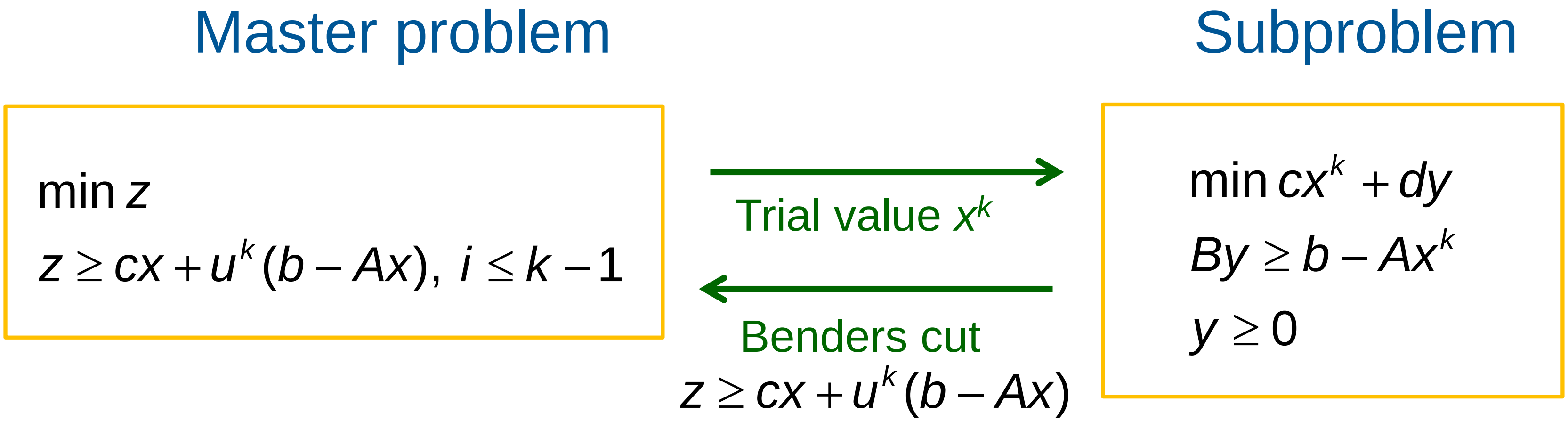
This implies $cx + u^k(b - Ax) \leq cx + v = z$

Classical Benders

- Solve problem of the form

$$\begin{aligned}&\min cx + dy\\ &Ax + By \geq b\\ &x, y \geq 0\end{aligned}$$

Iteration k :



Master problem

$$\begin{aligned}&\min z\\ &z \geq cx + u^k(b - Ax),\ i \leq k-1\end{aligned}$$

Trial value x^k

Benders cut

$$z \geq cx + u^k(b - Ax)$$

Subproblem

$$\begin{aligned}&\min cx^k + dy\\ &By \geq b - Ax^k\\ &y \geq 0\end{aligned}$$

Dual solution u^k proves optimality: $u^k By \geq u^k(b - Ax^k)$ dominates $dy \geq v^*$

So $u^k B \leq d$ and $u^k(b - Ax^k) = v^*$

But u^k remains dual feasible for any x, so by weak duality $u^k(b - Ax) \leq v$

This implies $cx + u^k(b - Ax) \leq cx + v = z$

Classical Benders

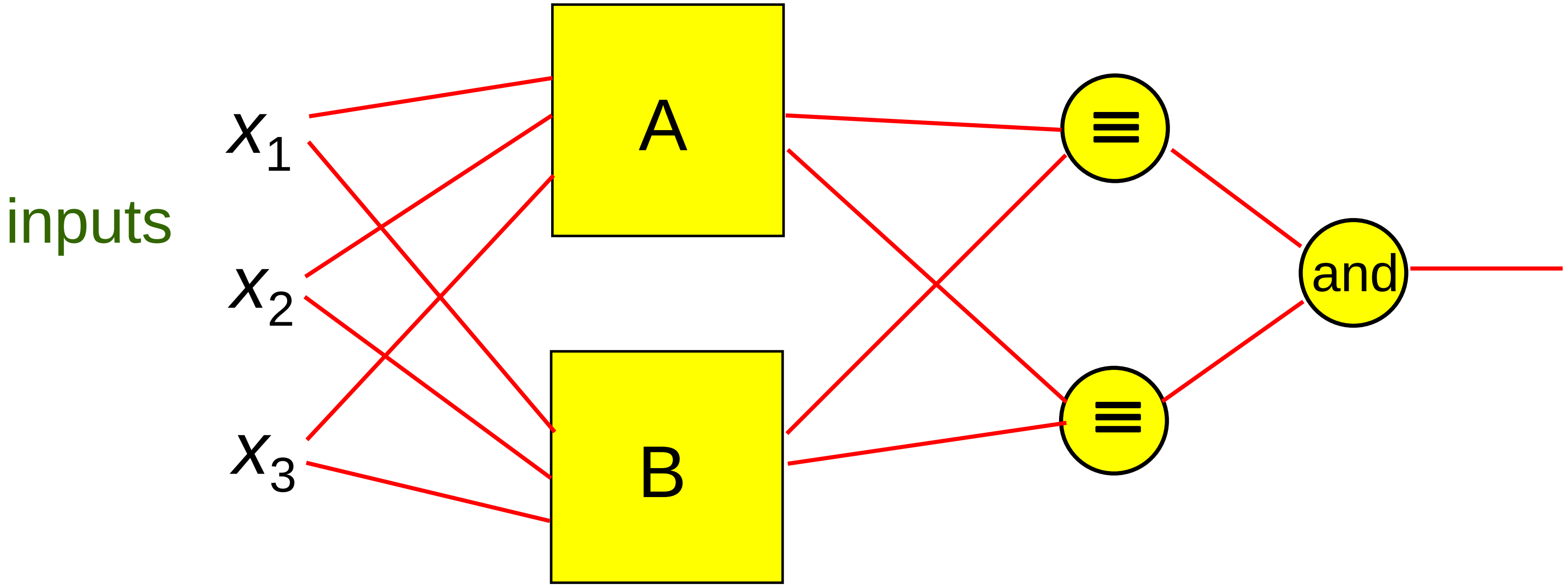
- Benders is often referred to as **row generation**.
  - as opposed to **column generation**.
- Row generation is much **more general**.
  - Applies to **any optimization problem** with constraints = rows
  - Column generation requires **columns**.
    - The constraint set must be linear ($Ax \geq b$, etc.)

Classical Benders

- Benders is often referred to as **row generation**.
  - as opposed to **column generation**.
- Row generation is much **more general**.
  - Applies to **any optimization problem** with constraints = rows
  - Column generation requires **columns**.
    - The constraint set must be linear ($Ax \geq b$, etc.)
- Benders is said to be **dual** to Dantzig-Wolfe decomposition (a form of column generation)
  - True for **classical Benders**.
  - **Not true** for logic-based Benders.
  - Logic-based Benders is **much more general** than D-W or column generation
    - D-W applies only to linear programming.

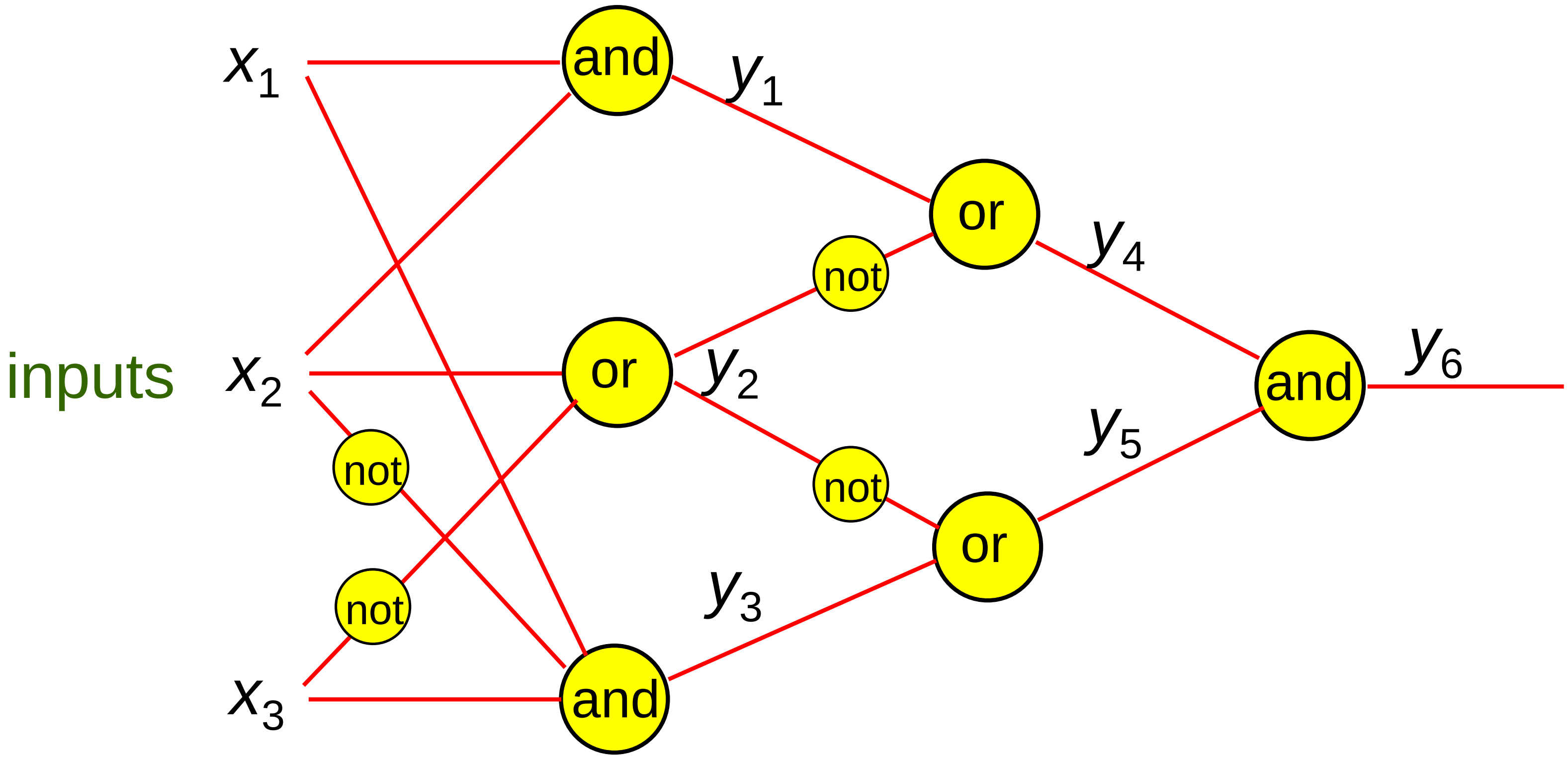
Example: Logic circuit verification

Logic circuits A and B are equivalent when the following circuit is a tautology:



The circuit is a tautology if the minimum output over all 0-1 inputs is 1.

For instance, check whether this circuit is a tautology:

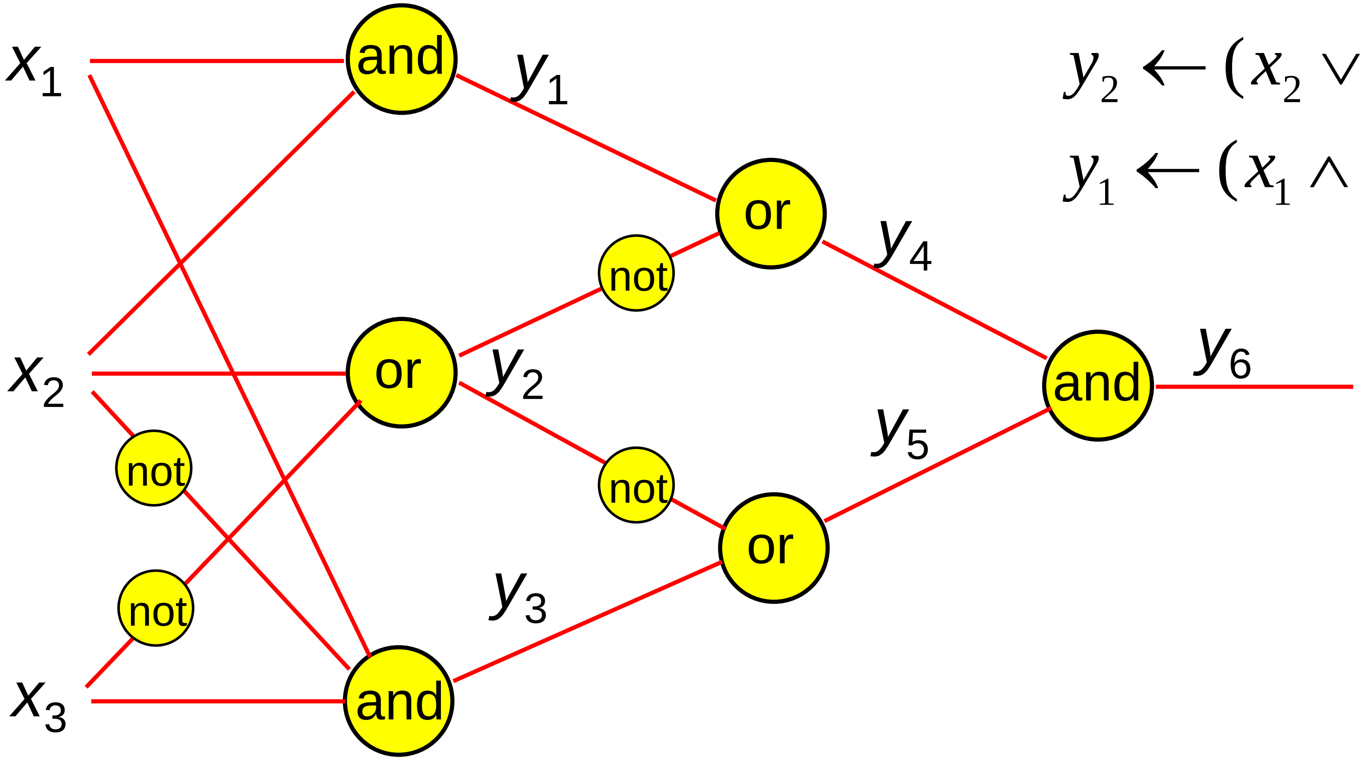


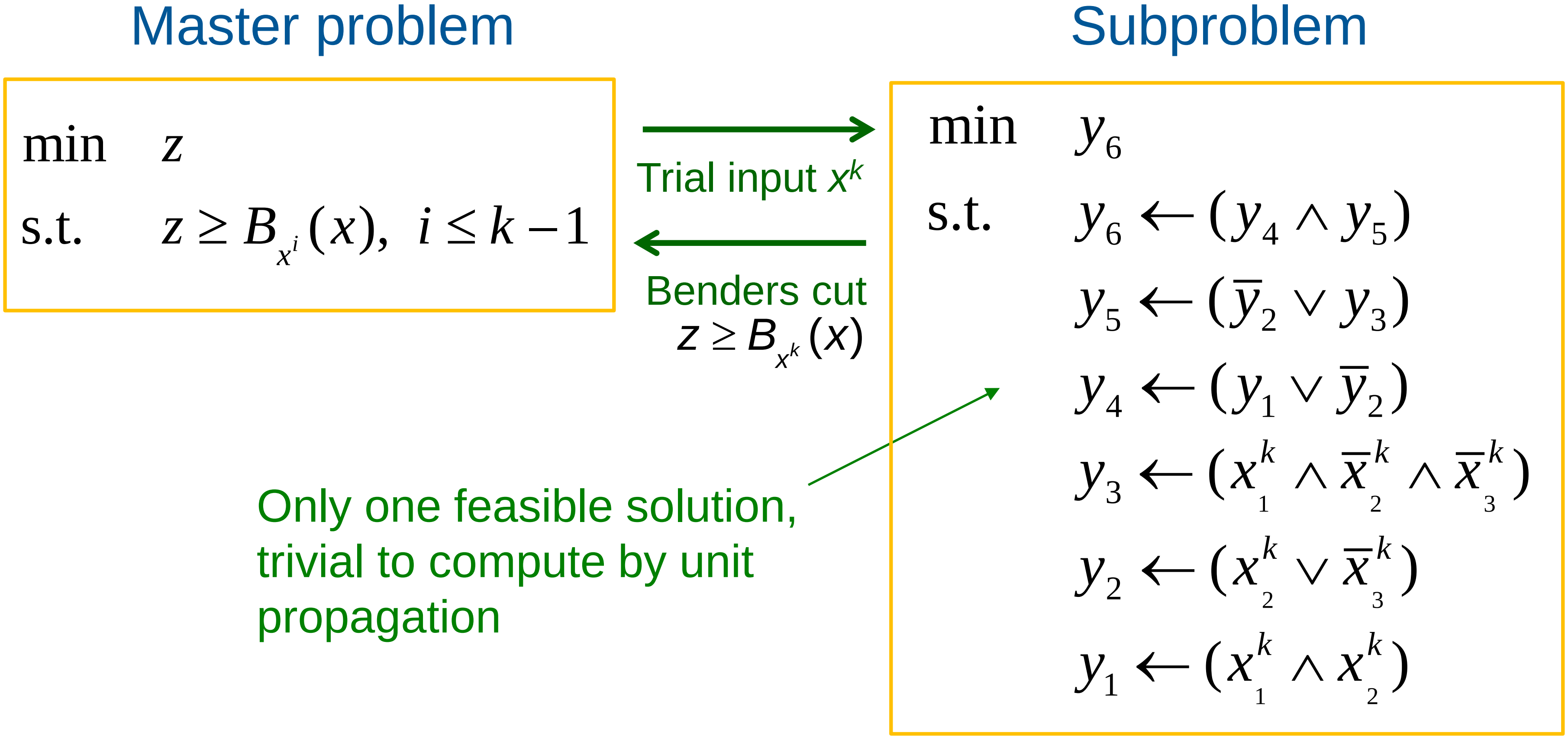
The subproblem is to minimize the output when the input *x* is fixed to a given value.

Minimum output is only feasible output, proved by unit propagation.

Formally, the problem is

$$\begin{aligned} \min \quad & y_6 \\ \text{s.t.} \quad & y_6 \leftarrow (y_4 \wedge y_5) \\ & y_5 \leftarrow (\bar{y}_2 \vee y_3) \\ & y_4 \leftarrow (y_1 \vee \bar{y}_2) \\ & y_3 \leftarrow (x_1 \wedge \bar{x}_2 \wedge \bar{x}_3) \\ & y_2 \leftarrow (x_2 \vee \bar{x}_3) \\ & y_1 \leftarrow (x_1 \wedge x_2) \end{aligned}$$





Master problem

$$\begin{aligned} \min \quad & z \\ \text{s.t.} \quad & z \geq B_{x^i}(x), \;\; i \leq k-1 \end{aligned}$$

Trial input x^k

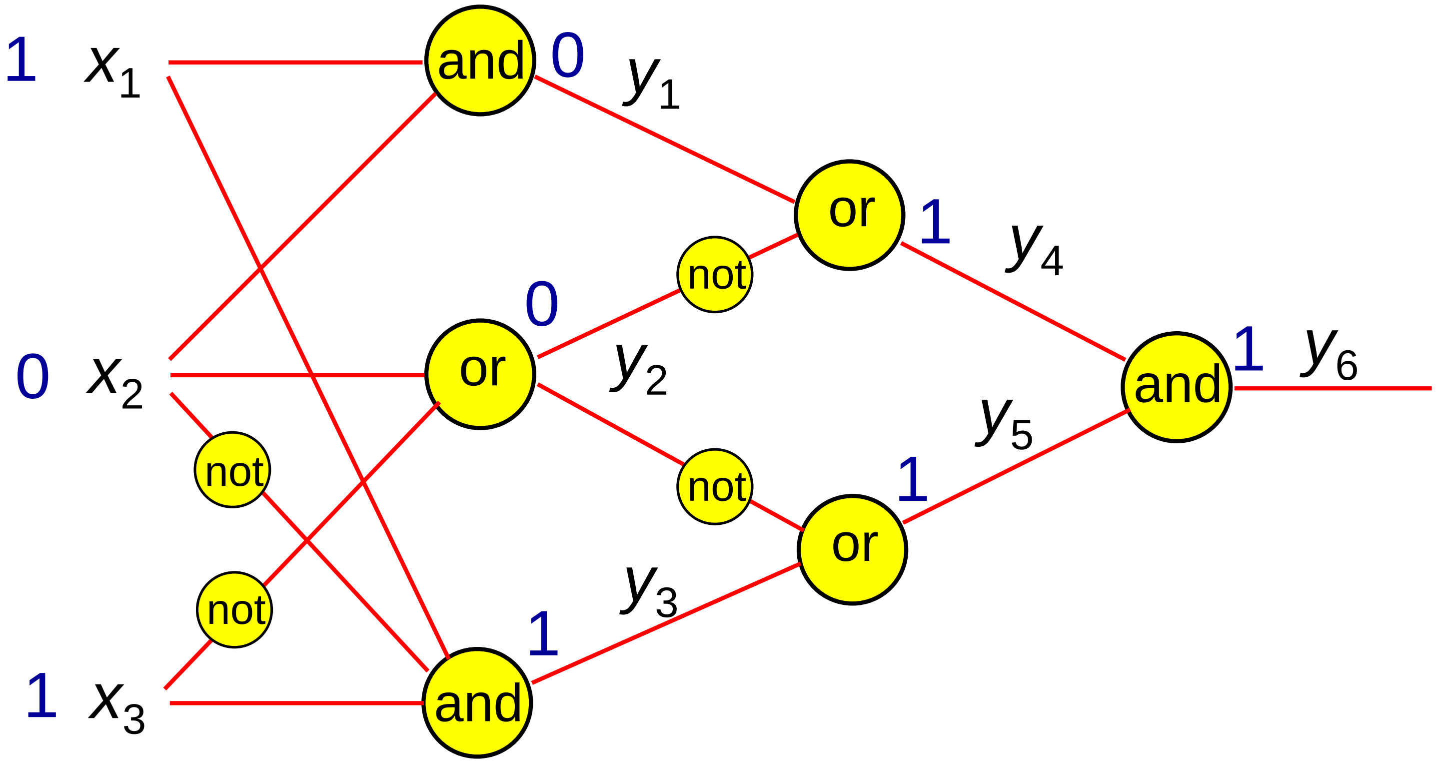
Benders cut
$z \geq B_{x^k}(x)$

Subproblem

$$\begin{aligned} \min \quad & y_6 \\ \text{s.t.} \quad & y_6 \leftarrow (y_4 \wedge y_5) \\ & y_5 \leftarrow (\bar{y}_2 \vee y_3) \\ & y_4 \leftarrow (y_1 \vee \bar{y}_2) \\ & y_3 \leftarrow (x_1^k \wedge \bar{x}_2^k \wedge \bar{x}_3^k) \\ & y_2 \leftarrow (x_2^k \vee \bar{x}_3^k) \\ & y_1 \leftarrow (x_1^k \wedge x_2^k) \end{aligned}$$

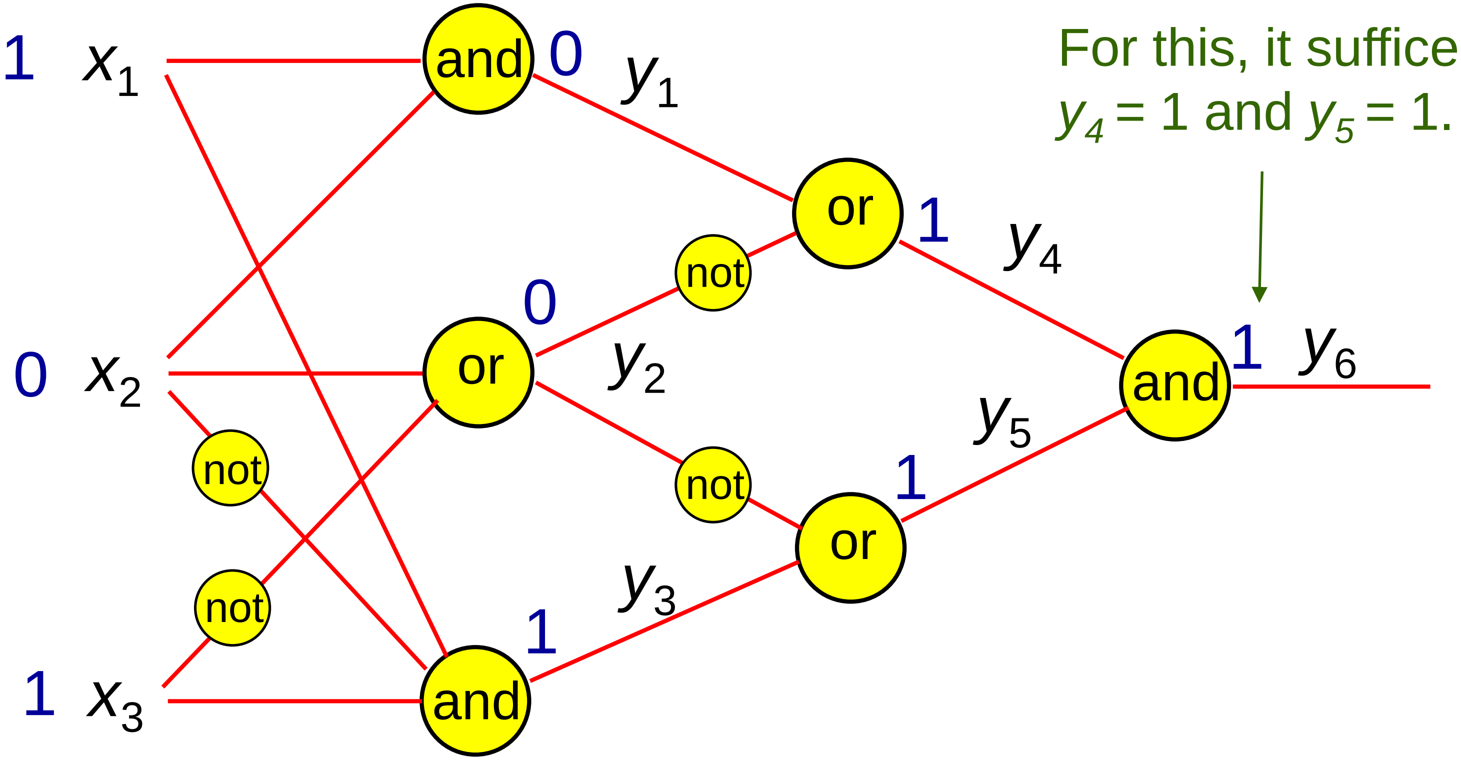
Only one feasible solution, trivial to compute by unit propagation

For example, let the inputs be *x* = (1,0,1).



To construct a Benders cut, identify some inputs x_i that are sufficient to derive an output of 1 by the same unit propagation.

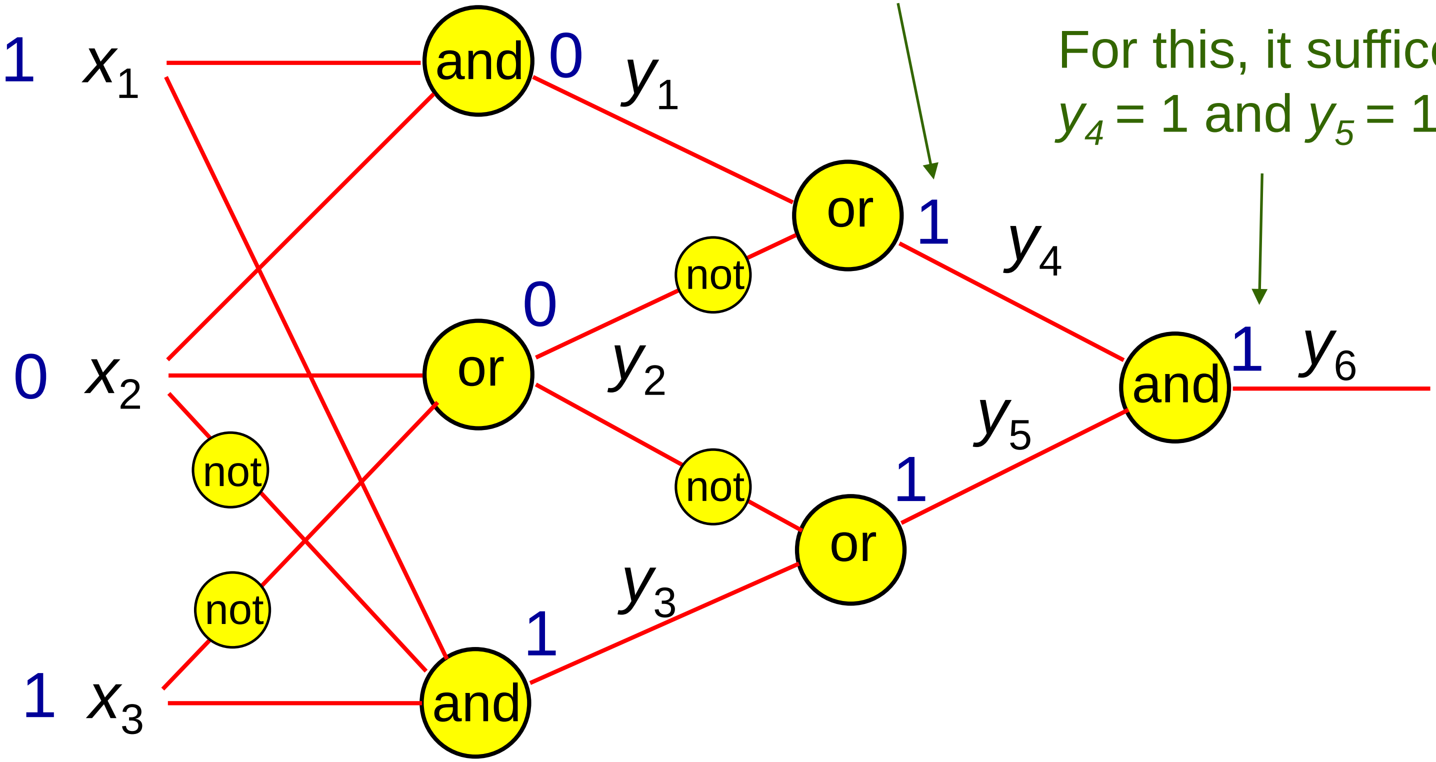
This can be done by reasoning backward.



For this, it suffices that $y_4 = 1$ and $y_5 = 1$.

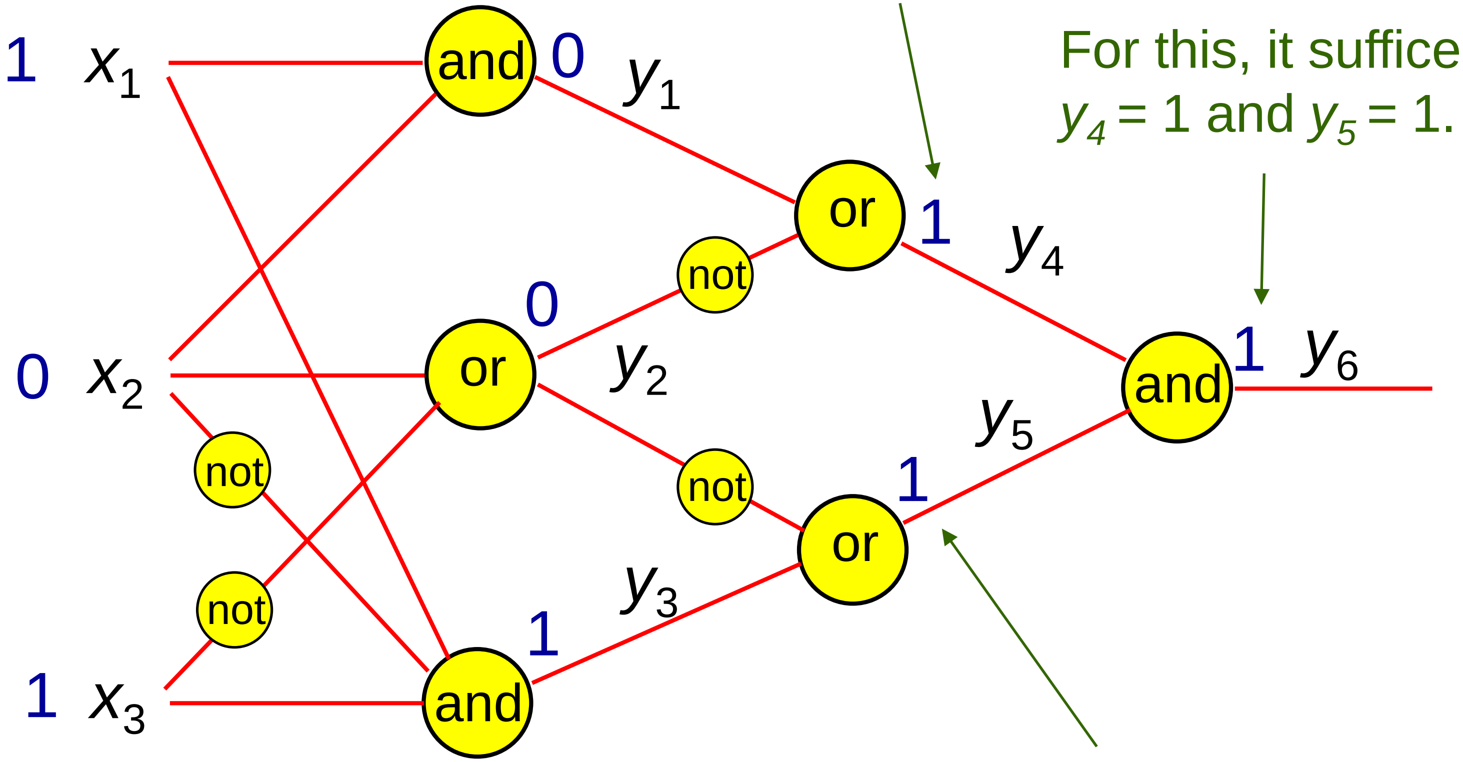
For this, it suffices that $y_2 = 0$.

For this, it suffices that $y_4 = 1$ and $y_5 = 1$.



For this, it suffices that $y_2 = 0$.

For this, it suffices that $y_4 = 1$ and $y_5 = 1$.

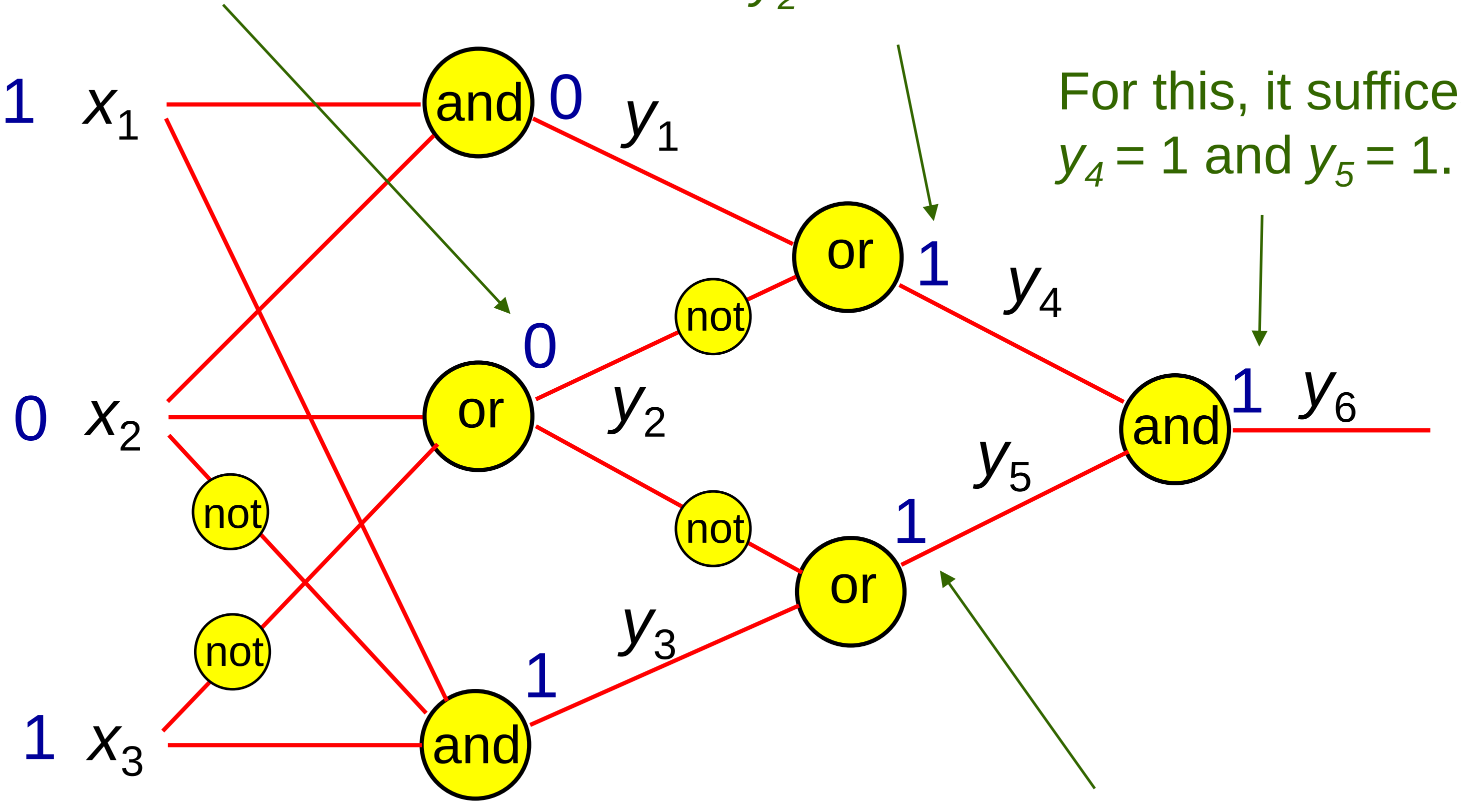


For this, it suffices that $y_2 = 0$.

For this, it suffices that $x_2 = 0$ and $x_3 = 1$.

For this, it suffices that $y_2 = 0$.

For this, it suffices that $y_4 = 1$ and $y_5 = 1$.



For this, it suffices that $y_2 = 0$.

So, Benders cut is $z \geq \bar{x}_2 \wedge x_3$

Now solve the master problem

$$\begin{array}{ll} \min & z \\ \text{s.t.} & z \geq \bar{x}_2 \wedge x_3 \end{array}$$

One solution is $(x_1, x_2, x_3) = (1, 0, 0), \quad z = 0$

This produces output 0 in the next subproblem, at which point master and subproblem values converge.

Since minimum output is 0, circuit is not a tautology.

Now solve the master problem

$$\begin{aligned} \min \quad & z \\ \text{s.t.} \quad & z \geq \bar{x}_2 \wedge x_3 \end{aligned}$$

One solution is $(x_1, x_2, x_3) = (1, 0, 0), \quad z = 0$

This produces output 0 in the next subproblem, at which point master and subproblem values converge.

Since minimum output is 0, circuit is not a tautology.

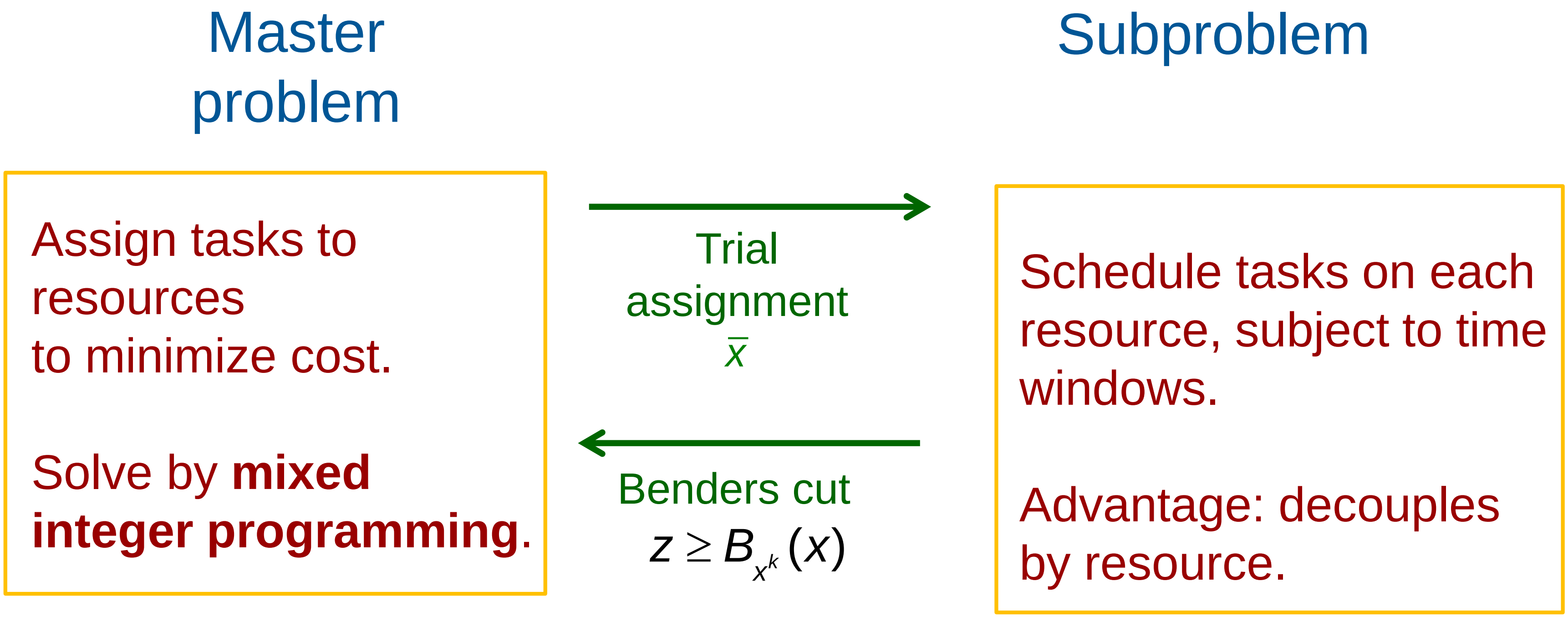
Note: This can also be solved by classical Benders. The subproblem can be written as an LP (a Horn-SAT problem).

Example: Planning & Scheduling

- Assign tasks to resources.
- Schedule tasks assign to each resource
  - Subject to time windows
  - No overlap (disjunctive scheduling)
- Appropriate objective
  - Min assignment cost
  - Min makespan
  - Min number of late tasks
  - Min total tardiness

Example: Planning & Scheduling

- Assign tasks in master, schedule in subproblem.
  - Can combine **mixed integer programming** and **constraint programming**



Master problem

Assign tasks to resources
to minimize cost.

Solve by **mixed integer programming**.

Trial assignment
$\bar{x}$

Benders cut
$z \geq B_{x^k}(x)$

Subproblem

Schedule tasks on each resource, subject to time windows.

Advantage: decouples by resource.

Example: Planning & Scheduling

- Objective function
  - Suppose cost is based on **task assignment only**.

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

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

Example: Planning & Scheduling

- Objective function
  - Suppose cost is based on **task assignment only**.

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

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

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

Example: Planning & Scheduling

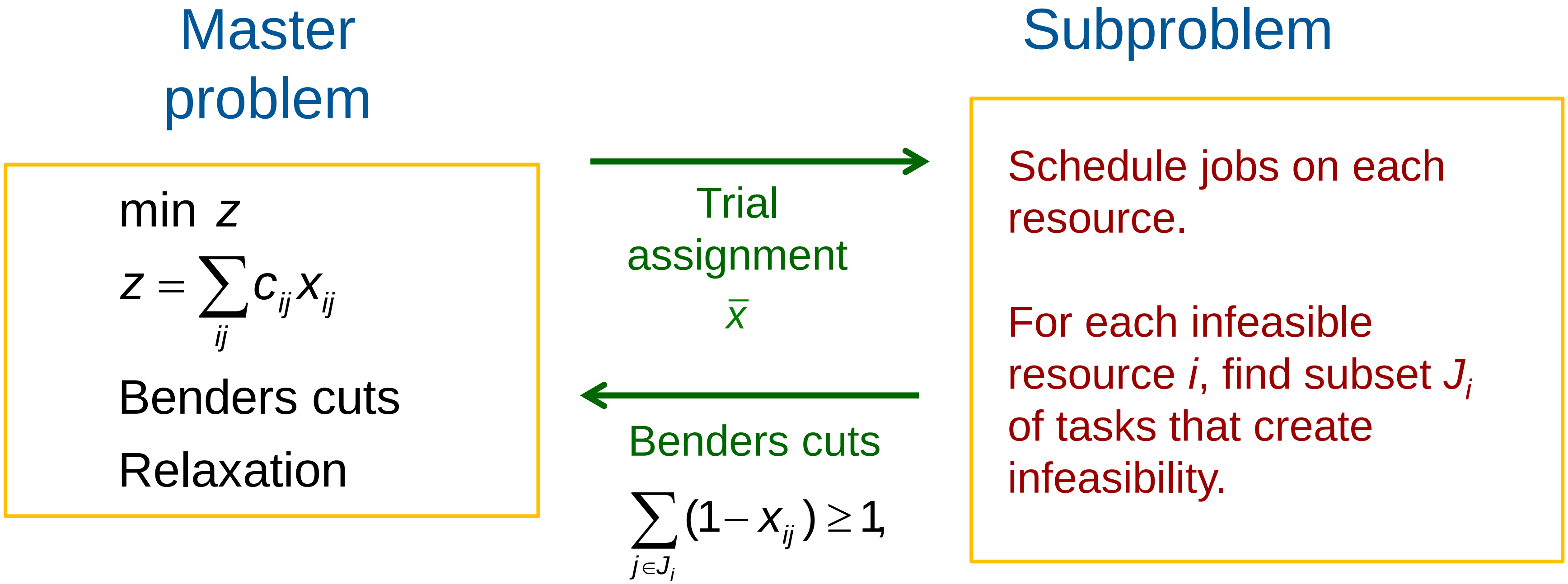
- Time window relaxation
  - For well-chosen time intervals [a,b],

$$\sum_{j \in J(a,b)} p_{ij} x_{ij} \le b - a, \quad \text{all } i$$

  - p_{ij} = processing time of task *j* on resource *i*
  - $J(a,b)$ = { tasks with time windows in [a,b] }

Example: Planning & Scheduling

- Resulting Benders decomposition:



Master problem

min z

$$z = \sum_{ij} c_{ij} x_{ij}$$

Benders cuts

Relaxation

Trial assignment

$\bar{x}$

Benders cuts

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

Subproblem

Schedule jobs on each resource.

For each infeasible resource *i*, find subset J_i of tasks that create infeasibility.

Terminate when subproblem is feasible.

Example: Planning & Scheduling

- Problem: We typically don't **have access** to infeasibility proof in subproblem solver.
  - So begin with simple **nogood cut** $\sum_{j \in J_i} (1 - x_{ij}) \geq 1, \quad \text{all } i$ where J_i contains all tasks assigned resource i.
  - Then **strengthen cut** by heuristically removing tasks from J_i until schedule on resource i becomes feasible.

Problem Instances

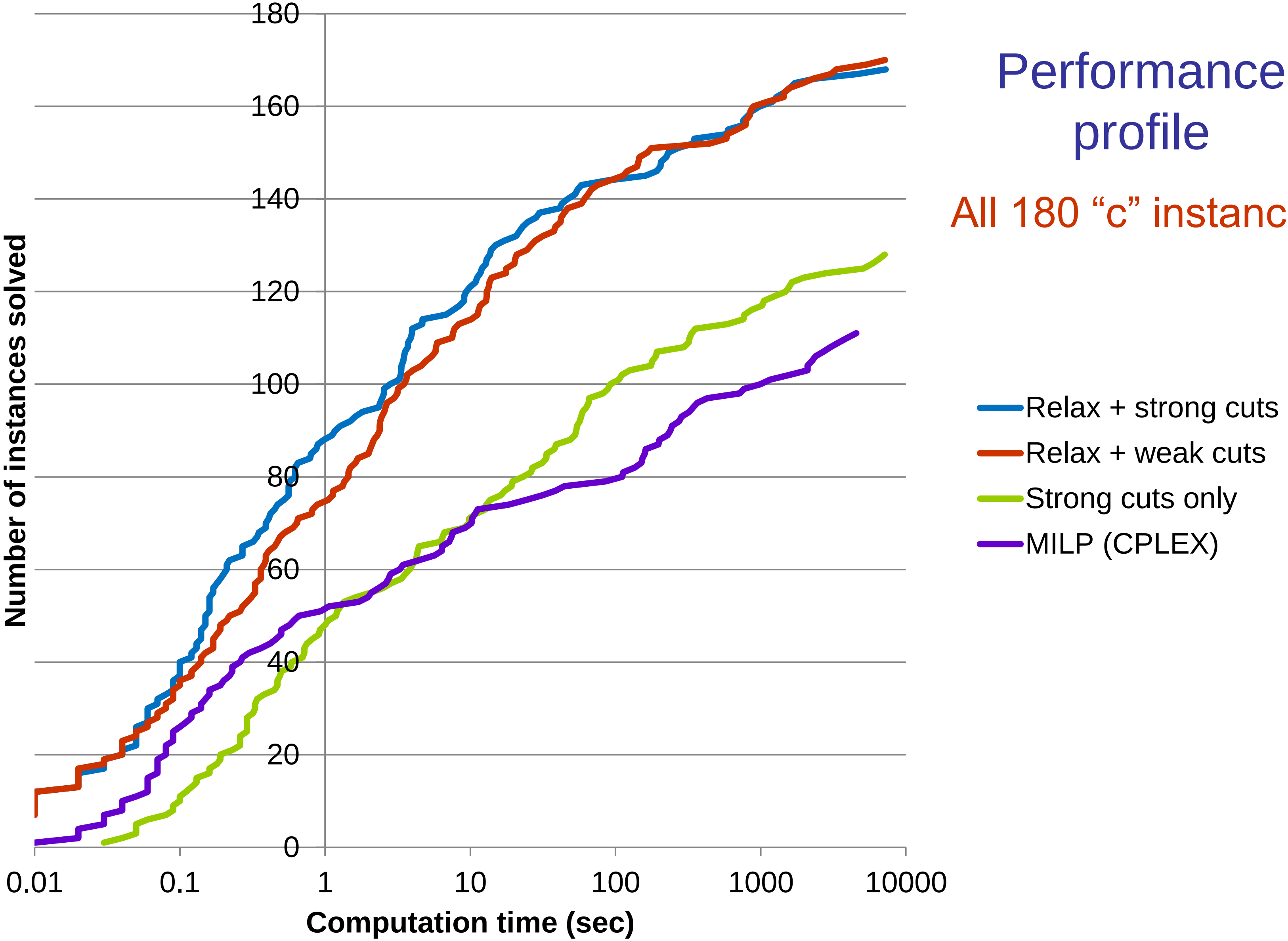
- “c” instances
  - **Hard** for LBBD.
    - Some resources much faster than others.
  - Computational bottleneck on fastest resource.
- “e” instances
  - Perhaps more realistic.
    - Resources differ by factor of ≤ 2 in processing speed.

Experimental Design

- Solve with LBBD
  - “**Strong**” Benders cuts only
    - Strengthened nogood cuts.
  - “**Weak**” cuts with subproblem **relaxation** in master.
    - Simple nogood cuts.
  - **Strong**” cuts with **relaxation**.

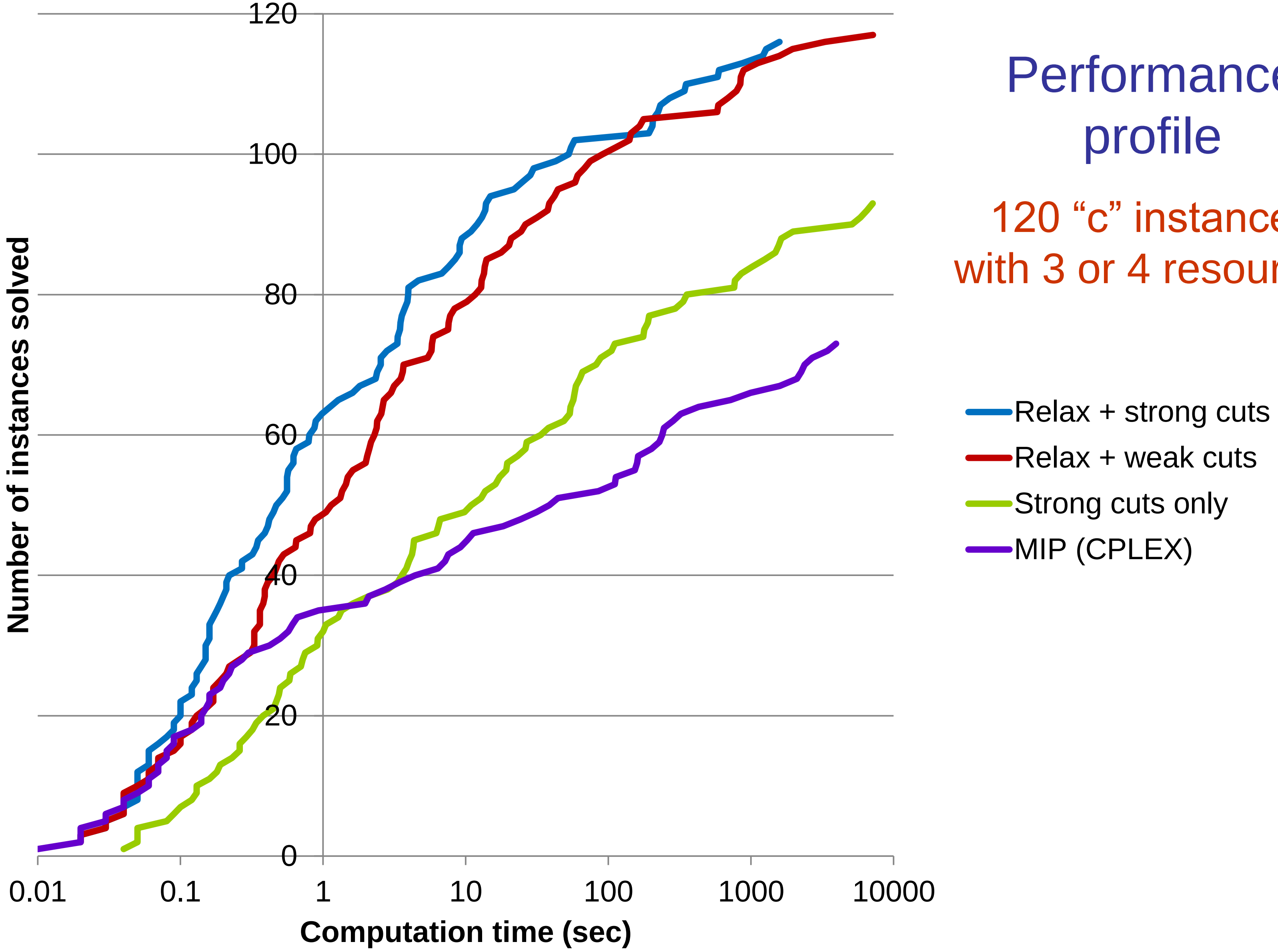
Performance profile

All 180 “c” instances



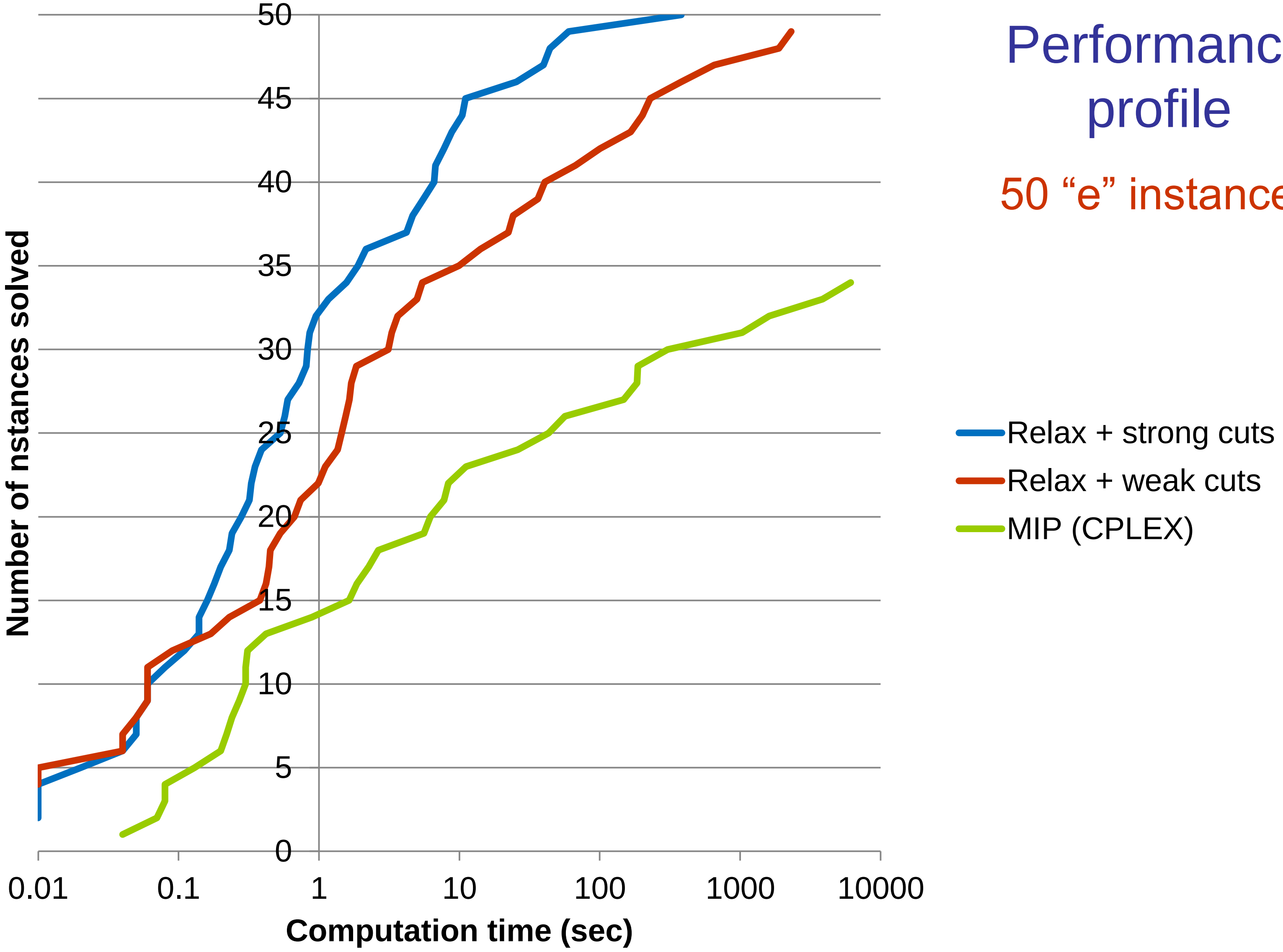
Performance profile

120 “c” instances with 3 or 4 resources



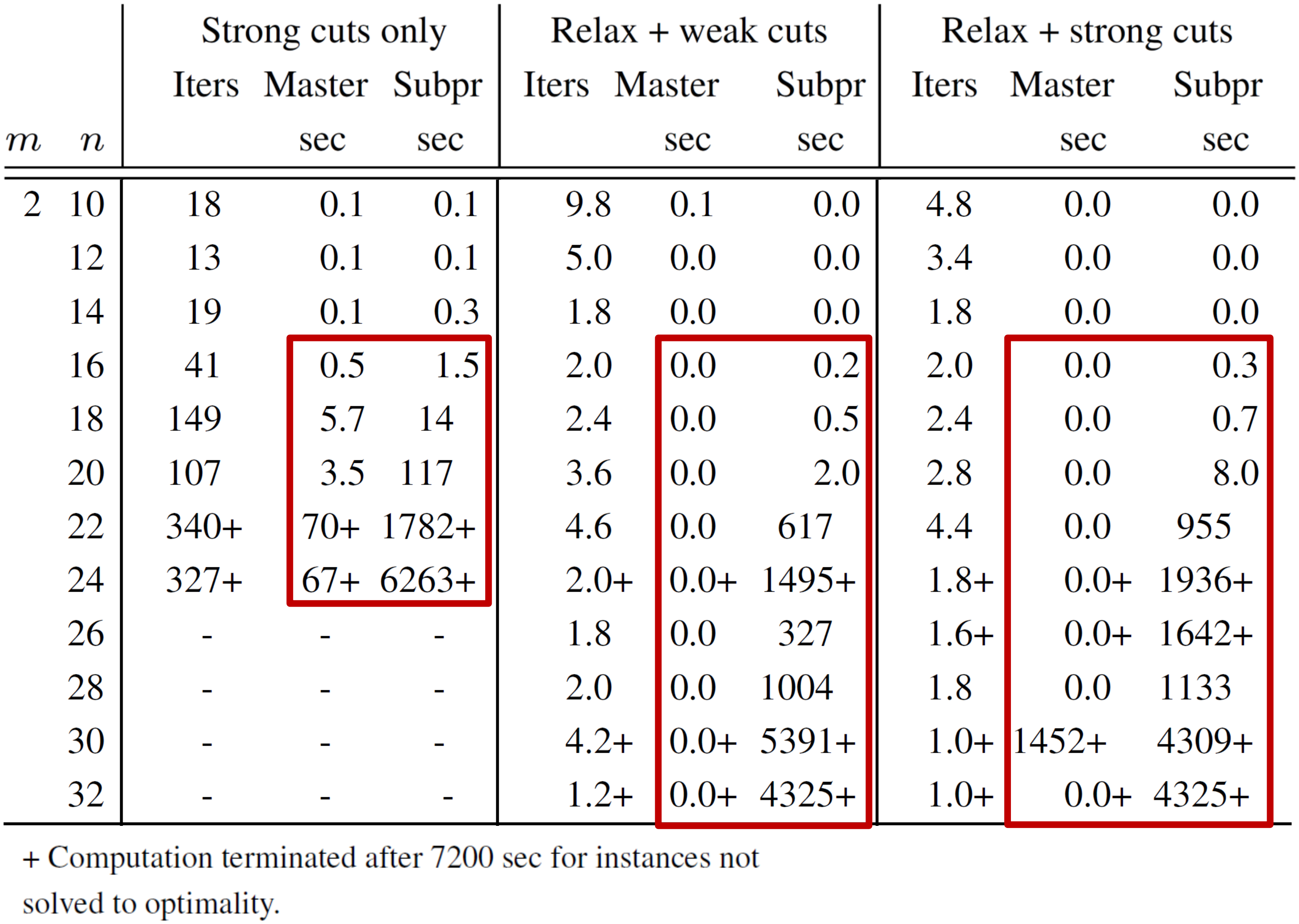
Performance profile

50 “e” instances



Severe imbalance of master and subproblem time, resulting in poorer performance for LBBD.

"c" instances, 2 resources

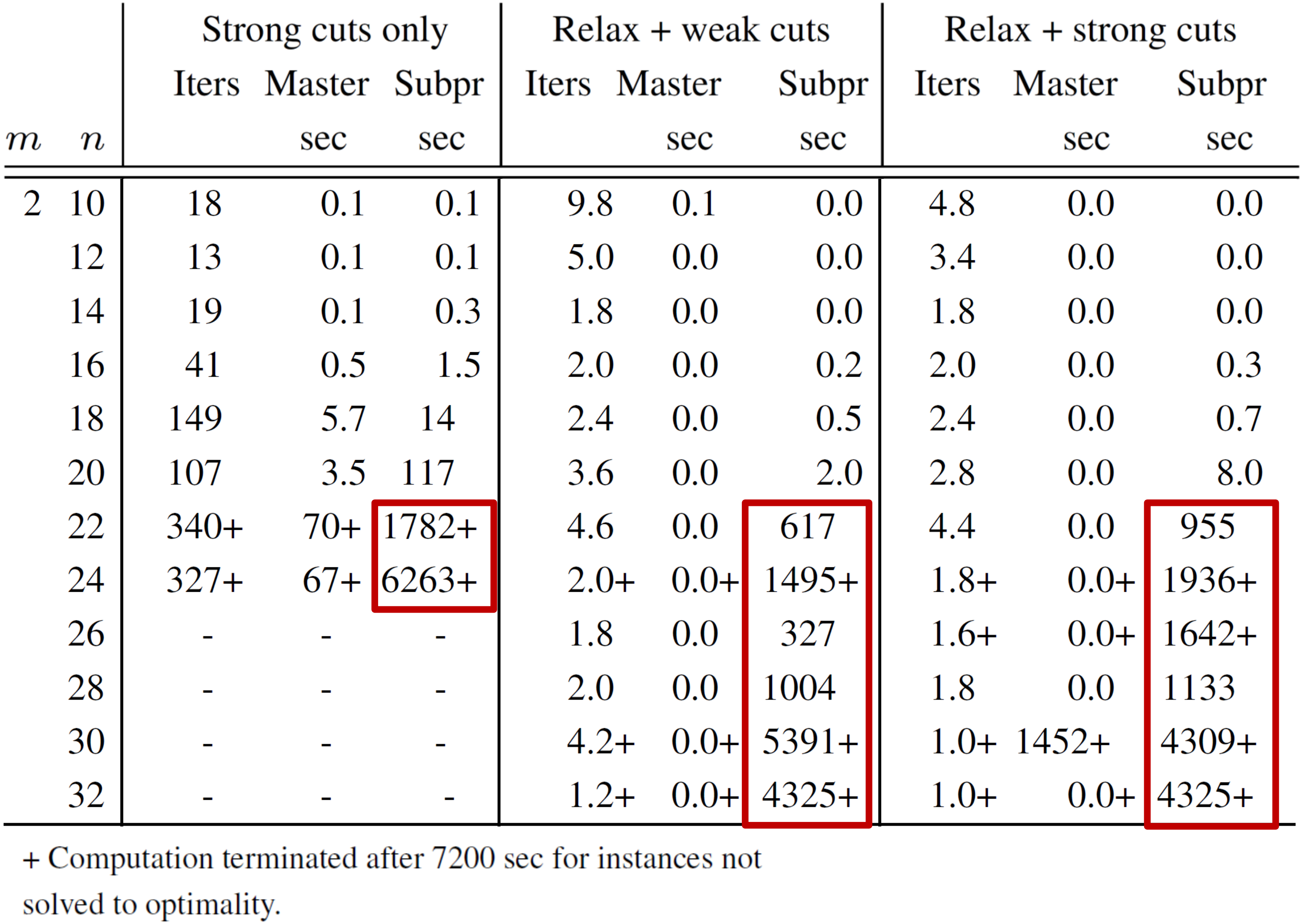


		Strong cuts only			Relax + weak cuts			Relax + strong cuts		
		Iters	Master	Subpr	Iters	Master	Subpr	Iters	Master	Subpr
m	n		sec	sec		sec	sec		sec	sec
2	10	18	0.1	0.1	9.8	0.1	0.0	4.8	0.0	0.0
	12	13	0.1	0.1	5.0	0.0	0.0	3.4	0.0	0.0
	14	19	0.1	0.3	1.8	0.0	0.0	1.8	0.0	0.0
	16	41	0.5	1.5	2.0	0.0	0.2	2.0	0.0	0.3
	18	149	5.7	14	2.4	0.0	0.5	2.4	0.0	0.7
	20	107	3.5	117	3.6	0.0	2.0	2.8	0.0	8.0
	22	340+	70+	1782+	4.6	0.0	617	4.4	0.0	955
	24	327+	67+	6263+	2.0+	0.0+	1495+	1.8+	0.0+	1936+
	26	-	-	-	1.8	0.0	327	1.6+	0.0+	1642+
	28	-	-	-	2.0	0.0	1004	1.8	0.0	1133
	30	-	-	-	4.2+	0.0+	5391+	1.0+	1452+	4309+
	32	-	-	-	1.2+	0.0+	4325+	1.0+	0.0+	4325+

+ Computation terminated after 7200 sec for instances not solved to optimality.

Subproblem blows up
when more than
10 tasks per resource
on average

“c” instances, 2 resources

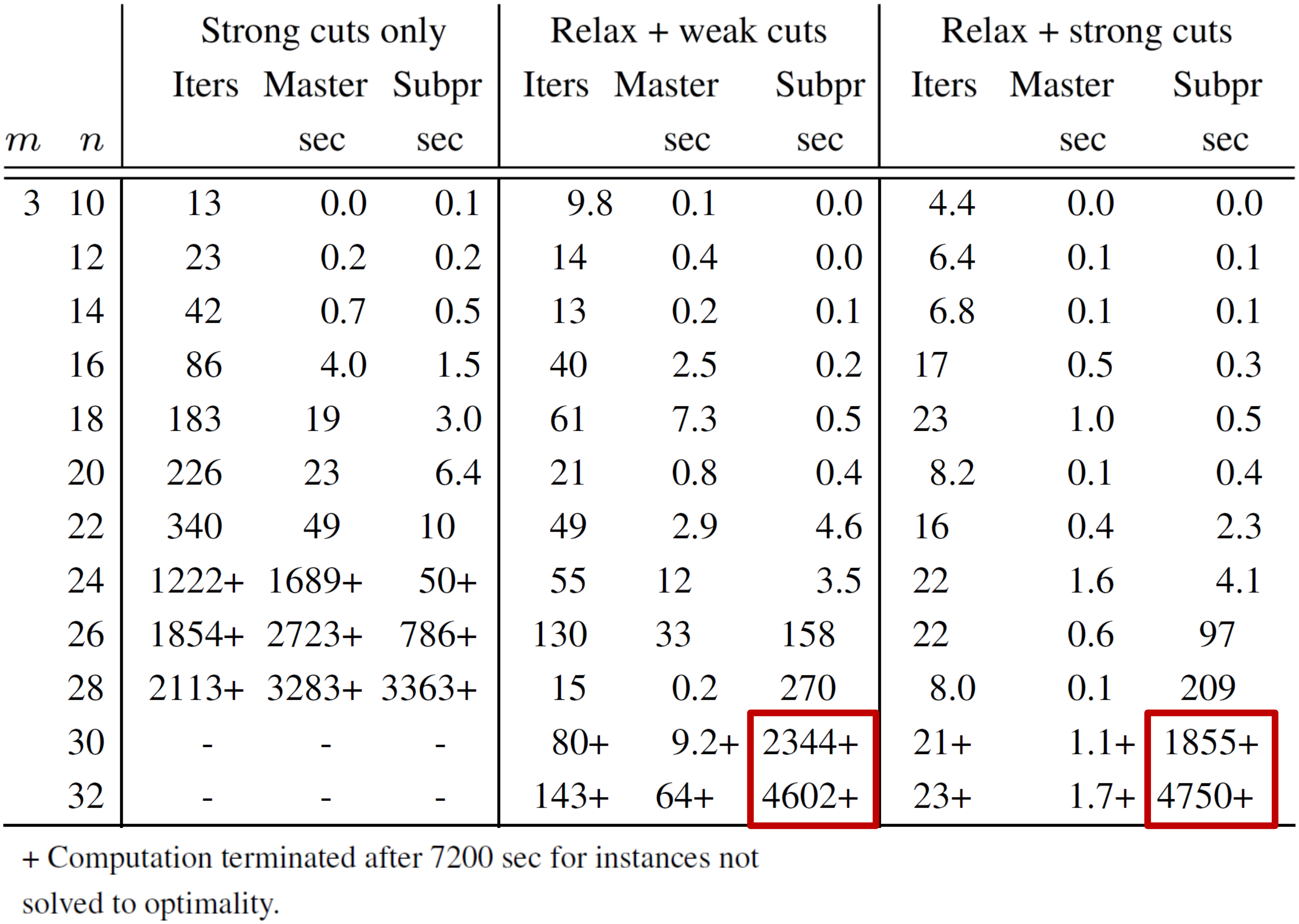


		Strong cuts only			Relax + weak cuts			Relax + strong cuts		
m	n	Iters	Master sec	Subpr sec	Iters	Master sec	Subpr sec	Iters	Master sec	Subpr sec
2	10	18	0.1	0.1	9.8	0.1	0.0	4.8	0.0	0.0
	12	13	0.1	0.1	5.0	0.0	0.0	3.4	0.0	0.0
	14	19	0.1	0.3	1.8	0.0	0.0	1.8	0.0	0.0
	16	41	0.5	1.5	2.0	0.0	0.2	2.0	0.0	0.3
	18	149	5.7	14	2.4	0.0	0.5	2.4	0.0	0.7
	20	107	3.5	117	3.6	0.0	2.0	2.8	0.0	8.0
	22	340+	70+	1782+	4.6	0.0	617	4.4	0.0	955
	24	327+	67+	6263+	2.0+	0.0+	1495+	1.8+	0.0+	1936+
	26	-	-	-	1.8	0.0	327	1.6+	0.0+	1642+
	28	-	-	-	2.0	0.0	1004	1.8	0.0	1133
	30	-	-	-	4.2+	0.0+	5391+	1.0+	1452+	4309+
	32	-	-	-	1.2+	0.0+	4325+	1.0+	0.0+	4325+

+ Computation terminated after 7200 sec for instances not solved to optimality.

Subproblem blows up when more than 10 tasks per resource **on average**

“c” instances, 3 resources



		Strong cuts only			Relax + weak cuts			Relax + strong cuts		
		Iters	Master	Subpr	Iters	Master	Subpr	Iters	Master	Subpr
m	n		sec	sec		sec	sec		sec	sec
3	10	13	0.0	0.1	9.8	0.1	0.0	4.4	0.0	0.0
	12	23	0.2	0.2	14	0.4	0.0	6.4	0.1	0.1
	14	42	0.7	0.5	13	0.2	0.1	6.8	0.1	0.1
	16	86	4.0	1.5	40	2.5	0.2	17	0.5	0.3
	18	183	19	3.0	61	7.3	0.5	23	1.0	0.5
	20	226	23	6.4	21	0.8	0.4	8.2	0.1	0.4
	22	340	49	10	49	2.9	4.6	16	0.4	2.3
	24	1222+	1689+	50+	55	12	3.5	22	1.6	4.1
	26	1854+	2723+	786+	130	33	158	22	0.6	97
	28	2113+	3283+	3363+	15	0.2	270	8.0	0.1	209
	30	-	-	-	80+	9.2+	2344+	21+	1.1+	1855+
	32	-	-	-	143+	64+	4602+	23+	1.7+	4750+

+ Computation terminated after 7200 sec for instances not solved to optimality.

Balance between master and subproblem results in superior performance

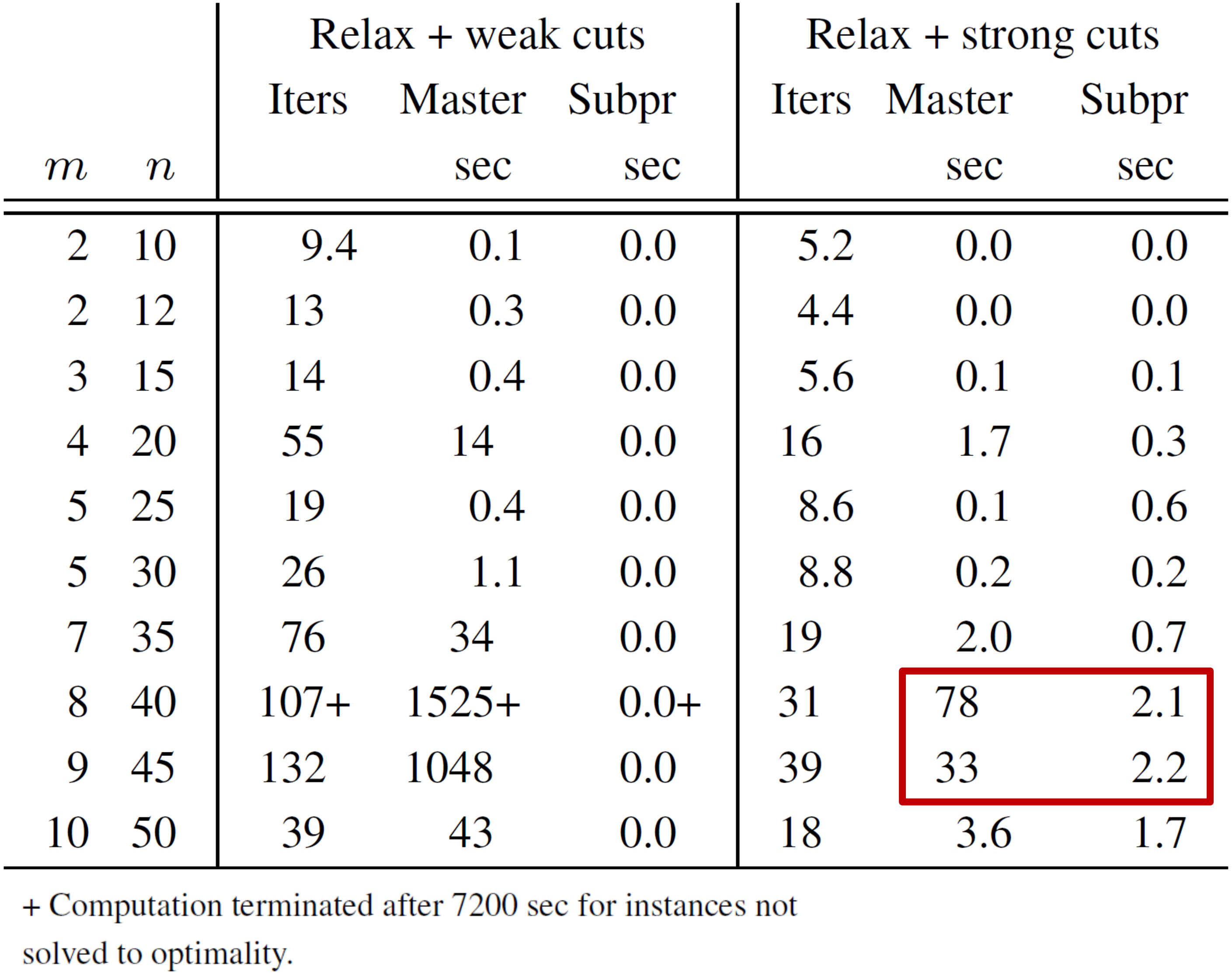
“e” instances

		Relax + weak cuts			Relax + strong cuts		
		Iters	Master	Subpr	Iters	Master	Subpr
m	n		sec	sec		sec	sec
2	10	9.4	0.1	0.0	5.2	0.0	0.0
2	12	13	0.3	0.0	4.4	0.0	0.0
3	15	14	0.4	0.0	5.6	0.1	0.1
4	20	55	14	0.0	16	1.7	0.3
5	25	19	0.4	0.0	8.6	0.1	0.6
5	30	26	1.1	0.0	8.8	0.2	0.2
7	35	76	34	0.0	19	2.0	0.7
8	40	107+	1525+	0.0+	31	78	2.1
9	45	132	1048	0.0	39	33	2.2
10	50	39	43	0.0	18	3.6	1.7

+ Computation terminated after 7200 sec for instances not solved to optimality.

Mild imbalance results in somewhat worse performance

"e" instances



		Relax + weak cuts			Relax + strong cuts		
		Iters	Master	Subpr	Iters	Master	Subpr
m	n		sec	sec		sec	sec
2	10	9.4	0.1	0.0	5.2	0.0	0.0
2	12	13	0.3	0.0	4.4	0.0	0.0
3	15	14	0.4	0.0	5.6	0.1	0.1
4	20	55	14	0.0	16	1.7	0.3
5	25	19	0.4	0.0	8.6	0.1	0.6
5	30	26	1.1	0.0	8.8	0.2	0.2
7	35	76	34	0.0	19	2.0	0.7
8	40	107+	1525+	0.0+	31	78	2.1
9	45	132	1048	0.0	39	33	2.2
10	50	39	43	0.0	18	3.6	1.7

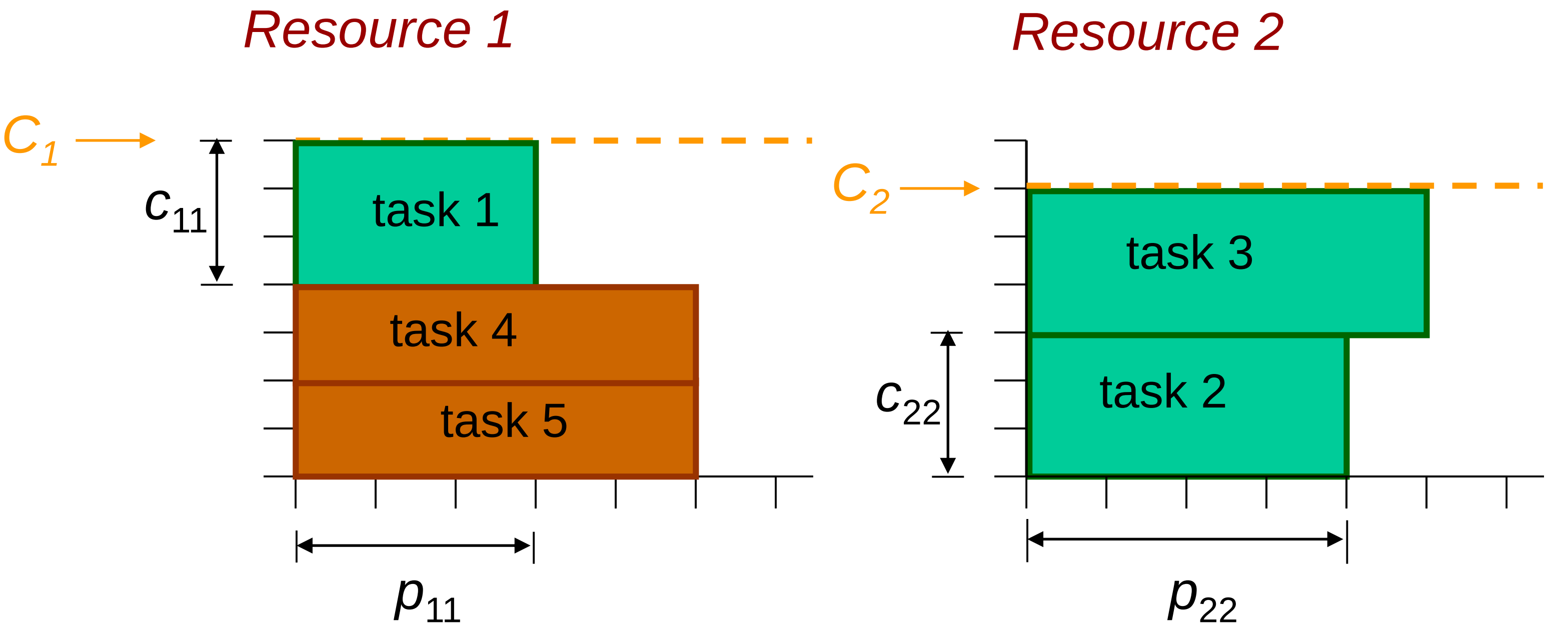
+ Computation terminated after 7200 sec for instances not solved to optimality.

Suggested Solution Strategies

- Tighter subproblem relaxations
  - Design tighter subproblem relaxations for the master
    - …**using subproblem variables**, whose values are discarded after master is solved
- Subproblem decomposition
  - Solve subproblem with LBBD when it **grows too large**.
- More dual information
  - Use subproblem solver that **reveals proof of optimality**, perhaps resulting in stronger Benders cuts.

Cumulative Scheduling Problems

p_{ij} = processing time of task *j* on resource *i*
c_{ij} = resource consumption of task *j* on resource *i*
C_i = resources available on resource *i*



Total resource consumption ≤ C_i at all times.

Min Cost Cumulative Scheduling

Master Problem: Assign tasks to resources

Formulate as MILP problem

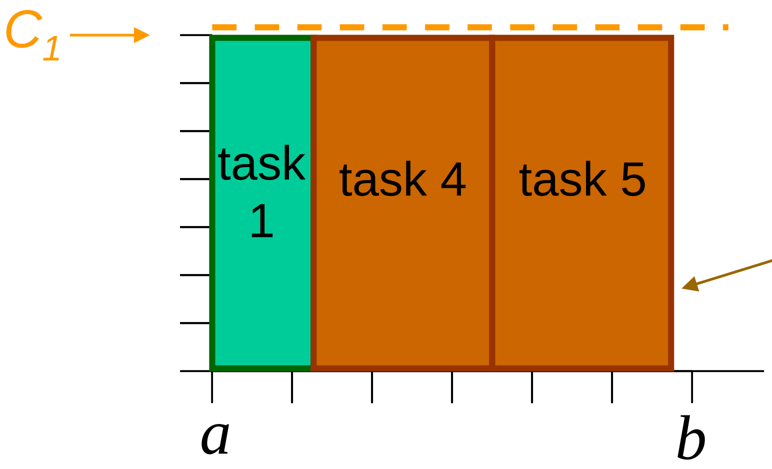
$$\min \qquad z$$

cost

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

$$\sum_{j\in J(b,a)} p_{ij}c_{ij}x_{ij} \le C_i(b-a), \quad \text{all } i, \ \text{various } [a,b]$$

Benders cuts

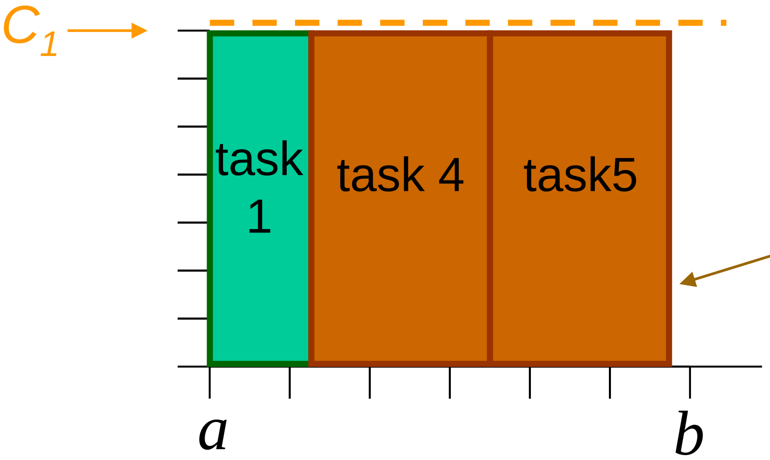


Relaxation of subproblem:
"Energy" of tasks must be at most energy available.

Min Cost Cumulative Scheduling

Benders cuts same as for disjunctive scheduling

$$\begin{aligned}
&\min && \boxed{z} \longleftarrow \text{cost} \\
&\text{subject to} && \sum_i x_{ij} = 1, \quad \text{all } j \\
& && \sum_{j\in J(b,a)} p_{ij}c_{ij}x_{ij} \le C_i(b-a), \quad \text{all } i, \ \text{various } [a,b] \\
& && \text{Benders cuts}
\end{aligned}$$



Relaxation of subproblem:
"Energy" of tasks must be at most energy available.

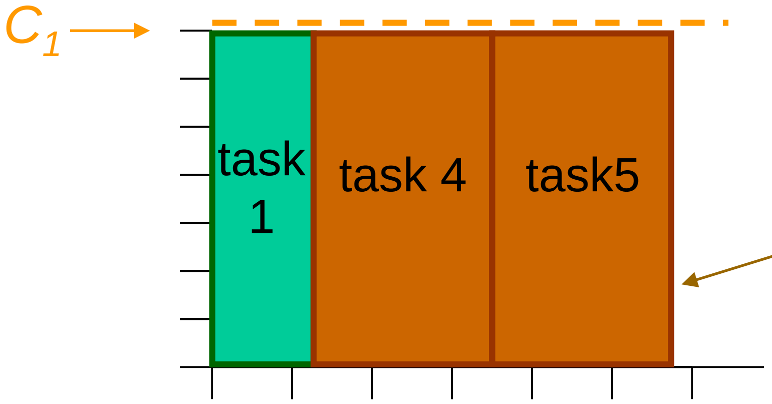
Min Makespan Cumulative Scheduling

Master Problem: Assign tasks to resources

Formulate as MILP problem

$$
\begin{array}{ll}
\text{min} & M \\
\text{subject to} & \sum_i x_{ij} = 1, \quad \text{all } j \\
 & M \geq \frac{1}{C_i} \sum_j p_{ij} c_{ij} x_{ij}, \quad \text{all } i \\
 & \text{Benders cuts}
\end{array}
$$

makespan



Relaxation of subproblem:
“Energy” of tasks provides
lower bound on makespan.

Min Makespan Cumulative Scheduling

Benders cuts are based on:

Lemma. If we remove tasks 1, … *s* from a resource, the minimum makespan on that resource is reduced by at most

$$\sum_{j=1}^{s} p_{ij} + \max_{j \le s}\{d_j\} - \min_{j \le s}\{d_j\}$$

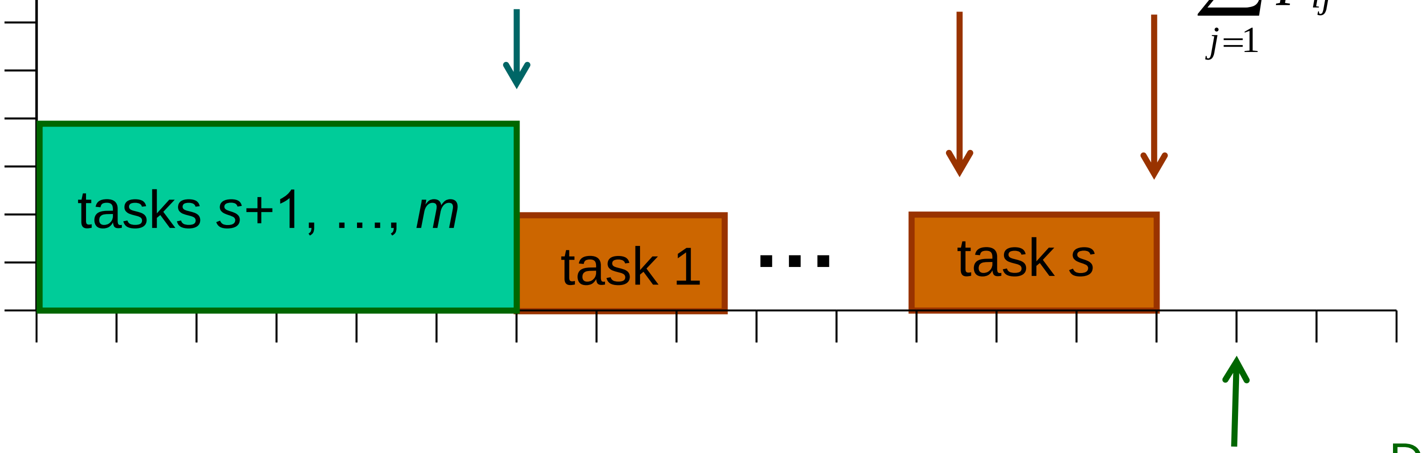
Assuming all deadlines d_i are the same, we get the Benders cut

$$M \ge M_{hi}^{*} - \sum_{j \in J_{hi}} (1 - x_{ij}) p_{ij}$$

Min makespan on resource *i* in last iteration

Why does this work? Assume all deadlines are the same. Add tasks 1,…,*s* sequentially at end of optimal schedule for other tasks…

Case I: resulting schedule meets deadline



Optimal makespan for tasks *s*+1,…,*m*

$\hat{M}$

Optimal makespan for all tasks

M^*

Feasible makespan for all tasks

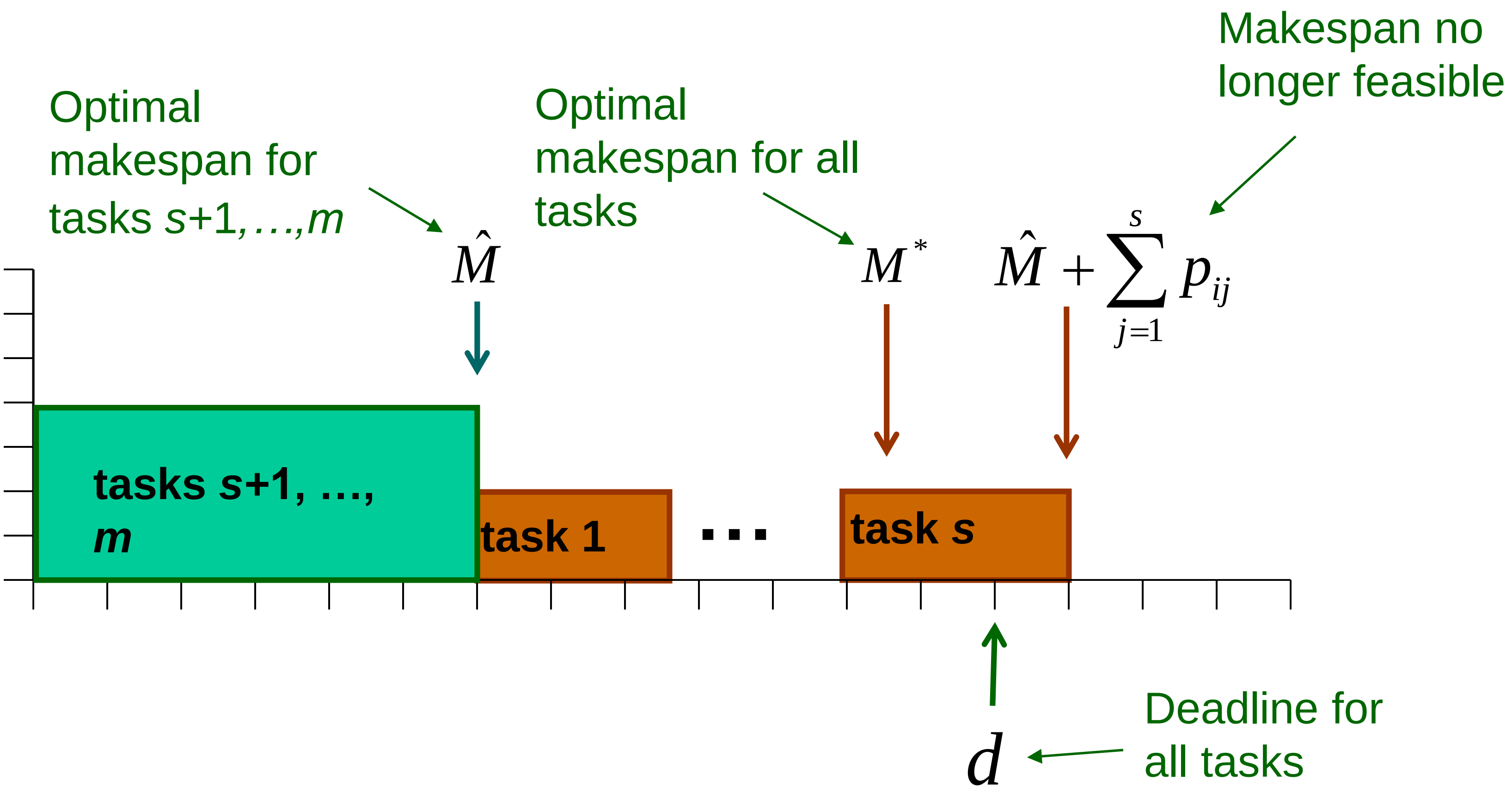
$\hat{M} + \sum_{j=1}^{s} p_{ij}$

d

Deadline for all tasks

$$M^* \leq \hat{M} + \sum_{j=1}^{s} p_{ij} \Rightarrow \hat{M} \geq M^* - \sum_{j=1}^{s} p_{ij}$$

Case II: resulting schedule exceeds deadline



$$M^* \le d \text{ and } \hat{M}+\sum_{j=1}^{s} p_{ij} > d \Rightarrow \hat{M} \ge M^* - \sum_{j=1}^{s} p_{ij}$$

Min Number of Late Tasks

Master problem: Assign tasks to resources

= 1 if task *j* is assigned to resource *i*

$$\begin{aligned} \min \quad & L \\ \text{subject to} \quad & \sum_i x_{ij} = 1, \quad \text{all } j \\ & \text{Benders cuts} \\ & \text{relaxation of subproblem} \\ & x_{ij} \in \{0,1\} \end{aligned}$$

Min Number of Late Tasks

Benders cuts

Lower bound on # late tasks on resource *i*

Min # late tasks on resource *i* (solution of subproblem)

$$L \geq \sum_i \hat{L}_{hi}$$

$$\hat{L}_{hi} \geq L^*_{hi} - L^*_{hi} \sum_{j \in J^0_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{L}_{hi} \geq L^*_{hi} - 1 - L^*_{hi} \sum_{j \in J^1_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{L}_{hi} \geq 0, \quad \text{all } i$$

Min Number of Late Tasks

Benders cuts

Lower bound on # late tasks on resource i

Min # late tasks on resource *i* (solution of subproblem)

$$L \geq \sum_i \hat{L}_{hi}$$

$$\hat{L}_{hi} \geq L^*_{hi} - L^*_{hi} \sum_{j \in J^0_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{L}_{hi} \geq L^*_{hi} - 1 - L^*_{hi} \sum_{j \in J^1_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{L}_{hi} \geq 0, \quad \text{all } i$$

subset of J_{hi} for which min # late tasks is still L_{hi}* (found by heuristic that repeatedly solves subproblem on resource *i*)

Min Number of Late Tasks

Benders cuts

Min # late tasks on resource *i* (solution of subproblem)

$$L \ge \sum_i \hat{L}_{hi}$$

$$\hat{L}_{hi} \ge L^*_{hi} - L^*_{hi} \sum_{j \in J^0_{hi}} (1 - x_{ij}), \quad \text{all } i$$

To reduce # late tasks, must remove one of the tasks in J^0_{hi} from resource *i*.

$$\hat{L}_{hi} \ge L^*_{hi} - 1 - L^*_{hi} \sum_{j \in J^1_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{L}_{hi} \ge 0, \quad \text{all } i$$

Min Number of Late Tasks

Benders cuts

$$L \ge \sum_i \hat{L}_{hi}$$

$$\hat{L}_{hi} \ge L^*_{hi} - L^*_{hi} \sum_{j \in J^0_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{L}_{hi} \ge L^*_{hi} - 1 - L^*_{hi} \sum_{j \in J^1_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{L}_{hi} \ge 0, \quad \text{all } i$$

Min # late tasks on resource *i* (solution of subproblem)

subset of J_{hi} for which min # late tasks is still L_{hi}* (found by heuristic that repeatedly solves subproblem on resource *i*)

Smaller subset of J_{hi} for which min # late tasks is L_{hi}* – 1 (found while running same heuristic)

Min Number of Late Tasks

Benders cuts

$$L \ge \sum_i \hat{L}_{hi}$$

Min # late tasks on resource *i* (solution of subproblem)

$$\hat{L}_{hi} \ge L^*_{hi} - L^*_{hi} \sum_{j \in J^0_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{L}_{hi} \ge L^*_{hi} - 1 - L^*_{hi} \sum_{j \in J^1_{hi}} (1 - x_{ij}), \quad \text{all } i$$

To reduce # late tasks by more than 1, must remove one of the tasks in J^1_{hi} from resource *i*.

$$\hat{L}_{hi} \ge 0, \quad \text{all } i$$

Min Number of Late Tasks

Benders cuts

$$L \geq \sum_{i} \hat{L}_{hi}$$

$$\hat{L}_{hi} \geq L^*_{hi} - L^*_{hi} \sum_{j \in J^0_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{L}_{hi} \geq L^*_{hi} - 1 - L^*_{hi} \sum_{j \in J^1_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{L}_{hi} \geq 0, \quad \text{all } i$$

These Benders cuts are added to the master problem in each iteration *h.*

Min Number of Late Tasks

Relaxation of subproblem

Lower bound on # late tasks on resource i

$$L \geq \sum_i L_i$$

$$L_i \geq \frac{\dfrac{1}{C_i} \displaystyle\sum_{k \in J(d_j)} c_{ik} p_{ik} x_{ik} - d_j}{\displaystyle\max_{k \in J(d_j)} \{p_{ik}\}}, \text{ all } j$$

Min Number of Late Tasks

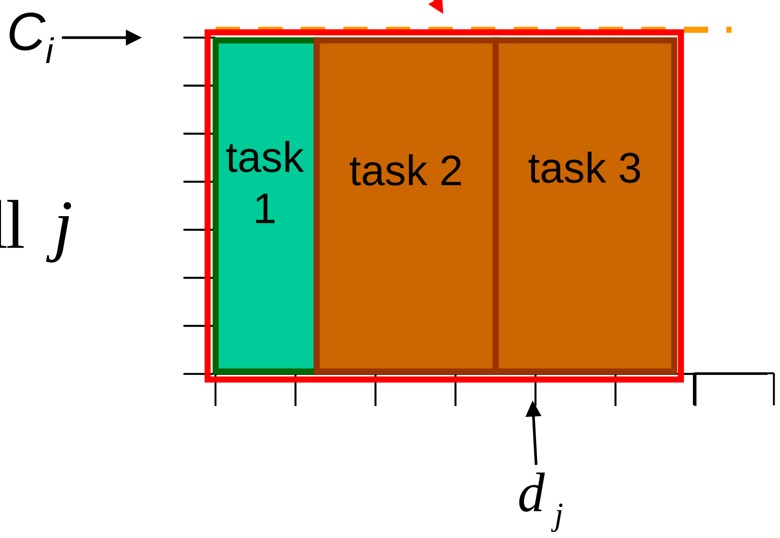
Relaxation of subproblem

Lower bound on # late tasks on resource i

$$L \ge \sum_i L_i$$

Set of tasks assigned to resource *i* with deadline at or before d_j

$$L_i \ge \frac{\frac{1}{C_i}\sum_{k\in J(d_j)} c_{ik}p_{ik}x_{ik} - d_j}{\max_{k\in J(d_j)}\{p_{ik}\}}, \text{ all } j$$



Min Number of Late Tasks

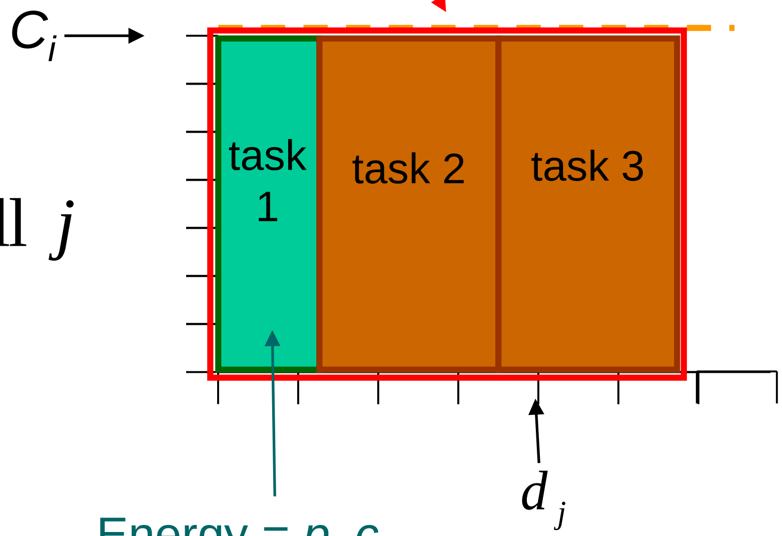
Relaxation of subproblem

Lower bound on # late tasks on resource i

$$L \geq \sum_i L_i$$

Set of tasks assigned to resource *i* with deadline at or before d_j

$$L_i \geq \frac{\dfrac{1}{C_i}\displaystyle\sum_{k\in J(d_j)} c_{ik}p_{ik}x_{ik} - d_j}{\displaystyle\max_{k\in J(d_j)}\{p_{ik}\}}, \quad \text{all } j$$



Energy = $p_{i1}c_{i1}$

Min Number of Late Tasks

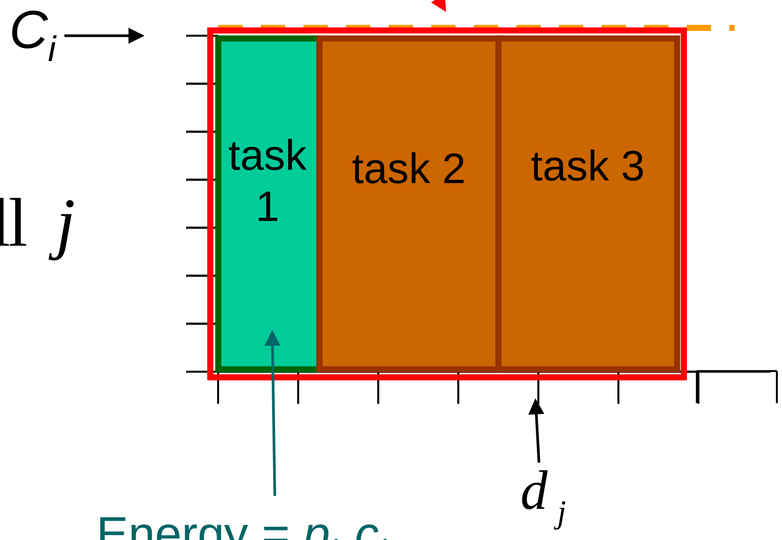
Relaxation of subproblem

Lower bound on # late tasks on resource i

$$L \geq \sum_i L_i$$

Area of tasks assigned to resource *i* with deadline at or before d_j

$$L_i \geq \frac{\dfrac{1}{C_i} \boxed{\displaystyle\sum_{k \in J(d_j)} c_{ik} p_{ik} x_{ik}} - d_j}{\displaystyle\max_{k \in J(d_j)} \{p_{ik}\}}, \text{ all } j$$



Energy = $p_{i1}c_{i1}$

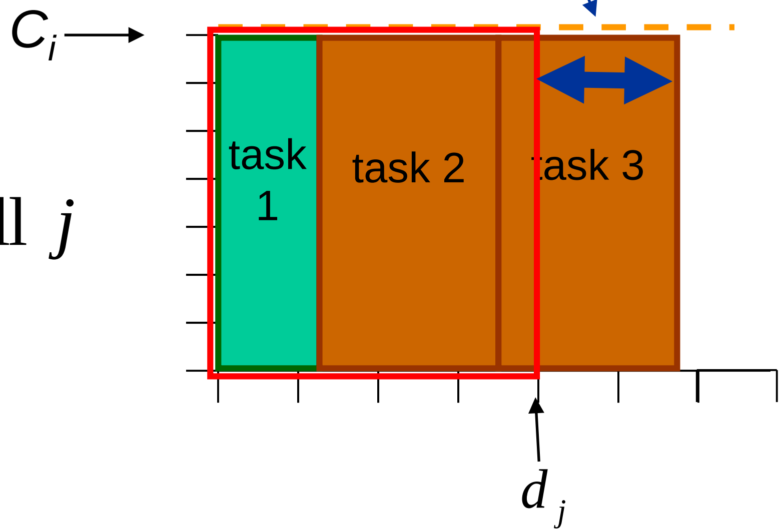
Min Number of Late Tasks

Relaxation of subproblem

Lower bound on (makespan – latest deadline)

$$L \ge \sum_i L_i$$

$$L_i \ge \frac{\boxed{\frac{1}{C_i}\sum_{k\in J(d_j)} c_{ik}\,p_{ik}\,x_{ik} - d_j}}{\max\limits_{k\in J(d_j)}\{p_{ik}\}}, \text{ all } j$$



Min Number of Late Tasks

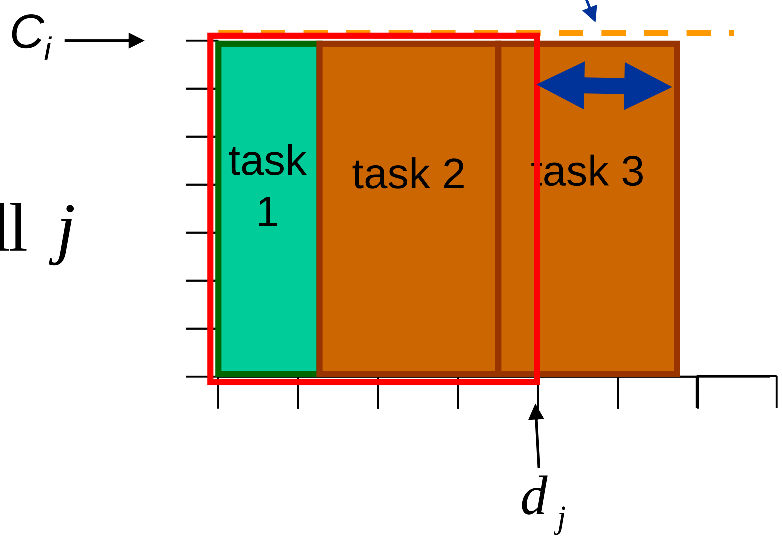
Relaxation of subproblem

Lower bound on (makespan – latest deadline)

$$L \ge \sum_i L_i$$

$$L_i \ge \frac{\boxed{\dfrac{1}{C_i}\displaystyle\sum_{k\in J(d_j)} c_{ik}\,p_{ik}\,x_{ik} - d_j}}{\boxed{\displaystyle\max_{k\in J(d_j)}\{p_{ik}\}}}, \text{ all } j$$

Max processing time



Min Number of Late Tasks

Relaxation of subproblem

$$L \geq \sum_i L_i$$

$$L_i \geq \boxed{\dfrac{\dfrac{1}{C_i} \displaystyle\sum_{k \in J(d_j)} c_{ik} p_{ik} x_{ik} - d_j}{\displaystyle\max_{k \in J(d_j)} \{p_{ik}\}}}, \text{ all } j$$

Min # of late jobs on resource *i*

Min Number of Late Tasks

Relaxation of subproblem

This relaxation is added to the master problem at the outset.

$$L \geq \sum_i L_i$$

$$L_i \geq \boxed{\dfrac{\dfrac{1}{C_i} \displaystyle\sum_{k \in J(d_j)} c_{ik} p_{ik} x_{ik} - d_j}{\displaystyle\max_{k \in J(d_j)} \{p_{ik}\}}}, \text{ all } j$$

Min # of late jobs on resource *i*

Min Tardiness Cumulative Scheduling

Master problem: assign tasks to resources

= 1 if task *j* is assigned to resource *i*

$$
\begin{array}{ll}
\min & L \\
\text{subject to} & \sum_i x_{ij} = 1, \quad \text{all } j \\
& \text{Benders cuts} \\
& \text{relaxation I of subproblem} \\
& \text{relaxation II of subproblem} \\
& x_{ij} \in \{0,1\}
\end{array}
$$

Min Tardiness Cumulative Scheduling

Benders cuts

Lower bound on tardiness for resource *i*

Min tardiness on resource *i*
(solution of subproblem)

$$T \geq \sum_i \hat{T}_{hi}$$

$$\hat{T}_{hi} \geq T^*_{hi} - T^*_{hi} \sum_{j \in J_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{T}_{hi} \geq T^0_{hi} - T^0_{hi} \sum_{j \in J_{hi} \setminus Z_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{T}_{hi} \geq 0, \quad \text{all } i$$

Min Tardiness Cumulative Scheduling

Benders cuts

Lower bound on tardiness for resource *i*

$$T \geq \sum_i \hat{T}_{hi}$$

Min tardiness on resource *i*
(solution of subproblem)

$$\hat{T}_{hi} \geq T^*_{hi} - T^*_{hi} \sum_{j \in J_{hi}} (1 - x_{ij}), \quad \text{all } i$$

To reduce tardiness on resource *i*, must remove one of the tasks assigned to it.

$$\hat{T}_{hi} \geq T^0_{hi} - T^0_{hi} \sum_{j \in J_{hi} \setminus Z_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{T}_{hi} \geq 0, \quad \text{all } i$$

Min Tardiness Cumulative Scheduling

Benders cuts

$$T \ge \sum_i \hat{T}_{hi}$$

$$\hat{T}_{hi} \ge T^*_{hi} - T^*_{hi} \sum_{j \in J_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{T}_{hi} \ge T^0_{hi} - T^0_{hi} \sum_{j \in J_{hi} \setminus Z_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{T}_{hi} \ge 0, \quad \text{all } i$$

Min tardiness on resource *i*
(solution of subproblem)

Set of tasks that can be removed,
one at a time from resource *i*
without reducing min tardiness.

Min Tardiness Cumulative Scheduling

Benders cuts

$$T \geq \sum_i \hat{T}_{hi}$$

$$\hat{T}_{hi} \geq T^*_{hi} - T^*_{hi} \sum_{j \in J_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{T}_{hi} \geq T^0_{hi} - T^0_{hi} \sum_{j \in J_{hi} \setminus Z_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{T}_{hi} \geq 0, \quad \text{all } i$$

Set of tasks that can be removed, one at a time from resource *i* without reducing min tardiness.

Min tardiness on resource *i* when all tasks in Z_{hi} are removed *simultaneously*.

Min Tardiness Cumulative Scheduling

Benders cuts

$$T \geq \sum_i \hat{T}_{hi}$$

$$\hat{T}_{hi} \geq T^*_{hi} - T^*_{hi} \sum_{j \in J_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{T}_{hi} \geq T^0_{hi} - T^0_{hi} \sum_{j \in J_{hi} \setminus Z_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{T}_{hi} \geq 0, \quad \text{all } i$$

To reduce tardiness below T^0_{hi} on resource *i*, must remove one of the tasks in $J_{hi} \setminus Z_{hi}$

Set of tasks that can be removed, one at a time from resource *i* without reducing min tardiness.

Min tardiness on resource *i* when all tasks in Z_{hi} are removed *simultaneously*.

Min Tardiness Cumulative Scheduling

These Benders cuts are added to the master problem in each iteration *h*

$$T \geq \sum_i \hat{T}_{hi}$$

$$\hat{T}_{hi} \geq T^*_{hi} - T^*_{hi} \sum_{j \in J_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{T}_{hi} \geq T^0_{hi} - T^0_{hi} \sum_{j \in J_{hi} \setminus Z_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{T}_{hi} \geq 0, \quad \text{all } i$$

Subproblem relaxation I

Lower bound on total tardiness for resource *i*

$$T \geq \sum_i \boxed{T_i}$$

$$T_i \geq \frac{1}{C_i} \sum_{j \in J(d_k)} c_{ij} p_{ij} x_{ij} - d_k, \text{ all } k$$

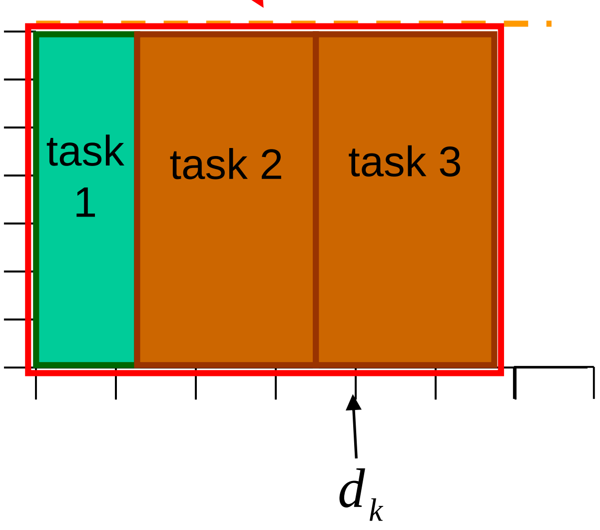
Subproblem relaxation I

Lower bound on total tardiness for resource *i*

$$T \geq \sum_i T_i$$

Set of tasks assigned to resource *i* with deadline at or before d_k

$$T_i \geq \frac{1}{C_i} \sum_{j \in J(d_k)} c_{ij} p_{ij} x_{ij} - d_k, \text{ all } k$$



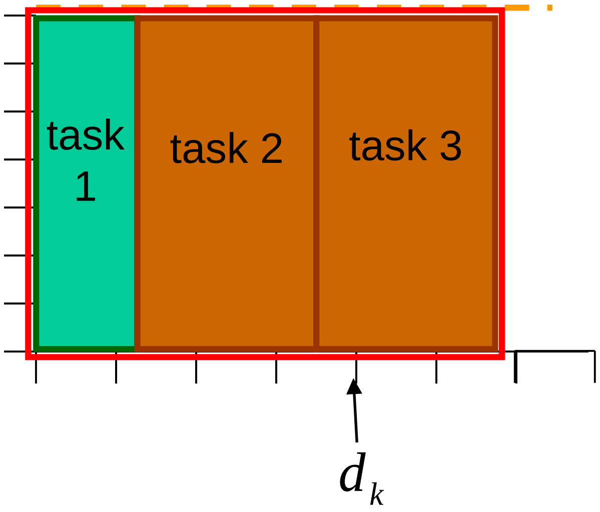
Subproblem relaxation I

Lower bound on total tardiness for resource *i*

Area of tasks assigned to resource *i* with deadline at or before d_k

$$T \geq \sum_i \boxed{T_i}$$

$$T_i \geq \frac{1}{C_i} \boxed{\sum_{j \in J(d_k)} c_{ij} p_{ij} x_{ij}} - d_k, \ \text{all } k$$



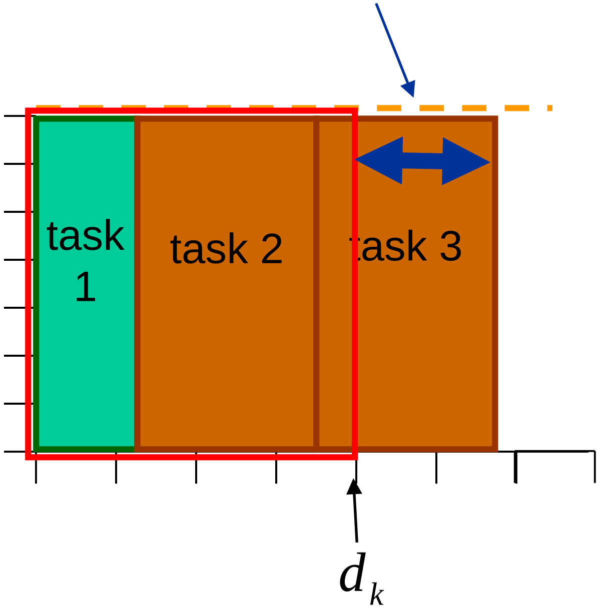
Subproblem relaxation I

Lower bound on total tardiness for resource *i*

Lower bound on total tardiness

$$T \ge \sum_i T_i$$

$$T_i \ge \boxed{\frac{1}{C_i}\sum_{j\in J(d_k)} c_{ij}\,p_{ij}\,x_{ij} - d_k,}\ \text{all } k$$



Subproblem relaxation II

Lemma. Consider a min tardiness problem that schedules tasks 1, …, n on resource i, where $d_1 \leq \ldots \leq d_n$. The min tardiness T^* is bounded below by

$$\bar{T} = \sum_{k=1}^{n} \bar{T}_k$$

where

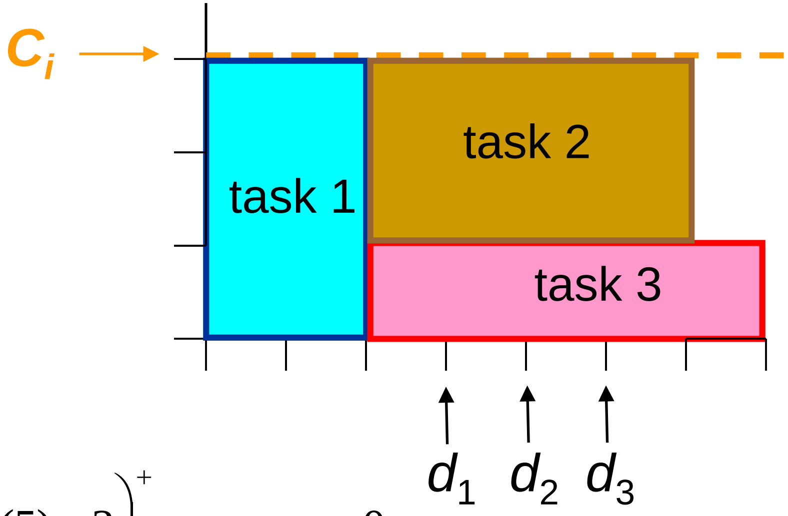
$$\bar{T}_k = \left(\frac{1}{C_i} \sum_{j=1}^{k} p_{i\pi_i(j)} c_{i\pi_i(j)} - d_k \right)^+$$

and π is a permutation of 1, …, n such that

$$p_{\pi_i(1)} c_{\pi_i(1)} \leq \cdots \leq p_{\pi_i(n)} c_{\pi_i(n)}$$

Example of Lemma

j	d_j	p_{ij}	c_{ij}	$p_{ij}c_{ij}$
1	3	2	3	6
2	4	4	2	8
3	5	5	1	5



$$\overline{T}_1 = \left(\frac{1}{C_i}(p_{i3}c_{i3}) - d_1\right)^+ = \left(\frac{1}{3}(5) - 3\right)^+ = 0$$

$$\overline{T}_2 = \left(\frac{1}{C_i}(p_{i3}c_{i3} + p_{i1}c_{i1}) - d_2\right)^+ = \left(\frac{1}{3}(5+6) - 4\right)^+ = 0$$

$$\overline{T}_3 = \left(\frac{1}{C_i}(p_{i3}c_{i3} + p_{i1}c_{i1} + p_{i2}c_{i2}) - d_3\right)^+ = \left(\frac{1}{3}(5+6+8) - 5\right)^+ = 4/3$$

$$\text{Lower bound on tardiness} = \left\lceil \overline{T}_1 + \overline{T}_2 + \overline{T}_3 \right\rceil = \left\lceil 4/3 \right\rceil = 2$$

Min tardiness = 4

Idea of proof

For a permutation σ of 1,…,n let $T(\sigma) = \sum_{k=1}^{n} T_k(\sigma)$

where $$T_k(\sigma) = \left(\frac{1}{C_i} \sum_{j=1}^{k} p_{i\pi_i(j)} c_{i\pi_i(j)} - d_{\sigma(k)} \right)^{+}$$

Let $\sigma_0(1)$, …, $\sigma_0(n)$ be order of jobs in any optimal solution, so that $t_{\sigma_0(1)} \le \cdots \le t_{\sigma_0(n)}$ and min tardiness is T^*

Consider bubble sort on $\sigma_0(1)$, …, $\sigma_0(n)$ to obtain 1,…,n. Let $\sigma_0, \ldots, \sigma_S$ be resulting sequence of permutations, so that σ_s, σ_{s+1} differ by a swap and $\sigma_S(j) = j$.

Now we have

swap *k* and *k+1*

$$T^* \geq T(\sigma_0) \geq \cdots \geq T(\sigma_s) \geq T(\sigma_{s+1}) \geq \cdots \geq T(\sigma_S) = \bar{T}$$

since $T^* = \sum_{j=1}^{n} \left(t_{\sigma_0(j)} + p_{i\sigma_0(j)} - d_{\sigma_0(j)}\right)^+ \geq \sum_{j=1}^{n} \left(\frac{1}{C_i} \sum_{j=1}^{k} p_{i\sigma_0(j)} c_{i\sigma_0(j)} - d_{\sigma_0(j)}\right)^+ \geq \sum_{j=1}^{n} \left(\frac{1}{C_i} \sum_{j=1}^{k} p_{i\pi_i(j)} c_{i\pi_i(j)} - d_{\sigma_0(j)}\right)^+ = T(\sigma_0)$

areas

def. of π

$$T(\sigma_s) = \sum_{j=1}^{k-1} T_j(\sigma_s) + T_k(\sigma_s) + T_{k+1}(\sigma_s) + \sum_{j=k+2}^{n} T_j(\sigma_s)$$

$$T(\sigma_{s+1}) = \sum_{j=1}^{k-1} T_j(\sigma_s) + T_k(\sigma_{s+1}) + T_{k+1}(\sigma_{s+1}) + \sum_{j=k+2}^{n} T_j(\sigma_s)$$

So

$$\begin{aligned} T(\sigma_s) - T(\sigma_{s+1}) &= T_k(\sigma_s) + T_{k+1}(\sigma_s) - T_k(\sigma_{s+1}) - T_{k+1}(\sigma_{s+1}) \\ &= (a-A)^+ + (A-b)^+ - (a-b)^+ - (A-B)^+ \geq 0 \end{aligned}$$

since *A* ≥ *a*, *B* ≥ *b*

Writing relaxation II

From the lemma, we can write the relaxation

$$T \geq \sum_i \sum_{k=1}^{n} T'_{ik} x_{ik}$$

where $T'_{ik} \geq \frac{1}{C_i} \sum_{j=1}^{k} p_{i\pi_i(j)} c_{i\pi_i(j)} x_{i\pi_i(j)} - d_k$

To linearize this, we write $T \geq \sum_i \sum_{k=1}^{n} T_{ik}$

and $T_{ik} \geq \frac{1}{C_i} \sum_{j=1}^{k} p_{i\pi_i(j)} c_{i\pi_i(j)} x_{i\pi_i(j)} - d_k - (1 - x_{ik}) M_{ik}$

where $M_{ik} = \frac{1}{C_i} \sum_{j=1}^{k} p_{i\pi_i(j)} c_{i\pi_i(j)} - d_k$

Computational Results

- Random problems on 2, 3, 4 resources.
- Facilities run at different speeds.
- All release times = 0.
  - Min cost and makespan problems: deadlines same/different.
  - Tardiness problems: random due date parameters set so that a few tasks tend to be late.
- No precedence or other side constraints.
  - Makes problem harder.
- Implement with OPL Studio

Min makespan, 2 resources

Average of 5 instances shown

Jobs	MILP	CP	Benders
10	3.4	0.8	0.24
12	12	4.0	0.31
14	2572+	299	5.0
16	5974+	3737	36
18		7200+	233
20			1268

+ At least one problem in the 5 exceeded 7200 sec (2 hours)

Min makespan, 3 resources

Average of 5 instances shown

Jobs	MILP	CP	Benders
10	3.9	0.9	0.23
12	12	7.5	0.38
14	524	981	1.4
16	1716+	4414	7.6
18	4619+	7200+	30
20			8.7
22			2012+

+ At least one problem in the 5 exceeded 7200 sec (2 hours)

Min makespan, 4 resources

Average of 5 instances shown

Jobs	MILP	CP	Benders
10	1.0	0.07	0.19
12	5.0	1.9	0.43
14	24	524	0.82
16	35	3898	1.0
18	3931+	7200+	6.4
20			4.4
22			28
24			945

+ At least one problem in the 5 exceeded 7200 sec (2 hours)

Min makespan, 3 resources
Different deadlines

Average of 5 instances shown

Jobs	MILP	CP	Benders
14	223	7.1	4.4
16	853	1620+	5.1
18	350	1928+	2.9
20	7200+	7200+	1449+
22	7200+		388
24	7200+		132

+ At least one problem in the 5 exceeded 7200 sec (2 hours)

Min # late tasks

Smaller instances

Tasks	Time (sec) CP	 MILP	 Benders	Min # late tasks
14	1092	5.8	0.5	1
	382	8.0	0.7	1
	265	3.2	0.7	2
	85	2.6	1.3	2
	5228	1315	665	3
16	304	2.7	0.5	0
	?	31	0.2	1
	310	22	0.4	1
	4925	29	2.7	2
	19	5.7	24	4
18	>7200	2.0	0.1	0
	?	8.0	0.2	1
	>7200	867	8.5	1
	>7200	6.3	1.4	2
	>7200	577	3.4	2

Min # late tasks

Larger instances

Tasks	Time (sec)		Best solution	
	MILP	Benders	MILP	Benders
20	97	0.4	0	0
	>7200	2.3	(1)	1
	219	5.0	1	1
	>7200	11	(2)	2
	843	166	3	3
22	16	1.3	0	0
	>7200	3.7	(1)	1
	>7200	49	(3)	2
	>7200	3453	(5)	2
	>7200	>7200	(6)	(6)
24	25	0.8	0	0
	>7200	18	(1)	0
	>7200	62	(2)	0
	>7200	124	(3)	1
	>7200	234	(2)	1

( ) = optimality not proved

Effect of subproblem relaxation

Min # late tasks

Tasks	Time (sec)	
	with relax	without relax
16	0.5	2.6
	0.4	1.5
	0.2	1.3
	2.7	4.2
	24	18
18	0.1	1.1
	0.2	0.7
	3.4	3.3
	1.4	15
	8.5	11
20	0.4	88
	2.3	9.7
	5.0	63
	11	19
	166	226

Min total tardiness

Smaller instances

Tasks	Time (sec) CP	 MILP	 Benders	Min tardiness
14	838	7.0	6.1	1
	7159	34	3.7	2
	1783	45	19	15
	>7200	73	40	19
	>7200	>7200	3269	26
16	>7200	19	1.4	0
	>7200	46	2.1	0
	>7200	52	4.2	4
	>7200	1105	156	20
	>7200	3424	3.4	31
18		187	2.8	0
		15	5.3	3
		46	49	5
		256	47	11
		>7200	1203	14

Min total tardiness

Larger instances

Tasks	Time (sec)		Best solution	
	MILP	Benders	MILP	Benders
20	105	18	0	0
	4141	23	1	1
	39	29	4	4
	1442	332	8	8
	>7200	>7200	(75)	(37)
22	6	19	0	0
	584	37	2	2
	>7200	>7200	(120)	(40)
	>7200	>7200	(162)	(46)
	>7200	>7200	(375)	(141)
24	10	324	0	0
	>7200	94	(20)	0
	>7200	110	(57)	0
	>7200	>7200	(20)	(5)
	>7200	>7200	(25)	(7)

( ) = optimality not proved

Effect of subproblem relaxation

Min total tardiness

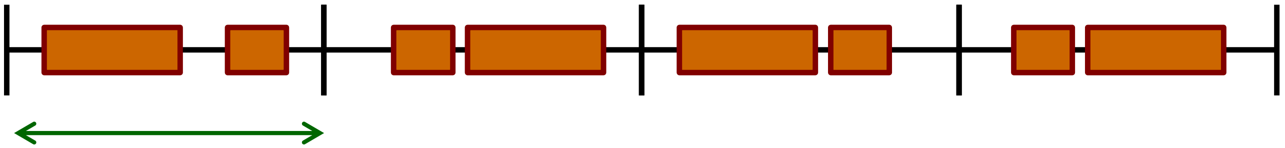
Tasks	Time (sec) with relax	 without relax
16	1.4	4.4
	2.1	6.5
	4.2	30
	156	199
	765	763
18	2.8	10
	5.3	17
	47	120
	49	354
	1203	5102
20	18	151
	23	1898
	29	55
	332	764
	>7200	>7200

Single-Resource Scheduling

- Apply logic-based Benders to single-resource scheduling with long time horizons and many jobs.
- Decompose the problem by assigning jobs to segments of time horizon.
  - Segmented problem – Jobs cannot cross segment boundaries (e.g., weekends).
  - Unsegmented problem – Jobs can cross segment boundaries.

Segmented problem

- Benders approach is very similar to that for the planning and scheduling problem.
    - Assign jobs to time segments rather than processors.
    - Benders cuts are the same.

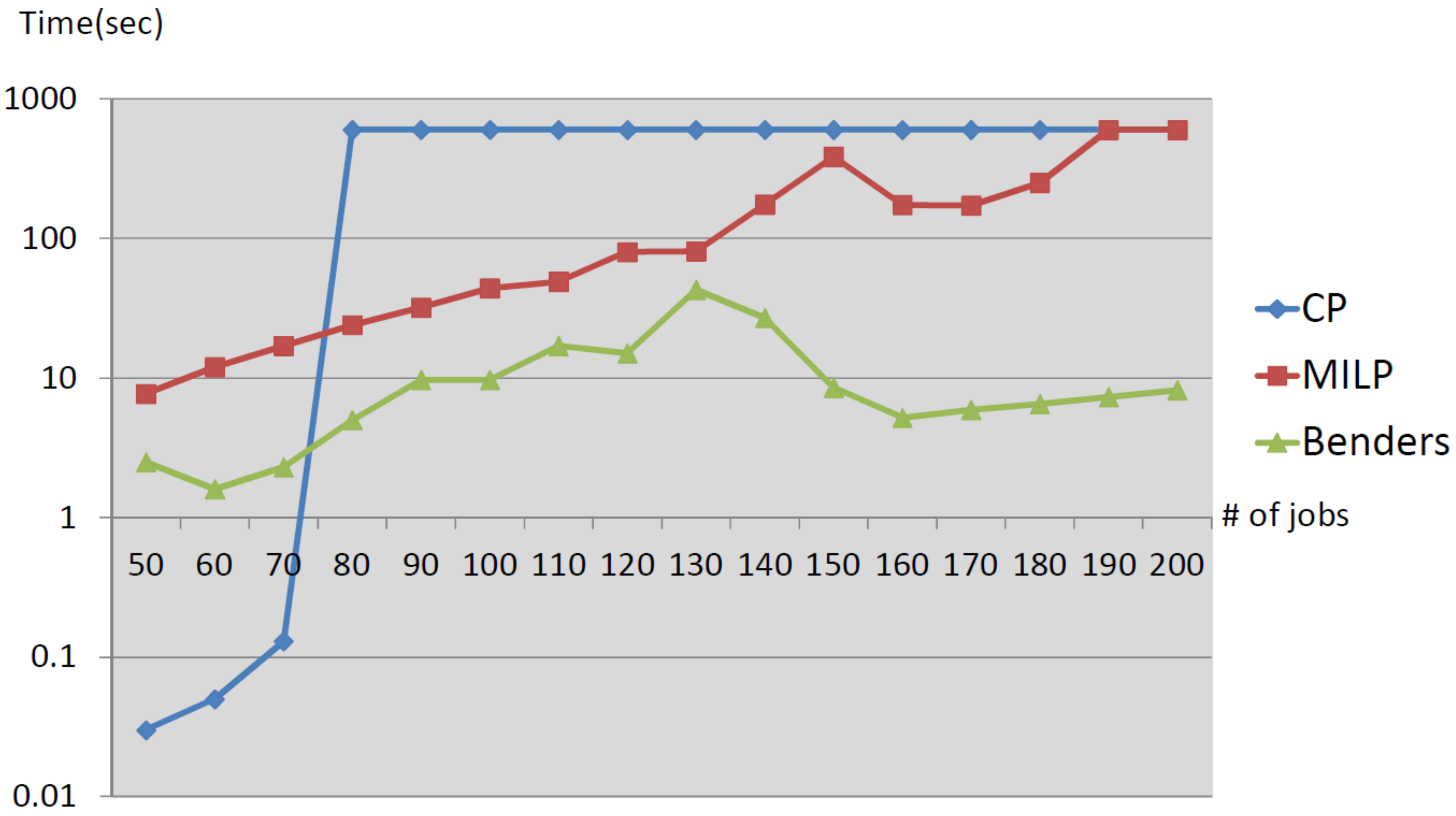


segment

Jobs do not overlap
segment boundaries

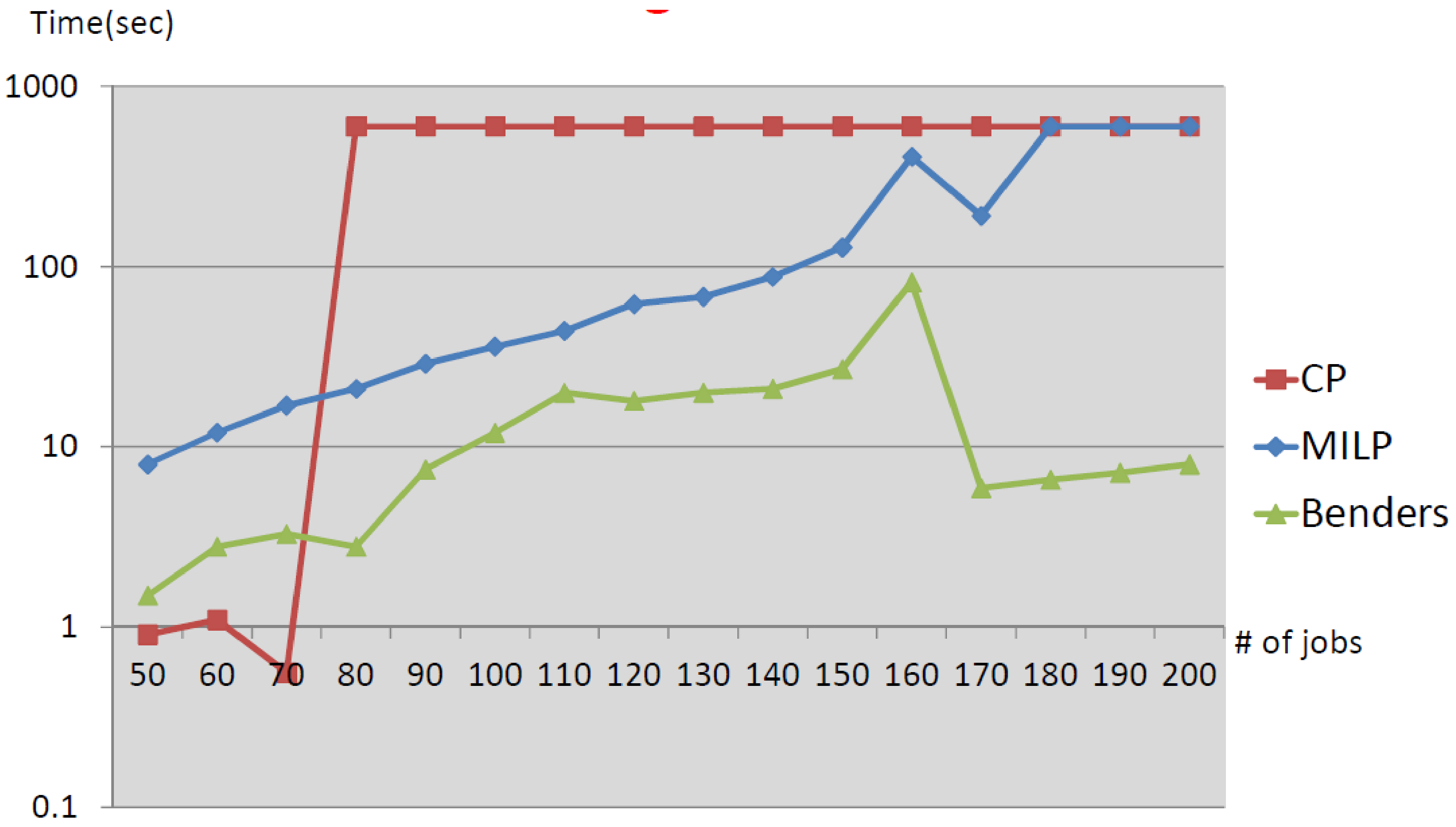
Segmented problem

Feasibility – Wide time windows (individual instances)



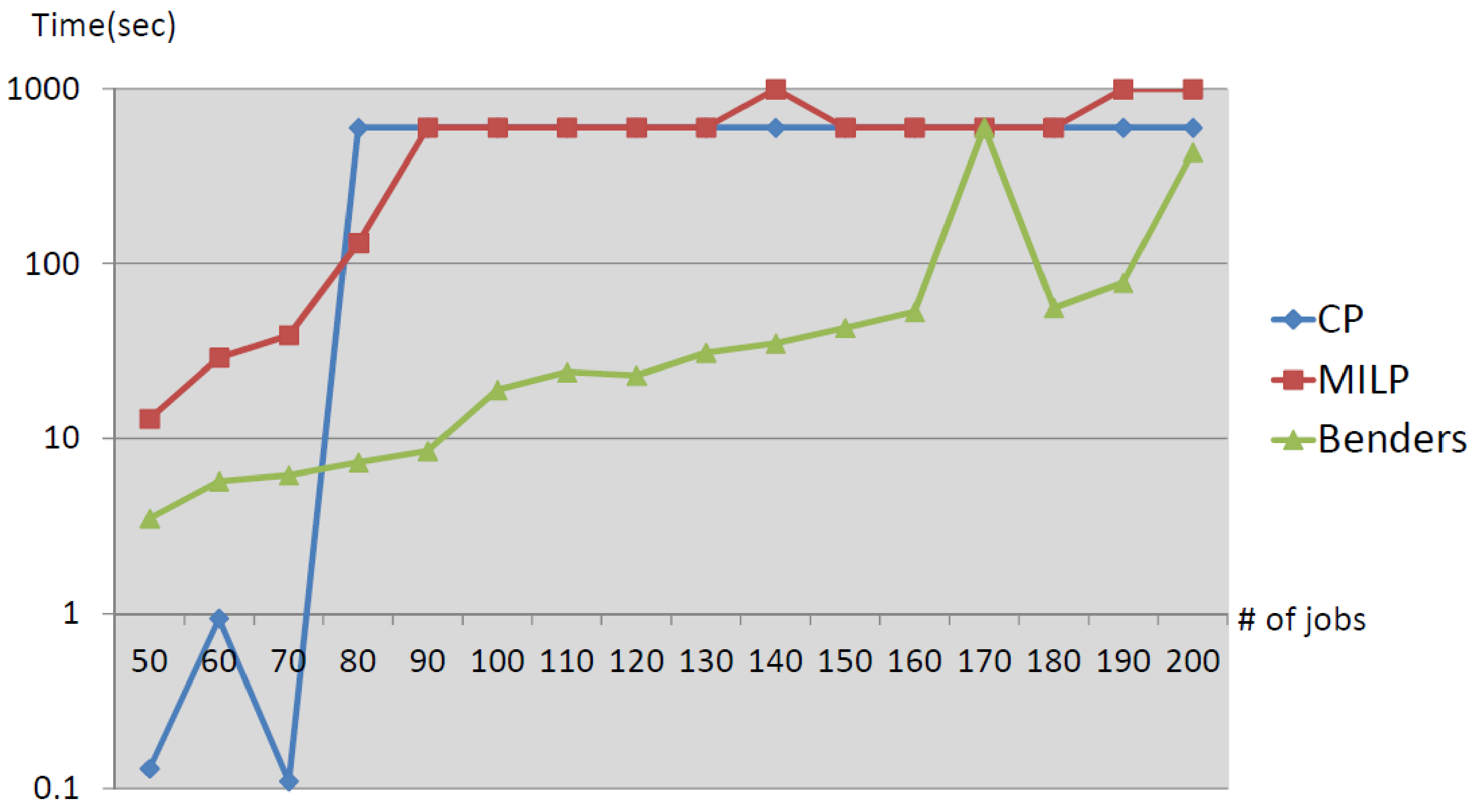
Segmented problem

Feasibility – Tight time windows (individual instances)



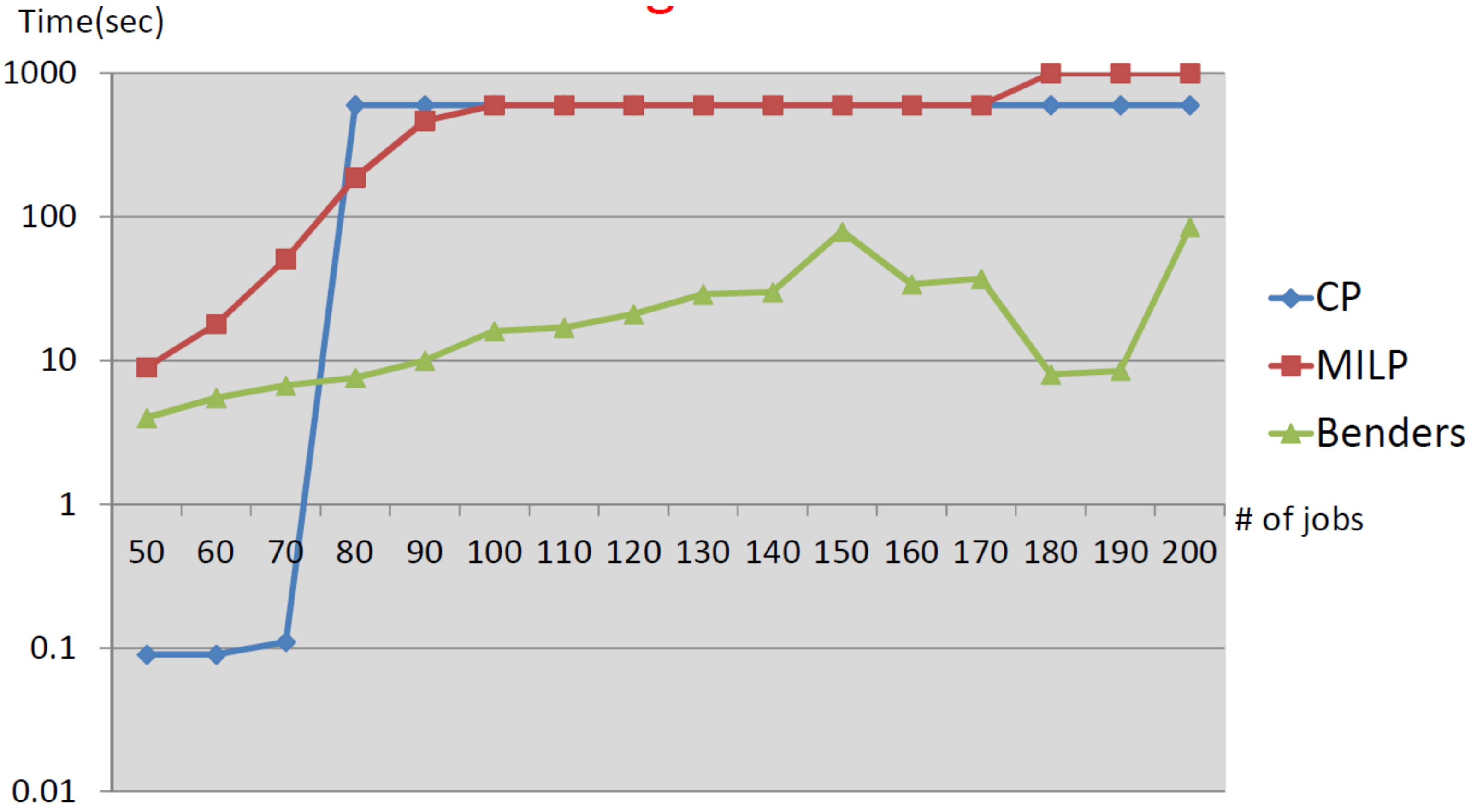
Segmented problem

Min makespan – Wide time windows (individual instances)



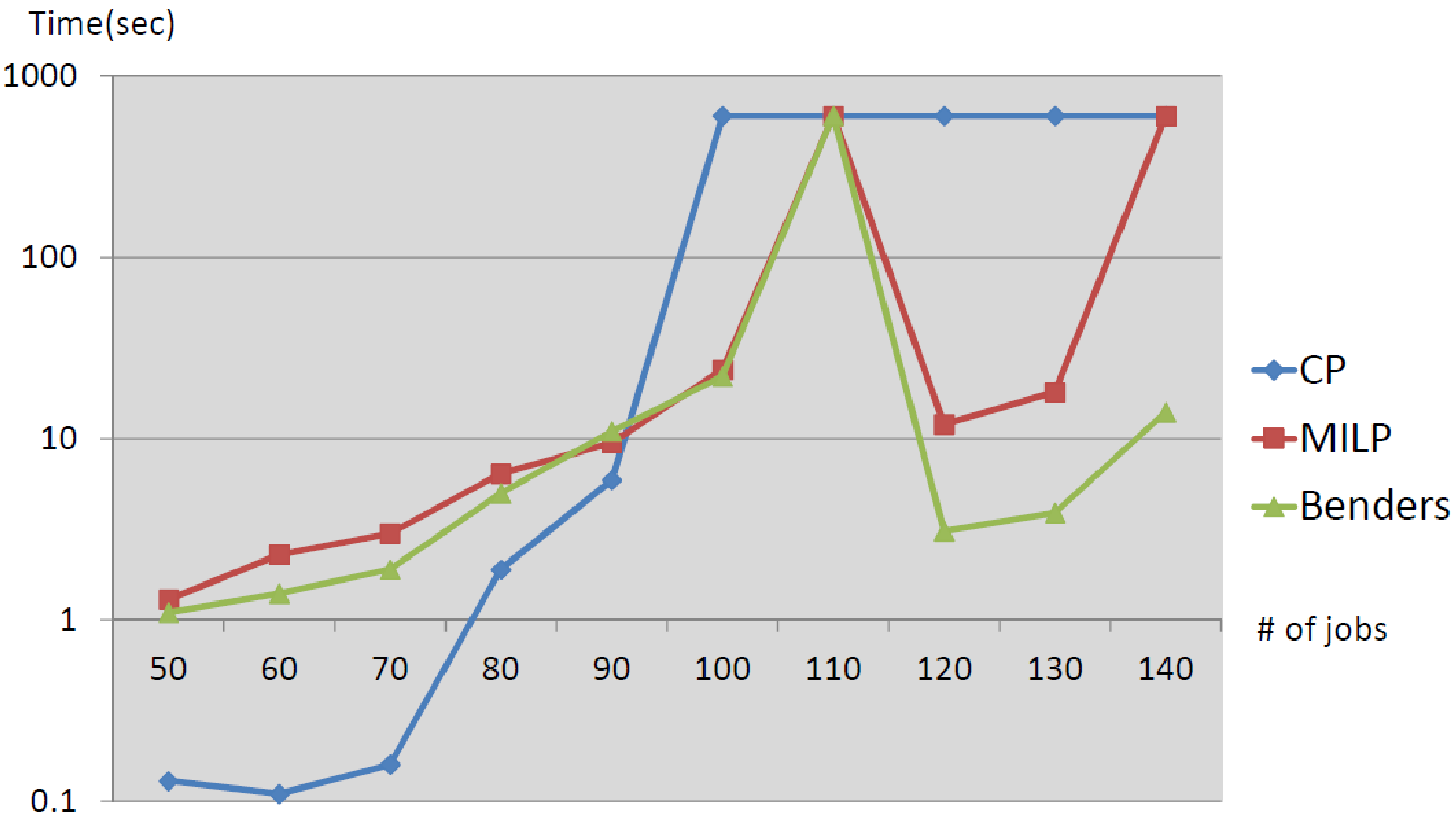
Segmented problem

Min makespan – Tight time windows (individual instances)



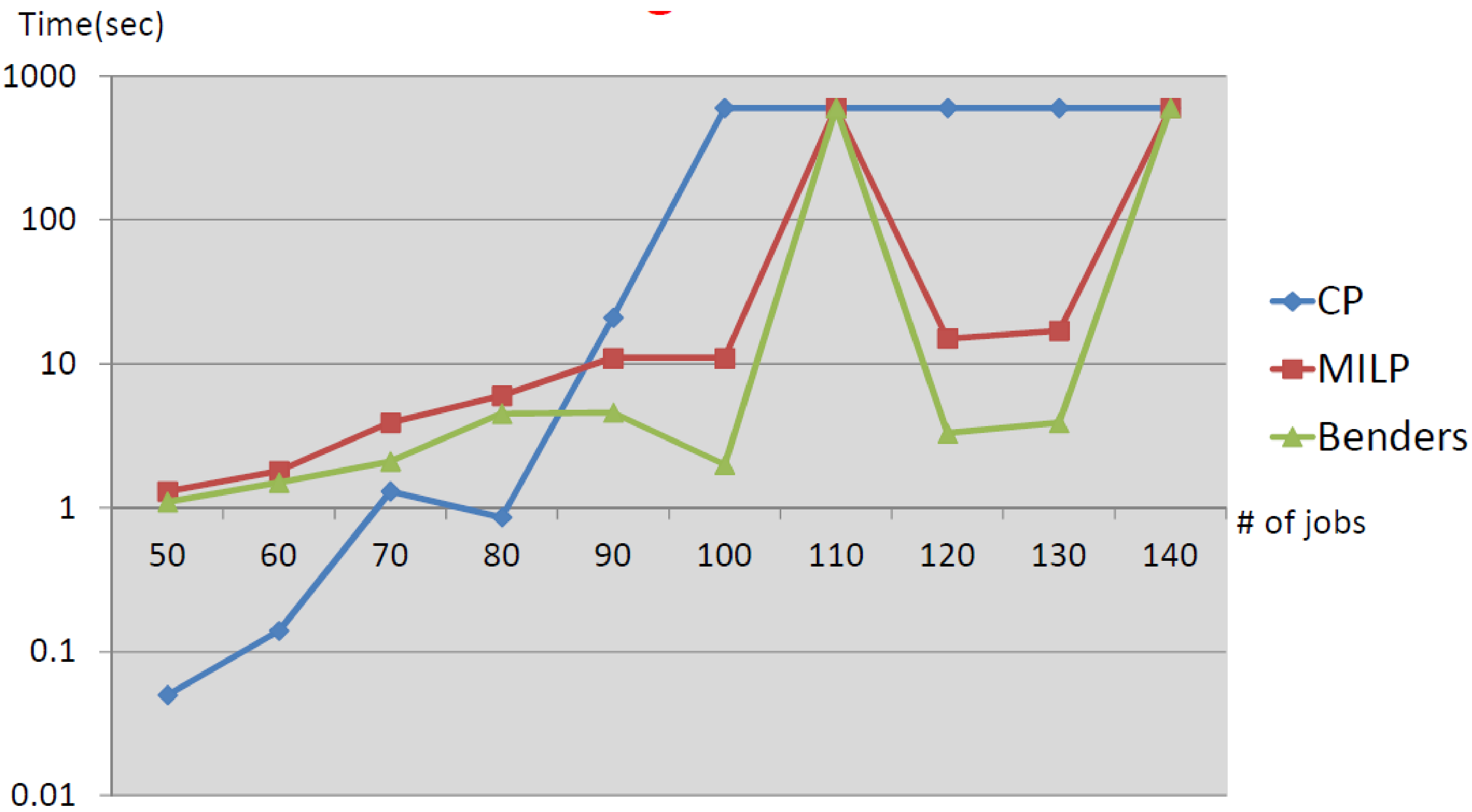
Segmented problem

Min tardiness – Wide time windows (individual instances)



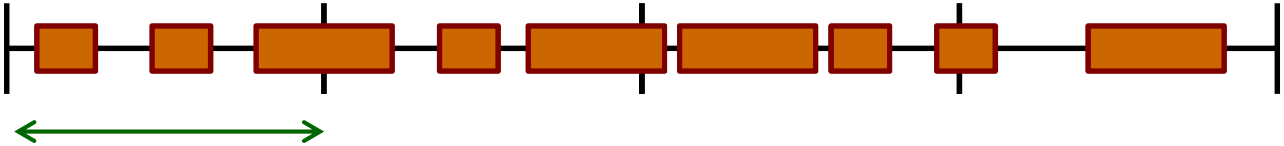
Segmented problem

Min tardiness – Tight time windows (individual instances)



Unsegmented problem

- Master problem is more complicated.
  - Jobs can overlap two or more segments.
  - Master problem variables must keep track of this.
- Benders cuts more sophisticated.



Jobs can overlap
segment boundaries

Unsegmented problem

- Master problem:

y_{ijk} variables keep track of whether job *j* starts, finishes, or runs entirely in segment *i.*

x_{ijk} variables keep track of how long a partial job *j* runs in segment *i*.

$$\sum_{i\in I} y_{ij} \ge 1, \quad j \in J$$

$$y_{ij} = y_{ij0} + y_{ij1} + y_{ij2} + y_{ij3}, \quad i \in I, j \in J$$

$$\sum_{j\in J} y_{ij1} \le 1, \quad \sum_{j\in J} y_{ij2} \le 1, \quad \sum_{j\in J} y_{ij3} \le 1, \quad i \in I$$

$$y_{ij1} \le y_{i-1,j,2} + y_{i-1,j,3}, \quad i \in I, i > 1, \ j \in J$$

$$y_{ij2} \le y_{i+1,j,1} + y_{i+1,j,3}, \quad i \in I, i < n, \ j \in J$$

$$y_{ij3} \le y_{i-1,j,3} + y_{i-1,j,2}, \quad i \in I, i > 1, \ j \in J$$

$$y_{ij3} \le y_{i+1,j,3} + y_{i+1,j,1}, \quad i \in I, i < n, \ j \in J$$

$$\sum_{i\in I} y_{ij0} \le 1, \quad \sum_{i\in I} y_{ij1} \le 1, \quad \sum_{i\in I} y_{ij2} \le 1, \quad j \in J$$

$$y_{1j1} = y_{1j3} = y_{nj2} = y_{nj3} = 0, \quad j \in J$$

$$\sum_{i\in I} y_{ij3} \le \left\lfloor \frac{p_j}{a_{i+1} - a_i} \right\rfloor, \quad j \in J$$

$$y_{ij}, y_{ij0}, y_{ij1}, y_{ij2}, y_{ij3} \in \{0, 1\}, \quad i \in I, j \in J$$

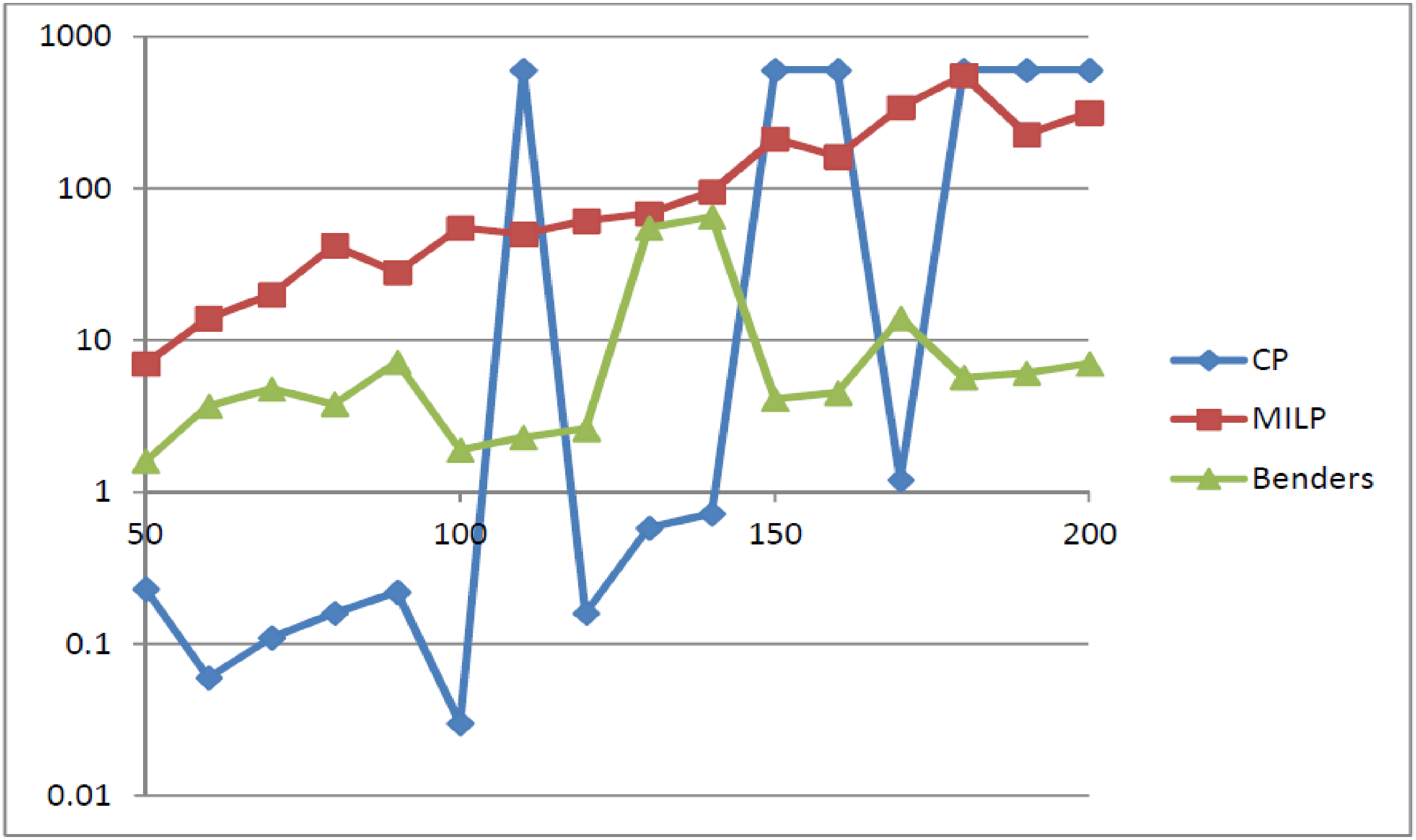
$$x_{ij1} \le p_j y_{ij1}, \quad x_{ij2} \le p_j y_{ij2}$$

$$x_{ij} = p_j y_{ij0} + x_{ij1} + x_{ij2} + (a_{i+1} - a_i) y_{ij3}$$

$$x_{ij1}, x_{ij2} \ge 0$$

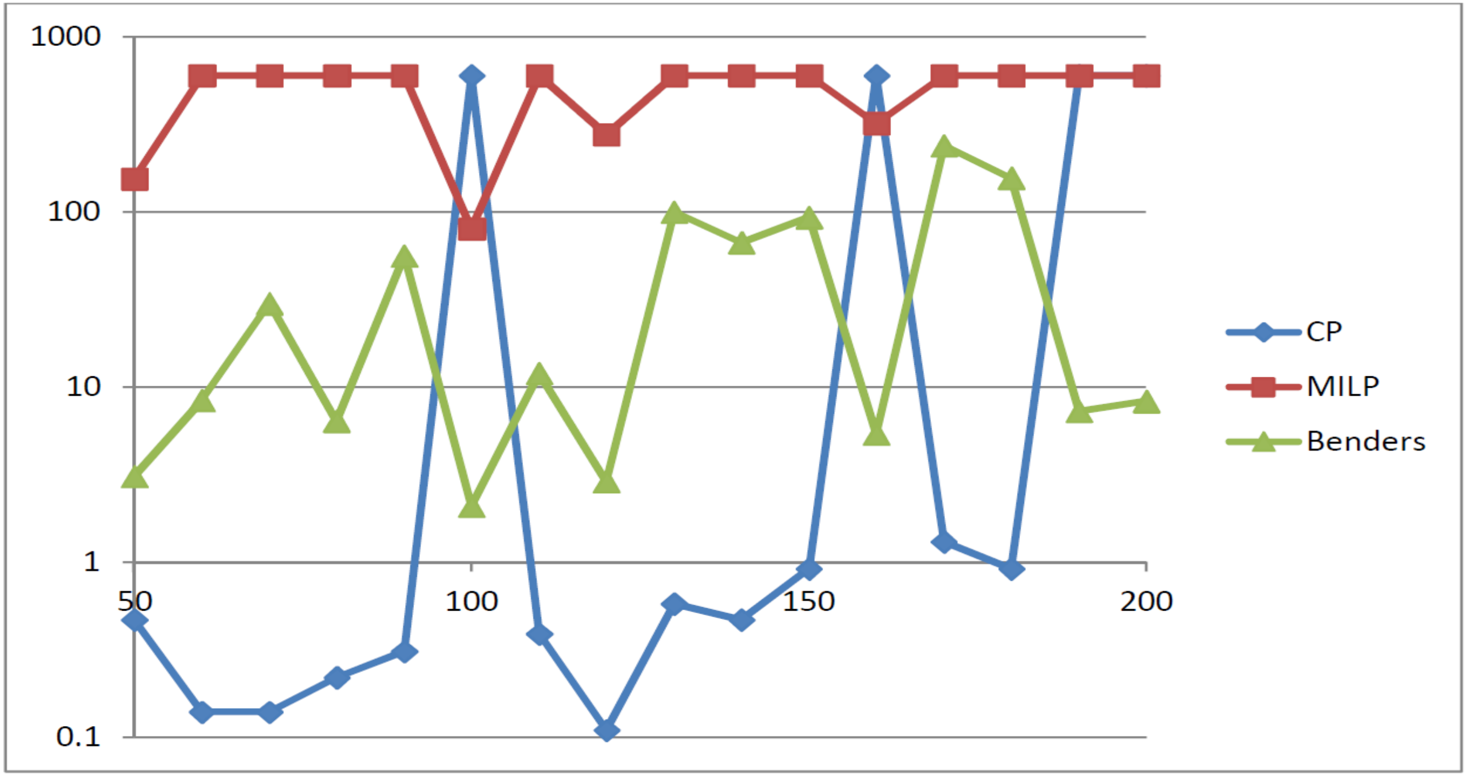
Unsegmented problem

Feasibility -- individual instances



Unsegmented problem

Min makespan – individual instances



Single-resource scheduling

- Segmented problems:
  - Benders is much faster for min cost and min makespan problems.
  - Benders is somewhat faster for min tardiness problem.

Single-resource scheduling

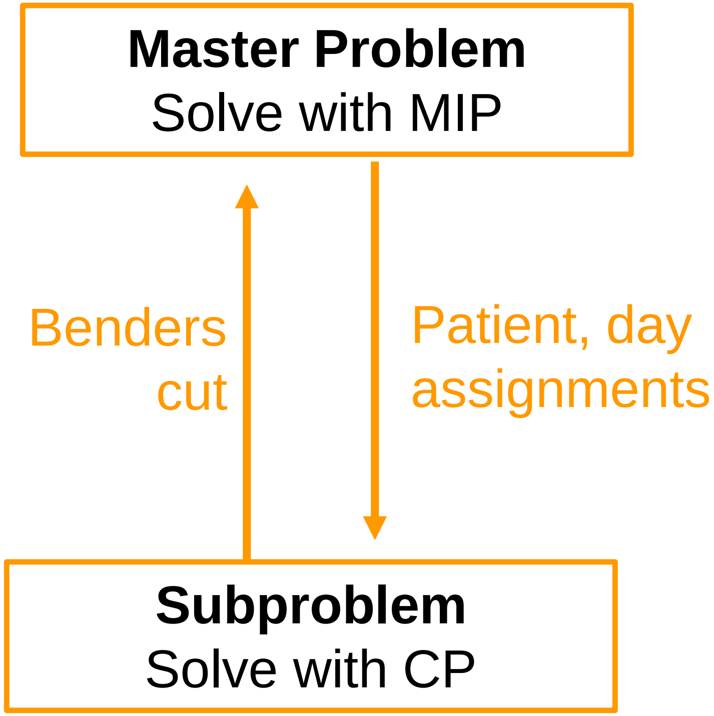
- Segmented problems:
  - Benders is much faster for min cost and min makespan problems.
  - Benders is somewhat faster for min tardiness problem.
- Unsegmented problems:
  - Benders and CP can work together.
  - Let CP run for 1 second.
  - If it fails to solve the problem, it will probably blow up. Switch to Benders for reasonably fast solution.

Home Hospice Care

- Assign aides to patients.
  - Schedule and route patient visits for each aide
    - Subject to time windows for aides and visits
    - Subject to aide qualification requirements
  - Weekly schedule
    - Number of visits per week specified for each patient
    - Must be same aide and time for each visit

Home Hospice Care

- Solve with Benders decomposition.
  - **Assign aides to patients** in master problem.
    - Maximize number of patients served by a given set of aides.



Master Problem
Solve with MIP

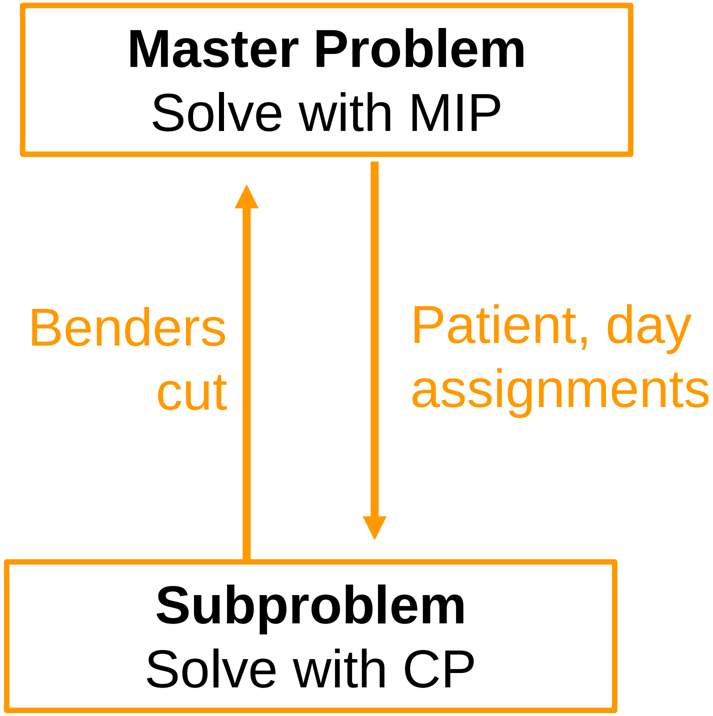
Benders cut

Patient, day assignments

Subproblem
Solve with CP

Home Hospice Care

- Solve with Benders decomposition.
  - **Assign aides to patients** in master problem.
    - Maximize number of patients served by a given set of aides.
  - **Schedule home visits** in subproblem.
    - Cyclic weekly schedule.
    - No visits on weekends.



Master Problem
Solve with MIP

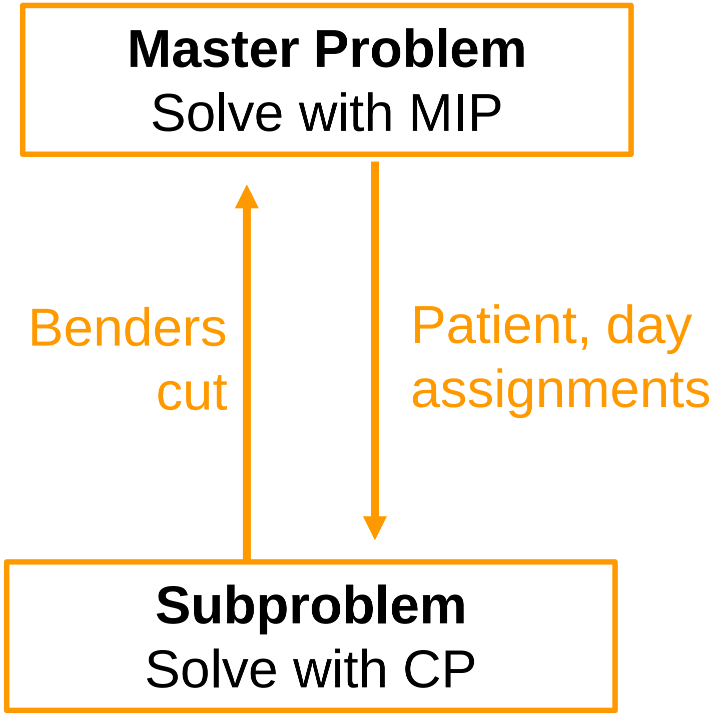
Benders cut

Patient, day assignments

Subproblem
Solve with CP

Home Hospice Care

- Solve with Benders decomposition.
  - **Assign aides to patients** in master problem.
    - Maximize number of patients served by a given set of aides.
  - **Schedule home visits** in subproblem.
    - Cyclic weekly schedule.
    - No visits on weekends.
  - Subproblem **decouples** into a scheduling problem for each aide and each day of the week.



Master Problem
Solve with MIP

Benders cut

Patient, day assignments

Subproblem
Solve with CP

Home Hospice Care

Master problem

= 1 if patient *j* scheduled

= 1 if patient *j*
assigned to aide *i*

$$\max \sum_j \boxed{\delta_j}$$

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

= 1 if patient *j*
assigned to aide *i*
on day *k*

Required number
of visits per week

$$\sum_{i,k} \boxed{y_{ijk}} = \boxed{v_j}\delta_j, \quad \text{all } j$$

$$y_{ijk} \le x_{ij}, \quad \text{all } i, j, k$$

Spacing constraints on visit days

Benders cuts

Relaxation of subproblem

$$\delta_j, x_{ij}, y_{ijk} \in \{0, 1\}$$

Home Hospice Care

- For a rolling schedule:
  - Schedule **new patients**, drop **departing patients** from schedule.
    - Provide continuity for remaining patients as follows:
  - Old patients served by **same aide** on **same days**.
    - Fix $y_{ijk} = 1$ for the relevant aides, patients, and days.

Home Hospice Care

- For a rolling schedule:
  - Schedule **new patients**, drop **departing patients** from schedule.
    - Provide continuity for remaining patients as follows:
  - Old patients served by **same aide** on **same days**.
    - Fix $y_{ijk} = 1$ for the relevant aides, patients, and days.
  - Alternative: Also served **at same time**.
    - Fix time windows to enforce their current schedule.
  - Alternative: served only by **same aide**.
    - Fix $x_{ij} = 1$ for the relevant aides, patients.

Home Hospice Care

Benders cuts

- Use strengthened nogood cuts
  - Find a smaller set of patients that create infeasibility…
    - …by re-solving the each infeasible scheduling problem repeatedly.

$$\sum_{j \in \boxed{\bar{P}_{ik}}} (1 - y_{ijk}) \geq 1$$

Reduced set of patients whose assignment to aide *i* on day *k* creates infeasibility

Home Hospice Care

Benders cuts

- Auxiliary cuts based on symmetries.
  - A cut for valid for aide *i*, day *k* is also valid for aide *i* on other days.
    - This gives rise to a large number of cuts.
  - The auxiliary cuts can be summed with sacrificing optimality.
    - Original cut ensures convergence to optimum.
    - This yields 2 cuts per aide:

$$\sum_{j \in \bar{P}_{ik}} (1 - y_{ijk}) \geq 1$$

$$\sum_{k \neq k} \sum_{j \in \bar{P}_{ik}} (1 - y_{ijk'}) \geq 4$$

Home Hospice Care

Subproblem relaxation

- Include relaxation of subproblem in the master problem.
  - Necessary for good performance.
  - Use **time window relaxation** for each scheduling problem.
  - Simplest relaxation for aide *i* and day *k*:

$$\sum_{j \in \boxed{J(a,b)}} p_j y_{ijk} \le b - a$$

Set of patients whose time window fits in interval [*a*, *b*].

Can use several intervals.

Home Hospice Care

- This relaxation is very weak.
  - Doesn’t take into account travel times.
- Improved relaxation.
  - **Basic idea:** Augment visit duration p_j with travel time to (or from) location j from **closest** patient or aide home base.
  - This is **weak** unless most assignments are **fixed**.
    - As in rolling schedule.
  - We partition day into 2 intervals.
    - Morning and afternoon.
    - Simplifies handling of aide time windows and home bases.
    - All patient time windows are in morning or afternoon.

Home Hospice Care

Time window relaxation for aide *i*, day *k* using intervals [*a*,*b*], [*b*,*c*]

$$\sum_{j \in J(a,b)} p'_{ijk} y_{ijk} \le b - a$$

$$\sum_{j \in J(b,c)} p''_{ijk} y_{ijk} \le c - b$$

where

$$[a, c] = \text{time window for aide } i$$

$$p'_{ijk} = p_j + \min\left\{t_{ij}, \min_{j' \in Q_{ik}} \{t_{j'j}\}\right\}$$

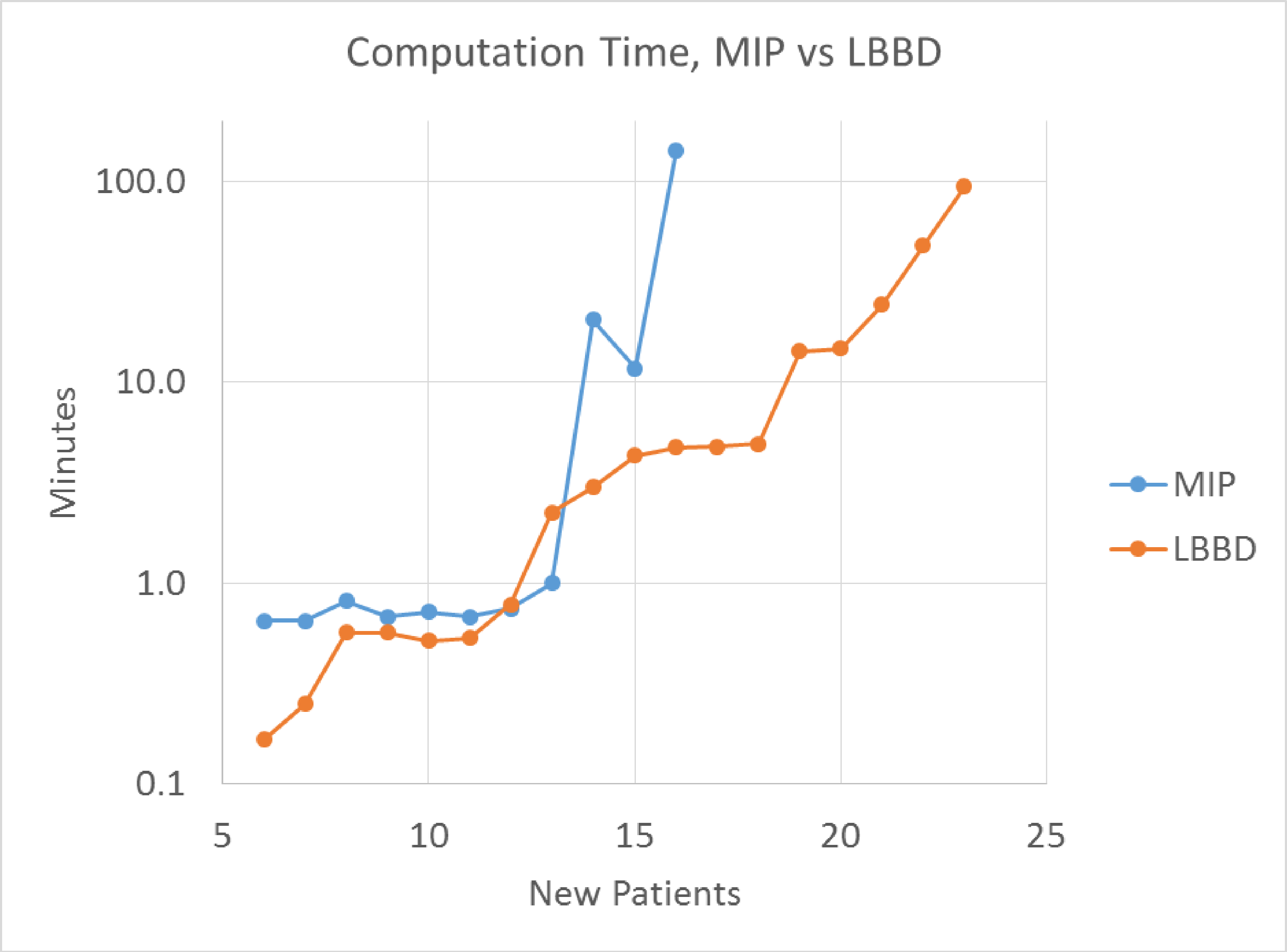
$$p''_{ijk} = p_j + \min\left\{ \min_{j' \in Q_{ik}} \{t_{jj'}\}, c\right\}$$

and where Q_{ik} = {patients unassigned or assigned to aide *i*, day *k*}

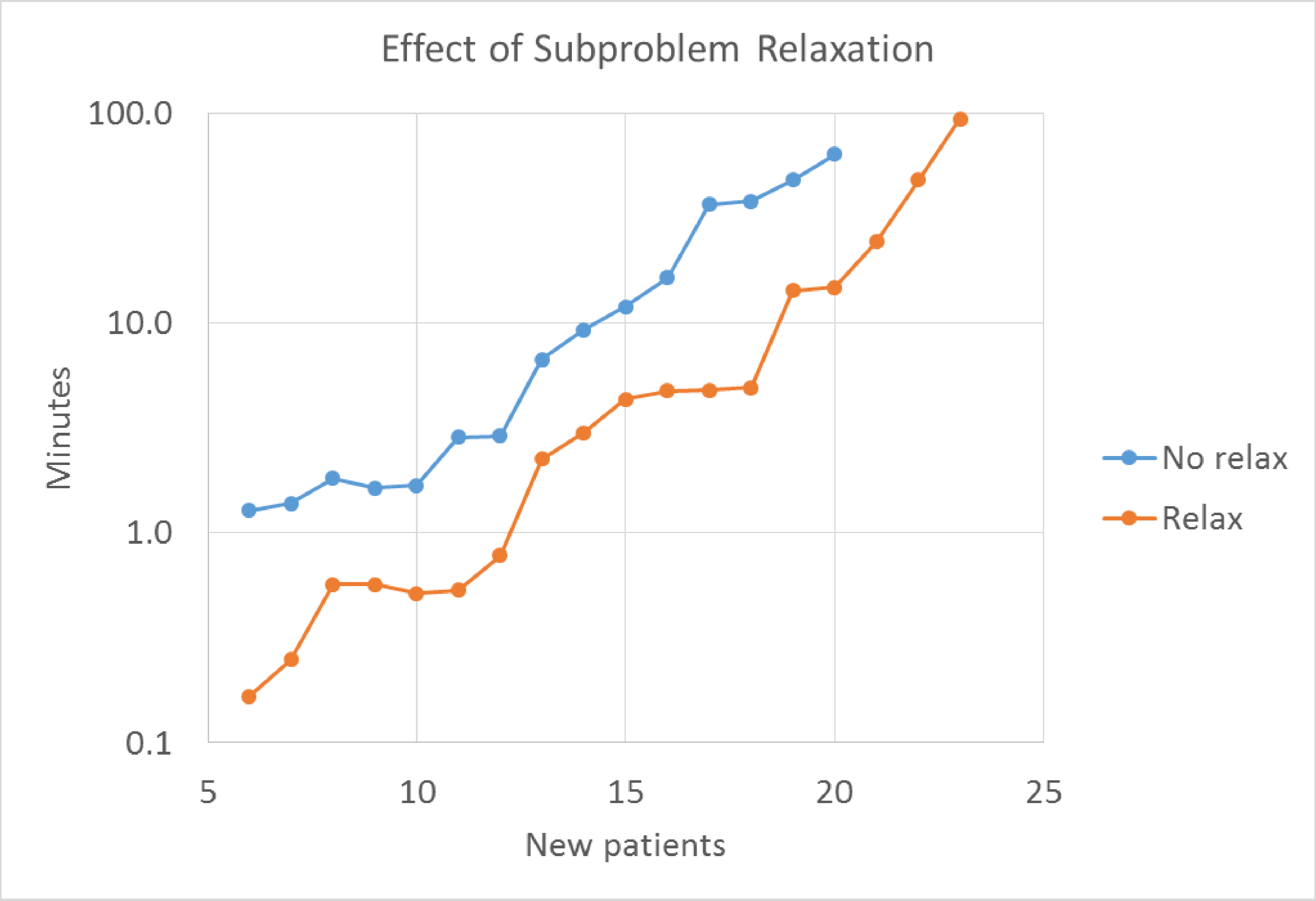
Home Hospice Care

- Instance generation
  - Start with (suboptimal) solution for the 60 patients
    - Fix this schedule for first *n* patients.
    - Schedule remaining 60 – *n* patients
  - Use 8 of the 18 aides to cover new patients
    - As well as the old patients they already cover.
    - This puts us near the phase transition.

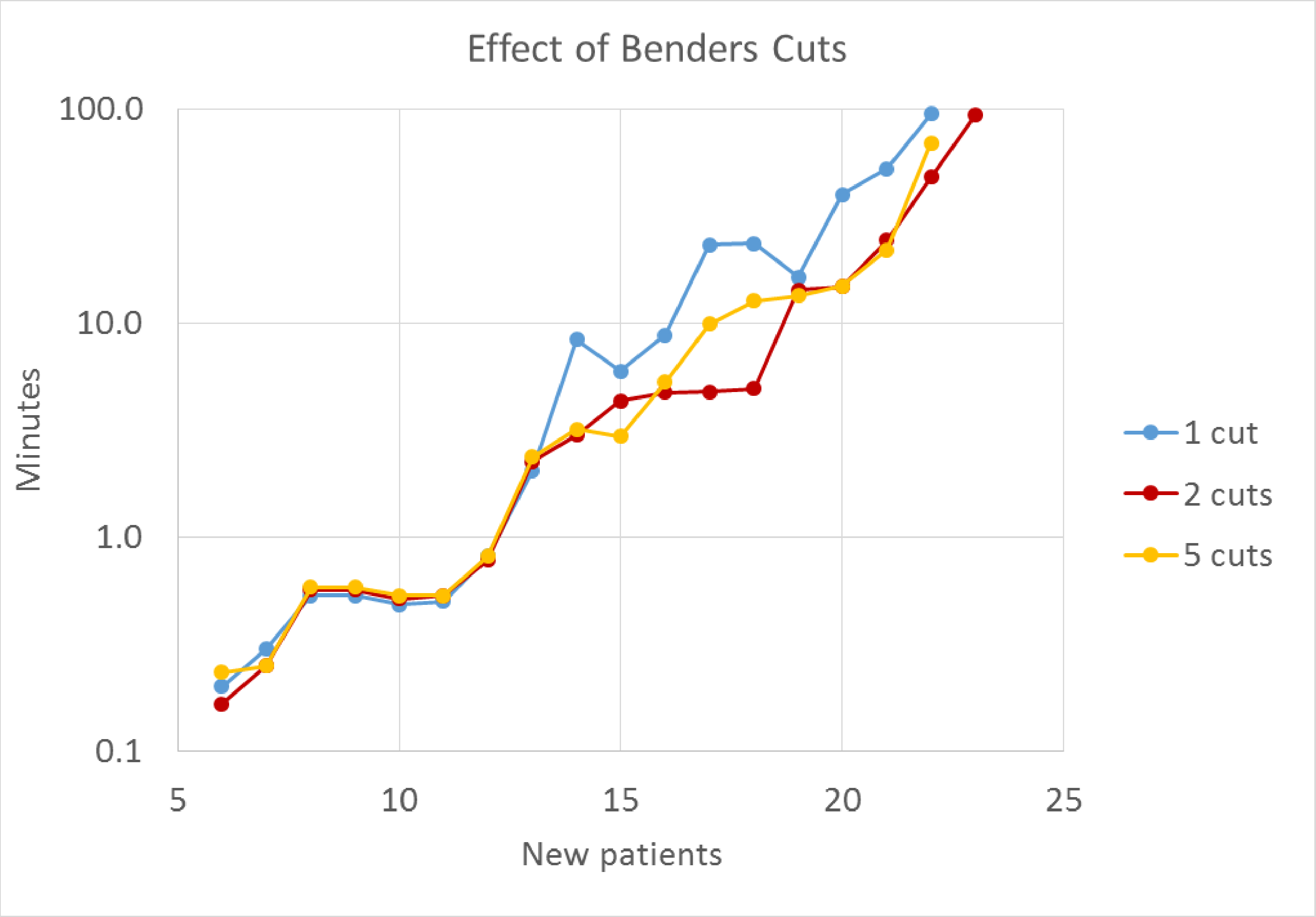
Home Hospice Care



Home Hospice Care



Home Hospice Care



Branch and check

- Generate Benders cuts at certain nodes of a branching tree
  - Variables fixed so far are search variables.
  - Unfixed variables go into subproblem.
- Not the same as branch and cut.
  - In branch and cut, the cuts contain unfixed variables.
  - In branch and check, the cuts contain fixed variables.
- When to use?
  - When master problem is the bottleneck.
  - Master is solved only once, with growing constraint set.

Inference as Projection

- Project onto propositional variables of interest
  - Suppose we wish to infer from these clauses everything we can about propositions x_1, x_2, x_3

$$
\begin{array}{|ccc|cccc}
\cline{1-3}
x_1 & & & \vee\, x_4 & \vee\, x_5 & \\
x_1 & & & \vee\, x_4 & \vee\, \bar{x}_5 & \\
x_1 & & & & \vee\, x_5 & \vee\, x_6 \\
x_1 & & & & \vee\, x_5 & \vee\, \bar{x}_6 \\
 & x_2 & & & \vee\, \bar{x}_5 & \vee\, x_6 \\
 & x_2 & & & \vee\, \bar{x}_5 & \vee\, \bar{x}_6 \\
 & & x_3 & \vee\, \bar{x}_4 & \vee\, x_5 & \\
 & & x_3 & \vee\, \bar{x}_4 & \vee\, \bar{x}_5 & \\
\cline{1-3}
\end{array}
$$

Inference as Projection

- Project onto propositional variables of interest
  - Suppose we wish to infer from these clauses everything we can about propositions x_1, x_2, x_3

$$\begin{array}{|ccc|llll}
x_1 & & & \vee\, x_4 & \vee\, x_5 & \\
x_1 & & & \vee\, x_4 & \vee\, \bar{x}_5 & \\
x_1 & & & & \vee\, x_5 & \vee\, x_6 \\
x_1 & & & & \vee\, x_5 & \vee\, \bar{x}_6 \\
 & x_2 & & & \vee\, \bar{x}_5 & \vee\, x_6 \\
 & x_2 & & & \vee\, \bar{x}_5 & \vee\, \bar{x}_6 \\
 & & x_3 & \vee\, \bar{x}_4 & \vee\, x_5 & \\
 & & x_3 & \vee\, \bar{x}_4 & \vee\, \bar{x}_5 &
\end{array}$$

We can deduce

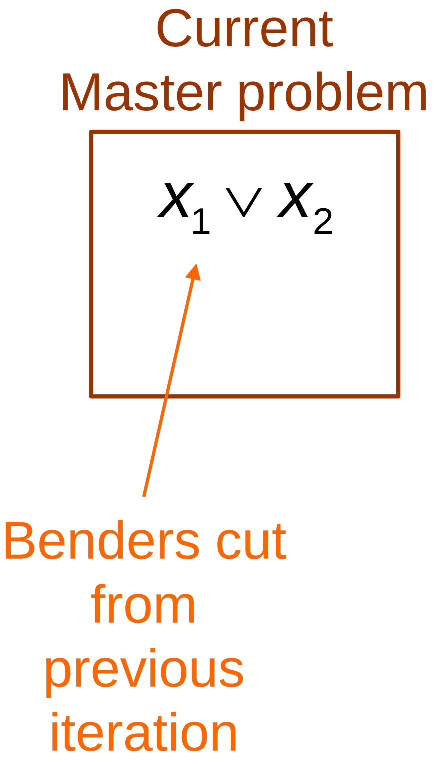
$$x_1 \vee x_2$$

$$x_1 \vee x_3$$

This is a projection onto x_1, x_2, x_3

Inference as Projection

- Benders decomposition computes a projection
  - Benders cuts describe projection onto master problem variables.



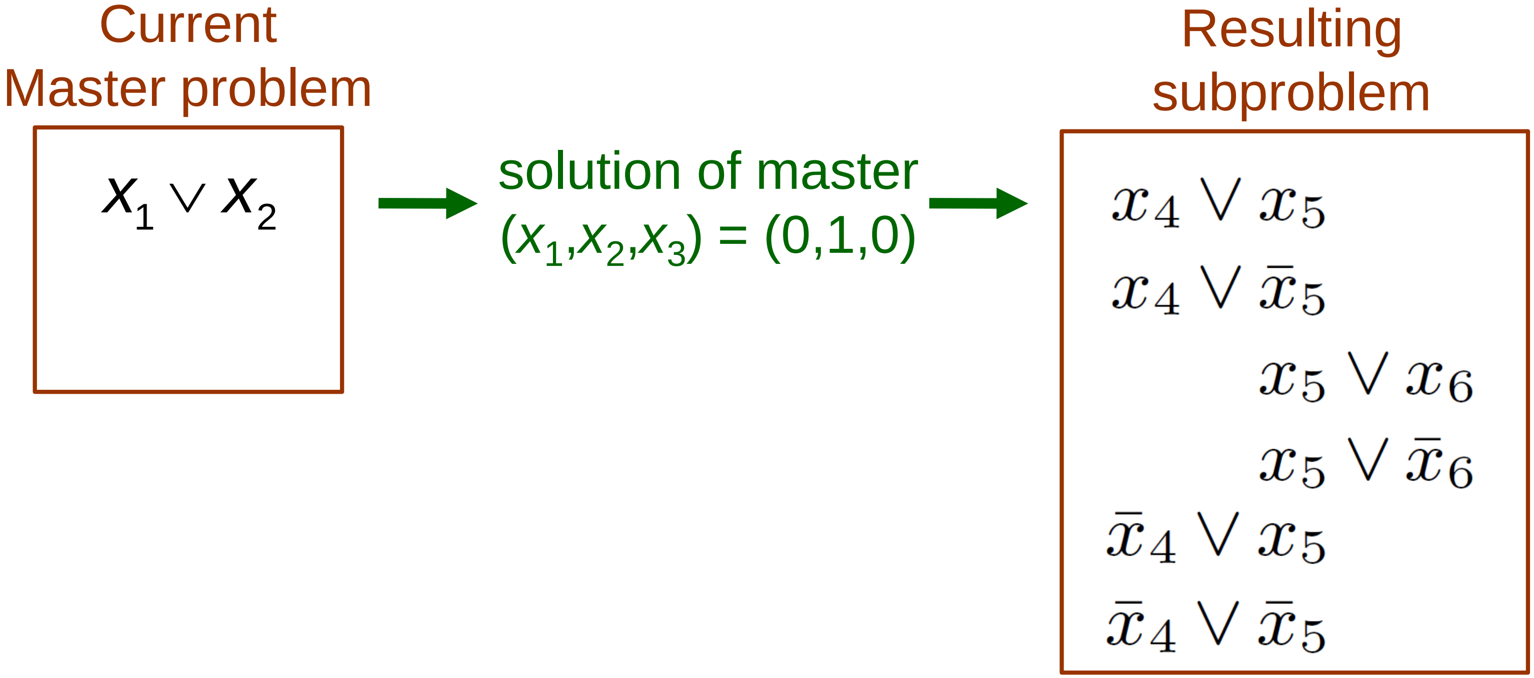
Current
Master problem

$x_1 \vee x_2$

Benders cut
from
previous
iteration

Inference as Projection

- Benders decomposition computes a projection
  - Benders cuts describe projection onto master problem variables.



Current Master problem

$x_1 \vee x_2$

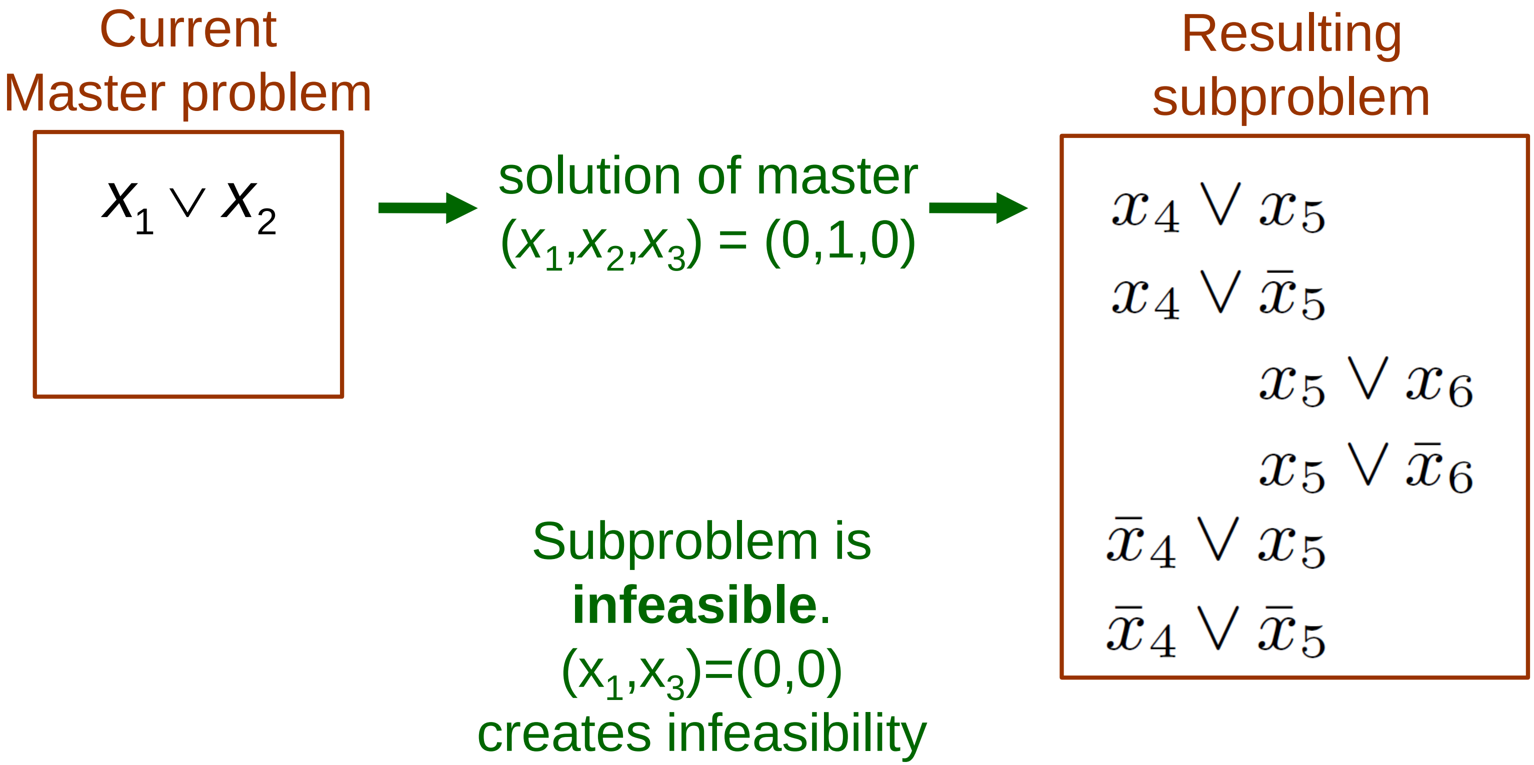
solution of master $(x_1,x_2,x_3) = (0,1,0)$

Resulting subproblem

$$\begin{array}{l} x_4 \vee x_5 \\ x_4 \vee \bar{x}_5 \\ \qquad\quad x_5 \vee x_6 \\ \qquad\quad x_5 \vee \bar{x}_6 \\ \bar{x}_4 \vee x_5 \\ \bar{x}_4 \vee \bar{x}_5 \end{array}$$

Inference as Projection

- Benders decomposition computes a projection
  - Benders cuts describe projection onto master problem variables.



Current Master problem

$x_1 \vee x_2$

solution of master
$(x_1,x_2,x_3) = (0,1,0)$

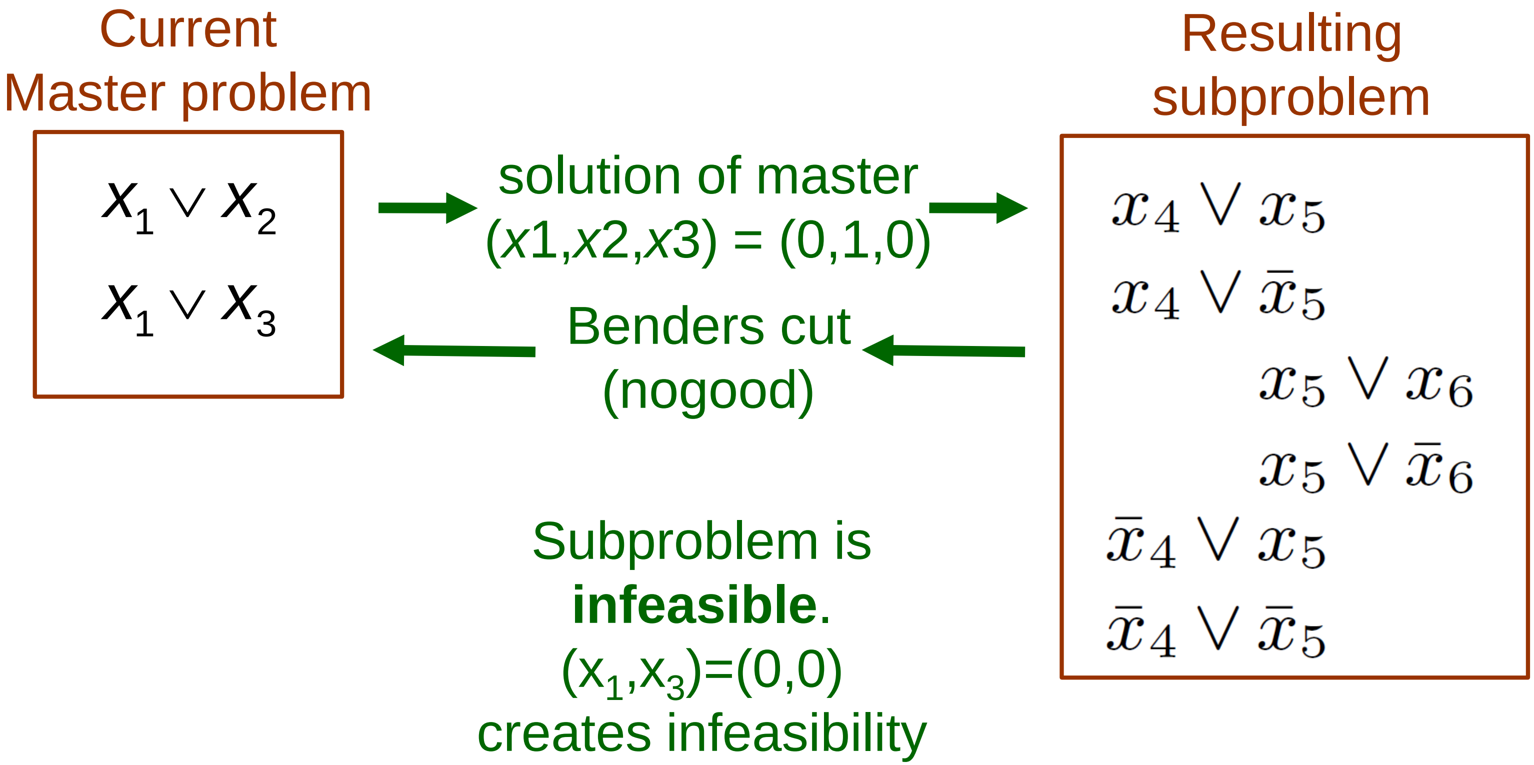
Resulting subproblem

$x_4 \vee x_5$
$x_4 \vee \bar{x}_5$
$x_5 \vee x_6$
$x_5 \vee \bar{x}_6$
$\bar{x}_4 \vee x_5$
$\bar{x}_4 \vee \bar{x}_5$

Subproblem is **infeasible**.
$(x_1,x_3)=(0,0)$
creates infeasibility

Inference as Projection

- Benders decomposition computes a projection
  - Benders cuts describe projection onto master problem variables.



Current Master problem

$x_1 \vee x_2$

$x_1 \vee x_3$

solution of master $(x1,x2,x3) = (0,1,0)$

Benders cut (nogood)

Resulting subproblem

$$x_4 \vee x_5$$
$$x_4 \vee \bar{x}_5$$
$$x_5 \vee x_6$$
$$x_5 \vee \bar{x}_6$$
$$\bar{x}_4 \vee x_5$$
$$\bar{x}_4 \vee \bar{x}_5$$

Subproblem is **infeasible**.
$(x_1,x_3)=(0,0)$
creates infeasibility

Inference as Projection

- Benders decomposition computes a projection
  - Benders cuts describe projection onto master problem variables.

Current
Master problem

$x_1 \vee x_2$

$x_1 \vee x_3$

solution of master
$(x_1,x_2,x_3) = (0,1,1)$

Resulting
subproblem

$x_4 \vee x_5$

$x_4 \vee \bar{x}_5$

$x_5 \vee x_6$

$x_5 \vee \bar{x}_6$

Inference as Projection

- Benders decomposition computes a projection
  - Benders cuts describe projection onto master problem variables.

Current
Master problem

$x_1 \vee x_2$

$x_1 \vee x_3$

solution of master
$(x_1,x_2,x_3) = (0,1,1)$

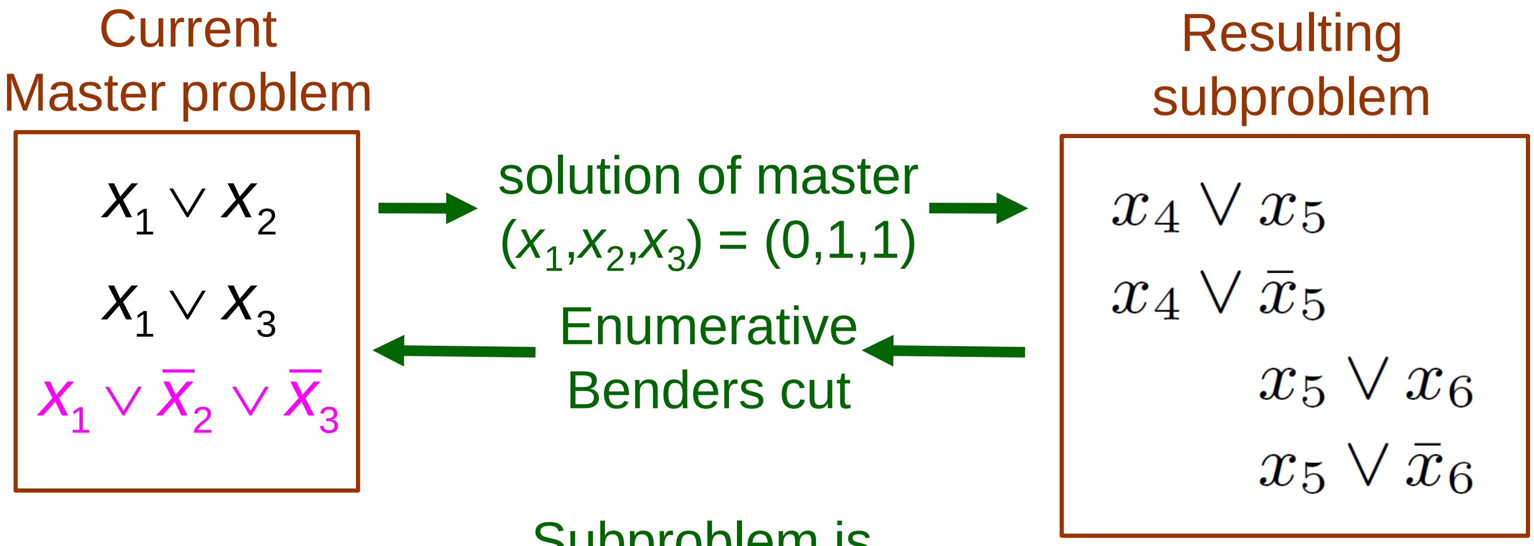
Resulting
subproblem

$$x_4 \vee x_5$$
$$x_4 \vee \bar{x}_5$$
$$x_5 \vee x_6$$
$$x_5 \vee \bar{x}_6$$

Subproblem is
feasible

Inference as Projection

- Benders decomposition computes a projection
  - Benders cuts describe projection onto master problem variables.



Current Master problem

$x_1 \vee x_2$

$x_1 \vee x_3$

$x_1 \vee \bar{x}_2 \vee \bar{x}_3$

solution of master $(x_1,x_2,x_3) = (0,1,1)$

Enumerative Benders cut

Resulting subproblem

$x_4 \vee x_5$

$x_4 \vee \bar{x}_5$

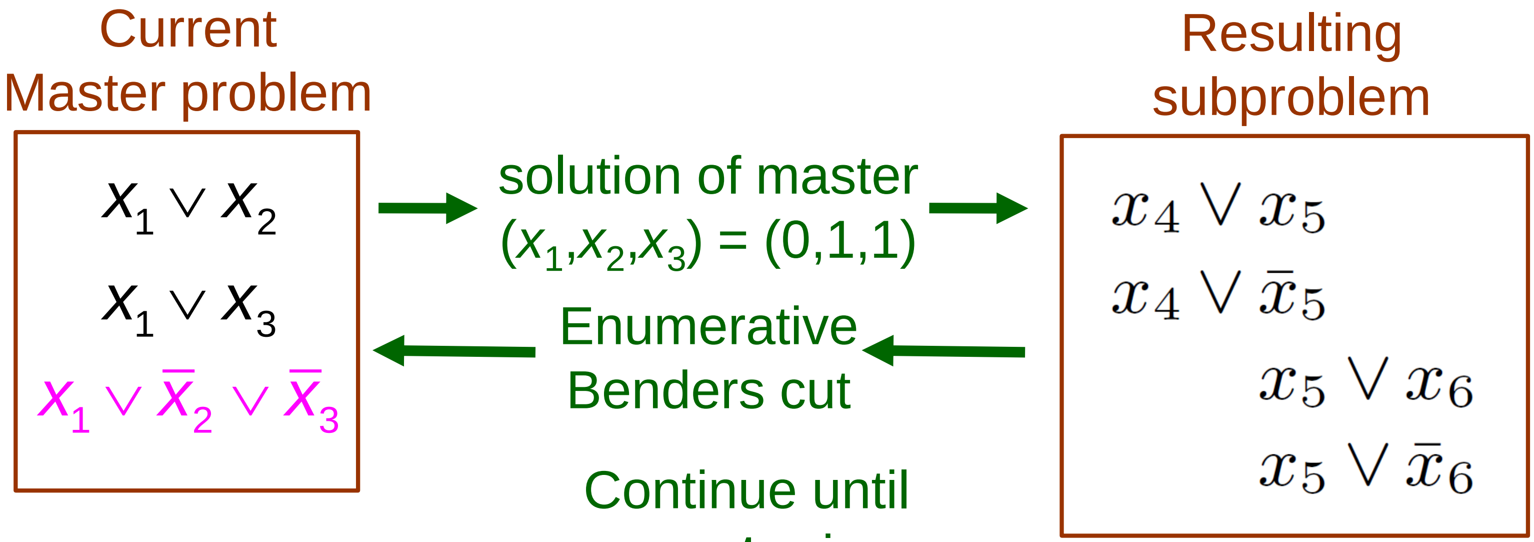
$x_5 \vee x_6$

$x_5 \vee \bar{x}_6$

Subproblem is **feasible**

Inference as Projection

- Benders decomposition computes a projection
  - Logic-based Benders cuts describe projection onto master problem variables.



Current Master problem

$x_1 \vee x_2$

$x_1 \vee x_3$

$x_1 \vee \bar{x}_2 \vee \bar{x}_3$

solution of master $(x_1,x_2,x_3) = (0,1,1)$

Enumerative Benders cut

Resulting subproblem

$x_4 \vee x_5$

$x_4 \vee \bar{x}_5$

$x_5 \vee x_6$

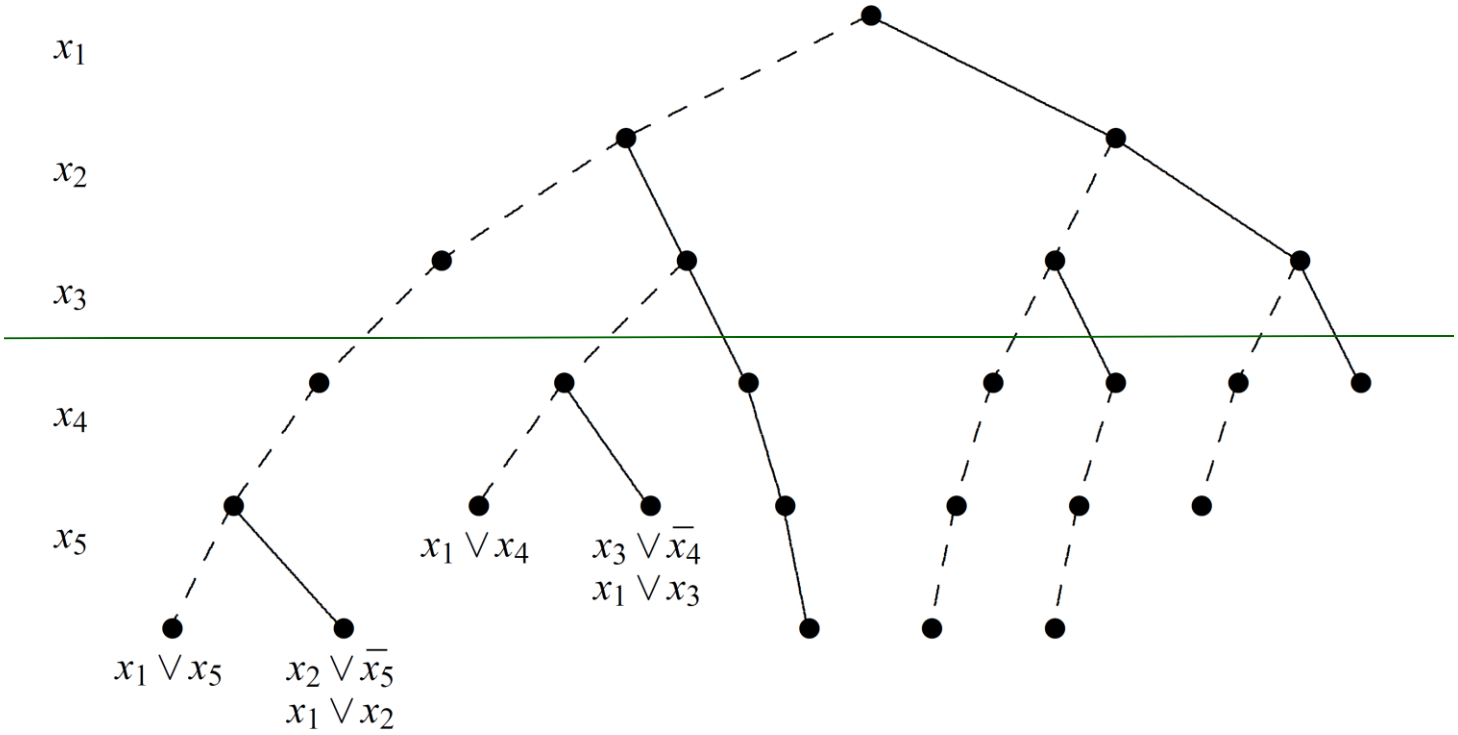
$x_5 \vee \bar{x}_6$

Continue until **master** is infeasible. Black Benders cuts describe projection.

JH and Yan (1995)
JH (2012)

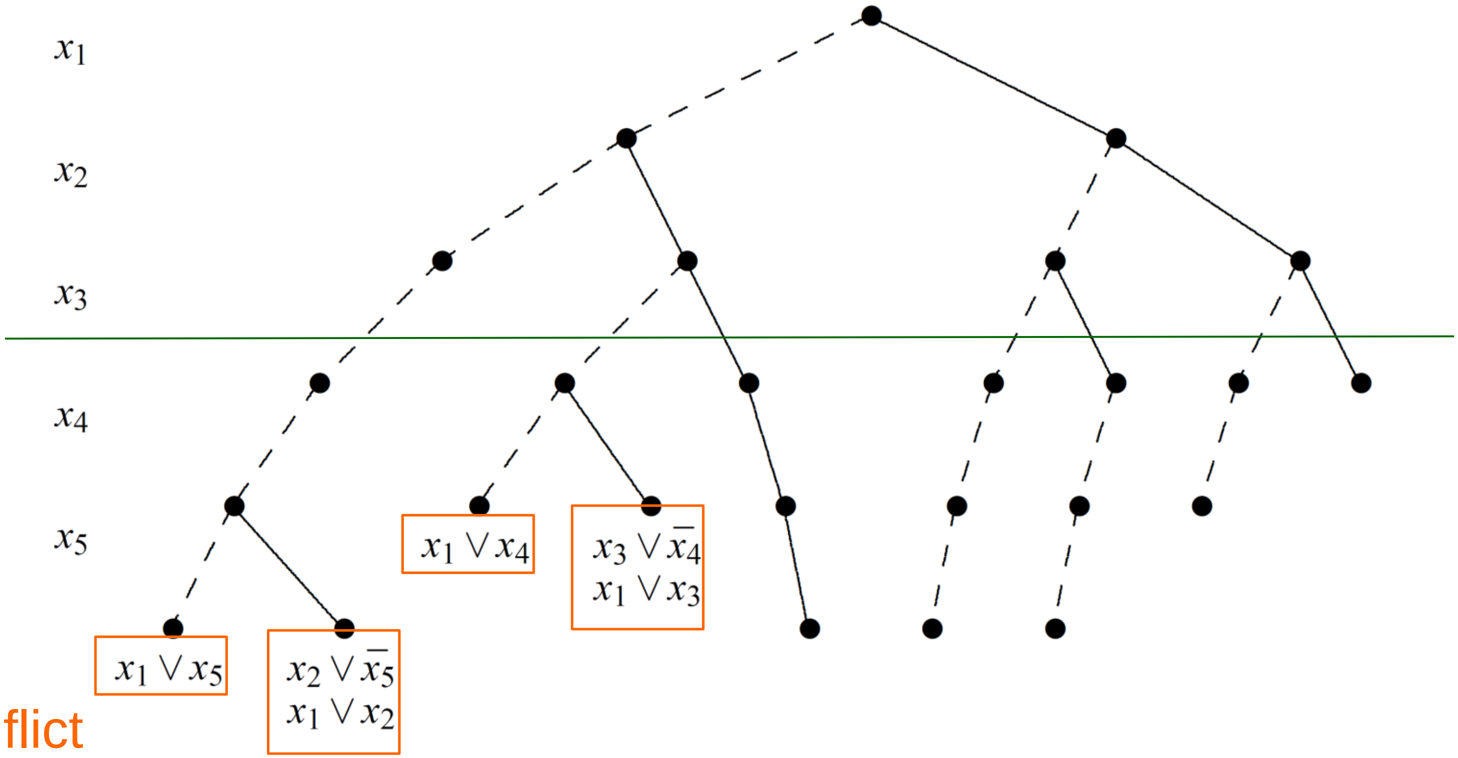
Inference as Projection

- Benders cuts = conflict clauses in a SAT algorithm
  - Branch on x_1, x_2, x_3 first.



Inference as Projection

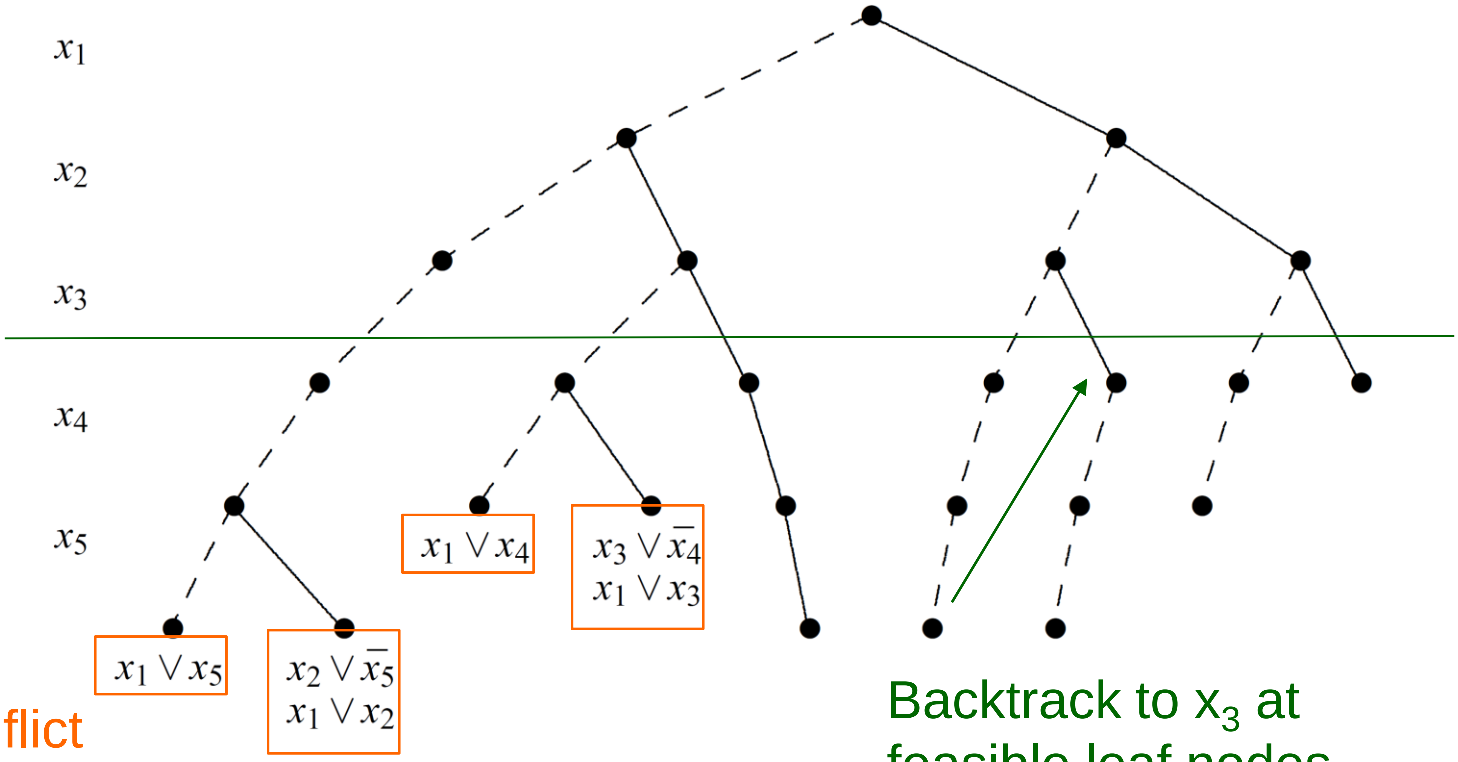
- Benders cuts = conflict clauses in a SAT algorithm
  - Branch on x_1, x_2, x_3 first.



Conflict clauses

Inference as Projection

- Benders cuts = conflict clauses in a SAT algorithm
  - Branch on x_1, x_2, x_3 first.

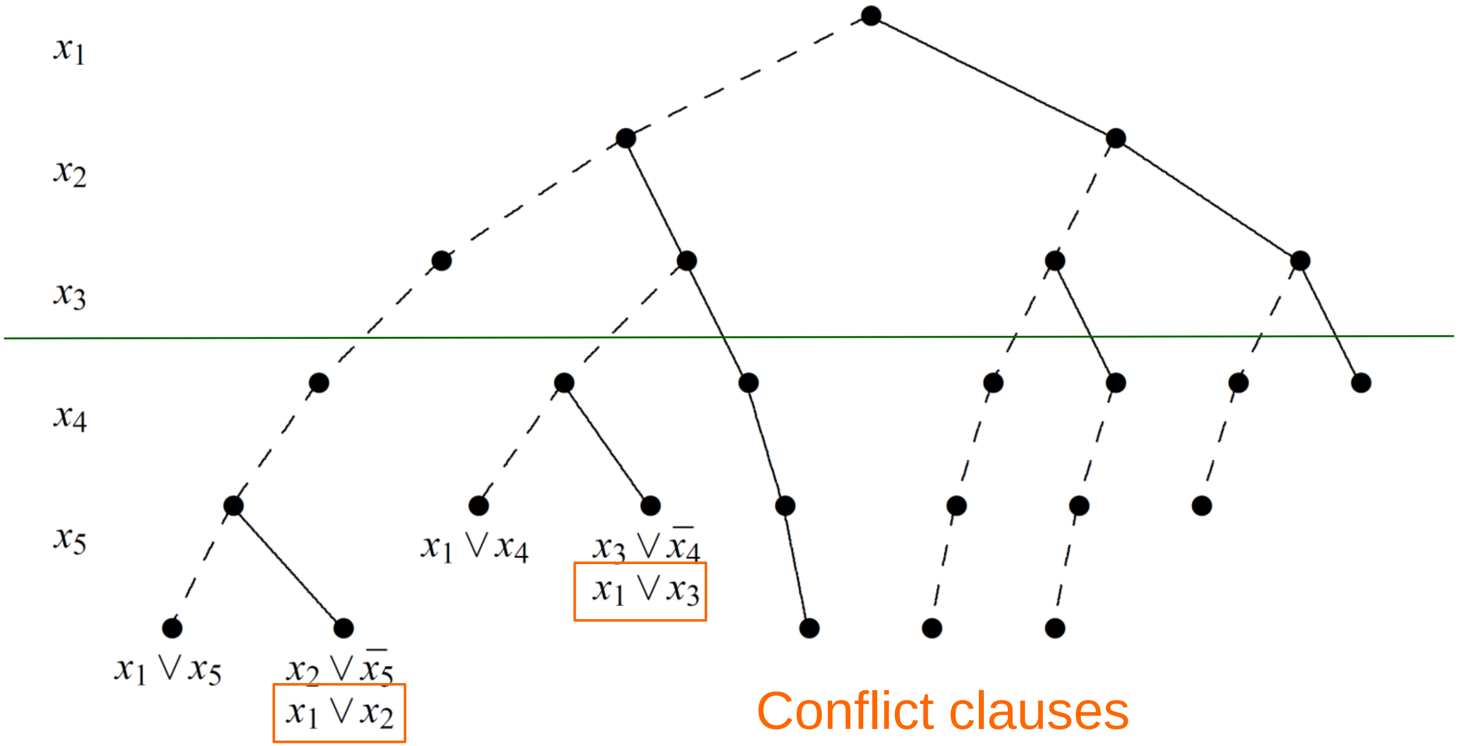


Conflict clauses

Backtrack to x_3 at feasible leaf nodes

Inference as Projection

- Benders cuts = conflict clauses in a SAT algorithm
  - Branch on x_1, x_2, x_3 first.



Conflict clauses containing x_1, x_2, x_3 describe projection

Benders decomposition [7] was introduced in 1962 to solve applications that become linear programming (LP) problems when certain *search variables* are fixed. “Generalized” Benders decomposition, proposed by Geoffrion in 1972 [25], extended the method to nonlinear programming subproblems.

Logic-based Benders decomposition (LBBD) allows the subproblem to be any optimization problem. LBBD was introduced in [32], formally developed in 2000 [33], and tested computationally in [39]. *Branch and check* is introduced in [33] and tested computationally in [69]. *Combinatorial Benders cuts* for mixed integer programming are proposed in [18].

One of the first applications [43] was a planning and scheduling problem. Updated experiments [17] show that LBBD is orders of magnitude faster than state-of-the-art MIP, with the advantage over CP even greater). Similar results have been obtained for various planning and scheduling problems [15, 21, 30, 34, 35, 37, 71].

Other successful applications of LBBD include steel production scheduling [29], inventory management [74], concrete delivery [44], shop scheduling [3, 13, 27, 28, 59], hospital scheduling [57], batch scheduling in chemical plants [49, 70], computer processor scheduling [8, 9, 12, 22, 31, 46, 47, 48, 58, 62], logic circuit verification [40], shift scheduling [5, 60], lock scheduling [73], facility location [23, 66], space packing [20, 50], vehicle routing [19, 51, 53, 56, 61, 75], bicycle sharing [45], network design [24, 52, 63, 65], home health care [16], service restoration [26], supply chain management [68], food distribution [64], queuing design and control [67], optimal control of dynamical systems [11], propositional satisfiability [1], quadratic programming [2, 41, 42], chordal completion [10], and sports scheduling [14, 54, 55, 72]. LBBD is compared with branch and check in [6]. It is implemented in the general-purpose solver SIMPL [76].

References

[1] F. Bacchus, S. Dalmao, and T. Pitassi. Relaxation search: A simple way of managing optional clauses. In *AAAI Conference on Artificial Intelligence*. 2014.

[2] L. Bai, J. E. Mitchell, and J.-S. Pang. On convex quadratic programs with linear complementarity constraints. *Computational Optimization and Applications*, 54:517–554, 2012.

[3] M. A. Bajestani and J. C. Beck. Scheduling a dynamic aircraft repair shop with limited repair resources. *Journal of Artificial Intelligence Research*, 47:35–70, 2013.

[4] P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*. Kluwer, Dordrecht, 2001.

[5] A. Y. Barlatt, A. M. Cohn, and O. Gusikhin. A hybridization of mathematical programming and dominance-driven enumeration for solving shift-selection and task-sequencing problems. *Computers and Operations Research*, 37:1298–1307, 2010.

[6] J. C. Beck. Checking up on branch-and-check. In D. Cohen, editor, *Principle and Practice of Constraint Programming (CP)*, volume 6308 of *Lecture Notes in Computer Science*, pages 84–98, 2010.

[7] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

[8] L. Benini, D. Bertozzi, A. Guerri, and M. Milano. Allocation and scheduling for MPSoCs via decomposition and no-good generation. In *Principles and Practice of Constraint Programming (CP 2005)*, volume 3709 of *Lecture Notes in Computer Science*, pages 107–121. Springer, 2005.

[9] L. Benini, M. Lombardi, M. Mantovani, M. Milano, and M. Ruggiero. Multi-stage Benders decomposition for optimizing multicore architectures. In L. Perron and M. A. Trick, editors, *CPAIOR 2008 Proceedings*, volume 5015 of *Lecture Notes in Computer Science*, pages 36–50. Springer, 2008.

[10] D. Bergman and A. U. Raghunathan. A Benders approach to the minimum chordal completion problem. In L. Michel, editor, *CPAIOR Proceedings*, volume 9075 of *Lecture Notes in Computer Science*, pages 47–64. Springer, 2015.

[11] A. H. Borzabadi and M. E. Sadjadi. Optimal control of hybrid systems by logic-based Benders decomposition. In A. Giua, C. Mahulea, M. Silva, and J. Zaytoon, editors, *Analysis and Design of Hybrid Systems*, volume 3, pages 104–107, 2009.

[12] H. Cambazard, P.-E. Hladik, A.-M. Déplanche, N. Jussien, and Y. Trinquet. Decomposition and learning for a hard real time task allocation problem. In M. Wallace, editor, *Principles and Practice of Constraint Programming (CP 2004)*, volume 3258 of *Lecture Notes in Computer Science*, pages 153–167. Springer, 2004.

[13] E. Çoban and J. N. Hooker. Single-facility scheduling by logic-based Benders decomposition. *Annals of Operations Research*, 210:245–272, 2013.

[14] K. K. H. Cheung. A Benders approach for computing lower bounds for the mirrored traveling tournament problem. *Discrete Optimization*, 6:189–196, 2009.

[15] Y. Chu and Q. Xia. A hybrid algorithm for a class of resource-constrained scheduling problems. In R. Barták and M. Milano, editors, *CPAIOR 2005 Proceedings*, volume 3524 of *Lecture Notes in Computer Science*, pages 110–124. Springer, 2005.

[16] A. Ciré and J. N. Hooker. A heuristic logic-based Benders method for the home health care problem. Presented at Matheuristics 2012, Angra dos Reis, Brazil, 2012.

[17] A. A. Ciré, E. Çoban, and J. N. Hooker. Mixed integer programming vs logic-based Benders decomposition for planning and scheduling. In C. Gomes and M. Sellmann, editors, *CPAIOR 2013 Proceedings*, pages 325–331, 2013.

[18] G. Codato and M. Fischetti. Combinatorial Benders cuts for mixed-integer linear programming. *Operations Research*, 54:756–766, 2006.

[19] A. I. Corréa, A. Langevin, and L. M. Rousseau. Dispatching and conflict-free routing of automated guided vehicles: A hybrid approach combining constraint programming and mixed integer programming. In J. C. Régin and M. Rueher, editors, *CPAIOR 2004 Proceedings*, volume 3011 of *Lecture Notes in Computer Science*, pages 370–378. Springer, 2004.

[20] J.-F. Côté, M. Dell'Amico, and M. Iori. Combinatorial Benders cuts for the strip packing problem. *Operations Research*, 62:643–661, 2014.

[21] T. O. Davies, A. R. Pearce, P. J. Stuckey, and N. Lipovetzky. Sequencing operator counts. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 61–69, 2015.

[22] A. Emeretlis, G. Theodoridis, P. Alefragis, and N. Voros. Mapping DAGs on heterogeneous platforms using logic-based Benders decompostion. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 119–124. IEEE, 2015.

[23] M. M. Fazel-Zarandi and J. C. Beck. Solving a location-allocation problem with logic-based Benders decomposition. In I. P. Gent, editor, *Principles and Practice of Constraint Programming (CP 2009)*, volume 5732 of *Lecture Notes in Computer Science*, pages 344–351, New York, 2009. Springer.

[24] B. Gendron, R. G. Garroppo, G. Nencioni, M. G. Scutellà, and L. Tavanti. Benders decomposition for a location-design problem in green wireless local area networks. *Electronic Notes in Discrete Mathematics*, 41:367–374, 2013.

[25] A. M. Geoffrion. Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10:237–260, 1972.

[26] J. Gong, E. E. Lee, J. E. Mitchell, and W. A. Wallace. Logic-based multiobjective optimization for restoration planning. In W. Chaovalitwongse, K. C. Furman, and P. M. Pardalos, editors, *Optimization and Logistics Challenges in the Enterprise*, pages 305–324. 2009.

[27] O. Guyon, P. Lemaire, E. Pinson, and D. Rivreau. Solving an integrated job-shop problem with human resource constraints. *Annals of Operations Research*, 213:147–171, 2014.

[28] I. Hamdi and T. Loukil. Logic-based Benders decomposition to solve the permutation flowshop scheduling problem with time lags. In *International Conference on Modeling, Simulation and Applied Optimization (ICMSAO)*, pages 1–7. IEEE, 2013.

[29] I. Harjunkoski and I. E. Grossmann. A decomposition approach for the scheduling of a steel plant production. *Computers and Chemical Engineering*, 25:1647–1660, 2001.

[30] I. Harjunkoski and I. E. Grossmann. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Computers and Chemical Engineering*, 26:1533–1552, 2002.

[31] P.-E. Hladik, H. Cambazard, A.-M. Déplanche, and N. Jussien. Solving a real-time allocation problem with constraint programming. *Journal of Systems and Software*, 81:132—149, 2008.

[32] J. N. Hooker. Logic-based Benders decomposition. In *INFORMS National Meeting (INFORMS 1995)*, 1995.

[33] J. N. Hooker. *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. Wiley, New York, 2000.

[34] J. N. Hooker. A hybrid method for planning and scheduling. *Constraints*, 10:385–401, 2005.

[35] J. N. Hooker. An integrated method for planning and scheduling to minimize tardiness. *Constraints*, 11:139–157, 2006.

[36] J. N. Hooker. *Integrated Methods for Optimization*. Springer, 2007.

[37] J. N. Hooker. Planning and scheduling by logic-based Benders decomposition. *Operations Research*, 55:588–602, 2007.

[38] J. N. Hooker. *Integrated Methods for Optimization, 2nd ed.* Springer, 2012.

[39] J. N. Hooker and G. Ottosson. Logic-based Benders decomposition. *Mathematical Programming*, 96:33–60, 2003.

[40] J. N. Hooker and H. Yan. Logic circuit verification by Benders decomposition. In V. Saraswat and P. Van Hentenryck, editors, *Principles and Practice of Constraint Programming: The Newport Papers*, pages 267–288, Cambridge, MA, 1995. MIT Press.

[41] J. Hu, J. E. Mitchell, and J.-S. Pang. An LPCC approach to nonconvex quadratic programs. *Mathematical Programming*, 133:243–277, 2012.

[42] J. Hu, J. E. Mitchell, J.-S. Pang, K. P. Bennett, and G. Kunapuli. On the global solution of linear programs with linear complementarity constraints. *SIAM Journal on Optimization*, 19:445–471, 2008.

[43] V. Jain and I. E. Grossmann. Algorithms for hybrid MILP/CP models for a class of optimization problems. *INFORMS Journal on Computing*, 13:258–276, 2001.

[44] J. Kinable and M. Trick. A logic-based Benders approach to the concrete delivery problem. In H. Simonis, editor, *CPAIOR Proceedings*, volume 8451 of *Lecture Notes in Computer Science*, pages 176–192. Springer, 2014.

[45] C. Kloimüllner, P. Papazek, B. Hu, and G. R. Raidl. A cluster-first route-second approach for balancing bicycle sharing systems. In *International Conference on Computer Aided Systems Theory (EUROCAST)*, volume 9520 of *Lecture Notes in Computer Science*, pages 439–446. Springer, 2015.

[46] W. Liu, Z. Gu, J. Xu, X. Wu, and Y. Ye. Satisfiability modulo graph theory for task mapping and scheduling on multiprocessor systems. *IEEE Transactions on Parallel and Distributed Systems*, 22:1382–1389, 2011.

[47] W. Liu, M. Yuan, X. He, Z. Gu, and X. Liu. Efficient SAT-based mapping and scheduling of homogeneous synchronous dataflow graphs for throughput optimization. In *Real-Time Systems Symposium*, pages 492–504. IEEE, 2008.

[48] M. Lombardi, M. Milano, M. Ruggiero, and L. Benini. Stochastic allocation and scheduling for conditional task graphs in multi-processor systems-on-chip. *Journal of Scheduling*, 13:315–345, 2010.

[49] C. T. Maravelias and I. E. Grossmann. Using MILP and CP for the scheduling of batch chemical processes. In J. C. Régin and M. Rueher, editors, *CPAIOR 2004 Proceedings*, volume 3011 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2004.

[50] J. Maschler and G. Raidl. Logic-based Benders decomposition for the 3-staged strip packing problem. In *International Conference on Operations Research (German OR Society)*, 2015.

[51] T. Nishi, Y. Hiranaka, and I. E. Grossmann. A bilevel decomposition algorithm for simultaneous production scheduling and conflict-free routing for automated guided vehicles. *Computers and Operations Research*, 38:876–888, 2011.

[52] B. Peterson and M. Trick. A Benders' approach to a transportation network design problem. In W.-J. van Hoeve and J. N. Hooker, editors, *CPAIOR 2009 Proceedings*, volume 5547 of *Lecture Notes in Computer Science*, pages 326–327, New York, 2009. Springer.

[53] G. R. Raidl, T. Baumhauer, and B. Hu. Speeding up logic-based Benders decomposition by a metaheuristic for a bi-level capacitated vehicle routing problem. In *International Workshop on Hybrid Metaheuristics*, volume 8457 of *Lecture Notes in Computer Science*, pages 183–197. Springer, 2014.

[54] R. V. Rasmussen. Scheduling a triple round robin tournament for the best Danish soccer league. *European Journal of Operational Research*, 20:795–810, 2008.

[55] R. V. Rasmussen and M. A. Trick. A Benders approach to the constrained minimum break problem. *European Journal of Operational Research*, 177:198–213, 2007.

[56] S. Riazi, C. Seatzu, O. Wigstrom, and B. Lennartson. Benders/gossip methods for heterogeneous multi-vehicle routing problems. In *IEEE Conference on Emerging Technologies Factory Automation (ETFA)*, pages 1–6, 2013.

[57] V. Roshanaei, D. M. Aleman, and D. Urbach. Logic-based Benders decomposition approaches with application to operating room scheduling. In *INFORMS National Meeting*, 2015.

[58] M. Ruggiero, A. Guerri, D. Bertozzi, F. Poletti, and M. Milano. Communication-aware allocation and scheduling framework for stream-oriented multi-processor systems-on-chip. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 3–8. European Design and Automation Association, 2006.

[59] R. Sadykov. A hybrid branch-and-cut algorithm for the one-machine scheduling problem. In J. C. Régin and M. Rueher, editors, *CPAIOR Proceedings*, volume 3011 of *Lecture Notes in Computer Science*, pages 409–415. Springer, 2004.

[60] D. Salvagnin and T. Walsh. A hybrid MIP/CP approach for multi-activity shift scheduling. In M. Milano, editor, *Principles and Practice of Constraint Programming*, volume 7514 of *Lecture Notes in Computer Science*, pages 633–646. Springer, 2012.

[61] R. Sarmad, O. Wigström, and S. Carla. Benders/gossip methods for heterogeneous multi-vehicle routing problems. In *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–6. IEEE, 2013.

[62] N. Satish, K. Ravindran, and K. Keutzer. A decomposition-based constraint optimization approach for statically scheduling task graphs with communication delays to multiprocessors. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 57–62. EDA Consortium, 2007.

[63] S. Shen and J. C. Smith. A decomposition approach for solving a broadcast domination network design problem. *Annals of Operations Research*, 210:333–360, 2011.

[64] S. Solak, C. Scherrer, and A. Ghoniem. The stop-and-drop problem in nonprofit food distribution networks. *Annals of Operations Research*, 221:407–426, 2014.

[65] Z. C. Taşkın, J. C. Smith, S. Ahmed, and A. J. Schaefer. Cutting plane algorithms for solving a stochastic edge-partition problem. *Discrete Optimization*, 6:420–435, 2009.

[66] S. Tarim, S. Armagan, and I. Miguel. A hybrid Benders decomposition method for solving stochastic constraint programs with linear recourse. In B. Hnich, M. Carlsson, F. Fages, and F. Rossi, editors, *International Workshop on Constraint Solving and Constraint Logic Programming (CSCLP)*, pages 133–148. Springer, 2006.

[67] D. Terekhov, J. C. Beck, and K. N. Brown. Solving a stochastic queueing design and control problem with constraint programming. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI 2007)*, volume 1, pages 261–266. AAAI Press, 2007.

[68] D. Terekhov, M. K. Doğru, U. Özen, and J. C. Beck. Solving two-machine assembly scheduling problems with inventory constraints. *Computers and Industrial Engineering*, 63:120–134, 2012.

[69] E. Thorsteinsson. Branch and check: A hybrid framework integrating mixed integer programming and constraint logic programming. In T. Walsh, editor, *Principles and Practice of Constraint Programming (CP 2001)*, volume 2239 of *Lecture Notes in Computer Science*, pages 16–30. Springer, 2001.

[70] C. Timpe. Solving planning and scheduling problems with combined integer and constraint programming. *OR Spectrum*, 24:431–448, 2002.

[71] T. T. Tran and J. C. Beck. Logic-based Benders decomposition for alternative resource scheduling with sequence dependent setups. In *European Conference on Artificial Intelligence (ECAI)*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 774–779. IOS Press, 2012.

[72] M. Trick and H. Yildiz. Benders cuts guided large neighborhood search for the traveling umpire problem. In P. Van Hentenryck and L. Wolsey, editors, *CPAIOR Proceedings*, volume 4510 of *Lecture Notes in Computer Science*, pages 332–345. Springer, 2007.

[73] J. Verstichel, J. Kinable, P. De Causmaecker, and G. Vanden Berghe. A combinatorial Benders decomposition for the lock scheduling problem. *Computers and Operations Research*, 54:117–128, 2015.

[74] D. Wheatley, F. Gzara, and E. Jewkes. Logic-based Benders decomposition for an inventory-location problem with service constraints. *Omega*, 55:10–23, 2015.

[75] Q. Xia, A. Eremin, and M. Wallace. Problem decomposition for traffic diversions. In J. C. Régin and M. Rueher, editors, *CPAIOR 2004 Proceedings*, volume 3011 of *Lecture Notes in Computer Science*, pages 348–363. Springer, 2004.

[76] T. H. Yunes, I. Aron, and J. N. Hooker. An integrated solver for optimization problems. *Operations Research*, 58:342–356, 2010.