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.

Subproblem
Simplified problem contains remaining variables $y$
Solve inference dual to obtain Benders cut that excludes solutions no better than current one.

Fix search variables $x$
Add Benders cut
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

Home

City 1

City 2

City 3

City 4

Village

High Pass

$100

$200

$200

$100

By air

By bus
Let $x = \text{flight destination}$ $\quad$ Find cheapest route $(x,y)$

$y = \text{bus route}$
Let $x =$ flight destination
$y =$ bus route

Master problem
Solve for cheapest flight $x$
...subject to Benders cuts generated so far

Subproblem
Find cheapest route $(x, y)$

Find cheapest bus route from airport to village.
Use proof of optimality to bound cost of other flights $x$. 

Fix flight $x$
Add Benders cut
Let $x =$ flight destination $\quad$ Find cheapest route $(x,y)$
$y =$ bus route

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

\[
\text{cost} \geq \text{airfare} + \text{bus from city to High Pass} + \$150
\]

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

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

Pick the city $x$ to minimize cost subject to

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

Clearly the solution is $x = \text{City 4}$, with cost $100$. 
Now let $x = \text{City 4}$ and pose the \textbf{subproblem}:

Find the cheapest route given that $x = \text{City 4}$. Optimal cost is $100 + 250 = \$350$. 

![Diagram showing the cheapest route from Home to Village via City 4, with costs marked on the routes.]
Again solve the master problem:

Pick the city \( x \) to minimize cost subject to

\[
\begin{align*}
\text{cost} &\geq B_{\text{City } 1}(x) = \begin{cases} 
$100 + 80 + 150 & \text{if } x = \text{City 1} \\
$200 + 150 & \text{if } x = \text{City 2,3} \\
$100 & \text{if } x = \text{City 4}
\end{cases} \\
\text{cost} &\geq B_{\text{City } 4}(x) = \begin{cases} 
$350 & \text{if } x = \text{City 1} \\
$0 & \text{otherwise}
\end{cases}
\end{align*}
\]

The solution is \( x = \text{City 1} \), 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
  \[ \min f(x, y) \]
  \[ (x, y) \in S \]

Iteration \( k \):

Master problem

\[
\begin{align*}
\min z \\
z \geq B_{x_i}(x), \; i \leq k - 1
\end{align*}
\]

Minimize cost \( z \) subject to Benders cuts

Subproblem

\[
\begin{align*}
\min f(x^k, y) \\
(x^k, y) \in S
\end{align*}
\]

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 \leq \text{optimal value} \leq \text{smallest subproblem value so far}

  • Continue until equality is obtained.

Master problem

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

Minimize cost \( z \) subject to Benders cuts

Trial value \( x^k \) that solves master

Subproblem

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

Solve \textit{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

\[
\begin{align*}
\min & \quad z \\
\text{s.t.} & \quad z \geq B_{x^i}(x), \ i \leq k - 1 \\
\end{align*}
\]

Minimize cost \( z \) subject to Benders cuts

Subproblem

\[
\begin{align*}
\min & \quad f(x^k, y) \\
( & \quad x^k, y \in S \\
\end{align*}
\]

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**.
Primal problem: optimization
\[ \min f(x) \]
\[ x \in S \]
Find **best** feasible solution by searching over **values** of \( x \).

Dual problem: Inference
\[ \max \nu \]
\[ x \in S \overset{P}{\Rightarrow} f(x) \geq \nu \]
\[ P \in \mathcal{P} \]
Find a proof of optimal value \( \nu^* \) by searching over **proofs** \( P \).
Inference Dual

- **Weak duality** always holds:

\[
\text{Min value of primal problem} \geq \text{Max value of dual problem}
\]

Difference = duality gap
Strong duality sometimes holds:

Min value of primal problem = Max value of dual problem

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

\[
\begin{align*}
\min & \quad cx \\
Ax & \geq b \\
x & \geq 0
\end{align*}
\]

Inference dual

\[
\begin{align*}
\max & \quad \nu \\
\left( \begin{array}{c}
Ax \geq b \\
x \geq 0
\end{array} \right) & \quad P \\
\nu & \geq cx
\end{align*}
\]
**Classical LP Dual**

Primal problem

\[
\begin{align*}
\text{min } & cx \\
A_x & \geq b \\
x & \geq 0
\end{align*}
\]

Inference dual

\[
\begin{align*}
\text{max } & \nu \\
\left(\begin{array}{c}
A_x \\
x
\end{array}\right) & \geq \begin{array}{c}
b \\
0
\end{array}^P \\
\Rightarrow & cx \geq \nu \\
P & \in \mathcal{P}
\end{align*}
\]

Proof family \(\mathcal{P}\) :

\[
\left(\begin{array}{c}
A_x \\
x
\end{array}\right) \geq \begin{array}{c}
b \\
0
\end{array}^P \\
\Rightarrow & cx \geq \nu \\
\text{when } & uA_x \geq ub \text{ dominates } cx \geq \nu \\
\text{for some } u & \geq 0
\]

Assuming \(Ax \geq b, x \geq 0\) is feasible.
Classical LP Dual

Primal problem

\[
\begin{align*}
\min & \quad cx \\
Ax & \geq b \\
x & \geq 0
\end{align*}
\]

Inference dual

\[
\begin{align*}
\max & \quad \nu \\
\left(\begin{array}{c}
Ax \geq b \\
x \geq 0
\end{array}\right) & \quad \Rightarrow \quad cx \geq \nu \\
P & \in \mathcal{P}
\end{align*}
\]

Proof family \(\mathcal{P}\) :

\[
\left(\begin{array}{c}
Ax \geq b \\
x \geq 0
\end{array}\right) \quad \Rightarrow \quad cx \geq \nu
\]

Assuming \(Ax \geq b, x \geq 0\) is feasible.

when \(uAx \geq ub\) dominates \(cx \geq \nu\)

for some \(u \geq 0\)

\[
\begin{align*}
uA & \leq c \\
ub & \geq \nu
\end{align*}
\]
Classical LP Dual

Primal problem

\[
\begin{aligned}
\min & \quad cx \\
A x & \geq b \\
x & \geq 0
\end{aligned}
\]

Inference dual

\[
\begin{aligned}
\max & \quad \nu \\
(A x \geq b) & \Rightarrow cx \geq \nu \\
P & \in \mathcal{P}
\end{aligned}
\]

Proof family \( \mathcal{P} \):

\[
\begin{aligned}
\left( \begin{array}{c}
A x \geq b \\
x \geq 0
\end{array} \right)^P & \Rightarrow cx \geq \nu \\
\text{when} & \quad uA x \geq ub \quad \text{dominates} \\
& \quad cx \geq \nu \\
& \quad \text{for some} \quad u \geq 0
\end{aligned}
\]

Assuming \( A x \geq b, \ x \geq 0 \) is feasible.

This is a complete inference method (due to Farkas Lemma)
Classical LP Dual

**Primal problem**

\[
\begin{align*}
\min\ cx & \\
Ax & \geq b & \\
x & \geq 0 \\
\end{align*}
\]

**Inference dual**

\[
\begin{align*}
\max\ v & \\
\left(\begin{array}{c}
Ax \geq b \\
x \geq 0
\end{array}\right) & \Rightarrow cx \geq v \\
\end{align*}
\]

\[P \in \mathcal{P}\]

\[
\begin{align*}
\max\ v & \\
uA & \leq c & \\
ub & \geq v & \\
u & \geq 0 & \\
\end{align*}
\]

Proof family \(\mathcal{P}\):

\[
\left(\begin{array}{c}
Ax \geq b \\
x \geq 0
\end{array}\right) & \Rightarrow cx \geq v
\]

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

Assuming \(Ax \geq b, x \geq 0\) is feasible.

This is a **complete** inference method

(due to Farkas Lemma)
Classical LP Dual

Primal problem

\[
\begin{align*}
\min & \quad cx \\
Ax & \geq b \\
x & \geq 0
\end{align*}
\]

Inference dual

\[
\begin{align*}
\max & \quad ub \\
uA & \leq c \\
u & \geq 0
\end{align*}
\]

\[
= \begin{align*}
\max & \quad v \\
uA & \leq c \\
ub & \geq v \\
u & \geq 0
\end{align*}
\]

Proof family \( \mathcal{P} \): 

\[
\begin{pmatrix} 
Ax & \geq b \\
x & \geq 0 
\end{pmatrix}^P \Rightarrow cx \geq v
\]

when \( uAx \geq ub \) dominates for some \( u \geq 0 \)

Assuming \( Ax \geq b, x \geq 0 \) is feasible.

This is a complete inference method (due to Farkas Lemma)
Classical LP Dual

Primal problem

\[
\begin{align*}
\min & \quad cx \\
Ax & \geq b \\
x & \geq 0
\end{align*}
\]

Classical LP dual

\[
\begin{align*}
\max & \quad ub \\
uA & \leq c \\
u & \geq 0
\end{align*}
\]

A strong dual due to Farkas Lemma

...assuming \(Ax \geq b, \ x \geq 0\) is feasible

Proof family \(P\):

\[
\begin{align*}
\left( \begin{array}{c}
Ax \geq b \\
x \geq 0
\end{array} \right) & \Rightarrow cx \geq v \\
\text{when} & \quad uAx \geq ub \text{ dominates } cx \geq v \\
& \quad \text{for some } u \geq 0
\end{align*}
\]

Assuming \(Ax \geq b, \ x \geq 0\) is feasible.

This is a complete inference method (due to Farkas Lemma)
## Inference Duals

<table>
<thead>
<tr>
<th>Problem</th>
<th>Inference Method</th>
<th>Inference dual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear programming</td>
<td>Linear combination + domination</td>
<td>Classical LP dual (strong)</td>
</tr>
<tr>
<td>Inequality constrained optimization</td>
<td>Linear combination + implication</td>
<td>Surrogate dual</td>
</tr>
<tr>
<td>Inequality constrained optimization</td>
<td>Linear combination + domination</td>
<td>Lagrangean dual</td>
</tr>
<tr>
<td>Integer programming</td>
<td>Chvátal-Gomory cuts</td>
<td>Subadditive dual (strong)</td>
</tr>
</tbody>
</table>
Classical Benders

- Solve problem of the form

\[
\begin{align*}
\text{Master problem} & : \\
\min & \quad z \\
\text{s.t.} & \quad z \geq B_{x^i}(x), \quad i \leq k - 1
\end{align*}
\]

Iteration \(k\):

\[
\begin{align*}
\text{Subproblem} & : \\
\min & \quad cx^k + dy \\
\text{s.t.} & \quad By \geq b - A x^k \\
& \quad y \geq 0
\end{align*}
\]
Classical Benders

• Solve problem of the form

\[
\min cx + dy
\]

\[
Ax + By \geq b
\]

\[
x, y \geq 0
\]

Iteration \(k\):

**Master problem**

\[
\min z
\]

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

**Subproblem**

\[
\min cx^k + dy
\]

\[
By \geq b - Ax^k
\]

\[
y \geq 0
\]

Trial value \(x^k\)

Benders cut

\[
z \geq B_{x^k}(x)
\]

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

\[
\min cx + dy \\
Ax + By \geq b \\
x, y \geq 0
\]

Iteration \(k\):

Master problem

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

Subproblem

\[
\min cx^k + dy \\
By \geq b - Ax^k \\
y \geq 0
\]

Trial value \(x^k\)

Benders cut

\[
z \geq B_{x^k}(x)
\]

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

\[
\begin{align*}
\min & \quad cx + dy \\
\text{s.t.} & \quad Ax + By \geq b \\
x, y & \geq 0
\end{align*}
\]

Iteration \(k\):

Master problem

\[
\begin{align*}
\min & \quad z \\
z & \geq B_{x^i} (x), \quad i \leq k - 1
\end{align*}
\]

Subproblem

\[
\begin{align*}
\min & \quad cx^k + dy \\
By & \geq b - Ax^k \\
y & \geq 0
\end{align*}
\]

Trial value \(x^k\)

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

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

\[
\begin{align*}
\min & \quad cx + dy \\
A & \quad Ax + By \geq b \\
x, y & \geq 0
\end{align*}
\]

Iteration \(k\):

**Master problem**

\[
\begin{align*}
\min & \quad z \\
z & \geq B_{x^i}(x), \quad i \leq k - 1
\end{align*}
\]

**Subproblem**

\[
\begin{align*}
\min & \quad cx^k + dy \\
By & \geq b - Ax^k \\
y & \geq 0
\end{align*}
\]

Trial value \(x^k\)

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

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

\[
\begin{align*}
\text{min } & \quad cx + dy \\
\text{Ax} + By & \geq b \\
x, y & \geq 0
\end{align*}
\]

Iteration \( k \):

Master problem

\[
\begin{align*}
\text{min } z \\
z & \geq cx + u^k (b - Ax), \quad i \leq k - 1
\end{align*}
\]

Subproblem

\[
\begin{align*}
\text{min } & \quad cx^k + dy \\
By & \geq b - Ax^k \\
y & \geq 0
\end{align*}
\]

Trial value \( x^k \)

Benders cut

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

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 \((Ax \geq b, \text{ etc.})\)
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 \((Ax \geq b, \text{ etc.})\)

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.
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

\[
\begin{align*}
\text{min} & \quad y_6 \\
\text{s.t.} & \quad y_6 \leftarrow (y_4 \land y_5) \\
& \quad y_5 \leftarrow (\overline{y}_2 \lor y_3) \\
& \quad y_4 \leftarrow (y_1 \lor \overline{y}_2) \\
& \quad y_3 \leftarrow (x_1 \land \overline{x}_2 \land \overline{x}_3) \\
& \quad y_2 \leftarrow (x_2 \lor \overline{x}_3) \\
& \quad y_1 \leftarrow (x_1 \land x_2)
\end{align*}
\]
Master problem

\[
\begin{align*}
\text{min} & \quad z \\
\text{s.t.} & \quad z \geq B_{x^i}(x), \ i \leq k - 1
\end{align*}
\]

Only one feasible solution, trivial to compute by unit propagation

Subproblem

\[
\begin{align*}
\text{min} & \quad y_6 \\
\text{s.t.} & \quad y_6 \leftarrow (y_4 \land y_5) \\
& \quad y_5 \leftarrow (\overline{y}_2 \lor y_3) \\
& \quad y_4 \leftarrow (y_1 \lor \overline{y}_2) \\
& \quad y_3 \leftarrow (x^k_1 \land \overline{x}^k_2 \land \overline{x}^k_3) \\
& \quad y_2 \leftarrow (x^k_2 \lor \overline{x}^k_3) \\
& \quad y_1 \leftarrow (x^k_1 \land x^k_2)
\end{align*}
\]
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 \( y_5 = 1 \).

\[
\begin{align*}
1 & \quad x_1 \\
0 & \quad x_2 \\
1 & \quad x_3
\end{align*}
\]
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 $z \geq \overline{x_2} \land x_3$
Now solve the master problem

$$\min \ z$$

s.t. \quad z \geq x_2 \land x_3$$

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

\[
\begin{align*}
\min & \quad z \\
\text{s.t.} & \quad z \geq \overline{x}_2 \land x_3
\end{align*}
\]

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.

Subproblem

Schedule tasks on each resource, subject to time windows.

Advantage: decouples by resource.

\[
\begin{align*}
\text{Assign tasks to resources to minimize cost.} \\
\text{Solve by mixed integer programming.} \\
\text{Schedule tasks on each resource, subject to time windows.} \\
\text{Advantage: decouples by resource.}
\end{align*}
\]
**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 \([a,b]\),
    \[
    \sum_{j \in J(a,b)} p_{ij} x_{ij} \leq b - a, \quad \text{all } i
    \]
  - \(p_{ij} = \text{processing time of task } j \text{ on resource } i\)
  - \(J(a,b) = \{ \text{tasks with time windows in } [a,b] \}\)
Example: Planning & Scheduling

- Resulting Benders decomposition:

Master problem

\[
\min z \\
z = \sum_{ij} c_{ij} x_{ij} \\
\text{Benders cuts}
\]

Relaxation

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 \( \sum_{j \in J_i} (1 - x_{ij}) \geq 1, \) 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 $\leq 2$ 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

Number of instances solved

Computation time (sec)

0.01 0.1 1 10 100 1000 10000

- Relax + strong cuts
- Relax + weak cuts
- Strong cuts only
- MILP (CPLEX)
Performance profile

120 “c” instances with 3 or 4 resources

- Relax + strong cuts
- Relax + weak cuts
- Strong cuts only
- MIP (CPLEX)
Performance profile

50 “e” instances

- Relax + strong cuts
- Relax + weak cuts
- MIP (CPLEX)
Severe imbalance of master and subproblem time, resulting in poorer performance for LBBD.

<table>
<thead>
<tr>
<th>(m)</th>
<th>(n)</th>
<th>(\text{Strong cuts only} )</th>
<th>(\text{Relax + weak cuts} )</th>
<th>(\text{Relax + strong cuts} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Iter</td>
<td>Master</td>
<td>Subpr</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>18</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>13</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>19</td>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>41</td>
<td>0.5</td>
<td>1.5</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>149</td>
<td>5.7</td>
<td>14</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>107</td>
<td>3.5</td>
<td>117</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>340+</td>
<td>70+</td>
<td>1782+</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>327+</td>
<td>67+</td>
<td>6263+</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>32</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

+ Computation terminated after 7200 sec for instances not solved to optimality.
Subproblem blows up when more than 10 tasks per resource on average

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

+ 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**

<table>
<thead>
<tr>
<th>m</th>
<th>n</th>
<th>Strong cuts only</th>
<th>Relax + weak cuts</th>
<th>Relax + strong cuts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Iters</td>
<td>Master</td>
<td>Subpr</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>13</td>
<td>0.0</td>
<td>0.1</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>23</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>42</td>
<td>0.7</td>
<td>0.5</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>86</td>
<td>4.0</td>
<td>1.5</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>183</td>
<td>19</td>
<td>3.0</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>226</td>
<td>23</td>
<td>6.4</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>340</td>
<td>49</td>
<td>10</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>1222+</td>
<td>1689+</td>
<td>50+</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>1854+</td>
<td>2723+</td>
<td>786+</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>2113+</td>
<td>3283+</td>
<td>3363+</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>32</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

+ Computation terminated after 7200 sec for instances not solved to optimality.
Balance between master and subproblem results in superior performance

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Relax + weak cuts</th>
<th></th>
<th>Relax + strong cuts</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Iters</td>
<td>Master</td>
<td>Subpr</td>
<td>Iters</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>-------</td>
<td>--------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>9.4</td>
<td>0.1</td>
<td>0.0</td>
<td>5.2</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>13</td>
<td>0.3</td>
<td>0.0</td>
<td>4.4</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>14</td>
<td>0.4</td>
<td>0.0</td>
<td>5.6</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>55</td>
<td>14</td>
<td>0.0</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>19</td>
<td>0.4</td>
<td>0.0</td>
<td>8.6</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>26</td>
<td>1.1</td>
<td>0.0</td>
<td>8.8</td>
</tr>
<tr>
<td>7</td>
<td>35</td>
<td>76</td>
<td>34</td>
<td>0.0</td>
<td>19</td>
</tr>
<tr>
<td>8</td>
<td>40</td>
<td>107+</td>
<td>1525+</td>
<td>0.0+</td>
<td>31</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
<td>132</td>
<td>1048</td>
<td>0.0</td>
<td>39</td>
</tr>
<tr>
<td>10</td>
<td>50</td>
<td>39</td>
<td>43</td>
<td>0.0</td>
<td>18</td>
</tr>
</tbody>
</table>

+ Computation terminated after 7200 sec for instances not solved to optimality.
Mild imbalance results in somewhat worse performance

<table>
<thead>
<tr>
<th></th>
<th>Relax + weak cuts</th>
<th></th>
<th>Relax + strong cuts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Iters</td>
<td>Master</td>
<td>Subpr</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>9.4</td>
<td>0.1</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>13</td>
<td>0.3</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>14</td>
<td>0.4</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>55</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>19</td>
<td>0.4</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>26</td>
<td>1.1</td>
</tr>
<tr>
<td>7</td>
<td>35</td>
<td>76</td>
<td>34</td>
</tr>
<tr>
<td>8</td>
<td>40</td>
<td>107+</td>
<td>1525+</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
<td>132</td>
<td>1048</td>
</tr>
<tr>
<td>10</td>
<td>50</td>
<td>39</td>
<td>43</td>
</tr>
</tbody>
</table>

+ 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

\[ p_{ij} = \text{processing time of task } j \text{ on resource } i \]
\[ c_{ij} = \text{resource consumption of task } j \text{ on resource } i \]
\[ C_i = \text{resources available on resource } i \]

Total resource consumption \( \leq C_i \) at all times.
**Min Cost Cumulative Scheduling**

**Master Problem:** Assign tasks to resources
Formulate as MILP problem

\[
\begin{align*}
\text{min} & \quad z \\
\text{subject to} & \quad \sum_{i} x_{ij} = 1, \quad \text{all } j \\
& \quad \sum_{j \in J_{(b,a)}} p_{ij} c_{ij} x_{ij} \leq C_i (b - a), \quad \text{all } i, \text{ various } [a, b]
\end{align*}
\]

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

\[
\begin{align*}
\min & \quad z \\
\text{subject to} & \quad \sum_{i} x_{ij} = 1, \quad \text{all } j \\
& \quad \sum_{j \in J \,(b,a)} p_{ij} c_{ij} x_{ij} \leq C_i (b - a), \quad \text{all } i, \text{ various } [a,b]
\end{align*}
\]

**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

\[
\begin{align*}
\text{min} & \quad \sum_{i} x_{ij} = 1, \quad \text{all } j \\
\text{subject to} & \quad M \geq \frac{1}{C_i} \sum_{j} p_{ij} c_{ij} x_{ij}, \quad \text{all } i \\
\text{Benders cuts} & \\
\end{align*}
\]

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, \( \ldots \), \( s \) from a resource, the minimum makespan on that resource is reduced by at most

\[
\sum_{j=1}^{s} p_{ij} + \max_{j \leq s} \{d_j\} - \min_{j \leq s} \{d_j\}
\]

Assuming all deadlines \( d_j \) are the same, we get the Benders cut

\[
M \geq M^{\ast}_{hi} - \sum_{j \in J_{hi}} (1 - x_{ij}) p_{ij}
\]

Min makespan on resource \( i \) in last iteration
Why does this work? Assume all deadlines are the same. Add tasks $1, \ldots, s$ sequentially at end of optimal schedule for other tasks…

**Case I: resulting schedule meets deadline**

Optimal makespan for tasks $s+1, \ldots, m$

Optimal makespan for all tasks

Feasible makespan for all tasks

Deadline for all tasks

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

$$d$$
Case II: resulting schedule exceeds deadline

Optimal makespan for tasks $s+1, \ldots, m$

Optimal makespan for all tasks

Deadline for all tasks

\[ M^* \leq d \quad \text{and} \quad \hat{M} + \sum_{j=1}^{s} p_{ij} > d \implies \hat{M} \geq M^* - \sum_{j=1}^{s} p_{ij} \]
Min Number of Late Tasks

Master problem: Assign tasks to resources

\[
\begin{align*}
\text{min} & \quad L \\
\text{subject to} & \quad \sum_{i} x_{ij} = 1, \quad \text{all } j \\
\end{align*}
\]

- Benders cuts
- relaxation of subproblem
- \( x_{ij} \in \{0, 1\} \)

= 1 if task \( j \) is assigned to resource \( i \)
Min Number of Late Tasks

Benders cuts

Lower bound on # late tasks on resource $i$

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

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

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

$$\hat{L}_{hi} \geq 0, \text{ all } i$$
Min Number of Late Tasks

Benders cuts

Lower bound on # late tasks on resource $i$

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

\[
L \geq \sum_i \hat{L}_{hi}
\]

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

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

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

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

\[ L \geq \sum_i \hat{L}_{hi} \]

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

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

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

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

To reduce # late tasks, must remove one of the tasks in \( J^0_{hi} \) from resource \( i \).
Min Number of Late Tasks

Benders cuts

\[
L \geq \sum_{i} \hat{L}_{hi}
\]

\[
\hat{L}_{hi} \geq L_{hi}^{*} - L_{hi}^{*} \sum_{j \in J_{hi}^{0}} (1 - x_{ij}), \text{ all } i
\]

\[
\hat{L}_{hi} \geq L_{hi}^{*} - 1 - L_{hi}^{*} \sum_{j \in J_{hi}^{1}} (1 - x_{ij}), \text{ all } i
\]

\[
\hat{L}_{hi} \geq 0, \text{ all } i
\]

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

\[ L \geq \sum \hat{L}_{hi} \]

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

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

\[ \hat{L}_{hi} \geq 0, \text{ all } i \]

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

To reduce # late tasks by more than 1, must remove one of the tasks in \( J^1_{hi} \) from resource \( i \).
Min Number of Late Tasks

Benders cuts

\[
L \geq \sum_{i} \hat{L}_{hi}
\]

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

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

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

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

\[ L \geq \sum_{i} L_{i} \]

\[ \frac{1}{C_{i}} \sum_{k \in J(d_{j})} c_{ik} p_{ik} x_{ik} - d_{j} \]

\[ L_{i} \geq \frac{\max_{k \in J(d_{j})} \{ p_{ik} \}}{\sum_{k \in J(d_{j})}} , \text{ all } j \]
Min Number of Late Tasks

Relaxation of subproblem

Lower bound on # late tasks on resource $i$

$$L \geq \sum_{i} L_{i}$$

$$L_{i} \geq \frac{1}{C_{i}} \sum_{k \in J(d_{j})} c_{ik} P_{ik} x_{ik} - d_{j} \max_{k \in J(d_{j})} \{ p_{ik} \}, \text{ all } j$$

Set of tasks assigned to resource $i$ with deadline at or before $d_{j}$
Min Number of Late Tasks

Relaxation of subproblem

Lower bound on # late tasks on resource $i$

$$L_i \geq \sum_i L_i$$

$$L \geq \frac{1}{C_i} \sum_{k \in J(d_j)} c_{ik} p_{ik} x_{ik} - d_j$$

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

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

Task 1

Task 2

Task 3
Min Number of Late Tasks

Relaxation of subproblem

Lower bound on # late tasks on resource \(i\):

\[
L \geq \sum_{i} L_i
\]

\[
L_i \geq \frac{1}{C_i} \sum_{k \in J(d_j)} c_{ik} p_{ik} x_{ik} - d_j
\]

Area of tasks assigned to resource \(i\) with deadline at or before \(d_j\):

Energy = \(p_{i1} c_{i1}\)
Min Number of Late Tasks

Relaxation of subproblem

\[
L \geq \sum_i L_i \\
L_i \geq \frac{1}{C_i} \sum_{k \in J(d_j)} c_{ik} p_{ik} x_{ik} - d_j \\
\text{max}_{k \in J(d_j)} \{ p_{ik} \}, \text{ all } j
\]

Lower bound on (makespan – latest deadline)
Min Number of Late Tasks

Relaxation of subproblem

\[ L \geq \sum_{i} L_i \]
\[ L_i \geq \frac{1}{C_i} \sum_{k \in J(d_j)} c_{ik} p_{ik} x_{ik} - d_j \]

\[ \max_{k \in J(d_j)} \{ p_{ik} \} \]

Max processing time

Lower bound on (makespan – latest deadline)

Max processing time

task 1

task 2

task 3

d_j
Min Number of Late Tasks

Relaxation of subproblem

\[ L \geq \sum_i L_i \]

\[ L_i \geq \frac{1}{C_i} \sum_{k \in J(d_j)} c_{ik} p_{ik} x_{ik} - d_j \]

\[ \max_{k \in J(d_j)} \{ p_{ik} \} \]

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.

\[ L \geq \sum_{i} L_i \]

\[ L_i \geq \frac{1}{C_i} \sum_{k \in J(d_j)} c_{ik} p_{ik} x_{ik} - d_j \]

\[ \max_{k \in J(d_j)} \{ p_{ik} \} \]

Min # of late jobs on resource \( i \)
Min Tardiness Cumulative Scheduling

**Master problem:** assign tasks to resources

\[
\begin{align*}
\text{min} & \quad \sum_i x_{ij} = 1, \quad \text{all } j \\
\text{subject to} \quad & \quad \sum_i x_{ij} = 1, \quad \text{all } j \\
& \quad \text{Benders cuts} \\
& \quad \text{relaxation I of subproblem} \\
& \quad \text{relaxation II of subproblem} \\
& \quad x_{ij} \in \{0,1\}
\end{align*}
\]
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)

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

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

$$\hat{T}_{hi} \geq 0, \text{ all } i$$
Min Tardiness Cumulative Scheduling

Benders cuts

Lower bound on tardiness for resource $i$

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

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

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

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

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

$$\hat{T}_{hi} \geq 0, \text{ all } i$$
Min Tardiness Cumulative Scheduling

**Benders cuts**

\[
T \geq \sum_i \hat{T}_{hi} \\
\hat{T}_{hi} \geq T_{hi}^* - T_{hi}^* \sum_{j \in J_{hi}} (1 - x_{ij}), \text{ all } i \\
\hat{T}_{hi} \geq T_{hi}^0 - T_{hi}^0 \sum_{j \in J_{hi}} (1 - x_{ij}), \text{ all } i \\
\hat{T}_{hi} \geq 0, \text{ all } i
\]

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

\[ T \geq \sum_{i} \hat{T}_{hi} \]

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

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

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

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

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

To reduce tardiness below \( T_{hi}^{0} \) 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 $h$

\[
T \geq \sum_{i} \hat{T}_{hi}
\]

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

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

\[
\hat{T}_{hi} \geq 0, \quad \text{all } i
\]
Subproblem relaxation I

Lower bound on total tardiness for resource $i$

$$T \geq \sum_{i} T_i$$

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

Lower bound on total tardiness for resource $i$

$$T \geq \sum_i T_i$$

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

Set of tasks assigned to resource $i$ with deadline at or before $d_k$
Subproblem relaxation I

Lower bound on total tardiness for resource $i$

\[ T \geq \sum_i T_i \]

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

Area of tasks assigned to resource $i$ with deadline at or before $d_k$
Subproblem relaxation I

Lower bound on total tardiness for resource $i$

\[ T \geq \sum_{i} T_i \]

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

**Lemma.** Consider a min tardiness problem that schedules tasks 1, ..., $n$ on resource $i$, where $d_1 \leq \cdots \leq d_n$. 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 $\pi$ is a permutation of 1, ..., $n$ such that

$$
P_{i\pi_i(1)} c_{i\pi_i(1)} \leq \cdots \leq P_{i\pi_i(n)} c_{i\pi_i(n)}$$
Example of Lemma

<table>
<thead>
<tr>
<th></th>
<th>$j$</th>
<th>$d_j$</th>
<th>$p_{ij}$</th>
<th>$c_{ij}$</th>
<th>$p_{ij}c_{ij}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

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

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

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

Lower bound on tardiness = $\left[ \overline{T}_1 + \overline{T}_2 + \overline{T}_3 \right] = \left[ \frac{4}{3} \right] = 2$

Min tardiness = 4
Idea of proof

For a permutation $\sigma$ of $1, \ldots, n$ 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 $\sigma_0(1), \ldots, \sigma_0(n)$ be order of jobs in any optimal solution, so that $t_{\sigma_0(1)} \leq \cdots \leq t_{\sigma_0(n)}$ and min tardiness is $T^*$.

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

106
Now we have

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

since

\[ T^* = \sum_{j=1}^{n} (t_{\sigma(j)} + p_{i\sigma(j)} - d_{\sigma(j)})^+ \geq \sum_{j=1}^{n} \left( \frac{1}{C_i} \sum_{j=1}^{k} p_{i\sigma(j)} c_i \sigma(j) - d_{\sigma(j)} \right)^+ \]

\[ \geq \sum_{j=1}^{n} \left( \frac{1}{C_i} \sum_{j=1}^{k} p_{i\sigma(j)} c_i \pi(j) - d_{\sigma(j)} \right)^+ = T(\sigma_0) \]

areas

def. of \( \pi \)

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

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

So

\[ T(\sigma_s) - T(\sigma_{s+1}) = T_k(\sigma_s) + T_{k+1}(\sigma_s) - T_k(\sigma_{s+1}) - T_{k+1}(\sigma_{s+1}) \]

\[ = (a - A)^+ + (A - b)^+ - (a - b)^+ - (A - B)^+ \geq 0 \]

since \( A \geq a, \ B \geq b \)
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

<table>
<thead>
<tr>
<th>Jobs</th>
<th>MILP</th>
<th>CP</th>
<th>Benders</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>3.4</td>
<td>0.8</td>
<td>0.24</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>4.0</td>
<td>0.31</td>
</tr>
<tr>
<td>14</td>
<td>2572+</td>
<td>299</td>
<td>5.0</td>
</tr>
<tr>
<td>16</td>
<td>5974+</td>
<td>3737</td>
<td>36</td>
</tr>
<tr>
<td>18</td>
<td>7200+</td>
<td>233</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>1268</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

+ At least one problem in the 5 exceeded 7200 sec (2 hours)
## Min makespan, 3 resources

Average of 5 instances shown

<table>
<thead>
<tr>
<th>Jobs</th>
<th>MILP</th>
<th>CP</th>
<th>Benders</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>3.9</td>
<td>0.9</td>
<td>0.23</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>7.5</td>
<td>0.38</td>
</tr>
<tr>
<td>14</td>
<td>524</td>
<td>981</td>
<td>1.4</td>
</tr>
<tr>
<td>16</td>
<td>1716+</td>
<td>4414</td>
<td>7.6</td>
</tr>
<tr>
<td>18</td>
<td>4619+</td>
<td>7200+</td>
<td>30</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td>8.7</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td>2012+</td>
</tr>
</tbody>
</table>

+ At least one problem in the 5 exceeded 7200 sec (2 hours)
Min makespan, 4 resources

Average of 5 instances shown

<table>
<thead>
<tr>
<th>Jobs</th>
<th>MILP</th>
<th>CP</th>
<th>Benders</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1.0</td>
<td>0.07</td>
<td>0.19</td>
</tr>
<tr>
<td>12</td>
<td>5.0</td>
<td>1.9</td>
<td>0.43</td>
</tr>
<tr>
<td>14</td>
<td>24</td>
<td>524</td>
<td>0.82</td>
</tr>
<tr>
<td>16</td>
<td>35</td>
<td>3898</td>
<td>1.0</td>
</tr>
<tr>
<td>18</td>
<td>3931+</td>
<td>7200+</td>
<td>6.4</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td>4.4</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>945</td>
<td></td>
</tr>
</tbody>
</table>

+ At least one problem in the 5 exceeded 7200 sec (2 hours)
Min makespan, 3 resources
Different deadlines

Average of 5 instances shown

<table>
<thead>
<tr>
<th>Jobs</th>
<th>MILP</th>
<th>CP</th>
<th>Benders</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>223</td>
<td>7.1</td>
<td>4.4</td>
</tr>
<tr>
<td>16</td>
<td>853</td>
<td>1620+</td>
<td>5.1</td>
</tr>
<tr>
<td>18</td>
<td>350</td>
<td>1928+</td>
<td>2.9</td>
</tr>
<tr>
<td>20</td>
<td>7200+</td>
<td>7200+</td>
<td>1449+</td>
</tr>
<tr>
<td>22</td>
<td>7200+</td>
<td></td>
<td>388</td>
</tr>
<tr>
<td>24</td>
<td>7200+</td>
<td></td>
<td>132</td>
</tr>
</tbody>
</table>

+ At least one problem in the 5 exceeded 7200 sec (2 hours)
<table>
<thead>
<tr>
<th>Tasks</th>
<th>Time (sec)</th>
<th>Min # late tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CP</td>
<td>MILP</td>
</tr>
<tr>
<td>14</td>
<td>1092</td>
<td>5.8</td>
</tr>
<tr>
<td></td>
<td>382</td>
<td>8.0</td>
</tr>
<tr>
<td></td>
<td>265</td>
<td>3.2</td>
</tr>
<tr>
<td></td>
<td>85</td>
<td>2.6</td>
</tr>
<tr>
<td></td>
<td>5228</td>
<td>1315</td>
</tr>
<tr>
<td>16</td>
<td>304</td>
<td>2.7</td>
</tr>
<tr>
<td></td>
<td>?</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>310</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>4925</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>5.7</td>
</tr>
<tr>
<td>18</td>
<td>&gt;7200</td>
<td>2.0</td>
</tr>
<tr>
<td></td>
<td>?</td>
<td>8.0</td>
</tr>
<tr>
<td></td>
<td>&gt;7200</td>
<td>867</td>
</tr>
<tr>
<td></td>
<td>&gt;7200</td>
<td>6.3</td>
</tr>
<tr>
<td></td>
<td>&gt;7200</td>
<td>577</td>
</tr>
</tbody>
</table>
### Min # late tasks

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Time (sec)</th>
<th>Best solution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MILP</td>
<td>Benders</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>97</td>
<td>0.4</td>
</tr>
<tr>
<td>&gt;7200</td>
<td></td>
<td>2.3</td>
</tr>
<tr>
<td>219</td>
<td></td>
<td>5.0</td>
</tr>
<tr>
<td>&gt;7200</td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>843</td>
<td></td>
<td>166</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>1.3</td>
</tr>
<tr>
<td>&gt;7200</td>
<td></td>
<td>3.7</td>
</tr>
<tr>
<td>&gt;7200</td>
<td></td>
<td>49</td>
</tr>
<tr>
<td>&gt;7200</td>
<td></td>
<td>3453</td>
</tr>
<tr>
<td>&gt;7200</td>
<td></td>
<td>&gt;7200</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>0.8</td>
</tr>
<tr>
<td>&gt;7200</td>
<td></td>
<td>18</td>
</tr>
<tr>
<td>&gt;7200</td>
<td></td>
<td>62</td>
</tr>
<tr>
<td>&gt;7200</td>
<td></td>
<td>124</td>
</tr>
<tr>
<td>&gt;7200</td>
<td></td>
<td>234</td>
</tr>
</tbody>
</table>

3 resources

Larger problems

( ) = optimality not proved
## Effect of subproblem relaxation

3 resources

### Min # late tasks

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Time (sec)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>with relax</td>
<td>without relax</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0.5</td>
<td>2.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>1.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.7</td>
<td>4.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>0.1</td>
<td>1.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3.4</td>
<td>3.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.4</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8.5</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0.4</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.3</td>
<td>9.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.0</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td></td>
<td>166</td>
<td>226</td>
<td></td>
</tr>
</tbody>
</table>
### Min total tardiness

#### 3 resources

#### Smaller problems

<table>
<thead>
<tr>
<th>Tasks</th>
<th>CP</th>
<th>Time (sec)</th>
<th>Min tardiness</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CP</td>
<td>MILP</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>838</td>
<td>7.0</td>
</tr>
<tr>
<td></td>
<td>7159</td>
<td>34</td>
<td>3.7</td>
</tr>
<tr>
<td></td>
<td>1783</td>
<td>45</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>&gt;7200</td>
<td>73</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>&gt;7200</td>
<td>&gt;7200</td>
<td>3269</td>
</tr>
<tr>
<td>16</td>
<td>&gt;7200</td>
<td>2</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>&gt;7200</td>
<td>8</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>&gt;7200</td>
<td>867</td>
<td>8.5</td>
</tr>
<tr>
<td></td>
<td>&gt;7200</td>
<td>6</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td>&gt;7200</td>
<td>577</td>
<td>3.4</td>
</tr>
<tr>
<td>18</td>
<td>187</td>
<td>2.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>5.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>46</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&gt;7200</td>
<td>1203</td>
<td></td>
</tr>
</tbody>
</table>
Min total tardiness
3 resources
Larger problems

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Time (sec)</th>
<th>Best solution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MILP</td>
<td>Benders</td>
</tr>
<tr>
<td>20</td>
<td>105</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>4141</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>39</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>1442</td>
<td>332</td>
</tr>
<tr>
<td></td>
<td>&gt;7200</td>
<td>&gt;7200</td>
</tr>
<tr>
<td>22</td>
<td>6</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>584</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>&gt;7200</td>
<td>&gt;7200</td>
</tr>
<tr>
<td></td>
<td>&gt;7200</td>
<td>&gt;7200</td>
</tr>
<tr>
<td></td>
<td>&gt;7200</td>
<td>&gt;7200</td>
</tr>
<tr>
<td>24</td>
<td>10</td>
<td>324</td>
</tr>
<tr>
<td></td>
<td>&gt;7200</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>&gt;7200</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td>&gt;7200</td>
<td>&gt;7200</td>
</tr>
<tr>
<td></td>
<td>&gt;7200</td>
<td>&gt;7200</td>
</tr>
</tbody>
</table>

( ) = optimality not proved
## Effect of subproblem relaxation

### 3 resources

**Min total tardiness**

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Time (sec) with relax</th>
<th>Time (sec) without relax</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>1.4</td>
<td>4.4</td>
</tr>
<tr>
<td></td>
<td>2.1</td>
<td>6.5</td>
</tr>
<tr>
<td></td>
<td>4.2</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>156</td>
<td>199</td>
</tr>
<tr>
<td></td>
<td>765</td>
<td>763</td>
</tr>
<tr>
<td>18</td>
<td>2.8</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>5.3</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>47</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td>49</td>
<td>354</td>
</tr>
<tr>
<td></td>
<td>1203</td>
<td>5102</td>
</tr>
<tr>
<td>20</td>
<td>18</td>
<td>151</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>1898</td>
</tr>
<tr>
<td></td>
<td>29</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>332</td>
<td>764</td>
</tr>
<tr>
<td></td>
<td>&gt;7200</td>
<td>&gt;7200</td>
</tr>
</tbody>
</table>
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.

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.
Unsegmented problem

• Master problem:

\[ \sum_{i \in I} y_{ij} \geq 1, \ j \in J \]

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

\[ \sum_{j \in J} y_{ij1} \leq 1, \sum_{j \in J} y_{ij2} \leq 1, \sum_{j \in J} y_{ij3} \leq 1, \ i \in I \]

\[ y_{ij1} \leq y_{i-1,j,2} + y_{i-1,j,3}, \ i \in I, i > 1, j \in J \]

\[ y_{ij2} \leq y_{i+1,j,1} + y_{i+1,j,3}, \ i \in I, i < n, j \in J \]

\[ y_{ij3} \leq y_{i-1,j,3} + y_{i-1,j,2}, \ i \in I, i > 1, j \in J \]

\[ y_{ij3} \leq y_{i+1,j,3} + y_{i+1,j,1}, \ i \in I, i < n, j \in J \]

\[ \sum_{i \in I} y_{ij0} \leq 1, \sum_{i \in I} y_{ij1} \leq 1, \sum_{i \in I} y_{ij2} \leq 1, \ j \in J \]

\[ y_{1j1} = y_{1j3} = y_{nj2} = y_{nj3} = 0, \ j \in J \]

\[ \sum_{i \in I} y_{ij3} \leq \left\lfloor \frac{p_j}{a_{i+1} - a_i} \right\rfloor, \ j \in J \]

\[ y_{ij0}, y_{ij1}, y_{ij2}, y_{ij3} \in \{0, 1\}, \ i \in I, j \in J \]

\[ x_{ij1} \leq p_j y_{ij1}, \ x_{ij2} \leq p_j y_{ij2} \]

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

\[ x_{ij1}, x_{ij2} \geq 0 \]

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

\[ x_{ijk} \text{ variables keep track of how long a partial job } j \text{ runs in segment } i. \]
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.
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

Subproblem
Solve with CP

Benders cut
Patient, day assignments
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.
Home Hospice Care

Master problem

\[
\begin{align*}
\max \quad & \sum_j \delta_j \\
\sum_i x_{ij} &= \delta_j, \quad \text{all } j \\
\sum_{i,k} y_{ijk} &= v_j \delta_j, \quad \text{all } j \\
y_{ijk} &\leq x_{ij}, \quad \text{all } i, j, k
\end{align*}
\]

Spacing constraints on visit days

Benders cuts

Relaxation of subproblem

\( \delta_j, x_{ij}, y_{ijk} \in \{0, 1\} \)
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.

\[
\sum_{j \in \overline{P}_{ik}} (1 - y_{ijk}) \geq 1
\]

Reduced set of patients whose assignment to aide \(i\) on day \(k\) creates infeasibility
Home Hospice Care

- 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 without sacrificing optimality.
    - Original cut ensures convergence to optimum.
    - This yields 2 cuts per aide:

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

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

\[
\sum_{j \in J(a,b)} p_j y_{ijk} \leq b - a
\]

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

Can use several intervals.
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 $[a,b]$, $[b,c]$

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

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

where

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

$p'_i = p_j + \min \{ t_{ij}, \min_{j' \in Q_{ik}} \{ t_{ij'} \} \}$

$p'' = p_j + \min \{ \min_{j' \in Q_{ik}} \{ t_{ij'} \}, c \}$

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

Computation Time, MIP vs LBBD

Minutes

New Patients

MIP
LBBD
Home Hospice Care

Effect of Subproblem Relaxation

- No relax
- Relax

Minutes vs. New patients graph.
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 $x_1$, $x_2$, $x_3$
Inference as Projection

- Project onto propositional variables of interest

  Suppose we wish to infer from these clauses everything we can about propositions $x_1, x_2, x_3$

We can deduce

\[ x_1 \lor x_2 \]
\[ x_1 \lor x_3 \]

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.
Inference as Projection

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

Current Master problem

\[ x_1 \lor x_2 \]

solution of master

\((x_1, x_2, x_3) = (0, 1, 0)\)

Resulting subproblem

\[
\begin{align*}
x_4 & \lor x_5 \\
x_4 & \lor \bar{x}_5 \\
x_5 & \lor x_6 \\
x_5 & \lor \bar{x}_6 \\
\bar{x}_4 & \lor x_5 \\
\bar{x}_4 & \lor \bar{x}_5
\end{align*}
\]
Inference as Projection

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

Current Master problem

\[ x_1 \lor x_2 \]

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

Resulting subproblem

\[
\begin{align*}
  x_4 & \lor x_5 \\
  x_4 & \lor \bar{x}_5 \\
  x_5 & \lor x_6 \\
  x_5 & \lor \bar{x}_6 \\
  \bar{x}_4 & \lor x_5 \\
  \bar{x}_4 & \lor \bar{x}_5
\end{align*}
\]

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

\[ x_1 \lor x_2 \]
\[ x_1 \lor x_3 \]

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

Benders cut (nogood)

Resulting subproblem

\[ x_4 \lor x_5 \]
\[ x_4 \lor \overline{x}_5 \]
\[ x_5 \lor x_6 \]
\[ x_5 \lor \overline{x}_6 \]
\[ \overline{x}_4 \lor x_5 \]
\[ \overline{x}_4 \lor \overline{x}_5 \]

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

\[ x_1 \lor x_2 \]
\[ x_1 \lor x_3 \]

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

Resulting subproblem

\[ x_4 \lor x_5 \]
\[ x_4 \lor \overline{x_5} \]
\[ x_5 \lor x_6 \]
\[ x_5 \lor \overline{x_6} \]
Inference as Projection

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

Current Master problem
\[ x_1 \lor x_2 \]
\[ x_1 \lor x_3 \]

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

Resulting subproblem
\[ x_4 \lor x_5 \]
\[ x_4 \lor \bar{x}_5 \]
\[ x_5 \lor x_6 \]
\[ x_5 \lor \bar{x}_6 \]

Subproblem is feasible
Inference as Projection

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

Current Master problem

\[
\begin{align*}
X_1 \lor X_2 \\
X_1 \lor X_3 \\
X_1 \lor \bar{X}_2 \lor \bar{X}_3
\end{align*}
\]

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

Enumerative Benders cut

Resulting subproblem

\[
\begin{align*}
x_4 \lor x_5 \\
x_4 \lor \bar{x}_5 \\
x_5 \lor x_6 \\
x_5 \lor \bar{x}_6
\end{align*}
\]

Subproblem is feasible
Inference as Projection

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

Current Master problem

\[ x_1 \lor x_2 \\
 x_1 \lor x_3 \\
 x_1 \lor \bar{x}_2 \lor \bar{x}_3 \]

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

Enumerative Benders cut

Continue until master is infeasible.

Resulting subproblem

\[ x_4 \lor x_5 \\
 x_4 \lor \bar{x}_5 \\
 x_5 \lor x_6 \\
 x_5 \lor \bar{x}_6 \]

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.
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.
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], 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


