Tutorial: Operations Research in Constraint Programming

John Hooker
Carnegie Mellon University

May 2009
Revised June 2009

Motivation

- **Benders decomposition** allows us to apply CP and OR to different parts of the problem.
- It searches over values of certain variables that, when fixed, result in a much simpler **subproblem**.
- The search learns from past experience by accumulating **Benders cuts** (a form of nogood).
- The technique can be **generalized** far beyond the original OR conception.
- Generalized Benders methods have resulted in the **greatest speedups** achieved by combining CP and OR.

Benders Decomposition in the Abstract

Benders decomposition can be applied to problems of the form

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

When x is fixed to some value, the resulting **subproblem** is much easier:

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

…perhaps because it decouples into smaller problems.

For example, suppose *x* assigns jobs to machines, and *y* schedules the jobs on the machines.

When *x* is fixed, the problem decouples into a separate scheduling subproblem for each machine.

Benders Decomposition

We will search over assignments to *x*. This is the **master problem**.

In iteration *k* we assume $x = x^k$ and solve the subproblem

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

and get optimal value v_k

We generate a **Benders cut** (a type of nogood) $\boxed{v} \geq B_{k+1}(x)$

that satisfies $B_{k+1}(x^k) = v_k$.

Cost in the original problem

The Benders cut says that if we set $x = x^k$ again, the resulting cost v will be at least v_k. To do better than v_k, we must try something else.

It also says that any other x will result in a cost of at least $B_{k+1}(x)$, perhaps due to some similarity between x and x^k.

Benders Decomposition

We will search over assignments to x. This is the **master problem**.

In iteration k we assume $x = x^k$ and solve the subproblem

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

and get optimal value v_k

We generate a **Benders cut** (a type of nogood) $\boxed{v} \geq B_{k+1}(x)$

that satisfies $B_{k+1}(x) = v_k$.

Cost in the original problem

We add the Benders cut to the master problem, which becomes

$$\begin{aligned} &\min\ v \\ &v \geq B_i(x),\ i = 1, \ldots, k+1 \longleftarrow \\ &x \in D_x \end{aligned}$$

Benders cuts generated so far

Benders Decomposition

We now solve the master problem

$$\begin{aligned} &\min \ v \\ &v \geq B_i(x), \ i = 1,\ldots,k+1 \\ &x \in D_x \end{aligned}$$

to get the next trial value x^{k+1}.

The master problem is a relaxation of the original problem, and its optimal value is a **lower bound** on the optimal value of the original problem.

The subproblem is a restriction, and its optimal value is an **upper bound**.

The process continues until the bounds meet.

The Benders cuts partially define the **projection** of the feasible set onto *x*. We hope not too many cuts are needed to find the optimum.

Classical Benders Decomposition

The classical method applies to problems of the form

$$\begin{aligned} &\min\ f(x) + cy \\ &g(x) + Ay \ge b \\ &x \in D_x,\ \ y \ge 0 \end{aligned}$$

and the subproblem is an LP

$$\begin{aligned} &\min\ f(x^k) + cy \\ &Ay \ge b - g(x^k) \quad (\lambda) \\ &y \ge 0 \end{aligned}$$

whose dual is

$$\begin{aligned} &\max\ f(x^k) + \lambda\left(b - g(x^k)\right) \\ &\lambda A \le c \\ &\lambda \ge 0 \end{aligned}$$

Let λ^k solve the dual.

By strong duality, $B_{k+1}(x) = f(x) + \lambda^k(b - g(x))$ is the tightest lower bound on the optimal value v of the original problem when $x = x^k$.

Even for other values of x, λ^k **remains feasible in the dual**. So by weak duality, $B_{k+1}(x)$ remains a lower bound on v.

Classical Benders

So the master problem

$$\begin{aligned} &\min\ v \\ &v \geq B_i(x),\ i = 1,\ldots,k+1 \\ &x \in D_x \end{aligned}$$

becomes

$$\begin{aligned} &\min\ v \\ &v \geq f(x) + \lambda^i(b - g(x)),\ i = 1,\ldots,k+1 \\ &x \in D_x \end{aligned}$$

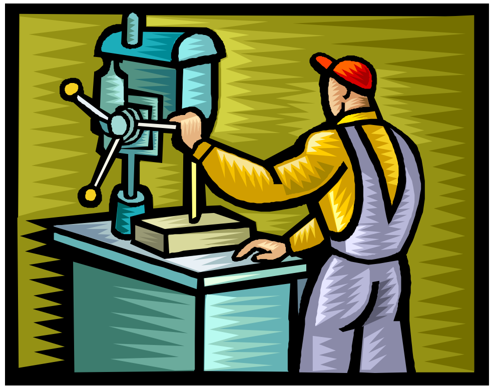
In most applications the master problem is

- an MILP
- a nonlinear programming problem (NLP), or
- a mixed integer/nonlinear programming problem (MINLP).

Example: Machine Scheduling

- Assign 5 jobs to 2 machines (A and B), and schedule the machines assigned to each machine within time windows.
- The objective is to minimize **makespan**.

Time lapse between start of first job and end of last job.



- Assign the jobs in the **master problem**, to be solved by **MILP**.
- Schedule the jobs in the **subproblem**, to be solved by **CP**.

Machine Scheduling

Job Data

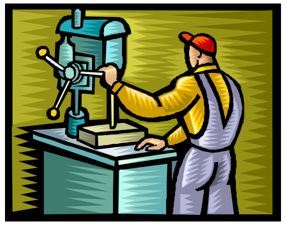
Job j	Release time r_j	Dead-line d_j	Processing time p_{Aj}	p_{Bj}
1	0	10	1	5
2	0	10	3	6
3	2	7	3	7
4	2	10	4	6
5	4	7	2	5

Machine A

Machine B

Once jobs are assigned, we can minimize overall makespan by minimizing makespan on each machine individually.

So the subproblem decouples.



Machine Scheduling

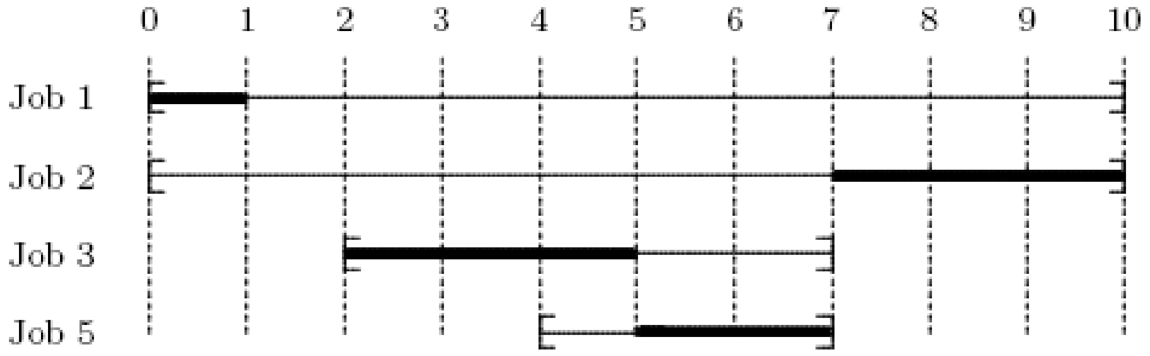
Job Data

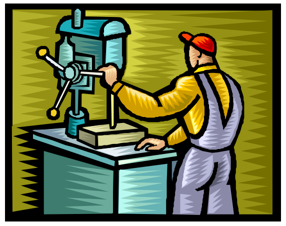
Job j	Release time r_j	Dead-line d_j	Processing time p_{Aj}	p_{Bj}
1	0	10	1	5
2	0	10	3	6
3	2	7	3	7
4	2	10	4	6
5	4	7	2	5

Once jobs are assigned, we can minimize overall makespan by minimizing makespan on each machine individually.

So the subproblem decouples.

Minimum makespan schedule for jobs 1, 2, 3, 5 on machine A





Machine Scheduling

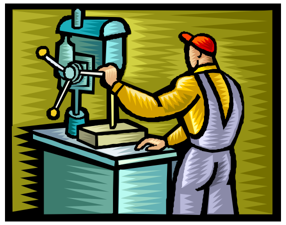
The problem is

Start time of job *j*

Time windows

Jobs cannot overlap

$$\begin{aligned}&\min\ M\\ &M \ge \boxed{s_j} + p_{x_j j},\ \text{all } j\\ &r_j \le s_j \le d_j - p_{x_j j},\ \text{all } j\\ &\text{disjunctive}\big((s_j \mid x_j = i),(p_{ij} \mid x_j = i)\big),\ \text{all } i\end{aligned}$$



Machine Scheduling

The problem is

Start time of job *j*

$$\min \; M$$

$$M \geq \boxed{s_j} + p_{x_j j}, \text{ all } j$$

Time windows

$$r_j \leq s_j \leq d_j - p_{x_j j}, \text{ all } j$$

Jobs cannot overlap

$$\text{disjunctive}\big((s_j \,|\, x_j = i), (p_{ij} \,|\, x_j = i)\big), \text{ all } i$$

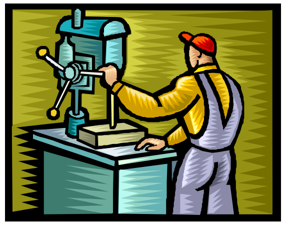
For a fixed assignment $\bar{x}$ the subproblem on each machine *i* is

$$\min \; M$$

$$M \geq s_j + p_{\bar{x}_j j}, \text{ all } j \text{ with } \bar{x}_j = i$$

$$r_j \leq s_j \leq d_j - p_{\bar{x}_j j}, \text{ all } j \text{ with } \bar{x}_j = i$$

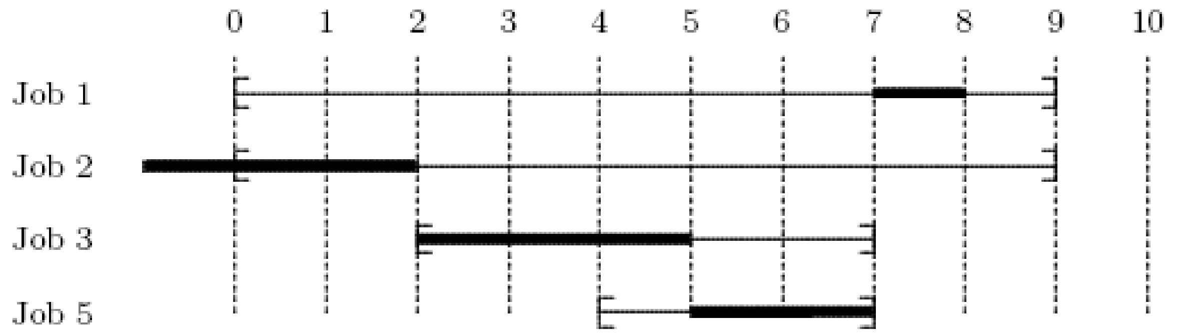
$$\text{disjunctive}\big((s_j \,|\, \bar{x}_j = i), (p_{ij} \,|\, \bar{x}_j = i)\big)$$



Benders cuts

Suppose we assign jobs 1,2,3,5 to machine A in iteration *k*.

We can prove that 10 is the optimal makespan by proving that the schedule is infeasible with makespan 9.



Edge finding derives infeasibility by reasoning only with jobs 2,3,5. So these jobs alone create a minimum makespan of 10.

So we have a Benders cut

$$v \geq B_{k+1}(x) = \begin{cases} 10 & \text{if } x_2 = x_3 = x_4 = A \\ 0 & \text{otherwise} \end{cases}$$

Benders cuts

We want the master problem to be an MILP, which is good for assignment problems.

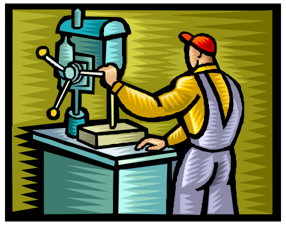
So we write the Benders cut

$$v \geq B_{k+1}(x) = \begin{cases} 10 & \text{if } x_2 = x_3 = x_4 = A \\ 0 & \text{otherwise} \end{cases}$$

Using 0-1 variables:

$$v \geq 10\left(x_{A2} + x_{A3} + \boxed{x_{A5}} - 2\right)$$
$$v \geq 0$$

= 1 if job 5 is assigned to machine A



Master problem

The master problem is an MILP:

$$\min \; v$$

$$\sum_{j=1}^{5} p_{Aj} x_{Aj} \le 10, \text{ etc.}$$

$$\sum_{j=1}^{5} p_{Bj} x_{Bj} \le 10, \text{ etc.}$$

Constraints derived from time windows

$$v \ge \sum_{j=1}^{5} p_{ij} x_{ij}, \quad v \ge 2 + \sum_{j=3}^{5} p_{ij} x_{ij}, \text{ etc.}, \quad i = A, B$$

Constraints derived from release times

$$v \ge 10(x_{A2} + x_{A3} + x_{A5} - 2)$$

Benders cut from machine A

$$v \ge 8 x_{B4}$$

Benders cut from machine B

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

Stronger Benders cuts

If all release times are the same, we can strengthen the Benders cuts.

We are now using the cut

$$v \geq M_{ik}\left(\sum_{j \in J_{ik}} x_{ij} - |J_{ik}| + 1\right)$$

Min makespan on machine *i* in iteration *k*

Set of jobs assigned to machine *i* in iteration *k*

A stronger cut provides a useful bound even if only some of the jobs in J_{ik} are assigned to machine *i*:

$$v \geq M_{ik} - \sum_{j \in J_{ik}} (1 - x_{ij}) p_{ij}$$

These results can be generalized to cumulative scheduling.

THE
END