# Logic-Based Benders Methods for Planning and Scheduling

John Hooker
Carnegie Mellon University

August 2003

# The Problem:
## Multiple-machine resource-constrained scheduling

- Schedule jobs on several machines.
  - Each job consumes some resource at a given rate.
  - Resource consumption rate on each machine must not exceed capacity.
  - Special case: jobs do not overlap.

- Apply logic-based Benders decomposition
  - Generalizes Benders to any subproblem (not just LP, NLP).
  - Solve with MILP and CP (constraint programming).

# Basic Idea

- **Decompose** problem into

| | | |
|---|---|---|
| assignment | + | resource-constrained scheduling |
| *assign jobs to machines* | | *schedule jobs on each machine* |

- Use **logic-based Benders** scheme to link these.
- Solve: master problem with **MILP**
  -- good at resource allocation
  subproblem with **CP**
  -- good at scheduling

- Applications in manufacturing/supply chain planning & scheduling.

# Previous Work

**1989** (Jeroslow & Wang) – View Horn SAT dual as inference problem.

**1995** (JH & Yan) – Apply logic-based Benders to circuit verification.

- Better than BDDs when circuit contains error.

**1995, 2000** (JH) – Formulate general logic-based Benders.

- Specialized Benders cuts must be designed for each problem class.
- Branch-and-check proposed.

**2001** (Jain & Grossmann) – Apply logic-based Benders to multiple-machine scheduling using CP/MILP.

- Several orders of magnitude speedup wrt CPLEX, ILOG Scheduler.
- But… easy problem for Benders approach (min cost).

**2001** (Thorsteinsson) – Apply branch-and-check to CP/MILP.

- 1-2 orders of magnitude speedup on multiple machine scheduling.

**2002** (JH, Ottosson) – Apply logic-based Benders to SAT, IP.

**2003** (JH, Yan) – Relaxation for resource-constrained scheduling (*cumulative* constraint).

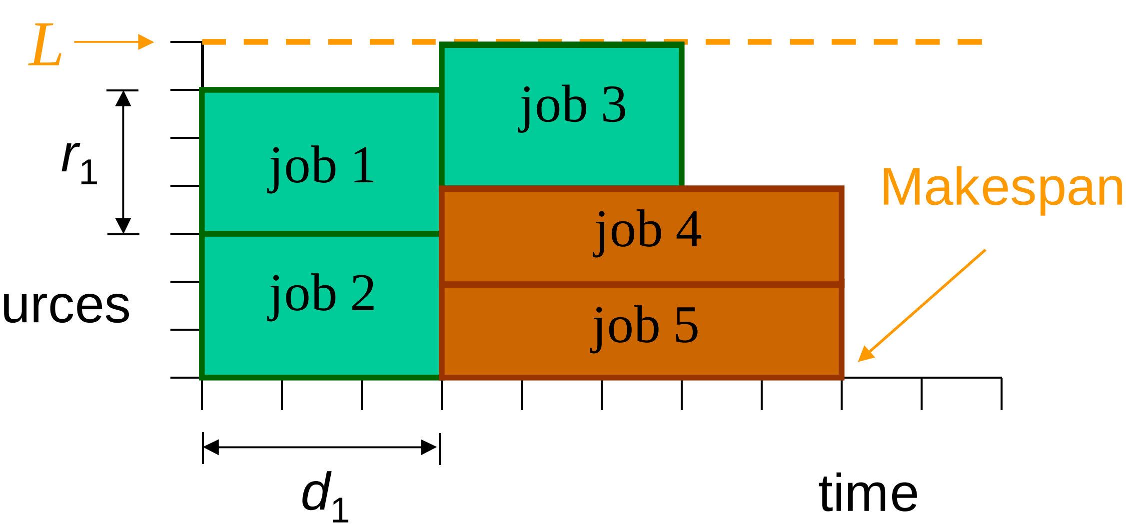
- Not useful computationally for planning and scheduling.
- A simpler relaxation works better.

**Today** – Apply logic-based Benders to resource-constrained planning/scheduling problems.

- Multiple machines, parallel processing with resource constraint on each machine
- Min cost and min makespan.

# Single-machine resource-constrained scheduling

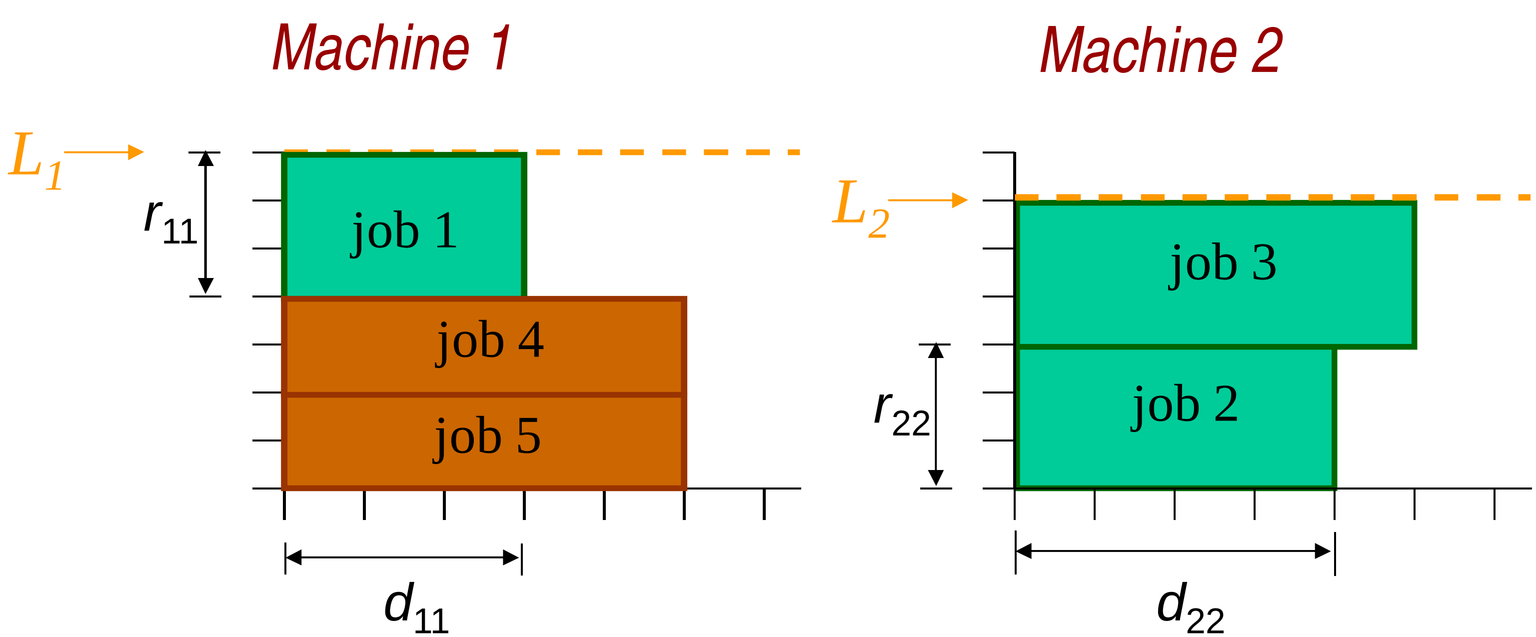
$d_j$ = duration of job $j$
$r_j$ = rate of resource consumption of job $j$
$L$ = resources available
$[a_j,b_j]$ = time window for job $j$




Total resource consumption $\leq L$ at all times.

# Multiple-machine resource-constrained scheduling

$d_{ij}$ = duration of job *j* on machine *i*
$r_{ij}$ = resource consumption of job *j* on machine *i*
$L_i$ = resources available on machine *i*




Total resource consumption ≤ $L_i$ at all times.

# Some Possible Objectives

Minimize cost = $\sum_{ij} c_{ij} x_{ij}$

- $x_{ij}$ = 1 if job *j* assigned to machine *i*
- $c_{ij}$: Cost of assigning job *j* to machine *i*

Minimize makespan = $\max_{ij} \{ t_j + d_{ij} \}$

- $t_j$: Start time of job *j*

Minimize tardiness = $\sum_{ij} \max \{ 0,\ t_j + d_{ij} - b_j \}$

- $b_j$: Deadline for job *j*

# Minimize Cost: Discrete Time MILP Model

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

= 1 if job $j$ starts at time point $t$ on machine $i$ $(t = 1,\ldots,N)$

$$\text{subject to} \quad \sum_{it} x_{ijt} = 1, \quad \text{all } j$$

Job $j$ starts at one time on one machine

$$\sum_{j} \sum_{\substack{t' \\ t-d_{ij} < t' \le t}} r_{ij} x_{ijt'} \le L_i, \quad \text{all } i,t$$

Jobs underway at time $t$ consume $\le L_i$ in resources

$$x_{ijt} = 0, \quad \text{all } j,t \text{ with } b_j - d_{ij} < t \le a_j$$

$$x_{ijt} = 0, \quad \text{all } j,t \text{ with } t > N - d_{ij} + 1$$

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

Jobs observe time windows

# Minimize Cost: Discrete Event MILP Model

$$
\begin{aligned}
\min \quad & \sum_{ijk} c_{ij} x_{ijk} \\
\text{subject to} \quad & \sum_{ik} x_{ijk} = 1, \quad \sum_{ik} y_{ijk} = 1, \quad \text{all } j \\
& \sum_{ij} x_{ijk} + y_{ijk} = 1, \quad \text{all } k \\
& \sum_{k} x_{ijk} = \sum_{k} y_{ijk}, \quad \text{all } i, j \\
& t_{i,k-1} \le t_{ik} \\
& \text{continued} \ldots
\end{aligned}
$$

$x_{ijk}$ = 1 if event $k$ is start of job $j$ on machine $i$ ($k = 1,\ldots,2N$)

$y_{ijk}$ = 1 if event $k$ is end of job

Each job is assigned to one machine and starts once and ends once

$t_{ik}$: Start time of event $k$ (disaggregated by machine)

Events in chronological order

Release date and deadline

Finish time of job *j* (disaggregated by machine)

Definition of finish time

$$a_j x_{ijk} \le t_{ik}, \quad f_{ij} \le b_j, \quad \text{all } i, j, k$$

$$t_{ik} + d_{ij} x_{ik} - M(1 - x_{ijk}) \le f_{ij} \le t_{ik} + d_{ik} x_{ijk} + M(1 - x_{ijk}), \quad \text{all } i, j, k$$

$$t_{ik} - M(1 - y_{ijk}) \le f_{ij} \le t_{ik} + M(1 - y_{ijk}), \quad \text{all } i, j, k$$

$$R_{ik} \le L_i, \quad \text{all } i, k$$

Resource limit

$$R_{i1} = R_{i1}^s, \quad R_{ik}^s = \sum_j r_{ij} x_{ijk}, \quad R_{ik}^f = \sum_j r_{ij} y_{ijk}, \quad \text{all } i, k$$

$$R_{ik}^s + R_{i,k-1} - R_{ik}^f = R_{ik}, \quad \text{all } i, k$$

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

Calculation of resource consumption on machine *i* at time of each event

# CP: The Cumulative Constraint

$$\text{cumulative}\begin{pmatrix} (t_1,\ldots,t_n) \\ (d_1,\ldots,d_n) \\ (r_1,\ldots,r_n) \\ L \end{pmatrix}$$

is equivalent to

$$\sum_{\substack{j \\ t_j \le t < t_j + d_{ij}}} r_j \le L \;, \quad \text{all}\, t$$

Schedules jobs at times $t_1, \ldots, t_n$ so as to observe resource constraint.

Edge-finding algorithms, etc., reduce domains of $t_j$.

# Minimize Cost: CP Model

$y_j$ = machine assigned to job $j$

$$\min \quad \sum_j c_{y_j j}$$

start times of jobs assigned to machine $i$

$$\text{subject to} \quad \text{cumulative}\begin{pmatrix} (t_j \mid y_j = i) \\ (d_{ij} \mid y_j = i) \\ (r_{ij} \mid y_j = i) \\ L_i \end{pmatrix}, \quad \text{all } i$$

$$a_j \le t_j \le b_j - d_{y_j j}, \quad \text{all } j$$

Observe time windows

Observe resource limit on each machine

# This is how it looks in OPL Studio…

```
[Declarations]
DiscreteResource machine [i in Machines] (Limit [i]);     <- enforces cumulative
AlternativeResources mset(machine);                        <- assigns jobs to machines
Activity sched [ j in Jobs];

minimize
  sum( j in Jobs) cost [ j ]
subject to {
  forall( j in Jobs) {
    sched [ j ] requires(jobm[i,j ].resource) mset;        <- defines resource requirements
    forall(i in Machines)
     activityHasSelectedResource(sched [ j ],mset,machine[ j ])
       <=> sched [ j ].duration = jobm[i,j ].duration &    <- determines cost and durations on the assigned machine
               cost [ j ] = jobm[i,j ].cost;
    sched [ j ].start >= job[ j ].release;
    sched [ j ].end <= job[ j ].deadline;                  <- time windows
  };
};
search {
  assignAlternatives;
  setTimes;                                                <- invokes specialized search procedure (needed for good performance)
};
```

# Inference Duality

| *Primal* | | *Dual* |
|---|---|---|
| $\min f(x)$ subject to $x \in S$, $x \in D$ | $=$ | $\max z$ subject to $x \in S \overset{x \in D}{\Rightarrow} f(x) \geq z$ |

$$\begin{array}{llcll} \min & f(x) & = & \max & z \\ \text{subject to} & x \in S & & \text{subject to} & x \in S \overset{x \in D}{\Rightarrow} f(x) \geq z \\ & x \in D & & & \end{array}$$

feasible set ($x \in S$)

domain ($x \in D$)

Solution is a value for *x*.

Solution is a *proof* that deduces $f(x) \geq z$ from $x \in S$.

Duality theorem based on completeness proof for inference method.

# Example of Inference Duality: Linear Programming

*Primal* | *Dual*

$$\min \ cx \quad \text{subject to} \quad Ax \ge b, \ x \ge 0 \quad = \quad \max \ z \quad \text{subject to} \quad Ax \ge b \overset{x\ge 0}{\Rightarrow} cx \ge z$$

**Farkas Lemma**

*Farkas Lemma*
(Completeness theorem for nonnegative linear combination as an inference method)

$Ax \ge b \Rightarrow cx \ge z$ iff
$uA \le c$, $ub \ge z$ for some $u \ge 0$
(provided primal or dual is feasible).

Solution is a *proof* that deduces $cx \ge z$ from $Ax \ge b$, encoded as a vector of multipliers $u$.

Farkas Lemma provides basis for classical LP duality theorem.

Since $u$ has polynomial size, LP belongs to NP and co-NP.

# Logic-Based Benders Decomposition

$$\begin{array}{ll} \min & f(x,y) \\ \text{subject to} & (x,y) \in S \\ & x \in D_x,\ y \in D_y \end{array}$$

Solution of master problem

*Master Problem*

$$\begin{array}{ll} \min\limits_{x,z} & z \\ \text{subject to} & z \ge B_{\bar{x}^k}(x), \text{ all } k \\ & x \in D_x,\ y \in D_y \end{array}$$

*Subproblem*

$$\begin{array}{ll} \min\limits_{y} & f(\bar{x},y) \\ \text{subject to} & (\bar{x},y) \in S \\ & y \in D_y \end{array}$$

Benders cuts for all iterations *k*

Solution of subproblem's inference dual is a proof of bound $z \ge B_{\bar{x}}(\bar{x})$ that is valid when $x = \bar{x}$

Benders cut is based on bound obtained from the *same proof schema* for other values of *x*.

# Example: Classical Benders

$$\begin{aligned} \min \quad & f(x) + cy \\ \text{subject to} \quad & g(x) + Ay \geq b \\ & x \in D_x,\ y \geq 0 \end{aligned}$$

*Master Problem*

$$\begin{aligned} \min_{x,z} \quad & z \\ \text{s.t.} \quad & z \geq f(x) + u^k(b - g(x)),\ \text{all } k \\ & x \in D_x \end{aligned}$$

Benders cuts

*Subproblem*

$$\begin{aligned} \min_{y} \quad & f(\bar{x}) + cy \\ \text{s.t.} \quad & Ay \geq b - g(\bar{x}) \\ & y \geq 0 \end{aligned}$$

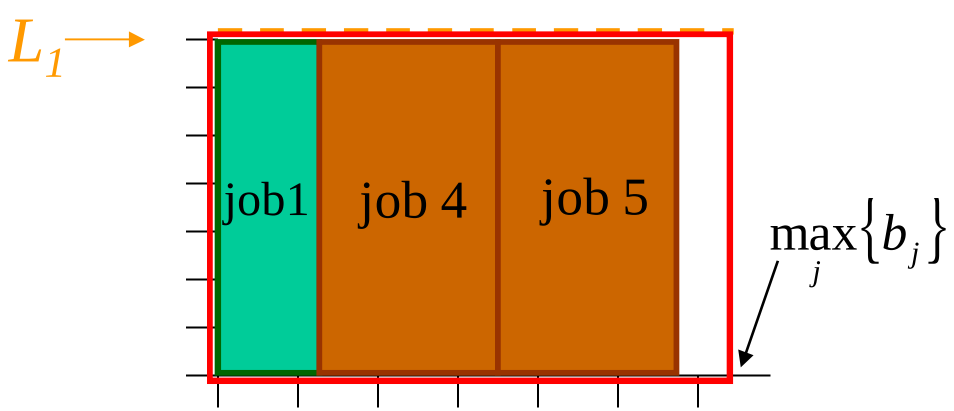
Solution of subproblem's inference dual is a proof (encoded by *u*) of bound $z \geq f(\bar{x}) + u(b - g(\bar{x}))$

Benders cut is based on bound obtained from *u,* which remains dual feasible (i.e., proof remains valid) for other values of *x*.

# Minimize Cost: Logic-Based Benders

*Master Problem: Assign jobs to machines*

$$\begin{aligned} \min \quad & \sum_{ij} c_{ij} x_{ij} \\ \text{subject to} \quad & \sum_i x_{ij} = 1, \quad \text{all } j \\ & \sum_j d_{ij} r_{ij} x_{ij} \le L_i \max_j \{b_j\}, \quad \text{all } i \\ & \text{Benders cuts} \end{aligned}$$




*Relaxation of subproblem:* "Area" $d_{ij}r_{ij}$ of jobs assigned to a machine fit in the space available before latest deadline.

## *Subproblem: Schedule jobs assigned to each machine*

## Solve by constraint programming

solution of master problem

$$\left\{ \begin{array}{l} \text{cumulative}\begin{pmatrix} (t_j \mid \bar{x}_{ij} = 1) \\ (d_{ij} \mid \bar{x}_{ij} = 1) \\ (r_{ij} \mid \bar{x}_{ij} = 1) \\ L_i \end{pmatrix} \\ a_j \le t_j \le b_j \end{array} \right\}, \quad \text{all } i$$

Let $J_{ik}$ = set of jobs assigned to machine *i* in iteration *k.*

If subproblem *i* is infeasible, solution of subproblem dual is a proof that not all jobs in $J_{ik}$ can be assigned to machine *i.*

This provides the basis for a (trivial) Benders cut.

## *Master Problem with Benders Cuts*
## Solve by MILP

$$
\begin{aligned}
\min \quad & \sum_{ij} c_{ij} x_{ij} \\
\text{subject to} \quad & \sum_{i} x_{ij} = 1, \quad \text{all } j \\
& \sum_{j} d_{ij} r_{ij} x_{ij} \le L_i \max_j \{b_j\}, \quad \text{all } i \\
& \sum_{j \in J_{ik}} (1 - x_{ij}) \ge 1, \quad \text{all } i, k \\
& x_{ij} \in \{0,1\}
\end{aligned}
$$

Benders cuts

*Important observation:* Putting a **relaxation of subproblem** in the master problem is essential for success.

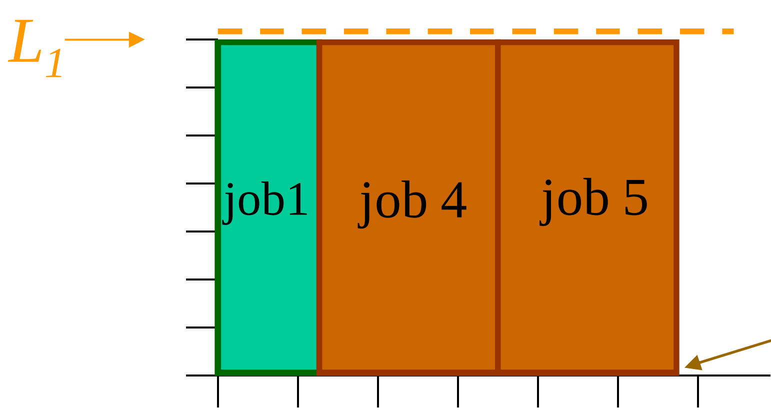
Min cost problem is particularly easy for logic-based decomposition:

| | *Min cost* | *Min makespan* |
|---|---|---|
| *Objective function* | Computed in master problem, which yields tighter bounds for MILP | Available only thru Benders cuts. |
| *Subproblem* | Feasibility problem, simple Benders cuts | Optimization problem (harder for CP), more interesting cuts |
| *Relaxation* | Trivial | More interesting, nice duality with cuts |

# Minimize Makespan: Logic-Based Benders

*Master Problem: Assign jobs to machines*

$$\begin{aligned} &\min && T \quad \leftarrow \text{makespan} \\ &\text{subject to} && \sum_i x_{ij} = 1, \quad \text{all } j \\ &&& T \geq \frac{\sum_j d_{ij} r_{ij} x_{ij}}{L_i}, \quad \text{all } i \\ &&& \text{Benders cuts} \end{aligned}$$




*Relaxation of subproblem:*
"Area" of jobs provides lower bound on makespan.

## *Subproblem: Schedule jobs assigned to each machine*
## Solve by constraint programming

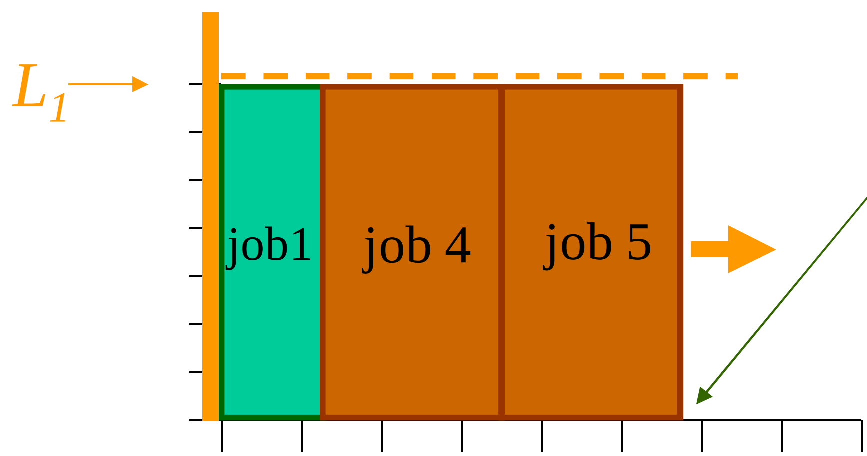
$$\min \quad T$$

$$\text{subject to} \quad \left\{ \begin{array}{l} T \ge t_j + d_{ij}, \quad \text{all } j \\ \text{cumulative}\begin{pmatrix} (t_j \mid \bar{x}_{ij} = 1) \\ (d_{ij} \mid \bar{x}_{ij} = 1) \\ (r_{ij} \mid \bar{x}_{ij} = 1) \\ L_i \end{pmatrix} \\ a_j \le t_j \le b_j, \quad \text{all } j \end{array} \right\}, \quad \text{all } i$$

Let $J_{ik}$ = set of jobs assigned to machine $i$ in iteration $k$.

We get a Benders cut even when subproblem is feasible.

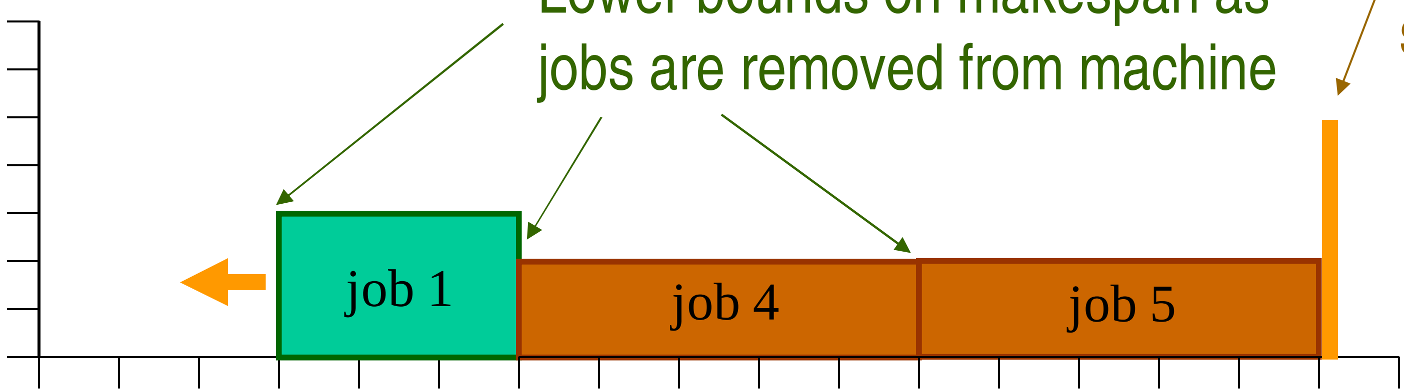
# Duality of Linear Relaxation and Linear Benders Cuts




*Relaxation:*
Lower bound on makespan

*Benders cut:*
Lower bounds on makespan as jobs are removed from machine

Minimum makespan for machine *i* in subproblem



## *Master Problem: Assign jobs to machines*
## Solve by MILP

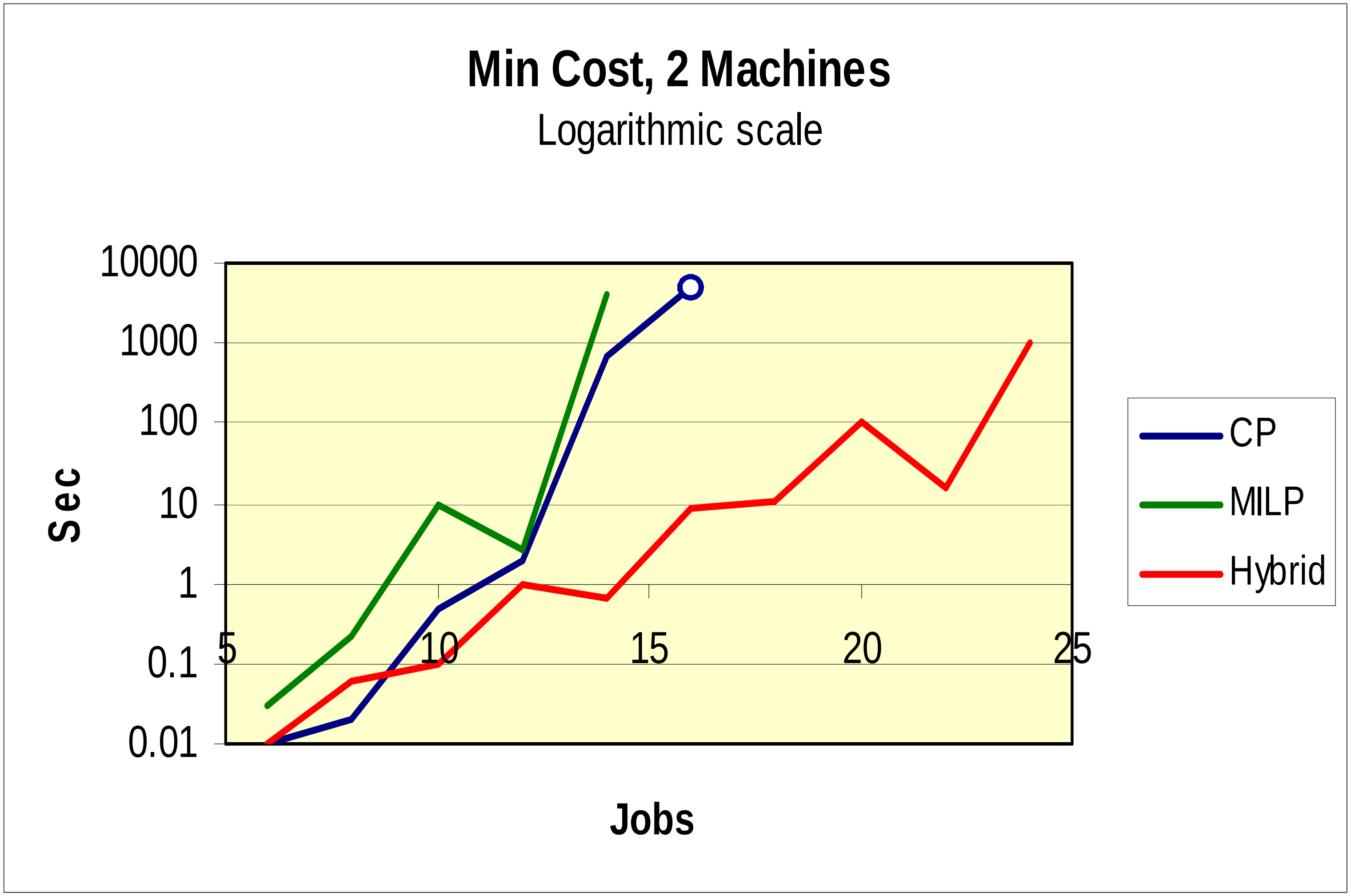
$$
\begin{aligned}
\min \quad & T \\
\text{subject to} \quad & \sum_i x_{ij} = 1, \quad \text{all } j \\
& T \geq \frac{\sum_j d_{ij} r_{ij} x_{ij}}{L_i}, \quad \text{all } i \qquad \leftarrow \text{Relaxation} \\
& T \geq T_k - \sum_{j \in J_{ik}} (1 - x_{ij}) d_{ij} \qquad \leftarrow \text{Benders cuts} \\
& x_{ij} \in \{0,1\}
\end{aligned}
$$

# Experimental Design

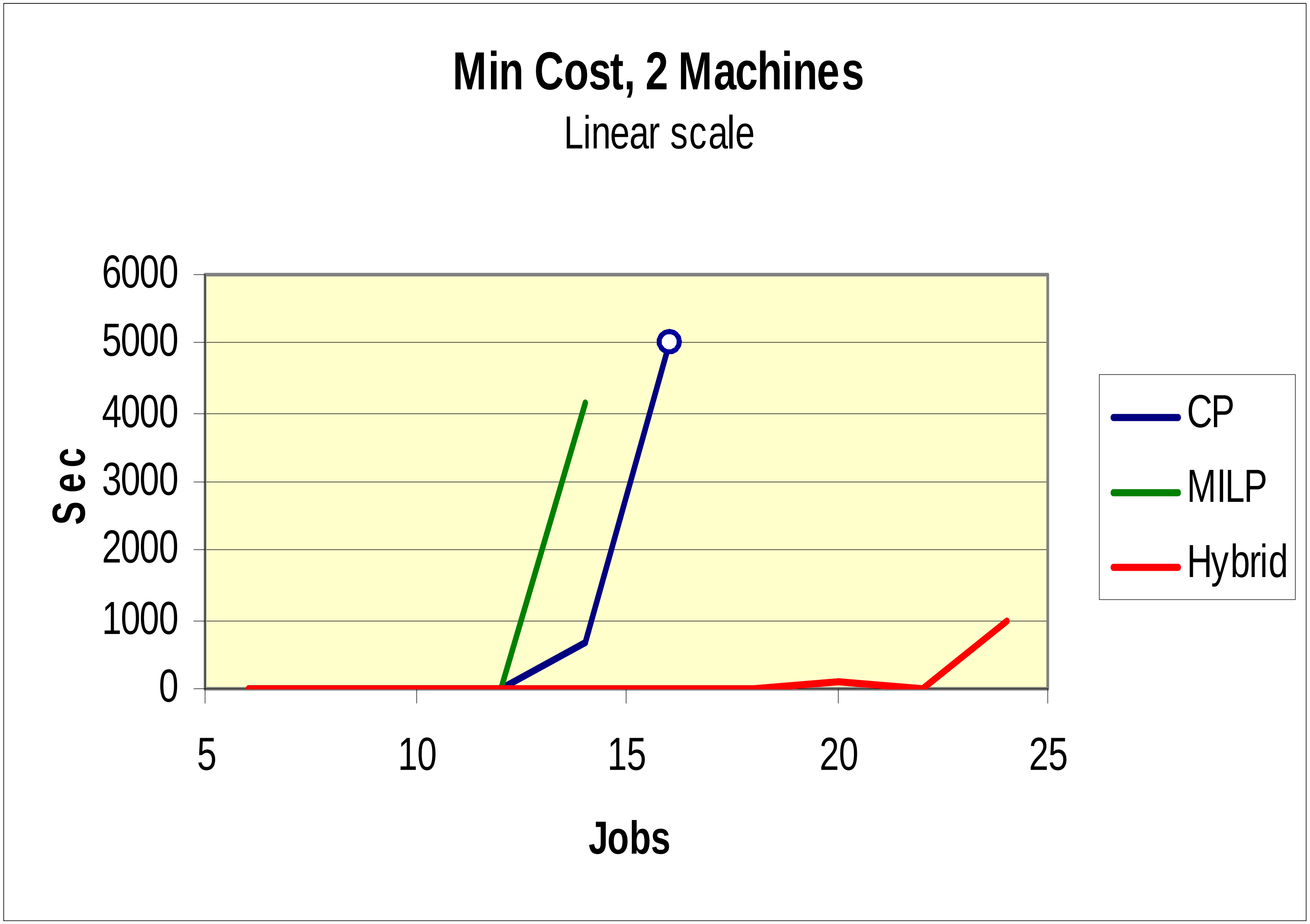
| | 2 Machines | 3 Machines | 4 Machines |
|---|---|---|---|
| CP (ILOG Scheduler) | 6, 8, 10, … jobs* | 6, 8, 10, … jobs | 6, 8, 10, … jobs |
| MILP (Discrete time model**) | 6, 8, 10, … jobs | 6, 8, 10, … jobs | 6, 8, 10, … jobs |
| Hybrid | 6, 8, 10, … jobs | 6, 8, 10, … jobs | 6, 8, 10, … jobs |

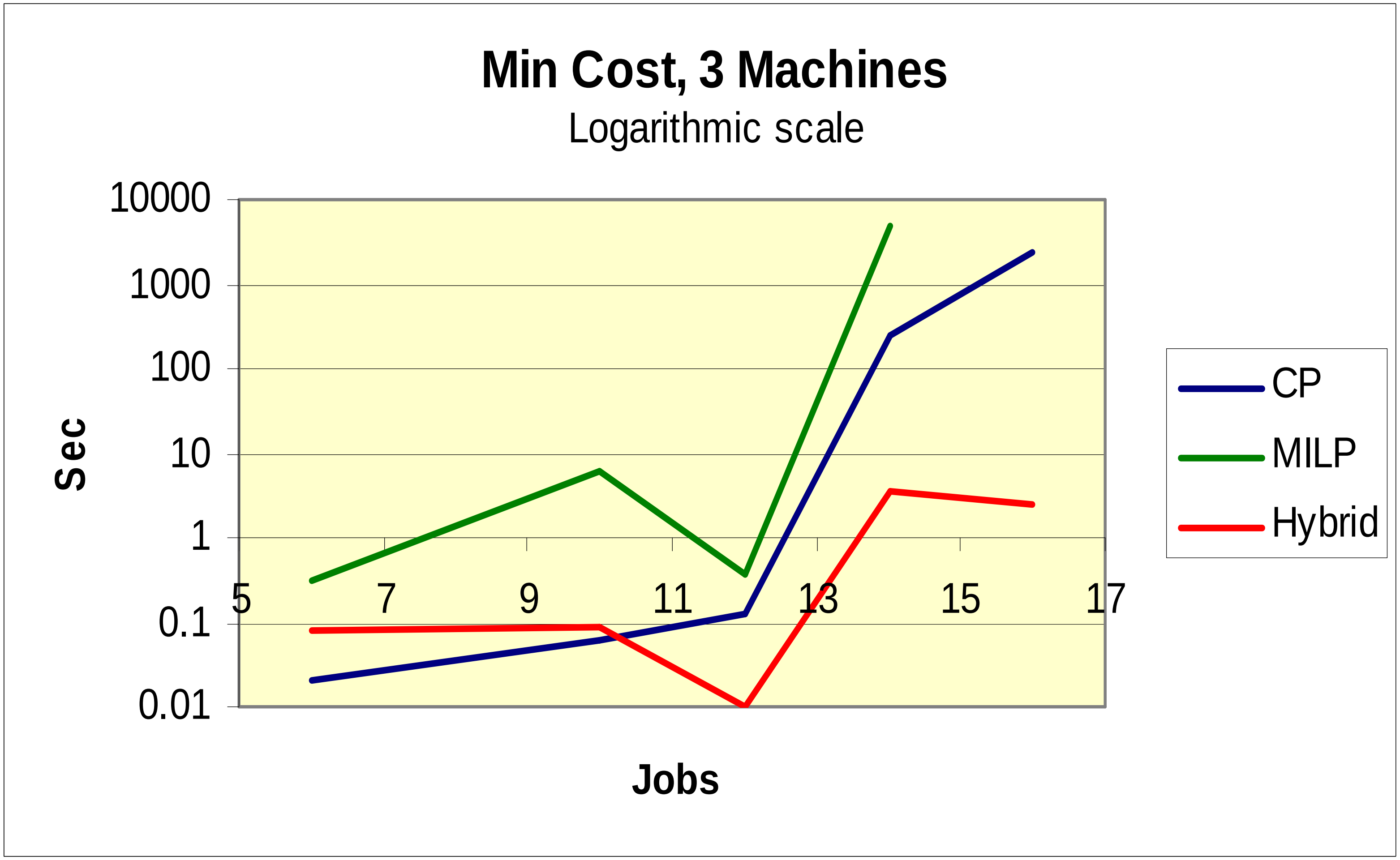
*1 problem instance each

** Discrete event model solved none of the problems

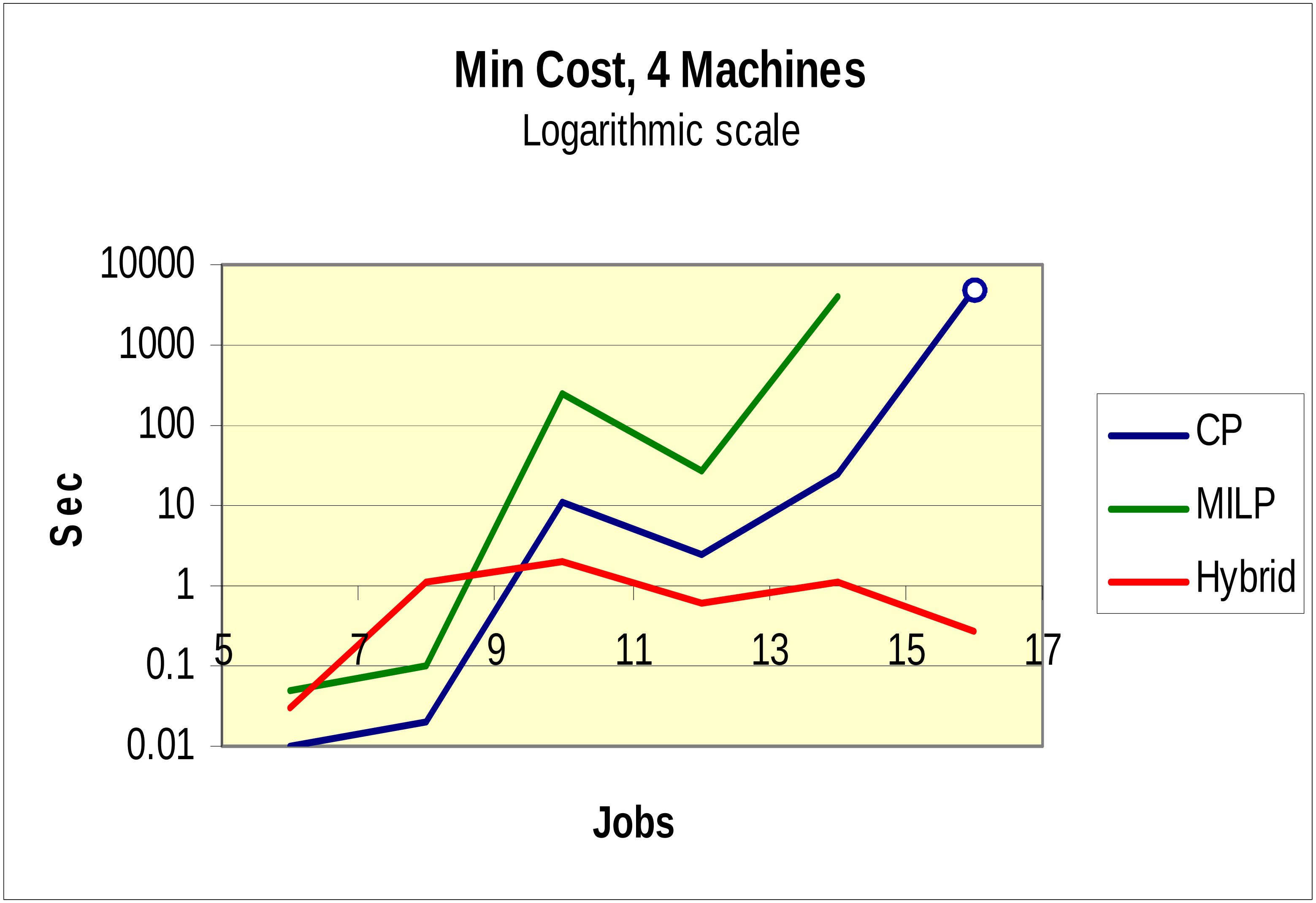


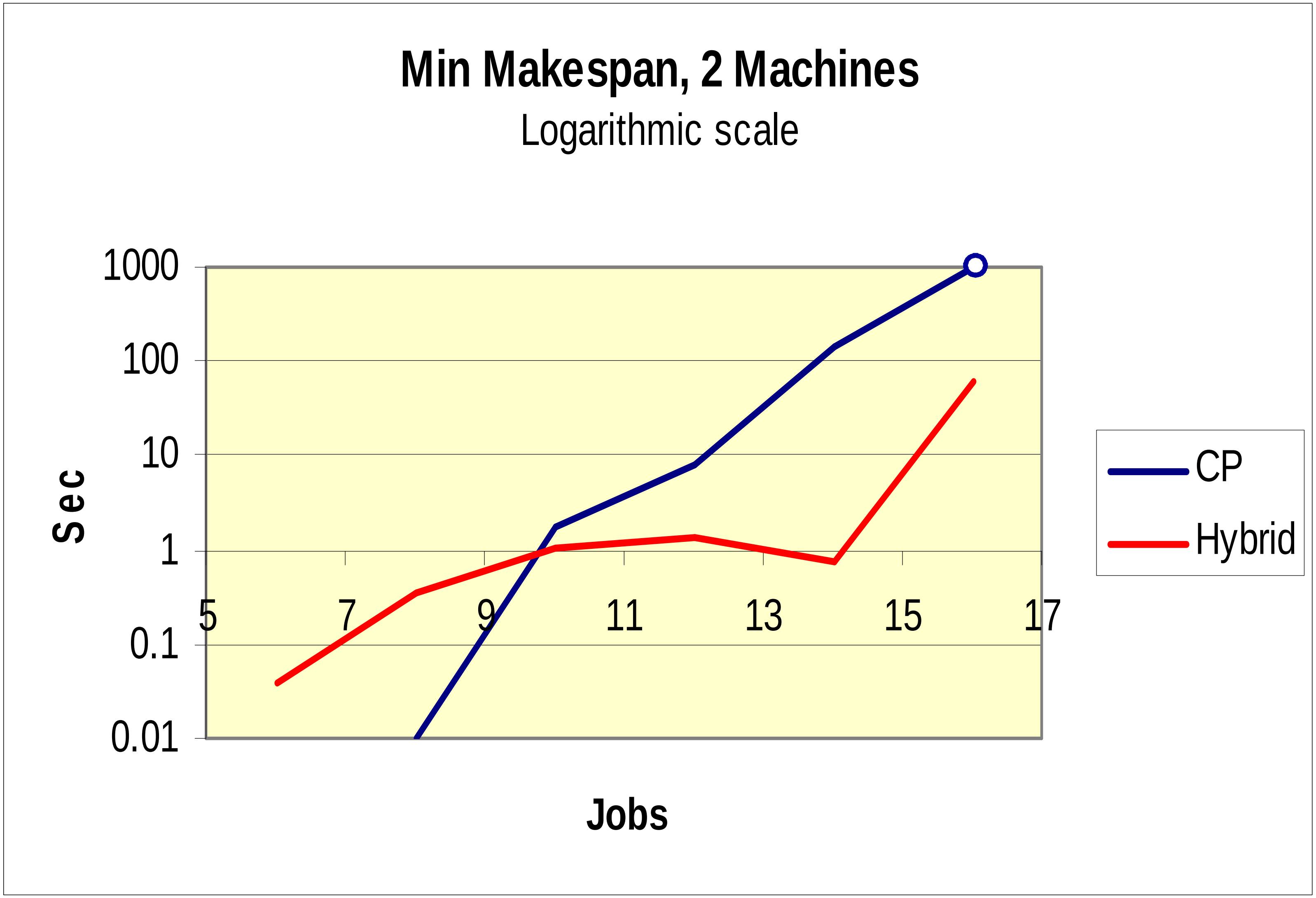

○ = computation terminated

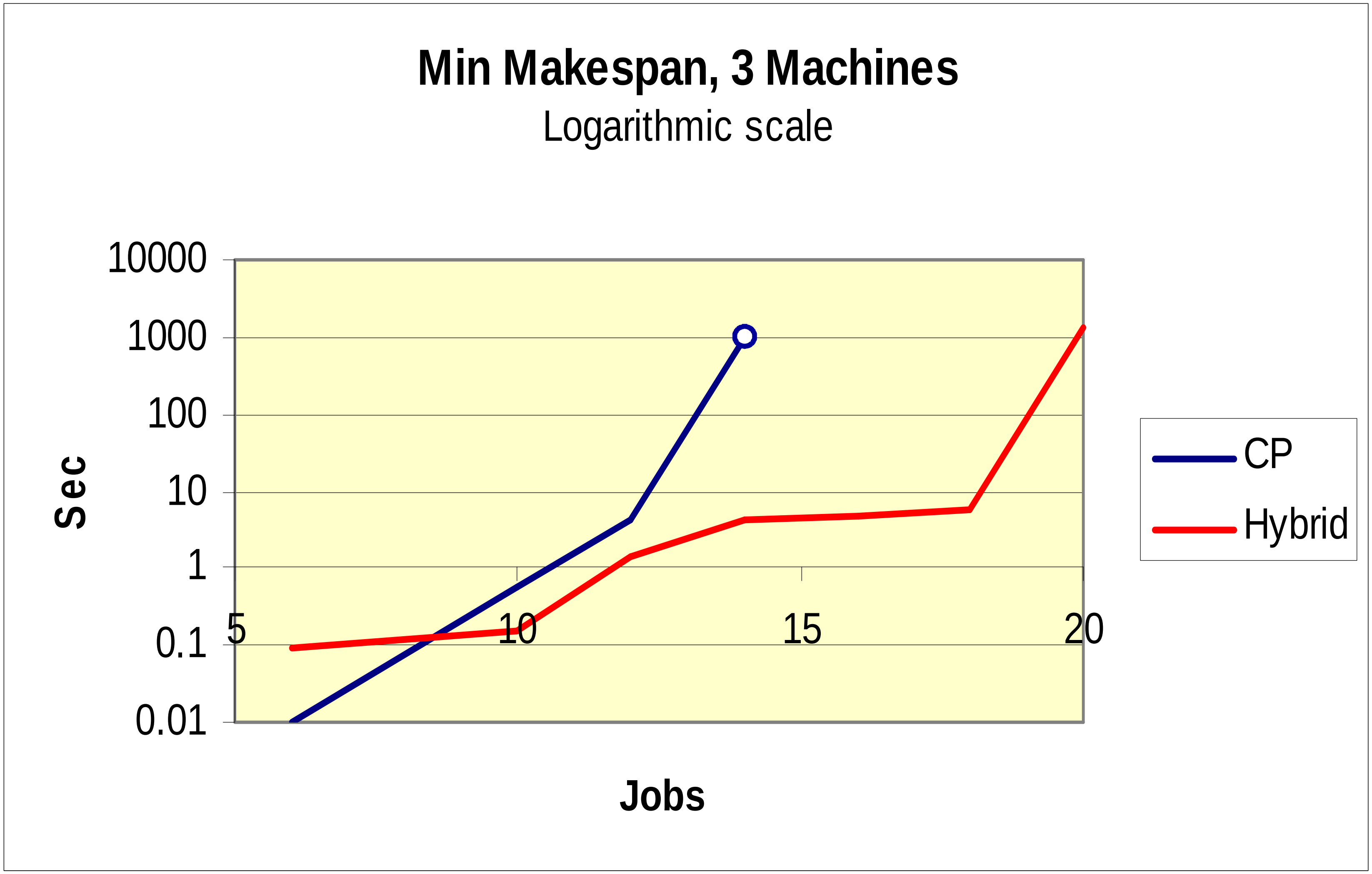

Min Cost, 2 Machines
Linear scale
Sec
6000
5000
4000
3000
2000
1000
0
5
10
15
20
25
Jobs
CP
MILP
Hybrid

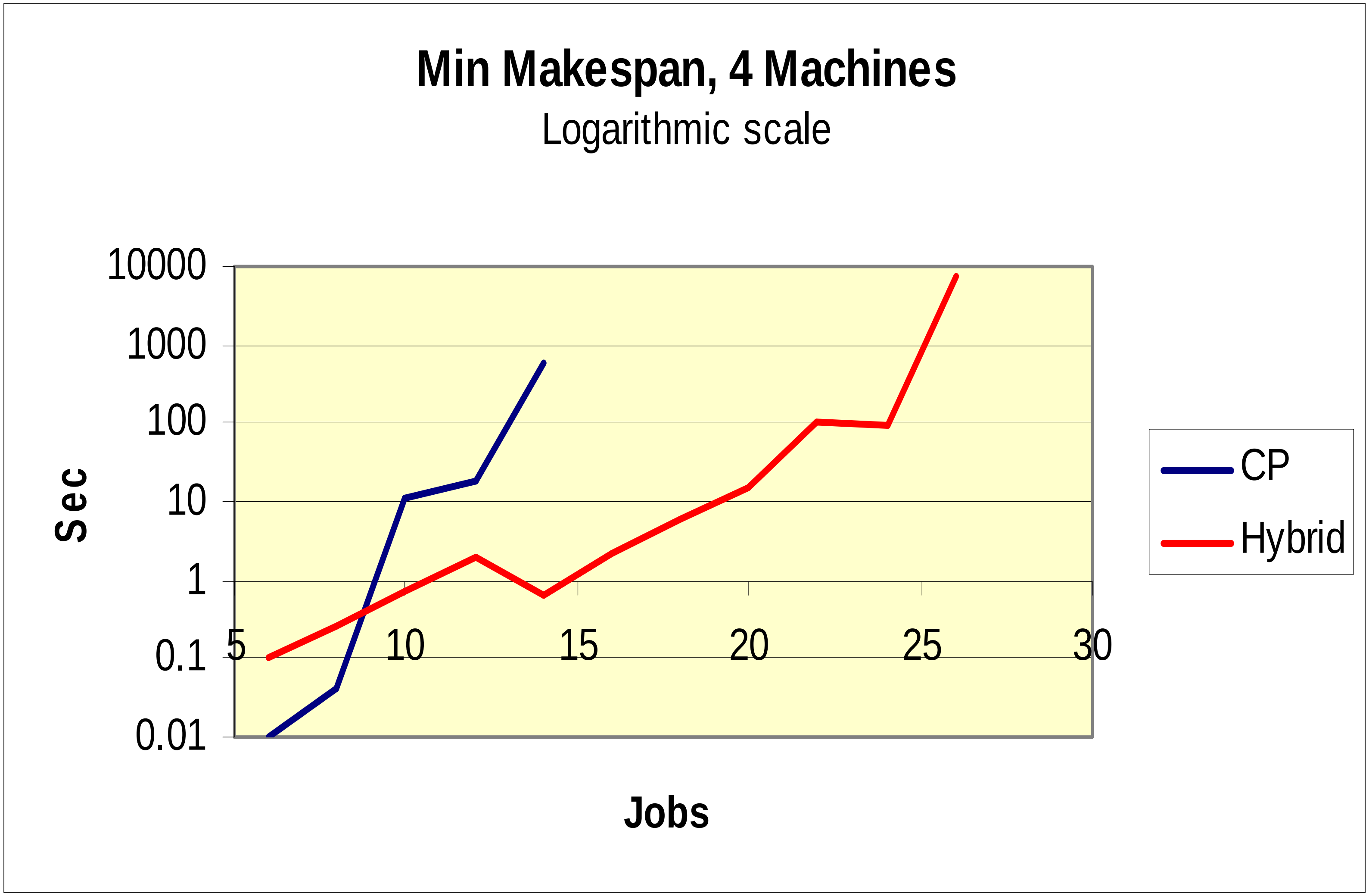


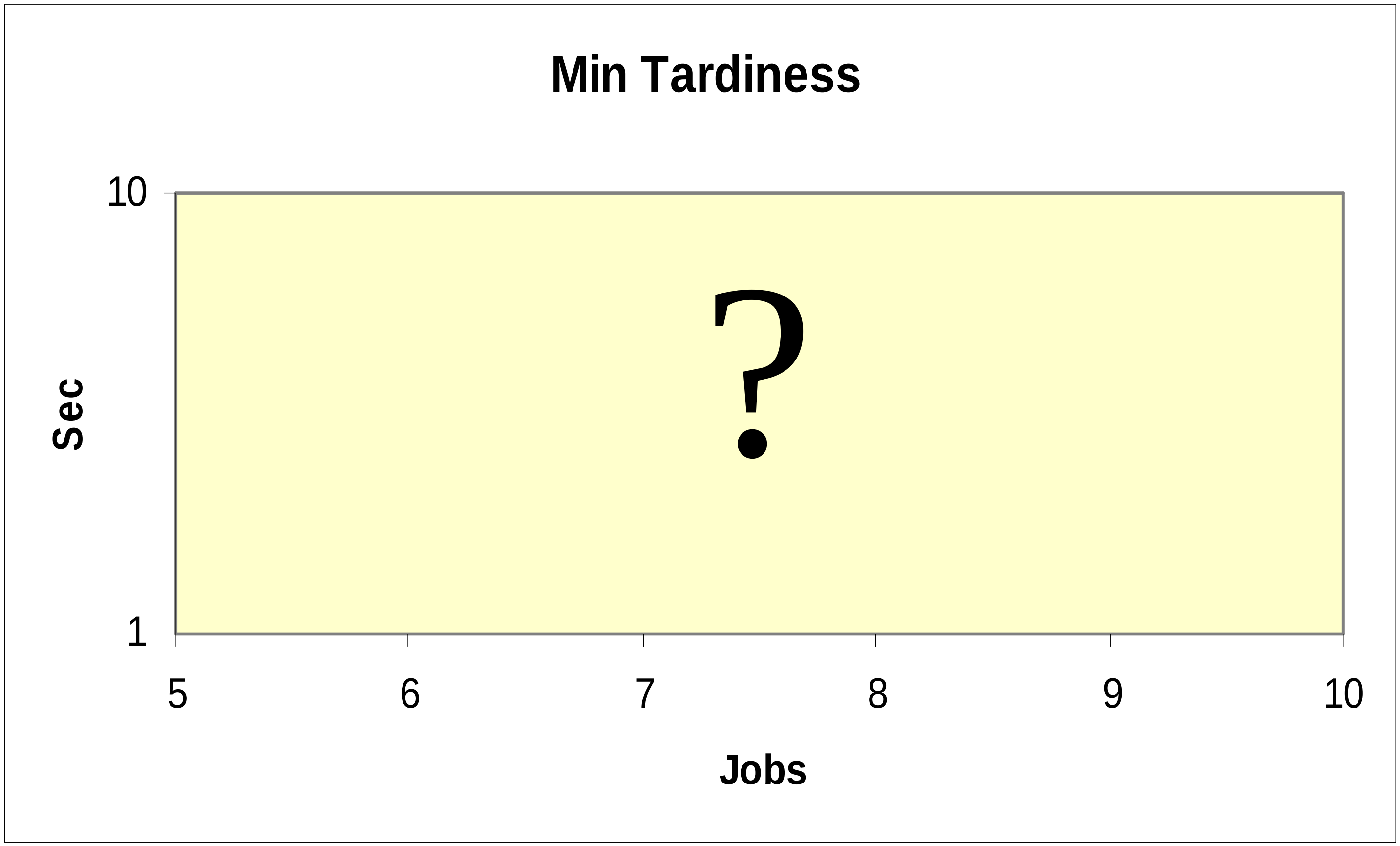
Min Cost, 3 Machines
Logarithmic scale
Sec
10000
1000
100
10
1
0.1
0.01
5
7
9
11
13
15
17
Jobs
CP
MILP
Hybrid


Min Cost, 4 Machines
Logarithmic scale
Sec
10000
1000
100
10
1
0.1
0.01
5
7
9
11
13
15
17
Jobs
CP
MILP
Hybrid


Min Makespan, 2 Machines
Logarithmic scale
Sec
1000
100
10
1
0.1
0.01
5
7
9
11
13
15
17
Jobs
CP
Hybrid


Min Makespan, 3 Machines
Logarithmic scale
Sec
10000
1000
100
10
1
0.1
0.01
5
10
15
20
Jobs
CP
Hybrid


Min Makespan, 4 Machines
Logarithmic scale
Sec
10000
1000
100
10
1
0.1
0.01
5
10
15
20
25
30
Jobs
CP
Hybrid


Min Tardiness
Sec
10
1
?
5
6
7
8
9
10
Jobs