Scheduling an Industrial Production Facility:
Applying theory in practice

Cliff Stein
Columbia University

http://www.columbia.edu/~cs2035

Joint work with
E. Asgiersson (Columbia University)
J. Berry (Lafayette College)
C. Phillips (Sandia National Labs)
D. Phillips (Columbia University)

J. Wein (Polytechnic University)

Overview

- Introduce the application
- Formulation as a large integer program
- Background from relevant theory
- Discussion of solution methods and experimental data

Messages:

- Some success with a real application.
- Ideas developed for (contrived?) theoretical problems have payoffs in a practical problem and lead to interesting methods in practice.

Swords to Plowshares

Background, described in [Kjelgaard et. al. 2000].

- “The end of the Cold War changed the missions of the facilities in the US nuclear weapons complex. They ceased production of new weapons and focused on dismantling old weapons and maintaining the safety, security and reliability of those remaining”
- Pantex Plant (DOE) is the sole facility for dismantlement, evaluation and maintenance activities for US nuclear stockpile.
- The Pantex Process Model (PPM) is a decision support tool to help Pantex plan capacity and deploy resources effectively. It does Evaluation Planning.
- PPM has provided “critical input to help the US form and defend position during arms-control-treaty negotiations.”
- “PPM has . . . resulted in Pantex exceeding weapon-dismantlement goals”

Pantex Plant

- 2850 employees
- $288M annual budget
- Pantex's particular problems are similar to those in many manufacturing facilities
  - effectively utilizing facilities
  - effectively utilizing people
  - meeting output requirements
- Differences are that output requirements are specified by US national policy, international treaties, and technical needs to evaluate and repair weapons.

Sample uses of PPM

- How will a proposed treaty stipulation affect Pantex's ability to accomplish its tests, and what recommendation should it make to US negotiators on this proposal?
- How will treaty verification activities affect Pantex operations
- Does Pantex have the resources to meet the requirements of the proposed treaty agreements?
- How rapidly can Pantex dismantle the weapons of a particular types, given its other responsibilities?
- Does Pantex have enough staging facilities for incoming weapons and storage areas for parts from dismantles weapons?
- Has Pantex trained enough technicians to run a planned dismantlement program?

Answering these questions requirements planning on a time scale of months or even several years.

High Level Problem Description

Evaluation and dismantlement are somewhat different technically, we will focus on evaluation. (Dismantlement has assembly-line (job shop) aspects that we will not deal with.)

Evaluation: Individual weapons are sampled from the enduring stockpile.

- Each weapon evaluation consists of a sequence of tasks.
- Precedence constraints between tasks.
- Due dates for intermediate tasks.
- Some deadlines are rigid.
- Some tasks have higher priority.
- Each tasks need certain facilities and technicians with certain certifications.

More details about facilities and technicians

- **Technicians need to be trained and certified for each operation.**
- **There are about 300 technicians each holding up to 5 out of 100 certifications.**
- **Operations require crews of two to four technicians, each holding certain certifications.**
- **There is a constraint about total radiation exposure for any technician.**
- **82 facilities organized into 29 types. These types form a hierarchy.**
- **Additional constraints because of laws or constraints on materials being in same place.**

Demand Levels

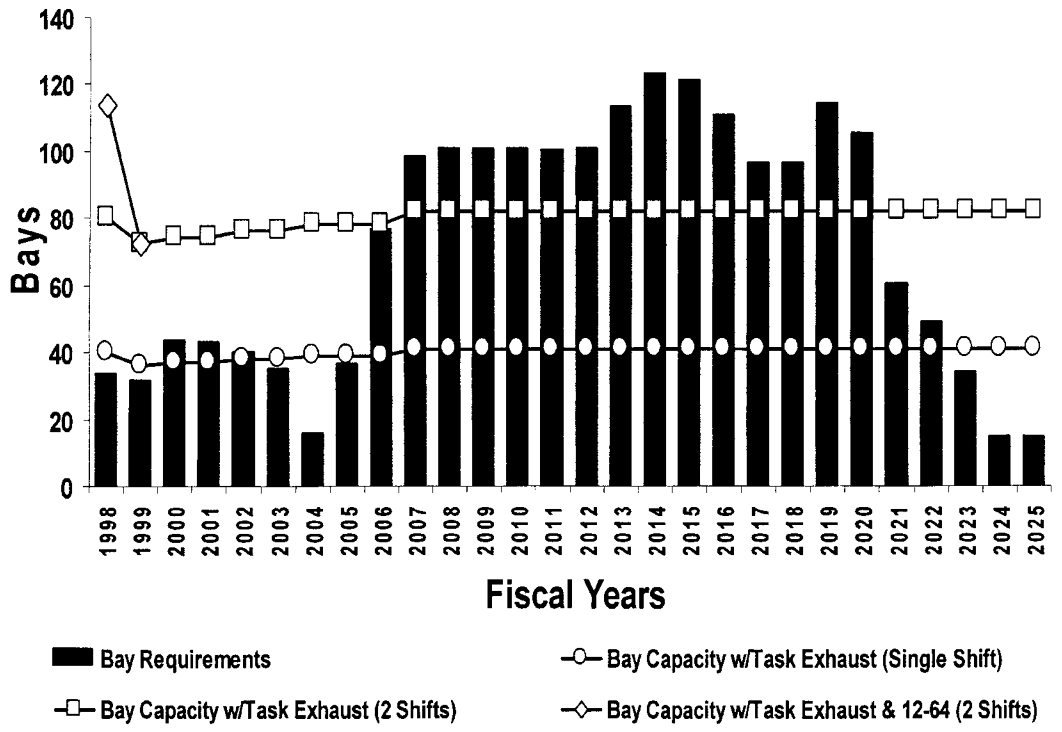


Figure 3: The demand for weapons operations bays is based on the overall workload, including treaty-related work. The two lines indicate single- and double-shift capacity per fiscal year. These requirements are based on the entire anticipated workload.

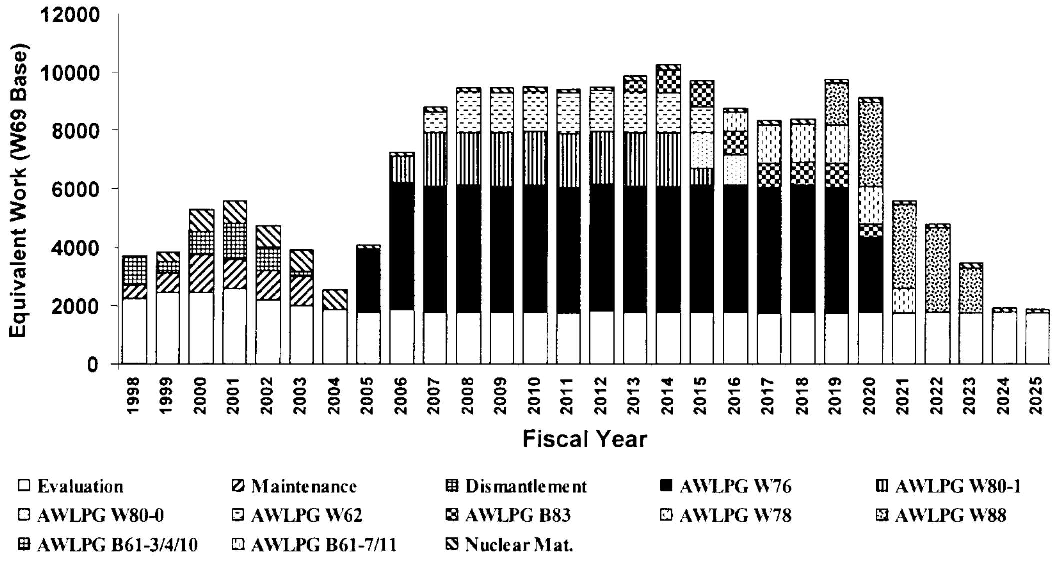


Figure 4: The demand for production technicians by workload type varies over a 28-year pe

Objective

- Jobs must be done by their deadlines and according to the given constraints.
- The goal is to minimize
  - the number of extra technicians that need to be hired and trained
  - the number of extra facilities that must be modified or constructed

Some data

- \$17M to modify an existing facility.
- \$70M to build a new facility.

Objective

- Jobs must be done by their deadlines and according to the given constraints.
- The goal is to minimize
  - the number of extra technicians that need to be hired and trained
  - the number of extra facilities that must be modified or constructed

Some data

- \$17M to modify an existing facility.
- \$70M to build a new facility.

The planning and evaluation problem can be formulated as a large integer program.

Problem description (for problem we will solve)

- n jobs
  - Each job consists of a set of tasks, related by precedence constraints.
  - Precedence constraints are outforrests and are "long and thin."
  - Each task has a
    * Processing time p_j (hours to months)
    * release date r_j
    * Hard deadline d_j
    * Processing is non-preemptive.
    * Must be processed in a facility of type k_j .
    * Requires a crew of size s_j , each having certification c_j
- Time:
- A week consists of 6 days, each consisting of 8 hours. Task lengths are given in hours.
- Tasks are scheduled at the granularity of hours.
- Tasks that take less than a day must be completed in one day.
- Facilities and technicians are scheduled at the granularity of an hour. (This may be overly optimistic.)

Recall that purpose is planning.

Feasible schedule

Tasks are assigned to time slots so that

1. all precedence constraints, release dates, and deadlines are obeyed,
2. short tasks are completely scheduled within a single day,
3. the total amount of work scheduled for each facility type during any particular week does not exceed the availability of such facilities,
4. for each week the requirements for technicians are matched by qualified technician assignments and each technician is assigned no more than a week of work, and
5. for each week, no technician is assigned to a particular certification for more hours than the sum of the task lengths (within that week) of tasks requiring that certification. (This prevents serialization of parallel task work, that is, allowing one worker to do the work of two provided he works twice as long.)

In order to satisfy all constraints, we need to possibly introduce additional ghost facilities and ghost technicians. We minimize the total number of hours used by ghost technicians and ghost facilities.

Scale

- **hundreds of jobs**
- **thousands of tasks**
- **30 facility types**
- **300 technician, each of whom holds 2–5 of the 80 – 100 possible certifications.**

Example: Single weapon

Task	r	d	p	**crew size**	**cert. req.**	**facility req.**
1	0	4	2	2	A or B	1
2	0	4	4	2	B	1
3	0	9	3	3	A	2
4	0	14	6	2	B	1

Technician	**Certification**
1	**A**
2	**B**
3	**A and B**

Precedence:

$$1, 2 \prec 3 \prec 4$$

Goals

1. Improve on the current methods used by Pantex evaluation model.
2. Gain insight into solving large industrial scheduling problems.
3. Gain understanding into the trade-off between heuristic and exact-solution methods for these types of problems.
4. Understand the value of α-point scheduling in real-life (no proof available) settings.

Some related work

- Most similar problem is resource constrained project scheduling with a objective of resource leveling.
- Similar but more complicated than $PS_{m,\infty}|temp|\Sigma c_i f(S,i)$ (nomenclature of [Brucker, et al. (1999)]), where $f(S,i)$ denotes the amount of resource i that exceed threshold over the course of schedule S (also called overload), and c_i is the unit cost for resource i .
- Exact methods (e.g., [Zimmermann and Engelhardt (1998)]) and heuristics (e.g., [Brinkmann and Neumann (1996)]) have been proposed and tested on small instances.
- Other applications in finance and construction.

Integer Program

Input parameters/constants/shorthand

p_j, r_j, d_j	The processing time (in hours), release day and deadline day of task j
ρ_j	The minimum number of full days needed to process task j .
$T(j)$	Set of possible start days for task j.
$p(j,t,\tau)$	Number of hours work on task j performed during week τ if j is started
$T(j,\tau)$	Set of start times t for task j such that $p(j,t,\tau) > 0$.
k_j, c_j, s_j	Facility type, technician certification and crew size for task j.
$C(w)$	The set of certifications held by worker w
$f_{k,\tau}$	The number of hours available in facilities of type k during week τ.
$b(j,j')$	1 if $j \prec j'$ and task j' can be "packed" with j, 0 otherwise

Integer variable

$$x_{jt} = \begin{cases} 1 & \text{if task } j \text{ starts on day } t \\ 0 & \text{otherwise} \end{cases}$$

Fractional variables

$y_{wc\tau}$	$=$	fraction of time worker w spends using certification c in week τ
$F_{k\tau}$	$=$	number of ghost facilities of type k in week τ
$G_{c\tau}$	$=$	number of ghost technicians with certification c in week τ

Time-indexed MIP formulation

$$\textbf{(TILP) minimize } \sum_{k,\tau} F_{k\tau} + \sum_{c,\tau} G_{c,\tau}$$

subject to

$$\begin{array}{rcll}
\sum_{t\in T(j)} x_{jt} & = & 1 & \forall j \quad (1)\\
\sum_{c\in C(w)} y_{wc\tau} & \leq & 1 & \forall w, \tau \quad (2)\\
x_{jt} & \leq & \sum_{r_{j'}\leq t'\leq t-\rho_{j'}+b(j',j)} x_{j't'} & \forall t\in T(j),\ \ j' \prec j \quad (3)\\
\sum_{j:k=k_j}\sum_{t\in T(j,\tau)} p(j,t,\tau)x_{jt} & \leq & f_{k,\tau} + 48F_{k\tau} & \forall \tau, k \quad (4)\\
\sum_{j:c_j=c}\sum_{t\in T(j,\tau)} p(j,t,\tau)s_j x_{jt} & \leq & \sum_w 48y_{wc\tau} + 48G_{c\tau} & \forall \tau, c \quad (5)\\
48y_{wc\tau} & \leq & \sum_{j:c_j=c}\sum_{t\in T(j,\tau)} p(j,t,\tau)x_{jt} & \forall c,\tau,w : c\in C(w) \quad (6)\\
x & \in & \{0,1\} & \quad (7)
\end{array}$$

Explanation of TLIP

$$minimize \ \sum_{k,\tau} F_{k\tau} + \sum_{c,\tau} G_{c,\tau}$$

Counts ghost hours (facilities plus technicians)

$$\sum_{t \in T(j)} x_{jt} = 1 \ \ \forall j$$

ensures that every task is completed.

$$\sum_{c \in C(w)} y_{wc\tau} \leq 1 \ \ \forall w, \tau$$

prevents over-scheduling any technician in any week.

$$x_{jt} \leq \sum_{r_{j'} \leq t' \leq t - \rho_{j'} + b(j',j)} x_{j't'} \ \ \forall t \in T(j), \ \ j' \prec j$$

ensures precedence constraints.

Explanation of TLIP

$$\sum_{j:k=k_j} \sum_{t \in T(j,\tau)} p(j,t,\tau) x_{jt} \leq f_{k,\tau} + 48 F_{k\tau} \ \ \forall \tau, k$$

ensures that there is sufficient facility capacity for work each week.

$$\sum_{j:c_j=c} \sum_{t \in T(j,\tau)} p(j,t,\tau) s_j x_{jt} \leq \sum_{w} 48 y_{wc\tau} + 48 G_{c\tau} \ \ \forall \tau, c$$

ensures that there is sufficient workers available each week.

$$48 y_{wc\tau} \leq \sum_{j:c_j=c} \sum_{t \in T(j,\tau)} p(j,t,\tau) x_{jt}];] \forall c, \tau, w : c \in C(w)$$

prevents one technician doing two hours of work simultaneously.

Solving the IP

Variables

- One for each task at each time period.
- If we take one year at a granularity of an hour, we have about 2.5M integer variables.
- If we use granularity of a day, we still have 300K integer variables. This is beyond the capabilities of current computers/software.

Our IP framework

- IP – used both CPLEX and PICO (parallel integer combinatorial optimizer) [Eskstein,Phillips,Hart 2000] developed at Sandia and RUTCOR.
- Use Cplex as LP-engine for small runs, and COIN LP for the large parallel runs.
- PICO is
  - easily configured with problem-specific branching rules, lower bounds and heuristics.
  - Still evolving.
  - Still evolving, not yet able to solve large instances of the Pantex problem.
- Cplex is not able to solve these large problems either.

We need ways to generate good, but not necessarily optimal solutions.

Our algorithmic framework

1. **Solve the linear programming relaxation of TILP to obtain x^*_{jt} .**
2. **Call the** Best-α **or** Fixed-α **subroutine to convert the TILP solution x^*_{jt} to a set, Π , of priority orderings for the jobs.**
3. **For each ordering $\pi \in \Pi$, call the** Comb **or** Greedy **subroutine to determine a schedule $\omega(\pi)$ for the tasks**
4. **Use local search to try to improve the objective for the schedules**

Our algorithmic framework

1. **Solve the linear programming relaxation of TILP to obtain x^*_{jt} .**
2. **Call the** BEST-α **or** FIXED-α **subroutine to convert the TILP solution x^*_{jt} to a set, Π , of priority orderings for the jobs.**
3. **For each ordering $\pi \in \Pi$, call the** COMB **or** GREEDY **subroutine to determine a schedule $\omega(\pi)$ for the tasks**
4. **Use local search to try to improve the objective for the schedules**

The use of α-points is a key contribution.

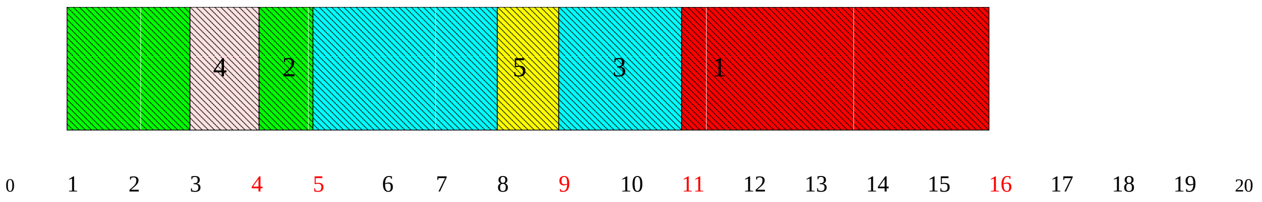
Digression: Understanding α-points

Approximating the NP-hard problem $1|r_j|\sum C_j$ [Phillips, Stein, Wein '95]

Sample input:

t j	r_j	p_j
1	0	6
2	1	3
3	1	5
4	3	1
5	8	1

Step 1: **Find the optimal preemptive schedule via Shortest Remaining Processing Time algorithm. [Baker '74]**



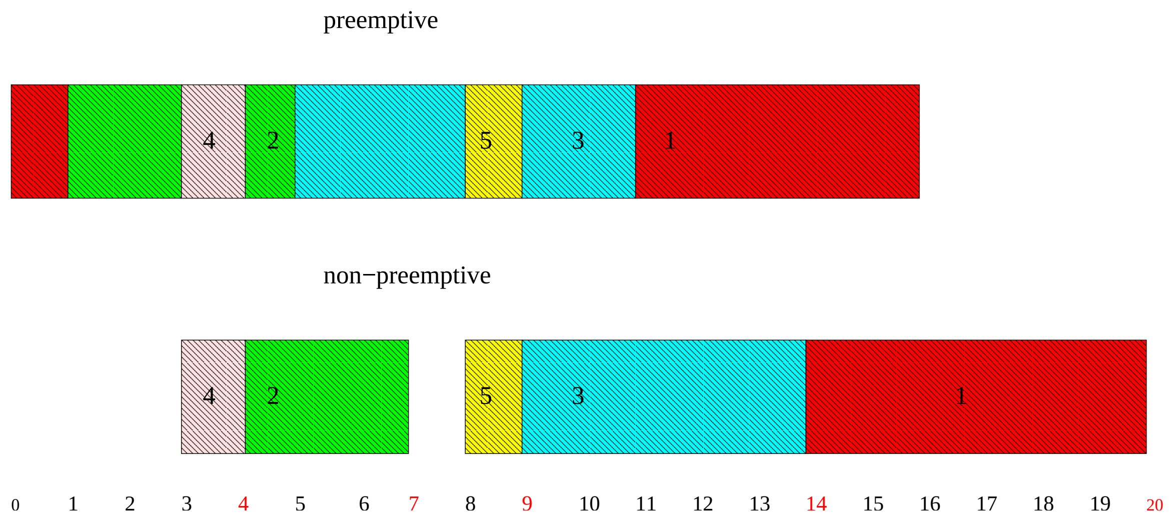
$$\sum C_j^P = 4 + 5 + 9 + 11 + 16 = 45$$

(Clearly $\sum C_j^P \leq OPT$ **)**

2-approximation algorithm

j	r_j	p_j
1	0	6
2	1	3
3	1	5
4	3	1
5	8	1

Step 2: **Non-preemptively schedule the jobs in the order they complete in the preemptive schedule, respecting release dates.**



$$\sum C_j^P = 4 + 5 + 9 + 11 + 16 = 45$$

$$\sum C_j^N = 4 + 7 + 9 + 14 + 20 = 54$$

Analysis and Generalization

Lemma: For each job, $C_j^N \leq 2C_j^P$. Thus, the algorithm is a 2-approximation.

Recap of algorithm for 1 machine, release dates

1. Solve the preemptive schedule to obtain preemptive completion times C_j^P .
2. Schedule the jobs in the order given by C_j^P, respecting release dates.

General Framework

1. Solve a relaxation of the given problem in order to obtain an ordering on the jobs.
2. Schedule the jobs according to the ordering, respecting constraints.

Alpha points

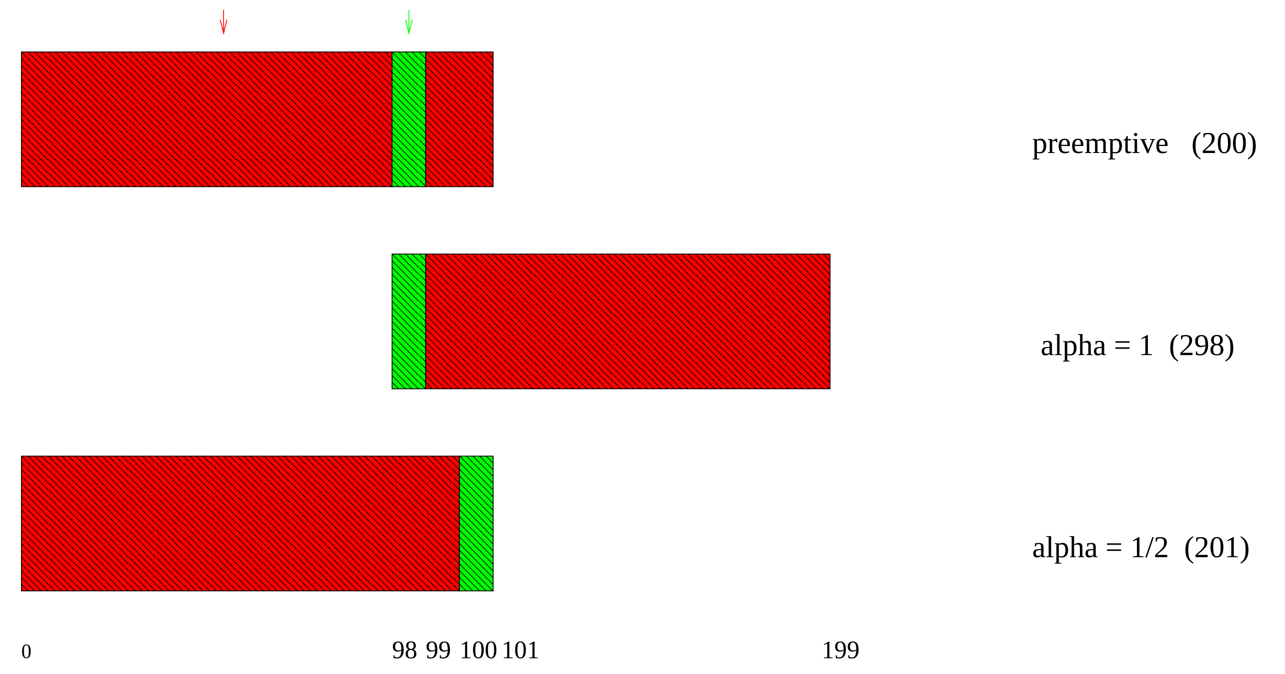
([Hall et.al. '96, Phillips, S, Wein '95, Goemans '97, Chekuri et.al. '97])

Alpha point: Let C_j^α be the earliest time at which αp_j of job j has completed.

Sample input:

j	r_j	p_j
1	0	100
2	98	1

Schedules:



Intuition: α-points can avoid "bad" case.

Scheduling by α-points

Schedule-by-α

1. Choose $\alpha \in [0,1]$
2. Solve the preemptive schedule to obtain preemptive completion times C_j^α .
3. Schedule the jobs in the order given by $C_j^\alpha,$ respecting release dates.

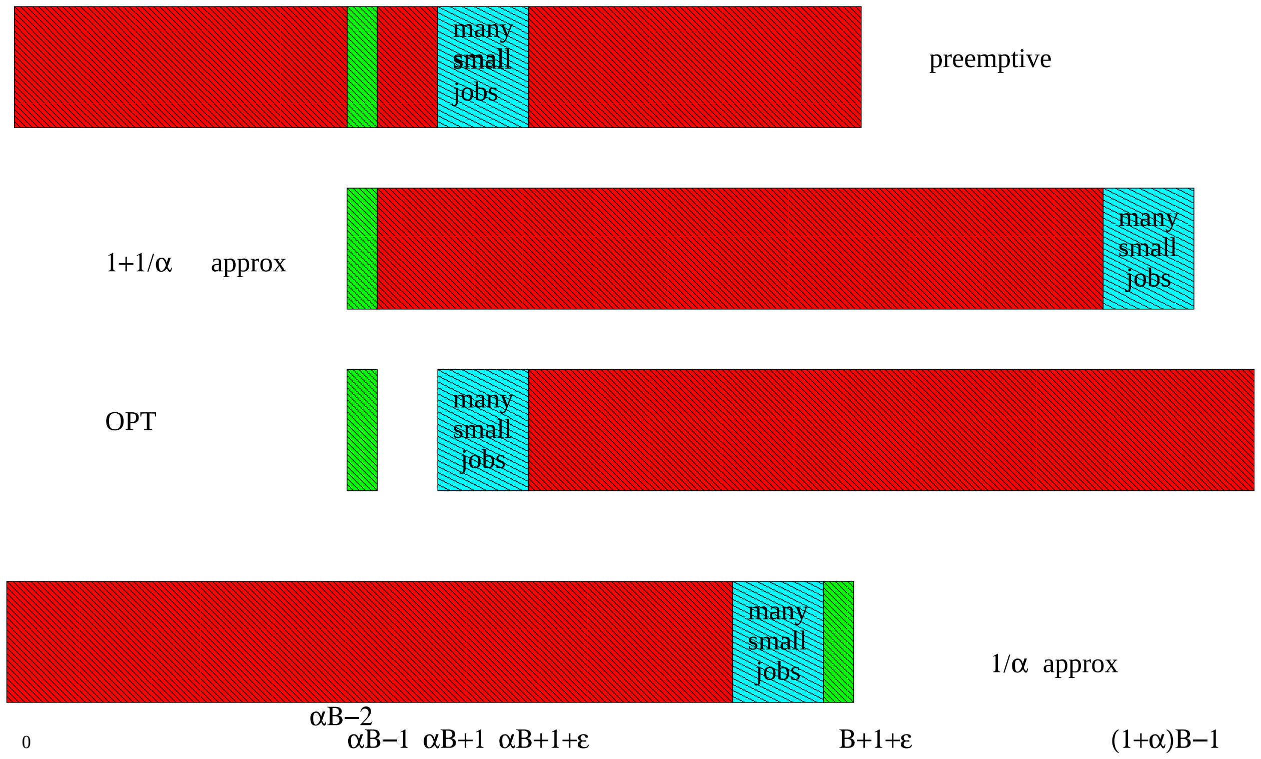
Theorem: Schedule-by-α is a $\left(1+\frac{1}{\alpha}\right)$-approximation algorithm.
This is no better than 2!

Insight for improvement

For any α, there is an input that can be as bad as $1 + \frac{1}{\alpha}$

BUT

for any input, most α's yield significantly better schedules.



Randomize to avoid worst case

Schedule-by-random-α

1. Choose $\alpha \in [0,1]$ according to some probability distribution.
2. Solve the preemptive schedule to obtain preemptive completion times C_j^α .
3. Schedule the jobs in the order given by C_j^α, respecting release dates.

Bounds

If we choose $f(\alpha) = \frac{e^{\alpha}}{e-1}$, then Schedule-by-random-α is an $\frac{e}{e-1} \approx 1.58$ -approximation algorithm and it runs in $O(n \log n)$ time.

- Can derandomize algorithm by choosing all $n-1$ combinatorially distinct α-points.
  ($O(n^2 \log n)$ time.)
- This defines an algorithm (Best-α), which has the same bound.

Main Idea for Best-α

- **Solve a relaxed version of the original problem to get a schedule.**
- **Use α -points to generate many different job orderings.**
- **Take the best of these orderings.**

Another example – $1|r_j, prec|\Sigma C_j$

Use a linear programming relaxation[Hall et. al. '96, Dyer Wolsey '91]

Variables: $y_{jt} = 1$ **if job j completes at time t**

$$\min \sum_{j=1}^{n} \sum_{t=1}^{T} t y_{jt}$$

subject to

$$\begin{array}{lll}
\sum_{t=1}^{T} y_{jt} = 1 & j = 1 \ldots n & \textbf{jobs run} \\
y_{jt} = 0 & \textbf{if } t < r_j + p_j & \\
\sum_{j=1}^{n} \sum_{s=t}^{t+p_j-1} y_{js} \leq 1 & t = 1 \ldots T & \textbf{machine} \\
\sum_{s=1}^{t} y_{js} \geq \sum_{s=1}^{t+p_k} y_{ks} & \textbf{if } j \to k, t = 1 \ldots T - p_k & \textbf{prec}
\end{array}$$

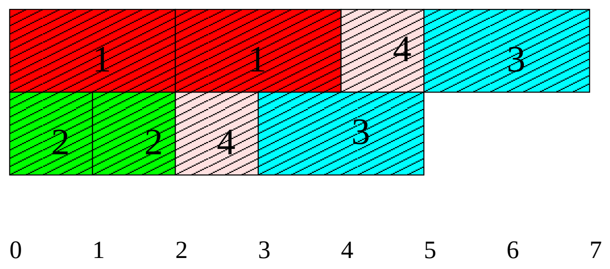
LP is lower bound, but how do we get ordering?

Example

j	r_j	p_j
1	**0**	**2**
2	**0**	**1**
3	**0**	**2**
4	**0**	**1**

$$\begin{tikzcd} 1 \arrow[d] & 2 \arrow[dl] \arrow[d] \\ 3 & 4 \end{tikzcd}$$

Solution to LP:



(1/2 point) 2 1 4 3

Variables:

$$y_{21} = y_{12} = y_{22} = y_{43} = y_{14} = y_{35} = y_{45} = y_{37} = \frac{1}{2}$$

Algorithm and Analysis ($\alpha = 1/2$)

General Framework

1. Solve the LP relaxation of the given problem to obtain $1/2$-points, the earliest point at which half of processing is done.
2. Schedule the jobs according to the ordering for the $1/2$-points, respecting release dates.

- release dates are obeyed
- precedence constraints are obeyed
- 5.33-approximation for $\alpha = 1/2$, 3-approximation for Best-α
- non-polynomial size can be handled

Algorithm and Analysis ($\alpha = 1/2$)

General Framework

1. Solve the LP relaxation of the given problem to obtain $1/2$-points, the earliest point at which half of processing is done.
2. Schedule the jobs according to the ordering for the $1/2$-points, respecting release dates.

- release dates are obeyed
- precedence constraints are obeyed
- 5.33-approximation for $\alpha = 1/2$, 3-approximation for Best-α
- non-polynomial size can be handled

A nice feature of α-point scheduling is that experimental results seem to validate theory.

Experimental Results

([Savelsberg, Uma, Wein, '98])

	arrival rate = 2								
(p_j, w_j)	$schedule - by - \bar{C}_j$			$schedule - by - fixed - \alpha$			$best - \alpha$		
	Mean	Std.Dev.	Max	Mean	Std.Dev.	Max	Mean	Std.Dev.	Max
(1,1)	1.360	0.186	1.781	1.271	0.105	1.480	1.114	0.049	1.199
(1,2)	1.351	0.190	1.842	1.250	0.111	1.455	1.101	0.044	1.179
(1,3)	1.400	0.196	1.833	1.262	0.119	1.549	1.147	0.094	1.407
(2,1)	1.307	0.119	1.530	1.210	0.076	1.359	1.086	0.021	1.140
(2,2)	1.247	0.111	1.515	1.205	0.082	1.352	1.084	0.035	1.152
(2,3)	1.262	0.120	1.518	1.187	0.068	1.289	1.100	0.070	1.276
(3,1)	1.341	0.207	1.810	1.246	0.141	1.622	1.129	0.089	1.367
(3,2)	1.305	0.154	1.645	1.238	0.120	1.548	1.095	0.057	1.258
(3,3)	1.287	0.147	1.612	1.173	0.096	1.303	1.079	0.053	1.198

Table 1: 50 jobs $\sum w_j F_j$. Performance of C_j-based heuristics, compared to C_j-based lower bound.

(n,a)	Schedule-by-$\bar{C}_j$			Schedule-by-Fixed-α			Best-α		
	Mean	Std.Dev.	Max	Mean	Std.Dev.	Max	Mean	Std.Dev.	Max
(50,2)	1.271	0.144	1.729	1.184	0.092	1.495	1.066	0.047	1.282
(50,5)	1.107	0.045	1.245	1.073	0.032	1.197	1.018	0.013	1.071
(100,2)	1.268	0.125	1.786	1.190	0.096	1.686	1.071	0.043	1.271
(100,5)	1.079	0.030	1.152	1.056	0.018	1.104	1.014	0.010	1.055

Table 2: Performance of algorithms applied to solution of C_j-relaxation for $\sum w_j F_j$. We report on ratio of algorithm performance to x_{jt}-relaxation lower bound.

Back to our problem

How can α-points help?

Back to our problem

How can α-points help?

Answer: **They can give us a way to generate, from one LP solution, many feasible schedules.**

Solving the LP is the computational bottleneck, so this is potentially a big win.

Recall our algorithmic framework

1. **Solve the linear programming relaxation of TILP to obtain x^*_{jt} .**
2. **Call the** Best-α **or** Fixed-α **subroutine to convert the TILP solution x^*_{jt} to a set, Π , of priority orderings for the jobs.**
3. **For each ordering $\pi \in \Pi$, call the** Comb **or** Greedy **subroutine to determine a schedule $\omega(\pi)$ for the tasks**
4. **Use local search to try to improve the objective for the schedules**

Example of LP relaxation

Recall example:

Task (type)	r	d	p	**crew size**	**cert. req.**	**facility req.**
1	0	4	2	2	A or B	1
2	0	4	4	2	B	1
3	0	9	3	3	A	2
4	0	14	6	2	B	1

Technician	**Certification**
1	**A**
2	**B**
3	**A and B**

Precedence:

$$1, 2 \prec 3 \prec 4$$

Example of LP relaxation

Single weapon, 4 task, 2 certification, 3 worker, 2 facility problem.

$$
\begin{array}{lll}
\min a_A + a_B + b_1 + b_2 & & \\
\hline
\Sigma_{t=0}^{8} x_{j,t} = 1, & & j = 1, \ldots, 4 \\
\hline
\Sigma_{t\leq\tau} x_{3,t} \leq \Sigma_{t\leq\tau-3} x_{1,t} & & \tau = 0, \ldots, 8 \\
\Sigma_{t\leq\tau} x_{3,t} \leq \Sigma_{t\leq\tau-2} x_{2,t} & & \tau = 0, \ldots, 8 \\
\Sigma_{t\leq\tau} x_{4,t} \leq \Sigma_{t\leq\tau-3} x_{3,t} & & \tau = 0, \ldots, 8 \\
\hline
\Sigma_{t=\tau-1,\tau} 2x_{1,t} + & & \\
\Sigma_{t=\tau-2,\tau-1,\tau} 3x_{3,t} \leq \Sigma_{w=1,3} y_{w,\tau} + a_{A,\tau} & & \forall \tau, c \\
\Sigma_{t=\tau-1,\tau} 2x_{1,t} + \Sigma_{t=\tau-3,\ldots,\tau} 2x_{3,t} & & \\
+ \Sigma_{t=\tau-5,\ldots,\tau} 2x_{3,t} \leq \Sigma_{w=2,3} y_{w,\tau} + a_{B,\tau} & & \forall \tau, c \\
\hline
y_{3,A,t} + y_{3,B,t} \leq 1 & & t = 0, \ldots, 8 \\
\hline
\Sigma_{j=1,2,4} x_{j,\tau} \leq \theta_{0,\tau} + b_{1,\tau}, & & \tau = 0, \ldots, 8 \\
\Sigma_{t=(\tau-3+1)^+}^{\tau} x_3 \leq \theta_{2,\tau} + b_{2,\tau}, & & \tau = 0, \ldots, 8 \\
\hline
x_{1,t} = 0 & & t = 3, \ldots, 8 \\
x_{2,t} = 0 & & t = 1, \ldots, 8 \\
x_{3,t} = 0 & & t = 7, 8 \\
\hline
\Sigma_\tau a_{A,\tau} = a_A & \Sigma_\tau a_{B,\tau} = a_B & \\
\Sigma_\tau b_{1,\tau} = b_1 & \Sigma_\tau b_{2,\tau} = b_2 & \\
\hline
0 \leq x_{j,t}, y_{1,c,t} \leq 1 & & j, c, t, f = \eta_j \\
\hline
a_c, a_{c,t}, b_f, b_{f,t} \geq 0 & &
\end{array}
$$

α-points

α-point scheduling for TILP : Given a solution, x, to the time-indexed LP, and an $\alpha \in (0,1)$, let

$$t_j^\alpha = \min\{t : \sum_{s \le t} x_{j,s} \ge \alpha\}.$$

Schedule jobs in ascending order of t_j^α.

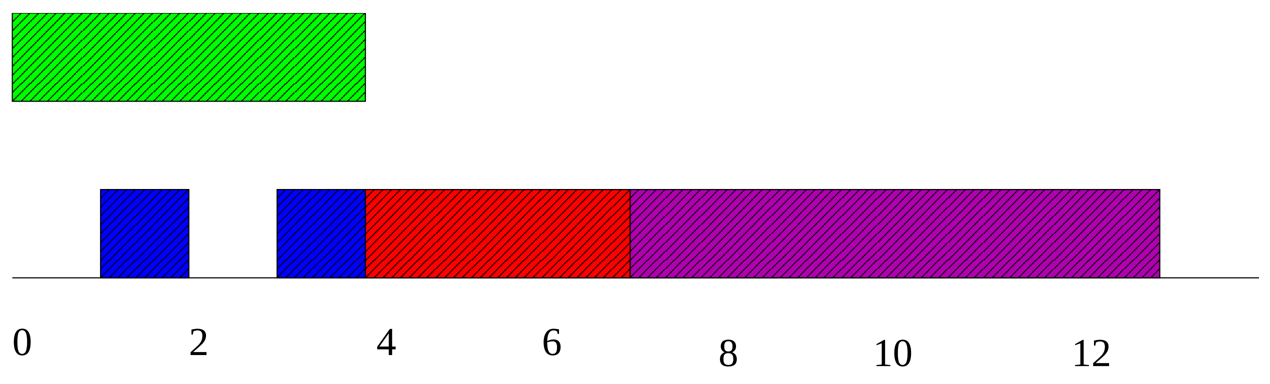
α-points, example

Solution (x only) to previous LP is:

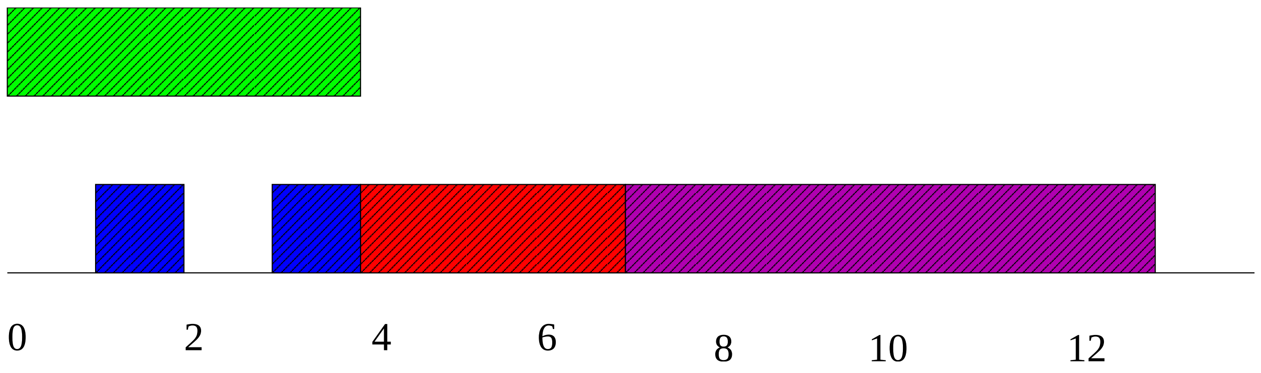
$$x_{1,1} = x_{1,3} = .5$$
$$x_{2,0} = 1$$
$$x_{3,4} = 1$$
$$x_{4,7} = 1$$

Recall processing times:

$$p_1 = 2,\ p_2 = 4,\ p_3 = 3,\ p_4 = 6$$



α-points, example



For $\alpha = .5$, the priority order is 1,2,3,4 (broke tie with index), as

$$\begin{aligned} t_1^{.5} &= 2 \\ t_2^{.5} &= 2 \\ t_3^{.5} &= 5.5 \\ t_4^{.5} &= 10 \end{aligned}$$

For $\alpha = .25$, the priority order is 2,1,3,4 (broke tie with index), as

$$\begin{aligned} t_1^{.25} &= 1.5 \\ t_2^{.25} &= 1 \\ t_3^{.25} &= 4.75 \\ t_4^{.25} &= 8.5 \end{aligned}$$

α-points

We considered

- **one fixed** α
- **several random**
- **all** α **.** (Best-α)

α-points

We considered

- **one fixed** α
- **several random**
- **all** α **.** (Best-α)

Best choice is Best-α**. For convenience, we just tried about 50 evenly spaced** α **. In practice, this approximates** Best-α**.**

α-points

α **points only give an order of tasks, ignoring facilities and technicians.**

We need to do additional work to get a legal schedule.
We have two different heuristics:

- **COMB heuristic**
- **Greedy heuristic**

The COMB Heuristic

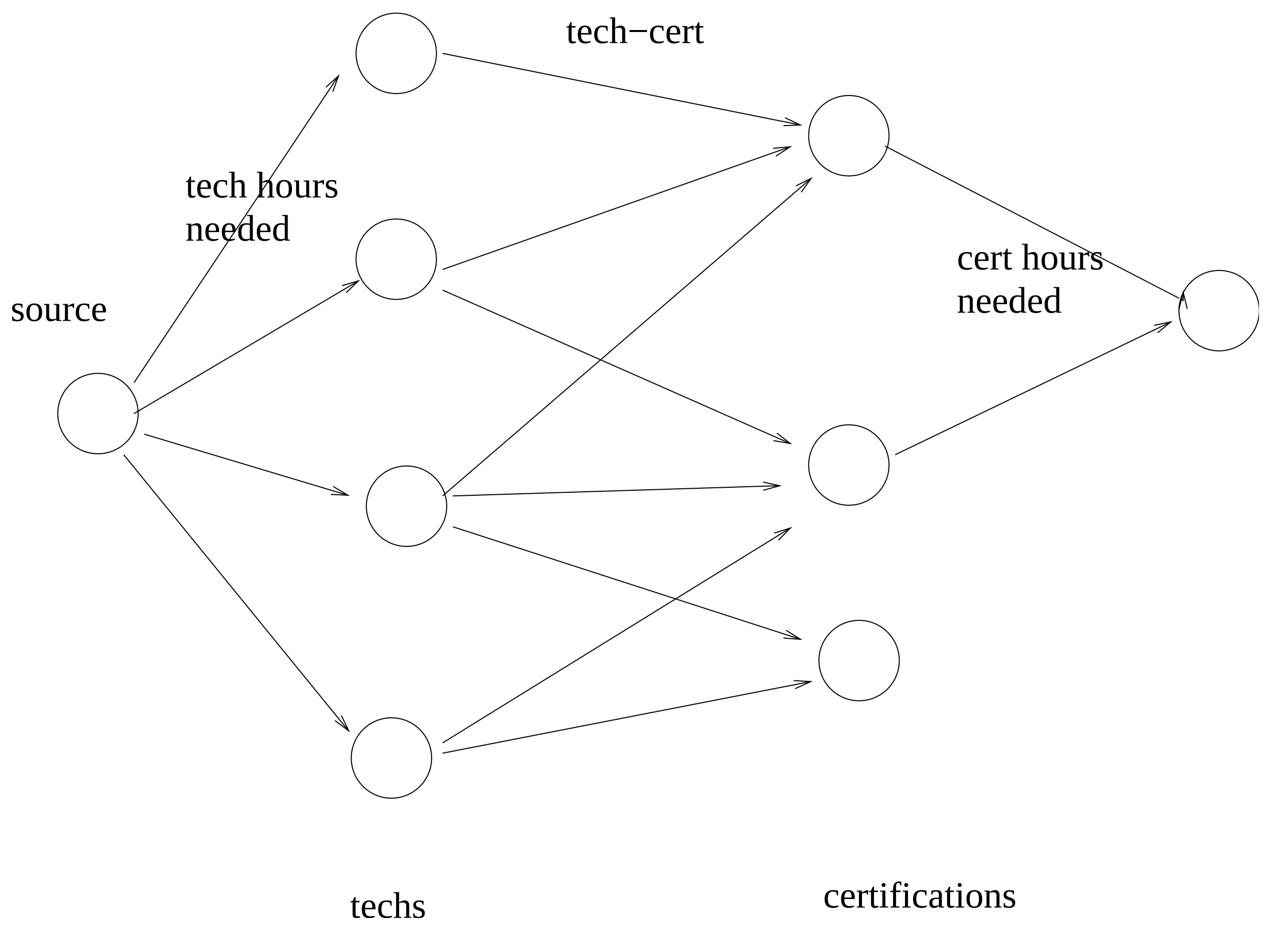
Overage: Given a schedule, with some jobs assigned and some technicians used, the overage of job j at time t, o_{jt} is the number of ghost technicians needed if job j starts at time j.

Step 1

- For each job j in order
  - For each time slot t in which job j could run
    - Compute o_{jt}
  - Let $o_{\min} = \min_t\{o_{jt}\}$.
  - Schedule job j at the earliest time with overage $o_{\min}$.
  - Update available technicians.

The COMB Heuristic

Step 2: **Use network flow to do the actual assignment.**



If you can saturate the edges into the sink, you have an assigment of techs to certs with predicted overage. If not, adjust ghost techs.

Step 3: **Assign facilities similarly.**

The Greedy Heuristic

Two choices:

- **Schedule technicians first, facilities second**
- **Schedule facilities first, technicians second**

The Greedy Heuristic

Two choices:

- **Schedule technicians first, facilities second**
- **Scehdule facilities first, technicians second**

Conclusion from experiments: **Scheduling technicians first performs better, so we explain that algorithm.**

Schedule Technicians First

For each job in priority order given by α-points

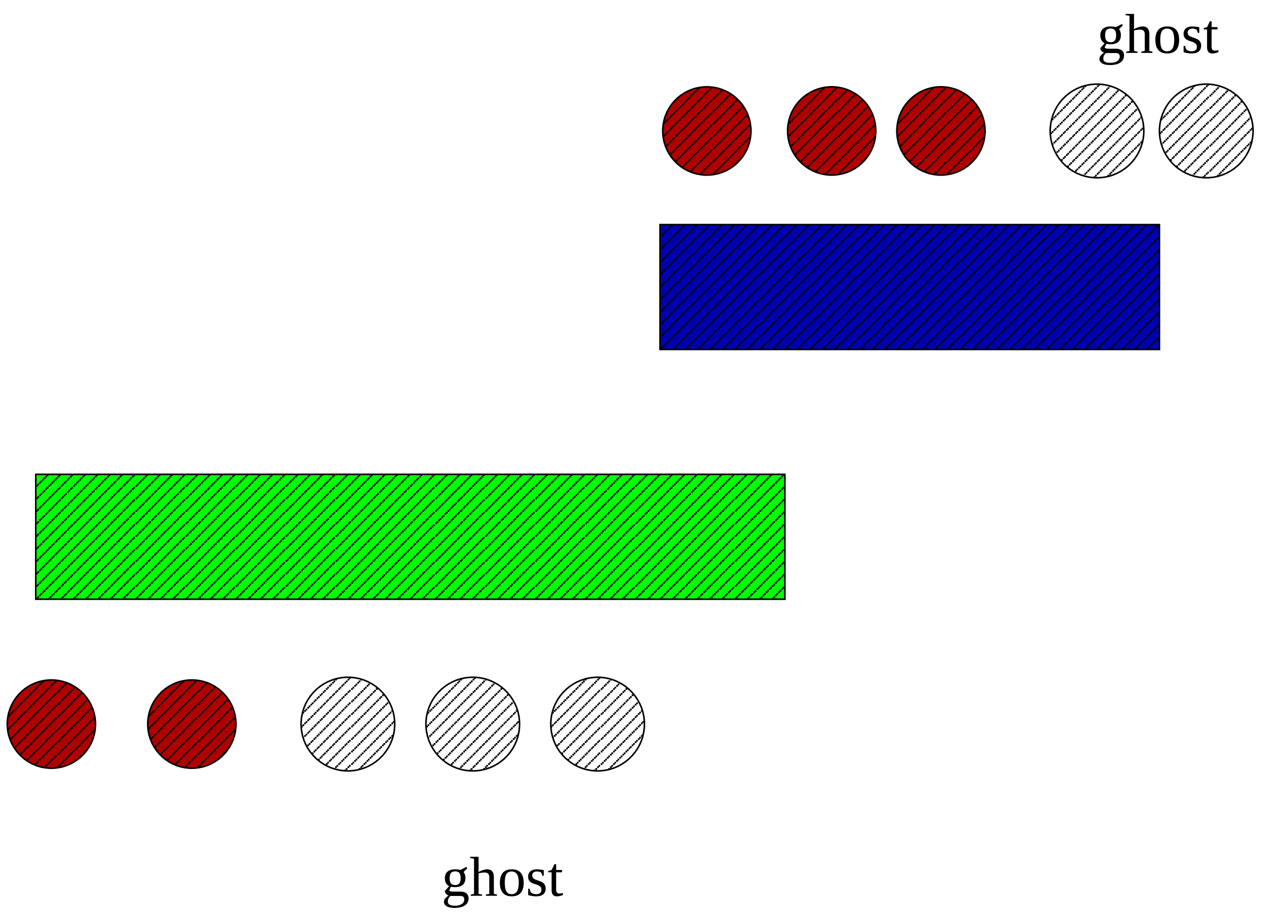
- Compute $r'_j = \min\{r_j, \max_{i:i\prec j}\{C_i\}\}$
- For each interval I_k of length p_j in the period $[r'_j, d_j]$
  - Compute the a_k number of available technicians in interval I_k .
- Let k' be the interval wih maximum a_k
- Schedule $\min\{a_k, s_j\}$ technicians, satisfy the rest with ghost technicians.

Assign Facilities Second is done similarly, in a greedy manner.

Local Search

We try to reallocate technicians in order to decrease ghost technician hours.

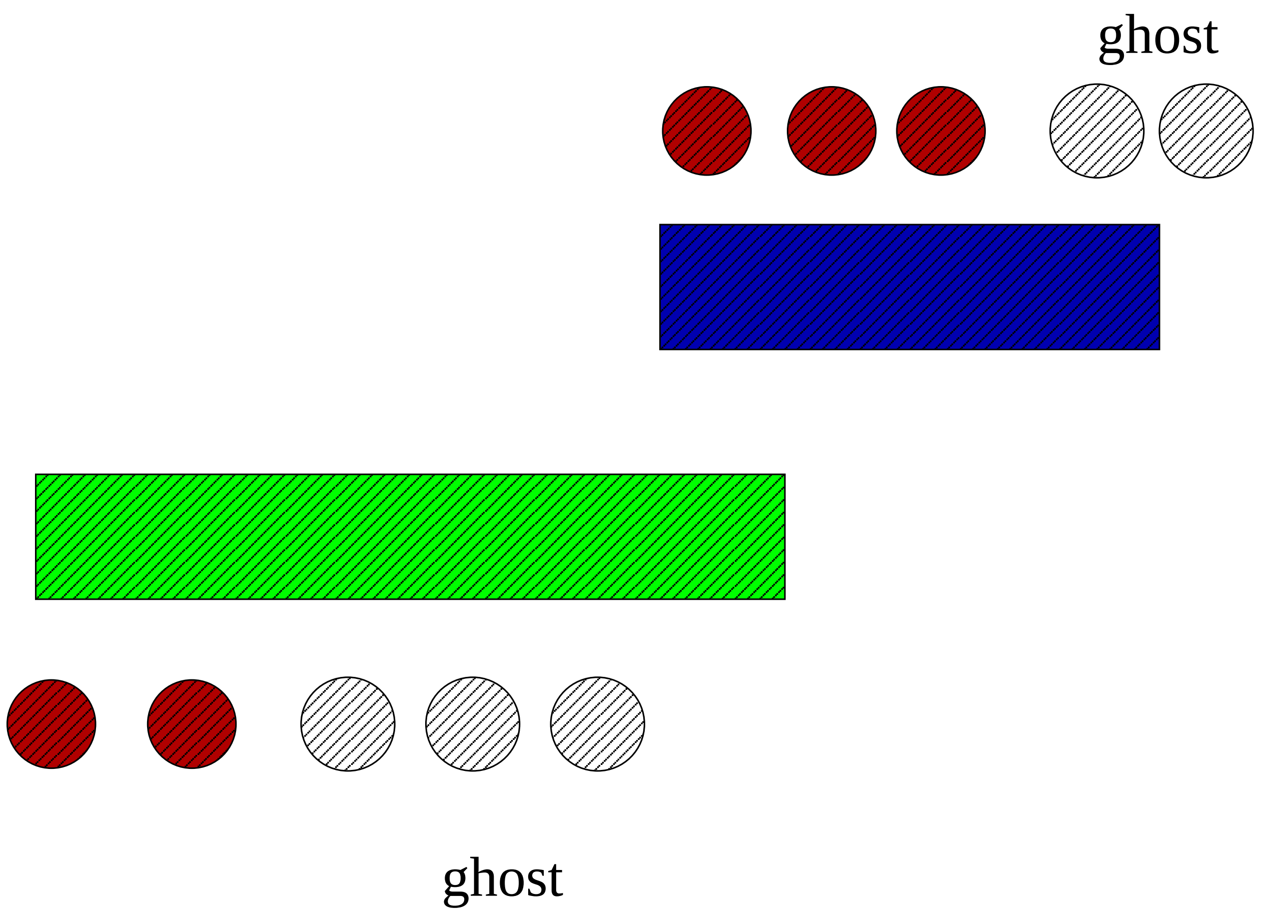
Consider



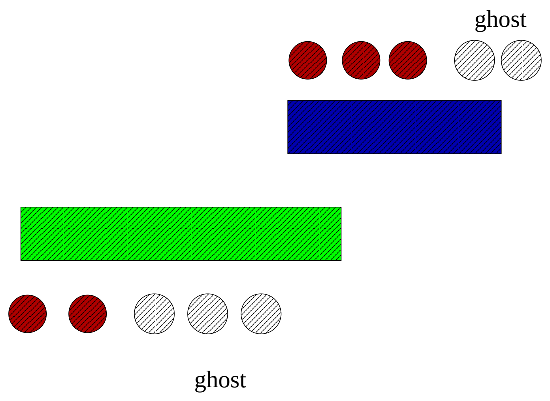
Local Search

We try to reallocate techinicans in order to decrease ghost technician hours.

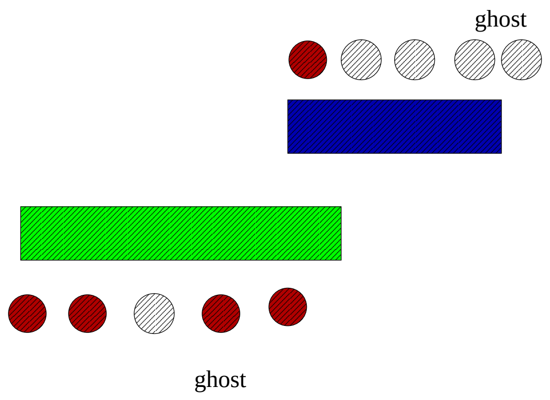
Consider



Local Search



If other constraints are satisfied, we can swap ghosts to decrease total ghost hours:



We repeat this. After swapping ghost locations, we may also be able to replace ghost technician hours with real technician hours.

Recall our algorithmic framework

1. Solve the linear programming relaxation of TILP to obtain x^*_{jt} .
2. Call the BEST-α or FIXED-α subroutine to convert the TILP solution x^*_{jt} to a set, Π , of priority orderings for the jobs.
3. For each ordering $\pi \in \Pi$, call the COMB or GREEDY subroutine to determine a schedule $\omega(\pi)$ for the tasks
4. Use local search to try to improve the objective for the schedules

Clear choices

- BEST-$\alpha$is a clear win over FIXED-α
- Scheduling techs before facilities is better.

To learn more, we need data

The Data

A security tale

- Those of us without appropriate clearance couldn’t get much real data.
- Those of us without appropriate clearance couldn’t be trusted with a generator written by those with appropriate clearance.

The Data

A security tale

- Those of us without appropriate clearance couldn’t get much real data.
- Those of us without appropriate clearance couldn’t be trusted with a generator written by those with appropriate clearance.

What we did

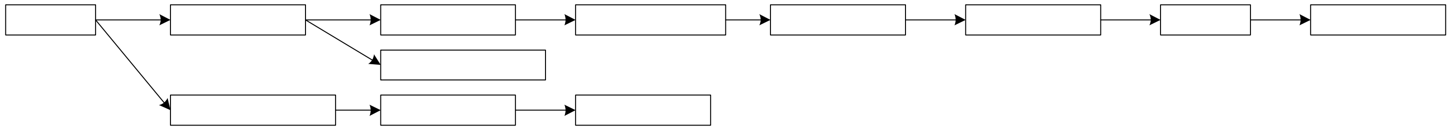
- Those of use without clearance asked many questions of those with clearance.
- We got answers to many questions, vague answers to other questions.
- Those without clearance wrote a generator that those with clearance said was fairly realistic.
- Those with clearance could then run their algorithms on the data generated by those without clearance.

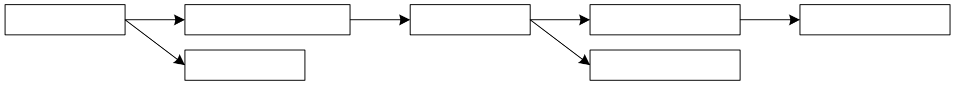
What does a job look like?

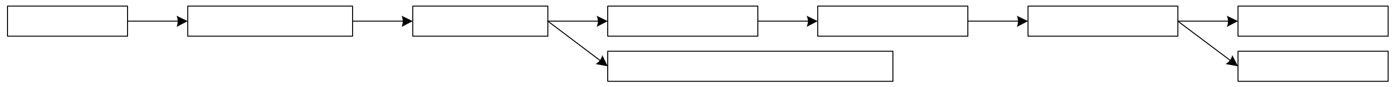
We generated a number of templates that captured typical jobs:

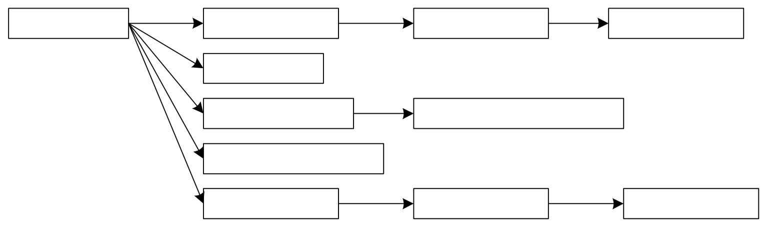
- chain-like
- out-tree
- many tasks have only one successor

Some typical job templates









What does a job look like

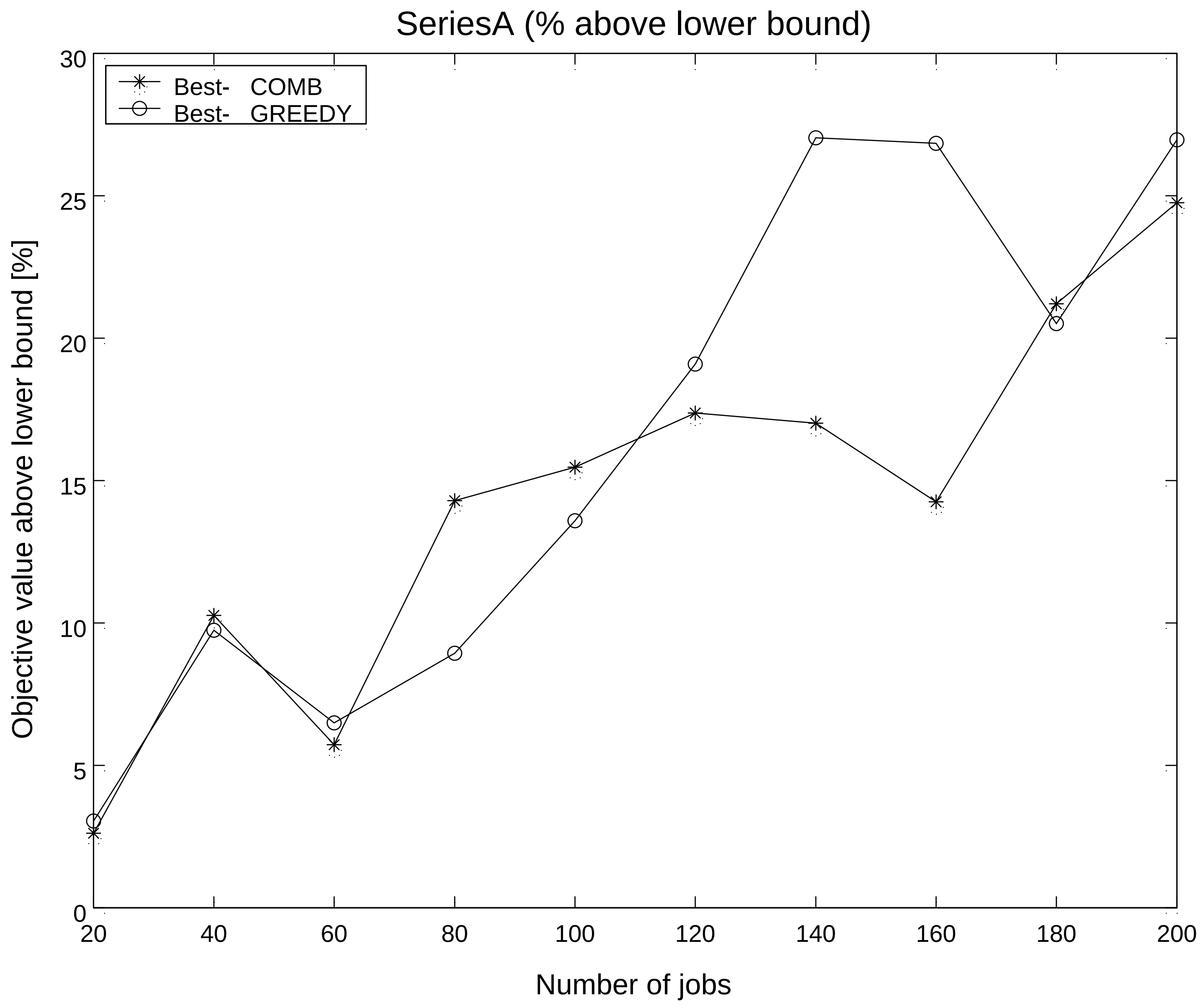
- Our data is probably less chain-like than the real data, but this only challenges our algorithms.
- We used a six month time horizon.
- We generated job lengths, certifications, facilities types randomly from appropriate distributions.

3 representative data series

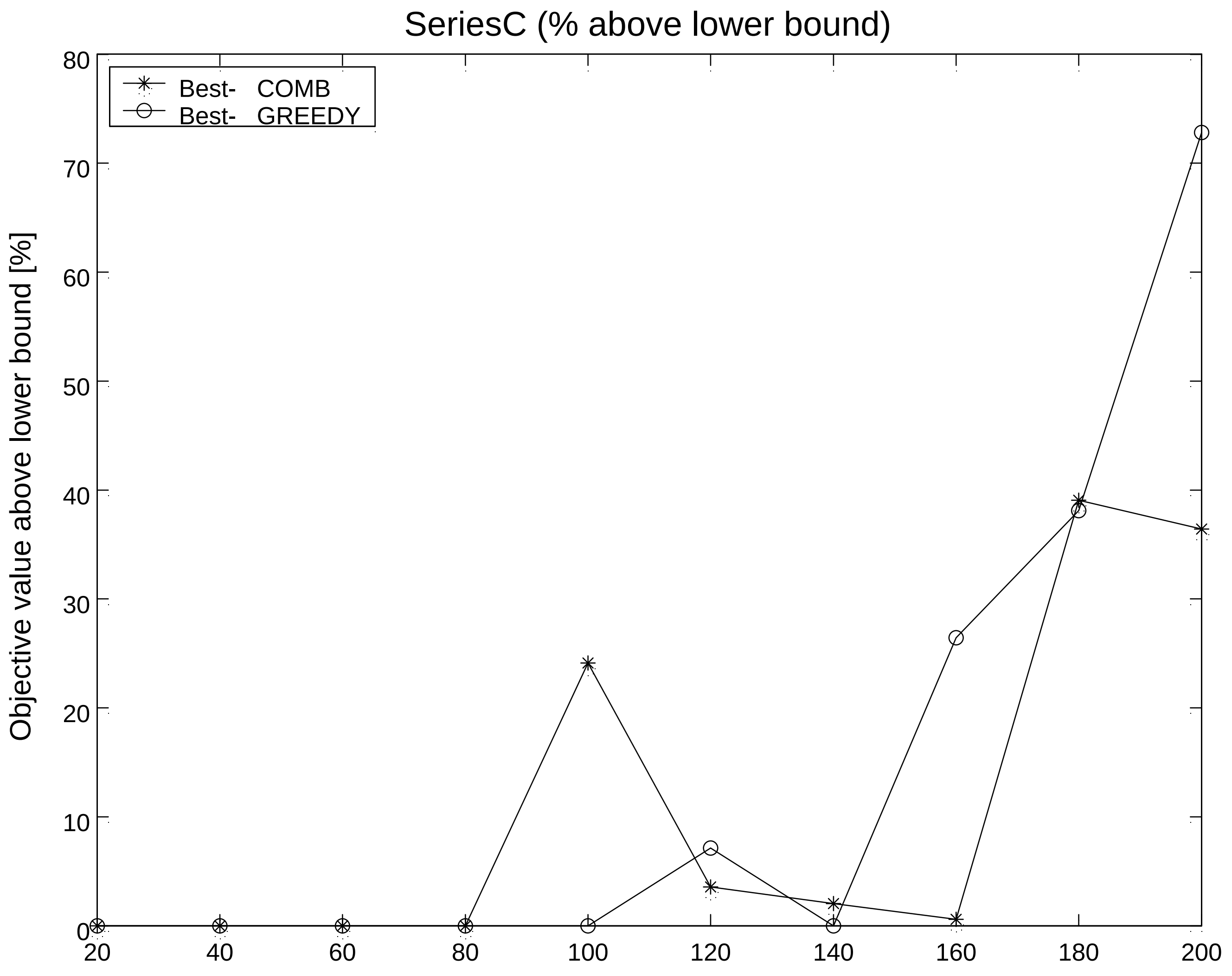
1. Series A measures effect of adding tasks, while technicians and facilities are held fixed.
2. Series B measures effect of adding tasks, while adding technicians and facilities.
3. Series C measured effect of being limited in facilities, rather than technicians.

Series (jobs)	Tasks	Crew	Certs	Techs	C.per.T.	Fac.Types	Fac.Hours
SeriesA - (20)	212	1-6	100	300	3-5	30	100
SeriesA - (100)	1060	1-6	100	300	3-5	30	100
SeriesA - (200)	2120	1-6	100	300	3-5	30	100
SeriesB - (20)	212	1-6	100	150	3-5	6	10
SeriesB - (100)	1060	1-6	100	750	3-5	6	50
SeriesB - (200)	2120	1-6	100	1500	3-5	6	100
SeriesC - (20)	212	1-2	100	300	3-5	30	100
SeriesC - (100)	1060	1-2	100	300	3-5	30	100
SeriesC - (200)	2120	1-2	100	300	3-5	30	100

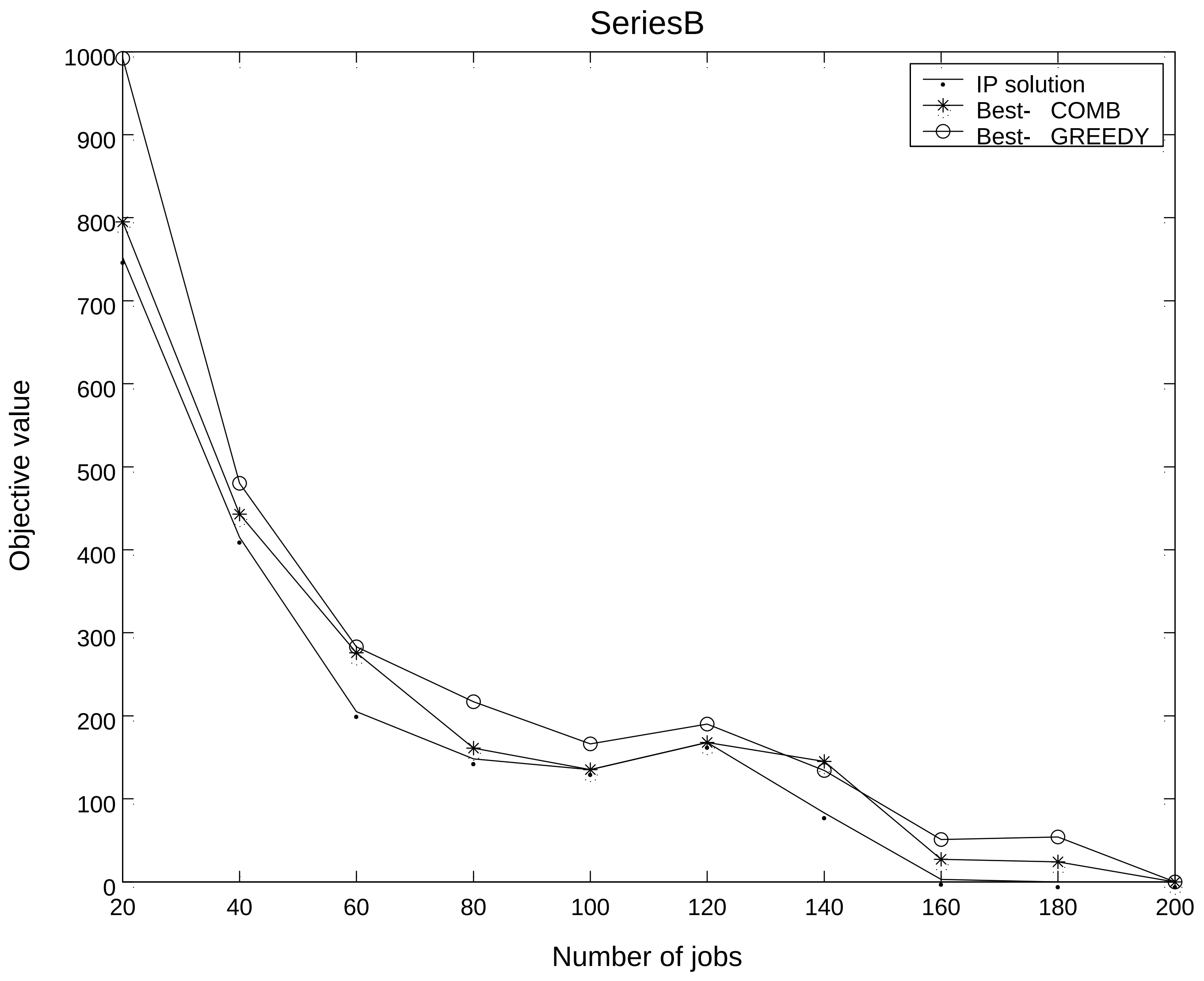
The effect of more jobs with the same resources.


SeriesA (% above lower bound)
Best- COMB
Best- GREEDY
Objective value above lower bound [%]
Number of jobs
0
5
10
15
20
25
30
20
40
60
80
100
120
140
160
180
200

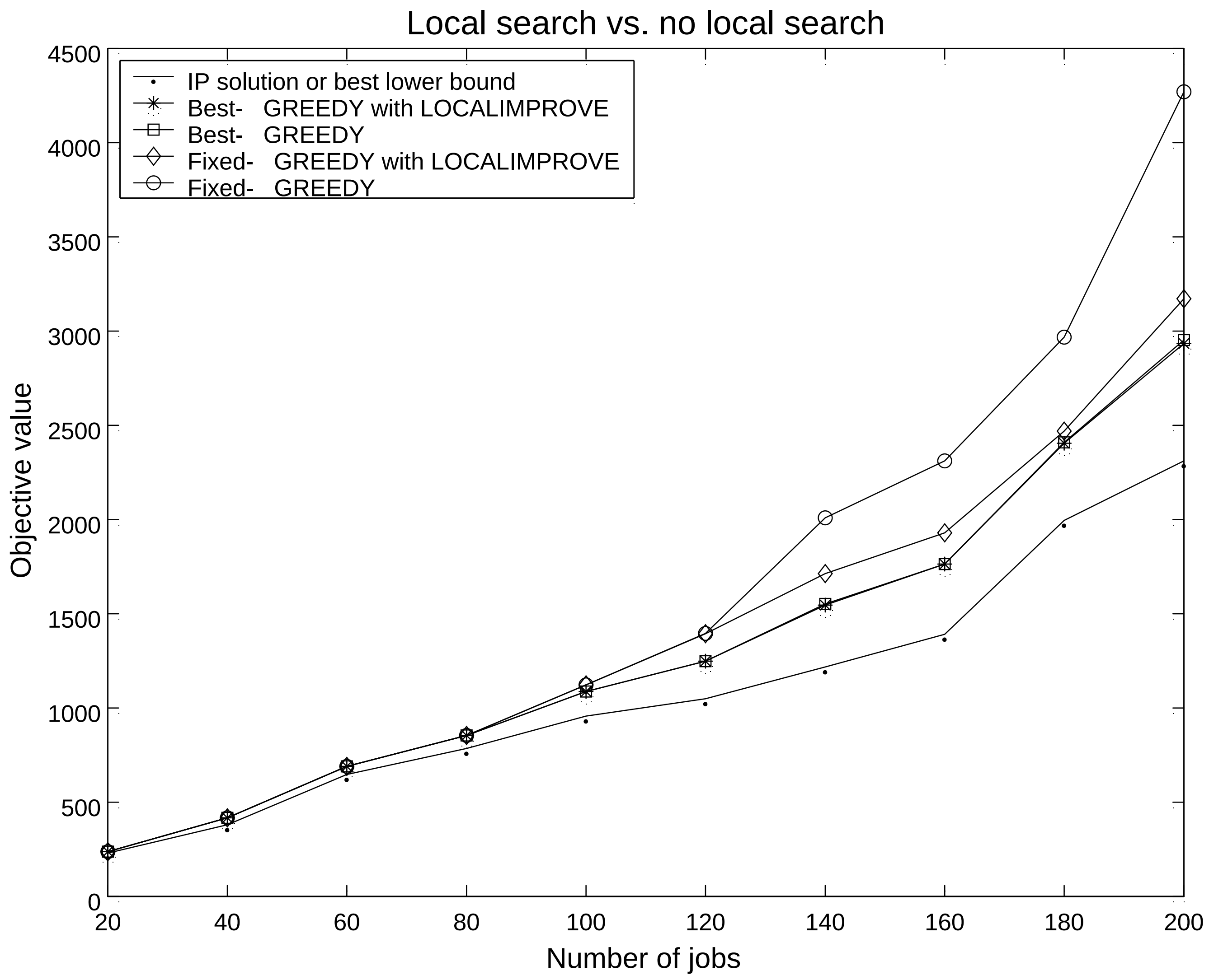
The effect of being limited by facilities


SeriesC (% above lower bound)
Best- COMB
Best- GREEDY
Objective value above lower bound [%]
Number of jobs
0
10
20
30
40
50
60
70
80
20
40
60
80
100
120
140
160
180
200

Problems of increasing size


SeriesB
Objective value
Number of jobs
IP solution
Best- COMB
Best- GREEDY
0
100
200
300
400
500
600
700
800
900
1000
20
40
60
80
100
120
140
160
180
200

Local search


Local search vs. no local search
IP solution or best lower bound
Best- GREEDY with LOCALIMPROVE
Best- GREEDY
Fixed- GREEDY with LOCALIMPROVE
Fixed- GREEDY
Objective value
Number of jobs
0
500
1000
1500
2000
2500
3000
3500
4000
4500
20
40
60
80
100
120
140
160
180
200

Conclusions

- Best-α is better than Fixed-α. (By a factor of up to 4)
- Comb performs better than Greedy.
  - Both algorithms found the optimal solution in 5 instances
  - Comb found a better solution than Greedy in 17 instances.
  - Greedy found a better solution than Comb in 8 instances.
  - When Greedyis better, the difference is usually small ($< 2\%$ in 5 of 8).
  - When Comb is better the difference is larger ($> 10\%$ in 9 of 17).
- LocalImprove helps Fixed-α but it does not help Best-α.
- A small decrease in accuracy gives a huge decrease in running time. For the most challenging series A problems (120 jobs and above), CPLEX and PICO both couldn't solve in a specified time limit (¿ 48 hours). The LP solutions and heuristics run in minutes However, the heuristics are usually within 10% of the optimal value.
- The best heuristic is Best-α Comb. Overall, our results indicate that Best-α and Comb is the best combination. LocalImprove does not yield any significant improvements on the Best-α and its running time hardly justifies its inclusion. Hence the best combination in terms of speed and results is Best-α Comb.

Meta-conclusions

- Sophisticated algorithmic ideas, designed for combinatorial scheduling problems and designed with approximation algorithms in mind, can be used in heuristics to obtain good solutions to a difficult, “real-life” project scheduling problem.
- α-points can yield reasonable approximations for a difficult project scheduling problem. α-points provide an inexpensive way to generate many feasible schedules from one linear program. We have also demonstrated that α-points can be used in situations with hard deadlines.

Future work

- Run on larger data.
- Better exact solutions. The ideas used to generate approximate solutions need to be incorporated completely in the branch-and-bound.
- More realistic objective.
- Start-based vs. completion based alpha-points.
- Other uses of alpha points to generate multiple solutions quickly.