Tightening ‘Big M’ Constraints

Lou Hafer

DIMACS Workshop on COIN-OR

July 17 – 20, 2006

Various portions joint with Mikhail Bilenky, Alexander Kononov, Cheryl Petreman, and Ken Collins.

Time-Expanded Models

Suppose we have jobs $j = 1, \ldots, J$ with duration d_j that use the same machine. The machine can handle one job at a time.

A time-expanded model breaks time into discrete steps $t = 1, \ldots, T$. A binary variable x_{jt} encodes job j starting at time t.

To express that at most one job can use the machine in a given time step τ, I can write

$$\sum_{j=1}^{J} \sum_{t=\tau-d_j+1}^{\tau} x_{jt} = 1$$

for every time slot t.

What’s my motivation?

A long time ago, when I was a bright-eyed Ph.D. student, I was working on digital hardware design automation. Essentially, it’s a resource-constrained scheduling problem: I have a bunch of tasks with precedence, and a box of parts, and I want to assign tasks to parts so as to minimise an objective that balances makespan and cost of parts.

When the execution time of a task is known in advance, you can write a time-expanded model. A constraint that says “one task at a time” looks like this. I have to look over a range of time indices, in order to see if a job started earlier is still active. The duration of the activity has to be a known constant to write this constraint.

Continuous Time Models

Suppose I use continuous variables s_i, f_i to encode the start and finish times of jobs. To express that at most one job can use the machine at a given time, I can write

$$\begin{aligned} s_q - f_p + \alpha_{pq} M &\geq 0 \\ s_p - f_q + (1 - \alpha_{pq}) M &\geq 0 \end{aligned}$$

for all pairs p, q which could overlap.

My problem was that I didn’t know the duration in advance. One of the decisions in digital hardware design affects the duration of a task.

So I tried using continuous variables, and ended up writing ‘big M’ constraints. M has to be big enough to satisfy the ‘wrong way’ constraint, once I’ve picked an order for p and q.

In my defense, I’d only just learned about MIP, and had no idea what a bad model this was.

After a while, I learned that researchers using the time-expanded model for digital hardware synthesis were getting much better computational results. And I found polyhedral theory.

Ever since, I’ve been interested in answering the question “What is it about the structure of the continuous time polyhedron that makes this model behave so badly, while the time-expanded model behaves so well?” Finally, I can offer a partial answer.

A Simple Example

A scheduling problem: Two activities, with no precedence, constant duration, upper and lower bounds on start times.

The constraint system is:

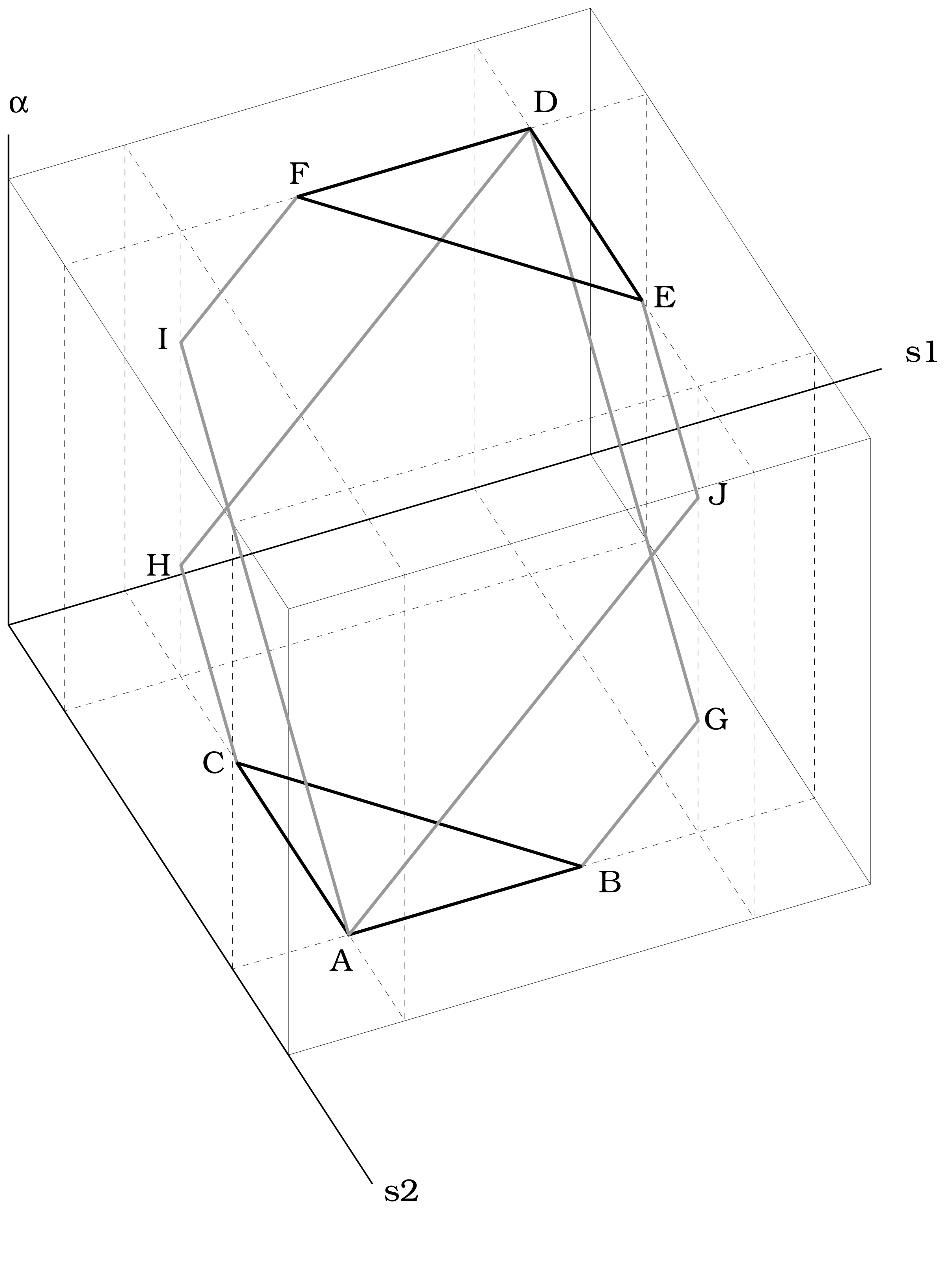
$$s_2 - (s_1 + d_1) + \alpha((\hat{s}_1 + d_1) - \check{s}_2) \geq 0$$
$$s_1 - (s_2 + d_2) + (1 - \alpha)((\hat{s}_2 + d_2) - \check{s}_1) \geq 0$$

$$\check{s}_1 \leq s_1 \leq \hat{s}_1$$
$$\check{s}_2 \leq s_2 \leq \hat{s}_2$$
$$0 \leq \alpha \leq 1$$

Here's a simple example. I've reverted to constant duration so I can stay in three dimensions.

The first constraint says that if the execution order is $x_1 \rightarrow x_2$ ($\alpha = 0$), then the start time of x_2 (s_2) must come after the finish time of x_1 ($s_1 + d_1$).

But what about when x_2 goes first? Then $\alpha = 1$, and the value $(\hat{s}_1 + d_1) - \check{s}_2$ is always larger than $s_2 - (s_1 + d_1)$.

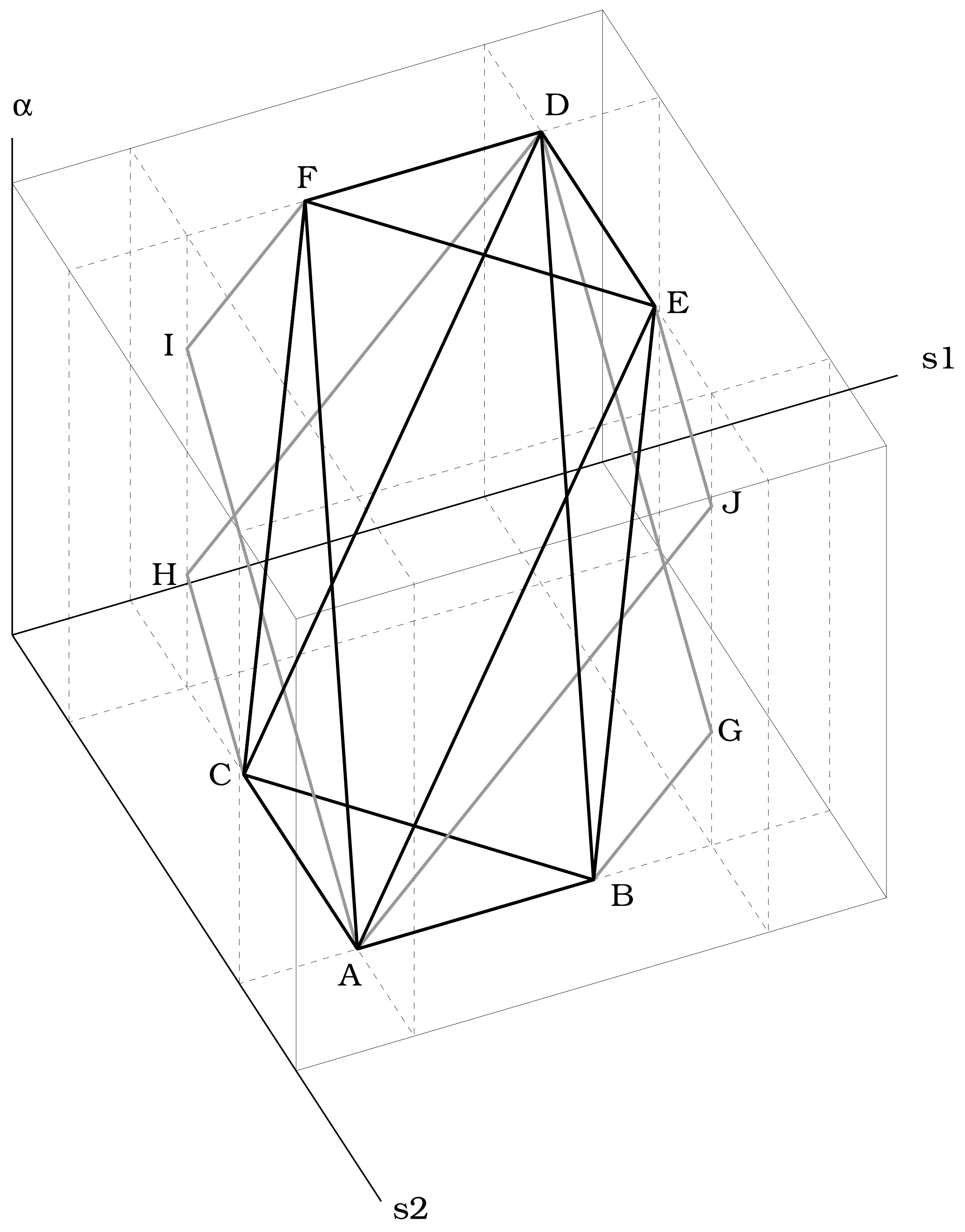

α
D
F
E
I
s1
J
H
G
C
B
A
s2

Here's the polytope. On the bottom, s_1 goes first. On the top, s_2 goes first. The ramps represent intermediate values of α. The dashed lines are the upper and lower bounds on the start times.

The ramp BCD is $s_2 - (s_1 + d_1) + \alpha((\hat{s}_1 + d_1) - \check{s}_2) \geq 0$.

The ramp EFA is $s_1 - (s_2 + d_2) + (1 - \alpha)((\hat{s}_2 + d_2) - \check{s}_1) \geq 0$.

We need to cut off the non-integral points, leaving only the integral points. By inspection, we can see the cutting planes which are necessary. ABE and BDE together will cut off G and J, for example.


α
D
F
E
I
s1
J
H
G
C
B
A
s2

The cutting planes are shown with dark lines. In effect, they will tighten the bound on a variable as we move from one feasible region to the other.

Non-Constant Duration

Given activities x_1 and x_2, to be serialised in some order.

Minimum durations d_1 and d_2.

Continuous variables s_1, s_2, for activity start times and f_1, f_2, for activity finish times, each with upper and lower bounds.

The minimum durations, and the upper and lower bounds on start and finish times, are assumed to be known constants.

The constraint system is

$$s_2 \geq f_1 \vee s_1 \geq f_2$$

$$f_1 \geq s_1 + d_1$$
$$f_2 \geq s_2 + d_2$$

$$\check{s}_i \leq s_i \leq \hat{s}_i$$
$$\check{f}_i \leq f_i \leq \hat{f}_i$$

Now, let's look at the case where we don't know execution time in advance.

Here's the constraint system: Two activities, x_1 and x_2, with minimum durations d_i. For each activity, the start time s_i and finish time f_i is represented as a continuous variable, with upper and lower bounds. The minimum duration and bounds on the start and finish times are assumed to be known *a priori*. For brevity, I'll refer to the 10 values as the parameters of the polytope.

We want to serialise the activities, in either order, and this gives the disjunctive constraint $s_2 \geq f_1 \vee s_1 \geq f_2$.

Linear Constraints

Rewritten in 'big M' form, the constraints are

$$s_2 - f_1 + (1-\alpha)(\hat{f}_1 - \check{s}_2) \geq 0$$
$$s_1 - f_2 + \alpha(\hat{f}_2 - \check{s}_1) \geq 0$$

$$f_1 \geq s_1 + d_1$$
$$f_2 \geq s_2 + d_2$$

$$\check{s}_i \leq s_i \leq \hat{s}_i$$
$$\check{f}_i \leq f_i \leq \hat{f}_i$$

One common transformation for linearising disjunctions is the ‘big M’ form. Here, if the variable $\alpha = 1$, activity x_1 should execute first; if $\alpha = 0$, activity x_2 should execute first. When α is equal to 0 or 1, one of the ordering constraints is enforced and the other is trivially satisfied.

As mentioned a moment ago, it’s well known that this model has lousy computational behaviour.

One oft-stated reason relates to the value of M. Many people have observed that it’s critically important that M be as small as possible — here we can see that it’s explicitly related to the upper and lower bounds on the variables.

The other reason is that the constraints shown here include only four of the 20 possible facets of this polytope.

Constraints for a Fixed Execution Order

Suppose x_1 precedes x_2. The fixed order polytope P^1 is described by this system of 11 constraints:

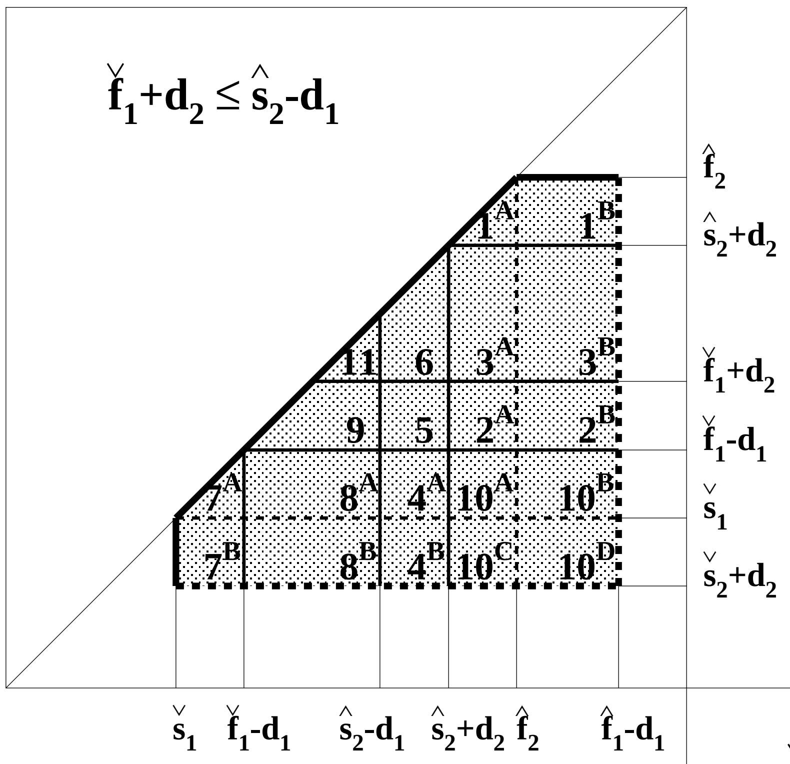
$$\begin{aligned}
s_2 - f_1 &\geq 0 \\
f_1 - s_1 &\geq d_1 \\
f_2 - s_2 &\geq d_2 \\
s_1 &\geq \check{s}_1 \\
-s_1 &\geq -\min\{\hat{s}_1, \hat{s}_2 - d_1\} \\
f_1 &\geq \check{f}_1 \\
-f_1 &\geq -\min\{\hat{f}_1, \hat{s}_2\} \\
s_2 &\geq \max\{\check{s}_2, \check{f}_1\} \\
-s_2 &\geq -\hat{s}_2 \\
f_2 &\geq \max\{\check{f}_2, \check{f}_1 + d_2\} \\
-f_2 &\geq -\hat{f}_2
\end{aligned}$$

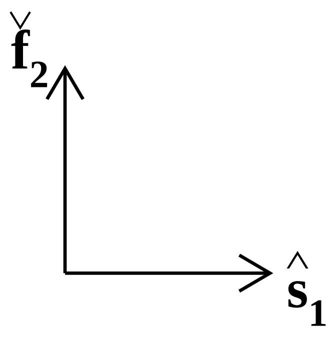
The lower and upper bound constraints make use of min$\{\bullet\}$ and max$\{\bullet\}$ because the values of the parameters $\check{s}_i$, $\hat{s}_i$, $\check{f}_i$, $\hat{f}_i$, and d_i determine the tighter bound. In effect, each max or min represents two constraints.

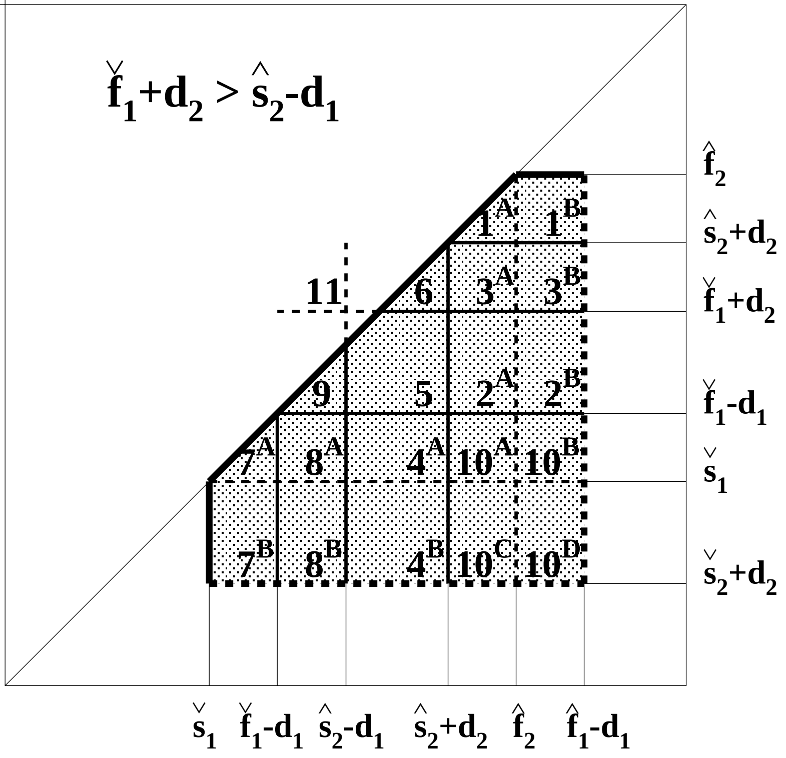
All constraints are facets of P^1 for some combination of values of the upper and lower bounds and minimum durations.

We can construct a similar set of constraints for the P^0 polytope where x_2 precedes x_1.

Regions of the parameter space for P^1


$\check{f}_1+d_2 \le \hat{s}_2-d_1$
$\hat{f}_2$
$\hat{s}_2+d_2$
$\check{f}_1+d_2$
$\check{f}_1-d_1$
$\check{s}_1$
$\check{s}_2+d_2$
1^A 1^B
11 6 3^A 3^B
9 5 2^A 2^B
7^A 8^A 4^A 10^A 10^B
7^B 8^B 4^B 10^C 10^D
$\check{s}_1$ $\check{f}_1-d_1$ $\hat{s}_2-d_1$ $\hat{s}_2+d_2$ $\hat{f}_2$ $\hat{f}_1-d_1$




$\check{f}_1+d_2 > \hat{s}_2-d_1$
$\hat{f}_2$
$\hat{s}_2+d_2$
$\check{f}_1+d_2$
$\check{f}_1-d_1$
$\check{s}_1$
$\check{s}_2+d_2$
1^A 1^B
11 6 3^A 3^B
9 5 2^A 2^B
7^A 8^A 4^A 10^A 10^B
7^B 8^B 4^B 10^C 10^D
$\check{s}_1$ $\check{f}_1-d_1$ $\hat{s}_2-d_1$ $\hat{s}_2+d_2$ $\hat{f}_2$ $\hat{f}_1-d_1$

This figure illustrates the many different shapes the fixed order polytope can assume, plotted as a function of $\check{f}_2$ and $\hat{s}_1$. The parameter space can be divided into 20 regions, corresponding to 20 different sets of supporting constraints for the P^1 polytope

The facial structure of the polytope changes with the values of the minimum durations and upper and lower bounds on the variables.

The dividing lines show where there are changes in the supporting constraints. The max and min terms in the constraints define most of the boundaries. A few other necessary relationships between the parameters produce the remainder.

The P^0 polytope is similar, as you'd expect from the symmetry of the constraint system.

Incorporating Order of Execution

Add a dimension with a binary variable α.

Embed P^0 and P^1 at $\alpha = 0$ and $\alpha = 1$, respectively.

Form the convex hull

$$P = \mathrm{conv}(P^0 \cup P^1)$$

To generate a polytope that incorporates both execution orders, we embed the fixed order polytopes in a larger space, using a binary variable to specify the execution order.

Then it's a matter of lifting the facets and faces of the fixed order polytopes to find the facets of the full polytope.

Facets of the Scheduling Polytope

$$s_1 - \alpha \cdot \check{s}_1 - (1-\alpha) \cdot \max\{\check{s}_1, \check{f}_2\} \geq 0 \tag{C-1}$$

$$- s_1 + \alpha \cdot \min\{\hat{s}_1, \hat{s}_2 - d_1\} + (1-\alpha) \cdot \hat{s}_1 \geq 0 \tag{C-2}$$

$$f_1 - \alpha \cdot \check{f}_1 - (1-\alpha) \cdot \max\{\check{f}_2 + d_1, \check{f}_1\} \geq 0 \tag{C-3}$$

$$- f_1 + \alpha \cdot \min\{\hat{f}_1, \hat{s}_2\} + (1-\alpha) \cdot \hat{f}_1 \geq 0 \tag{C-4}$$

$$s_2 - \alpha \cdot \max\{\check{s}_2, \check{f}_1\} - (1-\alpha) \cdot \check{s}_2 \geq 0 \tag{C-5}$$

$$- s_2 + \alpha \cdot \hat{s}_2 + (1-\alpha) \cdot \min\{\hat{s}_2, \hat{s}_1 - d_2\} \geq 0 \tag{C-6}$$

$$f_2 - \alpha \cdot \max\{\check{f}_2, \check{f}_1 + d_2\} - (1-\alpha) \cdot \check{f}_2 \geq 0 \tag{C-7}$$

$$- f_2 + \alpha \cdot \hat{f}_2 + (1-\alpha) \cdot \min\{\hat{f}_2, \hat{s}_1\} \geq 0 \tag{C-8}$$

$$f_1 - s_1 - d_1 \geq 0 \tag{C-9}$$

$$f_2 - s_2 - d_2 \geq 0 \tag{C-10}$$

$$s_2 - f_1 + (1-\alpha) \cdot (\hat{f}_1 - \check{s}_2) \geq 0 \tag{C-11}$$

$$s_1 - f_2 + \alpha \cdot (\hat{f}_2 - \check{s}_1) \geq 0 \tag{C-12}$$

$$s_2 - s_1 - \alpha \cdot d_1 + (1-\alpha) \cdot (\hat{s}_1 - \check{s}_2) \geq 0 \tag{C-13}$$

$$s_1 - s_2 + \alpha \cdot (\hat{s}_2 - \check{s}_1) + (1-\alpha) \cdot d_2 \geq 0 \tag{C-14}$$

$$f_2 - f_1 - \alpha \cdot d_2 + (1-\alpha) \cdot (\hat{f}_1 - \check{f}_2) \geq 0 \tag{C-15}$$

$$f_1 - f_2 + \alpha \cdot (\hat{f}_2 - \check{f}_1) - (1-\alpha) \cdot d_1 \geq 0 \tag{C-16}$$

$$f_2 - s_1 - \alpha \cdot (d_1 + d_2) + (1-\alpha) \cdot (\hat{s}_1 - \check{f}_2) \geq 0 \tag{C-17}$$

$$f_1 - s_2 + \alpha \cdot (\hat{s}_2 - \check{f}_1) - (1-\alpha) \cdot (d_1 + d_2) \geq 0 \tag{C-18}$$

$$\alpha \geq 0 \tag{C-19}$$

$$(1-\alpha) \geq 0 \tag{C-20}$$

The fixed order polytopes are four-dimensional, and the full polytope is five-dimensional. We know if we lift any facet of a fixed order polytope, it will form a facet of the full polytope. This gives us facets (C-1) – (C-12).

Facets (C-1) – (C-8) are essentially upper and lower bound constraints, augmented with the relevant bound due to serialisation. For example, (C-1) always enforces $s_1 \geq \check{s}_1$, and, when x_2 executes first ($\alpha = 0$), also enforces $s_1 \geq \check{f}_2$.

Facets (C-9) and (C-10) are the minimum duration constraints. Since they apply in any ordering, they have no α term.

Facets (C-11) and (C-12) are the 'big M' constraints used to linearise the disjunction.

Facets (C-19) and (C-20) are the bounds on α.

Unfortunately, we have a problem: It's theoretically possible to form a facet of the full polytope using faces of the fixed order polytopes. Specifically, a face of dimension 2 (three affine independent points) from one polytope and a face of dimension 1 (two affine independent points) from the other. For this polytope, that theoretical possibility is a reality, resulting in facets (C-13) – (C-18).

Relationships Among Constraints

$$\begin{tikzcd}
 & \begin{array}{c} s_2 \geq f_1 \\ f_1 \leq s_2 \\ s_2 - f_1 \geq 0 \end{array} \arrow[dl] \arrow[dr] & \\
\begin{array}{c} s_2 \geq s_1 + d_1 \\ s_1 + d_1 \leq s_2 \\ s_2 - (s_1 + d_1) \geq 0 \end{array} \arrow[dr] & & \begin{array}{c} f_2 - d_2 \geq f_1 \\ f_1 \leq f_2 - d_2 \\ (f_2 - d_2) - f_1 \geq 0 \end{array} \arrow[dl] \\
 & \begin{array}{c} f_2 - d_2 \geq s_1 + d_1 \\ s_1 + d_1 \leq f_2 - d_2 \\ (f_2 - d_2) - (s_1 + d_1) \geq 0 \end{array} &
\end{tikzcd}$$

One way of seeing these facets is to notice that they are, in a sense, derived from transitivity relationships. The tighter bound will depend on the values of the parameters and variables.

More mechanistically, we used the following procedure:

- Taking combinations of facets, we determined the constraints that defined all faces of dimension 2 and dimension 1. (Roughly $C_2^{15} + C_3^{15} = 2730$ small LPs.)

- Choose a face of dimension 2 from one fixed order polytope, and a face of dimension 1 from the other, and see if they can form a facet. (Another few thousand tests, each a feasibility problem plus an LP.)

Since all the problems are trivially small, the computation is not a problem.

How to Use These Facets?

The constraints are facets only if the upper and lower bounds on the continuous variables are tight.

To be effective, the facets must be continually up-dated.

The details of the analysis tell which constraints are actually facets for a given set of duration and upper and lower bound values.

The facets reduce nicely for the more common case where the duration of an activity is fixed in advance.

These facets can only be used effectively in an environment where the upper and lower bounds on the continuous variables are regularly tightened. Each time the bounds are tightened, the facet must be rewritten.

These facets would be ideally suited for use in a branch & cut algorithm that incorporated bound tightening and coefficient strengthening. An environment that allowed algorithmic cut specifications to be placed in a cut pool would help, so that the appropriate facet(s) could be generated on demand.

The analysis collapses nicely to the case where the duration of an activity is fixed (so that independent finish times are not required).

Implementation

What's required is to keep the upper and lower bounds on start and finish times tight, and rewrite the constraints as those bounds change.

COIN has much of the necessary support machinery:

- Two implementations of bound propagation (probing and integer presolve).
- Tracked cuts (Osi)

The cut generator will need to recognise the appropriate constraint patterns and generate and update the facets.

Both CglProbing and CglPreProcess now incorporate simple constraint propagation routines. One of the first tasks will be to try and pull them out and write a general CglPropagate class. No sense creating a third implementation.

OsiRowCut2 may already provide the necessary hooks to track constraints in the constraint matrix, which will make it easy to update the facets as the bounds are tightened.

With the support machinery in place, writing the cut generator will be a matter of recognising the appropriate form and installing the cuts.