Discussion Problems: "ILP Encodings"
Declarative Methods (JHU 600.432/632)
Prof. Jason Eisner

1. Encode each of the following using one or more ordinary linear inequality constraints. Do not use any syntactic sugar (such as ZIMPL's `vif` syntax).

   (a) [2 points] $A \rightarrow (B \wedge C)$ (where $A, B, C \in \{0, 1\}$)

   (b) [4 points] $A \leftrightarrow (B \vee C)$ (where $A, B, C \in \{0, 1\}$ and $\leftrightarrow$ means "if and only if")

   (c) [3 points] if $p = 1$ then $r \geq 20$ (where $p \in \{0, 1\}, r \in \mathbb{R}$)

   (d) [3 points] $\max(s, t) \leq \min(u, v)$ (where $s, t, u, v \in \mathbb{R}$)

   (e) [4 points] $cost = 3x$ if $x \leq 10$, otherwise $cost = 3x + 2(x - 10)$ (where $x, cost \in \mathbb{R}$; you may introduce additional real variables as helpers, but you should not need any integer variables)

2. You are trying to put away your n games and puzzles on a shelf of width 100.0. Each one is in a two-dimensional box that must remain level (i.e., you cannot turn it diagonally or sideways). The ith box has positive width $w_i \in \mathbb{R}$ and positive height $h_i \in \mathbb{R}$ (for $i = 1, 2, \ldots n$).

   As far as you are concerned, Figure 1 is a feasible solution to this *rectangle-packing* problem.[1] Note that you are not too worried about the laws of physics here (perhaps because you're 4 years old). So some settling of your arrangement may occur later (unless you inject some kind of filler into the open spaces or lay your toy shelf flat). That's fine with you.

   The important thing is not to use any part of the space for more than one box—you're smart enough to realize that two boxes may not intersect, except for possibly touching at their edges or corners. In other words, **for any two (distinct) boxes in a feasible solution, one of them is to the right of the other, or one of them is fully above the other, or both.**

   (a) [3 points] What is the simplest type of mathematical programming model that will find an arrangement of *minimum total height*?

   (*Note:* A later part of this problem asks you to actually do this encoding. You might prefer to do that first.)

[1] Rectangle-packing has various other practical applications, including certain problems of scheduling, stock-cutting, and VLSI chip design.

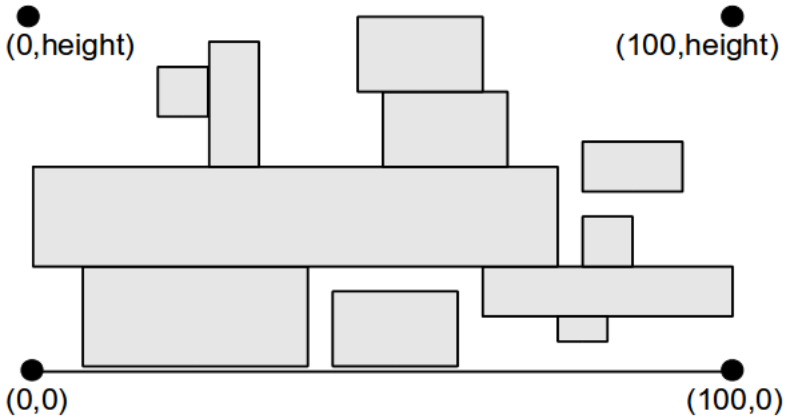

Figure 1: One feasible (though not optimal) plan for organizing your toy shelf. Here $n = 11$.

For full credit please give the name of the mathematical programming framework, e.g., linear programming. If you're not sure about that terminology, then specify what type of *variables*, *objective function*, and *constraints* are needed.

(b) [3 points] What is the simplest type of mathematical programming model that will find an arrangement of *minimum area*, in the sense of total height $\times$ total width? (You are still limited to maximum width of 100.0, but now you have an incentive to reduce that width.)

(c) [3 points] What is the simplest type of mathematical programming model that will find an arrangement of *minimum total height*, where you now allow the option of turning a box on its side (by 90^o)?

(d) [3 points] Suppose your n games and puzzles are stored in circles rather than rectangles. Now you have a circle-packing problem rather than a rectangle-packing problem. What is the simplest type of mathematical programming model that will find an arrangement of *minimum area*, in the sense of total height $\times$ total width?

(e) [3 points] Suppose you only have a solver for part (a). How could you use it to solve the problem for part (b)?

(f) [12 points] Write out the mathematical programming model for part (a). You may use ordinary mathematical notation or a ZIMPL-like notation. But do not use any ZIMPL extensions like `vabs` (which won't help anyway). You may find it necessary to use slack variables.

Please indicate how you will decode your output. For example, if the solver finds that

$x_3 = 20$, how does that affect your toy placement?

(g) [5 **extra credit** points] There may be many solutions that have the same total height and width. For example, within Figure 1, the boxes could be slid around considerably without changing the total height or width.

Suppose you would like to break ties by preferring solutions where the boxes are **as low as possible** in some appropriate sense.[2] (Among other things, this will rule out solutions like Figure 1 where some boxes are completely "floating"; such a solution is suboptimal since a floating box can necessarily be lowered to achieve a better solution. However, it does more than just rule out floating boxes—in general it will prefer to pile up the boxes in a way that most of them are close to the ground.)

How can you accomplish this? For full credit, try to ensure that your change *only* breaks ties. That is, while you do want to rearrange the boxes to get them lower, *other things equal*, you should never rearrange them in a way that will increase the total height of the optimal solution. (That is why it's an extra credit problem!)

[2]This only breaks "vertical ties." You could use the same ideas to prefer the boxes to be toward the left, thus breaking horizontal ties as well. But don't worry about that for purposes of this exam problem.

3. The 9/11 memorial at Ground Zero in New York City includes the names of the 2,982 victims, engraved on bronze panels around cascading pools of water where the World Trade Center towers used to be.

The memorial's architect first tried rearranging 2,982 index cards to choose the placement of names, but finally called in the computer scientists.[3] The positive result of this collaboration was reported in April 2011 in the newspaper,[4] and you can browse it at names.911memorial.org.

What was being optimized? The memorial foundation had decided that people who were

[3]The first few computer scientists they called said it couldn't be done, but I guess they hadn't taken this course.

[4]E.g., http://www.newyorker.com/talk/2011/05/16/110516ta_talk_paumgarten, http://www.nytimes.com/2011/05/05/nyregion/on-911-memorial-constructing-a-story-name-by-name.html, and several good blogs.

related in some way—kin, friends, co-workers, etc.—should appear together on the memorial. The image below shows how, at her family's request, Angela Houtz was listed near the 7 others who were at her last meeting. Such requests were made by the families of 1,200 victims, and there were also other sources of these "meaningful adjacencies."

Other constraints came from the physical shape of the bronze panels, the lengths of the names, and aesthetic criteria such as avoiding vertical blocks of space.

It should be possible to formulate this kind of problem using ILP (more precisely, MIP). For exam purposes, let's consider a simpler, 1-dimensional version where you are just trying to achieve "meaningful adjacencies" in a vertical list of names.

You are given a set of size n, called `Names`. You are also given a sparse symmetric nonnegative matrix of meaningful adjacencies:
`param a[Names*Names];`
For each pair of names x, y, we have `a`$[x, y] =$ `a`$[y, x] \geq 0$.

`a`$[x, y]$ is to be interpreted as "the importance of having x and y near each other." If it is 0, that means that x and y are unrelated. If it is positive, then you will be increasingly happy with an arrangement as x and y move closer together. For example, if `a`$[x, y] = 5$, then the costs of being 1,2,3,4,5,... positions apart should be 0,5,10,15,20,..., respectively.

This problem may remind you somewhat of the Linear Ordering Problem (LOP) and/or the Traveling Salesperson Problem (TSP). Both of those similarly seek a high-scoring permutation of n objects. The objective here will be different, though.

Throughout this problem, use approximate ZIMPL notation and/or mathematical notation. Please stick to plain linear constraints and integrality constraints. That is, do not use convenience constructions like `vabs`, `vif`, `alldifferent`, `!=`, but rather use ILP to carry out this encoding. Also, be careful that you do not accidentally multiply variables by other variables, which would give non-linear functions.

(Of course, you may may still write `x != y` where `x` and `y` are names, since those are merely ZIMPL variables used to generate the ILP program that the SCIP solver will see. They are not numeric variables that actually appear in the ILP program.)

(a) [8 points] In one encoding of the problem, you'll solve for the integer position of each name:
`var pos[Names] integer;`
Write the constraints and the max or min objective, declaring any auxiliary variables that they may refer to. You will need some hard constraints to ensure that the optimal collection of positions can be decoded into an ordered list.

(b) [4 points] In another encoding of the problem, you'll instead solve for the integer distances between each pair of names: `var dist[Names*Names] integer;`
Again, write the constraints and max or min objective, declaring any auxiliary variables that they may refer to. You will need some hard constraints to ensure that the optimal collection of distances can be decoded into an ordered list.

(*Hint:* Encode the ordering using non-negative distances as follows: if x falls 3 positions earlier than y, then `dist[`x, y`]=3` and `dist[`y, x`]=-3`. Thus, `dist[`x, y`]` plays the role of `pos[`y`]-pos[`x`]` in the earlier problem, and you should be able to take your previous solution as a starting point.)

(c) [5 points] Which encoding do you think would be more efficiently solved with SCIP? Discuss.

(*Hint:* Think about how you'd do it on index cards, and what SCIP strategy that corresponds to. Would that be a good strategy for SCIP too?)

(d) Let's try using a nonlinear cost. Specifically, if `a[`x, y`]` $= 5$, then the costs of x and y being 1,2,3,4,5,... positions apart should be 0,5,110,215,320,..., respectively. (This is 0,0,100,200,300,... more than our original objective.) In other words, we would like to be adjacent (distance 1), and we pay only a small price for distance 2, but we pay an extra penalty of 100 per unit of distance beyond 2.

i. [2 points] Is this a convex cost function?

ii. [2 points] Why does it matter whether it is a convex cost function?

iii. [5 points] Pick one of your encodings above (your choice), and explain how to modify it to use this cost function.

(e) Consider a different version of the above problem where you want the costs of distance 1,2,3,4,5,... to be 0,5,110,415,920,.... (This is 0,0,100,400,900,... more than our original objective. So the extra penalty is now proportional to the *square* of the distance beyond 2.)

i. [2 points] What would be the most natural type of mathematical programming to encode this?

ii. [3 points] If you didn't have access to a solver for that type of math programming, could you solve the problem with mixed integer programming? Explain.

(f) [6 points] Certain names should *not* be exactly adjacent in the vertical list, for typographic reasons. For example,

```
Jane Jones
John Smith
```

line up too neatly, inadvertently attracting the eye to this coincidence.[5]

Pick one of your solutions above, and explain how to modify it to handle this issue. Assume that as part of the input, you are given a sparse matrix of "bad adjacencies,"

```
param b[Names*Names];
```

These `b[`x, y`]` values are binary. If `b[`x, y`]=1`, you must impose a hard constraint that `dist[`x, y`]` $\neq 1$ (that is, x may not appear *immediately* before y).[6]

[5] Worse, there were two victims both named `Michael Francis Lynch`, who would have been placed together in an alphabetical ordering.

[6] If `b[`y, x`]=1` as well, then x may not appear immediately *after* y either. Typically the `b` matrix would be symmetric in this way, but that's up to the user who provides it.