Yet Another Math Programming Consultant

I am a full-time consultant and provide services related to the design, implementation and deployment of mathematical programming, optimization and data-science applications. I also teach courses and workshops. Usually I cannot blog about projects I am doing, but there are many technical notes I'd like to share. Not in the least so I have an easy way to search and find them again myself. You can reach me at erwin@amsterdamoptimization.com.

Tuesday, February 16, 2021

2d Bin Packing

In [1], a question was posed about a particular type of 2d bin-packing. This is a good reason to play with some models.

First I discuss three formulations with continuous no-overlap constraints and one for discrete data. In the next post, I'll discuss the variant described by the post [1]. This actually leads to a very different, but easy problem.

Problem statement and example data

We consider $n=50$ rectangular items of different sizes that have to be placed into bins. There are six different categories, each with a different size (height and width). All the bins have the same size. The goal is to use as few bins as possible.

```
----     37 PARAMETER itemSize  sizes of bin and items

               w           h

cat1       7.000      12.000
cat2       9.000       3.000
cat3       5.000      14.000
cat4      13.000       9.000
cat5       6.000       8.000
cat6      20.000       5.000
bin       40.000      60.000


----     37 SET itemMap  mapping between items and categories

item1 .cat1,    item2 .cat1,    item3 .cat1,    item4 .cat1,    item5 .cat1,    item6 .cat1,    item7 .cat1
item8 .cat1,    item9 .cat1,    item10.cat1,    item11.cat2,    item12.cat2,    item13.cat2,    item14.cat2
item15.cat2,    item16.cat2,    item17.cat2,    item18.cat2,    item19.cat2,    item20.cat2,    item21.cat3
item22.cat3,    item23.cat3,    item24.cat3,    item25.cat3,    item26.cat3,    item27.cat3,    item28.cat3
item29.cat3,    item30.cat3,    item31.cat4,    item32.cat4,    item33.cat4,    item34.cat4,    item35.cat4
item36.cat4,    item37.cat4,    item38.cat4,    item39.cat4,    item40.cat4,    item41.cat5,    item42.cat5
item43.cat5,    item44.cat5,    item45.cat5,    item46.cat6,    item47.cat6,    item48.cat6,    item49.cat6
item50.cat6
```

Furthermore, we assume items cannot be rotated.

Continuous Model A

In this formulation, we will **not** assume that our sizes are integer-valued. The rectangles are allowed to have any positive width and height.

The model consists of three parts:

- Assignment of items to bins.
- We position the items inside a bin. No-overlap constraints are used for the items placed in the same bin.
- Some ordering constraints. This can make the solution look better and reduce symmetry in the problem.

In the following, i and j indicate items, and k refers to a bin. The number of items is known. The number of bins is large enough such that we know all items can fit. Here I just use 10 bins. A better approach would be to have some heuristic that finds a good initial configuration.

Assignment
To model the assignment, we introduce binary variables

$$a_{i,k} = \begin{cases} 1 & \text{if item } i \text{ is placed in bin } k \\ 0 & \text{otherwise}\end{cases}$$

Each item should go to exactly one bin, so we have

$$\sum_k a_{i,k} = 1 \ \forall i$$

Bounds
The items should be placed inside a bin:

$$\begin{aligned} &x_i \in [0, w_{bin} - w_i] \\ &y_i \in [0, h_{bin} - h_i]\end{aligned}$$

No-overlap
This is a bit more involved. We need to make sure when comparing items i and j assigned to the same bin k that one of the following conditions hold: item i is to the left, or to the right, or above, or below item j. First we need to establish if two items i and j are in the same bin. This can be done with

$$\begin{aligned} &s_{i,j} \ge a_{i,k} + a_{j,k} - 1 \ \forall k, i<j \\ &s_{i,j} \in [0,1]\end{aligned}$$

This implements the implication

$$a_{i,k} = a_{j,k} = 1 \Rightarrow s_{i,j} = 1$$

To prevent double work, we only do this for $i<j$. With this we can implement our no-overlap constraints:

$$\begin{aligned} &x_i + w_i \le x_j + M(1-\delta_{i,j,1}) + M(1-s_{i,j}) \\ &x_i \ge x_j + w_j - M(1-\delta_{i,j,2}) - M(1-s_{i,j}) \\ &y_i + h_i \le y_j + M(1-\delta_{i,j,3}) + M(1-s_{i,j}) \\ &y_i \ge y_j + h_j - M(1-\delta_{i,j,4}) - M(1-s_{i,j}) \\ &\sum_p \delta_{i,j,p} \ge 1 \\ &\delta_{i,j,p} \in \{0,1\}\end{aligned}$$

A conservative value for M would be to set it equal to the bin width (for the x constraints) or height (for the y constraints).

Objective
The objective is to minimize the number of used bins. A bin is used if there at least one item in it. So we can write:

$$\begin{aligned} \min & \sum_k u_k \\ & u_k \ge a_{i,k} & \forall i,k \\ & u_k \in [0,1]\end{aligned}$$

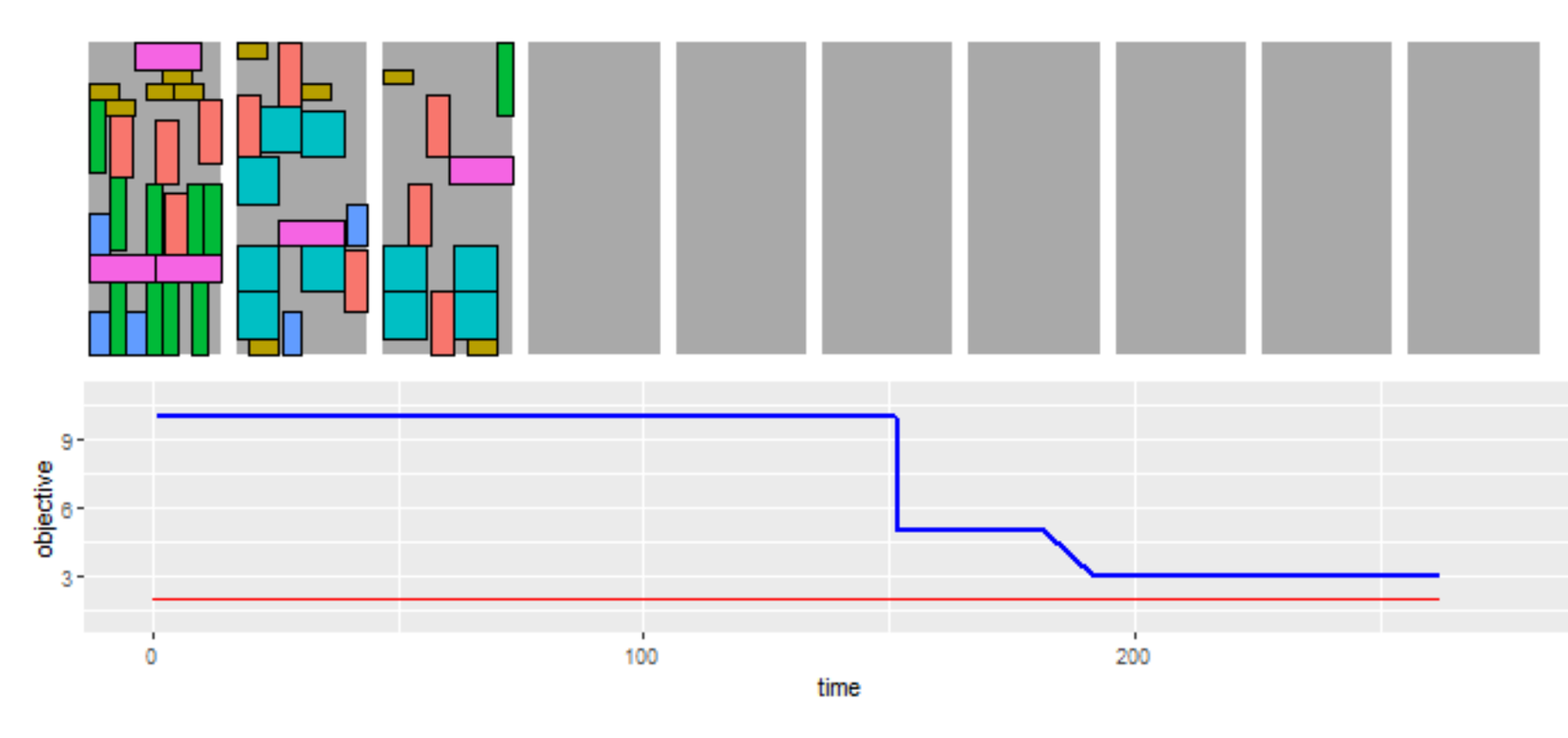
Order used bins
It may be preferable to have the unused bins be the higher-numbered ones and the used bins the lower-numbered ones. We can easily enforce that bin k is used before bin $k+1$:

$$u_k \ge u_{k+1}$$

Order by area
Optionally we can order the bins by area occupied. That would differentiate the bins more than the previous ordering.

Lower bound
We can calculate a lower bound on the objective by just looking at the area of the bin and the items. The total area of all the items does not fit in just one bin. So we know we need at least two bins. This also means that as soon as we have found a solution with two bins we can stop.

Does it work?
This can be a very difficult model to solve. In this case, the performance is not bad at all. Here we see the MIP solver in action:



Summary of the model

MIP Model A

$$\begin{aligned} \min & \sum_k u_k \\ & u_k \ge a_{i,k} & \forall i,k \\ & \sum_k a_{i,k} = 1 & \forall i \\ & s_{i,j} \ge a_{i,k} + a_{j,k} - 1 & \forall k, i<j \\ & x_i + w_i \le x_j + M(1-\delta_{i,j,1}) + M(1-s_{i,j}) & \forall i<j \\ & x_i \ge x_j + w_j - M(1-\delta_{i,j,2}) - M(1-s_{i,j}) & \forall i<j \\ & y_i + h_i \le y_j + M(1-\delta_{i,j,3}) + M(1-s_{i,j}) & \forall i<j \\ & y_i \ge y_j + h_j - M(1-\delta_{i,j,4}) - M(1-s_{i,j}) & \forall i<j \\ & \sum_p \delta_{i,j,p} \ge 1 & \forall i<j \\ & u_k \ge u_{k+1} & \forall k \text{ except last} \\ & u_k \in [0,1] \\ & a_{i,k} \in \{0,1\} \\ & s_{i,j} \in [0,1] \\ & \delta_{i,j,p} \in \{0,1\}\end{aligned}$$

The variables with domain $[0,1]$ (i.e. continuous between 0 and 1) can also be declared binary. In my test run, I used relaxed variables. It is noted that solvers may have a very difficult time proving optimality for this model. Often they find good solutions quite quickly, but actually terminating with a proven optimal solution may take a long time. Also, there is a wide variation in performance for very similar data sets.

Alternative model B

In [2] a somewhat different and strange MIP model is proposed. Let's have a look:

$$\begin{aligned} & Z = \sum_{i=0}^{n} \alpha_{ii} \rightarrow min & & (1) \\ & \sum_{k=0}^{n} \alpha_{ik} = 1 & i,k \in Q; i \ge k & \quad (2) \\ & \alpha_{ik} \le \alpha_{kk} & i,k \in Q; i \ge k & \quad (3) \\ & x_i + w_i \le W & i \in Q & \quad (4) \\ & y_i + h_i \le H & i \in Q & \quad (5) \\ & ul_{ij} + ua_{ij} + ur_{ij} + uu_{ij} = 1 & i,j \in Q; i<j & \quad (6) \\ & x_i + w_i \le x_j + W \cdot (3 - ul_{ij} - \alpha_{ik} - \alpha_{jk}) & i,j,k \in Q; k \le i < j & \quad (7) \\ & y_i + H \cdot (3 - ua_{ij} - \alpha_{ik} - \alpha_{jk}) \ge y_j + h_j & i,j,k \in Q; k \le i < j & \quad (8) \\ & x_i + W \cdot (3 - ur_{ij} - \alpha_{ik} - \alpha_{jk}) \ge x_j + w_j & i,j,k \in Q; k \le i < j & \quad (9) \\ & y_i + h_i \le y_j + H \cdot (3 - uu_{ij} - \alpha_{ik} - \alpha_{jk}) & i,j,k \in Q; k \le i < j & \quad (10)\end{aligned}$$

I don't immediately recognize much of my model in this version. The paper mentions that all data is assumed to be integer-valued. This is actually not required for this model: it is happy with floating-point values.

The first thing we have is a single set: $Q=\{1,\dots,n\}$ for both the items and the bins. That is interesting. In my model, I would dimension the set k (the bins) as small as possible (e.g. by using some heuristic). Here we have $n=50$ items that can fit in 2 bins, and this model assumes we dimension for 50 bins.

We also immediately notice problems in the indexing. The set Q starts with element 1, but the model uses indices 0. I assume we start the summations at 1.

The matrix $\alpha_{i,k} \in \{0,1\}$ indicates if item i is placed in bin k. This looks like my assignment variable $a_{i,k} \in \{0,1\}$. Well, that is only superficially so. Here $\alpha_{i,k}$ has the following properties:

- Only $\alpha_{i,k}$ for $i \ge k$ is used. (This essentially means $\alpha_{i,k}=0$ for $i<k$).
- The diagonal $\alpha_{k,k}$ indicates if bin k is used. This explains the formulation of the objective.
- If the diagonal element is zero, the whole column should be zero, This is stated in constraint (3). We can skip equation (3) for the case $i=k$. I.e. we only need to apply it for $i>k$ instead of $i \ge k$.

The variable ul, ur, uu, ua are similar to my variables δ. The term $W \cdot (3-ul_{ij} - \alpha_{ik} - \alpha_{jk})$ says that the corresponding constraint should be binding only if all $ul_{ij} = \alpha_{ik} = \alpha_{jk}$. Similar for the other no-overlap constraints. In my model, I don't need as many of these big-M constraints as I calculated $s_{i,j}$ separately.

Alternative model C

In [3,4] the following model is shown:

$$\begin{aligned} & \min v & (1) \\ & \text{subject to:} \\ & l_{ij} + l_{ji} + b_{ij} + b_{ji} + p_{ij} + p_{ji} \ge 1 \quad \text{for } i,j \in I, i<j & (2) \\ & x_i - x_j + Wl_{ij} \le W - w_i \quad \text{for } i,j \in I & (3) \\ & y_i - y_j + Hb_{ij} \le H - h_i \quad \text{for } i,j \in I & (4) \\ & m_i - m_j + np_{ij} \le n-1 \quad \text{for } i,j \in I & (5) \\ & x_i \le W - w_i \quad \text{for } i \in I & (6) \\ & y_i \le H - h_i \quad \text{for } i \in I & (7) \\ & m_i \le v \quad \text{for } i \in I & (8) \\ & 1 \le m_i \quad \text{for } i \in I & (9) \\ & m_i \le i \quad \text{for } i \in I & (10) \\ & l_{ij}, b_{ij}, p_{ij} \in \{0,1\} \quad \text{for } i,j \in I, i \ne j \\ & x_i, y_i, m_i, v \in \mathbb{N}\end{aligned}$$

The integer variable $m_i \ge 1$ is the bin number where item i is placed.

This model has binary variables:

- $l_{i,j}$ indicating that item i is to the left of item j,
- $b_{i,j}=1$ means: item i is below item j,
- $p_{i,j}=1$ indicates $m_i < m_j$.

Constraint (1) says: items i and j should not overlap (i.e. below or left of one another), or they should be assigned to a different bin. Equations (2) and (3) are the familiar no-overlap constraints (somewhat rearranged). Note that there we apply them for all $i \ne j$. Equation (5) implements the implication: $p_{i,j}=1 \Rightarrow m_i \le m_j - 1$.

The model is described in [3] for integer data. However, this model works perfectly fine for data using floating-point values.

Grid model D

Here we assume that place the items at integral grid points. I used a 60×40 grid. The idea of this model is to look at a grid point (r,c) and look at all possible placements that overlap this point. The sum of placements we can make in this set is 1.

Our decision is variable is:

$$x_{t,r,c,k} = \begin{cases} 1 & \text{if cell } (r,c) \text{ in bin } k \text{ is occupied by an item of category } t \\ 0 & \text{otherwise}\end{cases}$$

We need to place all items, so we use:

$$\sum_{r,c,k} x_{t,r,c,k} = num_t$$

where num_t is the number of items of category t.

To protect against overlap of cell (r,c), I created a huge set $affect(r,c,t,r',c')$ indicating which placements $x_{t,r',c',k}$ have an impact on cell (r,c). For our data set, this set has about one million elements. With this under our belt, we can form the constraint:

$$\sum_{t,r',c'|affect(r,c,t,r',c')} x_{t,r',c',k} \le 1 \ \forall r,c,k$$

To keep track of used bins we use the variable u_k and impose:

$$x_{t,r,c,k} \le u_k \ \forall t,r,c,k$$

As suggested by Rob Pratt, we can improve on this a bit by combining these constraints:

$$\sum_{t,r',c'|affect(r,c,t,r',c')} x_{t,r',c',k} \le u_k \ \forall r,c,k$$

This has two advantages: smaller and simpler model and more importantly this is a tighter formulation.

The complete model can look like:

Grid Model D

$$\begin{aligned} \min & \sum_k u_k \\ & \sum_{r,c,k} x_{t,r,c,k} = num_t & \forall t \\ & \sum_{t,r',c'|affect(r,c,t,r',c')} x_{t,r',c',k} \le u_k & \forall r,c,k \\ & u_k \ge u_{k+1} & \forall k \text{ except last} \\ & x_{t,r,c,k} \in \{0,1\} \\ & u_k \in [0,1]\end{aligned}$$

It is noted we could have expressed the x variables in terms of items i instead of categories t. That would have made the model even larger (it is already a big model).

Performance

A quick test on our small data set.

	Model A	Model B	Model C	Model D
Data	Continuous	Continuous	Continuous	Integer
Equations	18,945	85,801	8,625	24,016
Variables	6,746	6,276	7,501	144,011
Discrete variables	5,400	6,175	7,400	144,010
Nonzeros	63,289	425,176	29,500	10,558,099
Objective	2	2	2	2
Time	280	793	338	4,825
Nodes	1,932	10,109	104,495	0
Iterations	281,877	240,258	2,078,174	227,935

In all continuous models, I added the lowerbound

$$obj \ge \left\lceil \frac{\sum_i area_i}{area_{bin}} \right\rceil$$

This really helps. Without this bound the solver just spends a lot of time proving that 2 is the optimal objective. An area constraint (sum of area of assigned items to a bin cannot exceed bin area) can also help with this. I also used the solver option **mipemphasis** to put more effort into reaching optimality.

The integer model D is losing out because of its size. It is easy to solve,, but its size makes the presolving and preprocessing expensive. I had the reduce the setting for cut generation in order to make it progress during preprocessing.

References

1. How to model fixed width columns for 2d bin/strip packing problem?, https://stackoverflow.com/questions/66156154/how-to-model-fixed-width-columns-for-2d-bin-strip-packing-problem
2. Christian Blum, Verena Schmid and Lukas Baumgartner, *On Solving the Oriented Two-Dimensional under Free Guillotine Cutting: Exploiting the Power of Probabilistic Solution Construction*, sept. 2012,. https://www.researchgate.net/publication/230795080_On_Solving_the_Oriented_Two-Dimensional_Bin_Packing_Problem_under_FreeGuillotine_Cutting_Exploiting_the_Power_of_Probabilistic_SolutionConstruction
3. Christian Blum, Verena Schmid, *Solving the 2D bin packing problem by means of a hybrid evolutionary algorithm*, Procedia Computer Science 18 (2013) 899 – 908, https://www.sciencedirect.com/science/article/pii/S1877050913003980
4. David Pisinger, Mikkel Mühldorff Sigurd, *Using Decomposition Techniques and Constraint Programming for Solving the Two-Dimensional Bin-Packing Problem,* January 2007, Informs Journal on Computing 19(1):36-51

Posted by Erwin Kalvelagen at 6:05 AM
Labels: bin packing, Mixed-Integer Programming, Modeling

3 comments:

Greg Glockner February 16, 2021 at 1:56 PM
I looked at a similar model many years ago. In my case, everything fell onto a grid, so I wrote a very different model: x[i,j,k] if item k is in position i,j. Then I had a constraint for every position in the grid: sum (ii,jj,k covering i,j) x[i,j,k] <= 1 for all grid positions (i,j). Performance was much better - no disjunctive constraint.
Reply

Replies

Erwin Kalvelagen February 16, 2021 at 3:17 PM
I think in my case I need x(item,row,col,bin). About 1.2e6 variables if I did not make a mistake. Worth a try...

Rob Pratt February 17, 2021 at 7:53 AM
You can simultaneously strengthen and shrink Model D by replacing <= 1 with u[k] and then omitting the x <= u constraints, which are then dominated.

Reply

Enter your comment...
Comment as: huuhieuht2k@ Sign out
Publish Preview Notify me

Blog Archive

Search This Blog
Search