CPS - ASSIGNMENT 2

LINEAR PROGRAMMING

May 28, 2018

Kymry Burwell

UPC

Contents

1 VARIABLE DEFINITION

For this problem, I envisioned the paper roll as a set of boolean grid points. Each box either occupies or does not occupy a grid point. The following variables hold the input data.

- **IloInt w:** Width of the grid
- **IloInt h:** Height of the grid
- **loInt grid:** Size of the grid (width×height)
- **IloInt numB:** Number of boxes
- **IloInt M:** A large integer used for the big M trick
- **IloNumArray xDim:** X-dimensions of all boxes
- **IloNumArray yDim:** Y-dimensions of all boxes
- **IloInt minLength:** Theoretical minimum length of the roll. This may be smaller than the minimum, but will never be larger.

The following decision variables are used in the constraint and objective functions. CPLEX uses arrays and I envisioned the problem as a matrix, so each of the following variables are simply long arrays that represent a matrix.

- **IloNumVarArray b:** Boolean array that keeps track of which positions in the paper roll each box occupies. Therefore, every box has $width \times height$ positions in this array. The total length of the array is therefore $numB \times grid$. For example, $b[0] = 1$ signifies the first box occupies position (0,0) of the paper roll. Whereas if $b[0] = 0$, the first box would not occupy position (0,0) of the roll.
- **IloNumVarArray tl:** Boolean array that keeps track of the top-left starting position of each box. Again, this has a total length of $numB \times grid$.
- **IloNumVarArray boolOr:** Boolean array used for constraints 4 and 5 to represent an OR constraint.
- **IloNumVar length:** Single integer that holds the length of the paper roll.

2 CONSTRAINTS

In order to make the constraints easier to display and explain here, I will be interpreting the decision variables as three dimensional arrays. Here, **B** is the set of boxes, **X** is the set of x-coordinates on the paper roll, and **Y** is the set of y-coordinates on the paper roll.

For example, the decision variable **b** can be envisioned as b[box][x][y]. Then, b[2][3][4] == 1 states that box2 occupies grid point (3,4). Or, b[5][1][7] == 0 states that box5 does not occupy grid point (1,7).

Similarly, the decision variable **tl** can be envisioned as tl[box][x][y]. Then tl[3][1][2] == 1 states that box 3 has a top-left starting position of (1,2).

(1) Each box has one top-left position. Again, **tl** is a boolean variable that specifies if the box occupies the grid point as its top-left starting position.

$$\forall box \in B : \sum_{x \in X} \sum_{y \in Y} tl[box][x][y] == 1$$

(2) Each box occupies the correct number of grid points on the paper roll. Remember, the decision variable **b** is a boolean variable that specifies if the box occupies the grid point.

$$\forall box \in B : \sum_{x \in X} \sum_{y \in Y} b[box][x][y] == xDim[box] \times yDim[box]$$

(3) Boxes don't overlap (i.e. each grid point is occupied by at most one box)

$$\forall x \in X, \forall y \in Y : \sum_{box \in B} b[box][x][y] \leq 1$$

$$\forall x \in X, \forall y \in Y : \sum_{box \in B} tl[box][x][y] \leq 1$$

(4 and 5) Boxes can be place horizontally or vertically AND boxes must adhere to the proper dimensions.

$$\forall box \in B, \forall x \in X, \forall y \in Y:$$

$$tl[box][x][y] \times xDim[box] \times yDim[box] \leq \sum_{i=0}^{yDim[box]} \sum_{j=0}^{xDim[box]} b[b][x+j][y+i] + boolOr[box] \times M$$

AND

$$tl[box][x][y] \times xDim[box] \times yDim[box] \leq \sum_{i=0}^{yDim[box]} \sum_{j=0}^{xDim[box]} b[b][x+i][y+j] + (1 - boolOr[box]) \times M$$

(6) Symmetry Breaking: Boxes of same dimensions must be placed in increasingly higher grid positions. This constraint is very hard to show in equations, so I will just explain it in text. If there are two boxes that are of equal dimensions, say box1 and box2, then the top-left starting position (tl) of box2 must be strictly great than the top-left position of box1.

(7) Minimum length of the paper roll is recorded. This constraint simply sets the **length** decision variable to the length position of the box that is currently occupying the highest length.

$$\forall box \in B, \forall x \in X, \forall y \in Y : length \geq b[box][x][y] \times y$$

(8) Length is at least the theoretical minimum. This prevents the program from searching for a minimum length that is unfeasible.

$$length \geq minLength$$

3 Objective Function

The objective function is very simple, we simply need to find the minimum length. The constraints take care of most of the hard work, which is why it can be expressed in such a minimal way.

$$minimize : length$$