ENGSCI 761

Multiobjective Optimisation Assignment

Due Tuesday 29/05/2018 by 8pm
(submit all files in one zipped folder on Canvas)

1. Consider the multiple bin packing problem from the IP Labs.

   - You have a number of items $i \in \{1, 2, \ldots, m\}$ with volume v_i.
   - You need to pack these items into bins $j \in \{1, 2, \ldots, n\}$ to minimise waste in the bins being used.
   - Each item has a type $a_i \in \{1, 2, \ldots, T\}$.
   - In packing the bins you want to minimise the number of different types of items in each bin. To do this, you can count the number of items of each type in each bin. Obviously, there is always at least one type of item in a bin. Here we want to minimise the number of items of different types in each bin, exceeding one. For instance, we have items $1, 2, 3, 4, 5$ with types $a_1 = 2, a_2 = 2, a_3 = 1, a_4 = 3, a_5 = 1$. Assume for this examples that all items would fit in one bin. *Note that we won't make this assumption for parts (a) to (d) of the question.*

     (i) Pack all items $1, 2, 3, 4, 5$ into bin 1. Bin 1 now has three types of items, hence the total number of mixed types is 2. (Note there is one bin with mixed types, but there are 3 types mixed up, and two types are counted in the objective).

     (ii) Pack items $1, 2, 3$ into bin 1, and items $4, 5$ into bin 2. Then we have items of two different types in bin 1. We also have items of two different types in bin 2. The total number of mixed types across all bins is then 2.

     (iii) Pack items of type 1 (items $3, 5$) into bin 1 and pack items of type 2 and 3 (items $1, 2, 4$) into bin 2. This leads to a total number of 1 mixed type across all bins.

     (iv) An optimal solution (for single objective problem minimising total number of different types in each bin) could be to pack items of type 1 (items $3, 5$) into bin 1, items of type 2 (items $1, 2$) into bin 2, and items of type 3 (item 4) into bin 3, leading to no bins containing a mix of types.

The multiple bin packing problem minimising the waste in the bins is modelled as follows:

$$
\begin{aligned}
\min \quad & \sum_{j=1}^{n} w_j \\
\text{s.t.} \quad & \sum_{i=1}^{m} v_i y_{ij} + w_j = V_j x_j && j = 1,2,\ldots,n \\
& \sum_{j=1}^{n} y_{ij} = 1 && i = 1,2,\ldots,m \\
& x_j \le x_{j+1} && j = 1,2,\ldots,m-1 \\
& w \ge 0, w \in \mathbb{R}^n \\
& x \in \{0,1\}^n, y \in \{0,1\}^{m\times n},
\end{aligned}
$$

where $x_j = 1$ if bin j is used and 0 otherwise, and $y_{ij} = 1$ if item i is placed in bin j and 0 otherwise. Note that the model also includes constraints $x_j \le x_{j+1}$ to remove some of the symmetry due to choice of bins.

(a) Formulate (on paper) the second objective for this optimisation problem that minimises the number of different types of items across all bins as explained above. You may need to introduce new variables and constraints. Define all new notation. Hand in your formulation.

(b) Solve a *lexicographic* version of the bin packing problem with first-priority objective to minimise *waste in bins* $\mathcal{W}$ and second-priority objective to minimise *number of different types across all bins* $\mathcal{T}$. Solve this lexicographic optimisation problem by solving the bin packing problem with objective $\mathcal{W} + \lambda\mathcal{T}$ where you make an adequate choice of λ. Make sure you choose a small λ which is sufficiently large, based on the problem data.
Hand in

- The values of $\mathcal{W}$ and $\mathcal{T}$ for your lexicographic solution.
- A justification of your choice of λ.

(c) Solve the bi-objective bin packing problem with two objectives: $\mathcal{W}$ and $\mathcal{T}$ for the provided problem instance with $m = 20$ in `MO_bin.dat`.
Hand in

- A brief statement outlining the approach you chose to solve this bi-objective problem and why. Also state any assumptions you are making (if any). Be sure to state what stopping condition you use for your approach.
- Your implementation of the solution approach (with all files needed to run it). It needs to be straight forward to use, and run for other input data (in the same format) as well.
- A list or table of the objective function values of all solutions to the bi-objective problem. Two of your efficient solutions (i.e. the

allocation of items to bins) presented in a form that your manager could understand: The solution that minimises total waste and the next solution found by your algorithm.

(d) With the current model, having three different types in a single bin, as in example (i) above, or two different types in two bins, as in example (ii) above, results in the same objective function value. How could you modify your model so that mixing two types in two bins bin is considered a better solution than mixing three type in one bin, i.e. the first case (i) is considered worse than case (ii).
Hand in a description of the new model on paper, you don't have to implement it.

Implementing your solution

You are provided with a python (pulp) model of the bin packing problem problem `bin_pack_MO_start.py` and some data files. This model uses CBC or Gurobi as solver, not dippy. Please implement your solution method in python (using the pulp package), ideally based on the provided code.

You should be able to run the provided file in the python environment you set up in Mike's IP labs. Instructions on setting up an environment that contains pulp only for the purpose of this assignment are also on Canvas. There are also instructions how to modify a windows path variable so that you can call Gurobi from pulp. Gurobi is already installed on lab computers, just not on the system path, hence not callable from command window. (It's straight forward to add the required path variable and recommended as Gurobi has better output and is faster).

The following may be useful:

- If you define a constraint called `myConstraint` in pulp and later want to modify its right hand side, it works like this:

```
prob = pulp.LpProblem("My Problem",LpMinimize)
# ... omitted: define problem objective and constraints

# now define constraint of interest
#   (constraint formula omitted)
prob += ... <= 17, 'myConstraint'

# now change right hand side to 21:
prob.constraints['myConstraint'].changeRHS(21)
```

- Be careful when working with the values of continuous variables that represent integer quatities. Just casting these to integer may not yield the right results. For example `a = int(1.9999999)` in python gives `a = 1` which may not be what you expect.