Dantzig wolfe decomposition for the Multiple Knapsack Problem with Setup (MKPS)

This project must be handed in at the latest on Wednesday March 23rd 2022 at 23:59. The project should be carried out in groups of 2 to 3. The report you hand in must be made solely by your group. All models, programs, text, figures, tables and so on should be your own! It is important that you explain the methods employed (The point of this assignment is to show that you understand the theory and can apply it). Use of figures and illustrations is encouraged. Remember to answer all questions in your report.

The report should be in PDF format. Remember to indicate all group members on the report. The front page should contain a table that show each group members contribution to each of the 10 tasks in the assignment. Remember that this report counts in the grading of the course. Code and data should be uploaded to DTU Learn, but the assessment will be based on the report.

1 Background

In the Multiple Knapsack Problem with Setup (MKPS) we are given N classes of items. Each class $i \in \{1,\ldots,N\}$ contains n_i items. We are given T knapsacks and each knapsack $t \in \{1,\ldots T\}$ has capacity b_t.

We can obtain a profit when including an item it in one of the knapsacks. The profit of including item $j \in \{1,\ldots,n_i\}$ from class $i \in \{1,\ldots,N\}$ in knapsack $t \in \{1,\ldots T\}$ is given by the constant c_{ijt}. Item $j \in \{1,\ldots,n_i\}$ from class $i \in \{1,\ldots,N\}$ uses a_{ij} units of capacity from the knapsack it is placed in. Furthermore, we can only place an item in a knapsack if we have "setup" the knapsack for items of the particular class. There is a negative profit f_{it} (a cost) associated with setting up knapsack t for items from class i and setting up a knapsack for a class i also consumes d_i units of capacity from the knapsack. There is at most one knapsack that can be setup to accept items of a each class i, but we can setup a knapsack to accept multiple classes of items.

Let x_{ijt} be a binary variable that is one if item j from class i is placed in knapsack t and zero otherwise and let y_{it} be a binary variable that is one if knapsack t is setup to accept items from class i. Then a mathematical model is:

$$\max \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{n_t} c_{ijt}x_{ijt} + \sum_{t=1}^{T}\sum_{i=1}^{N} f_{it}y_{it} \tag{1}$$

s.t.

$$\sum_{i=1}^{N}\sum_{j=1}^{n_t} a_{ij}x_{ijt} + \sum_{i=1}^{N} d_i y_{it} \leq b_t \qquad \forall t \in \{1,\ldots,T\} \tag{2}$$

$$x_{ijt} \leq y_{it} \qquad \forall i \in \{1,\ldots,N\}, j \in \{1,\ldots,n_i\}, t \in \{1,\ldots,T\} \tag{3}$$

$$\sum_{t=1}^{T} y_{it} \leq 1 \qquad \forall i \in \{1,\ldots,N\} \tag{4}$$

$$x_{ijt} \in \{0,1\} \quad \forall i \in \{1,\ldots,N\}, j \in \{1,\ldots,n_i\}, t \in \{1,\ldots,T\} \tag{5}$$

$$y_{it} \in \{0,1\} \quad \forall i \in \{1,\ldots,N\}, t \in \{1,\ldots,T\} \tag{6}$$

The model was originally presented in [1] that also introduced the problem.

A Julia file is supplied that implements the model above. The Julia file also contains a function **readMKPS** that reads the instances used in [1]. The instances can be downloaded from DTU Learn. In this exercise we will work on the subset of the instances listed in Figure 1.

From "instances/NC/":
INS_5_10_1v.dat
INS_5_10_2v.dat
INS_5_10_3v.dat
INS_5_10_4v.dat
INS_5_10_5v.dat
INS_5_10_6v.dat
INS_5_10_7v.dat
INS_5_10_8v.dat
INS_5_10_9v.dat
INS_5_10_10v.dat

From "instances/50/":
INS_10_30_1v.dat
INS_10_30_2v.dat
INS_10_30_3v.dat
INS_10_30_4v.dat
INS_10_30_5v.dat
INS_10_30_6v.dat
INS_10_30_7v.dat
INS_10_30_8v.dat
INS_10_30_9v.dat
INS_10_30_10v.dat

Figure 1: Instances to consider in this report

If the instances have been unzipped in the current folder then it is possible to solve the first instance from the NC folder by typing the command in the Julia REPL:

```
MKPS.test("instances/NC/INS_5_10_1v.dat")
```

The code is set up to use CPLEX as the solver but can easily be changed to use Gurobi instead (just replace CPLEX with Gurobi and notice that a parameter has to be changed in the function setupMKPS(...) - there is a comment in the julia file about this. Results for the instances from the NC folder can be compared to the first 10 rows of table A.8 of [1] while results from the "50" folder can be compared to the row 51-60 in table A.9.

To just read in one of the instances to inspect the data (this will be useful in question C and H) one can use

```
T,N,n,b,c,f,d,a = MKPS.readMKPS("instances/NC/INS_5_10_1v.dat")
```

The values returned correspond to the parameters used in model (1)-(6).

1.1 Dantzig-Wolfe decomposition

We can apply Dantzig Wolfe reformulation to model (1)-(6) by keeping constraints (4) in the master and "convexifying" the remaining constraints.

2 Assignment

Question A Explain the objective and the constraints of model (1)-(6) in your own words.

Question B Solve the instances defined by the data in the table shown in Figure 1, using a MIP solver. Enforce a time limit of 20 minutes for each instance. Report the best upper and lower bound and the time spent by the MIP solver. You should also solve the LP relaxation of model (1)-(6) and report the bound provided by this.

Question C We now move to Dantzig-Wolfe reformulation. We consider a small instance with 2 knapsacks, 2 classes of items that each contains 2 items. This instance is named "mini-instance.txt" and can be found on DTU learn.

Write up the coefficients of the constraint matrix (for model (1)-(6)) that corresponds to this instance. Use a table like the one below. In your table you should indicate what constraint each row of the table corresponds to.

row\variable	x_{111}	x_{112}	x_{121}	x_{122}	x_{211}	x_{212}	x_{221}	x_{222}	y_{11}	y_{12}	y_{21}	y_{22}	RHS	constraint #
1														
2														
⋮														
⋮														

Question D Show that the sub-problem breaks into T independent sub-problems when applying the suggested Dantzig-Wolfe reformulation. It may be necessary to reorder columns and/or rows to better see the structure in the constraint matrix.

Question E Solve the Dantzig-Wolfe relaxation of "mini-instance.txt", using column generation. Use multiple (two) sub-problems (like you explained in part D). The master and sub-problem(s) can be solved by a solver but the column generation process should be done by hand. Explain your approach and report what happens in each iteration of the algorithm.

Question F Now we wish to solve the Dantzig-Wolfe relaxation of the instances from Figure 1 in an automated fashion (using multiple sub-problems). Implement a Julia program to do so. You are advised to use "ColGenGeneric.jl" from lecture 03, but you can also program this from scratch.

- Report the results together with the results from task B in one combined table and compare results. For each instance report if the Dantzig-Wolfe relaxed model resulted in an integer solution or if branching would be necessary.
- Be ready to abort column generation if more than 20 minutes is spent on an instance.
- Explain the changes you did to existing programs in order to solve this task (including screenshots of the changed parts).

Question G Now consider the instance defined in "mini-instance2.txt". Inspect the data in the instance and argue why the decomposition described in the introduction satisfies the "Identical sub-problem assumption" mentioned in lecture 4, part B when applied to this instance.

Question H Show how the master and sub-problem in this case can be simplified taking advantage of the identical sub-problem assumption.

Question I Solve the Dantzig-Wolfe relaxation using column generation based on the simplified models from question I. The master and sub-problem should be solved using a solver while the column generation process should be done by hand.

Question J - Reading academic literature In this question you are going to read one of the following five papers (available on DTU learn):

1. Pigatti, Alexandre, Marcus Poggi De Aragao, and Eduardo Uchoa. "Stabilized branch-and-cut-and-price for the generalized assignment problem." Electron. Notes Discret. Math. 19 (2005): 389-395.
2. Gélinas, Sylvie, and François Soumis. "Dantzig-Wolfe decomposition for job shop scheduling." In Column generation, pp. 271-302. Springer, Boston, MA, 2005.
3. Irnich, Stefan, and Guy Desaulniers. "Shortest path problems with resource constraints." In Column generation, pp. 33-65. Springer, Boston, MA, 2005.
4. Barnhart, Cynthia, Ellis L. Johnson, George L. Nemhauser, Martin WP Savelsbergh, and Pamela H. Vance. "Branch-and-price: Column generation for solving huge integer programs." Operations research 46, no. 3 (1998): 316-329.
5. Mehrotra, Anuj, and Michael A. Trick. "A column generation approach for graph coloring." informs Journal on Computing 8, no. 4 (1996): 344-354.

Use the following algorithm to determine which paper to read:

1. Add the study numbers of the group members together
2. Divide this number by 5
3. Take the remainder of the division and add one. This gives you the paper number that you should study

You can also compute the paper number using this Julia function. Input is the sum of your study numbers

```
function f(x)
  return x % 5 + 1
end
```

Please answer the following three questions after reading the paper:

1. What is the paper about?
2. How is the paper related to the course "42136 Large Scale Optimization using Decomposition" ?
3. What is new in the paper?

The answers to the three question are expected to take up between half a page and one page in total.

References

[1] Rahma Lahyani, Khalil Chebil, Mahdi Khemakhem, and Leandro C Coelho. Matheuristics for solving the multiple knapsack problem with setup. *Computers & Industrial Engineering*, 129:76–89, 2019.