IEOR 4008: Computational Discrete Optimization 03/12/2020

Assignment 3

Deadline: 04/01/2020, h: 1:10pm

Submit electronically via canvas.

Exercise 1

In this exercise, we will write and feed to Gurobi an IP for building an index fund that tracks S&P 500 index. In the same folder containing the assignment file, there is a file *data.txt*. This contains the correlation between the stock options under consideration for the period of 3 years ending on June 30th, 2017. Now open the file *Index.py*. It contains a file that you can feed to Gurobi in a similar way as we did for the *TSP.py* and *TSPplus.py*, but needs to be completed. In particular, constraints (other than binary constraints on variables) are missing.

a) [20pt] Complete the file *Index.py* so that, when fed to Gurobi, it solves the IP for producing an Index Fund seen in class[1]. Note that the usage of this file is *gurobi Index.py n*, where n is the number of stock options you want to be in your index.

b) [5pt] Launch it on Gurobi with n=20, 40, 60. Report the results you obtain and the running time.

Exercise 2

In the *Knapsack problem*, we are given n objects, where object i has weight w_i and profit p_i, and a capacity B, and we want to compute the set of objects of maximum total profit whose total weight does not exceed the capacity. Define by IP_1 the following formulation:

$$\max p_1x_1 + ... + p_nx_n \ : \ w_1x_1 + w_2x_2 + \cdots + w_nx_n \leq B,\ x \in \{0,1\}^n,$$

and by IP_2 the following formulation:

$$\max p_1x_1 + ... + p_nx_n \ : \ \sum_{i \in I} x_i \leq |I| - 1 \text{ for all } I \subseteq [n] : \textstyle\sum_{i \in I} w_i > B, x \in \{0,1\}^n.$$

a) [10pt] Show that IP_1 is a valid IP formulation for the Knapsack problem.

b) [bonus] Show that IP_2 is a valid IP formulation for the Knapsack problem.

Recall the concept of *tighter* formulation seen in class: we say that IP_1 is tighter than IP_2 if the linear relaxation LP_1 of IP_1 is always contained in the linear relaxation LP_2 of IP_2.

c) [5pt] Show that there are knapsack instances where $\text{LP}_2 \subseteq \text{LP}_1$.

d) [5pt] Show that there are knapsack instances where $\text{LP}_1 \subseteq \text{LP}_2$.

e) [5pt] Show that there are knapsack instances where *both* $\text{LP}_2 \nsubseteq \text{LP}_1$ and $\text{LP}_1 \nsubseteq \text{LP}_2$.

Exercise 3

Consider again the most parsimonious tree problem seen in the previous assignment. Suppose now that the following additional restriction hold:

Once a character has been developed, it cannot be undeveloped in future species.

[1] Use files *TSP.py* and *TSPplus.py* for reference.

We call this new problem the *original most parsimonious tree problem* (*OMPTP*). In this exercise, we will write an IP formulation for *OMPTP*. A *directed tree rooted at r* is a directed graph with no directed or undirected cycle, such that, for each vertex v, there is a directed path from r to v. An instance of the *Rooted Steiner Minimum Tree* (*RSMT*) problem is given by a directed graph $D(V, A)$ with weights on the arcs, a node $r \in V$ and a set of terminals $T \subseteq V \setminus \{r\}$. The goal is to find, among all subgraphs of D, a directed tree rooted at r of minimum total weight that contains r and all terminals.

a) [10pt] Argue that the OMPTP can be formulated as a RSMT problem.

b) [7.5pt] Write an IP formulation for the RSMT.

c) [7.5pt] Now suppose we add the constraints that no species can evolve from the modern species (i.e., the ones that are given as input) and that each character can only be developed once. Modify your IP from part b) so that you now satisfy those constraints.