Lecturer: Rekha Thomas

Problem Set 6

409 - Discrete Optimization

due Wednesday Feb 20

Exercise 1
Run the Branch & Bound algorithm to solve the following integer linear program

$$\begin{array}{rcl} \max\ 3x_1 + x_2 & & \\ -2x_1 + 3x_2 & \leq & 6 \\ 10x_1 + 4x_2 & \leq & 27 \\ x_1 & \geq & 0 \\ x_2 & \geq & \frac{1}{2} \\ x_1, x_2 & \in & \mathbb{Z} \end{array}$$

Also give the Branch & Bound tree.

Exercise 2
The Branch & Bound algorithm as we saw it in the lecture uses linear programming as a relaxation to solve integer linear programs. Actually, the arguments behind the algorithm would work in different settings where the "relaxation" is not in form of a linear program.

To keep the notation simple, we want to consider a slight variant of the TSP problem, which is called TSP PATH. For the TSP PATH problem, the input is an undirected graph $G = (V, E)$ (which does not need to be a complete graph) with non-negative edge cost $c : E \to \mathbb{R}_{\geq 0}$. The goal is to compute a path $P \subseteq E$ that visits each node exactly once (we allow any pair from V as end points). Note that this problem is NP-complete. We want to design a variant of the Branch & Bound algorithm to solve it. For a subset of edges $E' \subseteq E$, let $MST(E')$ be the minimum spanning tree in the subgraph (V, E'). Recall that $MST(E')$ can be computed in polynomial time using Kruskal's algorithm. For a set of edges $C \subseteq E$, we denote $c(C) := \sum_{e \in C} c_e$ as the sum of its cost. The algorithm for TSP PATH is now as follows:

(1) Set C^* as being `undefined`

(2) Initialize a stack with $\{E\}$

(3) WHILE stack is non-empty DO

  (4) Take and remove an item (= an edge set) from the stack and call it E'

  (5) If the graph (V, E') is not connected, goto (3)

  (5) Compute $T := MST(E')$

  (8) IF T is a path and $c(T) < c(C^*)$ (or C^* undefined) THEN update $C^* := T$ and goto (3)

  (9) IF T is a path and $c(T) \geq c(C^*)$ THEN goto (3)

(10) Let $v \in V$ be a node that is incident to at least 3 edges in T.

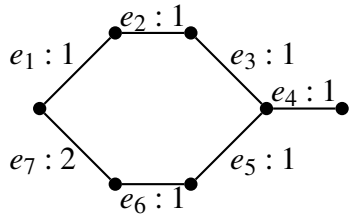
(11) Let $\{e_1, e_2, e_3\} \subseteq \delta(v) \cap T$ be 3 edges that are incident to v in T.

(12) Put the sets $E' \setminus \{e_1\}, E' \setminus \{e_2\}, E' \setminus \{e_3\}$ on the stack

(13) Return C^*

Answer the following:

a) How does the algorithm run on the following instance (edges labelled with $e : c(e)$)



b) Show that in general the algorithm computes an optimum solution to the TSP PATH problem. In particular, if the optimum TSP path is still contained in E', then why is it also contained in at least one of the edge sets that are added back to the stack in (12)? Also, why can we stop searching within E' in step (9)?

c) Argue, why the algorithm runs at most $2 \cdot 3^m$ times through the WHILE loop, if $m = |E|$ is the number of edges in the original graph.
Hint: Give an upper bound of 3^m on the number of leafs in the Branch & Bound tree.