Lecturer: Rekha Thomas

Problem Set 8

409 - Discrete Optimization

Spring 2018

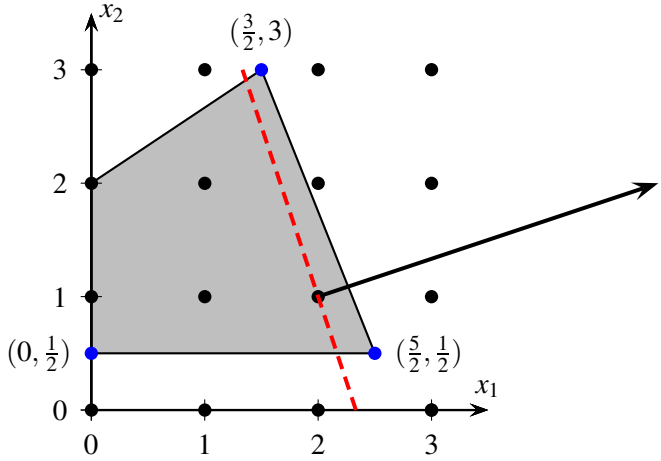
The following problems will not be collected or graded. My plan is to assign one problem per lecture. You will get the most out of the remaining two weeks of classes if you do these problems for the following lecture.

Exercise 1
(assigned 5/21, do for 5/23)
Run the Branch & Bound algorithm to solve the following integer linear program

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

Also give the Branch & Bound tree.

Solution:
The underlying polytope P looks as follows:

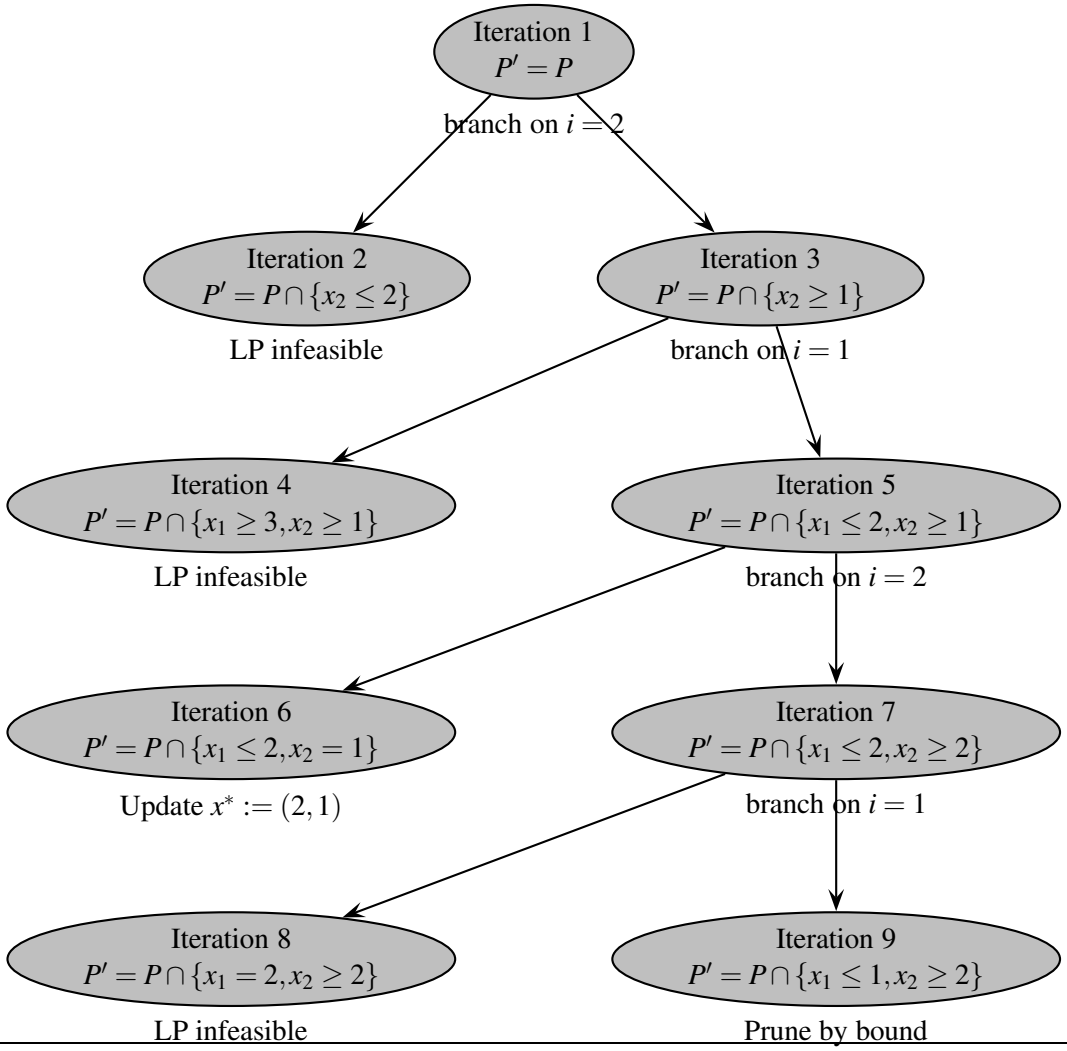


The run of the Branch & Bound algorithm is not unique and depends on the choices of coordinates to branch on and which problem from the stack to work on. One possible outcome is the following (we simplify the constraints if possible, which does not change how the algorithm works):

- **Iteration 1:** Stack: P. $P' = P$. LP solution: $\tilde{x} = (\frac{5}{2}, \frac{1}{2})$. Branch on $i = 2$.
- **Iteration 2:** Stack: $P \cap \{x \mid x_2 \leq 0\}$, $P \cap \{x \mid x_2 \geq 1\}$. $P' = P \cap \{x \mid x_2 \leq 0\}$. LP infeasible. Prune by infeasibility.

- **Iteration 3:** Stack: $P \cap \{x \mid x_2 \geq 1\}$. $P' = P \cap \{x \mid x_2 \geq 1\}$. LP solution $(2.3, 1)$. Branch on $i = 1$.
- **Iteration 4:** Stack: $P \cap \{x \mid x_1 \geq 3, x_2 \geq 1\}$, $P \cap \{x \mid x_1 \leq 2, x_2 \geq 1\}$. $P' = P \cap \{x \mid x_1 \geq 3, x_2 \geq 1\}$. LP is infeasible.
- **Iteration 5:** Stack: $P \cap \{x \mid x_1 \leq 2, x_2 \geq 1\}$. LP solution $\tilde{x} = (2, \frac{7}{4}) = (2, 1.75)$. Branch on $i = 2$.
- **Iteration 6:** Stack: $P \cap \{x \mid x_1 \leq 2, x_2 \geq 1, x_2 \leq 1\} = P \cap \{x \mid x_1 \leq 2, x_2 = 1\}$, $P \cap \{x \mid x_1 \leq 2, x_2 \geq 1, x_2 \geq 2\} = P \cap \{x \mid x_1 \leq 2, x_2 \geq 2\}$. $P' = P \cap \{x \mid x_1 \leq 2, x_2 = 1\}$. LP solution $\tilde{x} = (2, 1)$. Update $x^* := (2, 1)$.
- **Iteration 7:** Stack: $P \cap \{x \mid x_1 \leq 2,\ x_2 \geq 2\}$. LP solution $\tilde{x} = (1.9, 2)$ ($c\tilde{x} > cx^*$). Branch on $i = 1$.
- **Iteration 8:** Stack: $P \cap \{x \mid x_1 \leq 2, x_1 \geq 2, x_2 \geq 2\} = P \cap \{x \mid x_1 = 2, x_2 \geq 2\}$, $P \cap \{x \mid x_1 \leq 2, x_1 \leq 1, x_2 \geq 2\} = P \cap \{x \mid x_1 \leq 1, x_2 \geq 2\}$. $P' = P \cap \{x \mid x_1 = 2, x_2 \geq 2\}$. LP infeasible.
- **Iteration 9:** $P \cap \{x \mid x_1 \leq 1, x_2 \geq 2\}$. LP solution $(1, \frac{8}{3}) \approx (1, 2.6666)$. Now $c\tilde{x} \leq cx^*$ and we can prune by bound

Overall $x^* = (2, 1)$ is optimal.



Exercise 2
(assigned 5/23, do for 5/25)
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.

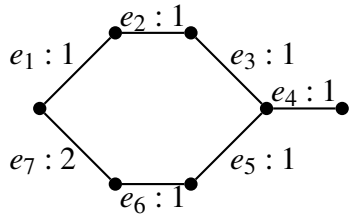
  (11) Let $\{e_1, e_2, e_3\} \subseteq \delta(v) \cap T$ be 3 edges that are incident to v in T.

  (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.

Solution:

a) In the first iteration, the minimum cost spanning tree is $T = E/\{e_7\}$. Then we select the node v with $\delta(v) = \{e_3, e_4, e_5\}$. We branch on those 3 edges and add $E/\{e_3\}, E/\{e_4\}, E/\{e_5\}$ to the stack. In the 2nd iteration we work, sau on E/e_4, realizing that it's not connected. In the 3rd iteration, if we work on $E \setminus \{e_3\}$, then $MST(E/\{e_3\}) = E/\{e_3\}$, which is a TSP path. We update $C^* := E \setminus \{e_3\}$. Finally, we process $E' = E \setminus \{e_5\}$ and have cost of $MST(E')$ coinciding with the cost of C^* which means we would not update C^* and return (in this case) $E \setminus \{e_3\}$.

b) The important observation is that the MST problem is a relaxation of the TSP path problem in the sense that every TSP path is also a spanning tree. In (8), if the minimum spanning tree happens to be a TSP path, then that has to be the optimum TSP path. In (12), we use that any TSP path has degree ≤ 2 at every node, hence one of the edges e_1, e_2, e_3 is not used in the optimum TSP path. The algorithm will eventually terminate since we only add edge sets back to the stack that have fewer edges.

c) Let us consider the Branch & Bound tree. The tree has out degree 3 (in the sense that each node in which we branch has 3 nodes). The depth of the tree (that is the maximum number of edges from the root to a leaf) is at most m since after removing m edges, there is no edge left. So the branch and bound tree has at most 3^m leafs. A tree with 3^m many leafs (and out-degree 3 for all interior nodes) has at most $2 \cdot 3^m$ many nodes (this is certainly not a tight estimate).

Exercise 3
(assigned 5/25, do for 5/30)
Let $G = (V, E)$ be an undirected graph. An *edge cover* in G is a collection of edges $C \subseteq E$ such that every $v \in V$ is incident to at least one edge in C.

1. Model the minimum cardinality edge cover problem in G as an integer program and write it in matrix notation.

2. Prove that when G is bipartite, the minimum cardinality edge cover problem can be solved by its LP relaxation.

3. Write down the dual of the LP relaxation in part 1.

4. If you required the variables to be integral in part 3, which combinatorial optimization does this IP model?

5. What combinatorial statement can you conclude from the pair of primal and dual LPs that you have created?

Exercise 4
(assigned 5/25, do for 5/30)
Use the dynamic programming algorithm to solve the following knapsack problem for $W = 10$.

	1	2	3	4
w_i	5	4	6	3
c_i	10	40	30	50