Score 8

# MATH 170 – PROBLEM SET 8 (DUE TUESDAY MARCH 21)
## SOLUTIONS BY FREDERICK LAW

1. (B&T 10.1) **(Disjunctive Constraints)** Suppose that we are given $m$ constraints $\mathbf{a}_i'\mathbf{x} \geq b_i$, $i = 1, \ldots, m$, but without the restriction $\mathbf{a}_i \geq \mathbf{0}$. Model the requirement that at least $k$ of them are satisfied. Assume that there exists a number $f$ such that $\mathbf{a}_i'\mathbf{x} \geq f$ for $i = 1, \ldots, m$ and for all feasible $\mathbf{x}$.

   **Solution:** To model this, we use the following set up: instead of multiplying $b_i$ by some binary variable, instead we add some factor which is in turn multiplied by a binary variable. Namely, we swap $\mathbf{a}_i'\mathbf{x} \geq b_i$ with $\mathbf{a}_i'\mathbf{x} \geq b_i + (1 - y_i)(f - b_i)$, where $y_i$ is a binary variable which returns 1 if the $i$th constraint is active and 0 otherwise. If we have at least $k$ constraints active, then at least $k$ of the $y_i$ will be 1 which means we are guaranteed that $\mathbf{a}_i'\mathbf{x} \geq b_i$ for these $k$ indices. For the rest we are only guaranteed $\mathbf{a}_j'\mathbf{x} \geq f$ but it could also be the case where these constraints are satisfied as well. Thus we have modeled that at least $k$ of these constraints are satisfied. More succinctly, we have

   $$\mathbf{a}_i'\mathbf{x} \geq b_i + (1 - y_i)(f - b_i)$$

   where $\sum_{i=1}^m y_i \geq k$ and $y_i \in \{0, 1\}$ for $i = 1, \ldots, m$. □

   (1)

2. (B&T 10.5) **(Plan for a move)** Suppose you are planning to move to your new house. You have $n$ items of size $a_j$, $j = 1, \ldots, n$, that need to be moved. You have rented a truck that has size $Q$ and you have bought $m$ boxes. Box $i$ has size $b_i$, $i = 1, \ldots, m$. Formulate an integer programming problem in order to decide whether the move is possible.

   **Solution:** We make some assumptions on the move. Firstly, we assume that we only travel with the truck once, so we are restricted to size $Q$. We also assume that multiple items can be put inside the same box as long as we do not overrun the capacity of the box, and that the items are indivisible. Lastly, we assume the items, boxes, and truck all have positive size. We introduce $nm$ binary variables $x_{ij}$ and $m$ binary variables $y_j$, where $x_{ij}$ returns 1 when item $i$ is placed in box $j$ and $y_j$ returns 1 when box $j$ is brought into the truck. For the move to be possible, we want to put each item in a box, so $\sum_{j=1}^m x_{ij} = 1$ for $i = 1, \ldots, n$. Moreover, we want to make such that each box that is taken with us doesn't exceed capacity, so $\sum_{i=1}^n a_i x_{ij} \leq b_j y_j$.

   We note that these first two constraints actually imply that we never put an item in a box we do not take, which would be bad. This is because if we put item $i$ in box $j$, but do not take box $j$, then $y_j = 0$ but $x_{ij} = 1$ so we get $a_i \leq \sum_{i=1}^n a_i x_{ij} \leq 0$ which is impossible since we assumed all the sizes were positive. Thus we have implicitly encoded the constraint that items must be placed in boxes which are taken into the truck. Lastly, we must ensure we do not break truck capacity $\sum_{k=1}^m b_k y_k \leq Q$. We can formulate this all as an integer program as follows:

   $$\begin{aligned}
   \text{minimize} \quad & 1 \\
   \text{subject to} \quad & \sum_{j=1}^m x_{ij} = 1 \quad , i = 1, \ldots, n \\
   & \sum_{i=1}^n a_i x_{ij} \leq b_j y_j \quad , j = 1, \ldots, m \\
   & \sum_{k=1}^m b_k y_k \leq Q \\
   & x_{ij} \in \{0, 1\} \quad , i = 1, \ldots, n, \text{ and } j = 1, \ldots, m \\
   & y_j \in \{0, 1\} \quad \forall j
   \end{aligned}$$

   where the integer program is feasible if and only if the move is possible. □

   (1)

3. (B&T 10.10) **(Job shop scheduling)** A factory consists of $m$ machines $M_1, \ldots, M_m$, and needs to process $n$ jobs every day. Job $j$ needs to be processed once by each machine in the order $\left(M_{j(1)}, \ldots, M_{j(m)}\right)$. Machine $M_i$ takes time $p_{ij}$ to process job $j$. A machine can only process one job at a time, and once a job is started on any machine, it must be processed to completion. The objective is to minimize the sum of the completion times of all the jobs. Provide an integer programming formulation for this problem.

**Solution:** We construct a mixed integer program (MIP). We assume quite a lot is known about our system. Namely, we assume that we know all the job process times on all the machines $p_{ij}$. Moreover, for every machine $M_k$ we assume that we know a value $z_i^k$ such that $M_k = M_{i(z_i^k)}$ for every job $i$. This just means that for all jobs we know the order of machines on which the job has to be processed, and that we can go backwards from this. So given a machine $M_k$, we know the order of every job that must be processed on this machine. Let $a_{ij}$ be the starting time of job $i$ on machine $M_{i(j)}$, thus $a_{ij}$ represents the $j$th processing step for the total job $i$. The constraint that a job $i$ has to be processed first by $M_{i(1)}$, then $M_{i(2)}$ and so on can be modeled by

$$a_{ij} + p_{i(j)i} \leq a_{i(j+1)} \quad \forall \text{ jobs } i \text{ and } j \in \{1, \ldots, m\}$$

where $p_{i(j)i}$ is the processing time for job $i$ at the $j$th processing step, on machine $M_{i(j)}$. This inequality tells us that the start time for the $j$th step of job $i$ plus the processing time of that step must be less than the starting time for the $(j+1)$th step, which means jobs are all processed sequentially.

Or other constraint is a disjunctive one, which says that no machine can process more than one job at a time, and this is where we make use of the $z_i^k$ that we identified earlier. Let us fix a job $i$ and a machine $M_k$. For any other job $j$, we need either $a_{i,z_i^k} + p_{ki} \leq a_{j,z_j^k}$ or $a_{j,z_j^k} + p_{kj} \leq a_{i,z_i^k}$. This means that machine $M_k$ either processes job $i$'s $z_i^k$th task to completion first, or does job $j$'s first. Note that if we fix machine $M_k$ and then range this over all different jobs $i$ and $j$, this gives us a total ordering for the jobs processed by $M_k$. Then if we range this over all machines $K$, this will give us a some ordering for all jobs on all machines. More succinctly, if we keep this XOR condition for every job, then it follows that no machine will every be processing more than one job at a time, and all jobs get processed to completion on all machines.

In order to model this disjunctive constraint, we employ the tactics of problem (1) in this problem set, by finding some $f$ such that $a_{i,z_i^k} - a_{j,z_j^k} \geq f$ for all $i \neq j$ jobs. Let $C = \sum_{i,j} p_{ij}$ which is the sum over all processing times on all machines. Then $C$ serves as a "bound" on how bad our processing time is, if we only processed one job on one machine at a time. So $a_{i,z_i^k} - a_{j,z_j^k} \geq -C$ for all $i \neq j$ jobs. Thus we introduce binary variables $y_{ij}^k$ such that

$$a_{i,z_i^k} - a_{j,z_j^k} \geq p_{ki} + \left(1 - y_{ij}^k\right)(-C - p_{ki})$$
$$a_{j,z_j^k} - a_{i,z_i^k} \geq p_{kj} + y_{ij}^k(-C - p_{kj})$$

where $y_{ij}^k = 1$ means that for machine $M_k$ that job $i$ gets processed before job $j$, and $y_{ij}^k = 0$ means the opposite. Moreover, we must add the constraint that $y_{ij}^k = 1 - y_{ji}^k$ for any machine $M_k$, since if $M_k$ processes $i$ before $j$ then it cannot also process $j$ before $i$.

Lastly, in our formulation of our problem, the total time spent processing is $\max_{i,j} a_{ij} + p_{i(j)i}$, the maximum of the start times for all jobs $i$ and tasks $i(j)$ plus the processing time for that

task for that job. So our MIP is

$$\begin{aligned}
\text{minimize} \quad & \max_{i,j} a_{ij} + p_{i(j)i} \\
\text{subject to} \quad & a_{ij} + p_{i(j)i} \leq a_{i(j+1)} \quad \forall \text{ jobs } i \text{ and } j \in \{1, \ldots, m\} \\
& a_{i,z_i^k} - a_{j,z_j^k} \geq p_{ki} + \left(1 - y_{ij}^k\right)(-C - p_{ki}) \quad \forall \text{ jobs } i \neq j \text{ and all machines } M_k \\
& a_{j,z_j^k} - a_{i,z_i^k} \geq p_{kj} + y_{ij}^k(-C - p_{kj}) \quad \forall \text{ jobs } i \neq j \text{ and all machines } M_k \\
& y_{ij}^k = 1 - y_{ji}^k \quad \forall \text{ jobs } i \neq j \text{ and all machines } M_k \\
& a_{ij} \geq 0 \quad \forall \text{ jobs } i \text{ and } j \in \{1, \ldots, m\} \\
& y_{ij}^k \in \{0, 1\} \quad \forall \text{ jobs } i \neq j \text{ and all machines } M_k
\end{aligned}$$

□

4. (B&T 10.12) Let $G = (\mathcal{N}, \mathcal{E})$ be an undirected graph with $n$ nodes. Show that $G$ is a tree if and only if the total number of edges is $n-1$, and for any nonempty set $S \subset \mathcal{N}$, the number of edges with both endpoints in $S$ is less than or equal to $|S| - 1$.

   **Solution:** Recall that the definition of a tree is a connected graph (undirected) with no cycles. Suppose that $G$ is a tree. Since $G$ has no cycles, then we get that for any $S \subset \mathcal{N}$ nonempty, we must have $|\mathcal{E}(S)| \leq |S| - 1$. Otherwise, if there is some subset $S$ so that $|\mathcal{E}(S)| \geq |S|$ then the graph restricted to the node set $S$ and edge set $\mathcal{E}(S)$ must contain a cycle which is impossible since $G$ is a tree. By the same logic, $|\mathcal{E}|$ must have less than $n$ edges, otherwise $G$ contains a cycle. But $G$ is connected, which means that there must exist a path between every two nodes in the graph, so $G$ must have $n-1$ edges. If there were less than $n-1$ edges, then there would be some node which we could not connect to the rest of the graph.

   Suppose that $G$ has $n-1$ edges and for any nonempty $S \subset \mathcal{N}$ then $|\mathcal{E}(S)| \leq |S| - 1$. We wish to prove that $G$ is both connected and contains no cycles. By the same logic as before, $G$ cannot contain cycles, since if there was a cycle in $G$, then we would have a collection of nodes $S$ where we have at least $|S|$ edges with both end points in this subcollection, which is impossible by our hypothesis. Thus it remains to prove that $G$ is connected. Suppose that $G$ is not connected. Then we can divide $G$ into disjoint subgraphs $G_1 = (\mathcal{N}_1, \mathcal{E}_1)$ and $G_2 = (\mathcal{N}_2, \mathcal{E}_2)$ where $\mathcal{N}_1$ and $\mathcal{N}_2$ partition $\mathcal{N}$ and likewise $\mathcal{E}_1$ and $\mathcal{E}_2$ partition $\mathcal{E}$. Moreover, these subgraphs are non-empty and both are unions of connected components. Since these are partitions, we know that $|\mathcal{N}_1| + |\mathcal{N}_2| = n$ and $|\mathcal{E}_1| + |\mathcal{E}_2| = n - 1$. But then there must exist some $i$ such that $|\mathcal{N}_i| \leq |\mathcal{E}_i|$. If this were not true, and $|\mathcal{E}_i| < |\mathcal{N}_i|$ for $i = 1, 2$, then we could write

$$n - 1 - |\mathcal{E}_1| < n - |\mathcal{N}_1| \implies |\mathcal{N}_1| < |\mathcal{E}_1| + 1 \implies |\mathcal{N}_1| \leq |\mathcal{E}_1|$$

   since our cardinalities are non-negative integer valued, and if one integer is strictly less than another it is also weakly less than the other minus 1. But we also have $|\mathcal{E}_1| < |\mathcal{N}_1|$, so combining inequalities we get $|\mathcal{E}_1| < |\mathcal{E}_1|$ which is impossible. So there exists some $i$ such that $|\mathcal{N}_i| \leq |\mathcal{E}_i|$ and WLOG suppose it is $i = 1$. Now we run into trouble, since $\mathcal{N}_1 \subset \mathcal{N}$ nonempty, so by hypothesis we must have $|\mathcal{E}(\mathcal{N}_1)| \leq |\mathcal{N}_1| - 1$. But $\mathcal{E}(\mathcal{N}_1) = \mathcal{E}_1$, so we have both $|\mathcal{E}_1| \leq |\mathcal{N}_1| - 1$ and $|\mathcal{N}_1| \leq |\mathcal{E}_1|$ which means $|\mathcal{N}_1| \leq |\mathcal{N}_1| - 1$ which is an impossibility. Therefore, $G$ must be connected, and so $G$ is a tree. □

5. (B&T 10.14) **(The undirected traveling salesman problem)** For the undirected traveling salesman problem, prove that

$$P_{\text{tspcut}} = P_{\text{tspsub}}$$

   **Solution:** In order to prove this, we first derive a useful equality. Consider $S \subset \mathcal{N}$ where $S \neq \varnothing, \mathcal{N}$. We know that the cut set is $\delta(S) = \{\{i, j\} \in \mathcal{E} : i \in S, j \notin S\}$ and the edge set is

$\mathcal{E}(S) = \{\{i,j\} \in \mathcal{E} : i \in S, j \in S\}$. So what happens if we take the cut set of all the singletons in the subset of nodes $S$? Well each edge in the cut set will get cut once, but every edge in the edge set will get cut twice, once on both sides. More precisely, if $e \in \delta(S)$ then there is a single node $i \in S$ such that $e \in \delta(\{i\})$, namely the node in $S$ which $e$ is attached to. But if $e \in \mathcal{E}(S)$ then there are exactly two nodes $i, j \in S$ such that $e \in \delta(\{i\}) \cap \delta(\{j\})$, namely the endpoints of the edge $e$. So if we have fixed edge weights, and we compute the sum of weights on the cut sets of the singletons in $S$, we get the weight of the cut set of $S$, but we have double counted the weight of the edge set of $S$. Therefore:

$$\sum_{e \in \delta(S)} x_e + 2 \sum_{e \in \mathcal{E}(S)} x_e = \sum_{j \in S} \sum_{e \in \delta(\{j\})} x_e$$

for any way to associate weights $x_e$ with edges $e \in \mathcal{E}$.

Now for our problem. Recall that

$$P_{\text{tspcut}} = \begin{cases} \sum_{e \in \delta(\{i\})} x_e = 2, \forall i \in \mathcal{N} \\ \sum_{e \in \delta(S)} x_e \geq 2, \forall S \subset \mathcal{N}, S \neq \varnothing \\ 0 \leq x_e \leq 1, \forall e \in \mathcal{E} \end{cases}, \; P_{\text{tspcut}} = \begin{cases} \sum_{e \in \delta(\{i\})} x_e = 2, \forall i \in \mathcal{N} \\ \sum_{e \in \mathcal{E}(S)} x_e \leq |S| - 1, \forall S \subset \mathcal{N}, S \neq \varnothing \\ 0 \leq x_e \leq 1, \forall e \in \mathcal{E} \end{cases}$$

where both these polyhedra live in Euclidean space of dimension $|\mathcal{N}|$. To prove $P_{\text{tspcut}} \subseteq P_{\text{tspsub}}$, suppose $\mathbf{x} \in P_{\text{tspcut}}$. Then $\sum_{e \in \delta(\{i\})} x_e = 2$ for all $i \in \mathcal{N}$ and for any nonempty subset $S \subset \mathcal{N}$, we have $\sum_{e \in \delta(S)} x_e \geq 2$. Using our equality from above, we have

$$2 + 2 \sum_{e \in \mathcal{E}(S)} x_e \leq \sum_{e \in \delta(S)} x_e + 2 \sum_{e \in \mathcal{E}(S)} x_e = \sum_{j \in S} \sum_{e \in \delta(\{j\})} x_e = 2|S|$$

Thus subtracting 2 from both sides and dividing by 2, it follows that $\sum_{e \in \mathcal{E}(S)} x_e \leq |S| - 1$, so $\mathbf{x} \in P_{\text{tspsub}}$ and $P_{\text{tspcut}} \subseteq P_{\text{tspsub}}$. To prove the opposite direction, suppose $\mathbf{x} \in P_{\text{tspsub}}$. Then $\sum_{e \in \delta(\{i\})} x_e = 2$ for all $i \in \mathcal{N}$ and for any nonempty subset $S \subset \mathcal{N}$, we have $\sum_{e \in \mathcal{E}(S)} x_e \leq |S| - 1$. Using our equality from above, we have

$$2|S| - \sum_{e \in \delta(S)} x_e = \sum_{j \in S} \sum_{e \in \delta(\{j\})} x_e - \sum_{e \in \delta(S)} x_e = 2 \sum_{e \in \mathcal{E}(S)} x_e \leq 2|S| - 2$$

Canceling $2|S|$ on both sides and then multiplying both sides by $-1$, we flip the inequality and get $\sum_{e \in \delta(S)} x_e \geq 2$. Therefore $\mathbf{x} \in P_{\text{tspcut}}$ and so $P_{\text{tspsub}} \subseteq P_{\text{tspcut}}$. Thus $P_{\text{tspcut}} = P_{\text{tspsub}}$. □

6. (B&T 11.1) Consider the integer programming problem:

$$\begin{aligned} \text{maximize} \quad & x_1 + 2x_2 \\ \text{subject to} \quad & -3x_1 + 4x_2 \leq 4 \\ & 3x_1 + 2x_2 \leq 11 \\ & 2x_1 - x_2 \leq 5 \\ & x_1, x_2 \geq 0, \text{ integer} \end{aligned}$$

Use a figure to answer the following questions. Figure is attached at the end of the problem set.

**Solution:** Much of this problem involved drawing graphs and solving graphically, so this solution was hand written. It is attached at the end of the typed problems.

7. (B&T 11.2) The goal of this exercise is to compare the optimal costs of an integer programming problem and its linear programming relaxation. Consider the integer programming problem

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}'\mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \geq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \\ & \mathbf{x} \text{ integer} \end{aligned}$$

where the entries of $\mathbf{A}$, $\mathbf{b}$, and $\mathbf{c}$ are integer, and its linear programming relaxation

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}'\mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \geq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned}$$

(a) Assume that the integer programming problem is feasible. Show that if the linear programming relaxation has optimal cost equal to $-\infty$, then the integer programming problem has optimal cost $-\infty$ as well.

**Solution:** Recall that by Theorem 4.14 in Bertsimas and Tsitsiklis[1] that an $LP$ has $-\infty$ optimal cost if and only if there is an extreme ray of the feasible region which has negative cost. Thus if our linear programming relaxation has optimal cost $-\infty$, then it follows that there is some extreme ray, an element of the recession cone of the feasible region, $\mathbf{d}$ such that $\mathbf{A}\mathbf{d} \geq \mathbf{0}$, $\mathbf{d} \geq \mathbf{0}$, and $\mathbf{c}'\mathbf{d} < 0$. If $\mathbf{d}$ has rational entries, then we win. Otherwise we perturb $\mathbf{d}$ by adding some $\mathbf{e}$ such that $e_i$ is 0 when $d_i$ is rational, and is otherwise chosen to make $d_i + e_i$ rational.Essentially, we perturb all of the irrational coordinates in a direction which keeps us in the recession cone, turns all irrational coordinates to rational ones, and the perturbations are chosen small enough so that $\mathbf{c}'(\mathbf{d}+\mathbf{e}) < 0$. Once the direction is chosen, we can choose a sufficiently small $\mathbf{e}$ due since $\mathbb{Q}$ is dense in $\mathbb{R}$, so we can keep cost negative, stay in the recession cone, and perturb the irrational coordinates to be rational.

Once we have done the above perturbation, we now have $\mathbf{d}+\mathbf{e}$ in the recession cone with rational coordinates such that $\mathbf{A}(\mathbf{d}+\mathbf{e}) \geq \mathbf{0}$, $\mathbf{d}+\mathbf{e} \geq \mathbf{0}$, and $\mathbf{c}'(\mathbf{d}+\mathbf{e}) < 0$. Assume that the coordinates of $\mathbf{d}+\mathbf{e}$ are in lowest terms, and choose $\theta > 0$ to be a multiple of all the denominators of the coordinates. Then $\theta(\mathbf{d}+\mathbf{e})$ is also in the recession cone, has negative cost, and now has integer coordinates. Since the IP is feasible, we choose some feasible integer valued $\mathbf{y}$. Then for any $n \in \mathbb{N}$ we have $\mathbf{y} + n\theta(\mathbf{d}+\mathbf{e}) \geq \mathbf{0}$, $\mathbf{A}(\mathbf{y}+n\theta(\mathbf{d}+\mathbf{e})) \geq \mathbf{b}$, $\mathbf{c}'(\mathbf{y}+n\theta(\mathbf{d}+\mathbf{e})) = \mathbf{c}'\mathbf{y} + n\theta\mathbf{c}'(\mathbf{d}+\mathbf{e})$ and most importantly $\mathbf{y}+n\theta(\mathbf{d}+\mathbf{e})$ has integer coordinates. Since $\mathbf{c}'(\mathbf{d}+\mathbf{e}) < 0$, and $\mathbf{y}+n\theta(\mathbf{d}+\mathbf{e})$ is feasible for the IP for any $n \in \mathbb{N}$, we can choose arbitrarily large $n$ for arbitrarily small cost, so the IP has optimal cost $-\infty$. □

(b) Is it true that there always exists an $a > 0$, such that $Z_{IP} \leq aZ_{LP}$?

**Solution:** No. We provide a counter example. Consider the integer program

$$\begin{aligned} \text{minimize} \quad & y \\ \text{subject to} \quad & -2x + 3y \geq -3 \\ & 2x + 3y \geq 3 \\ & -4x - 3y \geq -6 \\ & x, y \geq 0 \\ & x, y \text{ integer} \end{aligned}$$

There are only two feasible solutions to this $IP$: $(0,1)$ and $(0,2)$. The IP is then minimized at $(0,1)$ with $Z_{IP} = 1$. If we look at the linear programming relaxation, then the LP relaxation is minimized at $(1.5, 0)$, so $Z_{LP} = 0$. Thus there does not exist an $a > 0$ such that $Z_{IP} \leq aZ_{LP}$, since if there did exist such an $a > 0$ then we would have $1 \leq a * 0 = 0$ which is impossible. □

8. (B&T 11.3) **(Cuts for mixed integer programming problems)** Let

$$T = \left\{ (\mathbf{x}, \mathbf{y}) \,:\, \sum_{j \in N} a_j x_j + \sum_{j \in J} d_j y_j \leq b, \mathbf{x} \text{ integer}, \mathbf{x}, \mathbf{y} \geq \mathbf{0} \right\},$$

where $N = \{1, \ldots, n\}$ and $J = \{1, \ldots, p\}$. Show that the inequality

$$\sum_{j \in N} \lfloor a_j \rfloor x_j + \frac{1}{1 - b + \lfloor b \rfloor} \sum_{j \in J^-} d_j y_j \leq \lfloor b \rfloor$$

is satisfied by all points of $T$, where $J^- = \{j \in J \,:\, d_j < 0\}$. *Hint:* Consider separately the cases

$$\sum_{j \in J} d_j y_j > b - \lfloor b \rfloor - 1, \quad \text{and} \quad \sum_{j \in J} d_j y_j \leq b - \lfloor b \rfloor - 1$$

**Solution:** We consider cases as suggested by the hint. First consider the case where $(\mathbf{x}, \mathbf{y}) \in T$ and $\sum_{j \in J} d_j y_j > b - \lfloor b \rfloor - 1$. Then

$$\sum_{j \in N} a_j x_j + b - \lfloor b \rfloor - 1 < \sum_{j \in N} a_j x_j + \sum_{j \in J} d_j y_j \leq b \Longrightarrow \sum_{j \in N} a_j x_j < \lfloor b \rfloor + 1 = \lceil b \rceil$$

Then $\sum_{j \in N} \lfloor a_j \rfloor x_j < \lceil b \rceil$, where now both the left and right sides are integers, since $x_j$ is integer. But if we have strict inequality between integers, we can relax to a weak inequality after subtracting 1 from the larger integer. So $\sum_{j \in N} \lfloor a_j \rfloor x_j \leq \lceil b \rceil - 1 = \lfloor b \rfloor$. Moreover, since $\frac{1}{1 - b + \lfloor b \rfloor} \geq 1$ and $\sum_{j \in J^-} d_j y_j \leq 0$, because $d_j < 0$ for $j \in J^-$ and $\mathbf{y} \geq 0$, then we have

$$\sum_{j \in N} \lfloor a_j \rfloor x_j + \frac{1}{1 - b + \lfloor b \rfloor} \sum_{j \in J^-} d_j y_j \leq \sum_{j \in N} \lfloor a_j \rfloor x_j \leq \lfloor b \rfloor$$

Next we consider the case where $(\mathbf{x}, \mathbf{y}) \in T$ and $\sum_{j \in J} d_j y_j \leq b - \lfloor b \rfloor - 1$. Since $b - \lfloor b \rfloor - 1 < 0$, then we can express this as

$$\sum_{j \in J^-} d_j y_j \leq \sum_{j \in J} d_j y_j \leq -1 \left(1 - b - \lfloor b \rfloor\right) \Longrightarrow \frac{1}{1 - b - \lfloor b \rfloor} \sum_{j \in J^-} d_j y_j \leq -1$$

Next, note that since $(\mathbf{x}, \mathbf{y}) \in T$, we have

$$\begin{aligned} \sum_{j \in N} \lfloor a_j \rfloor x_j + \frac{1}{1 - b + \lfloor b \rfloor} \sum_{j \in J^-} d_j y_j + \frac{-b + \lfloor b \rfloor}{1 - b + \lfloor b \rfloor} \sum_{j \in J^-} d_j y_j &= \sum_{j \in N} \lfloor a_j \rfloor x_j + \sum_{j \in J^-} d_j y_j \\ &\leq \sum_{j \in N} a_j x_j + \sum_{j \in J} d_j y_j \leq b \end{aligned}$$

thus we have

$$\begin{aligned} \sum_{j \in N} \lfloor a_j \rfloor x_j + \frac{1}{1 - b + \lfloor b \rfloor} \sum_{j \in J^-} d_j y_j &\leq b + (b - \lfloor b \rfloor) \left( \frac{1}{1 - b + \lfloor b \rfloor} \sum_{j \in J^-} d_j y_j \right) \\ &\leq b + (b - \lfloor b \rfloor)(-1) = \lfloor b \rfloor \end{aligned}$$

and the inequality holds. Since we have proved both cases, we have proved the validity of the inequality when $(\mathbf{x}, \mathbf{y}) \in T$. □

# References


[1] Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific and Dynamic Ideas, LLC. Belmont, Massachusetts, 1997.

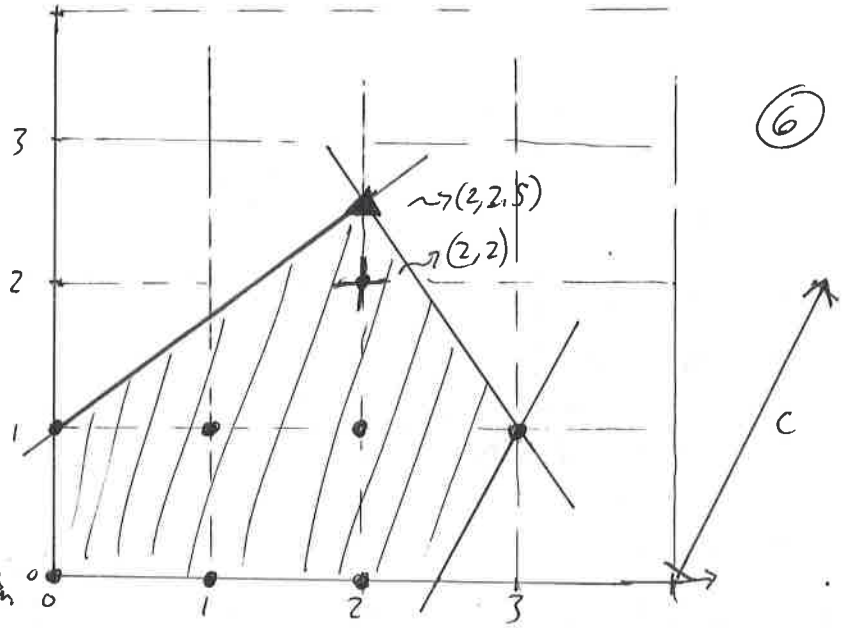
~~(6) B&T 11.1~~

# (6) B&T 11.1

(a.) The • points are the IP feasible region. The shaded /// region is the feasible set to the LP relaxation.

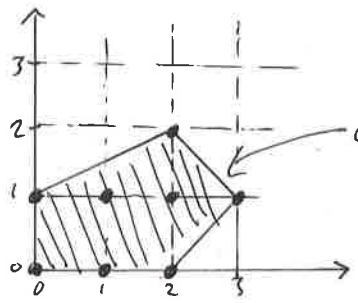
▲ represents the optimal solution to the LP relaxation

✚ represents the optimal solution to the IP.




- LP relaxation optimal at (2, 2.5)
  ⇒ LP optimal cost 7
- IP optimal at (2,2)
  ⇒ IP optimal cost 6

(b)

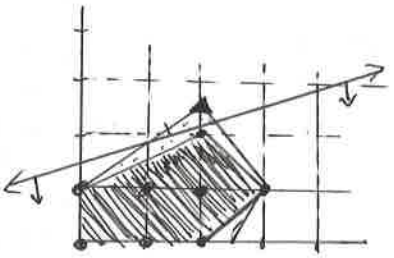



where T is the set of IP feasible solutions.
defined by $x_2 \le \frac{1}{2}x_1 + 1$, $x_2 \ge x_1 - 2$, $x_2 \le -x_1 + 4$

$$\Rightarrow \text{conv}(T) = \left\{ (x_1, x_2) : \begin{array}{l} -x_1 + 2x_2 \le 2 \\ x_1 - x_2 \le 2 \\ x_1 + x_2 \le 4 \end{array}, \begin{array}{l} x_1 \ge 0 \\ x_2 \ge 0 \end{array} \right\}$$

(c) For the Gomory cut, we would want to cut our feasible ~~region~~ set to the LP relaxation without cutting conv(T).

More succinctly, we cut between the the convex hull conv(T) and the optimal solution to the LP relaxation.



example of what a Gomory cut could look like.

we convert the LP relaxation to standard form:

$\min c^T x$
s.t. $Ax = b$, $x \ge 0$

where

$$c = \begin{pmatrix} -1 \\ -2 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad A = \begin{pmatrix} -3 & 4 & 1 & 0 & 0 \\ 3 & 2 & 0 & 1 & 0 \\ 2 & -1 & 0 & 0 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 4 \\ 11 \\ 5 \end{pmatrix}$$

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \} \text{slack variables.}$$

We run simplex and get the optimal tableau (thanks to R + MatLAB)

| | 7 | 0 | 0 | 2/9 | 5/9 | 0 |
|---|---|---|---|---|---|---|
| $x_1 \rightsquigarrow$ | 2 | 1 | 0 | -1/9 | 2/9 | 0 |
| $x_2 \rightsquigarrow$ | 2.5 | 0 | 1 | 1/6 | 1/6 | 0 |
| $x_5 \rightsquigarrow$ | 3.5 | 0 | 0 | 7/18 | -5/18 | 1 |

→

(c) cont.

Our optimal simplex tableau for the LP relaxation is

| | 7 | 0 | 0 | 2/9 | 5/9 | 0 |
|---|---|---|---|---|---|---|
| $x_1 \rightsquigarrow$ | 2 | 1 | 0 | -1/9 | 2/9 | 0 |
| $x_2 \rightsquigarrow$ | 2.5 | 0 | 1 | 1/6 | 1/6 | 0 |
| $x_3 \rightsquigarrow$ | 3.5 | 0 | 0 | 7/18 | -5/18 | 1 |

where $\overline{a_{ij}} = (B^{-1}A_j)_i$ (ith row, jth column)

$\overline{a_{i0}} = (B^{-1}b)_i$ (ith row, 0th column).

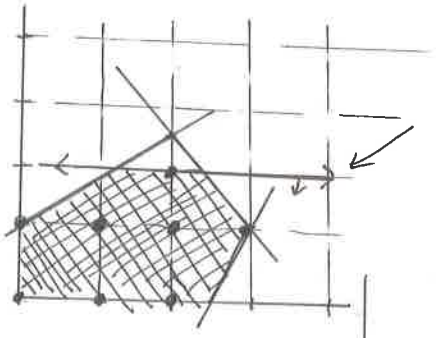
- choose $i=2$ since $\overline{a_{20}} = 2.5$ is not integer

$\Rightarrow x_2 + \frac{1}{6}x_3 + \frac{1}{6}x_4 = \frac{5}{2}$

$\Rightarrow$ our Gomory cut is introducing $x_2 + \lfloor \frac{1}{6} \rfloor x_3 + \lfloor \frac{1}{6} \rfloor x_4 \leq \lfloor \frac{5}{2} \rfloor$

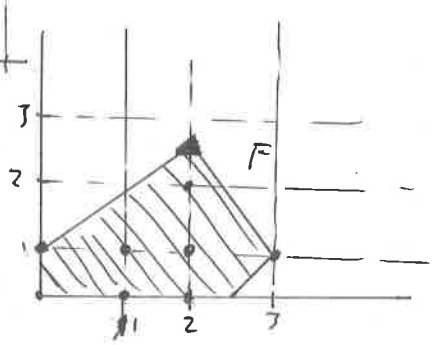
which is just $x_2 \leq 2$.

Shaded region is the LP relaxation feasible after the first Gomory cut $x_2 \leq 2$



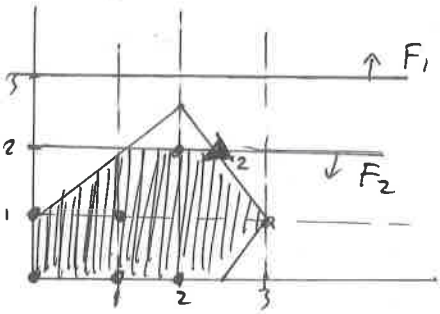

(d). For branch & bound, we initialize with $U = \infty$, and start with the LP relaxation as F

- note that at each step, $\blacktriangle_i$ indicates the optimal solution to the ith problem $F_i$.
- A is the list of active subproblems

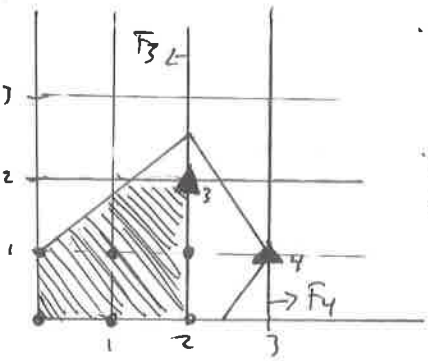



start!
- optimal solution to ~~the~~ initial LP relaxation at $\blacktriangle = (2, 2.5)$
  $b(F) = 7$
- branch on 2nd coord
  - $F_1$ w/ $x_2 \geq 3$
  - $F_2$ w/ $x_2 \leq 2$
- $A = \{F_1, F_2\}$
  $U = \infty$
- $F_1$ infeasible, remove from A
- $F_2$ optimal at $\blacktriangle_2 = (\frac{7}{3}, 2)$
  $b(F_2) = \frac{7}{3} + 4$
- branch $F_2$ into
  - $F_3$ w/ $x_1 \geq 3$
  - $F_4$ w/ $x_1 \leq 2$
- $A = \{F_3, F_4\}$, $U = \infty \to 6$
- $F_3$ optimal at $\blacktriangle_3 = (2, 2)$, IP feasible
  $b(F_3) = 6$, $U = 6$
  remove $F_3$
- $F_4$ optimal at $\blacktriangle_4 = (3, 1)$. IP feasible
  $b(F_4) = 5 \leq U$
  remove $F_4$

- F branches into $F_1, F_2$ (remove F)




- $F_1$ infeasible (remove $F_1$)
- $F_2$ branches into $F_3, F_4$ (remove $F_2$)




- $F_3$ gives feasible solution, update (remove $F_3$)
- $F_4$ gives feasible solution, not as good as U (remove $F_4$)

(terminate)

- $A = \{\}$, $U = 6$

since A is empty, terminate.

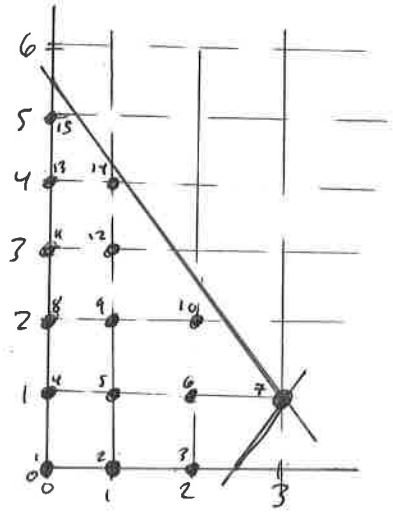
$\Rightarrow$ optimal solution to IP is (2,2) w/ cost $U = 6$.

(e) $\max\ x_1+2x_2$
s.t. $-3x_1+4x_2 \le 4$
$3x_1+2x_2 \le 11$
$2x_1-x_2 \le 5$
$x_1, x_2 \ge 0$
$x_1, x_2$ integer

dualize → $-3x_1+4x_2 \le 4$

$\max\ x_1+2x_2+p(4+3x_1-4x_2)$
s.t. $3x_1+2x_2 \le 11$
$2x_1-x_2 \le 5$
$x_1, x_2 \ge 0$
$x_1, x_2$ integer.

our feasible set looks now like → each integer point is some $(x_1^{(i)}, x_2^{(i)})$, $i=1,\dots,15$.



now 15 feasible integer solutions.

$$Z(p) = \max_i \left( x_1^{(i)} + 2x_2^{(i)} + p\left(4 + 3x_1^{(i)} - 4x_2^{(i)}\right)\right)$$

over $i=1,\dots,15$, and $Z_D = \min_{p\ge 0} Z(p)$.

- Using R and MatLAB, and GeoGebra, we can reduce this after plugging in all 15 points:

$$Z(p) = \begin{cases} -16p+10, & 0 \le p \le \frac{1}{7} \\ -9p+9, & \frac{1}{7} \le p \le \frac{2}{9} \\ 9p+5, & \frac{2}{9} \le p \le 3 \\ 10p+2, & p \ge 3. \end{cases}$$

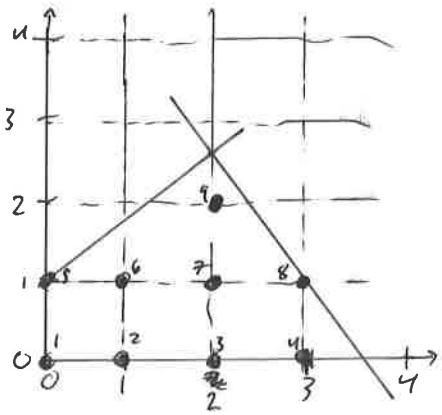
This is minimized at $p=\frac{2}{9} \Longrightarrow$ $\boxed{Z_D = Z\left(\frac{2}{9}\right) = 7}$

(f) $\max\ x_1+2x_2$
s.t. $-3x_1+4x_2 \le 4$
$3x_1+2x_2 \le 11$
$2x_1-x_2 \le 5$
$x_1, x_2 \ge 0$
$x_1, x_2$ integer

dualize → $2x_1-x_2 \le 5$

$\max\ x_1+2x_2+p(5-2x_1+x_2)$
s.t. $-3x_1+4x_2 \le 4$
$3x_1+2x_2 \le 11$
$x_1, x_2 \ge 0$
$x_1, x_2$ integer

new feasible set looks like → now we have 9 integer points $(x_1^{(i)}, x_2^{(i)})$



$$Z(p) = \max_i \left( x_1^{(i)} + 2x_2^{(i)} + p\left(5 - 2x_1^{(i)} + x_2^{(i)}\right)\right)$$

- Again, use R, MatLAB, and GeoGebra to plug in these 9 points and reduce:

$$Z(p) = \begin{cases} 3p+6, & 0 \le p \le \frac{4}{3} \\ 6p+2, & p \ge \frac{4}{3} \end{cases}$$

This is minimized at $p=0 \Longrightarrow$ $\boxed{Z_D = Z(0) = 6}$ □