ChE 597 Computational Optimization

Homework 6 Solutions

1. Convert the logic expression below into a system of inequalities with $0,1$ variables:

   (a)
   $$(P_1 \vee \neg P_2) \Rightarrow (P_3 \vee P_4)$$

   (b)
   $$\Big((P_1 \wedge P_2) \Rightarrow P_3\Big) \Rightarrow \Big(P_5 \vee P_6\Big)$$

Solution: (a)
$$(P_1 \vee \neg P_2) \Rightarrow (P_3 \vee P_4)$$

To turn the given logical expression into a set of inequalities using binary variables, we'll first transform the logic expression into Conjunctive Normal Form (CNF). We'll do this by following three steps we learned in our lecture slides

(1) Remove implications using the fact that $(\mathrm{A} \Rightarrow \mathrm{B}) \equiv (\neg \mathrm{A} \vee \mathrm{B})$

$$(P_1 \vee \neg P_2) \Rightarrow (P_3 \vee P_4) \equiv \neg(P_1 \vee \neg P_2) \vee (P_3 \vee P_4)$$

(2) Move negations inwards by applying De Morgan's theorem.

$$\neg(P_1 \vee \neg P_2) \vee (P_3 \vee P_4) \equiv (\neg P_1 \wedge P_2) \vee (P_3 \vee P_4)$$

(3) Recursively distribute OR over AND

$$(\neg P_1 \wedge P_2) \vee (P_3 \vee P_4) \equiv (\neg P_1 \vee P_3 \vee P_4) \wedge (P_2 \vee P_3 \vee P_4)$$

Now, we have the equation in CNF. Let's say we have binary variables y_1, y_2, y_3, y_4 corresponding to P_1, P_2, P_3, P_4 respectively. This means if P_1 is true, then $y_1 = 1$, and if P_1 is false, then $y_1 = 0$. We can set up constraints for the CNF logic expression like this:

$$1 - y_1 + y_3 + y_4 \geq 1$$
$$y_2 + y_3 + y_4 \geq 1$$

This above set of inequalities represents the logic expression we started with in the question.

(b)
$$\Big((P_1 \wedge P_2) \Rightarrow P_3\Big) \Rightarrow \Big(P_5 \vee P_6\Big)$$

We'll use the same steps we used for the previous problem. First, we'll convert the logic expression into Conjunctive Normal Form (CNF) in the following way:

(1) Remove implications using the fact that (A $\Rightarrow$ B) $\equiv$ ($\neg$A $\vee$ B)

$$\Big((P_1 \wedge P_2) \Rightarrow P_3\Big) \Rightarrow \Big(P_5 \vee P_6\Big) \equiv \Big(\neg(P_1 \wedge P_2) \vee P_3\Big) \Rightarrow \Big(P_5 \vee P_6\Big)$$
$$\Big(\neg(P_1 \wedge P_2) \vee P_3\Big) \Rightarrow \Big(P_5 \vee P_6\Big) \equiv \neg\Big(\neg(P_1 \wedge P_2) \vee P_3\Big) \vee \Big(P_5 \vee P_6\Big)$$

(2) Move negations inwards by applying De Morgan's theorem.

$$\neg\Big(\neg(P_1 \wedge P_2) \vee P_3\Big) \vee \Big(P_5 \vee P_6\Big) \equiv \neg\Big(\neg P_1 \vee \neg P_2 \vee P_3\Big) \vee \Big(P_5 \vee P_6\Big)$$
$$\neg\Big(\neg P_1 \vee \neg P_2 \vee P_3\Big) \vee \Big(P_5 \vee P_6\Big) \equiv \Big(P_1 \wedge P_2 \wedge \neg P_3\Big) \vee \Big(P_5 \vee P_6\Big)$$

(3) Recursively distribute OR over AND

$$(P_1 \wedge P_2 \wedge \neg P_3) \vee (P_5 \vee P_6) \equiv (P_1 \vee P_5 \vee P_6) \wedge (P_2 \vee P_5 \vee P_6) \wedge (\neg P_3 \vee P_5 \vee P_6)$$

Now, we have the equation in CNF. Let's say we have binary variables y_1, y_2, y_3, y_5, y_6 corresponding to P_1, P_2, P_3, P_5, P_6 respectively. This means if P_1 is true, then $y_1 = 1$, and if P_1 is false, then $y_1 = 0$. We can set up constraints for the CNF logic expression like this:

$$y_1 + y_5 + y_6 \geq 1$$
$$y_2 + y_5 + y_6 \geq 1$$
$$1 - y_3 + y_5 + y_6 \geq 1$$

This above set of inequalities represents the logic expression we started with in the question.

2. Formulate linear constraints in terms of binary variables for the following cases:

   (a) If A is true and B is true then C is true or D is true. (inclusive OR)

   (b) The choice of all 0-1 combinations for $y_j, j \in J$ is feasible, except the one for which $y_j = 0, j \in N, y_j = 1, j \in B$, where N and B are specified partitions of J.

   (c) If power must be generated in any time period 1,2 or 3, then install a gas turbine. Represent whether power is generated at the three time periods as three separate binary variables.

Solution: (a) If A is true and B is true then C is true or D is true. (inclusive OR)

The above statement is a **if-then** statement that can be mathematically expressed as follows:

$$(A \wedge B) \Rightarrow (C \vee D)$$

We can remove the implication sign as follows

$$(A \wedge B) \Rightarrow (C \vee D) \implies \neg(A \wedge B) \vee (C \vee D)$$

Using De Morgan's rule and taking the negation inside, we have

$$\neg(A \wedge B) \vee (C \vee D) \implies (\neg A \vee \neg B) \vee (C \vee D) \implies (\neg A \vee \neg B \vee C \vee D)$$

Considering that the binary variables y_A, y_B, y_C, y_D correspond to A, B, C, D respectively, we can write the constraint as follows:

$$1 - y_A + 1 - y_B + y_C + y_D \geq 1$$

(b) The choice of all 0-1 combinations for $y_j, j \in J$ is feasible, except the one for which $y_j = 0, j \in N, y_j = 1, j \in B$, where N and B are specified partitions of J.

According to the above statement, we want to consider all the combinations except the one in which $y_j = 0, j \in N, y_j = 1, j \in B$, where N and B are specified partitions of J. We can express this statement mathematically as follows:

$$\neg\Big(((y_j = 0) \wedge (j \in N)) \wedge ((y_j = 1) \wedge (j \in N))\Big)$$

The above logic expression is about negating the combination where $y_j = 0, j \in N, y_j = 1, j \in B$.
Considering $(y_j = 0) \equiv \neg y_j$ and $(y_j = 1) \equiv y_j$, we can rewrite the above logic expression as follows:

$$\neg\Big((\wedge_{j \in N} \neg y_j) \wedge (\wedge_{j \in B} y_j)\Big)$$

Using De Morgan's therorem, taking the negation inside, we have

$$\Big((\vee_{j \in N} y_j) \vee (\vee_{j \in B} \neg y_j)\Big)$$

The above logic expression is in CNF, hence it's corresponding linear constraint can be written as follows:

$$\sum_{j \in N} y_j + \sum_{j \in B} (1 - y_j) \geq 1$$

(c) If power must be generated in any time period 1,2 or 3, then install a gas turbine. Represent whether power is generated at the three time periods as three separate binary variables.

The above statement is a **if-then** statement that can be mathematically expressed as follows:

$$(T_1 \vee T_2 \vee T_3) \Rightarrow GT$$

Here, GT corresponds to gas turbine

Let's convert the logic expression into CNF by performing the following steps.

(1) Removing the implication sign and rewriting the logic expression, we have

$$\neg(T_1 \vee T_2 \vee T_3) \vee GT$$

(2) Using De Morgan's rule and taking the negation inside, we have

$$(\neg T_1 \wedge \neg T_2 \wedge \neg T_3) \vee GT$$

(3) Recursively distributing OR over AND, we have

$$(\neg T_1 \wedge \neg T_2 \wedge \neg T_3) \vee GT \implies (\neg T_1 \vee GT) \wedge (\neg T_2 \vee GT) \wedge (\neg T_3 \vee GT)$$

The above logic expression is in Conjunctive Normal Form (CNF). Now, let's introduce three binary variables: y_1, y_2, and y_3, which correspond to T_1, T_2, and T_3 respectively. If power needs to be generated during time period 1 (meaning T_1 is true), then y_1 equals 1. If no power needs to be generated during time period 1 (meaning T_1 is false), then y_1 equals 0. Additionally, let's introduce another binary variable, y_{GT}, corresponding to GT, which indicates whether the gas turbine is to be installed or not. Now, the constraints corresponding to the above CNF logic expression can be written as follows:

$$1 - y_1 + y_{GT} \geq 1$$
$$1 - y_2 + y_{GT} \geq 1$$
$$1 - y_3 + y_{GT} \geq 1$$

3. Formulate mixed-integer linear constraints for the following disjunctions, using both big-M and convex-hull formulations:

   (a) Either $0 \leq x \leq 10$ or $20 \leq x \leq 30$

   (b) The temperature approach constraint for a heat exchanger

   $$T_{in} - t_{out} \geq DTmin$$

   should only hold only if the exchanger is actually selected.

Solution: (a) Either $0 \leq x \leq 10$ or $20 \leq x \leq 30$

(1) Big-M formulation

We can rewrite the above disjunction mathematically as follows as follows:

$$(0 \leq x \leq 10) \veebar (20 \leq x \leq 30)$$

Since we use the regular $\vee$ instead of $\veebar$ in a disjunction, as it's obvious it represents an exclusive-or, we can rewrite the above statement as follows:

$$(0 \leq x \leq 10) \vee (20 \leq x \leq 30) \equiv \Big((-x \leq 0) \wedge (x \leq 10)\Big) \vee \Big((-x \leq -20) \wedge (x \leq 30)\Big)$$

Let y_1, y_2 be the binary variables for the two disjuncts we have in the above expression such that $y_j = 1$ if the linear inequalities in the disjunct j holds true, and $y_j = 0$ otherwise. We can write the Big-M constraints as follows:

$$\begin{aligned}
-x &\leq 0 + M_1(1 - y_1) \\
x &\leq 10 + M_2(1 - y_1) \\
-x &\leq -20 + M_3(1 - y_2) \\
x &\leq 30 + M_4(1 - y_2) \\
y_1 + y_2 &= 1 \\
y_1, y_2 &\in \{0, 1\}
\end{aligned}$$

Now, we find sufficiently large values of the Big-M parameters that give us the tightest fit in the following way. Consider the first inequlity constraint in the above set. This Big-M inequality constraint corresponds to the first disjunct. Now, when the first disjunct does not hold true, this inequality should satisfy the other disjunct, since that would hold true.
Therefore, we need our first constraint to satisfy the following inequality in that case:

$$(-x \leq -20)$$

For this inequality to hold true, we need $M_1 = -20$. Similarly, we can find that the other tightest Big-M parameter values are $M_2 = 20$, $M_3 = 20$, and $M_4 = -20$. Substituting these values in the above Big-M inequality constraints, we have:

$$-x \leq 0 - 20(1 - y_1)$$
$$x \leq 10 + 20(1 - y_1)$$
$$-x \leq -20 + 20(1 - y_2)$$
$$x \leq 30 - 20(1 - y_2)$$
$$y_1 + y_2 = 1$$
$$y_1, y_2 = \{0, 1\}$$

Now, the above set of linear constraints is sufficient to represent the disjunction we had in the first place, but we can reduce this system with two binary variables to one by replacing $y_2 = 1 - y_1$ in the above constraints set.

$$-x \leq 0 - 20(1 - y_1)$$
$$x \leq 10 + 20(1 - y_1)$$
$$-x \leq -20 + 20y_1 = 0 - 20(1 - y_1)$$
$$x \leq 30 - 20y_1 = 10 + 20(1 - y_1)$$
$$y_1 = \{0, 1\}$$

Here, we see we have two inequality constraints repeated twice, therefore, we can take one set off since they are redundant. Finally, we have

$$-x \leq 0 - 20(1 - y_1)$$
$$x \leq 10 + 20(1 - y_1)$$
$$y_1 = \{0, 1\}$$

The above constraint set suffices to represent the disjunction.

(2) Convex Hull formulation

We have the disjunction as follows:

$$(0 \leq x \leq 10) \vee (20 \leq x \leq 30)$$

Let us consider a binary variable y such that if $y = 1$, then the left disjunct is true, else if $y = 0$ then the right disjunct is true.

Let us consider $x = z_1 + z_2$,

Now, from the above disjunction, we can bound z_1 and z_2 as follows

$$0 * y_1 \leq z_1 \leq 10 y_1$$

$$20(1 - y_1) \leq z_2 \leq 30(1 - y_1)$$

Finally, we can write the convex hull formulation for the above disjunct as follows:

$$x = z_1 + z_2$$

$$0 \leq z_1 \leq 10y_1$$

$$20(1-y_1) \leq z_2 \leq 30(1-y_1)$$

$$y \in \{0,1\}$$

$$z_1, z_2 \in \mathbb{R}$$

(b) The temperature approach constraint for a heat exchanger

$$\mathrm{T_{in}} - \mathrm{t_{out}} \geq \mathrm{DTmin}$$

should only hold only if the exchanger is actually selected.

(1) Big-M formulation

Considering that $T_{\text{in}} - t_{\text{out}} \geq 0$, we can rewrite the problem statement as the following disjunction:

$$(T_{\text{in}} - t_{\text{out}} \geq DT_{\text{min}}) \vee (T_{\text{in}} - t_{\text{out}} < DT_{\text{min}})$$

We can get rid of the strict inequality on the right disjunct by subtracting its RHS with a small number $\varepsilon > 0$. Rewriting the above disjunction, we have

$$(T_{\text{in}} - t_{\text{out}} \geq DT_{\text{min}}) \vee (T_{\text{in}} - t_{\text{out}} \leq DT_{\text{min}} - \varepsilon)$$

Now, let us consider $x = T_{\text{in}} - t_{\text{out}}$, the disjunction becomes

$$(x \geq DT_{\text{min}}) \vee (x \leq DT_{\text{min}} - \varepsilon)$$

Let's use a binary variable y_{HE} to represent whether the heat exchanger is selected or not. If y_{HE} is equal to 1, it means the left disjunct (heat exchanger is selected) is true, and if y_{HE} is equal to 0, it means the right option (heat exchanger is not selected) is true.

Now, we can write the Big-M constraints as:

$$x \geq DT_{\text{min}} + M_1(1 - y_{HE})$$

$$x \leq DT_{\text{min}} - \varepsilon + M_2 y_{HE}$$

To begin with, our assumption that $T_{\text{in}} - t_{\text{out}} \geq 0$ necessitates that $x \geq 0$. Consequently, setting $M_1 = -DT_{\text{min}}$ serves as an appropriate approximation to satisfy the initial inequality condition when $y_{HE} = 0$. Likewise, given $x = T_{\text{in}} - t_{\text{out}} \geq 0$ which implies that $x \leq T_{\text{in}}$, it follows that $M_2 = T_{\text{in}} + \varepsilon - DT_{\text{min}}$ is required to fulfill the second inequality constraint for cases where $y_{HE} = 1$. By incorporating these determined values, we can express the Big-M formulation as:

$$x \geq DT_{\text{min}} - DT_{\text{min}}(1 - y_{HE})$$

$$x \leq DT_{\text{min}} - \varepsilon + (T_{\text{in}} + \varepsilon - DT_{\text{min}}) y_{HE}$$

$$y_{HE} \in \{0,1\}$$

$$x \in \mathbb{R}$$

(2) Convex Hull formulation

Considering that $T_{\text{in}} - t_{\text{out}} \geq 0$, we can rewrite the problem statement as the following disjunction:

$$(T_{\text{in}} - t_{\text{out}} \geq DT_{\text{min}}) \vee (T_{\text{in}} - t_{\text{out}} < DT_{\text{min}})$$

We can get rid of the strict inequality on the right disjunct by subtracting its RHS with a small number $\varepsilon > 0$. Rewriting the above disjunction, we have

$$(T_{\text{in}} - t_{\text{out}} \geq DT_{\text{min}}) \vee (T_{\text{in}} - t_{\text{out}} \leq DT_{\text{min}} - \varepsilon)$$

Now, let us consider $x = T_{\text{in}} - t_{\text{out}}$, the disjunction becomes

$$(x \geq DT_{\text{min}}) \vee (x \leq DT_{\text{min}} - \varepsilon)$$

Let's use a binary variable y_{HE} to represent whether the heat exchanger is selected or not. If y_{HE} is equal to 1, it means the left disjunct (heat exchanger is selected) is true, and if y_{HE} is equal to 0, it means the right option (heat exchanger is not selected) is true.

Let x be equal to the sum of two variables z_1 and z_2. Now, we can write the convex hull formulation as:

$$x = z_1 + z_2$$

$$z_1 \geq DT_{\text{min}} y_{HE}$$

$$z_2 \leq (DT_{\text{min}} - \varepsilon)(1 - y_{HT})$$

$$0 \leq z_1 \leq T_{\text{in}} y_{HE}$$

$$0 \leq z_2 \leq T_{\text{in}}(1 - y_{HE})$$

$$y_{HE} \in \{0,1\}$$

$$z_1, z_2 \in \mathbb{R}$$

4. Consider the following knapsack sets represented by mixed-integer linear constraints

$$K := \left\{x \in \{0,1\}^3 : 2x_1 + 3x_2 + 4x_3 \leq 5\right\}.$$

(a) Reformulate the knapsack problem using the minimal cover inequalities.

(b) Denote the feasible region of the minimal cover inequalities as K^C. Show that $K = K^C$

(c) Denote the LP relaxation of K and K^C as P and P^C, respectively. Show $P^C \subsetneq P$

Solution: (a) Reformulate the knapsack problem using the minimal cover inequalities.

Given a standard formulation:

$$K := \left\{x \in \{0,1\}^n : \sum_{i=1}^{n} a_i x_i \leq b\right\}$$

We know that the minimal cover formulation can be written as follows:

$$K^C := \left\{x \in \{0,1\}^n : \sum_{i \in C} x_i \leq |C| - 1 \text{ for every minimal cover } C \text{ for } K\right\}$$

Here, subset C of indices is a cover for K if $\sum_{i \in C} a_i > b$ and it is a **minimal cover** if $\sum_{i \in C \setminus \{j\}} a_i \leq b$ for every $j \in C$. That is, C is a cover if the knapsack cannot contain all items in C, and it is minimal if every proper subset of C can be loaded.

Now, let's look at our problem. We notice that we have two minimal cover subsets: $C_1 = \{1,3\}$ and $C_2 = \{2,3\}$. In C_1, there are two indices, 1 and 3, which form a minimal cover. This means that either 1 or 3 can be chosen, but not both at the same time. So, C_1 fits the definition of a minimal cover. The same reasoning applies to C_2.

To sum up, the reformulated Knapsack problem using minimal cover inequalities can be expressed as follows: [insert the reformulated problem here].

$$K^C := \left\{x \in \{0,1\}^3 : \begin{Bmatrix} x_1 + x_3 \leq 1 \\ x_2 + x_3 \leq 1 \end{Bmatrix}\right\}$$

(b) Denote the feasible region of the minimal cover inequalities as K^C. Show that $K = K^C$.

To show that $K = K^C$, we have to prove that every feasible solution in K^C also is also a feasible solution to K, and vice versa, i.e., in other words both K^C and K have the same feasible solution set

Now, for K^C, the constraint set is as follows

$$x_1 + x_3 \leq 1$$
$$x_2 + x_3 \leq 1$$
$$x \in \{0,1\}^3$$

Given this constraint set, the feasible solutions are:

$$(x_1, x_2, x_3) = \begin{cases} (1,0,0) \\ (0,1,0) \\ (0,0,1) \\ (1,1,0) \\ (0,0,0) \end{cases}$$

Similarly, the constraint set for K is as follows:

$$2x_1 + 3x_2 + 4x_3 \leq 5$$
$$x \in \{0,1\}^3$$

And the feasible solution set for K is as follows:

$$(x_1, x_2, x_3) = \begin{cases} (1,0,0) \\ (0,1,0) \\ (0,0,1) \\ (1,1,0) \\ (0,0,0) \end{cases}$$

Here, we see that the feasible solution set for both regions K and K^C are the same, hence we can conclude that $K = K^C$.

(c) Denote the LP relaxation of K and K^C as P and P^C, respectively. Show $P^C \subsetneq P$.

We can write the LP relaxations of K and K^C as follows:

$$P := \left\{ x \in [0,1]^3 : 2x_1 + 3x_2 + 4x_3 \leq 5 \right\}.$$
$$P^C := \left\{ x \in [0,1]^3 : \begin{cases} x_1 + x_3 \leq 1 \\ x_2 + x_3 \leq 1 \end{cases} \right\}$$

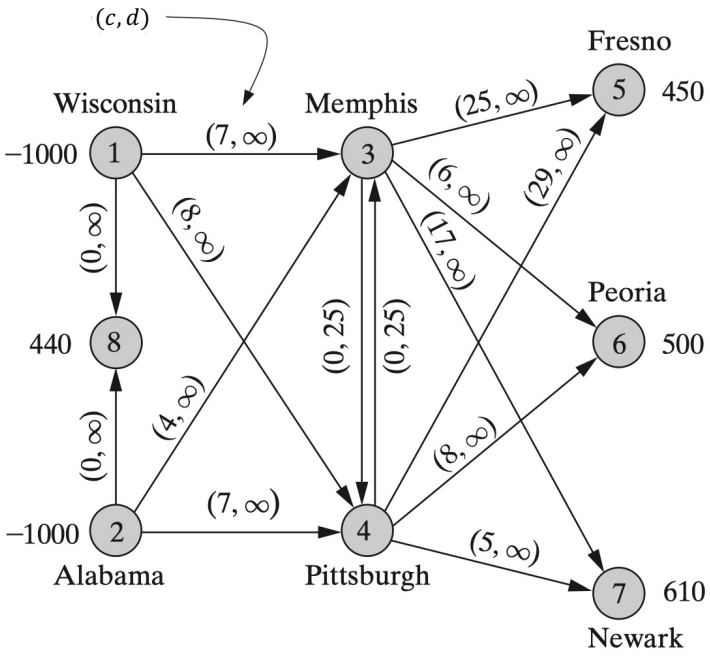
To demonstrate that $P^C \subsetneq P$, we need to show that every point in the feasible region of P^C is also in the feasible region of P, but not every point in the feasible region of P is in the feasible region of P^C.

Let's start by selecting a point in the feasible region of P^C that doesn't belong to the feasible region of P. Upon inspection, we find that every point in the feasible region of P^C is indeed within the feasible region of P.

Now, let's consider the opposite scenario. We'll choose a point in the feasible region of P that isn't within the feasible region of P^C. It's clear that the point $(0, 1/3, 1)$ lies within the feasible region of P but doesn't fall within the feasible region of P^C.

Thus, we can conclude that $P^C \subsetneq P$.

5. *Minimum Cost Network Flows:* Consider the following Minimum Cost Network Flows problem. The demand of each node are shown next to the node, e.g., the demand at Wisconsin is -1000, meaning that we need to send out 1000 units of flow from Wisconsin. The cost and capacity of each edge are also shown.



(a) Formulate the Minimum Cost Network Flows in pyomo and solve it with Gurobi using the primal simplex algorithm. Check if the solution is integral, explain why.

Hint: to use Gurobi's primal simplex algorithm the following option should be specified.

```
opt = SolverFactory('gurobi')
opt.options['Method'] = 0
opt.solve(model, tee=True)
```

(b) Solve the model using the barrier's algorithm without crossover, check if the solution is integral, explain why.

Hint: to use Gurobi's barrier algorithm without crossover, the following solver option should be used.

```
opt = SolverFactory('gurobi')
opt.options['Method'] = 2
opt.options['Crossover'] = 0
opt.solve(model, tee=True)
```

(c) Read the Gurobi user manual to understand these solver options.

LP methods: `https://www.gurobi.com/documentation/current/refman/method.html`

Crossover: `https://www.gurobi.com/documentation/current/refman/crossover.html`

Solution: (a) Formulate the Minimum Cost Network Flows in Pyomo and solve it with Gurobi using the primal simplex algorithm. Check if the solution is integral, explain why.

The LP formulation for the minimum cost network flow problem is as follows:

$$
\begin{array}{lll}
\text{minimize} & \sum_{(i,j)\in E} C_{i,j}x_{i,j} & \\
\text{subject to} & \sum_{j\in N_i^+} x_{i,j} = D_i & \forall i \in N_s \\
& \sum_{i\in N_j^-} x_{i,j} = D_j & \forall j \in N_d \\
& \sum_{i\in N_j^-} x_{i,j} = \sum_{k\in N_j^+} x_{j,k} & \forall j \in N \setminus (N_s \cup N_d) \\
\\
& x_{i,j} \le Cap_{i,j} & \forall (i,j) \in E \\
& x_{i,j} \ge 0 & \forall (i,j) \in E
\end{array}
$$

In the above formulation, $C_{i,j}$ and $X_{i,j}$ represent the cost and flow variables associated with edge $\{(i,j) \in E\}$. The objective of the problem is to minimize the total cost of the flow through the network.

In the first constraint, N_i^+ is the set of all nodes connected to node i with outward-directed edges, ensuring that the flow out of each node matches its supply.

Similarly, in the second constraint, N_j^- is the set of all nodes connected to node j with inward-directed edges, ensuring that the flow into each node matches its demand.

The third constraint represents the flow balance for all intermediate nodes.

The fourth constraint limits the capacity of each edge, while the last constraint ensures that the flow through each edge is non-negative.

The Pyomo code for this Minimum Cost Network Flow problem can be found here:

Running the code, we see that the solution obtained by Gurobi solver with the primal simplex algorithm is integral. When solving linear programming problems with integral coefficients, such as network flow problems, the choice of solution method can impact the integrality of the solution. The primal simplex algorithm maintains the integrality property by iteratively moving from one basic feasible solution to another along the edges of the feasible region. As a result, if the problem satisfies certain conditions, including having integer coefficients and all constraints being satisfied by integer values of the decision variables in every basic feasible solution, the primal simplex algorithm guarantees an optimal basic feasible solution with integer values for all decision variables. Thus, when using the primal simplex algorithm, the solution to network flow problems with integral coefficients is inherently integral.

(b) Solve the model using the barrier's algorithm without crossover, check if the solution is integral, explain why.

Running the code, we see that the solution obtained by Gurobi solver with barrier algorithm is not integral. This is because the barrier (or interior-point) algorithm does not inherently preserve integrality during its iterations. Unlike the simplex method, which moves along the edges of the feasible region, the barrier method traverses through the interior of the feasible region and does not directly enforce integrality of the solution. Consequently, the barrier algorithm may produce solutions that are not necessarily integral for linear programming problems with integer coefficients.