ChE 597 Computational Optimization

Homework 7

March 8th 11:59 pm

1. A company is considering to produce a chemical C which can be manufactured with either process II or process III, both of which use as raw material chemical B. B can be purchased from another company or else manufactured with process I which uses A as a raw material. The superstructure of the problem is show below

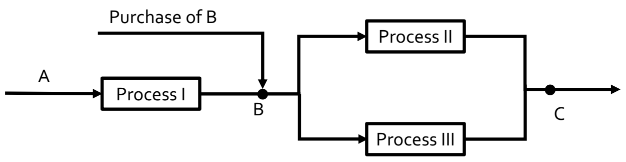


Figure 1: Superstructure

Given the specifications below, formulate an MILP model and solve it with Pyomo to decide:

(a) Which process to build (II and III are exclusive)?

(b) How to obtain chemical B?

(c) How much should be produced of product C? The objective is to maximize profit.

Consider the two following cases:

1. Maximum demand of C is 10 tons/hr with a selling price of $1800/ton.

2. Maximum demand of C is 15 tons/hr; the selling price for the first 10 ton/hr is $1800/ton, and $1500/ton for the excess.

Data:

	Fixed the variable costs	
	Fixed ($/hr)	Variable ($/ton raw mat)
Process I	1000	250
Process II	1500	400
Process III	2000	550

Prices:

- A: $500/ton
- B: $950/ton

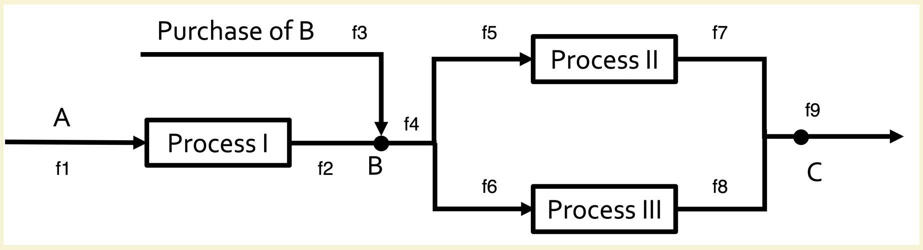
Conversions:

- Process I: 90% of A to B
- Process II: 82% of B to C
- Process III: 95% of B to C

Maximum supply of A: 16 tons/hr

NOTE: You may want to scale your cost coefficients (e.g., divide them by 100).

Solution: Code available at:
597HW7Q1



Label each stream and create variable f as shown above.

Create binary variable y to indicate the presence of process I, II and III.

The objective function is to take into account revenue, raw material cost, variable cost and fixed cost, maximizing the profit.

We have the following constraints:

Maximum demand of C and maximum supply of A

Mass balance at the places where chemical reaction, mixing or splitting happens.

Big-M constraints for f_1, f_5, f_6. Use the information of maximum supply, maximum demand and conversions to make M as small as possible.

XOR constraint - process II and process III are exclusive.

Case 1: solve the model above. (a) I and II; (b) from process I; (c) 10 tons/hr

Case 2: Partition f_9 into two flows: $f_9^1 \leq 10$ and $f_9^2 \leq 5$ with selling price \$1800 and \$1500.

In the objective function, calculate the revenue using f_9^1 and f_9^2 instead of f_9. Because the selling price of f_9^1 is higher than that of f_9^2, it will automatically choose f_9^1 to reach the maximum before increasing f_9^2.

Solve the model again. (a) I and III; (b) from process I; (c) 13.68 tons/hr

2. Solve the following lot sizing problem using pyomo

	Period, t					
	1	**2**	**3**	**4**	**5**	**6**
Demand, d_t	10	40	20	5	5	15
Fixed cost, f_t	50	50	50	50	50	50
Production cost, p_t	1	3	3	1	1	1
Holding cost, h_t	2	2	2	2	2	2

Assume the production limit $C = 25$

Solution: Code available at:
597HW7Q2

Create parameters d_t, f_t, p_t, h_t and C using the table.

Create variables y_t (Amount produced in period t), s_t (Stock at the end of period t) and x_t (occurence of production in period t).

Objective: minimize total production, storage, and fixed costs.

Mass balance at different time period $s_{t-1} + y_t = d_t + s_t$. Note that $s_0 = 0$.

Production limit $y_t \leq Cx_t$.

Solve the model. The minimal cost is 465.

3. Consider the following scheduling problem with release and due dates. Note there are two sets of durations. All the data can be accessible directly from this jupyter notebook `https://github.com/li-group/ChE-597-Computational-Optimization/blob/main/HW%207/HW7%20Q3.ipynb`

   (a) Solve problem set 1 in pyomo. What is the assignment of jobs to machines; what about the precedence of jobs and the total cost?

   (b) Solve problem set 2 in pyomo. What is the assignment of jobs to machines; what about the precedence of jobs and the total cost?

Table 1: Data

Order (i)	r_i	d_i	Cost on Machine c_{im}		
			1	2	3
1	2	16	10	6	8
2	3	13	8	5	6
3	4	21	12	7	10
4	5	28	10	6	8
5	10	24	8	5	7
6	1	28	12	7	10
7	2	23	12	7	10

Order (i)	Machine	Durations (p_{im})	
		Set 1	Set 2
1	1	10	5
	2	14	7
	3	12	6
2	1	6	3
	2	8	4
	3	7	3
3	1	11	2
	2	16	4
	3	13	3
4	1	6	3
	2	12	6
	3	8	4
5	1	10	2
	2	16	4
	3	12	3
6	1	7	1
	2	12	3
	3	10	2
7	1	10	1
	2	8	2
	3	10	1

Solution: We consider order as a task and machine as a unit. Let I be the set of tasks and M be the set of units. The variables for the problem are as follows

- $x_{im} = \begin{cases} 1 & \text{if task } i \in I \text{ is assigned to unit } m \in M \\ 0 & \text{otherwise} \end{cases}$
- $ts_i =$ start time of task $i \in I$
- $y_{ii'} = 1$ if task i before task i' on given unit; 0 otherwise for $i, i' \in I$

The objective is to minimize the cost of processing tasks:

$$\min \sum_{i \in I} \sum_{m \in M} c_{im} x_{im}$$

- **Earliest Start:** Start times must respect the earliest start constraints.
  $ts_i \geq r_i$ for all $i \in I$.
- **Latest Start:** Tasks must start early enough to finish before their deadlines.
  $ts_i \leq d_i - \sum_{m \in M} p_{im} x_{im}$ for all $i \in I$.
- **Assign to One Unit:** Each task is assigned to exactly one unit.
  $\sum_{m \in M} x_{im} = 1$ for all $i \in I$.
- **Redundant Constraint:** This ensures processing times are within a feasible range. This constraint tighten the LP relaxation.
  $\sum_{i \in I} p_{im} x_{im} \leq \max\{d_i\} - \min\{r_i\}$ for all $m \in M$.
- **Precedence constraint:** If x_{im} AND $x_{i'm}$ true then $y_{ii'}$ OR $y_{i'i}$ are true.
  $y_{ii'} + y_{i'i} \geq x_{im} + x_{i'm} - 1$ for all i, i' in I, $i > i'$, m in M
- **Sequencing constraint:** If $y_{ii'} = 1$ then $ts_{i'} \geq ts_i + p_{im}$.
  $ts_{i'} \geq ts_i + \sum_{m \in M} p_{im} x_{im} - M(1 - y_{ii'})$ for all i, i' in I, $i \neq i'$.

The above constraints ensure that the precedence of jobs and assignment of jobs to machines. However, the precedence of the jobs might be degenerate and might not be with respect to each unit. To make the precedence variables to be with respect to each unit and to reduce the degeneracy the following logic cut (constraints) can be added:

- **Logical cut:** If i and i' are assigned to different units, then $y_{i,i'} = y_{i',i} = 0$ $y_{ii'} + y_{i'i} + x_{im} + x_{i'm'} \leq 2$ for all i, i' in I, $i > i'$, m, m' in $M, m \neq m'$

We implement this model in pyomo with and without the logical cut. Code available at: 597HW7Q3

(a) Problem set a : Without the adding the logic cut, the following is the solution. The total cost is 60.

Table 2: Assignment variables

Order/Task (i)	Assignemnt x_{im}		
	1	2	3
1	1	0	0
2	0	1	0
3	0	0	1
4	1	0	0
5	1	0	0
6	0	0	1
7	0	1	0

Table 3: Precedence variables

Order/Task (i)	Precedence $y_{i,i'}$						
	1	2	3	4	5	6	7
1	N/A	0	0	1	1	0	0
2	0	N/A	0	0	0	0	1
3	0	0	N/A	0	0	1	0
4	0	0	0	N/A	0	0	0
5	0	0	0	1	N/A	0	0
6	0	0	0	0	0	N/A	0
7	0	0	0	0	0	0	N/A

In this case, the precedence variables are zero for tasks of different units

(b) Problem set 2 : Without the adding the logic cut, the following is the solution. The total cost is 44.

Table 4: Assignment variables

Order/Task (i)	Assignemnt x_{im}		
	1	2	3
1	0	1	0
2	0	0	1
3	0	1	0
4	0	1	0
5	0	1	0
6	0	1	0
7	0	1	0

Table 5: Precedence variables

Order/Task (i)	Precedence $y_{i,i'}$						
	1	2	3	4	5	6	7
1	N/A	0	1	1	1	0	0
2	1	N/A	1	1	1	0	0
3	0	0	N/A	1	1	0	0
4	0	0	0	N/A	0	0	0
5	0	0	0	1	N/A	0	0
6	1	1	1	1	1	N/A	1
7	1	0	1	1	1	0	N/A

Note that in this case, the precedence variables are non-zero for tasks of different units. For example, some of the precedence variables involving task 2 which is in a different unit from others are non-zero We add the logic cut the following and obtain the following solution :

Table 6: Assignment variables

Order/Task (i)	Assignemnt x_{im}		
	1	2	3
1	0	1	0
2	0	0	1
3	0	1	0
4	0	1	0
5	0	1	0
6	0	1	0
7	0	1	0

Table 7: Precedence variables

Order/Task (i)	Precedence $y_{i,i'}$						
	1	2	3	4	5	6	7
1	N/A	0	1	1	1	1	1
2	0	N/A	0	0	0	0	0
3	0	0	N/A	1	1	0	0
4	0	0	0	N/A	0	0	0
5	0	0	0	1	N/A	0	0
6	0	0	1	1	1	N/A	0
7	0	0	1	1	1	1	N/A

By adding the cut, we ensure that the precedence variables are non-zero only for tasks in the same unit. The cost remains the same.

4. Implement your branch and bound algorithm to solve the scheduling problem in 3. You can use Gurobi to solve the relaxation at each node. To obtain an upper bound, you can use the

rounding heuristic, i.e., round the binary variables to the nearest integer and solve the rest of the LP. Feel free to design your own heuristic based on your understanding of the problem. For node selection rule, use the best bound first rule.

(a) Implement a branch and bound algorithm with the "most fractional variable branching rule". Compare it with the solution you got from 3. How many branch and bound nodes are there in total for problem set 1 and 2, respectively?

(b) Implement a branch and bound algorithm with the "strong branching". Compare it with the solution you got from 3. How many branch and bound nodes are there in total for problem set 1 and 2, respectively?

Solution: The code for question 4 is in 597HW7Q4 We use the model without the logic cut added. The upper bound heuristic we use in our solution is as follows :

- For the assignment variable, we set the task to that unit for which it has the highest value from solving the LP. That is $m* = \arg\max_{m \in M} x_{i,m}, \quad x_{i,m*} = 1$ and $x_{i,m} = 0 \quad \forall \quad m \neq m*$ for all tasks $i \in I$.
- We round all the precedence variables to the nearest integer
- We then solve the LP fixing the assignment and precedence variables. If the LP is feasible, we compare the cost and check if it is lower than the existing upper bound to see if the upper bound can be modified. If the LP is infeasible, we don't modify the upper bound

(a) Part a)We get the same optimal cost as that of the solution in question 3 for both problem sets 1. In total, 1455 nodes were explored in problem set 1 out of which 777 were feasible. In problem set 2, a total of 395 nodes were explored out of which 265 were feasible.

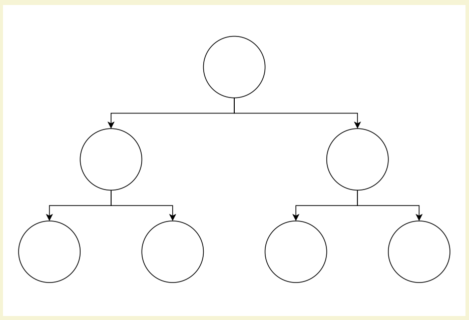
(b) Part b) We get the same optimal cost as that of the solution in question 3 for both problem sets 1. In total, 3061 nodes were explored in problem set 1 out of which 1628 were feasible. In problem set 2, a total of 155 nodes were explored out of which 138 were feasible.

5. Solve the following problems related to the branch and bound algorithm.

   (a) Show that the number of nodes in a tree where we represent all possible combinations of m $0-1$ binary variables is $2^{m+1} - 1$.

   (b) If a complete enumeration of all the nodes in the tree were required, by what factor would this enumeration increase with respect to the direct enumeration of all 0-1 combinations.

Solution: **Part (a)**



We have m variables, and each has 2 options (0 or 1). Starting with a 0 variable means starting with a single root node, and we have a 2^0 combination (2^0 node). Considering 1 variable means branching from the root node into 2 leaf nodes, and we have 2^1 combinations (2^1 more nodes). Considering 2 variables means further branching from the 2 leaf nodes into 4 sub-leaf nodes, we have 2^2 combinations (2^2 more nodes).

Hence, for m variables, the number of nodes is $2^0 + 2^1 + 2^2 + \cdots + 2^m = \frac{1 \times (1-2^{m+1})}{1-2} = 2^{m+1} - 1$ by calculating the sum of geometric series.

Note: For a geometric series $a + ar^1 + ar^2 + \cdots + ar^{n-1}$, the sum is $\frac{a(1-r^n)}{1-r}$.

Part (b)

For m variables, the number of all the nodes is $2^{m+1} - 1$ by the result of part (a), and the number of all $0 - 1$ combinations is 2^m by the description of part (a).

$$\text{factor} = \frac{\text{number of nodes}}{\text{number of combinations}} = \frac{2^{m+1} - 1}{2^m} = 2 - (\frac{1}{2})^m$$