Abstract

This project enhances the conventional simultaneous lot-sizing and scheduling problem by incorporating practicality through profit maximization. Traditional models aim to fulfill all demands, but this approach focuses on maximizing profit.

Two mathematical models are proposed to address this problem, each offering distinct lot-sizing strategies. By redefining the objective function and incorporating demand choice flexibility, this project presents a novel approach to lot-sizing and scheduling that aligns more closely with real-world business goals.

Introduction

Simultaneous lot-sizing and scheduling problems involve optimizing the allocation of resources and scheduling production activities simultaneously. Traditionally, lot-sizing and scheduling decisions are addressed hierarchically, but the General Lot-Sizing and Scheduling Problem (GLSP) integrates them due to their interdependence. Most existing models aim to minimize costs and assume that all customer demands must be met. However, fulfilling all demands in profit-driven businesses may not lead to optimal outcomes.

The present work investigates the objective of maximizing profit in a problem involving both lot-sizing and scheduling while also considering the flexibility of choosing demand. The accepted demand can fluctuate between its upper and lower limits during each period. The upper limit corresponds to the predicted demand, while the lower limit represents the organization's obligations to customers or the minimum production levels set by the production policy. This approach combines decisions regarding accepted demand, lot-sizing, and scheduling into a single problem. The objective function seeks to optimize the discrepancy between sales revenue and the costs associated with production, inventory holding, and setup. The model's output consists of the accepted demand levels for each product in each time period, the magnitude of production batches, and their order. This integrated approach considers medium-term planning (product mix) and short-term scheduling (lot-sizing and scheduling) as part of a single comprehensive problem.

Literature Review

Review 1:

One paper explored the profit maximization capacitated lot-sizing problem (PCLSP) and developed a mathematical model to address it. In this model, demand is a function of price, making the price a variable within the model. The model indirectly determines demand and maximizes sales revenue minus various costs. In another study, Haugen et al. examined the large-scale version of the same problem, focusing on handling the increased complexity and scale of the issue.

Review 2:

Another paper proposed a model that uses pricing decisions to determine the optimal level of demand the firm should meet to maximize profit.

Assumptions

1. Backlog is not allowed

2. Setup times and costs are sequence-dependent.

3. Costs are independent of the period of production.

4. The breakdown of setup time between two periods is not allowed, and the setup is finished within the same period in which it begins.

Notations and Model Parameters

P is the number of products

Ct = Available capacity in each period t.

LDjt□ = Demand lower bound for product j in period t.

UDjt = Demand upper bound for product j in period t.

h = Holding cost for one unit of product j

r = Sales revenue for one unit of product j in period t.

Cp = Production cost for one unit of product j.

Pj = Processing time for one unit of product j.

Scij = Setup cost for the transition from product i to product j.

Stij = Setup time for transition from product i to product j.

Io = Initial inventory level for product j.

where t=1 to T, i and j= 1 to P.

Approach 1

Decision Variables

Qjt = Production quantity of product j in period t to fulfill period t's demand.

Djt□ = The accepted demand of product j in period t.

Ajt = Positive variable, which is 1 when the machine is setup for product j at the beginning of period t.

Yjt = Auxiliary variable that assigns product j to period t.

Ijt = Inventory remaining after time period t for product j.

Xijt = Binary variable 1 when setup occurs from product i to product j in period t.

Mjt = Upper bound of the quantity of Production

Objective function

Maximize Profit: Sales Revenue * Accepted Demand- Production Cost * Quantity Produced - Setup cost for the transition from one product to another - holding cost for inventory

$$\text{Max} \sum_{j=1}^{P} \sum_{t=1}^{T} r_{jt} D_{jt} - \sum_{j=1}^{P} \sum_{t=1}^{T} CP_j Q_{jt} - \sum_{j=1}^{P} \sum_{i=1}^{P} \sum_{t=1}^{T} s_{ij} X_{ijt} - \sum_{j=1}^{P} \sum_{t=1}^{T} h_j I_{jt}.$$

Constraints

$$I_{jt} = I_{j(t-1)} + Q_{jt} - D_{jt} \qquad j = 1,\ldots,P,\; t = 1,\ldots,T,$$

Inventory balancing constraint.

$$Ld_{jt} \leqslant D_{jt} \leqslant Ud_{jt} \qquad j = 1,\ldots,P,\; t = 1,\ldots,T,$$

Upper, lower bound of demand.

$$\sum_{j=1}^{P} p_j Q_{jt} + \sum_{i=1}^{P}\sum_{j=1}^{P} st_{ij} X_{ijt} \leqslant C_t \qquad t = 1,\ldots,T,$$

This constraint shows the capacity limit for production.

$$Q_{jt} \leqslant M_{jt}\left(\sum_{i=1}^{P} X_{ijt} + A_{jt}\right) \qquad j = 1,\ldots,P,\; t = 1,\ldots,T,$$

Constraint that guarantees that a product is produced if the machine has been setup for it. Upper bound for the quantity of production in this constraint is given by:

$$M_{jt} = \min\left\{\frac{C_t}{p_j}, \sum_{k=t}^{T} Ud_{jk}\right\} \quad j = 1,\ldots,P,\; t = 1,\ldots,T.$$

$$\sum_{j=1}^{P} A_{jt} = 1 \qquad t = 1,\ldots,T,$$

Constraint shows that there is only one setup state at the beginning of each period.

$$A_{jt} + \sum_{i=1}^{P} X_{ijt} = A_{j(t+1)} + \sum_{i=1}^{P} X_{jit} \qquad j = 1,\ldots,P,\ t = 1,\ldots,T,$$

Constraint that guarantees that the setup states the network is a connected network

$$Y_{it} + PX_{ijt} - (P-1) - PA_{it} \leqslant Y_{jt} \qquad i = j = 1,\ldots,P,\ i \neq j,\ t = 1,\ldots,T,$$

Constraint for avoiding disconnected sub-tours.

$$X_{ijt}, \in \{0,1\} \qquad i = j = 1,\ldots,P,\ t = 1,\ldots,T,$$

$$Q_{jt}, I_{jt}, Y_{jt}, D_{jt}, A_{jt} \geqslant 0 \qquad j = 1,\ldots,P,\ t = 1,\ldots,T.$$

These constraints show the different kinds of variables in the model.

Approach - 2

Decision Variables

Qjtk = Production quantity of product j in period t to fulfill period k's demand.

Djt = The accepted demand of product j in period t.

Ajt = Positive variable, which is 1 when the machine is setup for product j at the beginning of period t.

Yjt = Auxiliary variable that assigns product j to period t.

Xijt = Binary variable, which is 1 when setup occurs from product i to product j in period t.

Mjtk = Upper bound of the quantity of Production

To consider initial inventory, which will be used in different periods or will remain unused at the end of the planning horizon, we have the following two new variables:

Qjok = Quantity of initial inventory used in period t

Rjo = Unused portion of initial inventory at the end of the planning horizon.

Objective function

Maximize Profit = Revenue - All the Costs Incurred.

Where costs incurred include - production and inventory costs of production quantities in different periods, setup costs, initial inventory holding cost until the used period, and the unused initial inventory holding cost.

$$\text{Max} \sum_{j=1}^{P}\sum_{t=1}^{T} r_{jt} D_{jt} - \sum_{j=1}^{P}\sum_{t=1}^{T}\sum_{k=t}^{T} CQ_{j(k-t)} Q_{jtk} - \sum_{j=1}^{P}\sum_{i=1}^{P}\sum_{t=1}^{T} s_{ij} X_{ijt} - \sum_{j=1}^{P} \sum_{t=1}^{T} Q_{j0t} h_j (t-1) - \sum_{j=1}^{P} Th_j R_{j0}.$$

CQ□□ is a parameter that combines the production cost and inventory cost of product *j,* which will be used *t* periods after the production period. This parameter is calculated as :

$$CQ_{jt} = Cp_j + t \times h_j \quad j = 1, \ldots, P, \quad t = 1, \ldots, T.$$

Constraints

$$\sum_{t=0}^{k} Q_{jtk} = D_{jk} \quad j = 1, \ldots, P,\ k = 1, \ldots, T,$$

This constraint shows that the accepted demand in period t is fulfilled by productions for this period or initial inventory.

$$Ld_{jt} \leqslant D_{jt} \leqslant Ud_{jt} \quad j = 1, \ldots, P,\ t = 1, \ldots, T,$$

This constraint guarantees that the accepted demand in each period is between its lower and upper bounds.

$$\sum_{k=1}^{T} Q_{j0k} + R_{j0} = I_{j0} \quad j = 1, \ldots, P,$$

This constraint ensures that the initial inventory is equal to the sum of the initial inventory used in different periods and the unused portion of the initial inventory at the end of the planning horizon.

$$Q_{jtk} \leqslant M_{jtk}\left(\sum_{i=1}^{P} X_{ijt} + A_{jt}\right) \quad j=1,\ldots,P,\, t=1,\ldots,T, K=t,\ldots,T,$$

This constraint guarantees that the machine is setup for production, and the production's upper bound in these constraints is shown:

$$M_{jtk} = \min\left\{\frac{C_t}{p_j}, Ud_{jk}\right\} \quad j=1,\ldots,P,\, t=1,\ldots,T,\, k=t,\ldots,T.$$

NOTE: Due to this upper bound, this constraint is tighter than the upper bound in the approach, which was,

$$M_{jt} = \min\left\{\frac{C_t}{p_j}, \sum_{k=t}^{T} Ud_{jk}\right\} \quad j=1,\ldots,P,\, t=1,\ldots,T.$$

$$\sum_{j=1}^{P}\sum_{k=t}^{T} p_j Q_{jtk} + \sum_{i=1}^{P}\sum_{j=1}^{P} St_{ij} X_{ijt} \leqslant C_t \quad t=1,\ldots,T,$$

This constraint shows the capacity limit for production.

$$\sum_{j=1}^{P} A_{jt} = 1 \quad t=1,\ldots,T,$$

$$A_{jt} + \sum_{i=1}^{P} X_{ijt} = A_{j(t+1)} + \sum_{i=1}^{P} X_{jit} \quad j=1,\ldots,P,\, t=1,\ldots,T,$$

$$Y_{it} + PX_{ijt} - (P-1) - PA_{it} \leqslant Y_{jt} \quad i=j=1,\ldots,P,\, i \neq j,\, t=1,\ldots,T,$$

Like the first model, these constraints define the sequence of products, set up state change in each period, and set up transmission between two sequential periods.

$$X_{ijt} \in \{0,1\} \qquad i = j = 1,\ldots,P,\ t = 1,\ldots,T,$$

$$Q_{jtk} \geqslant 0 \quad j = 1,\ldots,P,\ t = 1,\ldots,T,\ k = t,\ldots,T,$$

$$I_{jt}, Y_{jt}, A_{jt}, D_{jt} \geqslant 0 \quad j = 1,\ldots,P,\ t = 1,\ldots,T.$$

These constraints show the different kinds of variables in the model.

Methodology

For this project, we employed the CPlex solver to optimize both methods, utilizing Python programming within Google Colab to generate input data. CPlex facilitated efficient solution finding, while Python on Google Colab provided a collaborative environment for data generation.

Codes were executed on a computer with a 3.1GHz CPU and 16 GB RAM.

Result

In Approach 1, the objective is to maintain inventory within specified bounds. Setup initiates production, with production quantity regulated by capacity limits. Constraints ensure setup states remain connected, preventing disjointed sub-tours, resulting in an **objective function value of Rs. 32145.1551754541.**

```
D = [[51 59 58 52 68]

      [38 68 49 46 31]

     [38 69 70 67 70]];
```

In Approach 2, constraints ensure demand fulfillment via production and initial inventory. They set demand bounds, maintain inventory consistency, enforce production capacity and limits, ensure setup readiness, and define sequential setup and product sequencing, yielding an **objective function value of Rs. 32145.1551754541.**

D = [[51 59 58 52 68]

[38 68 49 46 31]

[38 69 70 67 70]];

We obtain the same results from both methods.

Conclusion

In conclusion, this project examined the complex relationship between lot-sizing and scheduling in production planning, introducing a new method that prioritizes profit maximization rather than cost reduction. Traditionally, the primary objective of production planning has been to minimize expenses while simultaneously satisfying all customer requirements. However, our study introduces a more dynamic model that prioritizes profit by redefining the objective function.

Our proposed model allows companies to strategically determine their production choices, timing, and order acceptance by prioritizing revenue maximization over cost reduction. This approach aligns production activities with market opportunities, allowing firms to optimize their operational and financial performance. The model has the potential to offer a pathway for companies to enhance profitability and competitiveness in complex, real-world manufacturing environments. Future research could expand on this framework by exploring its application in various industries and market conditions.

References

Literature Review 1:

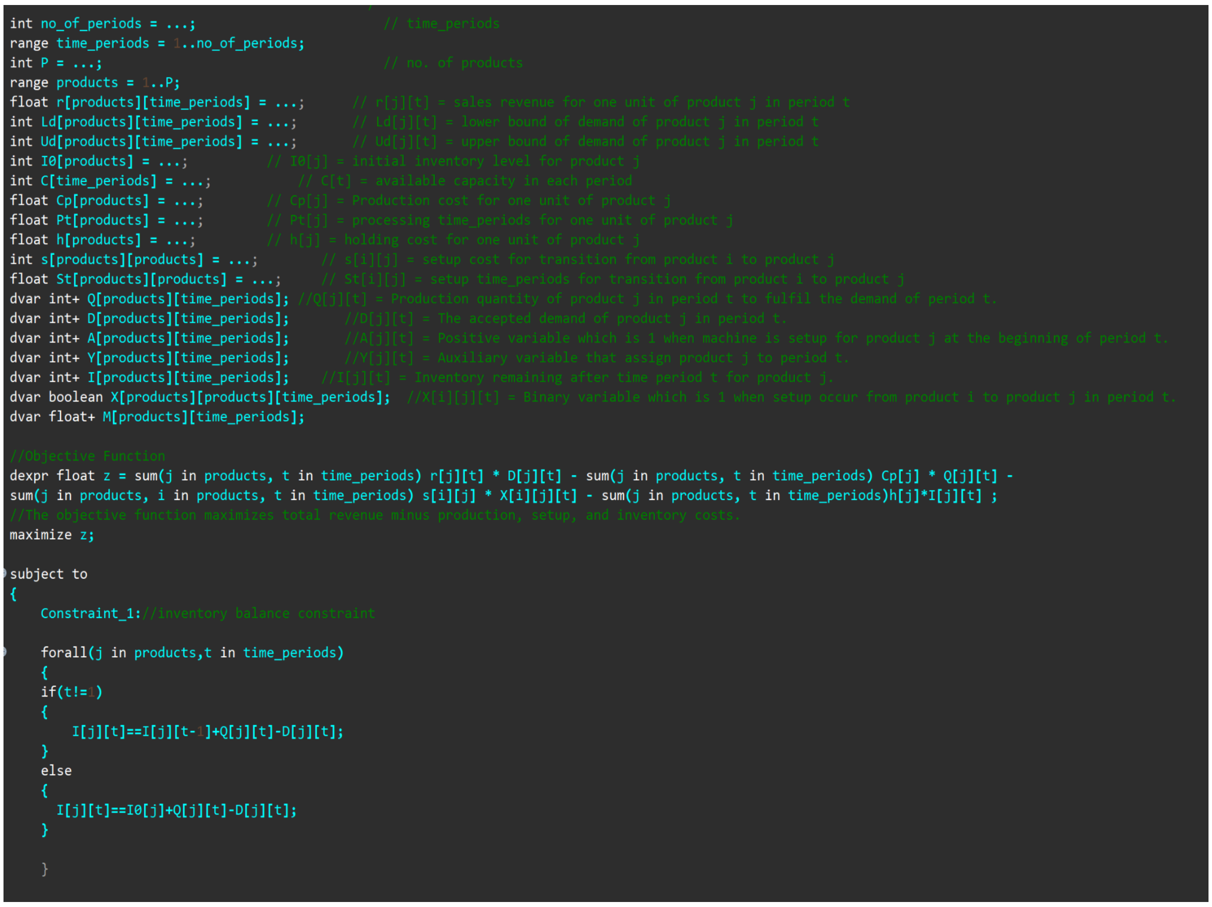
https://www.sciencedirect.com/science/article/pii/S0377221705005898

Literature Review 2:

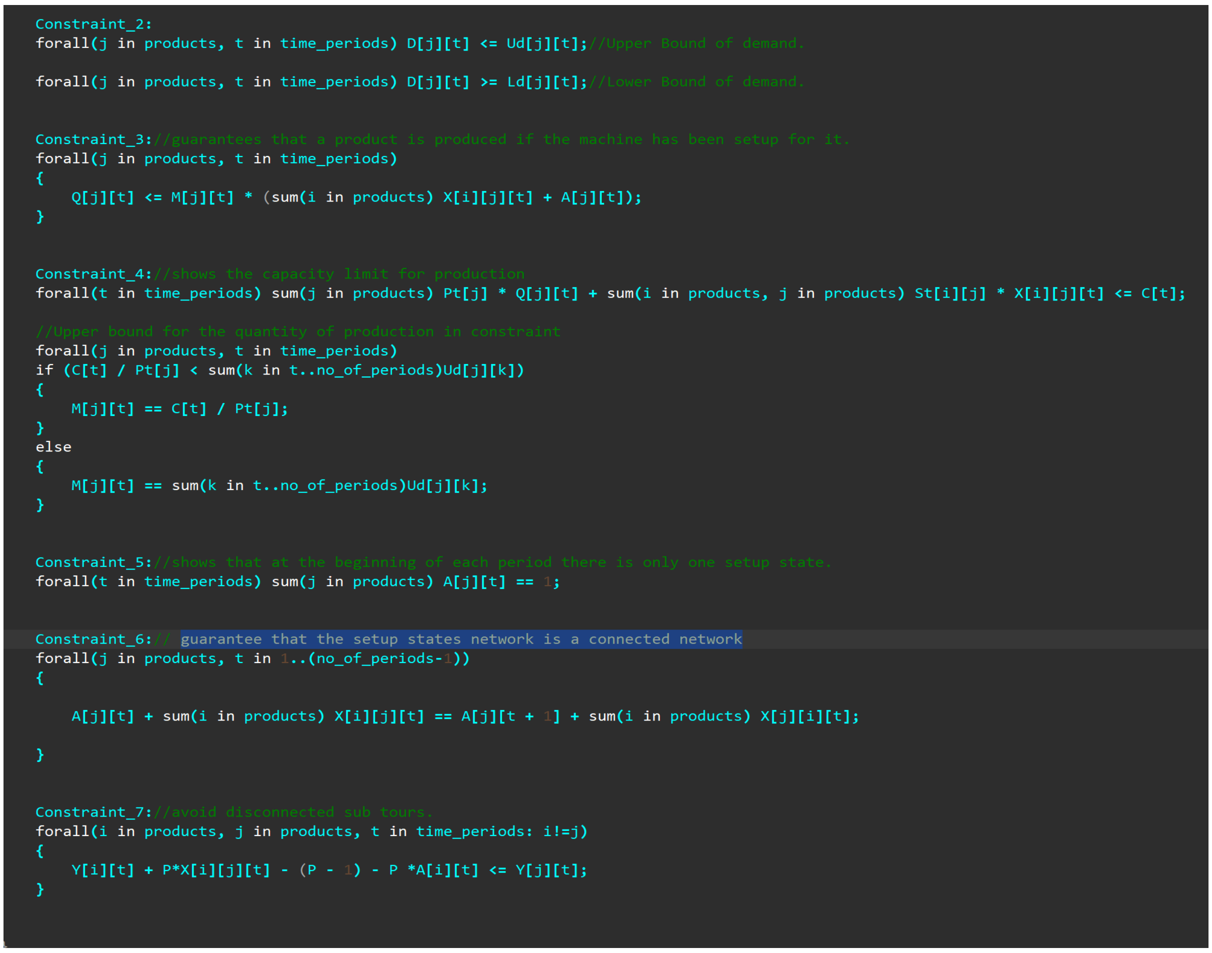
https://pubsonline.informs.org/doi/abs/10.1287/opre.1050.0255

Appendix

Approach 1 Code

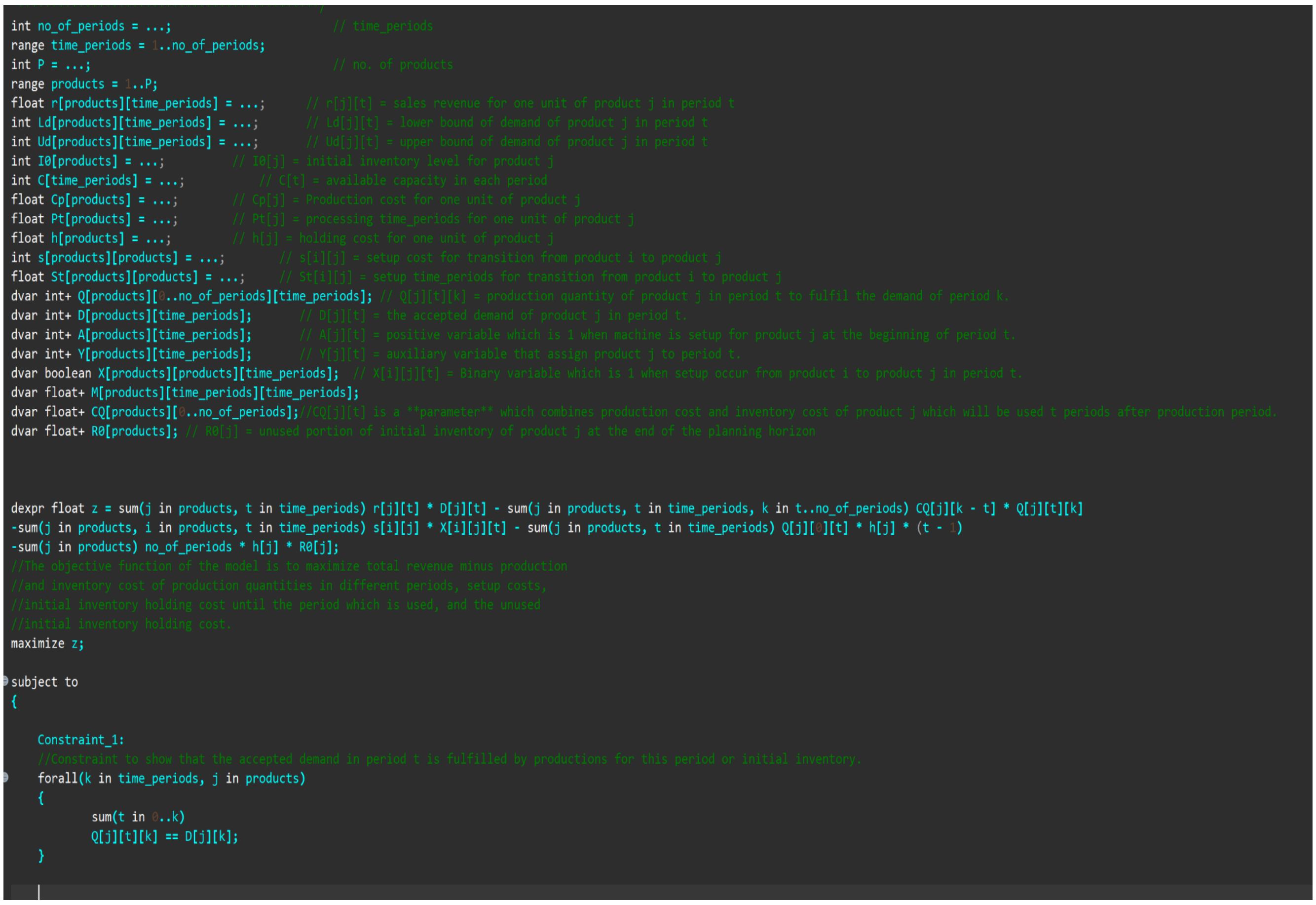


```
int no_of_periods = ...;                          // time_periods
range time_periods = 1..no_of_periods;
int P = ...;                                      // no. of products
range products = 1..P;
float r[products][time_periods] = ...;      // r[j][t] = sales revenue for one unit of product j in period t
int Ld[products][time_periods] = ...;       // Ld[j][t] = lower bound of demand of product j in period t
int Ud[products][time_periods] = ...;       // Ud[j][t] = upper bound of demand of product j in period t
int I0[products] = ...;          // I0[j] = initial inventory level for product j
int C[time_periods] = ...;           // C[t] = available capacity in each period
float Cp[products] = ...;        // Cp[j] = Production cost for one unit of product j
float Pt[products] = ...;        // Pt[j] = processing time_periods for one unit of product j
float h[products] = ...;         // h[j] = holding cost for one unit of product j
int s[products][products] = ...;        // s[i][j] = setup cost for transition from product i to product j
float St[products][products] = ...;     // St[i][j] = setup time_periods for transition from product i to product j
dvar int+ Q[products][time_periods]; //Q[j][t] = Production quantity of product j in period t to fulfil the demand of period t.
dvar int+ D[products][time_periods];       //D[j][t] = The accepted demand of product j in period t.
dvar int+ A[products][time_periods];       //A[j][t] = Positive variable which is 1 when machine is setup for product j at the beginning of period t.
dvar int+ Y[products][time_periods];       //Y[j][t] = Auxiliary variable that assign product j to period t.
dvar int+ I[products][time_periods];    //I[j][t] = Inventory remaining after time period t for product j.
dvar boolean X[products][products][time_periods];  //X[i][j][t] = Binary variable which is 1 when setup occur from product i to product j in period t.
dvar float+ M[products][time_periods];

//Objective Function
dexpr float z = sum(j in products, t in time_periods) r[j][t] * D[j][t] - sum(j in products, t in time_periods) Cp[j] * Q[j][t] -
sum(j in products, i in products, t in time_periods) s[i][j] * X[i][j][t] - sum(j in products, t in time_periods)h[j]*I[j][t] ;
//The objective function maximizes total revenue minus production, setup, and inventory costs.
maximize z;

subject to
{
    Constraint_1://inventory balance constraint

    forall(j in products,t in time_periods)
    {
    if(t!=1)
    {
        I[j][t]==I[j][t-1]+Q[j][t]-D[j][t];
    }
    else
    {
      I[j][t]==I0[j]+Q[j][t]-D[j][t];
    }

    }
```

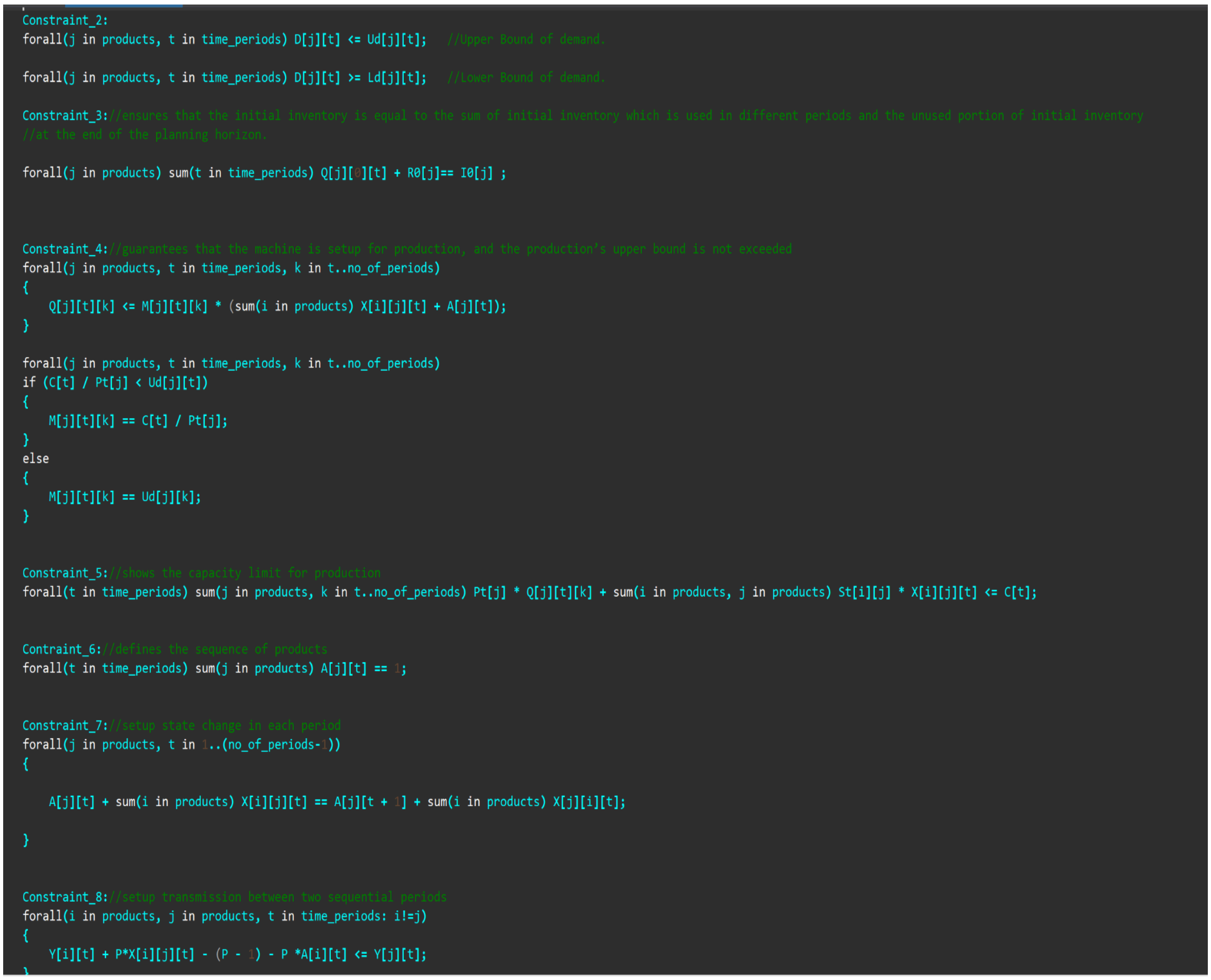


```
    Constraint_2:
    forall(j in products, t in time_periods) D[j][t] <= Ud[j][t];//Upper Bound of demand.

    forall(j in products, t in time_periods) D[j][t] >= Ld[j][t];//Lower Bound of demand.


    Constraint_3://guarantees that a product is produced if the machine has been setup for it.
    forall(j in products, t in time_periods)
    {
        Q[j][t] <= M[j][t] * (sum(i in products) X[i][j][t] + A[j][t]);
    }


    Constraint_4://shows the capacity limit for production
    forall(t in time_periods) sum(j in products) Pt[j] * Q[j][t] + sum(i in products, j in products) St[i][j] * X[i][j][t] <= C[t];

    //Upper bound for the quantity of production in constraint
    forall(j in products, t in time_periods)
    if (C[t] / Pt[j] < sum(k in t..no_of_periods)Ud[j][k])
    {
        M[j][t] == C[t] / Pt[j];
    }
    else
    {
        M[j][t] == sum(k in t..no_of_periods)Ud[j][k];
    }


    Constraint_5://shows that at the beginning of each period there is only one setup state.
    forall(t in time_periods) sum(j in products) A[j][t] == 1;


    Constraint_6:// guarantee that the setup states network is a connected network
    forall(j in products, t in 1..(no_of_periods-1))
    {

        A[j][t] + sum(i in products) X[i][j][t] == A[j][t + 1] + sum(i in products) X[j][i][t];

    }


    Constraint_7://avoid disconnected sub tours.
    forall(i in products, j in products, t in time_periods: i!=j)
    {
        Y[i][t] + P*X[i][j][t] - (P - 1) - P *A[i][t] <= Y[j][t];
    }
```

Approach 2 Code



```
int no_of_periods = ...;                              // time_periods
range time_periods = 1..no_of_periods;
int P = ...;                                          // no. of products
range products = 1..P;
float r[products][time_periods] = ...;      // r[j][t] = sales revenue for one unit of product j in period t
int Ld[products][time_periods] = ...;       // Ld[j][t] = lower bound of demand of product j in period t
int Ud[products][time_periods] = ...;       // Ud[j][t] = upper bound of demand of product j in period t
int I0[products] = ...;           // I0[j] = initial inventory level for product j
int C[time_periods] = ...;            // C[t] = available capacity in each period
float Cp[products] = ...;         // Cp[j] = Production cost for one unit of product j
float Pt[products] = ...;         // Pt[j] = processing time_periods for one unit of product j
float h[products] = ...;          // h[j] = holding cost for one unit of product j
int s[products][products] = ...;         // s[i][j] = setup cost for transition from product i to product j
float St[products][products] = ...;      // St[i][j] = setup time_periods for transition from product i to product j
dvar int+ Q[products][0..no_of_periods][time_periods]; // Q[j][t][k] = production quantity of product j in period t to fulfil the demand of period k.
dvar int+ D[products][time_periods];       // D[j][t] = the accepted demand of product j in period t.
dvar int+ A[products][time_periods];       // A[j][t] = positive variable which is 1 when machine is setup for product j at the beginning of period t.
dvar int+ Y[products][time_periods];       // Y[j][t] = auxiliary variable that assign product j to period t.
dvar boolean X[products][products][time_periods];  // X[i][j][t] = Binary variable which is 1 when setup occur from product i to product j in period t.
dvar float+ M[products][time_periods][time_periods];
dvar float+ CQ[products][0..no_of_periods];//CQ[j][t] is a **parameter** which combines production cost and inventory cost of product j which will be used t periods after production period.
dvar float+ R0[products]; // R0[j] = unused portion of initial inventory of product j at the end of the planning horizon



dexpr float z = sum(j in products, t in time_periods) r[j][t] * D[j][t] - sum(j in products, t in time_periods, k in t..no_of_periods) CQ[j][k - t] * Q[j][t][k]
-sum(j in products, i in products, t in time_periods) s[i][j] * X[i][j][t] - sum(j in products, t in time_periods) Q[j][0][t] * h[j] * (t - 1)
-sum(j in products) no_of_periods * h[j] * R0[j];
//The objective function of the model is to maximize total revenue minus production
//and inventory cost of production quantities in different periods, setup costs,
//initial inventory holding cost until the period which is used, and the unused
//initial inventory holding cost.
maximize z;

subject to
{

    Constraint_1:
    //Constraint to show that the accepted demand in period t is fulfilled by productions for this period or initial inventory.
    forall(k in time_periods, j in products)
    {
            sum(t in 0..k)
            Q[j][t][k] == D[j][k];
    }

    |
```



```
Constraint_2:
forall(j in products, t in time_periods) D[j][t] <= Ud[j][t];   //Upper Bound of demand.

forall(j in products, t in time_periods) D[j][t] >= Ld[j][t];   //Lower Bound of demand.

Constraint_3://ensures that the initial inventory is equal to the sum of initial inventory which is used in different periods and the unused portion of initial inventory
//at the end of the planning horizon.

forall(j in products) sum(t in time_periods) Q[j][0][t] + R0[j]== I0[j] ;



Constraint_4://guarantees that the machine is setup for production, and the production's upper bound is not exceeded
forall(j in products, t in time_periods, k in t..no_of_periods)
{
    Q[j][t][k] <= M[j][t][k] * (sum(i in products) X[i][j][t] + A[j][t]);
}

forall(j in products, t in time_periods, k in t..no_of_periods)
if (C[t] / Pt[j] < Ud[j][t])
{
    M[j][t][k] == C[t] / Pt[j];
}
else
{
    M[j][t][k] == Ud[j][k];
}


Constraint_5://shows the capacity limit for production
forall(t in time_periods) sum(j in products, k in t..no_of_periods) Pt[j] * Q[j][t][k] + sum(i in products, j in products) St[i][j] * X[i][j][t] <= C[t];


Contraint_6://defines the sequence of products
forall(t in time_periods) sum(j in products) A[j][t] == 1;


Constraint_7://setup state change in each period
forall(j in products, t in 1..(no_of_periods-1))
{

    A[j][t] + sum(i in products) X[i][j][t] == A[j][t + 1] + sum(i in products) X[j][i][t];

}


Constraint_8://setup transmission between two sequential periods
forall(i in products, j in products, t in time_periods: i!=j)
{
    Y[i][t] + P*X[i][j][t] - (P - 1) - P *A[i][t] <= Y[j][t];
```

```
//not a constraint, helps calculate CQ[j][t]
forall(j in products, t in 0..no_of_periods)
{
  CQ[j][t] == Cp[j] + t*h[j];
}
```

Data Generation:

Dataset_Generation_OR_PROJECT.ipynb