Supply Chain Operation Planning: A Linear and Combinatorial Approach

Nikolopoulos Emmanouil, 8th Semester, 1072678
Electrical and Computer Engineering, University of Patras, 2023

Abstract

Nowadays everything we use, everything we buy needs to be produced and to achieve so, multiple materials need to be either produced, transferred or assembled. The constantly rising population creates high demand for products, and the reduction of the available resources creates zero tolerance to waste of materials due to wrong management. Having understood the importance of the above, we can easily understand why we, now more than ever, need to have a very specified Supply Chain Operation Planning (SCOP). To meet those demands, we focus on production environments where arbitrary supply chains structures exist. In this paper, we will present a linear programming approach to this problem, introduced by J.M. Spitter at al. (2005) [1] and an example, which we will solve using Python.

Keywords: Supply Chain, Python, Linear Programming, Combinatorial Optimization

P.S.: The Python solution is attached in a Jupyter file named *code.ipynb.* The presentation, the report, and the code can be found on the following GitHub repository.

Supply Chain Operation Planning: A Linear and Combinatorial Approach

Introduction

In this paper, we will at first describe the SCOP, by implementing all the assumptions made for the purposes of this course. Then, we will suggest an arbitrary problem and a Python solver, using which we will find the optimal solution for the described situation. Through out the paper, the 14 assumptions made will be highlighted and numbered like this: (A_x), with x being the number of the assumption. Later on, we will discuss on further research that could be conducted and what are the latest and state of art applications of SCOP into aspects of everyday life.

SCOP and our assumptions

The Supply Chain Operation Planning problem is common and well know to many researchers in multiple fields, such as economy, mathematics and engineering. The main goal of SCOP is to coordinate the release of materials and resources in the supply network under consideration such that customer service constraints are met at minimal cost [2]. It combines the Master Production Schedule (MPS) with Rough Cut Capacity Planning (RCCP), Material Requirements Planning (MRP) and Capacity Requirements Planning (CRP) [4]. MPS is a plan that is based on production capacity, changes in inventory of finished products, fluctuations in demand, efficiency and utility of production factors, and lot size [3]. RCCP is the manufacturing process used to determine if a company has the resources to meet its production goals, pretty much the same as CRP. The base concept of arbitrary supply chain structures is that the products that are being sold to customers consist of multiple items, where each of those items also consists of multiple items and can be used to also construct multiple items. The SCOP function is responsible for the coordination of activities along

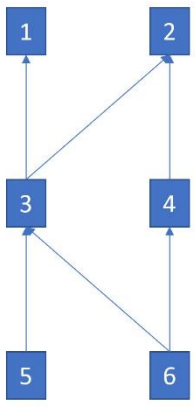

Image 1

the supply chain, by making decisions on the quantities and timing of material and resource releases [5]. As we can observe from image 1, we consider 3 kinds of items: “parent items”, “child items” and “end items”. Each parent items consists of multiple child items and some parent items are end items, that are not used by any other item. In image 1, 1 and 2 are end items, 2 is parent item for 3 and 4, 5 and 6 are child items to item 3 … etc. We will model capacity constraints on the quantity of items that can be assembled within a specific time interval (A_1). Also, items can be assigned to multiple resources (A_2). We assume that items can be produced in several different resources and that each of these resources can produce different items (A_3). The capacity of the resources is constraint (A_4) and we assume planned lead times with multiperiod capacity consumption and backorders for only end items (A_5). Backorders are considered the end items that were ordered in the passed, have not yet been delivered and thus they are being “owned”. As mentioned above, the basic issue of SCOP is to ensure that, given the constraints of the system (resource, material availability) the optimal possible quantity of the item is release at the best possible time to obtain the lowest costs. We also assume that orders released at the start of period must be processed before the given planned lead time expires (A_6) and that the capacity required to produce the order is consumed during the planned lead time interval (A_7). The items released are available at their due date only (A_8), and we do not allow inclusion for lot sizing constraints (A_9). Furthermore, we assume that the release work order R_{it} can be split up and each party may be produced on several resources that can operate in a parallel way (A_{10}). The work order is not available before time $t + \tau_i$ (A_{11}), the unit cost is the same for each of the facilities that can assemble an item (A_{12}). Lastly, we assume that there is no sense in starting the production of a request when earlier requests have not yet been finished (A_{13}) and that as starting values there are neither inventories nor backorders for any item (A_{14}).

Problem dynamics

Before we suggest a linear programming approach, we need to discuss the dynamics of the problem, which, when added to the assumptions made before, describe the problem in total. The observations made will be annotated like O_x. Firstly we consider that at time t there is no work in process and a work order quantity R_{it} is released (O_1) and that a work order released at time t will only be available for the next stage at time $t + \tau_i$. Also, (O_2) the units that are ready can be used in a dependent or independent demand at time $t + \tau_i$ but not before that time (O_3). After the work order release, the amounts of components are being gathered (O_4), based on the matrix h, which will be introduced later in the analysis. After the execution of each work order, in all periods the capacity constraints should be met. The released work order R_{it} can be split up, and each of these parts could be produced on several resources, not explicitly one (O_5).

After describing the problem's dynamics we can now proceed to explain the parameter which we will use to define and solve the problem.

Linear Programming Formulation

We begin by introducing the parameters. We need a planning horizon which defines for how long we are planning. So we consider,

T : $T \in N$, as the planning horizon

τ_i : $\tau_i \geq 1$, $i \in [1,n]$ as lead time, meaning, the time needed to create item i

n : $n \in N$, as the number of items

p_i : $p_i > 0$, as the unit cost in terms of capacity for the production of item i, as mention on A_{12}, all the facilities that can produce i need the same time

H : matrix where $h_{ij} \in N$, and h_{ij} represents how many units of item i are needed for the production of a single j unit. The matrix is low triangular and the diagonal of H is obviously zero, because we cannot use one item to create itself

a_{it} : $a_{it} \in N$ and $i \in [1,n]$, $t \in [1,T]$, as the inventory costs of holding one unit of item i during time t

b_{it} : $b_{it} \in N$ and $i \in [1,n]$, $t \in [1,T]$, as the backordering costs of one unit of item i during time t. Backordering could be considered the quantity of each item i that is being "owed" from previous transactions.

D_{it} : $D_{it} \in N$ and $i \in [1,n]$, $t \in [1,T-1]$, as the exogenously determined demand for each item i during time t

k : $k \in N$, the number of resources

c_{ut} : $c_{ut} \in N$ and $u \in [1, k]$, $t \in [1, T]$, as the maximum production time available on resource u in time slot t

R_i : as the set of resources that are used by each item i, $i \in [1,n]$

I_{it} : as the inventory of each item i at time t, where $i \in [1,n]$, $t \in [0, T - 1]$

G_{it} : as the gross requirements of each item i at time t, where $i \in [1,n]$, $t \in [0, T - 1]$

R_{it} : as the planned work order release of each item i at time t, where $i \in [1,n]$, $t \in [- \tau_i, T -1]$

B_{it} : as the backorders of each item i at time t, where $i \in [1,n]$, $t \in [0, T - 1]$

V_{iut} : as the capacity allocated to each item i at resource u, in time t, where $i \in [1, n]$, $u \in [1, k]$, $t \in [- \tau_i + 2, T]$

Z_{its} : as the part of the planned work order release of each item I, requested on at time t and executed in time s, where $i \in [1, n]$, $s \in [t+1, (t + \tau_i) \leq T]$, $t \in [- \tau_i + 1, T - 1]$.

From all these variables, we consider as input variables the T, τ_i, n, p_i, H, a_{it}, b_{it}, D_{it}, k, c_{ut}, R_i. We also need to mention that I_{it}, R_{it}, B_{it}, V_{iut}, Z_{its} are consider input variables, because the exist at time 0, referring to any preexisting situation. If we are talking about $t < 0$ they are input values, otherwise they are output values.

Now that we have efficiently described the variables of our problem it's high time we presented the model, introduced by J.M. Spitter at al. (2005) [1]. The main production variables we are mostly interested about are the R_{it}, for i = 1, …, n, and time t = 0, …,T-1.

$$Min \sum_{t=1}^{T} \sum_{i=1}^{n} a_{it} I_{i,t-1} + \sum_{t=1}^{T} \sum_{i=1}^{n} b_{it} B_{i,t-1} \quad (e_1)$$

As it is obvious, the goal is to minimize the costs, which consists of the inventory cost of each item, multiplied by the inventory at each given time, added to the cost that derives from the number of backorders at each time slot, multiplied by the cost of the backordering of each item.

The objective is subject to

$$I_{it} = I_{i,t-1} + R_{i,t-\tau_i} - D_{it} - G_{it} + B_{it} - B_{i,t-1}, \text{ for } i \in [1, n], t \in [0, T-1] \quad (e_2)$$

This equation (e_2) is about the inventory at each time period stating that the inventory equals to what existed, plus the incoming, minus the outgoing inventory, for each item i, and each time slot t.

$$G_{it} = \sum_{j=1}^{n} h_{ij} R_{jt}, \text{ for } i \in [1, n], t \in [0, T-1] \quad (e_3)$$

In equation (e_3), we relate the gross requirements of each item i, to the planned work order releases of items, of which item i is a component. In this way, we know the requirements of each item i, to create another item, at a given time slot t.

$$\sum_{s=-\tau_i}^{t} p_i R_{is} \geq \sum_{u \in R_i} \sum_{s=-\tau_i+1}^{t+1} V_{ius} \text{ , for } i \in [1, n], t \in [-\tau_i + 1, T-1] \quad (e_4)$$

$$\sum_{s=-\tau_i}^{t} p_i R_{is} \geq \sum_{u \in R_i} \sum_{s=-\tau_i+1}^{t+\tau_i, s \leq T} V_{ius} \text{ , for } i \in [1, n], t \in [-\tau_i + 1, T-1] \quad (e_5)$$

In equations (e_4) and (e_5) reassures that the capacity is not claimed before the request is made, and that the requests are produced within their lead time, respectively.

$$\sum_{i:u\in R_i} V_{iut} \leq c_{ut} \text{ , for i}\in[1, k], t\in[1, T] \ (e_6)$$

With equation (e_6), we assure that the total claimed capacity on a certain resource u, in a time slot t, is not exceeding the capacity of that certain resource.

$$B_{it} - B_{i,t-1} \leq D_{it}, \text{ for i}\in[1, n], t\in[1, T\text{-}1] \ (e_7)$$

In this last equation (e_7), we are making sure that backordering can only occur on actual exogenously determined demand.

Finally, we assume that $I_{it}, R_{it}, B_{it}, V_{iut} \geq 0$ (e_8).

Now that we sufficiently described the linear programming model, it is important to note that even though the planning horizon is considered to be T, the solutions are only obtained up to time T – 1, because the R values on time slot T, are depended on future inputs that will be given after time slot T – 1.

Example of our model

For the purposes of this paper, we have created an example to this problem, which we will solve using Python. To achieve so, we are using Python 3.7.5 and the libraries *numpy, pulp, pymprog* and *traceback*. The code can be found on the attached Jupyter file *(code.ipynb)* and contains further comments on the logic behind the code. For this example the assumptions will again be annotated like before [a_x]. We have decided to create a

scenario in which we have 3 levels, including 5, 4 and 2 items (a_1), and 3, 2 and 1 resources respectively(a_2). So, we have 11 items in total, with 2 being the end items(a_3). The relations between items can be seen on image 2 (a_4). Image 2 also summarizes all the information regarding the structure of our example. Planning horizon will be equal to 10, thus T = 10 (a_5), and τ_i is considered equal to 2 for all the items i (a_6). To keep the model simpler, we consider the initialization of I_{it}, R_{it}, B_{it}, V_{iut}, Z_{its} equal to 0, meaning there is no previous inputs (a_7). We also provide our solver with p_i, H_{ij} (which derives from image 2), T_i, a_{it}, b_{it}, D_{it}, k_i, k, c_{ut}, and R_{it}. Since all these matrixes, and variables sum up to a total of 391 arbitrary values, for the model described by a_1 – a_7, we cannot be sure about the results of our solver, since the number of random values is extremely high. Hence we only create this solver for the simulative purposes of this paper.

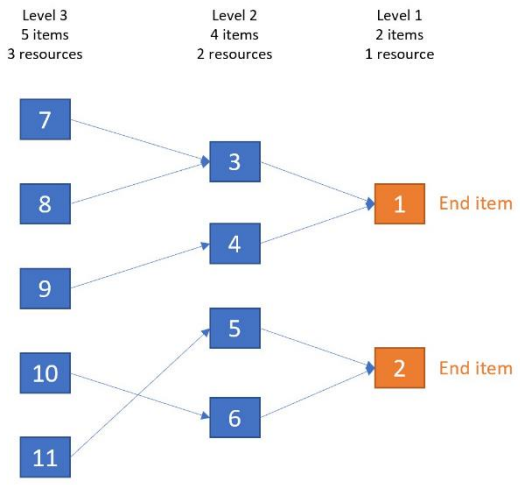

Image 2

After having carefully initialized the variables of our problem, we start the modeling using the pulp library. We create the problem, name it, and state that it is minimalization problem. We then create a variable matrix x for the R_{it} variables, which, as stated above, are the main decision variables. After that, we create the I, G, B, V, Z, for the matrixes I_{it}, G_{it}, B_{it}, V_{iut}, Z_{its} respectively, for which, we take into account the equations (e_8). We can now create the objective of the problem, as mentioned in (e_1) and add the constraints, as referred in (e_2) – (e_7). Then we ask Python to solve the problem and the

```
Problem Status: Optimal
[[ 2.  3.  2.  4.  4.  3.  2.  0.  0. nan]
 [ 0.  4.  1.  0.  0.  0.  3.  0.  0. nan]
 [ 7.  8. 10.  6.  6.  1.  1.  0.  0. nan]
 [ 8. 12. 14. 10.  7.  0.  0.  0.  0. nan]
 [ 4.  0.  1.  1. 11.  5.  3.  0.  0. nan]
 [ 5.  0.  1.  2.  6.  1.  0.  0.  0. nan]
 [11.  6.  7.  3.  2.  1.  0.  0. nan nan]
 [20. 12. 13.  4.  4.  1.  3.  2. nan nan]
 [56. 41. 30.  0.  0.  0.  0. 12. nan nan]
 [ 1.  3. 10.  3.  2.  1.  1.  0. nan nan]
 [ 3.  4. 25. 11.  7.  1.  1.  0. nan nan]]
```

Image 3

results are being printed in image 3. As we expected, we see a matrix, with rows being the 11 items, and columns being the time slots, from time slot 0 to time slot T. As we mentioned above, the final column includes NaN values, which derives from the fact that we need more information to calculate them, and more specifically more information regarding the next time slot. We also need to highlight that the for the T – 1 time slot and items 7, 8, 9, 10 and 11, we also find NaN values. This is happening due to the fact that we don't provide the system with information on the level's 1 items for the final production state. We need to highlight that we are uncapable of checking the correctness of this model by hand, due to its complexity, and the arbitrary selection of values, as mentioned above.

Alternative Solutions

Since Linear Programming is a topic with multiple applications, a huge variety of tools is provided to solve this kind of problems. Rather than pulp, we could use pymprog, which uses similar logic, but also methods using LP – CUM , LP – BAL and Dual Simplex could be used, as referred in literature [1]. Unfortunately, since we don't have access to CPLEX, we couldn't use them to compare their results to our Python approach for the example we mentioned above.

Further research and advanced topics

In this paper we focused our approach on a simple model that eliminated many more restrictions. For example, we could add complexity regarding the time each resource take to produce the different items. We could also assume different transportation variables, expiring products, and priority to create some products over others. We could consider that end items could be parent items to some other items, creating a non balanced tree format. Furthermore, it would be interesting to add a new variable for time seasons, for example, winter or summer, specific weeks of the year like holidays etc. And lastly, we could add uncertainty to our model.

Apart from the simple implementation of creating an item, let's say a pen, or a notebook, the SCOP approach used to simulate far more complicated situations, like the design of a power grid that will use IoT sensors, and the blockchain technology to create a supply chain approach. This model, which is introduced by Xueyong Tang at al [6], could reduce the equipment needed to create a power grid and provide stability. We can also see another very interesting use of SCOP, in [7], Kangli Zhao et al introduced an approach, in which they created a model for the problem of blood supply chain optimization under the uncertainty of supply and demand, by considering the impact of unreliability, demand difference and the multiple inventory allocation methods. In order to adopt the SCOP problem in the newest trend of Artificial Intelligence (AI), María Ángeles Rodríguez et al in [8] describe an approach which uses the supply chain logic improved by the use of AI techniques, which apart from the linear algebra, also provide the suitability, memory and the ability to manage uncertain and constantly alternating information. In the same mindset, Saureng Kumar et al [9], suggested an alternative approach which uses Machine Learning (ML) techniques and SVM, K – NN, DT, etc to create a model that would be able to evaluate a company's organization supply chain risk. In another very interesting research, Farid Kochakkashani et al [10], study the SCOP implementation regarding the vaccine and pharmaceutical case of COVID – 19, by creating a similar model as shown in image 4.

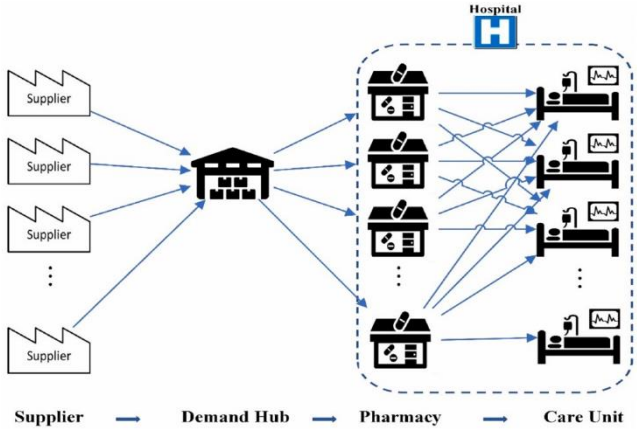

Image 4

Summary

To sum things up, at first we introduced the basics on the SCOP problem and the assumptions we made for the purposes of this paper. Then, we presented the linear

programming problem, the variables, the objective function, and the constraints. After that, we tried to give an example model and used Python to solve the problem. In the end, we mentioned how we could make the model much more precise and simultaneously more complicated, and mentioned the most interesting and up to date research regarding SCOP.

References

[1] Linear programming models with planned lead times for supply chain operations planning, J.M. Spitter et al, European Journal of Operational Research, Volume 163, Issue 3, 2005, Pages 706-720, ISSN 0377-2217, DOI: 10.1016/j.ejor.2004.01.019

[2] Planning Supply Chain Operations: Definition and Comparison of Planning Concepts, Ton G. de Kok, Jan C. Fransoo, Handbooks in Operations Research and Management Science, Elsevier, Volume 11, 2003, Pages 597-675, ISSN 0927-0507, ISBN 9780444513281

[3] Determining the changes in the Master Production Schedule (MPS) at the company with Make to Stock (MTS) and Make to Order (MTO) strategies, R Amaranti et al, 2020 IOP Conf. Ser.: Mater. Sci. Eng. 830 042003, DOI 10.1088/1757-899X/830/4/042003

[4] Evolution of operations planning and control: from production to supply chains, Jan Olhager (2013), International Journal of Production Research, 51:23-24, 6836-6843, DOI: 10.1080/00207543.2012.761363

[5] Planning Supply Chain Operations: Definition and Comparison of Planning Concepts, Ton G. de Kok et al, Handbooks in Operations Research and Management Science, Elsevier, Volume 11, 2003, Pages 597-675, ISSN 0927-0507, ISBN 9780444513281, DOI: 10.1016/S0927-0507(03)11012-2.

[6] Design and Research of Power Grid Equipment Supply Chain Based on Blockchain Technology, Xueyong Tang et al 2020 IOP Conf. Ser.: Earth Environ. Sci. 446 042003 DOI: 10.1088/1755-1315/446/4/042003

[7] Research on Blood Supply Chain Optimization under Supply Unreliable and Stochastic Demand, Kangli Zhao et al 2021 J. Phys.: Conf. Ser. 1910 012013 DOI: 10.1088/1742-6596/1910/1/012013

[8] Artificial Intelligence in Supply Chain Operations Planning: Collaboration and Digital Perspectives, María Ángeles Rodríguez et al 2020, L. M. Camarinha-Matos et al. (Eds.): PRO-VE 2020, IFIP AICT 598, pp. 365–378, 2020,DOI: 10.1007/978-3-030-62412-5_30

[9] Integrated Model for Predicting Supply Chain Risk Through Machine Learning Algorithms, Saureng Kumar et al, 2023, International Journal of Mathematical, Engineering and Management Sciences, Vol. 8, No. 3, 353-373, 2023, DOI: 10.33889/IJMEMS.2023.8.3.021

[10] Supply chain planning of vaccine and pharmaceutical clusters under uncertainty: The case of COVID-19, Farid Kochakkashani et al, Socio-Economic Planning Sciences, Volume 87, Part B, 2023, 101602, ISSN 0038-0121, DOI: 10.1016/j.seps.2023.101602.