Graphical Abstract

Multi-period and Multi-product Batch Processing Time Maximization Problem

Tatiana Balbi Fraga

Highlights

Multi-period and Multi-product Batch Processing Time Maximization Problem

Tatiana Balbi Fraga

- ...;
- ...;
- ...;
- ....

Multi-period and Multi-product Batch Processing Time Maximization Problem

Tatiana Balbi Fraga[a,*]

[a] *Agreste Academic Center - Federal University of Pernambuco, Avenida Marielle Franco, Nova Caruaru, Caruaru, 55014-900, PE, Brazil*

Abstract

...

Keywords: multiproduct batch, processing time maximization, C++, LINGO

1. Introduction

T....

As scientific contributions,....

In the next sections, a presentation of problem is made along with an application example. Section 3 presents an interger linear mathematical model for the problem and section **??** presents an analytical method for its solution. Section 5 presents the tests and results obtained by a solver in C++ applying the analytical solution and a solver developed in LINGO to new proposed benchmarks. In sections 7 and 8, the contributions to this work are informed and acknowledgments are given, respectively. Finally, section 6 presents the paper's conclusions and suggestions for future works.

2. Multi-period and multi-product batch processing time maximization problem

In MPMPBPTM problem we consider a planning horizon with ND days, (ND $\geq$ 1), and a daily demand ($\mathrm{UD}_{id} \geq 0$) during this period for each product of the batch. Additionally, there is already a prior production plan,

*corresponding author

Email address: `tatiana.balbi@ufpe.br` (Tatiana Balbi Fraga)

so that it is possible that there is already a previous production program for each product of the batch. Thus the availability for factory storage and shipping to outlets will be subject to change according to prior production planning. The problem is to determine a maximum processing time for the batch, taking into account the restrictions considered. For a better understanding of the problem, an example is presented below.

Example: A certain machine must process a batch containing 2 different products: A and B. The production rate of A is 60 g/min while the production rate of B is 40 g/min. The factory has free stock for a maximum of 3000 g of any product, and, according to the company's inventory policy, a maximum of 3000 g of product A and 2000 g of product B can be stocked at the factory. For the first day, there is a demand for 1000 g of product A and 500 g of product B, and for the second day, there is a demand for 500 g of product A and 300 g of product B. The factory has an outlet that has free space in stock of 1000 g, which can receive a maximum of 600 g of each product. There is a previous production planning for 1000 units of product A for the first day and 1500 units of product B for the second day. A maximum time of 100 minutes of this machine can be allocated for processing this batch. What is the maximum possible time for processing this batch ?

The following section presents a mathematical model for this problem.

3. Mathematical model

Given that:

UD_{id} is the demand for the product i on day d;
PO_{id} is the planned production of the product i on day d;
O is the maximum quantity allowed for shipment of all products to outlets;
UO_i is the maximum amount of product i that can be shipped to outlets;
UI_i is the factory free inventory for product p on day 0;
p_i is the production rate of product i;
Z is the timeout for batch processing;
UI_{id} is the maximum quantity allowed for stocking in the factory the product i on day d;
I_d is the free factory storage of all products on the end of the day d;

being I the initial free inventory for all products on factory on the start of the (*first*) planning day ($d = 1$);

P_i is the amount of product i to be produced on batch on the current date (first planning day);
D_{id} is the amount of product i delivered for the demand on day d;
O_{id} amount of product i shipped to factory outlets;
T is the batch processing time.

We have the problem:

$$max \quad T \tag{1}$$

$s.t.$

$$P_i - \mathrm{p}_i * T = 0 \quad \forall i \tag{2}$$

$$P_i + UI_{i1} - D_{i1} - O_{i1} = \mathrm{UI}_i - \mathrm{PO}_{i1} \tag{3}$$

$$UI_{id} - UI_{i(d-1)} - D_{id} - O_{id} = -\mathrm{PO}_{id} \quad \forall i, \forall (d > 1) \tag{4}$$

$$I_1 + \sum_i \{P_i - D_{i1} - O_{i1}\} = \mathrm{I}_0 - \sum_i \mathrm{PO}_{i1} \tag{5}$$

$$I_d - I_{d-1} + \sum_i \{P_i - D_{id} - O_{id}\} = -\sum_i \mathrm{PO}_{id} \quad \forall (d > 1) \tag{6}$$

$$\sum_d O_{id} \leq \mathrm{UO}_i \quad \forall i \tag{7}$$

$$\sum_i \sum_d O_{id} \leq \mathrm{O} \tag{8}$$

$$D_{i1} \leq \mathrm{UD}_{i1} \quad \forall i \tag{9}$$

$$D_{id} \leq \sum_{k=1}^{d} \mathrm{UD}_{ik} - \sum_{k=1}^{d-1} D_{ik} \quad \forall i, \forall d > 1 \tag{10}$$

$$T \leq \mathrm{Z} \tag{11}$$

$$UI_{id}, I_d, O_{id}, D_{id} \in \mathbb{Z}^+ \quad \forall i, d \tag{12}$$

$$I_d \in \mathbb{Z}^+ \quad \forall d \tag{13}$$

where:

Constraints in (2) relate the quantity produced, P_i, to batch processing time T. Constraints in (3) calculate the quantity produced, P_i, as a function of the primary variables, D_i, O_i and I_i. Constraints in (4), (5), and (7) state that the quantity delivered to demand, the quantity shipped to the autlets, and the factory-stocked quantity of each product must be less than their respective known limits. Constraints (6) and (8) state that both the sum of product quantities sent to the autlets and the sum of product quantities stocked in the factory must be less than their respective maximum allowed values. The restriction in (9) establishes that there is a batch processing time limit, Z, that must be respected. And finally, the constraints in (10) inform the nature of the decision variables.

4. Exact optimizatin method

Uma solução ótima para este problema pode ser construída através de seis etapas, conforme algoritmos apresentados a seguir:

5. Tests and results

To test the developed model and analytical solution method, we created a solver in C++ and a solver in LINGO (https://github.com/tbfraga/COPSolver). The tests were performed on a notebook with an Intel i7 processor. We tested the solvers developed for the benchmarks presented on Tables 1, 2 and 3, and for randon benchmarks generated by the following functions:

$$\mathrm{p}_i = \mathrm{rand}()\%30 + 10 \tag{14}$$

Algorithm 1 Solving MMBPTM problem — Part 01 - distribution planned production ignoring restrictions defined for the set.

Require: $\mathrm{UD}_{id}, \mathrm{PO}_{id}, \mathrm{UO}_i, \mathrm{UI}_i, \forall i; \mathrm{O}, \mathrm{I}.$

$O \leftarrow \mathrm{O}$
$I_0 \leftarrow \mathrm{I}$
for all d **do**
  for all i **do**
    if $d = 1$ **then**
      $UD_{id} \leftarrow \mathrm{UD}_{id}$
      $A_{id} \leftarrow \mathrm{PO}_{id}$
      $UI_{i(d-1)} \leftarrow \mathrm{UI}_i$
    else
      $UD_{id} \leftarrow \sum_{k=1}^{d}\{\mathrm{UD}_{ik}\} - \sum_{k=1}^{d-1}\{D_{ik}\}$
      $A_{id} \leftarrow \min\{\sum_{k=1}^{d-1}\{\mathrm{PO}_{ik} - D_{ik} - O_{ik}\}\} + \mathrm{PO}_{id}$
    end if
  end for

  $D_{id} \leftarrow \min\{UD_{id}, A_{id}\}$
  $UI_{id} \leftarrow UI_{i(d-1)} - \mathrm{PO}_{id} + D_{id}$
  $I_d \leftarrow I_{d-1} - \mathrm{PO}_{id} + D_{id}$

  if $UI_{id} < 0$ **then**
    $O_{i1} \leftarrow -UI_{i1}$
    $UO_i \leftarrow UO_i - O_{id}$
    $O \leftarrow O - O_{id}$
    $I_d \leftarrow I_d + O_{id}$
    $UI_{id} \leftarrow 0$

    if $UO_i < 0$ **or** $O < 0$ **then**
      Return: error: planning is not feasible !
    end if
  end if
end for
Return: $O_{id}, UI_{id}, \forall(i, d); I_d, \forall d; UO_i, \forall i; O.$

Algorithm 2 Solving MMBPTM problem — Part 02 - adjustment for atteining feasiability.

```
Require: O_id, UI_id, ∀(i,d); I_d, ∀d; UO_i, ∀i; O.
  for all d do
      if I_d < 0 then
          for all i do
              if UO_i ≥ |I_d| then
                  O_id ← O_id − I_d
                  UO_i ← UO_i + I_d
                  O ← O + I_d
                  for all k ≥ d do
                      UI_ik ← UI_ik − I_d
                      I_k ← I_k − I_d
                  end for
                  break;
              else
                  O_id ← O_id + UO_i
                  UO_i ← 0
                  O ← O − UO_i
                  for all k ≥ d do
                      UI_ik ← UI_ik + UO_i
                      I_k ← I_k + UO_i
                  end for
              end if
          end for
      end if
  end for
  Return: O_id, UI_id, ∀(i,d); I_d, ∀d; UO_i, ∀i; O.
```

Algorithm 3 Solving MMBPTM problem — Part 03 - generating a MBPTM problem.

Require: $O_{id}, UI_{id}, \forall(i,d); I_d, \forall d; UO_i, \forall i; O$.

for all i **do**

$\quad \mathrm{UD}_i = UD_{i1}$

$\quad \mathrm{UO}_i = UO_i$

$\quad \mathrm{UI}_i = \min_d\{UI_{id}\}$

end for

$\mathrm{O} = O$

$\mathrm{I} = \min_d\{I_d\}$

Return: $\mathrm{UD}_i, \mathrm{UO}_i, \mathrm{UI}_i, \forall i; \mathrm{O}, \mathrm{I}$.

Algorithm 4 Solving MMBPTM problem — Part 04 - find an optimal T to the MBPTM problem.

Require: $\mathrm{UD}_i, \mathrm{UO}_i, \mathrm{UI}_i, \mathrm{p}_i, \forall i, \mathrm{O}, \mathrm{I}, \mathrm{Z}$

$T^* \leftarrow \lfloor \min\{\min_{\forall i}\{(\mathrm{UO}_i + \mathrm{UI}_i + \mathrm{UD}_i)/\mathrm{p}_i\}, (\mathrm{O} + \mathrm{I} + \sum_i \mathrm{UD}_i)/\sum_i \mathrm{p}_i\}\rfloor$

$T \leftarrow \min\{T^*, \mathrm{Z}\}$

Return: T.

Algorithm 5 Solving MMBPTM problem — Part 05 - calculate D_i, O_i and I_i, $\forall i$, ignoring the restrictions defined for the set.

Require: $\mathrm{UD}_i, \mathrm{UO}_i, \mathrm{UI}_i, \mathrm{p}_i, \forall i, \mathrm{O}, \mathrm{I}, \mathrm{Z}, T$.

for all i **do**

$\quad P_i \leftarrow T * \mathrm{p}_i$

$\quad D_i \leftarrow \min\{\mathrm{UD}_i, E_i\}$

$\quad E_i \leftarrow P_i - D_i$

$\quad O_i \leftarrow \min\{\mathrm{UO}_i, E_i\}$

$\quad E_i = E_i - O_i$

$\quad \mathrm{UO}_i = \mathrm{UO}_i - O_i$

$\quad I_i = E_i$

$\quad \mathrm{UI}_i = \mathrm{UI}_i - I_i$

end for

$SO = \mathrm{O} - \sum_i O_i$

$SI = \mathrm{I} - \sum_i I_i$

Return: $D_i, O_i, I_i, P_i, E_i, \mathrm{UO}_i, \mathrm{UI}_i, \forall i, SO, SI$.

Algorithm 6 Solving MMBPTM problem — Part 06: redistribute production to comply with limitation restrictions for the batch products set.

Require: $O_i, I_i, \mathrm{UO}_i, \mathrm{UI}_i, \forall i, SO, SI$.

  if $SO < 0$ **then**
    for all i **do**
      $O_i \leftarrow O_i - \min\{O_i, UI_i, |SO|\}$
      $I_i \leftarrow I_i + \min\{O_i, UI_i, |SO|\}$
      $SO \leftarrow SO + \min\{O_i, UI_i, |SO|\}$
      if $SO = 0$ **then** break looping for;
      end if
    end for
  end if
  if $SI < 0$ **then**
    for all i **do**
      $I_i \leftarrow I_i - \min\{I_i, UO_i, |SI|\}$
      $O_i \leftarrow O_i + \min\{I_i, UO_i, |SI|\}$
      $SI \leftarrow SI + \min\{I_i, UO_i, |SI|\}$
      if $SI = 0$ **then** break looping for;
      end if
    end for
  end if
  Return: $O_i, I_i, \forall i$.

$$\mathrm{UD}_i = \mathrm{rand}()\%3000 + 800; \tag{15}$$

$$\mathrm{seed1} = \mathrm{rand}()\%3000 + 500; \tag{16}$$

$$\mathrm{seed2} = \mathrm{rand}()\%5000 + 1000; \tag{17}$$

$$\mathrm{O} = \mathrm{N}/2 * \mathrm{seed1}; \tag{18}$$

$$\mathrm{UO}_i = \mathrm{rand}()\%(\mathrm{seed1} - 500) + 500; \tag{19}$$

$$\mathrm{I} = \mathrm{N}/2 * \mathrm{seed2}; \tag{20}$$

$$\mathrm{UI}_i = \mathrm{rand}()\%(\mathrm{seed2} - 1000) + 1000; \tag{21}$$

$$\mathrm{Z} = 100; \tag{22}$$

i	1	2	total
p_i	60	40	
UD_i	1000	500	
UO_i	600	600	1000
UI_i	3000	2000	3000
		Z	100

Table 1: Benchmark MPBPTMP 001

Randomly generated benchmarks were named RMPBPTMP N, being N the number of products. Seeking to enable the reproduction of the results, in the computational construction of the benchmarks we used the function srand((unsigned) source), where source is a defined value. To build the results presented in this work, we used source=0. The benchmarks used for the tests performed can be consulted at github.com/blinded.

i	1	2	3	total
p_i	60	40	50	
UD_i	1000	500	800	
UO_i	600	600	600	1500
UI_i	3000	2000	1000	3500
			Z	100

Table 2: Benchmark MPBPTMP 002

i	1	2	3	4	5	6	7	8	9	10	total
p_i	60	40	50	40	30	50	60	10	20	40	
UD_i	1000	500	800	500	400	500	2000	300	500	1000	
UO_i	600	600	600	1500	300	200	500	800	0	200	3000
UI_i	3000	2000	1000	800	3000	1000	400	300	200	0	5000
										Z	100

Table 3: Benchmark MPBPTMP 003

Table 4 presents the results obtained by applying the analytical method (C++ solver) and the LINGO solver for the solution of the previously presented benchmarks.

It is important to point out that the LINGO solver was developed with the purpose of validating the results found by the proposed analytical method. As the analytical method is a polynomial time complexity method, we did not intend to compare computational costs, however it is possible to verify that both solvers are capable of finding optimal solutions for the proposed benchmarks very quickly, even for very large problems.

We have considered here a one-day period problem, so in a future work we will study the complexities of solving a multi-period problem and try to propose new solution methods if needed.

problem	LINGO solver	time (s)	analytical method	time (s)
MPBPTMP 1	55	0.04	55	0.001
MPBPTMP 2	48	0.06	48	0.001
MPBPTMP 3	30	0.09	30	0.001
RMPBPTMP 20	100	0.08	100	0.001
RMPBPTMP 50	98	0.12	98	0.001
RMPBPTMP 100	98	0.12	98	0.002
RMPBPTMP 1,000	78	3.10	78	0.002
RMPBPTMP 10,000	70	168.87	70	0.006

Table 4: Results obtained with the LINGO solver and the analytical method

6. Conclusions and suggestions for future works

In this paper we presented the Multi-product Batch Processing Time Maximization problem, as well as a mathematical model and an analytical solution method for this problem. The mathematical model and analytical method were tested, respectively, by a solver developed with the LINGO software, from LINDO Systems, and with a solver developed in C++ language. Optimum results were found for all proposed benchmarks, very quickly, even in the case of very large problems, which demonstrates the efficiency of the proposed method. As in this paper we considered a planning period of one day, in a future work we will study the complexities that arise when considering the same problem in a multi-period scenario. We will also verify if there is the possibility of extending the proposed analytical method to solve a class of linear integer programming problems with similar characteristics to the studied problem.

7. CRediT authorship contribution statement

T.B. Fraga: Conceptualization, Project administration, Supervision, Software, Methodology, Validation, Formal analysis, Writing – original draft, Writing – review & editing. Í.R.B. Aquino: Data curation.

8. Acknowledgments

We are enormously grateful to Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) and to Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) for the financial support provided to our projects. We also thank LINDO systems team for the LINGO software license, without which this work would not have been possible and the to the owner of the company in the plastics sector, who allowed us to learn about his company's production process. Finally, we would like to thank Pró-reitoria de Extensão e Cultura da UFPE (PROExC) and the Research Director of Propesqi (Pró-reitoria de Pesquisa e Inovação da UFPE) for their support and recognition of our work, and dear Professors Antônio José da Silva Neto and João Flávio Vieira de Vasconcellos from IPRJ/UERJ, who contributed significantly to the formation of essential skills for the development of our projects.

References

Eilon. (1985). Multi-product batch production on a single machine - A problem revisited. *OMEGA Int. J. of Mgmt Sci.*, Vol. 13 (5), pp. 453–468.

Kim, M., Jung, J. H. and Lee, I. (1996). Intelligent scheduling and monitoring for multi-product networked batch processes. *Computers chem. Engn*, Vol. 20 (Suppl.), pp. 1149–1154.

Li, C., Wang, F., Gupta, J.N.D., Chung, T. (2022). Scheduling identical parallel batch processing machines involving incompatible families with different job sizes and capacity constraints. *Computers & Industrial Engineering*, Vol. 169, pp. 108115.

Liu, G., Li, F., Yang, X., and Qiu. S. (2020). The multi-stage multi-product batch-sizing problem in the steel industry. *Applied Mathematics and Computation*, Vol. 369, 124830.

Méndez, C.A., Henning, G.P., Cerdá, J. (2000). Optimal scheduling of batch plants satisfying multiple product orders with different due-dates. *Computers and Chemical Engineering*, Vol. 24, pp. 2223–2245.

OMEGA Journal. (1993). Single Machine Multi-product Batch Scheduling: Testing Several Solution Methods. *OMEGA Int. J. of Mgmt Sci.*, Vol. 21 (6), pp. 709–711.

Ravemark, D. E., and Rippin, D. W. T. (1998). Optimal design of a multi-product batch plant. *Computers chem. Engng*, Vol. 22 (1-2), pp. 177–183.

Shi, B., Qian, X., Sun, S., Yan, L. (2017). Rule-based scheduling of multi-stage multi-product batch plants with parallel units. *Chinese Journal of Chemical Engineering*, in press.