Submetido para TEMA

A new branching rule to solve the Capacitated Lot Sizing and Scheduling Problem with sequence dependent setups

Abstract. In this paper, we deal with the Capacitated Lot Sizing and Scheduling Problem with sequence dependent setup times and costs - CLSD model. More specifically, we propose a simple reformulation for the CLSD model that enables us to define a new branching rule to be used in Branch-and-Bound (or Branch-and-Cut) algorithms to solve this NP-hard problem. Our branching rule can be easily implemented in commercial solvers. Computational tests performed in 240 test instances from the literature show that our approach can significantly reduce the running time to solve this problem using a Branch-and-Cut algorithm of a commercial MIP solver. Therefore, our approach can also improve the performance of other approaches that need to solve partial sub problems of the CLSD model in each iteration, such as Lagrangian approaches and heuristics based on the mathematical formulation of the problem.

Keywords. Lot Sizing, Scheduling, Mixed Integer Linear Programming, Branch-and-Bound Algorithm.

1. Introduction

In most production environments, companies need to decide on the size of production lots in order to obtain efficient inventory management and reduce costs. High inventory levels cause high holding costs and low inventory levels may cause undesirable delays in meeting customer demands.

The lot sizing problem (LP) consists of determining the optimal size of production lots with the aim to minimize costs and meet customer demands. The LP has received special attention from researchers due to its importance for the global economy ([10]).

On the other hand, the scheduling problem consists of determining the sequence of production lots in order to minimize the time and cost generated by product changeovers on production lines. When the cost and time generated by product changeovers depends on previously produced items and the item to be produced, it can be said that there is a sequence-dependent setup time and/or cost structure.

According to [1], when there is a sequence-dependent setup time/cost structure, the decisions about the size of production lots and the sequence of production

need to be taken simultaneously, because the solution obtained by an hierarchical approach may be infeasible or suboptimal. Therefore, the simultaneous lot sizing and scheduling problem (LSP) consists of simultaneously deciding the sizes and production sequences.

In the literature, there are various mathematical models to deal with LSP. We highlight the capacitated lot sizing and scheduling with sequence dependent setups - CLSD model ([13]), the general lot sizing and scheduling problem - GLSP model ([9], [16]) and the reformulation of GLSP model proposed in [5] - CC model. Recent reviews of the models to deal with LSP are presented in [12, 1, 6].

In [12], the authors compare various mathematical models for LSP and by using theoretical and computational results, it could be concluded that the CLSD model is a promising formulation to deal with LSP. The CLSD model has an interesting performance in exact solution approaches such as Brand-and-Bound algorithms from commercial MIP solvers.

In this paper, we introduce a very simple reformulation for the CLSD model (the CLSD^w model) that allows us to introduce a new branching rule able to significantly improve the computational performance of this model in Branch-and-Bound (Branch-and-Cut) algorithms. Therefore, our approach can be used to improve the performance of commercial solvers and approaches to deal with LSP that need to solve partial sub problems, such as Lagrangian approaches and MIP based heuristics.

We use a set of 240 test instances from the literature to compare the performance of the traditional CLSD with our approach in a commercial powerful solver Cplex 12.60. Computational results show that our approach can significantly improve the performance of the solver Cplex, reducing running time and proving optimality for more test instances. This paper is organized as follows: Section 2 presents a literature review for LSP; Section 3 presents the mathematical formulation CLSD^w and the branching rule and Section 4 presents the computational results comparing the performance of the traditional CLSD model with our approach in the Branch-and-Cut algorithm from Cplex 12.60. Finally, the conclusions and future proposals are presented in Section 5.

2. Literature review

[9] introduced the GLSP model to deal with LSP. The GLSP model considered several items to be produced on a single machine (production line) with dynamic deterministic demand and sequence-dependent setup costs and no setup times. This model is based on the idea that consists of splitting to split each period into several micro periods (with varying sizes) where only one item can be produced. Therefore, by determining which items will be produced in each micro period, the production lot schedules can then be automatically determined.

The original GLSP model was reformulated in [20] using a network flow problem structure and in [5] suppressed the setup state variables in the model. Computational tests performed in [12] showed that both reformulations can provide better

dual bounds by solving linear relaxation than the dual bounds obtained by solving the linear relaxation of the GLSP model .

In [13], the CLSD model was introduced using another strategy, consisting of introducing constraints and variables from the travelling salesman problem to map the start time and end time of production of each item in each period, to model the sequencing decisions. The CLSD model considers, originally, a single-stage system where several items have to be produced on a single machine in a finite planning horizon supposing a known dynamic deterministic demand which must be completely satisfied without backlogging.

Mixed integer programming models based on the CLSD model have been proposed to deal with problems from various real world production environments, such as [15], which addressed a yogurt industry and [21], which studied a semiconductor assembly and test manufacturing.

In other papers, extensions of the GLSP model were compared with the extension of the CLSD model to deal with different real problems. For example, [2] addressed the LSP on parallel production lines that need to be equipped with tools for processing. Models inspired by GLSP and CLSD were developed to synchronize these tools on the production lines. Computational tests showed that the CLSD model has a much better performance than the GLSP model considering the runtime and the ability to find feasible solutions.

Computational tests in [12] showed that the CLSD model performs better than all the models that use micro period structures. In particular, the CLSD model presented a lower average deviation from the best known solution (GAP) and running time than other models.

The LSP is a *NP-complete* problem ([9, 16, 17]) where instances based on real world problems can be difficult to solve by exact algorithms at an acceptable computational time. Therefore, various heuristic approaches have been developed to deal with the LSP. For example, [13] proposed a backward oriented heuristic to solve the LSP model without setup times, while [17] proposed a threshold accepting methaheuristic to deal with LSP considering sequence dependent setup times.

Heuristics based on the mathematical formulation of the problem (MIP based heuristics) have been frequently used to solve the LSP, in particular, the *relax-and-fix - RF* and *fix-and-optimise - FO*. FO heuristics have obtained good solutions for various types of lot sizing and scheduling problems and various heuristics approaches combining RF and FO heuristics have been proposed in the literature, such as [3, 7, 8, 22, 19].

In the simplified case, the RF heuristic split the set of all binary variables (B) of the model in a finite number of small subsets ($B^r \subset B, r = 1, \ldots, R$) and, in each iteration $r \in \{1, \ldots, R\}$, the binary variables in the sets B^k with $k < r$ have their values fixed in the incumbent value (obtained from the previous iterations), while the binary variables in the sets B^l with $l > k$ are linearly relaxed and just the binary variables in the set B^r are optimized. Usually, RF heuristics are used to obtain an initial solution.

The FO is an improvement heuristic that starts in any feasible solution and in each iteration a subset of binary variables are re-optimized while the other binary

variables have their values fixed on the value of the incumbent solution. For lot sizing and scheduling problems, the most used variable partitions are by periods, products and production lines. FO heuristics can also be used with an exact method (matheuristics). For example, [11] combined FO with column generation and [4] integrated FO with the *Variable Neighbourhood Search - VNS* methaheuristic.

Therefore, as MIP based heuristics solve, in each iteration, a small mixed integer linear programming model, these heuristics can be improved if a good model and a good exact solution algorithm are used to solve the sub problems in each iteration. Therefore, the reformulation and the branch rule proposed in this paper can improve the computational performance of some MIP based heuristics to deal with LSP.

3. Model formulation and solution approach

The traditional CLSD model with setup times can be found in [12] and the following parameters and variables are used to define it:

Parameters:

- T: number of periods (indexed by t);
- J: number of items (indexed by i and j);
- d_{jt}: demand of item j in period t;
- C_t: available capacity time in period t;
- a_j: consumed capacity time for production of a unit of item j;
- h_j: inventory cost of item j;
- sc_{ij}: setup cost for exchange between items i and j;
- st_{ij}: setup time for exchange between items i and j;

Variables:

- I_{jt}: inventory of item j at the end of period t;
- x_{jt}: produced quantity of item j in period t;
- V_{jt}: production order of item j in period t;
- y_{js}: 1, if the item j is the first item produced in period t and 0, otherwise;
- z_{ijs}: 1, if there is an exchange between items i and j in period t and 0, otherwise.

The CLSD model is given by (3.1) to (3.9).

$$\text{Min} \quad \sum_{j=1}^{J}\sum_{t=1}^{T} h_j I_{jt} + \sum_{i=1}^{J}\sum_{j=1}^{J}\sum_{t=1}^{T} sc_{ijt} z_{ijt} \tag{3.1}$$

$$\text{subject to} \quad \sum_{j=1}^{J} a_j x_{jt} + \sum_{i=1}^{J}\sum_{j=1}^{J} st_{ij} z_{ijt} \leq C_t, \quad \forall t, \tag{3.2}$$

$$x_{jt} \leq \frac{C_t}{a_j}\left(y_{jt} + \sum_{i=1}^{J} z_{ijt}\right), \quad \forall j,t, \tag{3.3}$$

$$\sum_{j=1}^{J} y_{jt} = 1, \quad \forall t, \tag{3.4}$$

$$y_{jt} + \sum_{i=1}^{J} z_{ijt} = \sum_{i=1}^{J} z_{jit} + y_{j,t+1}, \quad \forall j, \tag{3.5}$$

$$V_{jt} \geq V_{it} + 1 - J(1 - z_{ijt}), \quad \forall i,j,t, \tag{3.6}$$

$$I_{jt}, V_{jt}, x_{jt} \geq 0, \quad \forall j,t, \tag{3.7}$$

$$y_{jt} \in \{0,1\}, \quad \forall j,t, \tag{3.8}$$

$$z_{ijt} \in \{0,1\}, \quad \forall i,j,t. \tag{3.9}$$

The objective function (3.1) reflects the sum of holding costs and product changeover costs, while (3.2) are the capacity constraints and constraints (3.3) ensure that item j can only be produced if the production line is set up for it. Constraints (3.4) ensure that only one item is the first produced item in each period, while constraints (3.5) trace the machine configurations. Constraints (3.6) are MTZ (Miller-Tucker-Zemlin) constraints to eliminate sub tours and, finally, constraints (3.7) - (3.9) define the domain of decision variables.

To introduce our branching rule, we firstly define a reformulation of the CLSD model called the CLSD^w model. Consider new binary variables w_{jt} to indicate if item j is produced in period t ($w_{jt} = 1$) or not ($w_{jt} = 0$). Clearly, we have that

$$w_{jt} = y_{jt} + \sum_{i=1}^{J} z_{ijt}, \quad \forall j,t. \tag{3.10}$$

The CLSD^w model can be obtained introducing constraints (3.10) and (3.12) and replacing constraints (3.3) by constraints (3.11), where:

$$x_{jt} \leq \frac{C_t}{a_j} w_{jt}, \quad \forall j,t, \tag{3.11}$$

$$w_{jt} \in \{0,1\}, \quad \forall j,t. \tag{3.12}$$

3.1. Branching rule

Note that by (3.10), if item j is not produced in period t, i.e, $w_{jt} = 0$, then $y_{jt} = 0$, $z_{ijt} = 0$ and $z_{jit} = 0$, $\forall i$. Therefore, if we can identify that item j is not produced in one period, we can fix directly the value of $2J + 1$ binary variables on zero.

This fact motivated us to introduce the binary variables w_{jt} in the CLSD model, obtaining the CLSD^w model, and performing a branch-and-bound algorithm with a priority of branching for these variables. Note that this reformulation increases the number of binary variables in $J * T$, however it enables us to perform a more efficient branching scheme in a branch-and-bound or branch-and-cut algorithm.

In each search tree node, given an optimal solution of linear relaxation, our branching rule is: if there are variables w_{jt} with no integer values, we firstly perform the branching in these variables before variables y_{jt} and z_{ijt}.

Figure 1 presents an example comparing a traditional branch-and-bound algorithm in the CLSD model and a branch-and-bound algorithm using the CLSD^w model with our branching rule. The instance considered in Figure 1 has 15 products and 5 periods and we notice that with just six nodes explored, the best dual bound found with our branching rule is significantly better than the best dual bound found in a traditional branching scheme. In this test instance, the traditional branch-and-bound algorithm consumed around 56 seconds to solve the CLSD model to optimality, while the branch-and-bound algorithm with the branching rule introduced in this paper consumed only around 6 seconds to solve the CLSD^w model.

We can observe that with some values of variables w_{jt} fixed in binary numbers, the value of linear relaxation for variables y_{jt} and z_{ijt} also tends to be binary. For example, consider a simplified case where just two items can be produced in each period and suppose that in a given node, the values of variables w_{it} and w_{jt} were fixed in one for some i, j and t. Suppose that $st_{ij} < st_{ji}$ and $sc_{ij} < sc_{ji}$. Therefore, in the optimal solution of the linear relaxation in this node, we will have $y_{it} = 1$, $y_{jt} = 0$, $z_{ijt} = 1$ and $z_{jit} = 0$.

Our branching scheme has another advantage. It can be easily implemented using a commercial solver, and therefore, we can benefit from a general powerful branch-and-cut algorithm and various general heuristics to improve the convergence.

In Section 4, we present computational results to compare the performance of the traditional CLSD model with the performance of the CLSD^w model using our branching scheme implemented in the Cplex 12.60 solver on 240 test instances from the literature.

4. Computational results

4.1. Test environment

We implemented the traditional CLSD model and the CLSD^w model in C++ language using the library Concert Technology of the Cplex 12.60 solver. Except for our branching rule for the CLSD^w model, we use the default setting of the Cplex 12.60.

Figure 1: Representation of branching rule.

We ran the tests on a computer with two Intel Xeon processors, 2.8 GHz and 128 GB DDR3 RAM memory. The maximum running time was fixed to one hour. For each instance, we captured the best feasible solution and the best dual bound found. The deviation of the best feasible solution from the lower bound (GAP) was computed as $GAP = 100 * \left(\frac{z^f - z^d}{z^f}\right)$, where z^f is the best feasible solution and z^d is the best dual bound found.

4.2. Test instance features and computational results

To test the performance of model CLSD^w with our branching rule, we used a set of 240 test instances from the literature, and compared it with the performance of traditional CLSD model. The test instances were presented in [14] and can be obtained in http://www.mang.canterbury.ac.nz/people/rjames. The test instances have the following features:

1. $J \in \{15, 25\}$, $T \in \{5, 10, 15\}$;
2. $h_j \in \{2, \ldots, 9\}$;
3. $d_{jt} \in \{40, \ldots, 59\}$;

4. $st_{ij} \in \{5, \ldots, 10\}$ and $sc_{ij} = \theta st_{ij}$, where θ is a positive parameter;

5. $a_j = 1$;

6. $C_t = \dfrac{\sum_j d_{jt}}{Cut}$, where $0 < Cut < 1$ is a parameter that defines the capacity utilization.

Another parameter $CutVar$ was introduced in order to control the amount of capacity variation. The parameter $CutVar$ represents and controls the maximum total allowed variation from Cut, and therefore the actual capacity can vary ([14]). The value for $CutVar$ was fixed to 0.5 for all the test instances as in [14].

The test instances were grouped into twenty four classes, with 10 test instances each class, according to the value of parameters J, T, Cut, $CutVar$ and θ. The values of parameters for each class and the results are given in Table 1 while the results grouped by number of products (J), number of periods (T) and capacity utilization (Cut) are presented in Table 2.

In Table 1, it can be observed that our approach was able to reduce the average GAP in 12 classes. Moreover, the obtained GAP by our approach does not increase in any class compared to the traditional approach. The general average GAP was reduced by around 56% ($\text{GAP}_{CLSD} = 0.39$ and $\text{GAP}_{CLSD^w} = 0.17$), while the general average time was reduced by around 24% ($\text{Time}_{CLSD} = 1068.63$ and $\text{Time}_{CLSD^w} = 801.77$).

The average time was reduced by 19 classes, remaining constant in 4 classes and increased in only one class (class 12). The increase is due to an instance from class 12 the random access memory limit (128 GB) was exceeded before reaching the time limit by the traditional CLSD model, and therefore, the running time was slightly reduced. The largest reduction in the average time occurred in classes 6, 11 and 16 where our approach reduces the mean running time by around 86%. In Table 2, it can be observed that our approach could prove optimality in 176 test instances while the traditional approach could prove optimality in 157 test instances.

Table 2 shows that when the number of products or periods increased, then the problem becomes more difficult to solve resulting in longer average deviations and running times, as well as a reduction in the number of instances considering optimality. This fact is not impressive, because the branch-and-cut algorithm grows exponentially when the number of decisions is increased.

Finally, in the Figure 2 we present a general comparison between the CLSD model and the CLSD^w model solved by the branch-and-cut algorithm of the Cplex solver. The x-axis represents the average GAP (in percentage) obtained from the 240 test instances and the y-axis represents the average running time (in seconds). Therefore, the closer to the origin, the more promising the solution's approach. It can be observed that the approach proposed in this paper improved the GAP (x-axis) and the running time (y-axis).

Table 1: Test instances features and computational results.

					CLSD		Our approach	
Class	J	T	Cut	θ	GAP	Time	GAP	Time
1	15	5	0.6	50	0.00	1.77	0.00	1.58
2	15	5	0.6	100	0.00	3.23	0.00	2.24
3	15	5	0.8	50	0.00	2.25	0.00	1.72
4	15	5	0.8	100	0.00	17.32	0.00	4.78
5	15	10	0.6	50	0.00	16.78	0.00	10.72
6	15	10	0.6	100	0.02	1199.49	0.00	167.24
7	15	10	0.8	50	0.00	53.96	0.00	17.74
8	15	10	0.8	100	0.65	3073.14	0.00	881.05
9	15	15	0.6	50	0.00	1061.70	0.00	38.27
10	15	15	0.6	100	1.12	3600.00	0.60	3600.00
11	15	15	0.8	50	0.03	910.96	0.00	115.94
12	15	15	0.8	100	2.25	3534.61	1.18	3600.00
13	25	5	0.6	50	0.00	16.39	0.00	16.84
14	25	5	0.6	100	0.00	22.88	0.00	17.47
15	25	5	0.8	50	0.00	34.19	0.00	18.43
16	25	5	0.8	100	0.00	569.32	0.00	77.59
17	25	10	0.6	50	0.05	2348.67	0.03	1379.36
18	25	10	0.6	100	2.12	3600.00	0.81	3600.00
19	25	10	0.8	50	0.19	2960.61	0.02	2091.43
20	25	10	0.8	100	2.95	3600.00	1.41	3600.00
21	25	15	0.6	50	0.01	731.67	0.00	495.04
22	25	15	0.6	100	0.72	3481.34	0.13	2811.65
23	25	15	0.8	50	0.01	1143.15	0.01	518.13
24	25	15	0.8	100	1.40	3600.00	0.50	3600.00
Average					0.39	1068.63	0.17	801.77

Figure 2: Comparing CLSD and CLSD^w model with branching rule introduced in this paper.

5. Conclusions and future studies

In this paper, we deal with the lot sizing and scheduling problem with sequence dependent setup costs and times. We proposed a simple reformulation for the

Table 2: Results by parameters.

	GAP		Time		Optimality	
	CLSD	CLSD^w	CLSD	CLSD^w	CLSD	CLSD^w
J						
15	0.34	0.15	1041.25	703.44	91	100
25	0.62	0.24	1842.35	1518.83	66	76
T						
5	0.00	0.00	83.42	17.58	80	80
10	0.75	0.28	2106.58	1468.44	40	54
15	0.69	0.30	2135.40	1847.38	37	42
Cut						
0.6	0.34	0.13	1258.64	1011.70	84	91
0.8	0.62	0.26	1624.96	1210.57	73	85

CLSD model originating the CLSD^w model. The CLSD^w model consists of explicitly specifying the binary variables (w) to indicate if an item is produced in one period or not. This formulation allowed us to define a new branching rule to improve the performance of branch-and-bound algorithms. Our branching rule consists of firstly performing the branching in variables w before the other binary variables. We implemented the CLSD^w model and our branching rule in the Cplex 12.60 solver, tested the performance of our approach in 240 test instances from the literature and compared them with the performance of the traditional CLSD model. The computational results show that our approach can significantly reduce the running time and the average GAP.

The branching rule proposed in this paper can improve the performance of algorithms that need to partially solve the CLSD model in each iteration, such as the mixed integer programming based heuristics and Lagrangian based heuristics. As future studies, we highlight the investigation of these approaches using the CLSD^w model.

Acknowledgements The authors would like to thank the following funding agencies for the financial support: Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), and Fundação de Amparo Pesquisa do Estado de São Paulo (FAPESP) via CEPID No. 2013/07375-0.

Resumo. Neste artigo tratamos do desafiador problema integrado de dimensionamento de lotes e sequenciamento da produção na existência de tempos e custos de preparação para produção dependentes da sequência. Mais especificamente, nossa atenção é fixada no modelo CLSD, proposto em [13]. Propõe-se, neste trabalho, uma reformulação para o modelo CLSD (intitulada CLSD^w), bem como, uma nova regra de *branching* para ser utilizada em algoritmos do tipo *Branch-and-Bound* para solução do modelo CLSD^w. Por meio de testes computacionais realizados com base em instâncias da literatura, foi possível observar que a abordagem de solução proposta neste artigo é bastante promissora, uma vez que proporcionou significante

redução no tempo computacional para solução problema, elevada redução no desvio médio (GAP) entre a melhor solução e o melhor limitante dual conhecidos e uma significante elevação no número de instâncias resolvidas até a otimalidade quando comparado com a abordagem tradicional.

References

[1] B. Almada-Lobo, et al. "Industrial insights into lot sizing and scheduling modeling". *Pesquisa Operacional*, 2015.

[2] C. Almeder and B. Almada-Lobo. "Synchronisation of scarce resources for a parallel machine lotsizing problem". *International Journal of Production Research*, v. 49, n. 24, p. 7315-7335, 2011.

[3] S. A. de Araujo, M. N. Arenales and A. R. Clark. "Lot sizing and furnace scheduling in small foundries". *Computers & Operations Research* , 35.3: 916-932, 2008.

[4] H. Chen. "Fix-and-optimize and variable neighborhood search approaches for multi-level capacitated lot sizing problems". *Omega*, 2015.

[5] A. R. Clark and S. J. Clark. "Rolling-horizon lot-sizing when set-up times are sequence-dependent". *International Journal of Production Research*, 38.10: 2287-2307, 2000.

[6] K. Copil, et al. "Simultaneous lotsizing and scheduling problems: a classification and review of models". *OR Spectrum*, 2016.

[7] D. Ferreira, R. Morabito, and Socorro Rangel. "Solution approaches for the soft drink integrated production lot sizing and scheduling problem". *European Journal of Operational Research*, 196.2: 697-706, 2009.

[8] D. Ferreira, R. Morabito, and Socorro Rangel. "Relax and fix heuristics to solve one-stage one-machine lot-scheduling models for small-scale soft drink plants". *Computers & Operations Research* , 37.4: 684-691, 2010.

[9] B. Fleischmann and H. Meyr. "The general lotsizing and scheduling problem". *Operations-Research-Spektrum*, v. 19, n. 1, p. 11-21, 1997.

[10] C. H. Glock, E. H. Grosse, and J. M. Ries. "The lot sizing problem: A tertiary study". *International Journal of Production Economics*, 155: 39-51, 2014.

[11] L. Guimaraes, D. Klabjan, and B. Almada-Lobo. "Pricing, relaxing and fixing under lot sizing and scheduling". *European Journal of Operational Research*, 230.2: 399-411, 2013.

[12] L. Guimarães, D. Klabjan, and B. Almada-Lobo. "Modeling lotsizing and scheduling problems with sequence dependent setups". *European Journal of Operational Research*, v. 239, n. 3, p. 644-662, 2014.

[13] K. Haase. “Capacitated lot-sizing with sequence dependent setup costs”. *Operations-Research-Spektrum*, 18.1: 51-59, 1996.

[14] R. J. W. James and B. Almada-Lobo. “Single and parallel machine capacitated lotsizing and scheduling: New iterative MIP-based neighborhood search heuristics”. *Computers & Operations Research*, 38.12: 1816-1825, 2011.

[15] G. M. Kopanos, L. Puigjaner and C. T. Maravelias. “Production planning and scheduling of parallel continuous processes with product families”. *Industrial & engineering chemistry research*, 50.3: 1369-1378, 2010.

[16] H. Meyr. “Simultaneous lotsizing and scheduling by combining local search with dual reoptimization”. *European Journal of Operational Research*, v. 120, n. 2, p. 311-326, 2000.

[17] H. Meyr. “Simultaneous lotsizing and scheduling on parallel machines”. *European Journal of Operational Research*, v. 139, n. 2, p. 277-292, 2002.

[18] H. Meyr and M. Mann. “A decomposition approach for the General Lotsizing and Scheduling Problem for Parallel production Lines”. *European Journal of Operational Research*, v. 229, n. 3, p. 718-731, 2013.

[19] W. Wei, et al. “Tactical production and distribution planning with dependency issues on the production process”. *Omega*, 2016.

[20] L. A. Wolsey. “MIP modelling of changeovers in production planning and scheduling problems”. *European Journal of Operational Research*, v. 99, n. 1, p. 154-165, 1997.

[21] J. Xiao, et al. “A hybrid Lagrangian-simulated annealing-based heuristic for the parallel-machine capacitated lot-sizing and scheduling problem with sequence-dependent setup times”. *Computers & Operations Research*, 63: 72-82, 2015.

[22] S. Çağrı and B. Bilgen. “Hybrid simulation and MIP based heuristic algorithm for the production and distribution planning in the soft drink industry”. *Journal of Manufacturing Systems* 33.3: 385-399, 2014.