Pesquisa Operacional (2019) 39(3): 471-496

Printed version ISSN 0101-7438 / Online version ISSN 1678-5142
www.scielo.br/pope
doi: 10.1590/0101-7438.2019.039.03.0471

CAPACITATED LOT SIZING AND SCHEDULING WITH ORDER ACCEPTANCE AND DELIVERY TIME WINDOWS: MATHEMATICAL MODEL AND A MIP-BASED HEURISTIC

Willy A. de Oliveira Soler[1*], Kelly C. Poldi[2] and Maristela O. Santos[3]

Received February 26, 2019 / Accepted October 08, 2019

ABSTRACT. This research addresses a lot sizing and scheduling problem inspired by a real-world production environment where the customers make advanced orders and the industry need to decide which orders will be accepted with the aim of maximizing the profit respecting the production capacity constraints. Orders are composed of different types of items which must be delivered within a given time interval and, moreover, such orders cannot be split. A mixed integer programming (MIP) model is proposed to represent the problem and a MIP-based heuristic is also proposed to deliver good solutions at an acceptable computational time. The heuristic is composed of three phases (construction, deterministic improvement and stochastic improvement phases) and combines relax-and-fix, fix-and-optimize, and iterative MIP based neighborhood search procedures. Computational tests are presented in order to study the efficiency of the proposed approaches.

Keywords: Lot sizing and scheduling, order acceptance, MIP-based heuristics.

1 INTRODUCTION

Nowadays, global market is becoming more competitive, so it is essential for industries to have more efficient production plans. In many industries, the integrated lot sizing and scheduling problem (LSP) is crucial since it deals with the determining of the size of the production lots and the sequence of these lots in several production units with limited resources aiming to provide a production plan that minimizes the costs incurred in the production process (Almada-Lobo et al. (2015),Guimarães et al. (2014),Copil et al. (2017)).

*Corresponding author – https://orcid.org/0000-0002-7467-2307.
[1]Institute of Mathematics, INMA, Federal University of Mato Grosso do Sul. Av. Costa e Silva, s/n, 79070-900, Campo Grande, MS, Brazil.
[2]Department of Applied Mathematics, IMECC, University of Campinas, R. Sergio Buarque de Holanda, 651, 13083-859, Campinas, SP, Brazil.
[3]Institute of Mathematics and Computer Sciences, ICMC, University of São Paulo, Av. Trabalhador São-Carlense, 400, 13566-590, São Carlos, SP, Brazil.
E-mails: willy.oliveira@ufms.br, poldi@unicamp.br, mari@icmc.usp.br

Usually, lot sizing and scheduling problems are represented by mixed integer optimization models and they are $\mathscr{NP}$-hard in the strong sense. Therefore, researchers have proposed customized exact and heuristic methods to find good feasible solutions for large sized test instances, sometimes based on real-world applications of the LSP, at acceptable computational time (see Akartunalı et al. (2016) and Brahimi et al. (2017)). The most important approaches to deal with the LSP and its variations are reviewed in Brahimi et al. (2017) and Copil et al. (2017).

In this paper, we deal with a LSP based on a Brazilian food industry that produces an extensive catalogue of manufactured products in a single production line. In this type of industry, we can observe the following main characteristics:

i) significant *inventory holding costs*, since items have to be stored in places with monitored temperature;

ii) *sequence dependent setup costs and times*, i.e., changeovers between different items cause costs associated with loss of production time and the times and the incurred costs depend on the sequence of production of the items;

iii) *demand choice flexibility:* the customers make advanced orders composed of various types of items. Usually, due to production capacity constraints, the company cannot satisfy all customers' orders, and then the decision makers need to decide whether each order will be accepted or not with the aim of maximizing the obtained profit. Not accepted orders are lost;

iv) *indivisible orders:* customers are not willing to receive their orders partially, i.e., if an order is accepted, all of its items have to be delivered at the same time;

v) *delivery time windows:* the accepted orders must be delivered in a time window composed by some adjacent production periods.

Usually, literature addressing LSP problems is able to deal with the first two characteristics (see Guimarães et al. (2014); Oliveira & Santos (2017)). However the last three features are not common in literature and, to the best of our knowledge, there is no research in the literature simultaneously addressing all of the five characteristics considered in paper (the related works are detailed in Section 2). In this article we describe, model, and propose specific solution approaches to deal with the lot sizing and scheduling problem considering demand choice flexibility, delivery time window, and indivisible orders (LSP-DTI). The main contributions of this paper are the following:

- we extend the model introduced in Haase (1996) (and reformulated in Oliveira & Santos (2017)) to deal with the LSP-DTI;
- we propose a MIP based heuristic procedure, composed of three phases (construction, deterministic improvement, and stochastic improvement), to provide good solutions for the LSP-DTI at an acceptable computational time.

The researches presented in Furtado (2012), Sereshti & Bijari (2013), and da Silva et al. (2014) describe production environments that consider some of the characteristics addressed in this paper, for example, foundry industry, moquette weaving industry, and baking industry. Therefore, we believe that the model and methods proposed in this paper can be easily adapted for a range of industries, such as those previously mentioned.

The paper is organized as follows: In Section 2, we present the literature review emphasizing the works that addressed similar problems and we detail the description of the addressed problem, while in Section 3 we present a mixed integer programming model for the LSP-DTI and study its computational complexity. In Section 4 we propose a MIP based heuristic procedure to deliver good solutions for the studied problem and the computational results to study the efficiency of the proposed approaches (model and heuristic) are presented in Section 5. Finally, in Section 6 it is presented some conclusions and various research directions are appointed as future studies.

2 RELATED RESEARCHES AND PROBLEM DESCRIPTION

The lot sizing and scheduling problem (LSP) consists in simultaneously determining how many of each product must be produced and the sequence in which the products are produced in each production period of a finite planning horizon. The aim of the problem is to ensure the fulfillment of the customer demands and minimizing the costs incurred in the production process usually represented by inventory holding and setup costs.

Traditionally, researches addressing the LSP adopt the assumption that customers' demands should be fully met. However, as observed in Sereshti & Bijari (2013), in a business with a goal of maximizing profit, satisfying all potential demands may not be an optimal solution. Therefore, Sereshti & Bijari (2013) addressed a LSP considering the demand choice flexibility assumption. In this problem, the industry can decide which demands should be accepted in each period with the aim of maximizing the profit obtained with the accepted demands discounting the inventory holding and sequence dependent setup costs. The authors proposed two mixed integer programming models for an appropriate selection of demands to be met. This problem differ of the problem addressed in this paper, because the inventory is controlled "item by item", i.e., possible customers' orders containing various types of items are disaggregated by type of item (the indivisible orders assumption is not considered). Besides that, the time windows assumption is also not considered.

In Aouam et al. (2018), a production planning problem with demand choice flexibility and uncertain demand was addressed. In this problem, the orders are indivisible and the authors present two operational reasons for potentially rejecting an order in environments with limited production capacity:

- economies of scale: it might be more profitable for the company to reject an order if it requires high setup cost and cannot be aggregated with additional orders to justify the production setup; and

- congestion effects on the production lines: the authors argue that the lead time depends on the workload. Therefore, the more orders are accepted the higher are the production lead times, resulting in the possibility of missing customers due dates.

In Aouam et al. (2018), the scheduling part of the problem was not addressed, but the demand choice flexibility was integrated with lot sizing decisions and with the load-dependent lead times problem in two different models. The problems were formulated as mixed integer programming models and relax-and-fix and fix-and-optimize heuristics were applied to provide good solutions for the problems.

Supithak et al. (2010) addressed a lot sizing and scheduling problem with earliness, tardiness and setup penalties. In this problem, customers make indivisible orders and the industry must produce and deliver such orders. Each order has its own due date, but the industry can deliver the orders after the due date through a penalty in the objective function. Besides that, Supithak et al. (2010) adopted the follow assumptions: i) each order is composed by only one type of item; ii) in each period just one item can be produced; iii) the customers' demands are given in integer number of production batches. We highlight that the assumptions (i), (ii), and (iii) are so restrictive for our considered production environment (food industry), so that, the problem addressed in this paper can be viewed as a generalization of the problem addressed in Supithak et al. (2010). Moreover that, Supithak et al. (2010) do not consider the demand choice flexibility assumption.

Yang et al. (2017) also addressed a lot sizing and scheduling problem with indivisible orders. In this problem, all orders have to be produced (demand choice flexibility is not considered) and the aim is to minimize the completion time. It was not considered setup times and costs. A mixed integer programming model was presented and four heuristic solution approaches commonly used for solving the bin packing problem were adapted to solve the problem. Computational results showed that finding optimal solutions for instances with more than 60 orders may be challenging from computational perspective. Jaruphongsa & Lee (2008) consider a lot sizing problem with delivery time windows, container-based transportation costs and no decisions about the production sequence. They showed that the problem is NP-hard if each demand must be satisfied by exactly one delivery.

Regarding the solution methods used in this paper, in order to solve the studied problem, we highlight the paper Toledo et al. (2015) that has combined relax-and-fix (RF) and fix-and-optimize (FO) heuristics to obtain high quality feasible solutions for multi-level lot sizing problems (MLP). In Toledo et al. (2015) these heuristics were tested in a large number of test instances from the literature for the MLP and the computational results have indicated that the combination of RF with FO is very efficient and competitive, outperforming some genetic algorithms. Moreover, the authors have proposed a (novel) general and efficient way to define the RF and FO heuristics. Besides that, recently, Soler et al. (2019) proposed a RF heuristic to solve a lot sizing and scheduling problem on parallel machines that it is able to provide competitive dual bounds for the problem and high quality feasible solutions. In the procedure introduced in Soler et al. (2019) the set of binary variables was decomposed considering the particular structure of the

problem. Firstly, the decisions regarding the lot sizing aspect were taken into account together with decisions about which production lines to assemble. Secondly, fixing the decisions about the assembled production lines, the lot sizing aspect is newly taken into account, but now, together with the scheduling decisions.

In this paper, we address the integration of the demand choice flexibility (or order acceptance) with the simultaneous lot sizing and scheduling problems considering two additional features: indivisible orders and delivery time windows. This problem is an extension of some problems addressed in the literature as we have reported in the beginning of this section and it is based in a production system usually adopted by food industries, more specifically companies that produce meat products.

In these companies, the catalogue of products is very extensive including meat from several types of animals. In order to avoid contamination, it is necessary to perform some procedures when there is exchange of products in the production line. These procedures depend of the sequence in which the products are produced in the line, i.e., there are sequence dependent setup times and costs. Besides that, the items need to be stored in places with monitored temperature implying in significant inventory holding costs.

Usually, customers are restaurants and supermarkets. These clients make advanced orders composed by several types of products and they are not willing to receive their orders partially, because they need to ensure that their menu/catalogue is completely available all the time. Each order has its due date that consists of some adjacent production periods. Sometimes, the company needs to reject customers' orders due to capacity constraints. Note that the capacity constraints are complicated by the sequence dependent setup times.

3 MATHEMATICAL FORMULATION AND COMPLEXITY OF THE PROBLEM

In this paper, we extend the CLSD model proposed in Haase (1996) to represent the LSP-DTI. The reformulation of the CLSD model presented in Oliveira & Santos (2017) is also incorporated in order to obtain a stronger formulation.

The LSP-DTI consists in determining simultaneously: i) the customers' orders that will be accepted, ii) the period (in the respective time window) in which each accepted order will be delivered, iii) the sequence in which the lots will be produced in each period; and iv) the size of the production lots. The aim is to maximize the profit obtained by the accepted orders discounting the costs incurred by inventory holding and sequence dependent setups.

The parameters and variables are described in Table 1 and the proposed model (CLSD-DTI) consists in maximizing the objective function (1) subject to constraints (2) to (9). The proposed CLSD-DTI model is, then, given by:

$$\max \quad \sum_{n=1}^{N}\sum_{t=F_n}^{L_n} P_{nt}\cdot\gamma_{nt} - \sum_{t=1}^{T}\sum_{j=1}^{J} h_j\cdot I_{jt} - \sum_{t=1}^{T}\sum_{i=1}^{J}\sum_{j=1}^{J} sc_{ij}\cdot z_{ijt} \tag{1}$$

$$\text{s. t.} \quad I_{j,t-1} + x_{jt} = \sum_{n\in O_t} q_{jn}\cdot\gamma_{nt} + I_{jt}, \quad \forall j,t; \tag{2}$$

$$\sum_{t=F_n}^{L_n} \gamma_{nt} \leq 1, \quad \forall n; \tag{3}$$

$$\sum_{j=1}^{J} a_j\cdot x_{jt} + \sum_{i=1}^{J}\sum_{j=1}^{J} st_{ij}\cdot z_{ijt} \leq C_t, \quad \forall t; \tag{4}$$

$$x_{jt} \leq \frac{C_t}{a_j} w_{jt}, \quad \forall j,t; \tag{5}$$

$$w_{jt} = y_{jt} + \sum_{i=1}^{J} z_{ijt}, \quad \forall j,t; \tag{6}$$

$$\sum_{j=1}^{J} y_{jt} = 1, \quad \forall t; \tag{7}$$

$$y_{jt} + \sum_{i=1}^{J} z_{ijt} = \sum_{i=1}^{J} z_{jit} + y_{j,t+1}, \quad \forall j,t; \tag{8}$$

$$V_{jt} \geq V_{it} + 1 - J(1 - z_{ijt}), \quad \forall i,j,t. \tag{9}$$

The objective function (1) consists in maximizing profit obtained by the accepted orders discounting the sum of inventory holding and setup costs. Constraints (2) are the inventory balance constraints, and Constraints (3) ensure that the orders are delivered at most once within its time windows. Constraints (4) are capacity constraints, while Constraints (5) ensure that a product can only be produced if the respective production is set up. Constraints (6) ensure that all produced items are sequenced in the respective period, and Constraints (7) establish that exists only one first produced item by period. Constraints (8) ensure the flow balance for the sequencing of lots, and (9) are the MTZ constraints to eliminate sub-tours.

Note that, by Constraints (3) each order can be met or not within its time windows given us the demand choice flexibility. On the other hand, Constraints (2) ensure that if an order is accepted, then the amount of each item that composes this order need to be available, hence we obtain the indivisible orders assumption. Finally, the delivery time windows assumption is obtained when we define the variables γ_{nt} only for $t \in \{F_n, \ldots, L_n\}$ together with Constraints 3.

In Aouam et al. (2018), it shown that the integration of the order acceptance flexibility in the lot sizing problem results in a $\mathscr{NP}$-hard problem. The LSP-DTI is also $\mathscr{NP}$-hard as stated in the Proposition 1.

Proposition 1. *The LSP-DTI is $\mathscr{NP}$-hard.*

Table 1 – Parameters and variables.

Parameters	
N, T	Number of orders (indexed by n) and number of periods (indexed by t)
J	Number of items (indexed by i and j)
F_n, L_n	First and last periods of the delivery time window of order n
P_{nt}	Profit of order n in period $t \in \{F_n, \ldots, L_n\}$
O_t	Set of orders that can be delivered on period t, i.e., $O_t = \{n \mid F_n \leq t \leq L_n\}$
q_{jn}, C_t	Amount of item j in order n and production capacity of period t
h_j, a_j	Unitary inventory holding cost and unitary processing time of item j
sc_{ij}, st_{ij}	Setup cost and setup time for changeover from product i to j
Variables	
$\gamma_{nt} \in \{0,1\}$	1 if order n is delivered in period $t \in \{F_n, \ldots, L_n\}$, and 0 otherwise
$w_{jt} \in \{0,1\}$	1 if item j is produced in period t, and 0 otherwise
$y_{jt} \in \{0,1\}$	1 if item j is the first item produced in period t, and 0 otherwise
$z_{ijt} \in \{0,1\}$	1 if there is change of production from item i to item j in t, and 0 otherwise
$x_{jt} \geq 0$	Amount of item j produced in period t
$I_{jt} \geq 0$	Inventory of item j by the end of period t
$V_{jt} \geq 0$	Variables to represent the order of production of item j in period t

Proof. In order to show that the LSP-DTI is $\mathscr{NP}$-hard, we will show that every instance of the traditional lot-sizing and scheduling problem with sequence dependent setup times and costs (LSP) can be described as an instance of the LSP-DTI.

Let β be an instance of the LSP, where d_{jt} indicates the demand of the product j due to the period t and the other necessary parameters to describe β (setup costs and times, inventory holding costs, number of products and periods, capacity, and processing times) are the same indicated in Table 1. With the aim of defining an instance β' of the LSP-DTI, we will fix the additional parameters (number of orders, time windows of the orders, products that compose each order, and profit of the orders) as follows:

- $N = T$. The number of orders is the number of periods;
- $F_n = L_n = n, \forall n \in \{1, \ldots, N\} = \{1, \ldots, T\}$. Each order is due to only one period that coincides with the number of the order. Therefore, we also have that $O_t = \{t\}, \forall t = 1, \ldots, T$;
- $q_{jn} = d_{jn}, \forall j, n \in \{1, \ldots, N\} = \{1, \ldots, T\}$. Each order is composed by the demand of the corresponding period;

- $P_{nt} = 0, \forall n,t$. The profit of each order is equal to zero, i.e., we just want to minimize the incurred production costs.

Moreover, we fix $\gamma_{nt} = 1, \forall n$ and $t \in \{F_n, \ldots, L_n\} = \{n\}$ to indicate that every order must be met in its due date. Under these conditions, solving the instance β of the LSP is equivalent to solving the instance β' of the LSP-DTI. Therefore, as the LSP is $\mathscr{NP}$-hard (Fleischmann & Meyr (1997)), the LSP-DTI is also $\mathscr{NP}$-hard. □

4 MIP HEURISTIC PROCEDURES

Computational results observed in the literature regarding simpler problems than the LSP-DTI (e.g., Sereshti & Bijari (2013)) indicate that Branch-and-Cut algorithms from commercial solvers are not able to provide good solutions at an acceptable computational time. This fact is confirmed for the LSP-DTI by the results presented in Section 5. Therefore, we propose a heuristic procedure to solve the LSP-DTI. The proposed solution approach is composed of three phases. The first phase consists of a relax-and-fix procedure to quickly construct an initial feasible solution. The second phase consists of a deterministic fix-and-optimize procedure to improve the solutions constructed in the first phase. Finally, in the third phase we apply a stochastic improvement procedure, that consists of an Iterative MIP based Neighborhood Search - INS (proposed in James & Almada-Lobo (2011)) to scape of possible bad local optima solutions.

4.1 First phase: a relax-and-fix procedure

Relax-and-fix heuristics (RF) have been extensively used in the literature to build solutions for production planning problems (see Ferreira et al. (2009) James & Almada-Lobo (2011), Guimaraes et al. (2013), Sel & Bilgen (2014), Toledo et al. (2015), and Aouam et al. (2018)). In each iteration of a RF heuristic a small MIP problem is solved where: (i) some binary variables have their values fixed in the incumbent values (obtained from the previous iterations); (ii) some binary variables are linearly relaxed; and (iii) only few of the binary variables are enforced to be integer.

A detailed framework for the relax-and-fix heuristic can be found in Toledo et al. (2015). In order to build initial feasible solutions for the LSP-DTI, we investigate RF algorithms that decompose the set of the binary variables of the problem by periods. Next, we present our RF procedure using the framework introduced in Toledo et al. (2015).

Let $\Theta = (\theta_1, \theta_2, \ldots, \theta_T)$ be a vector of length T where each entry θ_t is the set of binary variables associated with period t, i.e.,

$$\theta_t = (\gamma_{1t}, \ldots, \gamma_{Nt}, w_{1t}, \ldots, w_{Jt}, y_{1t}, \ldots, y_{Jt}, z_{11t}, \ldots, z_{1Jt}, \ldots, z_{J1t}, \ldots, z_{JJt}).$$

Moreover, let *window* be the set of contiguous periods in which the binary variables associated with them are enforced to be integer, *windowSize* the number of periods inside the *window*,

overlap the overlap rate of periods that are re-optimized in each iteration, and *timeLimit* the maximum running time of the RF procedure.

Initially, all binary variables in the RF solution (sol.Θ) are linearly relaxed. A *window* is defined as a set that includes a fixed amount (*windowSize*) of contiguous periods. In the first RF iteration, the binary variables referring to the periods inside the *window* are enforced to be integer (set Θ_{MIP}), while the others binary variables are kept linearly relaxed (set Θ_{LP}). The resulting MIP is then solved and sol.Θ is updated. Next, in the second iteration, a new *window* is defined and the subset of binary variables that left the (previous) *window* are fixed in the incumbent values (set Θ_{fix}), while the another variables are kept in the set Θ_{LP}. The resulting subproblem is then solved and sol.Θ is updated. The algorithm continues processing this way until all variables are fixed, i.e., the algorithm is interrupted when $\Theta_{fix} = \Theta$.

In each iteration, the *window* is updated moving *step* periods forward with $step = (1 - overlap) * windowSize$, where $overlap \in [0,1]$. Figure 1 presents the framework of our RF heuristic considering $T = 5$, $windowSize = 2$, $overlap = 0.5$, and $step = 1$.

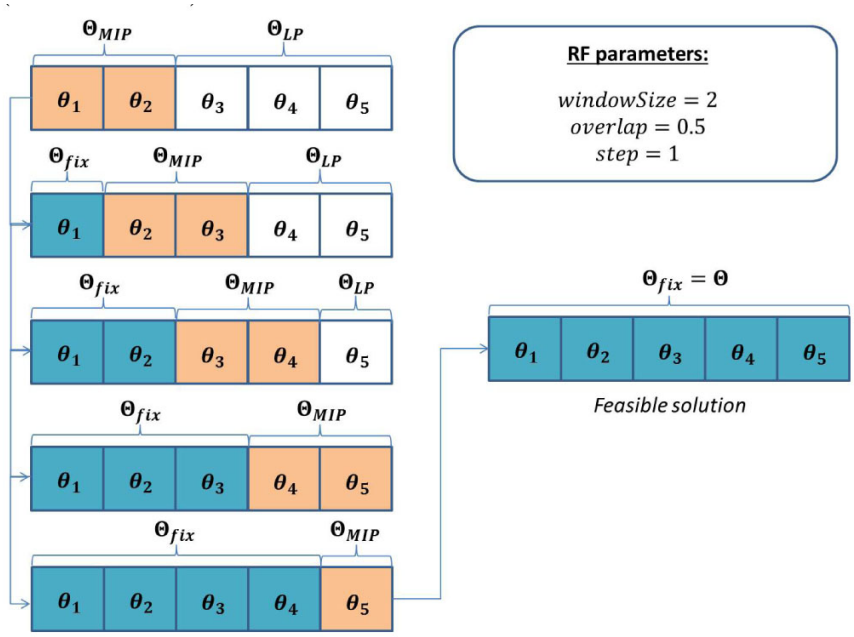

Figure 1 – Framework of the RF heuristic considering $T = 5$, $windowSize = 2$, $overlap = 0.5$, and $step = (1 - overlap) * windowSize = 1$.

We highlight that the demand choice flexibility assumption ensures that our relax-and-fix heuristic can always find a feasible solution for the problem, once the orders can be rejected, if necessary. In Section 5 we present computational results setting different values for parameters *windowSize*, *overlap*, and *timeLimit*.

4.2 Second phase: a deterministic fix-and-optimize procedure

The second phase of the proposed heuristic consists of a fix-and-optimize procedure. This heuristic starts with an initial feasible solution on hand and in each iteration a subproblem is solved optimizing just a subset of the binary variables of the original problem (and all possible continuous variables), while the other binary variables have their values fixed in the incumbent value. The incumbent solution is updated only when a better solution is found.

In our heuristic framework we use the initial feasible solution obtained on phase I to start the fix-and-optimize (FO) - phase II. As phase I builds solutions (successively) optimizing binary variables referring to adjacent periods, in the phase II we propose a different criterion to choose the periods to be optimized on each iteration, in order to diversify the search strategy.

More specifically, we propose a fix-and-optimize heuristic named FOTO (fix-and-optimize with total options of two periods combinations) that consists in optimizing, in each iteration, the binary variables regarding to two production periods, so that at the end of the procedure, all possible two periods combinations are tested. In the first iteration, we optimize the binary variables referring to the periods $t = 1$ (set Θ_1) and $t = 2$ (set Θ_2), while the other binary variables are fixed in the incumbent value. In the second iteration, we optimize the binary variables in the sets Θ_1 and Θ_3 (referring to the periods $t = 1$ and $t = 3$) and so on until iteration $\alpha = T - 1$ where the binary variables referring to the periods $t = 1$ and $t = T$ are optimized. Thereafter, in the iteration $\alpha = T$, we optimize the variable referring to the periods $t = 2$ and $t = 3$ and so on. In this way, the FOTO procedure is composed by $\frac{T(T-1)}{2}$ iterations. However, we highlight that the user can specify a maximum running time *timeLimit* for the FOTO heuristic and then the algorithm is interrupted when the running time exceed *timeLimit* (sometimes, before reaching $\frac{T(T-1)}{2}$ iterations). The continuous variables $x_{jt}, V_{jt}, \forall j,t$, and $I_{jt}, \forall j,t$ are optimized in all iterations.

The design of the FOTO heuristic is motivated by two main facts:

i. *Small subproblems.* In general, a subproblem that considers only the binary variables referring to two production periods can be quickly solved as shown in the computational results presented in Section 5;

ii. *Flexibility to change the set of orders that are meet in an initial feasible solution.* For example, let $n_1, n_2 \in \{1,\ldots,N\}$ be two different customer orders with delivery time windows $F_{n_1},\ldots,L_{n_1}$ and $F_{n_2},\ldots,L_{n_2}$, respectively. Moreover, suppose that $F_{n_2} > L_{n_1} + 1$. Now, consider an incumbent feasible solution (sol.Θ) in which the customer order n_1 is meet but n_2 is not meet. In this case, with the aim of to find a better feasible solution, it can be interesting to evaluate the possibility of meet the order n_2 rather than n_1 (i.e., to use the production capacity to produce order n_2 instead n_1). However, if a fix-and-optimize heuristic optimizes only binary variables from two adjacent periods this possibility can not be evaluated. Hence, in our FOTO procedure, we consider all two periods combinations.

Algorithm 1 presents a pseudo-code for the FOTO heuristic. The heuristic is interrupted if all possible two periods combinations have been optimized or if the maximum running time *timeLimit*

has been reached. In the line 8 of the Algorithm 1, the function solve(·) solves the incumbent subproblem optimizing just the binary variables in the set Θ_{MIP} (and all continuous variables), while the binary variables from the set Θ_{fix} are fixed in the incumbent values (according to sol.Θ). The solution obtained solving the subproblem is represented by new.Θ. The solution new.Θ is accepted as the new incumbent solution only if its profit is higher than the incumbent profit.

Algorithm 1 Phase II - FOTO heuristic

1: Given an initial solution sol.Θ;
2: Set time = 0;
3: Set $t1 \leftarrow 1$;
4: **while** time $<$ *timeLimit* and $t1 < T$ **do**
5: **for** $t2 = t1+1, \ldots, T$ **do**
6: $\Theta_{MIP} = \Theta_{t_1} \cup \Theta_{t_2}$;
7: $\Theta_{fix} = \Theta \setminus \Theta_{MIP}$;
8: solve(sol.Θ, Θ_{MIP}, Θ_{fix}, new.Θ);
9: **if** f(new.Θ) $>$ f(sol.Θ) **then**
10: sol.$\Theta \leftarrow$ new.Θ;
11: **end if**
12: Update time;
13: **end for**
14: $t1 \leftarrow t1+1$;
15: **end while**
16: **return** sol.Θ;

4.3 Third phase: a stochastic MIP procedure

The Iterative MIP-based Neighborhood Search (INS) heuristic was proposed in James & Almada-Lobo (2011) with the aim of solving a lot sizing and scheduling problem with sequence dependent setup times and costs. Just as RF and FOTO, the INS heuristic also decomposes the problem into smaller subproblems that can be more easily solved by using mathematical programming techniques, and just as variable neighborhood search (VNS) meta-heuristic, the INS uses a set of more than one neighborhood structure. Therefore, INS procedure starts with a feasible solution on hand and in each iteration a neighborhood search is performed according to a incumbent neighborhood structure. Each iteration of the INS heuristic is composed of two main steps:

i) neighborhood search step; and

ii) neighborhood structure update step.

As defined in James & Almada-Lobo (2011), the neighborhood search step can be viewed as a fix-and-optimize heuristic, while the neighborhood structures consist of rules to determine the

subset of binary/integer variables that are optimized in each iteration within the neighborhood search step. The Algorithm 2 presents a pseudocode for the INS procedure used in this paper.

Algorithm 2 INS heuristic

1: Given m neighborhood structures $NS_1, \ldots, NS_M$, an initial solution sol.Θ and a maximum number of iterations $R_{\max}$;
2: Set stop = false;
3: Set $r \leftarrow 0$;
4: **while** stop = false and $r < R_{\max}$ **do**
5: Set $k \leftarrow 1$;
6: **while** stop = false and $k < M$ **do**
7: sol.$\Theta^{'}$ = Stochastic Fix-and-Optimize(sol.Θ, NS_k);
8: **if** $f(\text{sol}.\Theta^{'}) > f(\text{sol}.\Theta)$ **then**
9: sol.$\Theta \leftarrow$ sol.$\Theta^{'}$;
10: $k \leftarrow 1$;
11: **else**
12: $k \leftarrow k+1$;
13: **end if**
14: Update stop;
15: **end while**
16: $r \leftarrow r+1$;
17: **end while**
18: **return** sol.Θ;

In the Algorithm 2, the loop defined in lines 4 to 17 is the major iteration, and the loop defined in lines 6 to 15 is the minor iteration. Line 7 consists of the neighborhood search step, i.e., a fix-and-optimize heuristic in which the variables to be optimized are determined according to the incumbent neighborhood structure (NS_k) and lines 8 to 13 define the neighborhood structure update step.

If Stochastic Fix-and-Optimize(sol.Θ, NS_k) returns a solution sol.$\Theta^{'}$ with greater profit than the incumbent solution, then we update the incumbent solution and in the next (minor) iteration the first neighborhood structure is explored. Otherwise, in the next (minor) iteration, we explore the next neighborhood structure (if possible).

A new major iteration is started when all available neighborhood structures have been explored. In James & Almada-Lobo (2011), it is highlighted that the user of the INS must specify the stop criteria and the neighborhood structures. We adopted maximum running time reached and maximum number of major iterations performed as stop criteria.

4.3.1 Neighborhood search step: Stochastic Fix-and-Optimize heuristic

In our INS approach, the neighborhood search step consists of a stochastic fix-and-optimize heuristic in which the set of variables to be optimized in each iteration is randomly selected according to the incumbent neighborhood structure. Algorithm 3 presents a pseudocode for the Stochastic Fix-and-Optimize heuristic used in this paper. In Algorithm 3, α is the counter of iterations, and stag is the number of consecutive iterations without improvement in the quality of the solution.

Algorithm 3 Stochastic Fix-and-Optimize

Given a feasible solution sol.Θ, a neighborhood structure NS, and a maximum number of iterations without improvement S;
Set $\alpha \leftarrow 0$;
Stop $\leftarrow$ False;
while Stop = False and stag $< S$ **do**
  Define the set Θ_{MIP} according to the neighborhood structure NS;
  $\Theta_{fix} = \Theta \setminus \Theta_{MIP}$;
  Solve($\Theta_{MIP}, \Theta_{fix}$) obtaining a solution sol.Θ^{α};
  if $f(\text{sol.}\Theta^{\alpha}) > f(\text{sol.}\Theta)$ **then**
    sol.$\Theta \leftarrow$ sol.Θ^{α};
    stag $\leftarrow 0$;
  else
    stag $\leftarrow$ stag + 1;
  end if
  Update Stop;
  $\alpha \leftarrow \alpha + 1$;
end while
return sol.Θ;

The Stochastic Fix-and-Optimize procedure is interrupted either when S consecutive iterations are performed without finding a better solution; or when the maximum running time is reached; or when the incumbent structure NS_k has been fully exploited.

4.3.2 Neighborhood structures

We used neighborhood structures that consist in decomposing the original problem by periods (similarly to James & Almada-Lobo (2011)) and by customers' orders. In phase II we optimize binary variables regarding two production periods and we probably find a local maximum solution. Therefore, in phase III, we propose neighbourhood structures that allow increasing the number of periods to be optimized in each iteration in order to escape of this possible local maximum solution.

The number of neighborhood structures (M) depends on the number of production periods (T) of the test instance. For the test instances proposed in Section 5, we empirically adopted $M = \max\{3, \lfloor \frac{T}{3} \rfloor\}$.

In general, the neighborhood structure NS_m, with $m = 1, \ldots, M-1$, consists of randomly select, in each iteration of the neighborhood search step (Stochastic Fix-and-Optimize), a set of $m+1$ consecutive periods to have their binary variables optimized, while the other binary variables have their value fixed in the value of the incumbent solution. Initially, all sets of consecutive periods have the same probability of being selected, however, these probabilities reduce according to the frequency in which the periods have been selected in the incumbent Stochastic Fix-and-Optimize. Let f_t be the frequency in which the set of periods $t, t+1, \ldots, t+(m-1)$ has been selected in the incumbent Stochastic Fix-and-Optimize, the probability of accepting the set of periods $t, t+1, \ldots, t+(m-1)$ in each iteration is given by $e^{-\frac{f_t}{\lambda}}$, where λ is a greater than 1 real parameter that was empirically determined. Furthermore, it is not allowed that a same period be selected in two consecutive iterations.

Note that, in the INS algorithm, when a better feasible solution is found we return to explore the first neighbourhood structure. Therefore, it is interesting that the first structure results in subproblems that can be quickly solved, while the other structures should be able to escape of possible local optima solutions found by the first structure. Therefore, our first neighbourhood structure consists in optimizing the binary variables referring to only two periods, while in the other structures we increase the number of periods to have their binary variables optimized in each iteration.

Finally, in the structure NS_M, in each Stochastic Fix-and-Optimize iteration, we randomly select a customer order n and optimize all binary variables referring to order $n^{'}$ in which $\{F_{n'}, \ldots, L_{n'}\} \subset \{F_n, \ldots, L_n\}$. Similarly to other neighborhood structures, the probability of selecting the order n in each Stochastic Fix-and-Optimize iteration is given by $e^{-\frac{g_n}{\lambda}}$, where g_n is the frequency in which the order n has been selected in the incumbent Stochastic Fix-and-Optimize.

We observe that the NS_M structure usually provides subproblems larger than the ones obtained by the other proposed structures. On one hand, NS_M structure can be able to escape from possible local optima solutions found by the previous structures; on the other hand, it requires greater computational effort to be solved. Therefore, this is the last structure to be considered in the solution process.

5 COMPUTATIONAL RESULTS

5.1 Test data and test environment

In order to evaluate the effectiveness of the proposed approaches, we propose a set of 100 test instances with a broad range of scenarios. As the real-world data cannot be disclosed due to confidentiality reasons, we randomly generated the test instances combining the parameter settings we observed in technical visits with the approach in Sereshti & Bijari (2013).

Using the notation $p \in \mathrm{U}\left[a^1, a^2\right]$ to indicate that the value of the parameter p was randomly selected in the set $\{a^1, \ldots, a^2\}$, the test instances were generated according to the following characteristics:

- Number of periods: $T \in \{5, 10, 15\}$. In practice, demands are known from one to three weeks in advance, hence we use $T \in \{5, 10, 15\}$ considering that the companies we have visited operate five days per week;
- Number of orders: $N \in \{30, 50, 60, 100, 150\}$. The number of considered orders depends on the length of the planning horizon and the size of the industry. Large sized industries can supply about 10 customer orders in each production period, while small sized industries can meet about 6 customer orders in each period;
- Number of items: $J \in \{15, 30, 45\}$. Companies producing meat from various animals (beef, pork and poultry) have product catalogues with about 45 items, while industries specialised in meat from one (or two) animals have a product catalogue with around 15 (or 30) items;
- Setup times: $st_{ij} \in \mathrm{U}[2, 10]$, and Setup costs: $sc_{ij} = 500 st_{ij}$. The setup costs (sc_{ij}) are proportional to the setup times (st_{ij}) as stated in the literature (Sereshti & Bijari (2013));
- Processing times: $a_j = 1$. In the meat industry, the processing time does not significantly vary;
- Inventory holding costs: $h_j \in \mathrm{U}[2, 9]$. Meat products require monitored temperature storage, therefore there exist inventory holding costs mainly associated with the consumption of electric energy. Some products for exportation need to be stored at lower temperatures (about -25º Celsius). Hence, the unitary inventory holding cost can reach R$9, while seasoned meat produced for national consumption are refrigerated (not frozen) and they need to be stored at about 5º Celsius, implying in lower unitary inventory holding costs (about R$2);
- Profit of the orders: $P_{nt} = \sum_j p^j q_{jn}$, where p^j is the sale price of the item j, with $p^j \in \mathrm{U}[50, 100]$. Note that, the total profit of an order is the sum of the individual sale prices of the products that make up the order. Each product consists of about 1 to 5 kilograms of a type of packaged meat and the (wholesale) prices in Brazil ranges from R$ 50 (a package with 5 kilograms of sausages or seasoned chicken meat) to R$ 100 (a package with 2 kilograms of a noble beef);
- Maximum time window: $tw^{\max} = \max\{3, \lfloor \frac{T}{3} \rfloor\}$, and Minimum time window: 0. Usually, the customers are supermarkets and restaurants. The restaurants need to ensure the availability of all items on their menu all the time, hence these customers require a restrictive delivery time window composed of few periods. On the other hand, the supermarkets accept delivery time windows composed of various periods;

- First and last periods of the time windows of the orders: $F_n \in \mathrm{U}[1, T - tw_n]$ and $L_n = F_n + tw_n - 1$, where tw_n is the size (in periods) of the time window of order n, with $tw_n \in \mathrm{U}[0, tw^{\max}]$;
- Each order is composed by $O_n \in \mathrm{U}\left[1, \lfloor \frac{J}{2} \rfloor\right]$ different items that were randomly selected;
- Quantity of each item in the orders: $q_{jn} \in \mathrm{U}[5, 15]$, if item j makes up the order n and $q_{jn} = 0$, otherwise;
- Production capacity: $C_t = \frac{0.8}{tw^{\max}} * \sum_{n \in \{n | F_n \leq t \leq L_n\}} \sum_j a_j q_{jn}$. The available production capacity allows that at maximum 80% of the customer orders could be accepted.

The proposed test instances were grouped into 10 classes (each class with 10 test instances) according to the parameter choices as presented in Table 5, where the notation N30J15T5 is used to indicate the class of instances with N = 30, J = 15, and T = 5, for example. The proposed model and the heuristic procedure were implemented in C++ language using the Concert Technology tool of the commercial solver Cplex 12.6. The MIP subproblems obtained by the proposed heuristic procedure were solved by the Branch-and-Bound algorithm of the Cplex solver using default settings. We ran the tests on a two Intel Xeon processors desktop with 2.8 GHz and 128 GB DDR3 RAM memory. For each instance, we captured the best feasible solution (z^f) and the best dual (upper) bound (z^d) found. The deviation of the best feasible solution from the upper bound (GAP) was computed as $GAP = 100 * \left(\frac{z^d - z^f}{z^d}\right)$.

The proposed test instances and the C++ files containing the implementation of the proposed approaches are available at https://inma.ufms.br/docentes/willy-alves-de-oliveira/willy/PO2019.

5.2 Numerical tests

5.2.1 Relax-and-fix heuristic

In this section we present computational results regarding the RF heuristic (phase I of our heuristic framework). The aim of this section consists of determining values for the parameters *windowSize* and *overlap* in order to obtain an efficient RF heuristic.

Table 2 presents the results obtained by fixing $timeLimit = 900$ and varying the parameter *windowSize* in the set $\{1, 2, 3\}$ and the parameter *overlap* in the set $\{0, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}\}$. Different values for the parameters *windowSize* and *overlap* imply in different number of iterations. The maximum running time of 900 seconds was equally distributed between the iterations. For example, if $windowSize = 1$ and $overlap = 0$, then the RF heuristic is composed by T iterations. Therefore, we have fixed the maximum running time to solve the subproblems related to each RF iteration in $\frac{900}{T}$. Table 2 shows that the running time (RT) significantly increases when the parameters *windowSize* or *overlap* increase, so that, the faster procedure is obtained when $windowSize = 1$ and $overlap = 0$. On the other hand, better feasible solutions were obtained when $windowSize = 2$ and $overlap = \frac{1}{2}$ as indicated by the average GAPs (AG).

Table 2 – Results of phase I - RF heuristic. The parameter *windowSize* is denoted by WS. The obtained average GAPs are reported in the columns *AG* (in %), while the running times are presented in the columns *RT* (in seconds).

Phase I: RF heuristic - $timeLimit = 900$ seconds										
	$WS = 1$ $overlap = 0$		$WS = 2$ $overlap = \frac{1}{2}$		$WS = 3$ $overlap = \frac{2}{3}$		$WS = 2$ $overlap = 0$		$WS = 3$ $overlap = \frac{1}{3}$	
Class	AG	RT	AG	RT	AG	RT	AG	RT	AG	RT
N30J15T5	21.81	2	8.16	13	4.26	63	17.89	8	5.71	50
N30J30T5	24.46	20	14.03	152	12.29	659	21.36	118	12.88	375
N30J45T5	26.47	99	16.85	488	17.91	916	25.49	348	17.96	579
N50J15T5	13.51	3	6.39	20	4.12	137	12.22	12	4.46	108
N50J30T5	18.36	28	10.62	415	10.81	885	15.86	274	10.64	561
N50J45T5	18.17	215	12.62	766	14.19	920	17.24	385	14.49	607
N60J45T10	28.41	207	22.73	823	27.32	902	24.99	577	26.86	788
N100J30T10	22.56	52	16.08	680	17.71	905	17.98	402	16.65	749
N100J45T10	21.88	319	19.25	909	22.38	903	18.52	760	21.33	853
N150J30T15	25.80	169	23.21	808	25.73	905	24.36	640	24.85	790
Average	22.14	112	14.99	507	15.67	719	19.59	352	15.58	546

Since the feasible solution obtained in phase I will be improved in the next phases, it is desirable that the RF heuristic could able to quickly build it ensuring that the improvement procedures could have plenty running time to find a good solution. Hence, we have adopted $windowSize = 1$ and $overlap = 0$ in our heuristic framework.

However, we also highlight that the results presented on Table 2 suggest that adopting the parameters values $windowSize = 3$ and $overlap = \frac{2}{3}$ can be a good strategy if the maximum running time ($timeLimit$) could be increased (note that this procedure consumed the maximum running time in the classes N50J45T5, N60J45T10, N100J30T10, N100J45T10, and N150J30T15). Therefore, on Table 3 we present the results obtained using $windowSize = 3$ and $overlap = \frac{2}{3}$ and $timeLimit = 3600$ (an hour). As in the previous computational experiments, the maximum running time was equally distributed between the RF iterations. Table 3 also reports the objective function value obtained by the same heuristic using $timeLimit = 900$ seconds in order to identify the impact generated by increased running time.

5.2.2 Efficiency analysis of the proposed heuristic

In this section, we present computational experiments in order to study the efficiency of the proposed heuristic. Firstly, we present results regarding the performance of the heuristic. Secondly, we compare our heuristic with the Branch-and-Bound algorithm of the Cplex solver.

• Performance of the proposed heuristic:

Table 3 – Results obtained using $windowSize = 3$, $overlap = \frac{2}{3}$, and $timeLimit = 3600$ in the RF heuristic. The obtained average GAPs are reported in the columns *AG* (in %), while the best and worst GAPs are reported in the columns (BG) and (WG) in %, respectively. The running times are presented in the columns *RT* (in seconds).

timeLimit	3600 sec					900 sec
Class	FO	WG	BG	AG	RT	FO
N30J15T5	48573	11.52	0.01	3.77	64	48573
N30J30T5	88041	13.48	8.18	10.77	1582	87591
N30J45T5	101358	20.87	11.38	15.28	3129	99176
N50J15T5	96776	5.04	0.70	3.09	139	96776
N50J30T5	163029	12.02	7.59	9.16	3121	161056
N50J45T5	224837	14.23	9.91	12.09	3539	220222
N60J45T10	225050	27.11	17.49	22.02	3600	210559
N100J30T10	307122	17.30	13.84	15.33	3572	299127
N100J45T10	423607	23.13	15.62	18.35	3600	403293
N150J30T15	371402	26.02	19.45	22.64	3600	357640
Average	204979	17.07	10.42	13.25	2595	198401

Table 4 presents the results obtained on each phase of the proposed heuristic considering the maximum running time of one hour. In this computational experiment, we have fixed the maximum running of the phases I and II in 900 and 1200 seconds, respectively. The residual running time of each instance (until 3600 seconds) were allocated to the phase III. The maximum running time to solve the subproblems regarding phases II and III was fixed in 300 seconds. We can observe, for example, that in class N30J15T5, the first phase of the proposed heuristic spent about 2 seconds to construct initial solutions with average GAP of 21.81%, while in the second phase, the FOTO heuristic spent about 7 seconds and it found solutions with average GAP of 7.69%. Finally, in the third phase, the INS heuristic spent the residual time (3524 seconds) to find solutions with average GAP of 1.44%. Since the heuristic approach is not able to provide dual bounds, the reported GAPs were calculated using the dual bounds found by the Branch-and-Bound algorithm of the Cplex solver.

We note that in the instances considering 15 products and 5 periods the heuristic spent (in average) only 2.5 seconds to build an initial feasible solution (phase I) and only 90.5 seconds with the deterministic improvement (phase II), leaving a great residual time for the phase 3. Therefore, better solutions were found for these classes presenting an average GAP of about 2.13%. Moreover, we highlight that both improvement phases were able to improve the quality of the solutions for all classes of instances. In general, phase I consumed about 112 seconds to provide solutions with average GAPs of about 22.14%, while phase II spent about 406 seconds to provide solutions with average GAPs of about 14.67% (representing a reduction of about 7.47% with respect the initial solutions), and finally, phase III consumed the residual running time of

Table 4 – Objective function value (FO), average GAP (AG - in %), and running time (RT - in seconds) obtained on each phase of the proposed heuristic with (total) maximum running time fixed in 3600 seconds.

	Phase I - RF			Phase II - FOTO			Phase III - INS		
Class	FO	AG	RT	FO	AG	RT	FO	AG	RT
N30J15T5	39847	21.81	2	46676	7.69	7	49547	1.44	3524
N30J30T5	75371	24.46	20	85754	14.19	32	88738	9.83	3527
N30J45T5	89207	26.47	99	99778	17.61	97	105892	12.81	3305
N50J15T5	87544	13.51	3	93868	7.02	174	96903	2.82	3420
N50J30T5	147399	18.36	28	160087	11.31	346	162944	8.99	3197
N50J45T5	210843	18.17	215	225756	12.17	526	229170	10.53	2587
N60J45T10	207436	28.41	207	226982	21.62	533	233444	19.13	2652
N100J30T10	282599	22.56	52	303618	16.53	222	309002	15.03	3326
N100J45T10	406477	21.88	319	434819	16.29	1115	440365	15.18	2166
N150J30T15	357549	25.80	169	374371	22.26	433	377925	21.49	2998
Average	190427	22.14	112	204171	14.67	406	209393	11.72	3182

about 3182 seconds to provide final solutions with average GAPs of about 11.72% (representing a reduction of about 2.95% with respect the solutions found on phase II).

The results reported on Table 4 are also represented on Figure 2. In this Figure, the x-axis represents the consumed running time (and it is defined in logarithmic scale), while the y-axis represents the obtained average GAPs.

As observed in Section 4, the number of iterations performed in the FOTO procedure depends of the number of periods considered in the test instance. More specifically, when T = 5, T = 10, and T = 15, we have a total number of iterations of 10, 45, and 105, respectively. As reported on Table 4, the FOTO procedure consumed in average 197 seconds (when T = 5), 623 seconds (when T = 10), and 433 seconds (when T = 15). We note that the average running time when T = 10 was higher than when T = 15, because the number of products (J) considered in the test instance also significantly impacted on the running time of the FOTO heuristic. For example, the running time observed for the class N100J45T10 was 1115 seconds, while for the class N150J30T15 was only 433 seconds.

We highlight that we have fixed the maximum running time of the FOTO procedure in 1200 seconds and this maximum running time was reached only for three test instances from class N100J45T10. In these three test instances, phase II was interrupted before test all possible two periods combinations. In the others 97 test instances, the FOTO procedure tested all possible two periods combinations without to reach the running time of 1200 seconds.

• Comparing the heuristic approach with the Branch-and-Bound algorithm of the Cplex solver:

On Table 5 we compare the performance of the proposed heuristic approach with the Branch-and-Bound algorithm of the Cplex solver fixing the maximum running time within 20 minutes

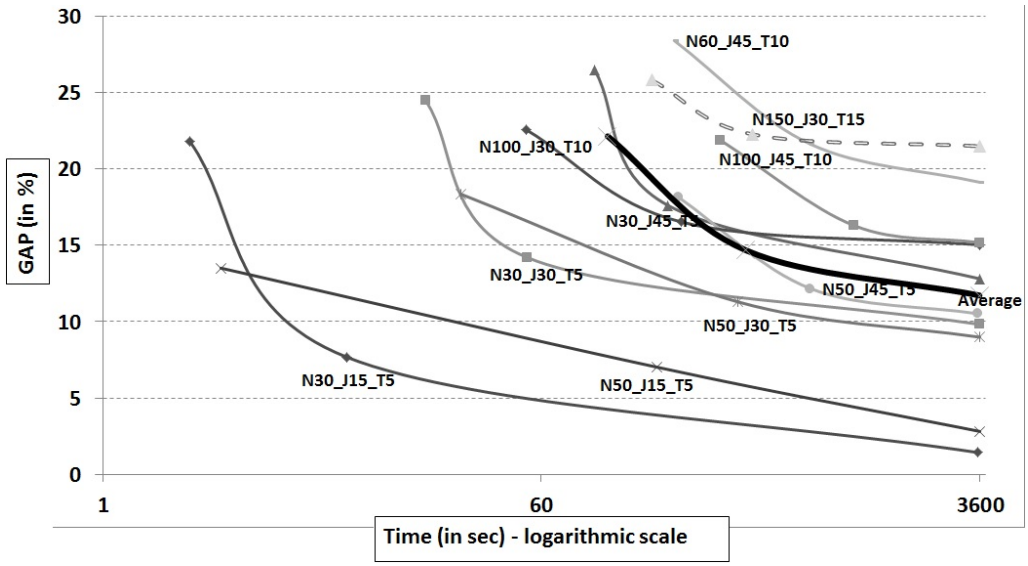

Figure 2 – Performance of the heuristic in each phase. The first node in each curve indicates the consumed time (in seconds) and the GAP (in %) obtained in the first phase (RF) of the heuristic. The second node presents the consumed time and the GAP obtained in the second phase (FOTO) and the third node indicates the consumed time and the GAP obtained for the INS procedure (third phase). The x-axis is presented in logarithmic scale.

(1200 seconds) and within 1 hour (3600 seconds). The column WG presents the worst GAP (in %), while in the column BG we present the best GAP observed for each class. Table 5 also presents the average GAPs (AG) and the average running times (RT).

We observe that the proposed heuristic performs better than the Cplex solver within the considered maximum running times. Such heuristic provided a better WG for all classes with exception for classes N30J15T5 and N30J45T5 when the maximum running time was fixed in 1200 seconds, and a better WG for all classes when the maximum running time was fixed in 3600 seconds. Considering the BG, the heuristic provided best results for all classes with exception for class N50J15T5 when the maximum running time was fixed in 1200 seconds, and for all classes when the maximum running time was fixed in 3600 seconds. In the average performance (AG), the heuristic did not overcome the Cplex solver only in classes N30J15T5 and N50J45T5 in both considered maximum running times, but the results obtained were competitive (mainly with the maximum running time of 3600 seconds).

The developed heuristic provided greater gains for the test instances with 45 products (specially considering 100 customer orders). For some of these classes, the GAP of the solutions provided by the Cplex solver exceeded 100%, i.e., the model provided a solution with negative profit. This fact occurred because Constraints 7 require that the production line remains set up for production all the time. Therefore, the solution that consists of not producing anything (with costs equal to zero) is not feasible for the problem.

Table 5 – Results of the proposed heuristic and of the Branch-and-Cut algorithm (Cplex solver).

Maximum time of 1200 seconds								
	Cplex				Heuristic			
Class	WG	BG	AG	RT	WG	BG	AG	RT
N30J15T5	4.61	0.01	1.26	1073	9.92	0.01	2.48	771
N30J30T5	20.08	11.75	15.31	1200	13.62	8.46	10.54	981
N30J45T5	20.08	16.93	25.48	1200	21.03	9.67	14.43	887
N50J15T5	6.42	0.36	3.25	1200	5.69	1.29	3.66	918
N50J30T5	19.12	10.87	13.58	1200	11.61	7.85	9.46	927
N50J45T5	24.11	15.76	19.54	1200	13.55	8.98	11.39	761
N60J45T10	102.05	73.86	97.31	1200	23.62	15.36	20.04	1200
N100J30T10	34.90	23.64	27.33	1200	18.01	13.79	15.63	1169
N100J45T10	101.55	98.97	100.47	1200	21.55	13.97	16.70	1200
N150J30T15	100.25	32.72	73.96	1200	26.61	20.30	22.25	1176
Average	43.32	28.49	37.75	1187	16.52	9.97	12.66	999
Maximum time of 3600 seconds								
Class	WG	BG	AG	RT	WG	BG	AG	RT
N30J15T5	4.61	0.01	1.06	2095	3.76	0.01	1.44	3536
N30J30T5	17.29	9.95	13.01	3600	13.07	8.26	9.83	3600
N30J45T5	29.32	15.20	23.53	3600	18.79	9.01	12.81	3600
N50J15T5	5.06	0.11	2.67	3600	5.10	0.01	2.82	3600
N50J30T5	13.86	9.42	11.17	3600	10.82	7.81	8.99	3600
N50J45T5	23.04	15.02	17.80	3600	12.64	8.76	10.53	3543
N60J45T10	55.38	39.47	47.17	3600	23.68	14.75	19.13	3600
N100J30T10	25.34	17.73	20.97	3600	16.70	13.39	15.03	3600
N100J45T10	101.35	36.58	86.00	3600	20.43	12.08	15.18	3600
N150J30T15	51.38	36.24	42.04	3600	25.31	18.41	21.49	3600
Average	32.66	17.97	26.54	3450	15.07	9.32	11.72	3587

The running times observed were close for both approaches, because the maximum running time was reached for most instances. This fact evidences that real-based test instances for the addressed problem are very challenging from a computational perspective.

The RF heuristic was able to quickly build feasible solutions (RT = 112 sec) with reasonable quality (AG = 22.14%). Therefore, in order to compare our heuristic framework with the Branch-and-Bound algorithm, we have performed a computational experiment that consists of starting the Branch-and-Bound with the feasible solution found by the RF procedure. This experiment was performed using the MIP Start methodology of the Cplex solver and the results are reported on Table 6.

From the results presented on Table 6, we can see that the utilization of the RF solutions was able to significantly improve the solutions found by the Branch-and-Bound algorithm for all classes of

instances, mainly for classes N60J45T10, N100J45T10, and N150J30T15. However, in general, it was not able to overcome our three phases heuristic framework as presented on Table 5.

Table 6 – Objective function value (FO), dual bound (DB), worst GAP (WG), best GAP (BG), average GAP (AG), and running time (RT) obtained starting the Branch-and-Bound algorithm with the feasible solution found by the RF procedure.

Starting the Branch-and-Bound with the RF solutions						
Class	FO	DB	WG	BG	AG	RT
N30J15T5	49846	50129	3.86	0.01	0.57	1734
N30J30T5	87005	97994	14.86	8.47	11.32	3600
N30J45T5	98895	118809	22.21	12.69	17.31	3600
N50J15T5	97415	100674	5.21	0.01	2.41	3600
N50J30T5	160749	179425	12.87	8.76	10.50	3600
N50J45T5	222958	254532	13.96	9.73	12.42	3600
N60J45T10	211466	288888	30.56	21.61	27.02	3600
N100J30T10	300007	362904	19.27	15.64	17.42	3600
N100J45T10	410060	517329	27.81	17.03	21.15	3600
N150J30T15	364738	481677	28.79	22.09	24.40	3600
Average	200314	245236	17.94	11.60	14.45	3413

5.2.3 Quality of the obtained bounds

In this Section, we present computational results to study the evolution of the bounds' quality over the time. In the first experiment, we used the Branch-and-Bound algorithm of the Cplex software to solve the instances from a small sized class (N30J15T5) and from a large sized class (N150J30T15) successively fixing the running time in 600, 1200, 3600, and 7200 seconds. For each considered running time, we observed the value of the primal and the dual bounds. The results are presented in Figure 3, where the x-axis presents the running times, while the y-axis presents the value of the bounds.

Considering the class N30J15T5, we can see that the value of the primal bound presented just a slight variation over the time (with a total variation of about 0.7% between 600 seconds and 7200 seconds), while the dual bound presented a variation of around 4.3%. The initial GAP was about 5.6% and the final GAP was approximately only 0.8%. Therefore, we can see that for small sized instances, the Cplex tends to find a good solution quickly (before 600 seconds), but it spent a long time to find a strong dual bound.

On the other hand, considering class N150J30T15 we observed a significant variation of the primal bound value (about 72% between 600 seconds and 7200 seconds), while the value of the dual bound presented a slight variation of about 2.3%. The initial GAP was approximately 81%, while the final GAP was about 30%. In this case, considering the large sized test instances, we

can see that the Cplex can significantly improve the quality of the primal bound over the time, while it presents difficulties to find a good dual bound at acceptable computational time.

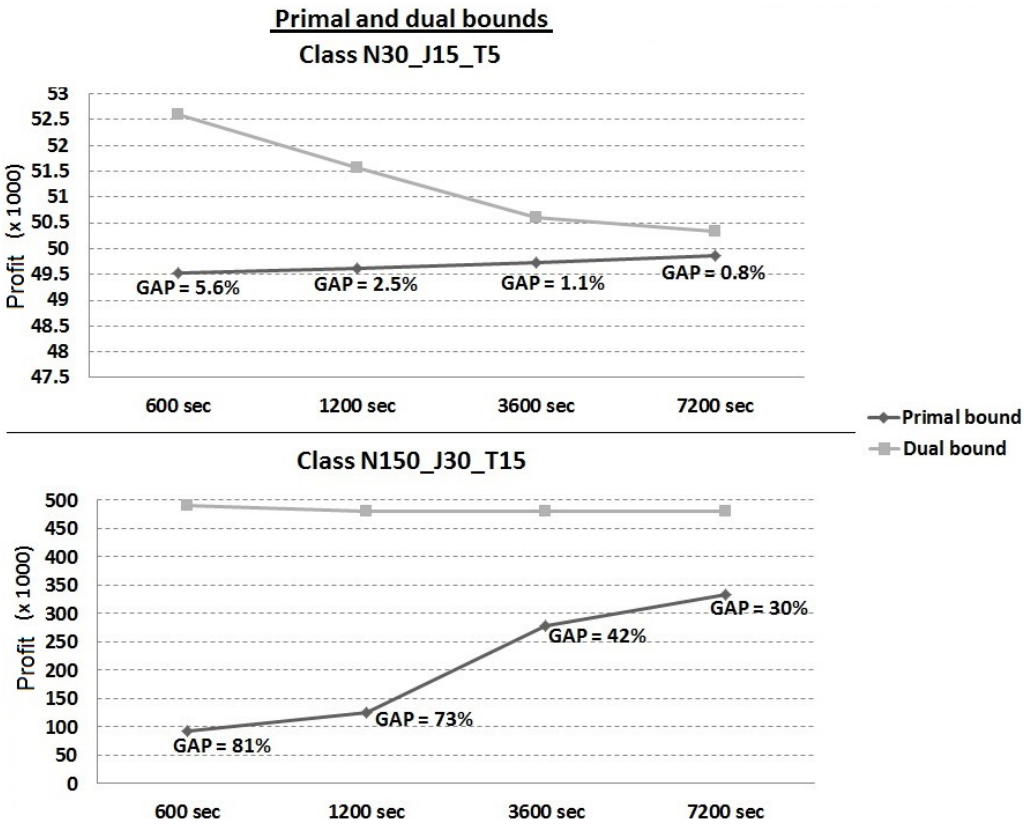

Figure 3 – Evolution of the bounds in the Branch-and-Cut algorithm of the Cplex solver.

In order to conclude this Section, in the Figure 4, we present the evolution of the primal and dual bound values obtained by the Cplex solver over a running time of 10 hours for an instance randomly selected from class N150J30T15. In the same graphic, we include the primal bounds obtained by the proposed heuristic approach over the same time horizon.

We can see that the quality of the solutions obtained by the heuristic overcame the solutions obtained by the Cplex solver all the time. Additionally, Cplex could not significantly improve the quality of dual bound over the considered running time. The GAP from the last solution obtained by the Cplex solver to the best dual bound is about 25.70%, while the GAP from the best solution found by our heuristic is about 17.87%.

6 CONCLUSIONS AND FUTURE PROPOSALS

In this research, we addressed a lot sizing and scheduling problem inspired by a real-world production environment with non-traditional characteristics: demand choice flexibility, delivery

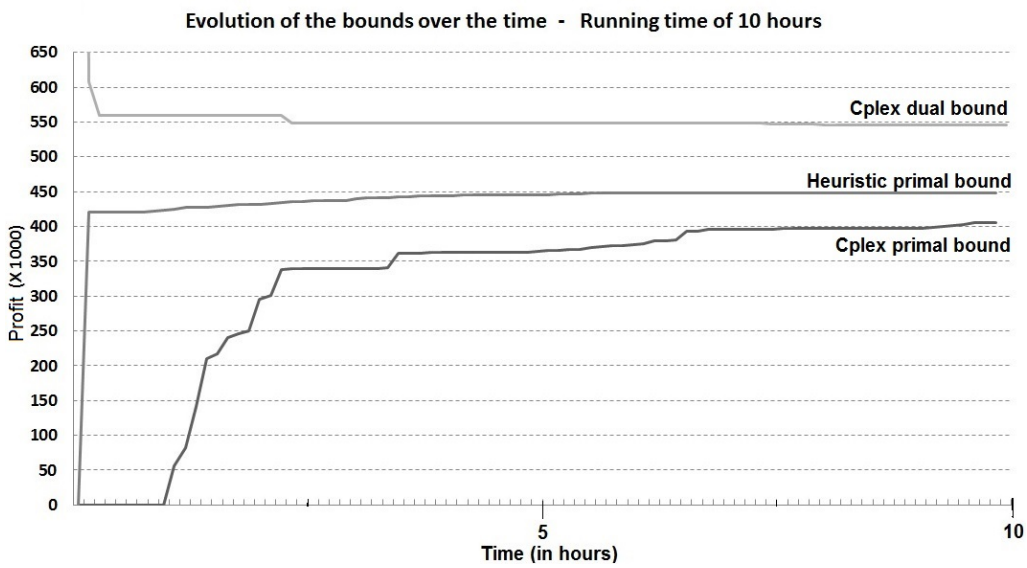

Figure 4 – Evolution of the primal and dual bounds over a running time of 10 hours.

time windows and indivisible orders. Firstly, we extended a MIP model from the literature to represent the problem, and secondly, we developed a MIP based heuristic procedure in order to find good feasible solution at an acceptable computational time. The proposed heuristic is composed by three phases: construction (that consists of a relax-and-fix procedure), deterministic improvement (a fix-and-optimize heuristic), and stochastic improvement (a INS procedure). Computational experiments were conducted with a set of 100 test instances that were generated combining information obtained in some technical visits with parameter settings described in the literature for related problems. The results showed that the proposed heuristic is able to provide good feasible solutions for the problem overcoming the Branch-and-Bound algorithm of the Cplex solver, mainly for large sized test instances. There are a number of directions for potential future research. First of all, we highlight that the computational experiments performed in this paper suggest that the dual bounds obtained by the Branch-and-Bound algorithm are far from the optimal solutions. Therefore, in the future researches we can investigate some exact methods (as Lagrangian relaxation and/or Dantzig-Wolfe decomposition) with the aim of obtaining good dual bounds. Besides that, it can be interesting to develop methods which combine these exact approaches with the heuristic proposed in this paper. On the other hand, as largely suggest on the literature, the simultaneous consideration of different production and logistic aspects in a same mathematical model can provide greater gains for the supply chain management. In this perspective, we are planning to study the integration of the problem described in this paper with the distribution/routing problem. We note that, the distribution and routing decisions can significantly impact on the set of orders that are accepted for production.

ACKNOWLEDGMENTS

The research of the first author was supported by the Universidade Federal de Mato Grosso do Sul and in part by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq). Research was carried out using the computational resources of the Center for Mathematical Sciences Applied to Industry (CeMEAI), funded by FAPESP (Grant 2013/07375-0). The authors are also grateful to three anonymous referees for their valuable comments, which helped improve the presentation of the paper.

References

[1] AKARTUNALI K, FRAGKOS I, MILLER AJ & WU T. 2016. Local cuts and two-period convex hull closures for big-bucket lot-sizing problems. *INFORMS Journal on Computing*, **28**(4): 766–780.

[2] ALMADA-LOBO B, CLARK A, GUIMARÃES L, FIGUEIRA G & AMORIM P. 2015. Industrial insights into lot sizing and scheduling modeling. *Pesquisa Operacional*, **35**(3): 439–464.

[3] AOUAM T, GERYL K, KUMAR K & BRAHIMI N. 2018. Production planning with order acceptance and demand uncertainty. *Computers & Operations Research*, **91**: 145–159.

[4] BRAHIMI N, ABSI N, DAUZÈRE-PÉRÈS S & NORDLI A. 2017. Single-item dynamic lot-sizing problems: An updated survey. *European Journal of Operational Research*, **263**(3): 838–863.

[5] COPIL K, WÖRBELAUER M, MEYR H & TEMPELMEIER H. 2017. Simultaneous lotsizing and scheduling problems: a classification and review of models. *OR spectrum*, **39**(1): 1–64.

[6] DA SILVA FAM, MORETTI AC & DE AZEVEDO AT. 2014. A scheduling problem in the baking industry. *Journal of Applied Mathematics*, **2014**.

[7] FERREIRA D, MORABITO R & RANGEL S. 2009. Solution approaches for the soft drink integrated production lot sizing and scheduling problem. *European Journal of Operational Research*, **196**(2): 697–706.

[8] FLEISCHMANN B & MEYR H. 1997. The general lotsizing and scheduling problem. *Operations-Research-Spektrum*, **19**(1): 11–21.

[9] FURTADO MGS. 2012. *O planejamento da produção de pedidos em fundições de pequeno porte*. Ph.D. thesis. Universidade de São Paulo.

[10] GUIMARAES L, KLABJAN D & ALMADA-LOBO B. 2013. Pricing, relaxing and fixing under lot sizing and scheduling. *European Journal of Operational Research*, **230**(2): 399–411.

[11] GUIMARÃES L, KLABJAN D & ALMADA-LOBO B. 2014. Modeling lotsizing and scheduling problems with sequence dependent setups. *European Journal of Operational Research*, **239**(3): 644–662.

[12] HAASE K. 1996. Capacitated lot-sizing with sequence dependent setup costs. *Operations-Research-Spektrum*, **18**(1): 51–59.

[13] JAMES RJ & ALMADA-LOBO B. 2011. Single and parallel machine capacitated lotsizing and scheduling: New iterative MIP-based neighborhood search heuristics. *Computers & Operations Research*, **38**(12): 1816–1825.

[14] JARUPHONGSA W & LEE CY. 2008. Dynamic lot-sizing problem with demand time windows and container-based transportation cost. *Optimization Letters*, **2**(1): 39–51.

[15] OLIVEIRA WA & SANTOS MO. 2017. A New Branching Rule to Solve the Capacitated Lot Sizing and Scheduling Problem with Sequence Dependent Setups. *TEMA (São Carlos)*, **18**(3): 515–529.

[16] SEL Ç & BILGEN B. 2014. 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.

[17] SERESHTI N & BIJARI M. 2013. Profit maximization in simultaneous lot-sizing and scheduling problem. *Applied Mathematical Modelling*, **37**(23): 9516–9523.

[18] SOLER WA, SANTOS MO & AKARTUNALI K. 2019. MIP approaches for a lot sizing and scheduling problem on multiple production lines with scarce resources, temporary workstations, and perishable products. *Journal of the Operational Research Society*, pp. 1–16.

[19] SUPITHAK W, LIMAN SD & MONTES EJ. 2010. Lot-sizing and scheduling problem with earliness tardiness and setup penalties. *Computers & Industrial Engineering*, **58**(3): 363–372.

[20] TOLEDO CFM, DA SILVA ARANTES M, HOSSOMI MYB, FRANÇA PM & AKARTUNALI K. 2015. A relax-and-fix with fix-and-optimize heuristic applied to multi-level lot-sizing problems. *Journal of Heuristics*, **21**(5): 687–717.

[21] YANG DL, HOU YT & KUO WH. 2017. A note on a single-machine lot scheduling problem with indivisible orders. *Computers & Operations Research*, **79**: 34–38.