HOPS - Hamming-Oriented Partition Search for production planning in the spinning industry

Victor C.B. Camargo[a,b,*], Franklina M.B. Toledo[b], Bernardo Almada-Lobo[a]

[a] *INESC TEC, Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias s/n, Porto 4200-465, Portugal*

[b] *Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, Caixa Postal 668, São Carlos - SP 13560-970, Brazil*

Abstract

In this paper, we investigate a two-stage lot-sizing and scheduling problem in a spinning industry. A new hybrid method called HOPS (Hamming-Oriented Partition Search), which is a branch-and-bound based procedure that incorporates a fix-and-optimize improvement method is proposed to solve the problem. An innovative partition choice for the fix-and-optimize is developed. The computational tests with generated instances based on real data show that HOPS is a good alternative for solving mixed integer problems with recognized partitions such as the lot-sizing and scheduling problem.

Keywords: matheuristic, lot-sizing and scheduling problems, textile industry, mixed integer programming

1. Introduction

The textile industry is very important for the Brazilian economy as it generates employment and increases regional development. According to the annual report of the textile sector (ABIT, 2011), Brazil plays a major role in the textile and apparel international market, as it generated a revenue of US$ 41.8 billion in 2010. There are currently 1.6 million employees in textile industries in Brazil and spinning factories employ approximately 80,000 workers.

The production system of the spinning industry is characterized by a multi-stage process, with single or parallel machines on each stage. In general, these machines have dissimilar production characteristics, such as processing speeds and changeover times. The setups depend on the sequence of products manufactured by the machines. The main products are made of natural or synthetic fibers, but the most commonly used raw material is cotton. Production orders or

*Corresponding author

Email addresses: victor@icmc.usp.br (Victor C.B. Camargo), fran@icmc.usp.br (Franklina M.B. Toledo), almada.lobo@fe.up.pt (Bernardo Almada-Lobo)

lots must be achieved in order to satisfy customer demand in the most cost-effective manner. Therefore, the spinning industry requires simultaneous size and sequence production lots. The integration of scheduling and lot-sizing problems helps to define better production plans than those generated from the solution of these problems in a hierarchical way. The integrated lot-sizing and scheduling problem is considered the major issue in operational production planning (Drexl and Kimms, 1997) and is also present in other industrial processes, such as foundry (Araujo et al., 2007), glass containers (Almada-Lobo et al., 2008), soft drinks (Ferreira et al., 2009) and beverages (Guimarães et al., 2012).

The research reported in the literature on production planning in the textile industry is rather scarce. Silva and Magalhaes (2006) focus on the lot-sizing and scheduling problem of an acrylic fiber industry. The fiber manufacturing process appears upstream the spinning in the textile supply chain. Nevertheless, three studies address the production planning problem downstream the spinning. Serafini and Speranza (1992) focus on the weaving process, Karacapilidis and Pappis (1996) tackle the integrated process of the warp making, starching and weaving machines and Pimentel et al. (2011) address the knitting process. To the best of our knowledge, few studies have focused on the production planning in the spinning process. In addition to that, a weak point regarding the efficiency of this industry is the difficulty to quickly devise good production plans. By overcoming this weakness, companies' competitiveness may be increased.

Solution methods for this type of problem rely on MIP solvers or (meta)heuristics. Usually, MIP solvers are exact methods that aim to provably find optimal solutions. In some cases, the running time may be prohibitive. On the other hand, (meta)heuristics provide better feasible solutions faster, although the optimality proof is (often) not guaranteed. Recent research focused on problem-oriented methods (James and Almada-Lobo, 2011), such as relax-and-fix (Beraldi et al., 2008) and fix-and-optimize (Sahling et al., 2009) heuristics. Variables belonging to a problem partition are fixed to a certain value to create a reduced problem of an easier solution. These problem-oriented methods carry problem aspects into the solution procedure, as the variable partition is inherent to the problem.

State-of-the-art MIP solvers incorporate heuristics in the exact methods (e.g. relaxation induced neighborhood search (Danna et al., 2005), local branching (Fischetti and Lodi, 2003) and some variants) to quickly find better solutions. These hybrid methods follow the variable fixing ideas. Each branch-and-bound node provides information for the decision variables and a reduced problem is created by fixing some variables based on this information. In case the reduced

problem returns a solution, it is also a solution to the original problem and this new solution can be injected into the branch-and-bound tree as a new upper bound. These improvement strategies can accelerate the optimality proof. The cross fertilization between exact and (meta)heuristic methods is called matheuristics (Maniezzo et al., 2010).

In this paper, we propose a mathematical model that integrates the lot-sizing and scheduling decisions of a spinning industry taking into account the synchronization between the first and second stages. The model is based on the multi-stage general lot-sizing and scheduling problem. Equally important, the exact approach called HOPS (Hamming-Oriented Partition Search) is proposed as a new method based on mathematical programming to solve this problem. It involves the problem solution with the branch-and-bound method combined with a problem-oriented procedure that injects new and better upper bounds into the original problem. The Hamming distance (Hamming, 1950) between the solutions found when exploring the branch-and-bound tree is used to define the problem partitions of stabilized values. The stabilized partitions are fixed to create the reduced problems and their solutions can be injected into the branch-and-bound tree of the original problem. Initially, the HOPS is tested on a well-known lot-sizing problem and its performance is compared against that of a recent tailored heuristic reported in the literature. For the spinning problem, an instance generator is designed to conduct computational experiments in an extensive dataset. The HOPS validation in the production planning in the spinning industry is also supported by real-world data based on instance settings.

The paper is organized as follows: Section 2 describes the production problem of the spinning industry, which is then formulated by adapting a known mathematical model; Section 3 gives an overview of the fixing variable matheuristics; Section 4 introduces the HOPS algorithm; Section 5 sets benchmarks for the results of HOPS compared to those obtained by other state-of-the-art approaches based on mathematical program available in commercial software. Section 6 addresses some concluding remarks.

2. The production system

The manufacturing system in a spinning industry can be defined as a procedure in which fibers are processed and then used to make different types of yarns. This procedure is described in the sequence of processes represented in Figure 1. In the beginning, lints from various bales are mixed and blended in order to achieve a uniform blend of fiber properties. The fiber blend is blown by air from a feeder through ducts to intermediate machines for cleaning and carding

and these machines separate and align the fibers into a thin web. The fiber web is then pulled through a conical device, providing the sliver that is pulled again and then twisted to make the sliver tighter and thinner until it complies with yarn specifications. After spinning, the yarns are tightly wound around bobbins or tubes. The yarn packages are then ready for distribution.

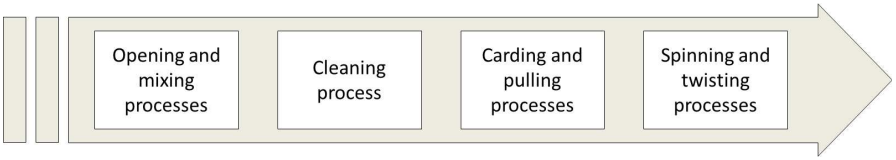

Figure 1: Sequence of processes in the production system of a spinning industry.

The spinning process is categorized according to the types of spinning, such as open-end, ring, air jet and Vortex spinning. In the open-end spinning, the yarn is directly produced from the sliver, eliminating the additional pulling process presented in the ring spinning. The air jet and Vortex spinning systems have also eliminated the need for the additional pulling process.

This study deals with open-end spinning and all its intermediate processes can work in a synchronized mode with the opening and mixing processes. The capacities of the first processes are sufficient to maintain the spinning process working. Therefore, the last process - spinning - is considered the production bottleneck.

In order to detail the production features, consider N yarns to be scheduled on M parallel machines over a finite planning horizon with a given length T. The demand for yarns is known prior to each period of the planning horizon that should be met in case capacity is sufficient. Although the production line operates 24 hours on a 7-day week basis, there may occur delays when the demand for yarns is high, thus backlogs must be represented in the model.

A yarn can not be manufactured in a given time period unless a fiber blend that ensures its quality is also processed in this period. Therefore, the planning of the first-stage product is also required. A fiber blend is often used in several yarns, but a yarn is only made using a fiber blend that respects its quality limits. As only one fiber blend can be processed in this type of production line, all the machines must produce yarns that require the same quality of fibers at each point in time. The production dependency between the two stages is represented in Figure 2.

The machines may differ in the processing rates of the same yarn, thus the fiber blend can be consumed at different speeds. A setup changeover from one yarn to another consumes capacity time, which is dependent on the sequence in which the yarns are processed. The setup can be

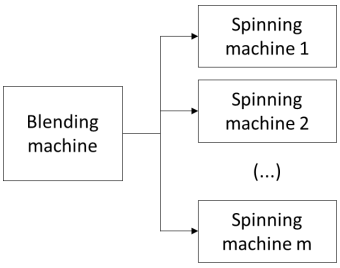

Figure 2: Diagram of a two-stage production environment in a spinning industry.

carried from one period to the next. The setup changeover for the fiber blend can be considered null as another fiber blend is immediately available for a later use in the production. This fact reinforces the statement that the fiber blend sub-process is not the production bottleneck.

The production plan must determine the size and sequence of the lots on all machines with minimal backlog, inventory and changeover costs. An example of a production plan for the spinning case is illustrated in Figure 3. Five yarns of two different fiber qualities are scheduled on three parallel machines over three planning periods. The dark gray rectangles represent the time wasted on setting up the machine for the yarn production. The first two blends are dedicated to the same product family, that is, the set of yarns that requires the same fiber blend quality. The third blend is dedicated to a different product family.

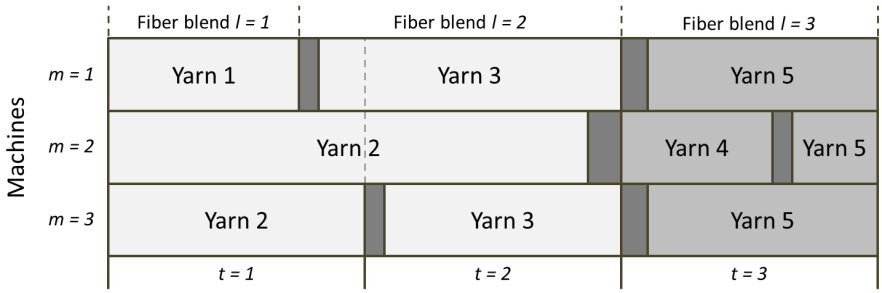

Figure 3: An illustrative production plan.

2.1. Mathematical model

This section describes an MIP formulation adapted from the Multi-Stage General Lot-sizing and Scheduling Problem (MSGLSP) - introduced in Camargo et al. (2012) - for the problem addressed. Backlogging must be taken into account as it is a common feature in the spinning industry. The MSGLSP divides the planning horizon into T periods. Each period is sub-divided into a number of micro-periods in which only one yarn can be produced as a small-bucket model (Meyr, 2002). Therefore, the sequence of production is obtained in a rather straightforward

manner allowing for the incorporation of sequence-dependent setups. The fundamental assumption is that a user-defined parameter restricts the number of micro-periods per period and, consequently, the upper bound on the number of setups. However, the micro-periods have no predefined length. A micro-period is confined to be within a single period and its length can not exceed the length of its period. Without loss of generality, we have assumed the length of each period to be equal to one. This assumption ensures some parameters to be standardized accordingly. The model makes use of the same micro-period length across all machines (that is, a fixed time grid). This feature enables the easy synchronization between the fiber blend (first stage) and the yarns produced by the spinning machine (second stage).

An optimal solution involves sizing and scheduling the production lots, respecting the constraints with the minimum setup, backlogging and inventory costs. The solution to the general lot-sizing and scheduling problem - GLSP (Fleischmann and Meyr, 1997) and its extensions (e.g. MSGLSP) from an MIP solver is limited to small-sized instances - see (Santos and Almada-Lobo, 2012). However, improvements such as reformulations, provide tighter bounds and can speed up the solution to the problem or even partially overcome these limitations. The SPL - simple plant location reformulation (Krarup and Bilde, 1977) is one of the improvements for the lot-sizing problem - see (Gao et al., 2008) for a comparative study. Hereafter, the SPL is used to reformulate the MSGLSP. In addition, a stronger set of equations (8)-(10) that ensures a setup changeover in consecutive periods and micro-periods provides a much better alternative for the MSGLSP proposed in (Camargo et al., 2012). Such constraints were first proposed for the DLSP - discrete lot-sizing and scheduling problem (Belvaux and Wolsey, 2001).

In order to clearly present the model formulation, the notation is given as follows:

Indices	
$i,\ j = 1,\ \ldots,\ N$	yarns;
$k = 1,\ \ldots,\ K$	blend that ensures the quality of the family $S(k)$ of yarns;
$m = 1,\ \ldots,\ M$	spinning machines;
$r = 1,\ \ldots,\ R$	micro-periods per period;
$t,\ t' = 1,\ \ldots,\ T$	time periods.

Parameters	
$c_{itt'}$	production cost for yarn i produced in period t to meet demand in period t';
σ_{mij}	setup cost of a changeover in machine m from yarn i to j;
d_{it}	demand of yarn i in period t;
C	maximum capacity (kg) per loading of the bale laydown machine - fiber blend process;
p_{mi}	processing time in machine m of one kilogram of yarn i;
s_{mij}	setup time of a changeover in machine m from yarn i to j;
$S(k)$	set of yarns that belong to family k.

Variables	
$X_{mitrt'} \geq 0$	production quantity in machine m of yarn i in micro-period r of period t to meet demand in period t';
Z_{mijtr}	takes on 1, if there is a changeover in machine m from yarn i to yarn j in period t and micro-period r; 0 otherwise;
$\mu^s_{tr} \geq 0$	start time of micro-period r in period t;
$\mu^f_{tr} \geq 0$	end time of micro-period r in period t;
$Y_{mitr} \in \{0,1\}$	takes on 1, if machine m is set up for yarn i in period t and micro-period r; 0 otherwise;
$U_{trk} \in \{0,1\}$	takes on 1, if fiber blend processed in period t and micro-period r meets the quality to produce yarns of family k; 0 otherwise.

The stronger Multi-Stage General Lot-sizing and Scheduling Problem (sMSGLSP) reads:

$$\textbf{Minimize} \quad \sum_{m=1}^{M}\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{r=1}^{R}\sum_{t'=1}^{T} c_{itt'} \cdot X_{mitrt'} + \sum_{m=1}^{M}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{t=1}^{T}\sum_{r=1}^{R} \sigma_{mij} \cdot Z_{mijtr} \tag{1}$$

subject to

$$\sum_{m=1}^{M}\sum_{t=1}^{T}\sum_{r=1}^{R} X_{mitrt'} = d_{it'} \qquad \forall i, t' \tag{2}$$

$$\mu^s_{t1} = t - 1 \qquad \forall t \tag{3}$$

$$\mu^f_{tR} = t \qquad \forall t \tag{4}$$

$$\mu^s_{tr} \geq \mu^f_{t(r-1)} \qquad \forall t, r > 1 \tag{5}$$

$$\mu^f_{tr} - \mu^s_{tr} \geq \sum_{i=1}^{N}\sum_{j=1}^{N}(s_{mji} \cdot Z_{mjitr}) + \sum_{i=1}^{N}\sum_{t'=1}^{T} p_{mi} \cdot X_{mitrt'} \qquad \forall m, t, r \tag{6}$$

$$X_{mitrt'} \leq d_{it'} \cdot Y_{mitr} \qquad \forall m, i, t, r, t' \tag{7}$$

$$Y_{mitr} = \sum_{j=1}^{N} Z_{mjitr} \qquad \forall m, i, t, r \tag{8}$$

$$Y_{mit(r-1)} = \sum_{j=1}^{N} Z_{mijtr} \qquad \forall m, i, t, r > 1 \tag{9}$$

$$Y_{mi(t-1)R} = \sum_{j=1}^{N} Z_{mijt1} \qquad \forall m, i, t > 1 \tag{10}$$

$$\sum_{i=1}^{N} Y_{mitr} = 1 \qquad \forall m, t, r \tag{11}$$

$$X_{mitrt'} \leq d_{it'} \cdot U_{trk} \qquad \forall m, i \in S(k), t, r, t', k \tag{12}$$

$$\sum_{m=1}^{M}\sum_{i=1}^{N}\sum_{t'=1}^{T} X_{mitrt'} \leq C \cdot \sum_{k=1}^{K} U_{trk} \qquad \forall t, r \tag{13}$$

$$\sum_{k=1}^{K} U_{trk} \leq 1 \qquad \forall t, r \qquad (14)$$

$$Y_{mitr} \in \{0,1\}\,;\; U_{trk} \in \{0,1\} \qquad \forall m, i, j, t, r, k \qquad (15)$$

$$\text{all other variables are non-negative and continuous.} \qquad (16)$$

This type of model is known as the *disaggregate formulation* (Brahimi et al., 2006). The objective function (1) is to minimize the sum of backlogging, inventory and sequence dependent setup costs. Here, the amount produced before the delivery date is considered inventory, given by $X_{mitrt'}$ in case $t < t'$. On the other hand, $X_{mitrt'}$ when $t > t'$ is the amount produced after the delivery date, that is, backlogging orders. Similarly, the production costs $c_{itt'}$ refer to inventory costs in case $t < t'$ and to backlogging costs in case $t > t'$. For $t = t'$, $c_{itt'}$ equals zero. Constraints (2) attempt to satisfy the demand, by taking into account inventories and backlogging. Constraints (3)-(5) define the start and end times of each micro-period r, hence its length. Constraints (3) block the starting time of the first micro-period to the beginning of the respective macro-period. Likewise, constraints (4) block the ending time of the last micro-period to the end of the respective macro-period. It must be highlighted that $\mu_{tR}^{f} - \mu_{t1}^{s} = 1$. This condition will be clear when the capacity limits are discussed later on. Constraints (5) avoid the micro-periods overlapping. A spinning machine works continuously during the period, therefore p_{mi} can be comprehended as the fraction of the period consumed by machine m to produce one kilogram of yarn i. The capacity limitation of the parallel spinning machines is implicitly expressed and Constraints (6) ensure that $p_{mi} \cdot X_{mitr}$ respects this capacity limitation, that is, the machine capacity, the processing rates and the setup times are normalized to 1. Moreover, they define the lower bound on each micro-period length based on the production and setup times of all machines. The same requirements determine the upper bound on the amount produced in each micro-period and set a fixed time grid of micro-periods for the machines. Observe that the model allows for idle time intervals, which may take place within a micro-period or between micro-periods.

Constraints (7) ensure that the production of a yarn in a micro-period occurs only if the machine is set up for it. Equations (8)-(10) link the setup variables to the changeover variables. By means of Constraints (8), machine m can produce yarn i in micro-period r of period t if and only if a changeover from another yarn j to yarn i takes place at the beginning of micro-period ($i = j$ is possible). Similarly, (9) and (10) make sure that if a changeover from yarn

i to yarn j takes place in a given micro-period, then the precedent micro-period is set up for yarn i. Requirements (11) establish that each machine is set up for one and only one yarn in each micro-period. By Constraints (12), in each micro-period, all the machines produce yarns of the same family. Constraints (13) restrict the production in a micro-period to the maximum size of the bale laydown. As only one machine is considered in the first stage, Constraints (14) limit the number of families (or, in other words, the number of fiber blends) to be produced per micro-period to one. Constraints (15) and (16) express the variable domains.

Figure 4 illustrates a production plan based on the sMSGLSP variables. Three groups of active variables are shown. Recall that variables μ^s_{tr} and μ^f_{tr} indicate the start and end of each micro-period. The fiber blend produced in each micro-period that supplies all the spinning machines is expressed by U_{trk}. Finally, variables Y_{mitr} represent the yarn prepared for the production. The variables defining production quantities, inventory and backlogging levels, as well as the setup changeover are associated with the variables depicted in Figure 4.

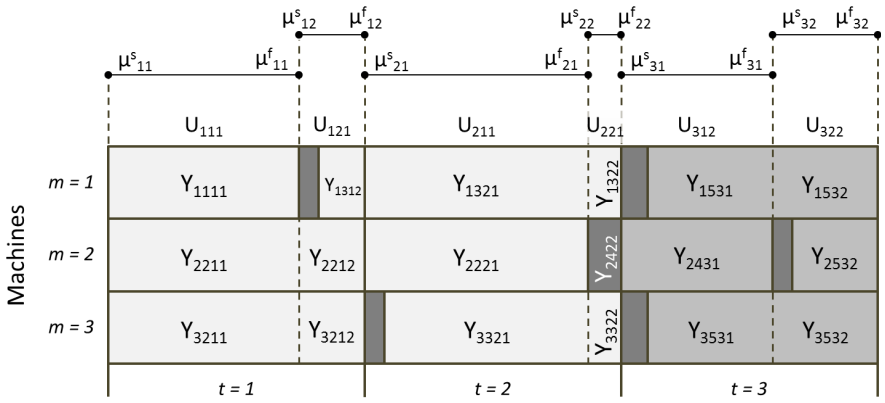

Figure 4: An illustrative production plan based on the decision variables.

For this example, the plan for machine 1 consists of six micro-periods and the yarn production sequence of 1-3-3-3-5-5. The sequences for machines 2 and 3 are 2-2-2-4-4-5 and 2-2-3-3-5-5, respectively. Note that a micro-period can allocate time for both production and changeover. Although the idle time is not illustrated in the example, it may also occur within the micro-periods.

3. Solution methods

As a first attempt to address the sMSGLSP, one may rely on an MIP solver. However, as reported in (Camargo et al., 2012), the MSGLSP is a hard problem and the exact methods

embedded in such solvers can only find feasible solutions for small-sized instances. The MIP solver does not often find solutions for MSGLSP concerning medium and large instances, which are relevant in practical applications. Exact methods try to find an optimal solution and prove its optimality, but their respective running times increase exponentially according to the instance size. A (meta)heuristic method, on the other hand, is able to find a good solution in a limited time, at the cost of sacrificing the guarantee of finding optimal solutions. Therefore, when choosing between an exact method or a (meta)heuristic method to solve this problem, one should consider the best option to ensure optimality and acceptable running times.

Combinations of exact and (meta)heuristic methods have been studied in depth by researchers. These combinations, usually referred to as hybrid metaheuristics, try to cross-fertilize different optimization strategies. Puchinger and Raidl (2005) classify the hybrid metaheuristic into two main groups. First, the collaborative combinations are algorithms that exchange information, but no algorithm is incorporated into another. The group of collaborative combinations can be sub-divided into sequential, intertwined or parallel executions. The second group contains the integrative combinations, in which the algorithm is a dependent embedded component of another algorithm. This second group is sub-divided into (i) metaheuristics that incorporate exact algorithms and (ii) exact algorithms that incorporate metaheuristics. The second group of hybrid methods is also called Matheuristics (Maniezzo et al., 2010).

In this matheuristics class, various approaches have adopted the strategy of fixing a variable set to certain values in order to create a reduced problem. These approaches decrease the number of freed variables, facilitating the solution to the problem. Soft and hard variable fixing methods have been developed to construct and improve solutions. On the one hand, the soft variable fixing concept is characterized by the non-imposition of the variable set which is fixed. Fischetti et al. (2005) introduced the feasibility pump, in which a feasible solution is obtained from successive roundings based on the linear programming relaxation. As a soft fixing method, the roundings are performed in a non-imposed variable set. The Local Branching - LB (Fischetti and Lodi, 2003) is an iterative local search method that solves reduced problems. The reduced problems are created by introducing a local branching constraint based on an incumbent solution. This constraint ensures that the difference of the 0-1 variables between the incumbent solution and an improved solution (measured by the Hamming distance) is limited to a given positive integer parameter k. Therefore, the variables are soft fixed.

On the other hand, hard fixing methods impose the variables to be fixed. The relax-and-fix

heuristic - as proposed in (Araujo et al., 2007), (Beraldi et al., 2008) and (Ferreira et al., 2009) - solves the lot-sizing and scheduling problems as a sequence of partially relaxed sub-MIPs. Each sub-MIP retains the variables of some time-partitions with integrality constraints and the following partitions are relaxed. The integer variables are progressively fixed at the optimal values obtained in earlier iterations. Relax-and-fix is often used as a construction-heuristic. The fix-and-optimize heuristic, such as those presented in (Sahling et al., 2009; Helber and Sahling, 2010; James and Almada-Lobo, 2011), is an iterative improvement method. At each iteration, the integer variables are fixed at the best previously found value, except for a limited set of integer variables. This fixing strategy relates to the original problem, but by fixing part of the solution, just a partition of the search space is explored. The fix-and-optimize heuristic requires two elements: the original problem to be solved and a partition scheme that can generate new solutions from a current incumbent solution. Besides the traditional time-partitions, James and Almada-Lobo (2011) implemented partitions based on product and machine. However, the choice of the partition to be optimized still requires attention. Problem-oriented methods, such as the relax-and-fix and fix-and-optimize heuristics, bring elements of the problem into the solution procedure, in such cases, the partitions of the problem.

Another two hard-fixing matheuristics, the Relaxation Induced Neighborhood Search - RINS (Danna et al., 2005) and the Distance Induced Neighborhood Search - DINS (Ghosh, 2007) are improvement procedures within the branch-and-bound method in an effort to quickly obtain a good feasible solution, that maintains the optimality guarantee - which is an advantage of the exact method. They share the core of large neighborhood search running under the branch-and-bound exact method. Moreover, their execution depends on the existence of a relaxation and/or an incumbent solution. In the RINS and DINS, a set of decision variables is fixed to a bound and only the remaining variables are optimized by the MIP solver. If the MIP solver finds an improved solution using the reduced problem, it becomes the new incumbent of the original problem. The idea of the RINS is to occasionally devise a subproblem at any node of the branch-and-bound tree that corresponds to a neighborhood of an incumbent solution. The variables whose values are the same in both incumbent and current solution of the linear relaxation are fixed and the reduced problem is then solved. Similarly, DINS fixes the variables using a metric between the variables of the linear relaxation solution at the current node and the variables of the best known feasible solution. RINS performs hard fixing by setting every variable that has the same value as the linear relaxation. Finally, DINS plays around with hard and soft

fixing as it fixes variables through imposed and non-imposed ways. A motivation for using this type of matheuristics is that it produces a good upper bound early in the solution procedure and prunes as much of the branch-and-bound tree as possible. These algorithms exploit the fact that smaller problems can be easily generated from the original problem and can often be efficiently solved by exact methods, as MIP solvers. As previously mentioned, the reduced problem can be defined in different ways and its solution may provide better upper bounds to the original problem.

As far as our problem is concerned, the backlogging feature permits the MIP solver to quickly achieve a feasible solution. Therefore, hybrid methods that improve the upper bound, such as RINS, seem to be a good alternative to solve the sMSGLSP. Bearing this in mind, a novel MIP improvement heuristic is proposed in the next section.

4. HOPS - Hamming-Oriented Partition Search

Our approach is oriented by problem features when it comes to selecting the set of decision variables to be fixed in order to create the reduced problem. When following the problem-oriented methods (e.g. (James and Almada-Lobo, 2011)), the variable set selection is based on clearly recognized problem partitions from the model, such as periods, micro-periods, machines or products. The choice is performed according to a ranking that identifies the most promising partitions to be optimized. The procedure to determine the improvement potential of a partition is designated as Partition Attractiveness Measure and provides a clever partition choice. This idea resembles the consistent variables as defined by (Glover, 1977) to define the variables that repeatedly assume a particular value. In the partition case, the idea can be extended to the variable set that belongs to the partition. Thus, following the variable fixing concept, HOPS is classified as a hard fixing method since it creates the reduced problem by freezing the values of certain variables.

To sum up, we bring and compile ideas to propose a new matheuristic called HOPS (Hamming-Oriented Partition Search), which incorporates interesting strategies to solve large problems by reducing them to smaller and more tractable sub-problems.

The algorithm scheme is divided into three parts. The first part illustrates the main method incorporated in the branch-and-bound exact method (Section 4.1). The second part presents the function that defines the metric to choose the variable partition that will be fixed (Section 4.2). Finally, in the third part we show the necessary steps to fix the decision variables in order to

create and solve the reduced problem (Section 4.3).

4.1. The HOPS core

Algorithm 1 shows the pseudo code that must be embedded in the branch-and-bound scheme. The HOPS - Hamming-Oriented Partition Search may be performed after the branch-and-bound has processed each node. Let P be the problem to be solved by the MIP solver with HOPS. (For our previously addressed problem, P is the sMSGLSP):

$$P = \textbf{Minimize} \quad \sum_{m=1}^{M}\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{r=1}^{R}\sum_{t'=1}^{T} c_{itt'} \cdot X_{mitrt'} + \sum_{m=1}^{M}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{t=1}^{T}\sum_{r=1}^{R} \sigma_{mij} \cdot Z_{mijtr}$$

$$\textbf{subject to} \quad (2) - (16).$$

The search for improvements is performed every time a new incumbent x^0 is found throughout the exploration of the tree. As previously explained, x^0 can be improved by freeing one partition of that solution and optimizing the reduced problem. The partition choice relies on a ranking based on the frequency of the variable changes - belonging to that partition - in the past iterations. In other words, one can determine the attractiveness of each partition to be optimized by calculating a measure for the stability of the 0-1 variables throughout the search. Intuitively, one may say that every feasible solution found along the branch-and-bound tree contains parts of the optimal solution (or even the entire optimal solution). Therefore, partitions with stable variable values might be part of the optimal solution and do not need to be iteratively optimized. On the other hand, unstable variables should be more frequently re-optimized to promote convergence to the optimal solution. Some elements are required in the procedure and must be defined. Let $\mathbf{z}$ be the smooth array that saves the history of the solutions obtained along the branch-and-bound tree. The array storage values are obtained by using the previous individual decision variable values. Each new feasible solution is incorporated into $\mathbf{z}$ with an importance factor α. This aims to smoothly integrate the solutions without loosing their memory. $\mathbf{z}$ is called *smooth array* as the most recent solutions are given more weight than the less recent ones and the respective weights decay exponentially as their recency lowers. In order to create the reduced problem, the partitions to be considered by HOPS are listed in set $\mathcal{A}$. For the sMSGLSP example, $\mathcal{A}$ may contain machine, period, micro-period and/or product partitions. The partition selection is based on the Partition Attractiveness Measure H, which determines the distance between each partition's new incumbent and array $\mathbf{z}$.

Algorithm 1: HOPS

Data: Problem P, incumbent solution x^0, smooth array $\mathbf{z}$, parameter α, set of partitions $\mathcal{A}$, and Partition Attractiveness array H.
Result: A new feasible solution $\bar{x}$ to P.

```
1 begin
2     while Branch-and-bound active do
3         if x^0 is a new incumbent to P then
4             H ⟵ PartitionAttractivenessMeasure(x^0, z, α, A)
5             x̄ ⟵ Fix-and-Optimize(P, H, x^0, A)
6             if x̄ is feasible then
7                 inject x̄ in the branch-and-bound as a new incumbent
```

The HOPS core consists of two main procedures. First (line 4), if the new incumbent found by the branch-and-bound is the best known solution, the attractiveness for the optimization (H) of each partition is determined. Second (line 5), the reduced problem is created based on the original problem, the new incumbent solution x^0 and the partitions to be fixed following the Partition Attractiveness Measure. The reduced problem is also solved by an MIP solver. In case a feasible solution to the reduced problem is found, then it is injected into the branch-and-bound tree of the original problem P as the new incumbent and the new upper bound. Therefore, when solving the reduced problem, better upper bounds are searched for P. It is important to note that the optimal solution of the reduced problem is at least as good as the best solution obtained so far. By fixing the variables of the partitions that usually undergo no (or few) changes along the branch-and-bound exploration, one may reach good partial solutions and the optimization of the remaining part of the solution may improve the upper bound faster.

HOPS can be invoked in situations other than when a new incumbent is found. For example, in line 3, the number of nodes between two invocations may be another condition to execute HOPS. This alternative provides the control over the frequency of applying the improvement heuristic. In this case, at least one feasible solution must have been previously generated.

4.2. Partition attractiveness measure

The choice of which variables should be fixed is the main difference between HOPS and other methods from the literature. RINS and DINS fix variables without considering the relationships between them. In contrast, our approach takes into account the inherent relationship between the variables presented in the problem. The relationship is expressed as partitions of the problem, such as periods, products or machines. In the sMSGLSP example, if only the machine partition is considered, $\mathcal{A} = \{\text{machine } m^1,\ \text{machine } m^2,\ \ldots,\ \text{machine } m^M\}$, where *machine* m^1 denotes

the set of decision variables related to machine 1. This setting is applied when other partitions are considered simultaneously with this partition. The procedure to define the attractiveness measure for each partition to be optimized is illustrated in Algorithm 2. To understand the procedure, let $\mathcal{S} = \left\{j : x_j^0 = 1\right\}$ denote the set of binary variables with a non-zero value of a new solution x^0 found by the branch-and-bound.

Algorithm 2: PartitionAttractivenessMeasure

Data: Incumbent solution x^0, smooth array $\mathbf{z}$, parameter α, and set of partitions $\mathcal{A}$.
Result: Partition Attractiveness array H updated.

1 **begin**
2 **foreach** *partition* $\mathcal{A}_k \in \mathcal{A}$ **do**
3 $$H(\mathcal{A}_k) = \frac{\sum_{j \in (\mathcal{A}_k \cap \mathcal{S})} \left(x_j^0 - z_j\right) + \sum_{j \in (\mathcal{A}_k \setminus \mathcal{S})} z_j}{|\mathcal{A}_k|}$$
4 **foreach** j **do**
5 $z'_j \longleftarrow \alpha \cdot x_j^0 + (1 - \alpha) \cdot z_j$
6 $z_j \longleftarrow z'_j$

The Partition Attractiveness Measure algorithm ranks the partitions every time a new feasible solution is found. Moreover, the history of solutions (smooth array) found must be updated. The first step (lines 2 and 3) returns the information on how much each variable value has changed in comparison to the feasible solution history. The Hamming distance between the new incumbent and the smooth array $\mathbf{z}$ is used and, by grouping the Hamming distances of the variables, one can quantify and rank the partitions with more changes. It is important to note that the same variable might belong to more than one partition. This grouping strategy defines the stable partitions, that is, the average Hamming distance of all variables belonging to a partition defines its stabilization level. By comparing the solution history and the new solution, an attractiveness measure for each partition optimization is defined. The variable values of those partitions that have no (or few) changes are considered changeless, whereas the other part must be stabilized. Finally, the feasible solution history in smooth array z is updated with the new solution x^0 (lines 4, 5 and 6). The linear combination creates the history in which each new x^0 has an importance factor α over z.

For example, let $\mathbf{z} = \{0, 0, 1, 0, 0.6, 0.4, 0, 0\}$ be the smooth array that must be updated with the solution $x = \{0, 0, 1, 0, 0, 1, 0, 1\}$ and importance factor $\alpha = 0.6$. The new smooth array is $\mathbf{z'} = \{0, 0, 1, 0, 0.24, 0.76, 0, 0.6\}$. Let $\mathcal{A}_k$ contain six partitions as follows: $A_1 \subseteq \{x_1, x_2\}$, $A_2 \subseteq \{x_3, x_4\}$, $A_3 \subseteq \{x_5, x_6\}$, $A_4 \subseteq \{x_7, x_8\}$, $A_5 \subseteq \{x_1, x_4, x_5, x_8\}$, $A_6 \subseteq \{x_2, x_3, x_6, x_7\}$. The differ-

ence between **z'** and the newest solution $x' = \{1, 0, 1, 0, 0, 1, 0, 0\}$ is $H(\mathcal{A}_k) = \{0.5, 0, 0.24, 0.3, 0.46, 0.06\}$.

4.3. Fix-and-Optimize

This section introduces a heuristic that optimizes the reduced problems. Similarly to the fix-and-optimize heuristic used in (James and Almada-Lobo, 2011) or (Sahling et al., 2009), our design creates the reduced problem ($P1$) from the original problem (P) fixing all, but a predefined partition of the solution. The solution to the reduced problem $P1$ respects all the constraints of the original problem. In addition, the objective function value is limited to be lower than the upper bound ($f(x^0)$) and higher than the lower bound ($Z(P)$) in the original problem. It implies that a better solution for P is obtained or the reduced problem delivers an infeasibility status. The variables are fixed, leaving out those belonging to the chosen partition.

Algorithm 3: Fix-and-Optimize

Data: Problem P, Partition Attractiveness array H, incumbent solution x^0, and set of partitions $\mathcal{A}$.
Result: A new feasible solution $\bar{x}$ to P.

```
begin
    Define partition A_k from A that must be improved
    Let ε be a small number
    Let P1 be a reduced problem from P with additional constraints:
        x ⟵ x^0 ∀ x ∉ A_k
        f(x) ≤ f(x^0) − ε
        f(x) ≥ Z(P)
    Solve(P1)
    if a new solution x̄ is obtained from P1 then
        return a new feasible solution x̄ to P
    else
        return ∅
```

Obviously, even by using the Partition Attractiveness Measure to choose the best portions to be optimized, the selection of the partition variables requires a strong trade-off. The number of variables of the freed partition directly influences the improvement of the solution. In fact, it plays with diversification and intensification mechanisms. The higher the number of freed variables, the wider the search space, which opens the possibility to diversify the solution. In case the solution space is small, the available computational time is used to intensify the search in the region closer to the incumbent solution. On the other hand, too large partitions might result in excessive running times to solve the reduced problem with the MIP solver. In order to avoid this scenario, the MIP solver is aborted when the established time limit is reached and the best solution (if available) is considered.

4.4. Accelerators

This section describes a set of six strategies to boost the HOPS method over its core. These strategies mitigate the computational effort and add HOPS invocations.

HOPS node invocation

HOPS is initially invoked by default when the first MIP feasible solution is found. It can also be called at every node in the tree or several consecutive times at the same node. Each invocation comprehends the improvement attempt of one partition. Parameter $nl \geq 1$ indicates the number of nodes explored by the branch-and-bound between two consecutive invocations. In case HOPS is invoked without a new feasible solution since the last call, the history of solutions found remains unchanged and the procedure to update the smooth array $\mathbf{z}$ is skipped.

Unambitious choice

The choice of partition to be fixed is based on the attractiveness measure of the partitions. The higher the attractiveness measure, the higher the unstability of the partition and its priority to be fixed. Preliminary tests showed that HOPS performs slightly better when the partition choice is not pure greedy, that is, the selection is made from a restricted candidate list of partitions based on the value of the attractiveness measure.

Prohibited partition list

With an incumbent solution x^0 at hand, the same reduced problem is generated consecutively if partition $\mathcal{A}_k$ does not change. By designing a prohibited partition list, the method invests computational efforts only in the unexplored partitions. Each explored partition is incorporated in the list avoiding a future generation of the same reduced problem. In case a new incumbent solution is found, the prohibited partition list is emptied.

Infeasibility invocation

Constraints limiting the upper and lower bounds in the reduced problem may accelerate its solution. On the one hand, it can return an improved solution quickly. However, the reduced problem can be too tight and may return an infeasibility status.

Each invocation is expected to return one new improved solution, therefore an infeasibility invocation is designed to generate another reduced problem, if an infeasibility response is returned in a minimum time.

Partition oscillating growth

The *prohibited partition list* and the *infeasibility invocation* enable a quicker exploration of partition set $\mathcal{A}$. If the improvement of the incumbent solution fails, the partitions can be combined

in pairs, triples or larger clusters to expand the search space. If an improved solution is found, the partition combinations move back and the prohibited partition list is again emptied.

HOPS turn on/off

HOPS is turned on when the MIP exploration tree finds its first feasible solution. If the *partition oscillating growth* does not retract, HOPS can be turned off. This adaptive tool permits the branch-and-bound to invest more time in exploring the original problem tree. It may speed up the lower bound improvement and/or find a new incumbent solution to the problem. In case of a new incumbent solution, HOPS is turned on.

5. Computational results

This section reports two computational tests performed to validate the HOPS method. Firstly, HOPS was tested on a well-known lot-sizing problem and its performance was compared against that of the state-of-the-art method proposed by Muller et al. (2012). Our goal is to show HOPS's competitiveness on a standard lot-sizing problem, for which best results are available in the literature. Secondly, HOPS was run on a set of instances generated based on real data from a spinning industry and compared to the results provided by RINS and LB. The purpose is to analyze the HOPS front results against other hard and soft variable fixing strategies. HOPS was implemented in C++ and used the ILOG CPLEX version 12.1 with default settings to solve both the original and reduced problems. The experiment settings are described in the following sections.

5.1. HOPS on a classical lot-sizing problem

In the first test, HOPS was executed for a widely used LSP, first proposed by Trigeiro et al. (1989). Several variants of this problem were introduced in recent years, such as the variant without setup costs in (Sural et al., 2009). To the best of our knowledge, the best results to this LSP variant were published by Muller et al. (2012). The authors consider two instance classes: Heterogeneous and homogeneous instances. Instances of 30 and 45 periods, and of 12 and 24 products are tested, totalizing 40 instances. For further details, the reader is referred to (Muller et al., 2012).

The experiments of Muller et al. (2012) were performed on a 2.66 GHz Intel (R) Xeon (R) X5355 machine with an 8 GB memory using ILOG CPLEX version 12.1 with default settings. The HOPS tests were conducted on a 3.0 GHz Intel Q9650, 4GB RAM and Ubuntu 12.10

operational system, which is considered to be a similar machine for these tests according to SPEC (www.spec.org).

Parameters for HOPS and its accelerators were defined by preliminary tests and are provided hereafter. The reduced problems were created by freeing a variable set of the incumbent solution. The variable partitions must be clearly recognized in the problem. For the LSP with setup times and no setup costs, the variable sets of periods and products were used in the HOPS computational tests. The partition attractiveness, which determines which partition is freed to create the reduced problem, was measured from the smooth array $\mathbf{z}$. In these tests, a new feasible solution x^0 was incorporated into the smooth array with an importance factor $\alpha = 0.4$. For the *HOPS node invocation*, the parameter to define the nodes explored between two consecutive invocations was set to $nl = 500$. The minimum time parameter to solve a reduced problem required in *Infeasibility invocation* was set to 2 seconds. Nevertheless, the maximum time spent on searching for an improved upper bound was 15 seconds. HOPS was turned off after the partition list has been explored twice without any new solutions been found. A variation of the parameters used in the computational tests to validate HOPS was also tested. Nevertheless, it is worth to note that parameter calibration was not a priority in this work. Its emphasis is rather on the presentation of a new matheuristic.

In order to resemble the computational experiments proposed by Muller et al. (2012), the established time limit was 300 seconds and the results were calculated from the average of 10 runs for both the heuristic of Muller et al. (2012) and HOPS. Table 1 shows the compiled results for heterogeneous and homogeneous classes, in which $Gap = 1 - lb/ub$.

Table 1: Compiled gap results for the heterogeneous and homogeneous classes.

Class	Heterogeneous class		Homogeneous class	
	(Muller et al., 2012)	HOPS	(Muller et al., 2012)	HOPS
G*-30	7.52	7.08	12.40	12.39
G*-45	7.55	7.16	12.02	12.03

The results clearly indicate the competitiveness of HOPS to solve a standard lot-sizing problem. On average, HOPS is slightly superior to the heuristic by Muller et al. (2012) for the heterogeneous class and for the 30-periods instances that belong to the homogeneous class.

5.2. HOPS versus matheuristics of commercial implementations to solve the sMSGLSP

In the second test, the sMSGLSP was solved on instances based on real-world data. HOPS was analyzed against other hard and soft variable fixing strategies. The experimental results

were compared to the results given by RINS and LB.

Most of the practical cases present difficulties to obtain stable static data. The real-world case delivers a relative low number of instances and statical parameters. Therefore, to demonstrate the HOPS performance with respect to the solution quality over the problem, it is important to provide the ground for systematic testing. An instance generator was designed to supply an extensive set of instances that reflects the characteristics of real-world cases.

5.2.1. Problem-instance generation

By drawing an instance generator, the parameters can be systematically varied and a large number of problem instances with specific desired properties are generated. Here, the problem is posed with a constructive procedure. Period after period, product setups are assigned to each machine and the maximum level for the production of each yarn is determined. The other main points to generate the instances are randomly designed, but their proportions are similar to the real data. Given a set of parameters N (number of yarns), K (number of families), M (number of machines) and T (number of periods), the remaining data are determined as follows:

- all yarns are randomly assigned to a product family;
- the set of yarns of the same family that might be allocated to each machine is randomly chosen;
- the setup times (with triangle inequalities respected) and the processing times for all yarns on each machine are randomly generated;
- all costs are derived from the opportunity cost per yarn package unit;
- the demand of a yarn and the capacity of the first-stage machine are chosen with respect to the capacity requirements.

In our tests, the evaluated instances had the following parameters: the number of products N was four, five, seven or nine, and can belong to two or three product families K. The number of spinning machines M considered was three, five, seven or nine. The number of periods T was equal to five and the number of micro-periods R was fixed to three, which represents the working days with three shifts. The generator was constructed to provide feasible solutions for the case without backlogging, that is, the whole yarn demand can be satisfied in the planning horizon. In order to evaluate the backlogging feature (presented in real-world cases), factors

$df = \{1.0; 1.5; 2.0\}$ were adopted to inflate the yarn demand. For each combination $M/N/K/df$, ten different instances were randomly generated, making a total set $\mathcal{P}$ of $n_p = 840$ instances.

5.2.2. HOPS versus RINS and Local Branching

The MIP solver CPLEX has the RINS implemented and activated by default in version 12.1. Similarly, the Local Branching is part of the CPLEX package, but must be activated by setting parameter `IloCplex::LBHeur` to 1. HOPS was implemented in C++ and used CPLEX to solve the original and reduced problems. In order to ensure a fair comparison, RINS was disabled from CPLEX when running LB and HOPS by setting parameter `IloCplex::RINSHeur` to -1. No other CPLEX parameter was changed.

Similar to the test in Section 5.1, the parameters for HOPS and its accelerators were defined by preliminary tests. The variable partition used in these tests were micro-period and machine decomposition. The *Partition oscillating growth* accelerator was implemented allowing for the period partition implicit usage in case the micro-period partition size increased. The minimum time to solve a reduced problem was set to 5 seconds and the maximum time to 36 seconds. Other parameters were set to the parameter indicated in Section 5.1.

All these computational tests were conducted on an Intel Xeon (2 GHz) with 5 Gb of random access memory, running under Linux. In all cases, the methods were unable to compute the optimal solution within a time limit of one CPU hour for each problem instance. Thus, the comparison was focused on the best solution found and the respective optimality gap considering the time limit of one hour.

Despite the fact that CPLEX is able to provide the optimal solution for a problem (if it exists), in some cases this is not possible to be achieved in reasonable time. For the sMSGLSP problem and the instances provided in this validation, the optimality proof was not complete within the time limit. It is well-known that the newest CPLEX versions have a non-deterministic exploration of the tree nodes and that the truncated results may therefore be divergent. Thus, each instance was run five times to dilute any possible dissonant solution. The results reported below are the average of the five runs.

The instances were grouped into three different sets following the factors that determine the yarn demand (df). Three classes are presented: Class A details the instances where $df = 1.0$, Class B and Class C lay out the instances where $df = 1.5$ and $df = 2.0$, respectively. This division makes the effectiveness of the different methods visible in three different yarn demand scenarios.

Three measures were used to evaluate the performance of the methods. After one hour of running time, the upper bound (best feasible solution found) and its respective optimality gap were collected. In Tables 2, 3 and 4, the ten different instances of each class $M/N/K/df$ are grouped together. On each line, "$\#UB$" represents the number of instances for which the method found the best upper bound. If the same best upper bound was found by two methods, it is written up for both. Similarly, "$\#GAP$" refers to the number of instances for which the method found the smallest relative gap. The relative gap provided by CPLEX is defined as $GAP_{p,s} = 1 - LB_{p,s}/UB_{p,s}$, where $LB_{p,s}$ is the lower bound and $UB_{p,s}$ the upper bound for the instance $p \in \mathcal{P}$ found by method s after 1 hour of running time. The third measure, "$r_{p,s}$" depicts the aggregated relative performance of each method for each instance class and refers to the ratio of the solution found by each method over the best solution reported by the three methods all together. For the HOPS example, $r_{p,HOPS} = UB_{p,HOPS}/min(UB_{p,HOPS}, UB_{p,RINS}, UB_{p,LB})$. This metric is an average of the ten instances of each class. In the last line, "*Total* $\#UB$" represents the percentage of instances in which the method found the best solution. "*Total* $\#GAP$" is the percentage of instances for which the method reported the minimum gap. Subsequently, "*Total* $r_{p,s}$" is the ratio average of the 280 instances.

From the results shown in Table 2, HOPS achieves the best solution for 55% of 280 instances belonging to Class A. The "*Total* $r_{p,HOPS}$" value of 1.041 indicates that the solution values found by HOPS are, on average, 4.1% worse than the best solutions. On the other hand, the solutions from RINS and LB are, on average, 20.6% and 41.8% worse than the best solutions. The HOPS optimality gaps are as good as the RINS gaps. It is important to note that as the problem dimension increases, the superiority of HOPS over the other two methods seems to be clearer. For every instance of class 9/9/3/1.0, HOPS presents the best solution and gap.

Table 3 shows the results of Class B, in which yarn demands are increased by factor $df = 1.5$. For this class, backlogging may occur to satisfy the demand constraints. Once again HOPS reached the best objective value for 45% of 280 instances against 42% by RINS. Nevertheless, the optimality gap provided by HOPS is the best for 85% of the instances. These results indicate that HOPS finds good solutions quicker and concentrates more efforts on improving the lower bound. In terms of solution quality, both HOPS and RINS found solutions not worse than 0.8% (on average) to the best solutions. Clearly, the LB method yields a poor relative performance

Table 2: Comparison among HOPS, RINS and LB for Class A.

			HOPS			RINS			LB		
M	N	K	#UB	$r_{p,s}$	#GAP	#UB	$r_{p,s}$	#GAP	#UB	$r_{p,s}$	#GAP
3	4	2	7	1.016	4	4	1.013	6	5	1.023	7
3	5	2	4	1.018	2	5	1.010	4	3	1.026	7
3	7	2	5	1.038	1	1	1.068	5	4	1.030	5
3	9	2	4	1.017	2	5	1.026	5	1	1.056	3
5	4	2	6	1.034	3	2	1.043	6	2	1.032	3
5	5	2	4	1.010	3	4	1.031	6	2	1.067	1
5	7	2	7	1.003	4	3	1.096	5	0	1.158	4
5	9	2	6	1.031	6	3	1.173	3	1	1.285	2
7	4	2	6	1.026	2	1	1.049	4	3	1.044	7
7	5	2	3	1.055	3	6	1.050	6	1	1.097	2
7	7	2	7	1.023	8	3	1.092	4	0	1.376	0
7	9	2	5	1.085	4	4	1.187	5	1	1.481	1
9	4	2	4	1.030	2	3	1.058	6	3	1.071	4
9	5	2	4	1.023	3	4	1.082	5	2	1.072	4
9	7	2	6	1.083	6	3	1.100	4	1	1.484	1
9	9	2	5	1.027	4	5	1.138	6	0	2.198	1
3	5	3	5	1.016	2	2	1.051	5	5	1.012	6
3	7	3	7	1.025	3	0	1.130	3	3	1.048	4
3	9	3	1	1.034	3	1	1.040	5	8	1.016	6
5	5	3	5	1.020	3	3	1.081	4	2	1.076	5
5	7	3	6	1.153	4	4	1.093	7	0	1.300	0
5	9	3	5	1.082	4	3	1.296	3	2	1.267	3
7	5	3	7	1.017	6	2	1.128	3	1	1.368	2
7	7	3	5	1.079	7	2	1.451	3	3	1.798	3
7	9	3	6	1.132	6	4	1.324	5	0	2.772	0
9	5	3	7	1.024	8	3	1.110	4	0	1.389	0
9	7	3	8	1.041	8	2	2.244	2	0	3.057	0
9	9	3	10	1.000	10	0	2.593	0	0	3.099	0
Total			55%	1.041	43%	29%	1.206	44%	19%	1.418	29%

Table 3: Comparison among HOPS, RINS and LB for Class B.

			HOPS			RINS			LB		
M	N	K	#UB	$r_{p,s}$	#GAP	#UB	$r_{p,s}$	#GAP	#UB	$r_{p,s}$	#GAP
3	4	2	7	1.002	9	5	1.002	4	1	1.004	3
3	5	2	7	1.002	8	4	1.002	6	2	1.005	5
3	7	2	7	1.000	10	1	1.011	2	4	1.004	4
3	9	2	5	1.004	7	3	1.006	3	2	1.007	3
5	4	2	2	1.004	9	4	1.004	3	5	1.002	4
5	5	2	4	1.004	9	2	1.002	1	4	1.001	1
5	7	2	4	1.004	8	4	1.002	4	2	1.016	1
5	9	2	5	1.008	9	5	1.005	1	0	1.034	1
7	4	2	3	1.004	8	4	1.004	4	3	1.004	1
7	5	2	7	1.003	10	0	1.008	0	3	1.015	0
7	7	2	7	1.008	10	3	1.015	1	0	1.034	0
7	9	2	4	1.013	9	6	1.011	3	0	1.047	0
9	4	2	3	1.007	9	5	1.006	0	2	1.007	1
9	5	2	5	1.007	9	3	1.014	4	2	1.016	0
9	7	2	2	1.013	8	7	1.003	2	1	1.027	1
9	9	2	3	1.021	8	6	1.006	3	1	1.064	0
3	5	3	4	1.001	9	5	1.004	0	2	1.004	4
3	7	3	4	1.004	10	2	1.012	1	4	1.006	3
3	9	3	4	1.008	6	4	1.006	2	2	1.012	4
5	5	3	4	1.005	9	5	1.01	0	1	1.016	2
5	7	3	2	1.021	7	7	1.004	3	1	1.026	1
5	9	3	3	1.017	7	7	1.003	4	0	1.025	1
7	5	3	7	1.006	9	1	1.015	3	2	1.016	1
7	7	3	5	1.010	9	3	1.007	3	2	1.032	1
7	9	3	6	1.006	10	4	1.008	0	0	1.055	0
9	5	3	3	1.013	8	6	1.008	3	1	1.024	0
9	7	3	2	1.015	7	7	1.009	4	1	1.081	0
9	9	3	6	1.010	8	4	1.034	2	0	1.141	0
Total			45%	1.008	85%	42%	1.008	24%	17%	1.026	15%

for the whole set instances.

Table 4 shows the results from Class C. Similarly to Class B, the solution qualities for both HOPS and RINS are equivalent and deviate on average 0.2% from the best solutions. However,

HOPS outperforms RINS and LB when it comes to find the best solutions (53%) and the best optimality gaps (94%).

Table 4: Comparison among HOPS, RINS and LB for Class C.

			HOPS			RINS			LB		
M	N	K	#UB	$r_{p,s}$	#GAP	#UB	$r_{p,s}$	#GAP	#UB	$r_{p,s}$	#GAP
3	4	2	6	1.000	9	5	1.000	6	3	1.000	5
3	5	2	5	1.001	9	5	1.001	3	1	1.000	4
3	7	2	7	1.000	8	2	1.002	5	2	1.001	5
3	9	2	5	1.002	9	5	1.001	5	1	1.002	3
5	4	2	5	1.000	9	3	1.000	1	4	1.000	1
5	5	2	6	1.001	10	2	1.002	2	2	1.001	2
5	7	2	5	1.002	9	3	1.001	3	2	1.004	3
5	9	2	5	1.001	8	3	1.002	2	2	1.004	2
7	4	2	6	1.001	10	3	1.001	3	1	1.002	3
7	5	2	6	1.001	9	3	1.000	6	1	1.003	3
7	7	2	4	1.002	9	5	1.003	1	1	1.008	1
7	9	2	4	1.005	9	5	1.003	4	1	1.014	0
9	4	2	5	1.001	10	3	1.002	4	2	1.005	1
9	5	2	7	1.001	10	2	1.002	1	1	1.005	1
9	7	2	6	1.001	10	3	1.004	1	1	1.011	0
9	9	2	3	1.003	10	6	1.003	4	1	1.018	1
3	5	3	6	1.001	10	1	1.002	4	3	1.001	3
3	7	3	7	1.001	10	2	1.004	3	1	1.003	1
3	9	3	7	1.000	10	2	1.004	3	1	1.005	1
5	5	3	8	1.000	10	1	1.003	3	1	1.006	2
5	7	3	4	1.003	10	3	1.003	2	3	1.008	2
5	9	3	4	1.004	10	5	1.004	6	1	1.011	2
7	5	3	5	1.002	10	5	1.002	1	0	1.006	2
7	7	3	4	1.005	10	5	1.002	2	1	1.010	2
7	9	3	4	1.007	9	5	1.005	2	1	1.027	1
9	5	3	8	1.002	10	2	1.005	1	0	1.006	0
9	7	3	4	1.010	8	5	1.004	4	1	1.014	2
9	9	3	2	1.014	7	8	1.002	6	0	1.044	0
Total			53%	1.002	94%	36%	1.002	31%	14%	1.008	19%

Below, the performance profiles (Dolan and Moré, 2002) are plotted to compile the information from Tables 2, 3 and 4. Given the set $\mathcal{P}$ of 840 instances and $s \in \mathcal{S} = [HOPS,\ RINS,\ LB]$ methods under evaluation, the performance ratio is determined as:

$$r_{p,s} = \frac{UB_{p,s}}{min\left(UB_{p,HOPS};\ UB_{p,RINS};\ UB_{p,LB}\right)}.$$

Let $\rho_s(\tau)$ be the proportion of instances from $\mathcal{P}$ that were solved by method s with relative performance within a factor $\tau \geq 1$, that is, $\rho_s(\tau)$ is expressed as:

$$\rho_s(\tau) = \frac{|\mathcal{P} : r_{p,s} \leq \tau; p \in \mathcal{P}|}{|\mathcal{P}|}.$$

Figure 5 summarizes the comparison of the performance over set $\mathcal{P}$. By taking into account all 840 instances, HOPS found the best solution for 50.9% of the instances, RINS for 35.8% and LB for 16.6%. Regarding solution robustness, HOPS has proved to be by far the best. For 95% of the instances, HOPS provided solutions that dist less than 7.9% of the best solutions. On the other hand, RINS and LB solutions are up to 26.0% and 66.6% away from the best

solutions, respectively. In the worst case, when $\rho_s(\tau)$ is equal to 100, there is a problem p such that $r_{p,HOPS} = 1.79$ (that is, HOPS is 79% away from the best solution). For the RINS and LB cases, $r_{p,RINS} = 6.55$ and $r_{p,LB} = 8.88$ (not shown in the trucated graphic).

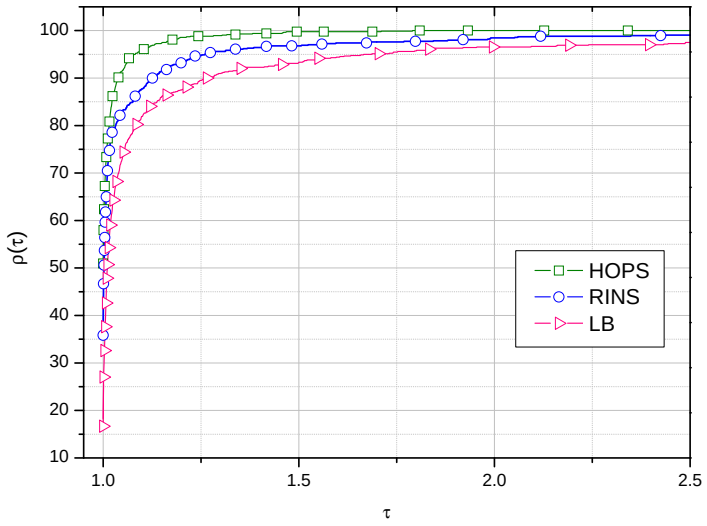

Figure 5: Performance profiles.

HOPS is indeed more robust and can provide better solutions than RINS and LB. Figure 6 illustrates the solution value evolution along the search for HOPS, RINS and LB on a particular instance of Class C. We recall that the backlogging feature benefits the CPLEX to provide a quick feasible solution. All methods start from the same solution at node 0. HOPS improves the solution faster not only at the beginning of the run (before 600 seconds), but also throughout the one hour of computational time.

6. Final remarks

This paper presents an integrated industrial problem at a spinning industry. The first-stage process deals with cotton blending whereas the second stage concerns the yarn production. The first-stage production must be synchronized with the second-stage product, in product quality terms. The problem becomes harder with the occurrence of sequence-dependent setup features. A mathematical modeling aims to determine the production plan of the first and second stages with minimum setup changeover, inventory and backlogging costs while trying to meet the due dates of the required demand.

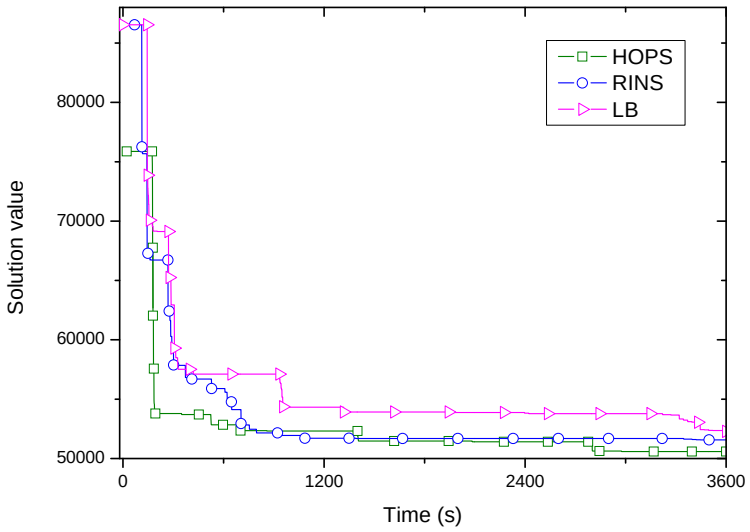

Figure 6: Solution quality evolution over time for a Class C example.

This two-stage lot-sizing and scheduling problem is approached by a new matheuristic. The HOPS - Hamming-Oriented Partition Search is an exact method that incorporates a heuristic, in which a fix-and-optimize procedure finds better feasible solutions to boost the branch-and-bound procedure. The Partition Attractiveness Measure is used to choose the variable set that is considered unstable and can potentially improve an incumbent solution. The new solution is injected into the branch-and-bound tree, accelerating the optimality proof.

HOPS was first tested on instances of a well-known lot-sizing problem and its performance was positively assessed against recent results from the literature. For the production planning problem of the spinning industry, a large set of instances that assumes proportions similar to real-world data was designed to validate the new matheuristic. A comparison of HOPS against commercial implementations of improvement matheuristics shows that HOPS outperforms both RINS and LB in obtaining good feasible solutions within a certain time limit. The computational results support the argument that in the context of problems addressing production backlogging, exact methods with improvement procedures are good options for providing better solutions. HOPS results show that a problem-oriented improvement procedure can provide better solutions for problems with a known partition structure (in this case, machine, period, micro-period and product).

Our work is in line with research gaps pointed out in the literature by addressing lot sizing

and scheduling considering backlogging and sequence-dependent setups (Zhu and Wilhelm, 2006), multiple stages with parallel machines and synchronization of resources (Clark et al., 2011) and problem-oriented methods (James and Almada-Lobo, 2011).

Although HOPS has been designed to solve the production planning in the spinning industry, some of the ideas can be applied to different lot-sizing and scheduling problems or other problems not covered by this research subject. The main requirement is to recognize the "natural" partitions of the problem. We believe HOPS will serve as a guidance for solving many applications.

Acknowledgments

The authors would like to thank the anonymous referees for their valuable comments. This research was funded by FAPESP (2008/09953-2) and (2012/20773-1), CNPq (300713/2010-0) and CAPES (BEX-1545/11-6) from Brazil, and FP7-PEOPLE-2009-IRSES project no. 246881 and Erasmus Mundus External Cooperation Windows Programme from the European Commission. This project was also partially supported by funds granted by the ERDF through the Programme COMPETE and by the Portuguese Government through FCT - Foundation for Science and Technology, project PTDC/EGE-GES/117692/2010. The authors are indebted to engineer Carlos Eduardo Rossignolo for sharing his expertise in the spinning production system.

References

ABIT (2011). Brazilian textile and apparel industry association. http://www.abit.org.br (25/01/2012).

Almada-Lobo, B., Oliveira, J. F., and Carravilla, M. A. (2008). Production planning and scheduling in the glass container industry: A VNS approach. *International Journal of Production Economics*, *114*, 363–375.

Araujo, S. A., Arenales, M. N., and Clark, A. R. (2007). Joint rolling-horizon scheduling of materials processing and lot-sizing with sequence-dependent setups. *Journal of Heuristics*, *13*, 337–358.

Belvaux, G., and Wolsey, L. A. (2001). Modelling practical lot-sizing problems as mixed-integer programs. *Management Science*, *47*, 993–1007.

Beraldi, P., Ghiani, G., Grieco, A., and Guerriero, E. (2008). Rolling-horizon and fix-and-relax heuristics for the parallel machine lot-sizing and scheduling problem with sequence-dependent set-up costs. *Computers & Operations Research*, *35*, 3644–3656.

Brahimi, N., Dauzere-Peres, S., Najid, N. M., and Nordli, A. (2006). Single item lot sizing problems. *European Journal of Operational Research*, *168*, 1–16.

Camargo, V. C. B., Toledo, F. M. B., and Almada-Lobo, B. (2012). Three time-based scale formulations for the two-stage lot sizing and scheduling in process industries. *Journal of the Operational Research Society*, *63*, 1613–1630.

Clark, A. R., Almada-Lobo, B., and Almeder, C. (2011). Editorial on lotsizing and scheduling: industrial extensions and research opportunities. *International Journal of Production Research*, *49*, 2457–61.

Danna, E., Rothberg, E., and Pape, C. L. (2005). Exploring relaxation induced neighborhoods to improve MIP solution. *Mathematical Programming*, *102*, 71–90.

Dolan, E., and Moré, J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, *91*, 201–213.

Drexl, A., and Kimms, A. (1997). Lot sizing and scheduling - survey and extensions. *European Journal of Operational Research*, *99*, 221–235.

Ferreira, D., Morabito, R., and Rangel, S. (2009). Solution approaches for the soft drink integrated production lot sizing and scheduling problem. *European Journal of Operational Research*, *196*, 697–706.

Fischetti, M., Glover, F., and Lodi, A. (2005). The feasibility pump. *Mathematical Programming*, *104*, 91–104.

Fischetti, M., and Lodi, A. (2003). Local branching. *Mathematical Programming*, *98*, 23–47.

Fleischmann, B., and Meyr, H. (1997). The general lotsizing and scheduling problem. *OR Spectrum*, *19*, 11–21.

Gao, L.-L., Altay, N., and Robinson, E. P. (2008). A comparative study of modeling and solution approaches for the coordinated lot-size problem with dynamic demand. *Mathematical and Computer Modelling*, *47*, 1254–1263.

Ghosh, S. (2007). DINS, a MIP improvement heuristic. In *Integer Programming and Combinatorial Optimization* (pp. 310–323).

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decisions Science*, *8*, 156–166.

Guimarães, L., Klabjan, D., and Almada-Lobo, B. (2012). Annual production budget in the beverage industry. *Engineering Applications of Artificial Intelligence*, *25*, 229 – 241.

Hamming, R. (1950). Error-detecting and error-correcting codes. *Bell System Technical Journal*, *292*, 147–160.

Helber, S., and Sahling, F. (2010). A fix-and-optimize approach for the multi-level capacitated lot sizing problem. *International Journal of Production Economics*, *123*, 247–256.

James, R. J., and Almada-Lobo, B. (2011). Single and parallel machine capacitated lotsizing and scheduling: New iterative MIP-based neighborhood search heuristics. *Computers & Operations Research*, *38*, 1816–1825.

Karacapilidis, N. I., and Pappis, C. P. (1996). Production planning and control in textile industry: A case study. *Computers in Industry*, *30*, 127–144.

Krarup, J., and Bilde, O. (1977). Optimierung bei graphenthe-oretischen und ganzzahligen probleme. chapter Plant location, set covering and economic lot sizes: an O(mn) algorithm for structured problems. (pp. 155–180). Birkhauser Verlag, Basel.

Maniezzo, V., Stützle, T., and Voß, S. (2010). *Matheuristics - Hybridizing Metaheuristics and Mathematical Programming*. Springer.

Meyr, H. (2002). Simultaneous lotsizing and scheduling on parallel machines. *European Journal of Operational Research*, *139*, 277 – 292.

Muller, L. F., Spoorendonk, S., and Pisinger, D. (2012). A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times. *European Journal of Operational Research*, *218*, 614–623.

Pimentel, C., Alvelos, F. P., Duarte, A., and de Carvalho, J. M. V. (2011). Exact and heuristic approaches for lot splitting and scheduling on identical parallel machines. *International Journal Manufacturing Technology and Management*, *22*, 39–57.

Puchinger, J., and Raidl, G. R. (2005). Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach* (pp. 113–124). Springer Berlin / Heidelberg volume 3562 of *Lecture Notes in Computer Science*.

Sahling, F., Buschkhl, L., Tempelmeier, H., and Helber, S. (2009). Solving a multi-level capacitated lot sizing problem with multi-period setup carry-over via a fix-and-optimize heuristic. *Computers & Operations Research*, *36*, 2546–2553.

Santos, M. O., and Almada-Lobo, B. (2012). Integrated pulp and paper mill planning and scheduling. *Computers & Industrial Engineering*, *63*, 1–12.

Serafini, P., and Speranza, M. G. (1992). Production scheduling problems in a textile industry. *European Journal of Operational Research*, *58*, 173–190.

Silva, C., and Magalhaes, J. M. (2006). Heuristic lot size scheduling on unrelated parallel machines with applications in the textile industry. *Computers & Industrial Engineering*, *50*, 76–89.

Sural, H., Denizel, M., and Wassenhove, L. (2009). Lagrangean based heuristics for lotsizing with setup times. *European Journal of Operational Research*, *194*, 51–63.

Trigeiro, W. W., Thomas, L. J., and McClain, J. O. (1989). Capacited lot sizing with setup times. *Management Science*, *35*, 353–366.

Zhu, X., and Wilhelm, W. E. (2006). Scheduling and lot sizing with sequence-dependent setup: A literature review. *IIE Transactions*, *38*, 987–1007.