Journal of Industrial Engineering and Management

doi:10.3926/jiem.2009.v2n1.p60-89 ©© JIEM, 2009 – 2(1): 60-89 – ISSN: 2013-0953

A hybrid flow shop model for an ice cream production scheduling problem

Imma Ribas Vila, Ramon Companys Pascual

Universitat Politècnica de Catalunya (SPAIN)

imma.ribas@upc.edu; ramon.companys@upc.edu

Received January 2008
Accepted March 2009

Abstract: In this paper we address the scheduling problem that comes from an ice cream manufacturing company. This production system can be modelled as a three stage nowait hybrid flow shop with batch dependent setup costs. To contribute reducing the gap between theory and practice we have considered the real constraints and the criteria used by planners. The problem considered has been formulated as a mixed integer programming. Further, two competitive heuristic procedures have been developed and one of them will be proposed to schedule in the ice cream factory.

Keywords: scheduling, hybrid flow shop, mathematical programming, heuristics

1 Introduction

The first research papers about hybrid flow shop appear in the 70's. Salvador (1973) was one of the pioneer papers published on hybrid flow shop with more than two stages. The main motivation for this article was to obtain a programming procedure in a nylon polymerization factory. Although some authors, from this moment on, were concerned with the study of such systems, it was at the end of 80's when hybrid flow shop systems began to have a real interest to researchers. This interest is caused by the increasing use of this configuration in our industry due to its flexibility. Even so, most of the published papers consider the programming problem in this environment from a theoretical point of view, and very few deal with real cases. According to the state of the art from Vignier, Billaut, and Proust (1999), only Narastmhan and Panwalkar (1984), Proust and Grunenberguer (1995), Paul (1979) and Sherali, Sarin and Kodialam (1990) are concerned on industrial applications. Subsequent to the publication to this state of the art, Wong, Chan and Ip (2001) propose a genetic algorithm to schedule spreading cutting and sewing operations in an apparel manufacture. Göthe-Lundgren, Lundgren and Persson (2002) solve the programming problem in an oil refinery company using mixed integer programming. Jin, Ohno, Ito and Elmaghraby (2002) develop a genetic algorithm to schedule orders in a printed circuit board assembly line. Lin and Liao (2003) propose a heuristic procedure to schedule one day’s orders in a label stickers manufacturing company to minimize the weighted maximal tardiness. Bertel and Billaut (2004) treat the processing checks system as a three-stage hybrid flow shop with recirculation and propose a heuristic procedure to minimize the weighted number of tardy jobs. Lee, Kim and Choi (2004) analyze the production scheduling problem in a leadframes manufacturing plant. The authors propose a bottleneck-focused heuristic procedure to minimize total tardiness of a given set of jobs. Ruiz and Maroto (2006) studied the scheduling problem in a ceramic tiles manufacturing and developed a genetic algorithm that performs very competitively. Ruiz, Serifoglu and Urlings (2008) trying to get closer to the real flow shop scheduling environment, investigated the effect of including realistic considerations, characteristic and constraints, on problem difficulty.

Conscious that an important gap between theory and practice still exists, we visited different types of factories in our surroundings to identify what productive systems can be formulated as hybrid flow shop and to detect, not only the most important constraints that have effects on the scheduling problem but also the criteria used by the planners. It has been possible to verify that different types of manufacturing systems, very different to each other, can be formulated as hybrid flow shop to develop efficient scheduling procedures. Between them, we included the manufacturing system on a labels factory, on an acrylic sheets factory, on a cocoa powder form factory, on an active pharmaceutical ingredients (API) factory, on a cold cuts factory or on an ice cream factory. Some special constraints have been detected on each manufacturing system (Ribas, 2007), but also some constraints that are common to all of them, in particular the effect of setup times. In this paper we have considered the characteristics found in the ice cream factory.

The rest of the paper is organized as follows: Section 2 analyzes the ice cream production system; Section 3 develops a mathematical model using mixed integer programming (MIP). Section 4 proposes a heuristic procedure, Section 5 shows the results obtained in the computational experience and Section 6 concludes.

2 Production process in the ice cream factory

The analyzed ice cream production system (Figure 1) consists of three stages: in each stage there is a set of eligible machines for each lot, operating in parallel. In the first stage there are two mixers used to mix the raw materials according to recipe. The mix is sent to one of the three tanks with 10.000 litres of capacity, where it is pasteurized, homogenized and where the colouring and flavourings are added. The product must remain in the tank for a minimum of four hours and a maximum of forty eight. Then, the mix goes to a cooler which cools and gives consistency to the ice cream. There are 32 coolers of three types. Each product can be processed in any given type. The product that comes from the cooler arrives to a filling machine which shapes the ice cream. Next, it goes through a freezing tunnel and finally to the packing machine. There are three types of freezing tunnels: of ammonia, nitrogen and brine. The lines which use the ammonia and brine tunnels are fixed but the lines which use the nitrogen tunnels are mobile and in occasions, two filling machines can feed the same tunnel of freezing. One filling machine can receive products from more than one mixer because every mixer produces only one flavour. According to the product to produce, the filling machine can require installing an additional tool.

The considered process begins when the mix goes to one of the maturation tanks. Next, it goes to the cooler where is pumped to one of the filling machines. According to this schema, this is a no wait system because the product which comes from one stage goes directly to the next stage without waiting in an intermediate buffer. Additionally, there are setup costs depending on the sequence of the family of products to produce due to the installation of lines or to the cleaning works. Therefore, the scheduling problem in the ice cream factory can be modelled as a nowait hybrid flow shop with batch dependent setup costs.

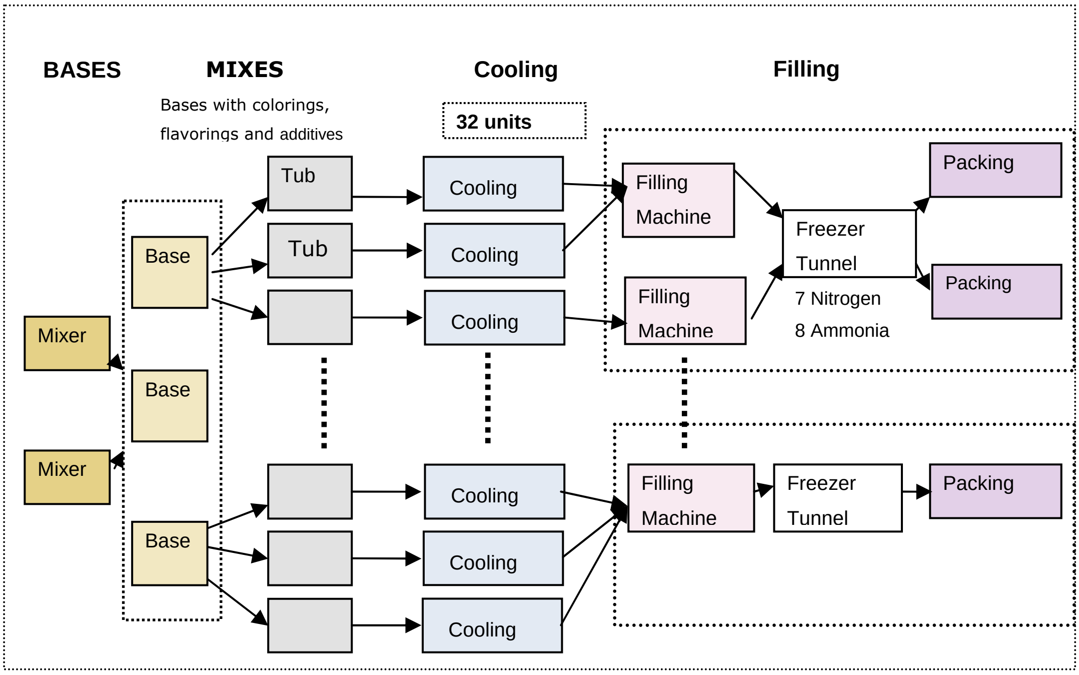

Figure 1. "Work-flow of the production system". Source: Authors

The elaboration of the scheduling must be done according to the system limitations. One of the most important limitations is the manpower. The manufacture of each product requires a certain number of persons. The difference between the number required by each one is considerable. In addition, due to be a very seasonal product one must take into account the opportunity cost of advancing production (stock) with regard to the backorder cost due to a punctual load or unexpected events (absenteeism, strikes...). Another critical limitation is the reduced space in the productive plant. The occupied space by different machines and the experimented growth in the latter years has converted the physical space in a real constraint that has to be considered. The flexibility that permits to install and uninstall the lines to produce different products without increasing the space in the productive plant has to be managed carefully, because the first productive hours in a new installed line causes important opportunity costs due to readjustments in the line that provokes defective products and a decreased production speed. In addition, related to this limitation, the scheduler has to consider the incompatibilities between machines because some of them occupy more space than others and the installation of one machine can lead to the impossibility of installing others. Another consideration during the scheduling is that each product has a set of eligible machines in each stage and that the product can require more than one machine in some stage. Also, we must take into consideration the number of available tools that some products require. These tools are used to shape or add condiments to the final product. They are very sophisticated and expensive. Another factor is the availability of coolers because they must be connected to the filling machines. For this reason, only certain coolers can be connected to the filling machine assigned. The machines require a setup time when there is a change in the type of product to produce. In particular, this factor has leaded the managers to avoid certain sequences of families because the setup cost associated is prohibitive. Another interesting factor to consider is that the products can either be delivered to the customers or can be stored according to the market where they will be sold. Products for the national market are produced make-to-stocks (MTS) according to a forecasting; on the other hand, products for the international market are produced make-to-order (MTO). To model this change in the production policy we have used an index that the planner can fix according to the due date and the inventory policy of the product (MTS or MTO). For a similar due date, one MTO product will have bigger priority index than a MTS product, and for the same policy, the bigger index is for the product with a critical due date.

In the real environments, the production scheduling considers, implicitly or explicitly, more than one simple criterion of efficiency. The most habitual criteria are:

- The punctuality in the deliveries. In some companies this criteria is measured using the sum of delays or with the sum of weighted delays, if there are some delays more critical than others.

- The inventory level. To advance the production can lead to high inventory level and can be as critical as to delay the deliveries. In this case it is necessary to penalize the earliness of deliveries according to due dates.

- The level of work in process that can be measured using the mean flowtime.

- The throughput and resource utilization that can be maximized minimizing the maximum completion time (i.e. makespan).

The planners in the ice cream factory use two criteria to schedule. One of them is to give a good service to the customer, that means to fulfil orders on time, and the other considers economic factors. Taking these two criteria into account can be valid, also, in other industrial environments. In our system, the economic factors considers three cost: the cost to install and uninstall machines in the plant, that correspond to the unproductive cost during the first hours due to adjustments in the line installed, the setup cost that corresponds to the required setup time to pass from one family to another family of product and the associated cost to the work in process.

3 Model formulation

The production scheduling can be formulated using mixed integer programming but its application to the industrial environments is reduced due to the huge number of variables required. Even so, Göthe-Lundgren et al. (2002) use the mixed integer programming to solve the scheduling problem in an oil refinery company. Some authors use the mathematical formulation only to focus the considered problem as in Guinet and Solomon (1996) that studied the scheduling problem in a hybrid flow shop to minimize the maximum tardiness. Bertel and Billaut (2004) consider a hybrid flow shop to minimize the weighted number of tardy jobs, Guirchoun, Martineau, and Billaut (2005) focused on the scheduling problem in a hybrid flow shop no wait with two stages, and Garcia and Lozano (2005) considered the scheduling problem in a hybrid flow shop with two stages with time windows to maximize the value of orders served.

In this paper, the scheduling problem associated to the ice cream factory is formulated which has been intended to solve by using the commercial software OPL Studio 3.7. We have made a time-discretization of the planning period as in Göthe-Lundgren et al. (2002) and Maravelias and Grossmann (2003), staying the planning horizon divided in periods of same length. The time-discretization model is well adapted to the ice cream production process because the process time to each lot is a multiple of shift. Therefore, the planning horizon has been divided in shifts and the production has to be scheduled for each one of them. We have considered, as it is done in the factory, that the lot is the production unit. The production of each lot can require one or two shifts. Following the schema shown in Figure 1, we consider that in the tank of maturation there is the required quantity of product to produce one lot. The product, which goes out from the maturation tank, goes through the cooler during 10 minutes and is pumped up to the filling machine. We suppose that all phases begin in time t, when product is in the tank and machines are available, and finish in $t+p_{i,}$ being p_i the processing time of lot i.

The problem has been formulated with a general notation in order to be able to be used in similar systems. We denote as *n*, the number of lots to schedule, K is the number of stages, M is the set of machines in the plant and M(*k*) is the set of machines in stage *k*. H+1 is the number of different resources, the index 0 is reserved to the manpower, T is the number of time points (shifts) to schedule. Each lot *i* has associated a processing time p_i which represent the number of time points required to be manufactured, a due date d_i and a priority index IP_i. We denote as G_h to the available quantity of resource *h*, P to the maximum number of filling machines that can be installed at the same time, B is the set of possible pairs of products and machines. $\overline{J}$ is the set of pairs of incompatible machines. We denote as $g_{i,h}$ as the required quantity of the resource type *h* to produce lot *i*, and $\varphi_{i,k}$ is the number of required machines to produce lot *i* en stage *k*. $Ch_{i,j,m}$ is the changeover cost when lot *i* is followed by lot *j* in machine *m*. These costs are due to, basically, the required cleaning and, in certain occasions, can be prohibitive. For example when in a machine that has been producing cream products proceeds to produce water products: The cleaning work must be exhaustive and is only justified in a night shift o during weekend (unproductive time points).

Initially the associated decisions to the problem (the assignment of lots to machines and the scheduling of lots in the machines), were considered in the same model but the impossibility to solve it advised to divide the model in two parts as is proposed in Harjunkoski and Grossmann (2002). The authors divide the global problem in an assignment problem and a scheduling problem. The first model is used to obtain a feasible solution according to the general limitations of the problem, and the second model tries to improve it. We have used a similar schema. The assignment model, which has the objective of maximizing the utility of the lots to manufacture, defined by the priority index IP_i, decides which lots have to be manufactured and assigns them to a certain time point and to one machine in each stage. In this first model, even though the decisions variables consider the time point in which the lots are manufactured, because it is necessary to guaranty the feasibility of the solutions, this decision is not transferred to the scheduling model. The scheduling model, therefore, receives the lots to manufacture and the associated machines to them and, by his objective function and the constraints of the problem, decides the time point (shift) in which each lot will be manufactured and the sequence of lots in the machines.

3.1 Assignment model

The binary variables used in this model have value 1 according to the following definition: u_i if the lot *i* is manufactured inside the scheduling horizon, $x_{i,t}$ if lot *i* is manufactured in time point *t*, $v_{j,t}$ if machine *j* is used in time point *t*, $y_{i,j,t}$ if machine *j* is used to produce the lot *i* in time point *t*.

The mixed integer programming associated to the assignment model is formulated as follows:

$$MAX\left[\sum_{i=1}^{n} IP_i \cdot u_i\right] \qquad (1)$$

$$\sum_{m\in M(K)} v_{m,t} \le P \qquad t=1,2,..,T \qquad (2)$$

$$\sum_{t=1}^{T} x_{i,t} \le p_i \cdot u_i \qquad i=1,2,3,...,n \qquad (3)$$

$$\sum_{i=1}^{n} g_{i,h} \cdot x_{i,t} \le G_h \qquad h=0,1,2,...,H \quad t=1,2,...,T \qquad (4)$$

$$\sum_{m\in M(k)} y_{i,m,t} = x_{i,t} \cdot \phi_{i,k} \qquad i=1,2,...,n \quad t=1,2,....,T \quad k=1,2,...,K \qquad (5)$$

$$\sum_{i=1}^{n} y_{i,m,t} = v_{m,t} \qquad m=1,2,...,M \quad t=1,2,...,T$$

(6)

$$y_{i,m,t} = 0 \qquad i=1,2,...,n \quad t=1,2,...,T \quad \forall(i,m)\notin B \qquad (7)$$

$$v_{m,t} + v_{m',t} \le 1 \qquad t=1,2,...,T \quad \forall(m,m')\in \overline{J} \qquad (8)$$

$$\sum_{i=1}^{n}\sum_{t=1}^{T} y_{m,i,t} \le T \qquad \forall m\in M \qquad (9)$$

The objective function (1) maximizes a global priority function. The IP$_{i,}$ is associated with due dates but also permits to the planner to give priority to lots based on other criteria to achieve the needs of each moment. Constraints (2) guarantee that only P lots can be manufactured in the same time point because the available space in the plant only allow installing P filling machines. Constraints (3) oblige to assign the required number of time points to each lot to be manufactured. Constraints (4) take control over the availability of tools. Constraints (5) oblige to assign, in each stage, the number of machines that the product requires. Constraints (6) indicate that a machine can only process one lot at a time, (7) avoid to assign lots to incompatible machine, (8) controls the incompatibility between machines. In constraints (9) the capacity of machines is controlled.

3.2 Scheduling model

The variables used in the scheduling model are the following: $x_{i,j,t}$ has value 1 if the lot *i* is the immediate predecessor of the lot *j* in the time point *t*. Due to the difference between processing time of the lots, it has been defined as follows: $Ws_{i,t}$ has value 1 if lot *i* starts to be manufactured at time point *t*, $Wp_{i,t}$ has value 1 if the lot *i* is being processed at time point *t* and $Wf_{i,t}$ has value 1 if lot *i* finishes at beginning of time point *t*. In the same way, variables have been defined to indicate the use of machines in each period: $Zs_{m,t}$ has value 1 if machine *m* is assigned to start in the time point *t*, $Zp_{m,t}$ has value 1 if machine is being used in time point *t,* $Zf_{m,t}$ has value 1 if machine *m* finishes the assigned lot at beginning of time period *t*. To penalize the installing and uninstalling of machines $C_{m,t}$ has value 1 if machine *m* has to be installed at time point *t*. At the beginning, the installations of machines are not computed because we have supposed that machines have been installed before. Finally, Ts_i is the start time point of lot *i*, Tf_i is the finish time point of lot *i* and T_i is its tardiness.

The scheduling model uses the solution obtained by the assignment model which indicates the lots to be manufactured (u_i), the machines in which they have to be processed ($y_{m.i}$) and it decides in which time point each lot has to be manufactured to satisfy the objectives. Let S the set of lots to produce and $\mu(i)$ the set of machines associate to lot *i* obtained by the assignment model. Therefore, we define as $\pi(m)$ the lots associate to machine *m*.

$$[MIN]Z = \alpha \cdot \left[\beta \cdot \sum_{m=1}^{M} \sum_{t=1}^{T} C_{m,t} + (1-\beta) \cdot \sum_{t=1}^{T} \sum_{m=1}^{M} \sum_{i \in \pi(m)} \sum_{j \in \pi(m)} (Ch_{i,j,m} \cdot x_{i,j,t}) \right] + \\ + (1-\alpha) \cdot \left[\gamma \cdot \sum_{i \in S} T_i + (1-\gamma) \cdot \sum_{i \in S} Tf_i \right] \qquad (10)$$

$$\sum_{i \in \pi(m)} Ws_{i,t} \leq 1 \qquad \forall m, \quad t = 1,2,...T \qquad (11)$$

$$\sum_{i\in\pi(m)} Wf_{i,t} \le 1 \qquad \forall m, \quad t = 2,...T+1 \qquad (12)$$

$$u_i = \sum_{t=1}^{T} Ws_{i,t}, \quad u_i = \sum_{t=2}^{T+1} Wf_{i,t} \qquad \forall i \in S \qquad (13)$$

$$\sum_{t=1}^{T} Ws_{i,t} - \sum_{t=2}^{T+1} Wf_{i,t} = 0 \qquad \forall i \in S \qquad (14)$$

$$\sum_{\forall i\in S} \sum_{t=1}^{t'} (Ws_{i,t} - Wf_{i,t}) \le 1 \qquad t' = 1,2,...,T+1 \qquad (15)$$

$$Wp_{i,t} = \sum_{t'=1}^{t-1} Ws_{i,t'} - \sum_{t'=1}^{t} Wf_{i,t'} \qquad \forall i \in S \quad t = 2,3,...,T \qquad (16)$$

$$\sum_{t=2}^{T+1} t \cdot Wf_{i,t} - \sum_{t'=1}^{T} t' \cdot Ws_{i,t} = p_i \qquad \forall i \in S \qquad (17)$$

$$Wf_{i,1} = 0 \qquad \forall i \in S \qquad (18)$$

$$Ws_{i,T+1} = 0 \qquad \forall i \in S \qquad (19)$$

$$Wp_{i,T+1} = 0 \qquad \forall i \in S \qquad (20)$$

$$Zs_{m,t} = \sum_{i\in\pi(m)} Ws_{i,t} \qquad t = 1,2,...T \qquad (21)$$

$$Zf_{m,t} = \sum_{i\in\pi(m)} Wf_{i,t} \qquad t = 2,...T+1 \qquad (22)$$

$$Zp_{m,t} = \sum_{t'=1}^{t-1} Zs_{m,t'} - \sum_{t'=2}^{t} Zf_{m,t'} \qquad m = 1,2,...M \quad t = 2,...,T+1 \qquad (23)$$

$$x_{i,j,t} \ge Wf_{i,t} + Ws_{j,t} - 1 \qquad t = 1,2,...T+1 \quad \forall i,j \in S \qquad (24)$$

$$C_{m,t} + Zf_{m,t} \ge Zs_{m,t} \qquad t = 2,3,...T \quad \forall m \in M \qquad (25)$$

$$\sum_{m\in M(K)} Zs_{m,t} + \sum_{m\in M(K)} Zp_{m,t} \le P \qquad t = 1,2,...,T \qquad (26)$$

$$\sum_{\forall i\in S} Ws_{i,t} \cdot g_{i,h} + \sum_{\forall i\in S} Wp_{i,t} \cdot g_{i,h} \le G_h \qquad t = 1,2,...T \quad h = 0,1,...H \qquad (27)$$

$$Zs_{m,t} + Zp_{m,t} + Zs_{k,t} + Zp_{k,t} \le 1 \qquad (m,k) \in \overline{J} \quad t = 1,2,...,T \qquad (28)$$

$$Tf_i \le Ts_i + p_i + M \cdot (1 - Ws_{i,t}) \qquad t = 1,2,...,T+1 \quad \forall i \in S \qquad (29)$$

$$Tf_i \geq Ts_i + p_i - M \cdot (1 - Ws_{i,t}) \qquad t = 1,2,\ldots,T+1 \qquad \forall i \in S \tag{30}$$

$$Tf_i \leq Ts_i + p_i + M \cdot (1 - Wf_{i,t}) \qquad t = 1,2,\ldots,T+1 \qquad \forall i \in S \tag{31}$$

$$Tf_i \geq Ts_i + p_i - M \cdot (1 - Wf_{i,t}) \qquad t = 1,2,\ldots,T+1 \qquad \forall i \in S \tag{32}$$

$$Tf_i \leq T + 1 \qquad i = 1,2,\ldots,n \tag{33}$$

$$Ts_i \geq t \cdot Wf_{i,t} - p_i \qquad t = 1,2,\ldots,T+1 \qquad \forall i \in S \tag{34}$$

$$\begin{aligned} &T_i \geq Tf_i - d_i \qquad \forall i \in S \\ &T_i \geq 0 \end{aligned} \tag{35}$$

The objective function (10) is composed by four terms. First term penalizes to install and uninstall machines, favouring that lots which can be produced in the same machine are done in consecutive time points. Second term takes into account the changeover cost from one family of product to another. Third term penalize the tardiness and the fourth the work in process. Parameters α, β and ɣ can weigh the importance to each of these terms, up to a point, opposed. The first terms try to group lots of the same family, without taking into account the tardiness that this policy can produce over the rest of the lots. This effect is reduced with the later terms. Constraints (11), (12), (13) y (14) control the start and finish time point of lots. With constraints (16) the lots are forced to start at the beginning of the time point and to finish at the beginning of the following time point after its processing time. This model allow us to be close in availability of lots and, therefore, to its tardiness. Constraints (17) guaranty that a lot is processed during p_i time points. Constraints (18), (19) and (20) fix boundary conditions consistent with constraints (15). Constraints (21) and (22) match the start and finish time point of a lot with the use of a machine and prevent a machine to process more than one lot at a time. Constraints (23) ensure the continuation of the process of a lot on a machine during the time point *t*. Constraints (24) expresses a condition of immediately precedence. The following constraints are specific to the scheduling problem in the ice cream factory: constraints (26) indicate that only P filling machines can be installed at a time, constraints (27) controls the availability of tools in a time point, constraints (28) the incompatibility between filling machines installed at the same time point. Constraints (29)-(33) are necessary to calculate the finish time of a lot, constraints (34) to calculate the start time of a lot and constraints (35) to calculate the tardiness.

doi:10.3926/jiem.2009.v2n1.p60-89 ©© JIEM, 2009 – 2(1): 60-89 – ISSN: 2013-0953

Separating the assignment and scheduling decisions can lead to have a feasible solution for the assignment model that cannot lead to a feasible solution for the scheduling model. In our case, moreover, we have not included any constraints that oblige the assignment model to maintain the same machine if the lot has a processing time bigger than one period. We have observed that this case happens when the capacity of machines is saturated. To avoid finishing the program without finding a global feasible solution we have implemented an interchange program between the assignment model and the scheduling model. This program can modify the scheduling horizon considered in *Threshold* units ($T = T + Threshold$). The variable *Threshold,* at the beginning, has value 0 and its value is increased by one unit when the solution transferred to the scheduling model cannot lead to a feasible solution. This increment means to increase the scheduling horizon that will lead, in most cases, to a new assignment. We would be able to provoke a new solution using or modifying other factors as the priority index of lots, or obliging to manufacture some of the discarded lots. The proposed schema, Figure 2, continues until the solution is feasible in both models.

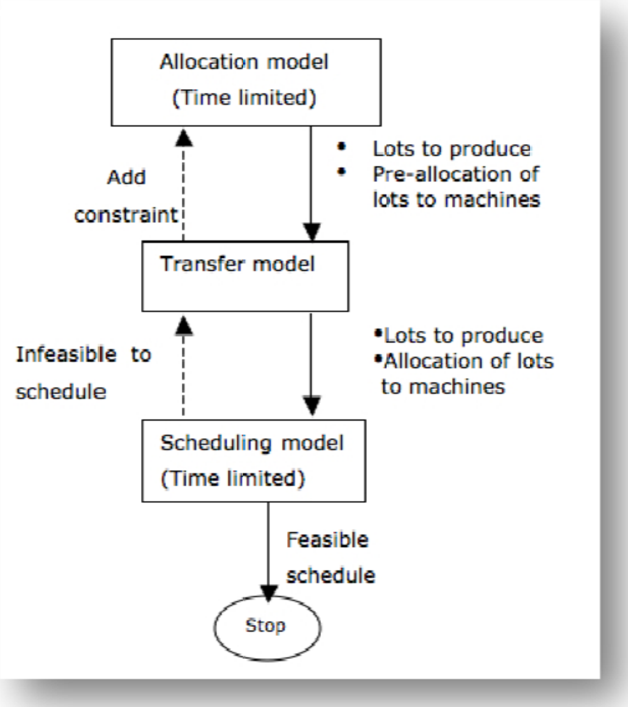

Figure 2. “Model proposed”. Source: Authors

Dividing the model and transferring solutions between models presupposes that the optimal solution found by the assignment model is transferred to the scheduling model and that it will be adapted according to the constraints of the scheduling model in order to obtain an optimal solution for the global problem. But as the difficulty of instances increase, the required computational time to obtain a solution becomes unviable. To be able to obtain a feasible solution a workflow with a maximum computational time has been defined in both models. In our implementation, the maximum computational time is fixed to ten minutes. This procedure allows to transfer to the scheduling model, if the optimal solution is not found before, the best solution obtained and the scheduling model tries to schedule the lots in the machines where they has to be produced. If the scheduling model obtains a solution before the maximum computational time fixed, the schedule found is the optimum for the assignment proposed.

4 Heuristic procedure

The model formulated previously allows solving only instances of little size, as it can be noticed in the computational experience in Section 5. In order to be able to solve real size instances, a heuristic procedure, formed by four tools, (Figure 3), has been implemented. The first tool (H1) transforms a sequential batch into a schedule, the second (H2) values the schedule, the third (H3) creates an initial solution and the fourth (H4), using the previous three tools, searches a solution, with better evaluation, by exploring the neighbourhood of the incumbent solution. The application of heuristics consists of applying, fifty times, the steps which are showed in Figure 3, retaining the best solution found. This way allows being able to explore neighbourhood that can have been discarded because of the ties.

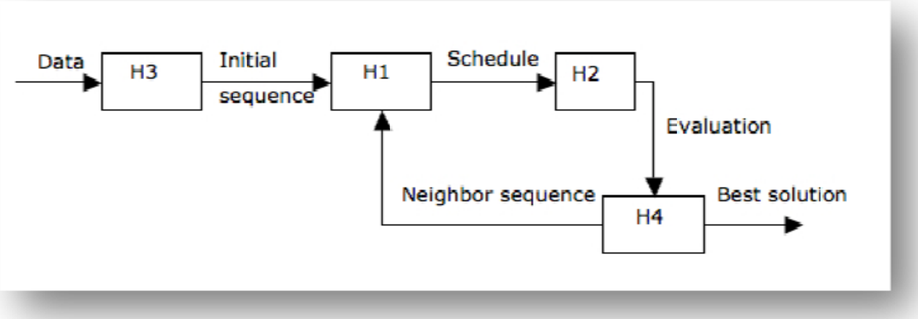

Figure 3. "Heuristic procedure". Source: Authors

These tools can be implemented in different way. In this paper, two variants of the initial solution procedure have been considered, which leads to two heuristics (Table 1).

Improvement procedure	Initial solution	
	S1	S2
SSA	GRASP	U

Table 1. "Implemented heuristics". Source: Authors

The first tool, named Basic Real Algorithm (BRA), is based on a dispatching procedure which considers the particularities of the ice cream factory. This procedure has been defined as follows: in time *t*, there are *x*-1 lots already scheduled and we have to schedule the lot that occupies position *x*. The available resource to process this lot is checked. If there is not enough available resources, the time is increased in one unit (*t*=*t*+1) until we find a possible time to process the lot or until the finish of schedule horizon (*t*=T). If it is not possible to process the lot inside the schedule horizon, it proceeds to try to schedule the next lot (*x*+1). The lots are scheduled from the later stage to the first one. Therefore, it begins from stage K and the lot *x* attempts to be assigned to one of the eligible machine for this lot that is available in *t*, and the compatibility with other machines installed in *t* is checked. Next, it checks if the machine was installed and, depending on the answer, a penalty is assigned to the machine:

- If the machine was already installed in t-1 and if there is no cost to change to this lot then the penalty is 0.
- If the machine was not installed in t-1 then the penalty is 1.
- If the machine was installed in t-1 but there is a cost to change to this product then the penalty is 2.

Finally, the product is assigned to the machine with a minor penalty. In case of ties the machine is randomly select. Ties are considered because the assigned machine in one stage can commit the assignment of the machines in previous stages due to the incompatibilities between machines. So, the assignments done for a product can change in every schedule, allowing obtaining different results every time. When the evaluation in stage K is finished and an assignable machine to *x* has been found, one proceeds to the schedule in the previous stage. The procedure from stage K-1 to the first stage is similar but not only the availability of machines in *t* must be checked, but also the compatibility with machines assigned in later stages. Also, in each stage, the number of machines that the product requires must be assigned. If, in some stage, the number of machines required cannot be assigned, then the schedule must begin again in *t*+1 and so on until the schedule can be completed or the final of the schedule horizon has been reached, in which case it has to proceed to the schedule of the next lot (*x*=*x*+1). When the lot has been scheduled, the availability of the assigned machines and tools and number of installed filling machines must be updated and it proceeds to schedule the next lot. When a lot cannot be scheduled his finish time is fixed after the limit of the scheduling horizon, in order to penalize the tardiness.

The second tool evaluates the obtained solution according to the number of lots to schedule and the expression (10).

4.1 Heuristic procedure S1

To obtain an initial solution the CR index is dynamically calculated for each lot as (36).

$$CR_{i,j} = \lambda \cdot d_i + (1-\lambda) \cdot \left(\max\{\tau_{1,j} + s_{j,h,g(i)}, r_{1,i}\} + \sum_{k=1}^{K} p_{k,i} + \sum_{k=2}^{K} s_{med(k)} \right) \qquad (36)$$

Parameter λ can take values from 0 to 1. In our test λ=0.5. Let d_i be the due date of lot i, $p_{k,i}$ the processing time of lot *i* in stage k, $s_{j,h,g(i)}$ the setup time required to change from family *h* to family g(i) in machine *j*, $\tau_{1,j}$ the available time of machine *j* in stage 1, $r_{1,i}$ the available time of lot *i* to be processed in first stage and $s_{med\ (k)}$ the mean setup time in each stage. λ value represents the commitment between two criteria: to prioritize lots with a critical due date or the lots that can finish before.

Next, the minimum value of CR is multiplied by $1+\eta$, $(0 \le \eta \le 1)$ and is taken as reference value. One of the lots is randomly chosen with an index, CR value, lower than this value. In our implementation, $\eta = 0.2$. The selected lot is tried to be scheduled. If this lot cannot be scheduled due to unavailability of resources or due to the incompatibility of machines already installed, the following lot, the next one in increasing order of CR, is tried to be scheduled. The scheduled lot is eliminated from the unscheduled vector, the availability of machines and resources used is updated and the indexes of lots are recalculated. The procedure continues until all lots are scheduled or until the final of the scheduling horizon is reached. The associated sequence to this schedule is formed by the lots according the order in which they have been scheduled. The sequence does not contain lots that have not been scheduled.

4.2 Heuristic procedure S2

Lots are sequenced in a non increasing order of its priority index IP$_i$. In case of ties, the lot with a minimum code is chosen. Then, the initial solution is obtained when the sequence is processed by BRA algorithm.

4.3 Improvement procedure SSA

The local search implemented is a variant of the non exhaustive descent algorithm (NEDA). The name chosen for this variant is Soft Simulated Annealing (SSA).

NEDA tries to improve the solution by swapping any two positions in the sequence. The interchange becomes all against all generating $n \cdot (n-1)/2$ neighbours. If during the process a new permutation improves the value of the objective function, it becomes the new current solution and the process continues until all the positions have been permuted without improvement. In this procedure the exploration of the neighbourhood is always made in the same order. The SSA algorithm uses an auxiliary vector, called revolver, which allows exploring the neighbourhood randomly. The revolver is a pointer vector whose components are initialized with the different positions that a job can have in the sequence. Next, the components are randomly mixed and used to codify the searching positions in the solution's neighbourhood. Given two pointers to positions *i, j* in the job sequence, their equivalent i_{rev} and j_{rev} are searched in the revolver vector *rev*, being $i_{rev} = rev(i)$ and $j_{rev} = rev(j)$. These new positions are used when the non-exhaustive descents search is applied. The evaluation of the candidate solutions is made using the BRA algorithm. The best solution is that which manufactures all lots in the initial sequence and obtains the minimal value in the objective function. In addition, during the procedure, solutions with the same value of the objective function and with same number of lots to manufacture (ties) are accepted as current solution with certain probability. When all the neighbourhood of the current solution have been explored without improving the solution, the process restarts again accepting ties with a certain probability, δ. The improvement phase finishes when the number of ties reaches a predefined number Γ or there is no change in the incumbent solution. If, after accepting ties, the solution improves, the accountant of ties is initialized and the process continues without accepting ties. In our implementation δ =0.5 and $\Gamma = 100$.

5 Computational Experience

The test has been done on instances generate according to the real characteristics. The instances generator requires introducing some factors which give the dimension to the instance. These factors are: number of lots, number of stages and number of machines in each stage, number of different tools and quantity of each one, number of shifts to schedule and maximum number of filling machines that can be installed in a shift. Further, other factors of random character are considered when data are calculated:

- Lot processing time: one shift with a probability of 90% and two shifts with a probability of 10%.

- Number of workers that one lot needs to be processed: that is a random variable between 1 and 4.

- Requirement of an additional tool. 50% of lots require it.

- Number of machines required by one lot in a certain stage: one with a probability of 85% and two with a probability of 15%. Number of machines required by one lot in a certain stage: one with a probability of 85% and two with a probability of 15%.

- Versatility of machines: One machine can process a certain lot with a probability of 90%

- Incompatibility between machines: two machines can be incompatible with a probability of 10%.

- Setup cost and setup time to change from processing one lot to another. For each pair of lots *i* and *j* a cost (time) of 10 the change from *i* to *j* with a probability of 15% is assigned.

- Due date of lots that corresponds to a random number of shifts between one and the maximum number of shifts.

- Index of priority of a certain lot that is calculated dividing the number of maximum shifts by its due date.

- In all tests done α =0.5, β=1 and γ=0.5.

5.1 MIP model Evaluation

Even though the model formulated has been divided in two stages in order to accelerate its resolution; both submodels have a great number of variables and constraints that limit its application on large size instances. In order to analyze the limits in its application the model has been applied on instances of different size. Table 2 shows the values of the factors used to generate instances in this test, in all of them the maximum number of stages is three and a maximum of two different tools have been considered.

No	Lots	Shifts	Stages			P	Resources	
			1ª	2ª	3ª		0	1
1	2	6	2	3	5	4	15	2
2	2	8	2	3	5	4	15	2
3	2	6	3	5		4	15	2
4	2	8	3	5		4	15	2
5	5	5	2	3		2	20	1
6	5	6	3	5		4	20	2
7	5	8	3	5		4	20	2
8	5	6	3	5	6	4	20	2
9	5	8	3	5	6	4	20	2
10	10	6	4	7		6	20	2
11	10	8	4	7		6	20	2
12	10	6	4	7	10	8	20	2
13	10	8	4	7	10	8	20	2

Table 2. "Considered factors to obtain the instances". Source: Authors

In Table 2 column *No* indicates the number of the instance, column *Lots* indicates the number of lots, column S*hift* indicates the number of considered shifts, column *Stages* is divided in three in each one there is the number of machines in the stage, column *P* indicates the maximum number of filling machines installed at the same shift and column R*esources* is divided in two and in each one there is the number of resources of this type. Only one instance of each size has been generated.

Table 3 shows the obtained results by the assignment model, for each instance, and Table 4 the results obtained by the scheduling model. In these tables column *No* indicates the number of the instance, column *Threshold* (only in table 3) indicates the increase done in the scheduling horizon of machines in order to be able to have an assignment that can be scheduled, column *Var.* indicates the number of variables, columns *Constr.* the number of constraints, column *Objective Value* indicates the value of the solution, the * after the number indicates an optimal solution (some of the other solutions can be also optimal but they have not been guaranteed by the algorithm) and *Time (s.)* indicates the CPU time, in seconds, to obtain the solution found.

No	Var.	Constr.	Objective Value	Time
1	194	342	24*	0,063
2	258	468	53*	0,094
3	158	304	20*	0,141
4	210	394	21*	0,001
5	180	240	103*	0,001
6	323	505	88*	0,016
7	429	653	115*	2,782
8	539	829	72	600,000
9	717	1115	91*	159,094
10	796	1203	212	600,000
11	1058	1589	159	600,000
12	1456	2251	30	600,000
13	1938	3031	170	600,000

Table 3. “Obtained results by the assignment model”. Source: Authors

No	Var.	Constr.	Threshold	Objective Value	Time
1	352	592	0	2,25*	0.28
2	456	722	0	1,50*	0.81
3	300	464	0	1,00*	0.20
4	384	610	0	1,00*	0.17
5	375	476	0	5,75*	0.28
6	519	811	0	6,50*	1.17
7	663	995	0	4,25*	4.97
8	687	817	0	4,00*	602.61
9	879	1385	0	6,00*	164.70
10	1248	1314	0	8,00	1200.00
11	1596	1286	0	3,75	1200.00
12	1528	1083	0	1,75	602.60
13	1956	1540	0	2,50	606.17

Table 4. “Obtained results by the scheduling model”. Source: Authors

It should be noted that times showed in Table 4 are the accumulative time because the run begins with the assignment model and the scheduling model is executed immediately following it.

Also note that, in both tables, the required time to obtain a solution, in general, grows with the number of lots to schedule and with the number of stages, although there are other factors which affect significantly the difficulty of the instance as can be the versatility of machines or the processing time of lots. If lots require more than one shift the instance in more difficult because there is less flexibility. Further, as it has been noted before, the assignment model does not avoid to change the assigned machine to one lot when is processed during more than one shift. That can occur when the capacity of the machines is saturated. To correct this undesirable situation in the industrial environment, the transfer model (Figure 2), chooses between one of the assigned machines. In particular, it chooses, in each stage, the machines with a minor code. This reassignment can lead to an unfeasible schedule (probably the assignment model would have allocated the same machine if it was able to). This is the reason for the *Threshold* variable which allows to increase the scheduling horizon of machines and to provoke a new assignment. In Figure 4 the solution obtained by the assignment model for the instance number 5 is shown. This solution indicates the lots to be manufactured (2, 3, 4, and 5) and the machines in which they must be processed (machines: (2, 3) for lot 2, (1, 5) for lot 3, (1, 2, 5) for lot 4 y (1, 5) for lot 5). This solution proposes to manufacture lot 4 in machine 2 during the first shift and in machine 1 during the second one.

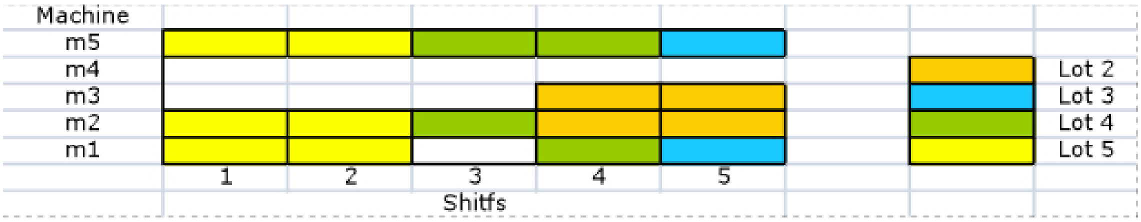

Figure 4. "Solution obtained by assignment model on instance number 5". Source: Authors

In this case there is not any limitation to process the lot in the same machine during the two shifts, and, consequently, when the transfer model proposes to process lot 4 in machine 1, there will not be problems and the *Threshold* variable will have value 0. If the division of the lot was because machine 1 was occupied during the three first shifts, the solution proposed by the transfer model would not be feasible. In consequence, variable *Threshold* would have value 1 that should provoke a new assignment for a scheduling horizon of 6 shifts.

Figure 5 shows the solution obtained, finally, by scheduling model for lots and machines proposed (previously corrected by transfer model) by assignment model. The objective value for this solution is 5.75.

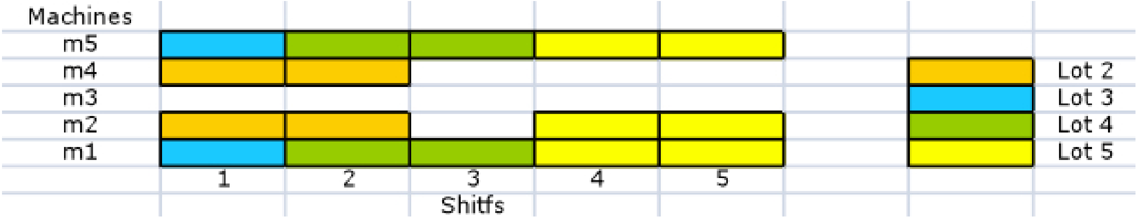

Figure 5. "Solution obtained by scheduling model for instance number 5". Source: Authors

In results shown in table 3, notice that the assignment model has obtained the optimal solution for instances with 2 lots and in four of five instances of 5 lots. On the other hand, one solution was obtained, for instances with 10 lots and in the fixed CPU time, but it cannot be guaranteed that the solution is optimal.

In table 4 one may note that the scheduling model has obtained the optimal solution in the same instances that assignment model did (instances 1, 2, 3, 4, 5, 6, 7 and 9). In instances 8, 12 and 13, scheduling model can obtain a solution in a

CPU time inferior to the maximum fixed. On the other hand, in instances 10 and 11, a feasible solution has been obtained for the assignments done in the maximum allowed time. Note that no instances have required the *Threshold* variable, which indicates that the solution found by assignment model, corrected if necessary by the transfer model, was already a feasible solution for the global problem.

Results showed in Table 3 and 4 lead to think that the assignment model limits the results obtained. However, we cannot guarantee that, if the assignment model obtains the optimal solution in all instances, then the scheduling model will do so also; it would be easier to evaluate the quality of the solution because, if the limitation comes from the decision of the amount lots to produce, the improvement margin is minor (efficiency in the use of machines and fulfil orders in a due dates). Notice, also, that even dividing the problem, the optimal solutions are reserved for instances of little size (according to the realized test it may say that up to 5 lots). This indicates that alternative procedures must be studied in order to be used in the industrial environment.

5.2 Evaluation of Heuristics

No	n	T	K	m1	m2	m3	m4	P	Resources	r0	r1	r2	r3	CPU time
14	10	8	2	5	7			7	3	5	2	3		0.50
15	10	8	2	5	7			6	2	20	2			0.56
16	10	8	2	5	7			6	4	25	2	2	2	0.46
17	10	8	3	3	5	6		6	3	25	2	3		0.27
18	10	8	3	3	5	6		6	2	20	2			0.24
19	10	8	3	3	5	6		5	4	25	2	2	2	0.20
20	10	8	4	4	5	5	6	5	3	25	2	3		0.21
21	10	8	4	4	5	5	6	5	2	20	2			0.21
22	10	8	4	4	5	5	6	5	4	25	2	2	2	0.22
23	10	10	4	4	5	5	6	6	3	25	2	3		0.23
24	10	10	4	4	5	5	6	6	2	20	2			0.19
25	10	10	4	4	5	5	6	6	4	25	2	2	2	0.29
26	15	10	2	5	7			7	3	25	2	3		0.56
27	15	10	2	5	7			7	2	20	2			0.70
28	15	10	2	5	7			6	4	25	2	2	2	1.21
29	15	10	3	3	5	6		6	3	25	2	3		0.66
30	15	10	3	3	5	6		5	2	20	2			0.59
31	15	10	3	3	5	6		5	4	25	2	2	2	0.63
32	15	10	4	4	5	5	6	5	3	25	2	3		0.80
33	15	10	4	4	5	5	6	5	2	20	2			0.85
34	15	10	4	4	5	5	6	5	4	25	2	2	2	0.71
35	15	16	4	4	5	5	6	5	3	25	2	3		0.67
36	15	16	4	4	5	5	6	6	2	20	2			0.89
37	15	16	4	4	5	5	6	6	4	25	2	2	2	0.72
38	20	20	2	5	7			6	3	25	2	3		2.73
39	20	20	2	5	7			6	2	20	2			2.09
40	20	20	2	5	7			6	4	25	2	2	2	1.36

41	20	20	3	3	5	6		6	3	25	2	3		2.05
42	20	20	3	3	5	6		5	2	20	2			1.93
43	20	20	3	3	5	6		5	4	25	2	2	2	1.33
44	20	20	4	4	5	5	6	6	3	25	2	3		1.65
45	20	20	4	4	5	5	6	6	2	20	2			1.39
46	20	20	4	4	5	5	6	6	4	25	2	2	2	1.97
47	20	20	4	4	5	5	6	5	3	25	2	3		1.42
48	20	20	4	4	5	5	6	5	2	20	2			1.84
49	20	20	4	4	5	5	6	5	4	25	2	2	2	2.35
50	25	20	2	5	7			6	3	25	2	3		1.98
51	25	20	2	5	7			6	2	20	2			2.03
52	25	20	2	5	7			7	4	25	2	2	2	2.38
53	25	20	3	3	5	6		5	3	25	2	3		3.48
54	25	20	3	3	5	6		5	2	20	2			2.95
55	25	20	3	3	5	6		6	4	25	2	2	2	2.82
56	25	20	4	4	5	5	6	6	3	25	2	3		3.67
57	25	20	4	4	5	5	6	6	2	20	2			3.53
58	25	20	4	4	5	5	6	6	4	25	2	2	2	3.60
59	25	20	4	4	5	5	6	5	3	25	2	3		3.52
60	25	20	4	4	5	5	6	5	2	20	2			3.01
61	25	20	4	4	5	5	6	5	4	25	2	2	2	3.23
62	25	20	4	4	4	3	5	4	2	20	2			2.88
63	25	20	4	4	4	3	5	5	4	25	2	2	2	3.20

Table 5. “Factors used to generate each instance”. Source: Authors

The second test has been done on 50 instances. In Table 5 the parameters used to define each instance and the used CPU time, in seconds, to solve the instance are shown.

In Table 5 column *No* indicates the instance number, columns *n* the number of lots to schedule, column T the number of shifts, column *K* the number of stages, *m1, m2, m3* and *m4* indicates the number of identical parallel machines in each stage. Column *P* indicates the maximum number of machines installed, at the same time, in the late stage. Column *Resources* indicates the maximum number of different resources and *r0*, *r1, r2, r3* the quantity of available resources of each type. Finally, column *CPU time* indicates the average time, in seconds, to obtain the solution in each instance. The time has been calculated as the average time used by heuristics to obtain a solution because, in a previous tests, we have observed that times used by each of the implemented heuristics are similar.

The comparison between procedures has been carried out by means of index $I_{h,heuristics}$ calculated as (37) where *h* indicates the instance number and *heuristics* the procedure used.

$$I_{h,heuristics} = (\mu_{h,\text{heuristics}} - v_{h,\min}) / v_{h,\min} *100 \qquad (37)$$

Being $\mu_{h,heuristic}$ the average of the results obtained for the instance number *h* by procedure *heuristics* and $v_{h,min}$ the minimum value obtained in the objective function for this instance, in any run or procedure. Three runs have been done for each instance and procedure. Table 6 shows the average (μ) and the standard deviation (σ) of $I_{h,heuristics}$. It is noted that both procedures have a similar behaviour with a little advantage of procedure U. We have also evaluated the dispersion of results by procedure in order to study the stability of each one. This evaluation has been done with index D_i calculated for each instance as (38) on the three runs done.

Heuristics	**μ**	**σ**
GRASP	2.23	2.12
U	1.95	2.19

Table 6. "μ and σ values of index $I_{h,heuristics}$ (en %)". Source: Authors

$$D_i = (v_{max,h} - v_{min,h}) / \mu_{h,i}*100 \qquad (38)$$

Where $v_{max,h}$ is the maximum value obtained by procedure *h* in the instance number *i*, $v_{min,h}$ is the minimum value obtained by procedure *h* in the instance number *i* and $\mu_{h,i}$ is the average value obtained in three runs. Table 7 shows the average and standard deviation of D_i by procedure for the 50 instances. Note that the behaviour of both procedures is very similar.

Heuristics	**μ**	**σ**
GRASP	2.82	2.71
U	2.75	2.86

Table 7. "μ and σ of D by procedure". Source: Authors

Finally, we have contrasted the results obtained by the mathematical model with those obtained by the heuristics. First, we have applied both heuristics on the instance number 5 which has been previously solved by the mathematical model (Figure 5). The main characteristics of each lot are shown in Table 8 where column *Lot* indicates the number of each lot, *Stage1* the number of machines necessary to process each lot in stage 1, *Stage2* the number of required machines by each lot in stage 2, p_i the processing time (in shifts), IP_i is the priority index of lot *i* and d_i is its due date.

Lot	Stage1	Stage 2	p_i	IP_i	d_i
1	1	1	2	8	5
2	1	1	2	13	3
3	1	1	1	40	1
4	1	1	2	10	4
5	2	1	2	40	1

Table 8. "Data of instance number 5". Source: Authors

Both heuristic procedures propose the same solution (Figure 6) whose objective value is 4.25. This solution does not coincide with the proposed by the mathematical model (Figure 5) whose objective value is 5.75. Observe that lots proposed by heuristics are others than proposed by the model.

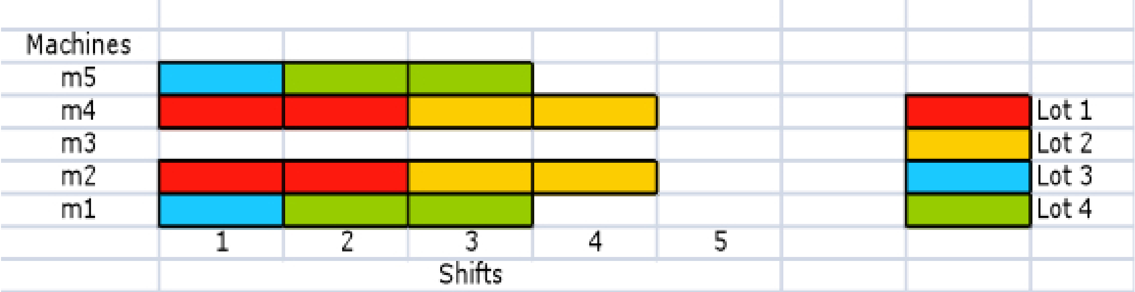

Figure 6. "Solution obtained by GRASP and U procedures". Source: Authors

In order to check the solution obtained by heuristics, we have imposed the scheduling model to produce lots number 2, 3, 4 and 5 in the same machines that were proposed by heuristics U and GRASP. The solution and the obtained value is the same that the heuristics. Then, why doesn't the mathematical model propose this solution if obtained value is better? The answer is that, in this case, the lots proposed to be manufactured do not maximize the global priority because this solution changes lot number 5, with the greatest priority, for lot number 1 which has less priority. In the mathematical model the final decision of the lots to manufacture and in which machines will be processed is given by the assignment model whose objective function maximize the global priority. On the other hand, in heuristics the objective function is the same than in the scheduling model which is used to face with all decisions (lots, machines and shifts). Observe that in Figure 6 the obtained solution by U and GRASP is more compact, the lots end before and it has installed the minimal number of machines, but, in contrast, one of the lots with highest priority is not manufactured. For the two procedures to be comparable it would be necessary to filter, according priority, for example, the lots that the heuristics must process or to incorporate in the objective function, explicitly, the priority of the lots. Following this reasoning, we have evaluated the results obtained by the heuristic procedures on instances of 10 lots that have been used in the first test (instances number 10, 11, 12 and 13), considering only those lots proposed by the assignment model. The comparison between the results obtained by the heuristics and by the mathematical model is shown in Table 9.

In this table column *No* indicates the number of the instance, *Lots* the number of lots to schedule according the assignment model. For each procedure (MIP, GRASP and U), column *Objective function* indicates the value of the proposed solution and column *CPU time* indicates the time, in seconds, required to solve the instance. Noted that the obtained value in the objective function is the same in instances 11 and 13, however, in instance number 10 the heuristic procedures find a better result than the MIP procedure and in instance 12 the MIP model is the best one. One must remark that the required time by heuristics is lower than the required by MIP model.

No	**Lots**	**MIP**		**GRASP**		**U**	
		Objective Function	**CPU Time**	**Objective Function**	**CPU Time**	**Objective Function**	**CPU Time**
10	8	8.00	1200,00	6.5	4.65	6.5	5
11	6	3.75	1200,00	3.75	3.84	3.75	4.42
12	3	1.75	602,60	2.00	0.39	2	0.31
13	4	2.50	606,17	2.50	2.50	2.5	2.48

Tabla 9. "Comparison between results obtained by the proposed procedures".
Source: Authors

Finally, the efficiency of the proposed heuristic procedures has been evaluated applying them on these four instances but considering the 10 lots. The results obtained are shown in Table 10 where column *No* indicates the number of the instance and, for each procedure, column *Lots* indicates the number of lots proposed to manufacture, column *Mo* the value of the objective function obtained as (10), and column *CPU* indicates the required time, in seconds, to solve the instance. These results lead to the conclusion that the heuristics are more efficient than the MIP model because they obtain a solution in less CPU time and, moreover, they propose to produce more lots, which is the main objective of the assignment model.

No	GRASP			U		
	Lots	Mo	CPU	Lots	Mo	CPU
10	10	9	11.98	10	9	10.01
11	10	8.25	7.14	10	8.25	6.9
12	10	11.5	13.75	10	11.5	11.57
13	10	11.75	25.83	10	11.75	26.15

Table 10. "Comparison between the results obtained by both heuristic procedures". Source: Authors

Since in the objective function (10) the factors are weighted by parameters α, β and γ, the company must adjust them in order to give the correct weight to each criterion and to be able to obtain solutions adjusted to the specific requirements of each moment.

6 Conclusions

The different industrial example shows that the problem faced is frequently found in our industry taking as prototype the ice cream factory. This manufacturing system can be modelled as a nowait hybrid flow shop with batch dependent setup costs. After an initial analysis the problem has been formulated using mixed integer programming. Due to the impossibility to solve, in a competitive time, a formulation which considered the decisions of lots to produce, assigned machines to each lot and time point to manufacture each lot, the global problem was divided in two. The first model decides the lots to be produced and the assigned machines to them. The second model schedules these lots in the assigned machines. Some constraints in the assignment model, that would guarantee the feasibility of the obtained solution in both models, have been omitted in order to accelerate its resolution. To communicate both models and to solve the possible situation produced by this omitted constraints, we have implemented a transfer model which receives a signal from the scheduling model when the solution obtained by the assignment model is not feasible and sends a signal to the assignment model in order to obtain a new solution. In our implementation the signal sent to the assignment problem provokes an increase in the scheduling horizon of machines which permits to generate a new solution. The results obtained show that the proposed model is only efficient in little size instances and for this reason we have developed heuristic procedures to use in the ice cream factory. We have implemented two procedures that have been contrasted with the MIP model proposed. This contrast has done, firstly, solving one instance of 5 lots with each one of the three proposed procedures, the mathematical model and the two heuristics. We have shown that if the obtained results between procedures are to be contrasted, then the heuristic procedures must only consider those lots proposed by the assignment model. Following this conclusion, the 10 lots instances have been solved by the heuristic procedures and the obtained results have been similar to those obtained by the scheduling model but in considerably less time. Therefore, both heuristic procedures are efficient to solve the scheduling in the ice cream factory. The late test done, in order to compare these two procedures, shows that the behaviour of both procedures is similar but we will propose to the managers the U procedure, properly adapted, because is easier and more intuitive.

References

Bertel, S., & Billaut, J. (2004). A genetic algorithm for an industrial multiprocessor flow shop scheduling problem with recirculation. *European Journal of Operational Research*, 159(3), 651-662.

Garcia, J. M., & Lozano, S. (2005). Production and delivery scheduling problem with time windows. *Computers & Industrial Engineering*, 48, 733-742.

Göthe-Lundgren, M., T. Lundgren, J., & A. Persson, J. (2002). An optimization model for refinery production scheduling. *International Journal of Production Economics*, 78(3), 255-270.

Guinet, A. G. P., & Solomon, M. M. (1996). Scheduling hybrid flowshops to minimize maximum tardiness or maximum completion time. *International Journal of Production Research*, 34(6), 1643-1654.

Guirchoun, S., Martineau, P., & Billaut, J. -. (2005). Total completion time minimization in a computer system with a server and two parallel processors. *Computers & Operations Research*, 32(3), 599-611.

Harjunkoski, I., & Grossmann, I. E. (2002). Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Computers & Chemical Engineering*, 26(11), 1533-1552.

Jin, Z. H., Ohno, K., Ito, T., & Elmaghraby, S. E. (2002). Scheduling hybrid flowshops in printed circuit board assembly lines. *Production and Operations Management*, 11(2), 216-230.

Lee, G. C., Kim, Y. D., & Choi, S. W. (2004). Bottleneck-focused scheduling for a hybrid flowshop. *International Journal of Production Research*, 42(1), 165-181.

Lin, H. T., & Liao, C. J. (2003). A case study in a two-stage hybrid flow shop with setup time and dedicated machines. *International Journal of Production Economics,* 86(2), 133-143.

Maravelias, C. T., & Grossmann, I. E. (2003). New general continuous-time State−Task network formulation for short-term scheduling of multipurpose batch plants. *Industrial & Engineering Chemistry Research*, 42(13), 3056-3074.

Narastmhan, S. L., & Panwalkar, S. S. (1984). Scheduling in a two-stage manufacturing process. *International Journal of Production Research*, 22(4), 555-564.

Paul, R. J. (1979). A production scheduling problem in the glass-container industry. *Operations Research*, 27(2), 290-302.

Proust, C., & Grunenberguer, E. (1995). Planification de production dans un contexte de flow-shop hybride à deux étages: Conception et interprogrammation d'ARIANNE 2000. *RAPA*, 8(5), 715-734.

Ribas, I. (2007). Programación multicriterio de un sistema productivo con flujo regular sin esperas y estaciones en paralelo. aplicación a una fábrica de helados. *Universistat Politècnica de Valencia*.

Ruiz, R., & Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research*, 169(3), 781-800.

Ruiz, R., Serifoglu, F. S., & Urlings, T. (2008). Modeling realistic hybrid flexible flowshop scheduling problems. *Computers & Operations Research*, 35(4), 1151-1175.

Salvador, M. S. (1973). A solution to a special class of flow shop scheduling problems. In S. E. Elmaghraby (Ed.), *Symposium on the theory of scheduling and its applications*. (pp. 83-91). Berlin: Springer.

Sherali, H. D., Sarin, S. C., & Kodialam, M. S. (1990). Models and algorithms for a two-stage production process. *Production Planning & Control*, 1(1), 27-39.

Vignier, A., Billaut, J. C., & Proust, C. (1999). Hybrid flowshop scheduling problems: State of the art. *Rairo-Recherche Operationnelle-Operations Research*, 33(2), 117-183.

Wong, W. K., Chan, C. K., & Ip, W. H. (2001). A hybrid flowshop scheduling model for apparel manufacture. *International Journal of Clothing Science and Technology*, 13(2), 115-131.