Flexible job-shop scheduling in a glass factory

R. Alvarez-Valdes [†], **A. Fuertes** [‡], **J.M. Tamarit** [†], **G. Giménez**[§] **and R. Ramos**[§]

[†]Dept. of Statistics and Operations Research. University of Valencia, Spain
[‡]Institute of Robotics. University of Valencia, Spain
[§]DATADEC. Valencia, Spain

Abstract

We describe the design and implementation of a scheduling system in a glass factory. The factory produces a large variety of manufactured glass objects in a complex process ranging from melting the glass in the furnaces and blowing it automatically or manually to decorating, assembling and packing it. The structure basically corresponds to a flexible job-shop scheduling problem with some special characteristics. On the one hand, dealing with hot liquid glass imposes no-wait constraints on some operations. On the other hand, skilled workers performing some manual tasks are modelled as special machines. The system can be used to establish delivery dates for new customer orders, taking into account current machine workloads, or to schedule a set of orders, trying to meet given customer due dates.

Keywords: scheduling, job shop, production, heuristics

1 Introduction

This paper describes the design and implementation of a scheduling system in a glass factory. The factory produces a large range of manufactured glass objects, more than 3000 different types of pieces, and works under a make-to-order policy. Whenever a new customer order arrives, the information system checks if the required products are in stock and, if is the case, the order is served from existing stock. Otherwise, the system basically creates two kinds of production orders for the shop. Some production orders correspond to finished goods, ready to be sent out to customers, and involve operations such as decorating, assembling, packing and palletization. Other production orders correspond to semi-elaborated products which involve operations in the furnaces, blowing, cooling and bulk packing. Lot sizing considerations determine the number of pieces of each intermediate product in these orders, the number of which is typically much larger than the quantity required by a customer. For instance, if an order demands 500 wine glasses of a given model, a production order would consist of decorating and packing these 500 glasses, and a previous production order would consist of blowing 10000 glasses, 500 of which would proceed to the other order and the remaining 9500 would be stored. So this scheme introduces precedence relationships between production orders.

The operations to be performed on the pieces can, and in fact should, overlap. If an order consists of decorating and packing 500 glasses, we do not have to wait until all the glasses have been decorated to start packing them. Sometimes this overlapping is automatic, that is, as soon as the first piece has been processed on one machine it goes directly to the next machine. On other occasions the overlapping is limited by structural constraints, such as the dimensions of the box to be packed or the capacity of the container used to move the pieces from one machine to the next. Overlapping can even happen between operations of different orders, linked by precedence constraints.

Some orders have an operation in which raw materials (basically recycled glass) are melted in one furnace to produce liquid glass and then another operation on a blowing machine in which the glass is shaped while still hot. These operations must be processed in a special type of no-wait mode, in which every time a glass bit comes from the furnace, it is immediately shaped in the blowing machine. Therefore, rather than talking of no-wait mode, a term usually applied to operations, we talk of *synchronizing* these operations, piece by piece. This synchronization is not restricted to a pair of operations, but extended to other operations in the same production order or in some of its successors. In fact, after being shaped, the pieces are placed on a conveyor belt where, while they cool down, they are transported at a constant speed to the packing area. Once there, the pieces must be packed before they arrive at the end of the belt. Hence, the packing operation must also be synchronized with the blowing operation with a fixed delay specified by the length of the belt and its speed.

The factory has different types of machines. Apart from machines that can be considered standard in terms of scheduling theory, there are several machines specific to this type of factory. The furnaces in which the glass is melted can serve several blowing machines simultaneously. In this sense, they are multi-task machines, working on several production orders at the same time. Packing is done manually by a team of workers. The number of packers varies throughout the day and they process several packing operations at a time, attending to the pieces coming from the different belts. Some of the blowing operations are carried out by automatic blowing machines and some others by teams of highly skilled workers. In the latter case, the operations can usually be performed by several different teams and the production manager gives each of these teams a priority in processing the operation according to the type of operation. Some machine flexibility is therefore introduced because the team to do these operations has to be assigned by the scheduling system.

Some machines are continuously available, but others are not. When an operation is assigned to a machine working only a daily shift and its processing time exceeds the shift length, the process stops when the shift ends and starts again at the beginning of the following shift. However, other interruptions, such as preemption, are not allowed. Other types of unavailability also have to be considered. On the one hand, each machine can have its own calendar, with non-working days, holidays or maintenance breaks. On the other hand, when scheduling a set of new orders the manager can decide not to modify the existing production plan and so the new orders must be scheduled in a way that respects the previous machine workload.

The company intends to use the scheduling system in two ways. The user can input a set of orders and ask the system to schedule them as soon as possible in order to propose a delivery date to the client. Another possibility is to set a due date for each order and ask the system to satisfy as much as possible as many of those due dates. In both situations, apart from the primary objective there are some secondary objectives to consider, such as minimizing the work in progress and using, whenever possible, machines with high priorities to process the operations.

2 The scheduling problem

The problem is basically a flexible job shop, a model that has received some attention in the scheduling literature [4],[5],[6]. However, our problem has some special characteristics which have to be modelled adequately. The elements of the problem are:

- *Jobs*
  - A set of jobs $J_1, ..., J_m$ has to be scheduled. Some of the jobs correspond to finished goods and others are components for the other jobs.
  - There are precedence constraints between jobs, specifying the order of production. When there is a precedence constraint between jobs J_u and J_v, it is because job J_v requires some units processed by job J_u. We denote by r_{uv} the proportion of job J_u required by job J_v.
  - Jobs without successors may have a due date, d_j, an early date, e_j, and a late date, l_j. If the due date cannot be met exactly, the job should be

finished in the interval (e_j, d_j). If that is not possible, it should finish in the interval (d_j, l_j). If that is not possible, it should finish before e_j, while the less desirable option is to finish it after l_j.

  - Jobs also have an urgency level, u_j, reflecting the importance of satisfying their due dates. This level can be different for each job, being similar to the weight assigned to jobs in scheduling theory, or it can be used to partition jobs into subsets defined by different urgency levels.

- *Machines*

  - The shop consists of a set of machines $M_1, ..., M_n$. The set includes teams of workers specialized in particular types of operations (blowing, packing, decorating,....).
  - Each machine has its own set of working shifts and its own calendar of available days. Besides that, each machine may have some unavailable intervals as a result of previous commitments.
  - Some machines can only process one job at a time, but others can process several jobs at the same time (*multi-task machines)*.
  - The number of workers doing packing operations varies throughout the day. For instance, 15 workers do the morning shift (06:00-14:00), 12 the afternoon shift (14:00-22:00) and 3 the night shift (22:00-06:00). This situation is modelled as a multi-task machine with varying speed, where s_{k,t_1,t_2} is the speed of machine k in the interval (t_1, t_2).

- *Operations*

  - Each job consists of one or more operations O_{ij}.
  - Each operation O_{ij} must be processed on one machine of the set μ_{ij}, where p_{ijk} is the processing time of operation O_{ij} on machine $M_k \in \mu_{ij}$. The set μ_{ij} is ordered according to user-defined priorities for using every machine for a given operation.
  - The operations of a job can overlap. The extent of this overlapping is limited by a coefficient, ov_{ij}, defined as the proportion of operation O_{ij} that has to be processed before that its successor $O_{i,j+1}$ can start. This coefficient is 1 if no overlapping is allowed and can be very low, near to 0, if there is automatic overlapping, in which as soon as the first unit of operation O_{ij} is processed it becomes part of the next operation and is immediately processed.

    The overlapping, ov_{ij}, of two operations determines the earliest starting time of the second operation, so that it is guaranteed that a sufficient proportion of O_{ij} is already processed and also that $O_{i,j+1}$ does not finish before the last units coming from O_{ij} have arrived. In the left hand part of Figure 1 the first condition applies, but in the right hand part the first condition would produce too early a starting time for the second operation, depicted by the dotted arrow, and the second condition determines the starting time of $O_{i,j+1}$.

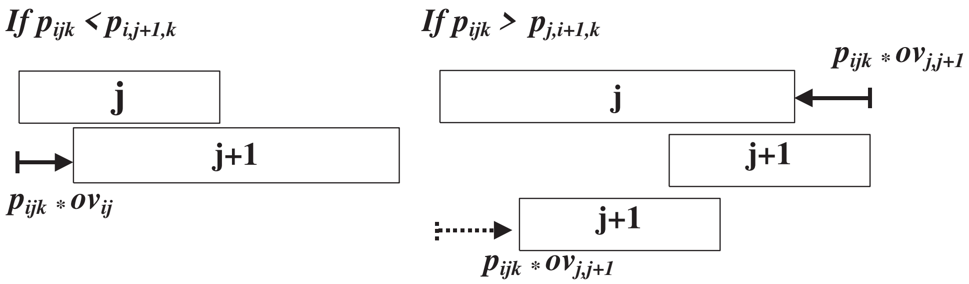

Figure 1: *Overlapping of operations*

– The last operation O_{uk} of a job J_u and the first operation O_{v1} of one of its successors, job J_v, can also overlap. On the one hand, job J_v only needs a proportion r_{uv} of its predecessor J_u and could start as soon as that proportion of O_{uk} is processed. On the other hand, operations O_{uk} and O_{v1} can overlap and the starting time of O_{v1} depends on r_{uv} and ov_{uk} (Figure 2).

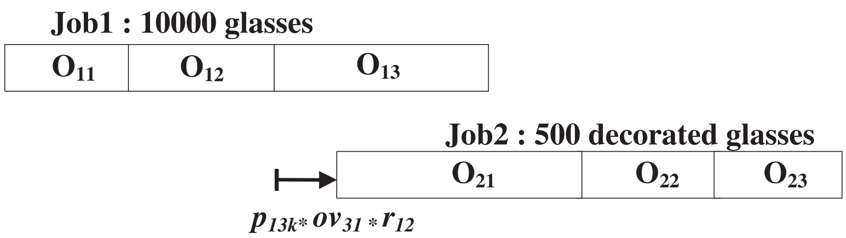

Figure 2: *Overlapping of jobs*

In fact, the starting time of O_{v1} also depends on other possible successors of job J_u. In Figure 3, $Job2$ and $Job3$ *compete* for the units processed in operation O_{13}. If $Job2$ gets its required proportion of $Job1$ first, and its first operation O_{21} starts according to r_{12} and ov_{13}, $Job3$ must wait and cannot start in the time indicated by the dotted arrow because its required proportion r_{13} will come from O_{13} after r_{12} of $Job2$.

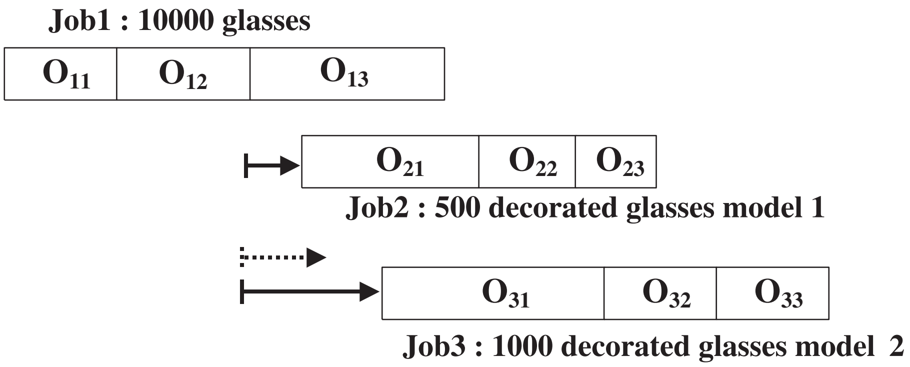

Figure 3: *Competition between jobs*

- *Synchronizing the operations of a job*
  - Some operations, like those handling hot liquid glass, must be processed in a no-wait mode. But the no-wait condition has to be applied piece by piece. In this sense, we talk of synchronizing the operations: the first unit processed in the first operation goes directly to the second operation and so on until the last piece is processed. Hence, the starting times of both operations must be almost simultaneous just as their finishing times must also be. If the processing times of these operations are not equal, we need to modify one of them, the shortest one, to make them of equal length. In Figure 4, a pair of operations has to be synchronized, $O_{i,j}$ involving a furnace M_k and $O_{i,j+1}$ involving a blowing machine M_l. In this case the first one is shorter, $p_{i,j,k} = 10$, and its length is stretched to match the second $p_{i,j+1,l} = 100$, as indicated by the arrows. As the furnace is a multi-task machine, if it has 100 minutes to process an operation which only takes 10, it uses just 10% of its capacity, and the remaining 90% is available to operations of other jobs requiring this furnace. The pair of operations constitutes a *block* and the block is considered as a single item to be scheduled in the planning process.

    Sometimes this synchronization involves more than two operations, as in Figure 5. In this example, the four operations of the job must be processed in this way, including a fixed waiting time before packing, which

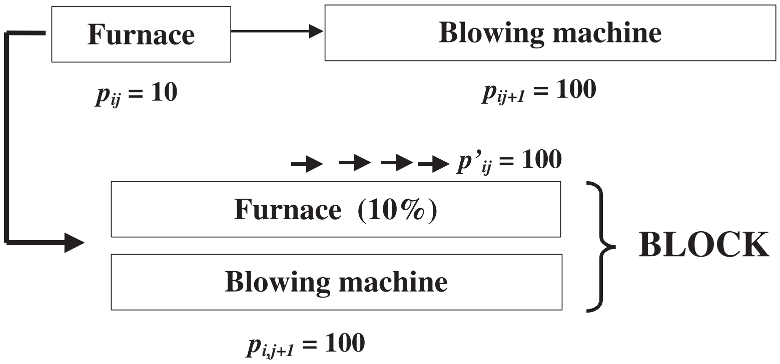

Figure 4: *Synchronising two operations*

is the time needed for the pieces to be transported by the conveyor belt.

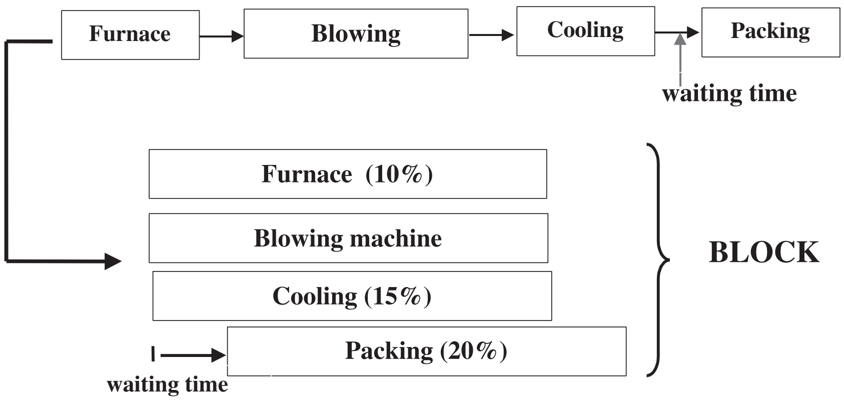

Figure 5: *Synchronising several operations of a job*

The synchronization must take into account the possibility of different working shifts for the machines involved. If a machine stops the process of an operation at a given time, the other operations must also stop at the corresponding times, as appears in Figure 6 in which we represent the availability of the machines involved in the block of Figure 5. The non-working periods of the blowing machine, the shaded areas, are passed to the other machines, also taking into account the waiting time for the last operation of the block.

- *Synchronizing operations of different jobs*

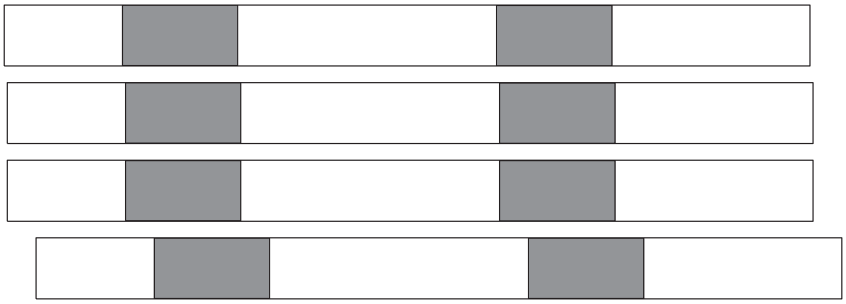

Figure 6: *Synchronising working shifts*

– In Figure 3 we showed the competition between jobs, that it is to say, the situation in which more than one successor is waiting for the units processed by the last operation of a job. There the arrows indicated minimum starting times for the operations. But sometimes these operations, though belonging to different jobs, must be synchronised. Therefore, the time lags depicted by the arrows are fixed and must be exactly satisfied. In this case, we have a *multiple block* which is also considered as a single item to be scheduled.

- *Objectives*

– The main objective is to minimise the total cost associated to the finishing times of the jobs corresponding to final products. If we denote by f_i the finishing time of job J_i, then we want to

$$Min \sum_i u_i c(f_i) \tag{1}$$

where

$$c(f_i) = \begin{cases} 0 & \text{if } f_i = d_i \\ k_1(d_i - f_i) & \text{if } e_i \leq f_i \leq d_i \\ k_1(d_i - e_i) + k_2(f_i - d_i) & \text{if } d_i \leq f_i \leq l_i \\ k_1(d_i - e_i) + k_2(l_i - d_i) + k_3(e_i - f_i) & \text{if } f_i \leq e_i \\ k_1(d_i - e_i) + k_2(l_i - d_i) + k_3 e_i + k_4(f_i - l_i) & \text{if } f_i \geq l_i \end{cases} \tag{2}$$

In this expression, k_1, k_2, k_3 are parameters whose non-decreasing values reflect the cost of the position of f_i with respect to d_i, e_i and l_i. This is a non-convex function to be minimised, as is shown in Figure 7.

– If the system is used to determine the earliest starting time of a job and no due date is given, the program assigns the initial time of the

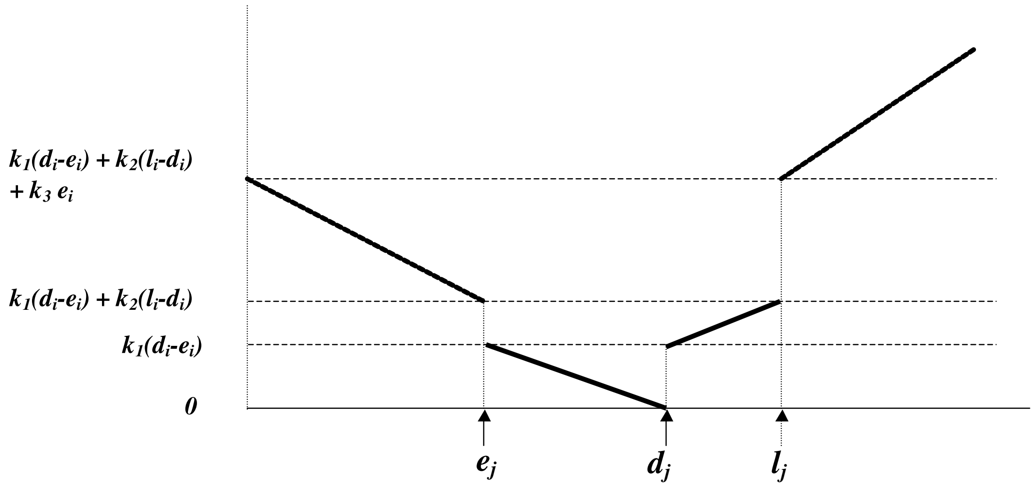

Figure 7: *Cost function of a job*

planning horizon as its due date. In that way, the job is always delayed and minimising the cost associated to the job is equivalent to scheduling it at the earliest opportunity.

- There are some secondary objectives to be taken into account. Among them, the most important is minimising the work in progress, that is, the intervals between consecutive operations of a job and between jobs linked by precedence constraints. Another objective is to use, whenever possible, machines with high priorities for processing the operations.

3 Solving the scheduling problem

We have developed a heuristic algorithm to produce good quality solutions in short computing times because on many occasions the system will be used on-line to establish delivery dates for new customer orders. In order to solve the problem, we first decompose it according to urgency levels and solve the subproblems in order on a decreasing level of urgency. The algorithm that solves the subproblems has two phases. First, it builds an initial feasible solution. Second, the solution is improved by trying to minimise the primary cost function and also by taking into account the secondary objectives.

3.1 Building an initial feasible solution

When building the initial solution, we try to process each operation as soon as possible, satisfying the precedence and resource constraints described above. We intend to develop a compact schedule, maximising machine utilization and minimising work in progress. If some jobs finish ahead of their due dates, it would be relatively easy to move them nearer to those due dates in a second phase.

The algorithm proceeds forward in time, starting at time 0 until a given planning horizon T is reached or all the jobs have been finished. It basically follows the parallel scheme used in resource-constrained project scheduling, but with two differences. First, an operation can start before its predecessors are finished, according to the allowed overlapping between them. Second, when an operation O_{ij} is a candidate for being put in process at a period t, it may be that more than one machine from its set μ_{ij} is available and different alternatives to process the operation can be considered.

During the description of the algorithms in this section, we will use the term *operation* in a wide sense, including simple operations as well as blocks and multiple blocks. When a block is considered, all its operations are taken into account in the checking for antecessors, successors and machine availabilities.

The algorithm can be summarised in the following steps:

- *Step 0: Initialization*
  - $t = 0$, where t is the current planning time.
  - $Q_k = \{O_{i1} \mid \text{Job } J_i \text{ has no predecessors and } M_k \in \mu_{i1}\}$
    Q_k is the queue of machine k, the set of operations that can be processed by that machine at that time.
- *Step 1: Assigning operations to machines at time t*
  - For each machine M_k, available at time t, with $Q_k \neq \emptyset$:
    * Build the list of candidates

      $$L_{kt} = \{O_{ij} \in Q_k \text{ that can start at time } t\}, \tag{3}$$

      This is the more complex step of the algorithm. For each operation, all of its characteristics, such as overlapping and competition between jobs, must be considered. For blocks, their operations are first synchronized and then the characteristics of each operation are considered.
    * Calculate the priority of each operation $O_{ij} \in L_{kt}$

- Assign operations to machines according to the priorities, while there is still some capacity left.
  * Delete them from the corresponding Q_k
  * Update the availability of machines M_k involved.

  The process of a block involves several machines. Therefore, when a block is put in process, the candidate lists of these machines must be updated before trying to process a new operation.

  If the starting time of an operation O_{ij} on a machine M_k is set to time t, its finishing time is calculated taking into account its processing time p_{ijk}, the working shifts of M_k and its speed s_{k,t_1,t_2}, where $t_1 = t$ and t_2 is the end of the process of O_{ij}.

- *Step 2: Termination test*

  If all the jobs are finished, stop.

  Otherwise, go on to Step 3.

- *Step 3: Updating Q_k*

  For each operation O_{ij} with all its predecessors sequenced

  - For each machine $M_k \in \mu_{ij}$, $Q_k = Q_k \bigcup \{O_{ij}\}$
  - Compute a lower bound LB_{ijk} on the minimum time in which O_{ij} could start in machine M_k, taking into account all the operation's characteristics and machine availability.

- *Step 4: Determining a new time t*

  For each machine M_k, let a_k=nearest time, beyond t, in which M_k will be available.

  Then, the new time $t = min_{1 \leq k \leq m}\{max\{a_k, min\{LB_{ijk}, O_{ij} \in Q_k\}\}\}$

  If $t > T$, stop. The planning horizon has been reached.

  Otherwise, go back to Step 1.

3.2 Priority rule

At Step 1, the set of candidates is ordered according to a priority criterion and put in process as long as the machine has enough capacity left. For monotask machines, that is reduced to putting in process the highest priority operation, but in multitask machines more than one operation can be processed simultaneously.

Several priority rules have been considered: MDD[2], MOD[3], ATC[7], $CR+SPT$[1]. An exploratory study has shown that MDD works very well, in fact slightly better than other alternatives, and it is being used in the current implementation. According to this rule, if at a given time t the job J_i is awaiting its $k-th$ operation and it requires n_i operations to be completed with processing times p_{ijk}, the modified due date for the job J_i is given by

$$d_i' = max\{d_i, t + \sum_{j=k}^{n_i} p_{ijk}\}$$

.

3.3 Improving the initial solution

The output of the first phase is a compact schedule in which operations are processed as soon as their predecessors and machine availability allow them to start. The due dates have so far been just an indirect reference used in the priority criterion.

In this second phase, a local search procedure will adjust the finishing times of the jobs to their due dates, also taking into account the secondary objectives. We define a simple move in which an operation, or a block, changes its starting time. Only moves preserving the feasibility of the solution are allowed. That implies that the positions of predecessors and successors severely limit the set of periods which an operation can be moved to.

When defining the search strategy we take into account two characteristics of the solution. First, it is usually much easier to move the operations of a job to the right than to the left, especially if they do not have successors. Second, the jobs whose due dates are considered in the primary objective function are precisely those without successors.

The search goes iteratively through two phases:

- *Phase I: Moving the jobs without successors, adjusting their due dates*

  Order the jobs by non-increasing due dates.

  For each job J_i, if $f_i < d_i$, try to move it to the right.

  - First, we try to make $f_i = d_i$
  - If that is not possible, we try to put f_i in the interval (e_i, d_i), as near as possible to d_i.
  - If that is not possible, we try to put f_i in the interval (d_i, l_i), as near as possible to d_i.
  - If that is not possible, we try to put f_i as near as possible to e_i.

If $f_i > d_i$, we try to move it to the left as much as possible.

This process is repeated while the number of moves is different from zero.

When a job is selected for moving, we consider its operations or blocks one at a time, starting from the end. For each operation we study all its possible starting times, as we did in Step 1 of Section 3.1. If a new position is found, we make the move and go to the previous operation of the job. If the operation cannot be moved, we still try to move its previous operation, in order to obtain a process of the job which is as compact as possible.

- *Phase II: Moving the jobs with successors to minimize the work in progress*

  For each job J_i,

  - If all its successors finish by their due dates, move the job as much as possible to the right.
  - If any successor finishes after its due date, move the job as much as possible to the left, hopefully leaving room for its successor to also be moved to the left, nearer to its due date.

  If Phase II produces some moves, we go back to Phase I and the whole process is repeated again.

  Note that most of the moves produce an improvement of the objective function but, when a job is moved to the left, its finishing time may go from (d_i, l_i) to (e_i, d_i), temporarily worsening its cost. Every time a new period t is considered as the starting time of an operation, the highest priority machine is always the first to be considered, and this secondary objective is taken into account.

4 Implementation

The algorithms described above have been coded in C++. The scheduling system is integrated into the information system of the factory. When it is called from the production planning system, it reads the relevant information from the ORACLE database and the solution obtained is written again in the database, then becoming available to other systems.

The preliminary tests run so far show that the scheduling system is able to provide tight due dates for incoming customer orders in a few seconds and can perform a complete mid-term plan, involving several hundred jobs with thousands of operations in a few minutes. For instance, a problem with 700 jobs, 2000 operations and 60 machines took less than 2 minutes. The system is currently being implemented at the factory and is now in a process of fine tuning in which all the details that the model cannot accommodate completely are being taking into account.

A future line of work will be to develop more complex moves, based on the interchange of the operations being processed on a machine [4], [5], [6]. That will provide the improving phase with greater flexibility to satisfy the due dates more efficiently and to minimize the work in progress.

References

[1] ANDERSON,E.J. AND NYIRENDA, J.C. (1990) Two new rules to minimize tardiness in a job shop, *International Journal of Production Research* **28** (12), 2277-2292.

[2] BAKER,K.R. AND BERTRAND,J.W.M. (1982) A dynamic priority rule for scheduling against due-dates, *Journal of Operations Management* **3** (1), 37-42.

[3] BAKER,K.R. AND KANET,J.J. (1983) Job shop scheduling with modified due dates, *Journal of Operations Management* **4** (1), 11-22.

[4] BRANDIMARTE, P. (1993) Routing and scheduling in a flexible job shop by tabu search, *Annals of Operations Research* **41**, 157-183.

[5] DAUZERE-PERES, S. AND PAULLI,J. (1997) An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search, *Annals of Operations Research* **70**, 281-306.

[6] HURINK,J., JURISCH,B. AND THOLE,M. (1994) Tabu search for the job-shop scheduling problem with multi-purpose machines, *OR Spektrum* **15**, 205-215.

[7] VEPSALAINEN,A.P.J. AND MORTON,T.E. (1987) Priority rules for job shops with weighted tardiness cost, *Management Science* **33** (8), 1035-1047.