Workload simulation and optimization in multi-criteria hybrid flowshop scheduling: a case study

Arianna Alfieri

Workload simulation and optimization in multi-criteria hybrid flowshop scheduling: a case study

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2007-IJPR-0792.R1
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	01-Feb-2008
Complete List of Authors:	Alfieri, Arianna; Politecnico di Torino, DSPEA
Keywords:	FLOW SHOP SCHEDULING, SIMULATION
Keywords (user):	MULTI-CRITERIA OPTIMIZATION

Workload simulation and optimization in multi-criteria hybrid flowshop scheduling: a case study

A. ALFIERI*

Dipt. Sistemi di Produzione ed Economia dell'Azienda, Politecnico di Torino, Torino, Italy

We study a real-life multiple objective flowshop scheduling problem in a cardboard company which differs from the conventional flowshop scheduling problem in several aspects, such as multi-machine stations, sequence-dependent setup times, work calendars on resources, reentrant flows, external operations, and transfer batches between stations. We present a simulation-based environment in which the production sequence can be interactively chosen by the user or found by a tabu-search based heuristic algorithm while a discrete-event simulation deals with the timing aspect.

Keywords*:* General Flowshop; Simulation; Multi-objective Optimization; Tabu Search; Interactive Scheduling

1. Introduction

We study a scheduling problem arising in the cardboard industry, with the objective to develop a modular decision support system for the daily workload planning. Our study has been motivated by a real industrial case but we believe that it can be easily extended to other industries as well. The production process in this industry is a hybrid flowshop, i.e., a multistage production system where, at every stage, a number of (non-identical) machines operate in parallel. Each customer order, depending on the product type, must be processed on each stage of the line, or only on some stages, or has to undergo some external processing

between two successive internal operations. Other peculiar aspects are the presence of sequence-dependent setups, transfer batches (different from production batches), and work calendars on resources. Moreover, there are tight and dynamic due dates, setups are very time-consuming and it is difficult to quantify priorities and the relative importance of orders and customers.

On the one hand, the firm wants to avoid late orders, especially for the most important customers; on the other hand, it is also important to try and reduce the total setup time. These two objectives are conflicting, since to satisfy the first (avoiding late orders) one must schedule orders on the basis of individual due dates, while the second (minimizing setup time) requires to consolidate orders belonging to the same family.

Due to the complexity of the problem, we considered the sequencing and timing aspects separately. For the sequencing we developed an interactive framework and a tabu search algorithm that can be alternatively used, depending on the user's preference. After the sequence has been created, a discrete-event simulator is used to deal with the timing of the sequence.

The aim of our work is not to develop the best algorithm to solve a scheduling problem, but to create a *scheduling environment*, flexible and easily controllable by the user, that is also effective in situations where not all the constraints and/or the objectives can be formalized.

Several papers appeared in literature dealing with multi-objective and hybrid flowshop scheduling problems, but most of these studies approached just one of the two aspects, i.e., either considered single-objective hybrid flowshops, or dealt with a multi-objective optimization in layouts such as single-machine or conventional flowshops.

Examples of multi-objective optimization in a single-machine layout can be found in Wan and Yen (2002), Choobineh *et al.* (2006) and Eren and Guner (2006). In Wan and Yen (2002) the total weighted earliness and tardiness is minimized, when jobs have distinct due-windows, by developing a tabu search to generate

* Email: arianna.alfieri@polito.it

sequences and using an optimal timing algorithm to determine job completion times. A tabu search algorithm is also used in Choobineh *et al.* (2006), to solve a three-objective optimization (makespan, maximum tardiness and number of tardy jobs) in presence of sequence-dependent setup times. The minimization of the sum of total tardiness and total completion time, with sequence-dependent setup times is approached in Eren and Guner (2006) by a combination of a special heuristic algorithm and tabu search.

Multi-criteria flowshop scheduling problems were, instead, considered in Rajendran (1995), Chou and Lee (1999), Armentano and Arroyo (2005) and Gupta (1999). In Rajendran (1995) the problem of minimizing makespan and total flowtime is studied, and heuristics based on preference relations are proposed. Potential job interchanges were checked for possible improvements with respect to the two objectives. Makespan and total flowtime minimization, in a two-stage flowshop, was dealt with also in Gupta (1999). The solution algorithm is based on a tabu search algorithm and total flowtime is considered as a secondary criterion, i.e., first the makespan minimization is considered, then the total flowtime is minimized with a constraint on the makespan value previously found. Almost the same problem was treated in Chou and Lee (1999) for a two-stage flowshop with release dates for jobs. They developed a heuristic algorithm based on a look-ahead priority number. A genetic algorithm to solve the makespan and maximum tardiness (or total tardiness) optimization is studied in Armentano and Arroyo (2005), using Pareto dominance to assign fitness to solutions.

In the field of hybrid flowshop literature, most of the studies were devoted to single-objective optimization. To cite some examples, Choi *et al.* (2005), Lee and Kim (2004) and Lee *et al.* (2004) considered the total tardiness optimization using different approaches: list scheduling in Choi *et al.* (2005), branch-and bound with dominance properties in Lee and Kim (2004) and a heuristic method based on the bottleneck stage in Lee *et al.* (2004). While Lee and Kim (2004) and Lee *et al.* (2004) consider no critical aspects, the peculiarity of Choi *et al.* (2005) is the presence of reentrant flows on some stages. The number of tardy jobs, another

common measure of service level, is optimized in Gupta and Tunc (1998) using list scheduling or dominance property on the second stage. Adaptation of several local search approaches to deal with a multiprocessor non-permutation flowshop scheduling problems was, instead, presented in Negenman (2001), with the objective to minimize the makespan. Makespan minimization is also studied in Jin *et al.* (2006) when the hybrid flowshop has more than three stages. Firstly jobs are sequenced and afterwards allocated to the identical machines of a stage by using an algorithm based on simulated annealing and variable-depth search. While in most of the cases hybrid flowshops are considered to have identical processors in each stage, in Soewandi and Elmaghraby (1998) a two-stage flowshop with uniform machines in each stage is studied and several heuristics are developed to minimize the makespan.

Differently from the above cited papers, Yang *et al.* (2004) and Janiak *et al.* (2007) dealt with both the aspects of multi-processor stages and multi-objective optimization. A general neighborhood-search algorithm for multi-objective hybrid flowshops is presented in Yang *et al.* (2004). They used a Pareto dominance concept to select a solution from the neighborhood and applied this technique to the problem of minimizing the makespan and the maximum tardiness. Earliness, tardiness and waiting time were, instead, considered in Janiak *et al.* (2007). In this case, however, the multi-objective optimization is reduced to a single-objective one by "transforming" all the objectives in monetary costs. They designed several solution procedures based on tabu search and simulated annealing and operating on the notion of operation processing order.

In the cited papers, only few complicating aspects are considered. In general (with the exception of Wan and Yen (2002)) there is no separation or distinction between the sequencing and the timing aspect of the scheduling problem. This is due to the fact that the objective functions they consider are time-dependent regular objective functions.

An approach similar to ours is presented in Armentano and Arroyo (2004), where a tabu search algorithm for finding the sequence on a permutation hybrid flow shop was developed. They considered a case study

from multilayer ceramic capacitator manufacturing and used a discrete-event simulation to compute the performance of a given sequence, in presence of stochastic elements, when the objective was the minimization of the total tardiness. The neighborhood was the exchange of two adjacent jobs and the first solution improving the current solution was chosen. The main difference from the last cited paper and our work is the fact that they do not consider complicating factors such as sequence-dependent setups, transfer batches, resource work calendars (i.e., time-windows) and multiple objective.

The paper is organized as follows. In section 2, the cardboard industry case study is introduced in detail. The scheduling environment is described in section 3, while sections 4 and 5 respectively deal with sequence creation (interactive and algorithmic) and timing. In section 6, the results of the application of our approach on the cardboard industry are reported. Finally, in section 7, some conclusions are drawn.

2. Case study: a cardboard industry

This paper presents a study that has been carried out in a small firm which produces and customizes cardboard boxes to order. A range of materials and coatings can be dealt with, including specific paintings, blisters, paper coupled to aluminum, PVC windows, reliefs, plasticization, and packaging in general. The operations for the manufacturer are carried on different machines, grouped in four stages (see Figure 1).

INSERT FIGURE 1 about HERE

The first stage is printing, where two offset printing machines print figures with 2 to 5 colors on sheets of given material. If more than 5 colors are needed, the order has to be processed twice on the printing stage (i.e., reentrant flow on the printing stage must be allowed). The two machines are different both in terms of printing speed and of maximum format of sheet they can accommodate. The second stage is the punch cutters, where

three automatic machines cut the printed sheet according to the shape of the box/display. Also in this case, the machines are different with respect to cutting speed and format they can accommodate. After cutting, additional elements, such as saw, windows, etc. can be applied, depending on the box or display we are producing. These applications are made in the third stage, consisting of two highly flexible finishing machines. Finally, boxes and displays are bent and glued. This task is performed on the last stage, where three identical machines are able to bend and glue any type of box/display.

The third stage can be skipped if no additional elements have to be included in the product. Moreover, it is possible to have surface treatments on printed sheets, that cannot be done in the firm and must be outsourced to contract manufacturers. In these cases, sheets are sent to specialized firms to undergo those treatments after printing; when those orders come back, they are queued directly to punch cutters.

Finally, there are other very special operations that have to be carried on a small number of orders and that can be placed in different positions of the process plan, depending on the order characteristics. This aspect complicates further the system we are considering.

Customer orders usually correspond to production jobs and for this reason, in the following, we will use the two terms interchangeably. Jobs consist of a possibly huge number of identical boxes to be produced, which could correspond to several pallets of sheets. In this situation, to reduce the manufacturing lead time, transfer batches different from process batches are used. This means that, after a pallet of sheets has been processed in one stage, it is immediately transferred to the next, without waiting for the other pallets of the same job.

If jobs are very different in terms of colors, shape and additional elements, then significant setup times are required in each stage. Moreover, while on stage from 2 to 4 (punch cutters, finishing and gluing) setup times depend only on the job to be processed, they are sequence-dependent on printing machines. Even for the

stages where setup are independent of the sequence, however, they cannot be included in the processing times, since the setup time for a job depends on whether the order is produced for the first time or not.

Finally, since products are highly customized (e.g., customers' brand usually appears on boxes or displays, customers use to frequently vary their product to make them more appealing and so on), the production is make-to-order and the main strength of the firm is the flexibility and the ability to meet customer needs, who set their orders with, usually tight, deadlines.

3. An interactive simulation-based environment

The problem described in the previous section is a hybrid flowshop with non-identical processors, reentrant flows, transfer batches, setup times separated from process times and sequence-dependent, and jobs are associated with different relative importance (in terms of different customer priority). It is clear how, in this environment, the problem of scheduling becomes complex. At present, there is no systematic procedure for scheduling. The daily workload is mostly done by sequencing the orders of the most important customers first, despite the status of the others, with a negative effect on the overall service level.

Due to the complexity of the problem, an optimization approach seems highly impractical, or even impossible, in this industrial environment.

For this reason, the aim of our study was to design and implement a practical decision support system for production scheduling. We developed an interactive simulation-based environment consisting of two basic functionalities: 1) work-order (i.e., sequence) creation; 2) schedule simulator.

The decision support system has been designed and implemented keeping modularity in mind, so that alternative approaches can be pursued. The two functionalities (sequence creation and simulation), are contained in two macro-modules, either one composed by separated modules which interact through standardized data structures. The software architecture is depicted in Figure 2.

INSERT FIGURE 2 about HERE

Through a data entry module, the user can choose how to perform work order creation on the first stage. In fact, the work order creation can be carried out in two different ways: 1) interactively decided by the user; 2) using a heuristic algorithm based on tabu search. Details of both these approaches are reported in section 4. Notice that when an *automatic* approach is chosen, the user has also to explicitly define the objective function, which is needed for the scheduling algorithm.

Once the work orders have been created (either manually or by the scheduling module), they are used as input sequences in the schedule simulator, which loads all the relevant information on jobs and resources, and simulates job completion. All the complicating aspects are dealt with by the simulator. In fact, we cope with different work calendars on resources, machine availability, residual queues, external operations, etc.

In case we choose the automatic generator with tabu search refinements, a strict interaction between the simulator and the tabu search sequence perturbation module is used to improve the solution. A similar interaction, but manually controlled, can be used by the planner, who can decide to directly perturb the sequence she has created.

The output consists of detailed work schedules for each resource and each order, with emphasis on due date performance. Since more than one solution can be kept in the final pool, the user can evaluate which solution fits her objectives best. Once the user has chosen the "best" solution, work order can be created and released to the shop floor for execution.

4. Sequencing

In our system, jobs are usually composed by a huge number of cardboard sheets, which requires a lot of space to be stocked during the process. This, in combination with the significant setup times, makes preemption in practice not possible. The scarcity of space on the floor also prevents, in practice, changes in the job sequence between two successive production stages. Finally, the printing stage is the only one with sequence-dependent setup times, and for this reason, in practice, it affects more than the others the performance of the whole production system.

In view of these practicalities, we chose to schedule orders (i.e., create work orders) on the first stage, and propagate the sequence to the other stages, allowing only permutation and non-preemptive schedules.

Since we are considering a multi-objective optimization, with possibly conflicting objectives, we may seek to generate a set of Pareto optimal solutions. A Pareto optimal solution is a non-dominated solution, i.e., a solution that is not worse than all others (in the feasible space) on all objectives and is better on at least one objective. This means that, if one solution is Pareto optimal, no objective can be improved without worsening at least one others. For example, in our problem, a sequence of jobs is Pareto optimal if we cannot reduce the total delay in orders delivery without increasing the time spent in setups or the number of tardy jobs.

For this reason, we developed two sequencing modules that can be alternatively used by the planner (see Figure 2). The first module uses priority rules to find initial sequences and a tabu search algorithm to possibly improve them. The output of this module is not a single solution, but a set of *good* solutions, from among which the planner can choose. The tabu search module can be also skipped if the user does not want any refinement on the initial solutions.

The alternative sequencing module has a simple graphical user interface that allows the planner to directly choose and perturb the sequence on the first stage. Also in this case, all the solutions explored by the user can be kept in a final pool.

4.1 Tabu search based heuristic

The creation of the sequences is the core of our system and it is done by a tabu search based heuristic. Tabu search is a heuristic neighborhood search algorithm developed for solving complex optimization problems (Glover (1989,1990)).

A basic tabu search algorithm operates in the following way. It starts from an initial feasible solution from which a set of neighbors are generated using a number of previously determined movement strategies. The objective function is evaluated for each solution in the neighborhood and the best one (in the neighborhood) replaces the current solution, even though it is worse. In this way it is possible to escape from local optima. The algorithm iterates until some given stopping condition is reached.

The set of allowed neighbors is restricted by a tabu list designed to prevent going back to recently visited solutions and becoming trapped in a loop. At each step, the neighborhood from which the next solution will be drawn is redefined on the basis of the conditions that label certain moves as tabu. The efficiency and effectiveness of the algorithm highly depend on the attribute used to classify a solution as *tabu* and on the tabu list length.

In the multi-objective optimization, a unique optimal solution may not exist, and hence a unique best solution may not exists in a neighborhood. A way to deal with this problem is to use the concept of Pareto optimality to characterize optimal solutions. Given the concept of Pareto optimality, a multi-objective tabu search can works with a set of current solutions which are simultaneously optimized towards the non-dominated frontier.

A number of papers have dealt with multi-objective tabu search. To cite some example, in Baykasoglu *et al.* (1999) a general tabu search scheme for multi-objective optimization to find the Pareto optimal solutions is considered, and applications to several examples are presented. A general adaptation of tabu search for

generating an approximation of the non-dominated set for multi-objective optimization problems of combinatorial nature is, instead, presented in Pilegaard (1997).

A way to achieve the Pareto optimal solution is the (linear) scalarization of the objective function. In practice, the multi-objective (vector) function is replaced by a weighted (scalar) function to evaluate the solution found by the tabu search algorithm and store only the "best" in each neighborhood. If the feasible region is not convex, the linear scalarization might fail in finding the solutions on the efficient frontier and different methods must be used; in these cases, linear scalarization leads in general to sub-optimal solutions.

The algorithm we developed is based on the dominance concept and on the weighted objective function and works in the following way.

All the initial solutions, both non-dominated and dominated, become seeds, that is, will be used to start a neighborhood search. The initial non-dominated solutions are inserted in an *optimality list*, containing, at the end of the search, the solutions among which the user can choose.

During the exploration of each neighborhood, we keep the non-dominated solutions, that are included in the *optimality list*, but we use a weighted function to evaluate the best solution that becomes the seed for the successive tabu search iteration.

Since our primary objective was to develop a decision support system, we also designed a rather basic tabu search algorithm, without any additional feature such as aspiration criteria, long-term memory, and strategic oscillation, that could improve its searching capability. However, due to the modularity of the system, more refinements in the tabu search heuristic can be added in the future.

In the following, we discuss in detail the algorithm used to find the initial set of solutions, the neighborhood structure and the tabu attribute used to search the solution space.

4.1.1 Initial solution

We find an initial set of solutions to be possibly used as seeds for the tabu search algorithm by applying an ad hoc constructed priority rule. Priority rules are mechanisms to assign to each order a value representing its importance. The higher the value, the higher the importance and hence the priority of the order. The production sequence is created on the basis of these values, placing the most important orders at the beginning of the sequence.

Of course, the rule used to assign the priority to the orders depends on the objective function. In the literature, many rules have been developed for various scheduling problems, but in real cases and for complex environment, ad hoc rules have to be designed.

In the case of cardboard industry, it is important to meet due dates and to reduce time spent in (sequence-dependent) setups at the printing machine stage. To try and achieve these two objectives, we have created a parametric composed rule as follows:

$$\pi_i = \left(\alpha \left(\frac{DD_i}{w_i} \right) + (1-\alpha) ts_{ki} \right)^{-1},$$

where π_i, DD_i and w_i represent priority, due date and weight (how important, in respect to the others, is the customer) of order i respectively, ts_{ki} is the time necessary to set the printing machine for order i if it was printing order k.

Notice that π_i increases if DD_i and/or ts_{ki} are smaller, which means that an order is considered to be more urgent if its due date is closer in time and/or the preparation time for the printing machine is small. The more urgent an order, the fewer predecessors in the sequence it will have, and hence the sooner it will be processed.

Depending on the situation (a period with high demand or a reduced capacity due to maintenance or else), it could be more important to reduce setup times or to meet due dates, that is, the two criteria can have a different importance for the planner. To assess the tradeoffs between efficiency (setup times) and effectiveness (meeting customer due dates), we use the parameter α. When α is equal to 1, the sequence is constructed on the basis of due dates only (i.e. weighted EDD); instead, if α is equal to 0, only setup times are considered. In our case, the algorithm iterates over α, starting from $\alpha = 0$ and augmenting it with a constant step till it reaches 1, so as to generate a larger set of initial solutions.

Varying α, in fact, we can generate various sequences, all potentially good but structurally different. If our scheduling algorithm would stop here, the user could evaluate on each sequence the tradeoff between the objectives and then choose which one to use. Alternatively, the sequences generated in this way can be the seeds for the successive tabu search, since, in general, they represent only a subset of all the good solutions and, in many cases, they can be improved.

Among the generated sequences, some could be dominated by others, but we do not eliminate them at this point, so as to have many different starting points for the tabu search algorithm, and this can help in exploring different regions of the solution space. Then we insert all the initial solutions in a *seed list* and the non dominated ones in an *optimality list*.

4.1.2 Neighborhood structure and tabu list

The sequencing achieved by the priority rule is quite myopic, since the consequences of each choice on the next sequencing decisions are not considered. To have a less myopic algorithm, other sequences can be obtained from the initial ones by the tabu search algorithm.

Given a sequence *s*, we defined the neighborhood *N(s)* as the set of sequences found by (1) exchanging adjacent orders on the same printing machine and (2) moving an order from one printing machine to the other (if allowed by its process plan), in the most *favorable* position in terms of setup time.

Given an initial solution *s* in the *seed list*, we explore its complete neighborhood *N(s)* (i.e., perturbing it in all the possible ways). Two situations can happen:

1. at least one sequence non dominated by any of the sequences in the *optimality list* is found;

2. no sequence is found that is non dominated with respect to the sequences in the *optimality list*.

In the first case, we use the best non dominated sequence as the next current solution for the tabu search algorithm and add all the non-dominated sequences to the *optimality list*. Moreover, we check if any of the sequences that were already in the *optimality list* are now dominated by the new found solution. If this is the case, the dominated sequences are deleted from the *optimality list*. We do not apply the same logic to the *seed list*, because this might result in a too restricted search, i.e., we keep a solution in the *seed list* even if it is dominated by some solution found by searching a neighborhood of another seed.

In the second case, when all the sequences in *N(s)* are dominated by some sequence in the *optimality list*, we choose the best one to be used as the next current solution for the tabu search algorithm.

The "best" sequence (dominated or non-dominated) is chosen by using a weighted function of delivery delay *DDelay* and total setup time *TotSetup*:

$$obj = \beta * DDelay + (1 - \beta) * TotSetup$$

The tabu search algorithm is performed several times, starting from the same *seed list*. Each time, a different $\beta \in (0,1)$ is used in above equation, starting from $\beta = 0$ and augmenting it with a constant step till $\beta = 1$. For each β, all the non-dominated solutions found are kept in the *optimality list*, which is evaluated by the user at the end of the search.

To prevent looping, we use as tabu attribute (as often done in literature) the opposite of the move originating the sequence. The algorithm stops when it reaches a predefined number of iterations, or when all the possible moves from each seed are tabu.

In practice, for each initial solution found by our heuristic algorithm (based on the described priority rule), we perform the neighborhood search for several different $\beta \in (0,1)$.

We tried also another tabu attribute: the value of the objective functions (i.e., the total time spent in setup and the total day delay), but it turned out to be too restrictive and the search was prematurely stopped with worse results.

4.2 User defined sequence

The tabu search heuristic allows to explore automatically several different sequences from among which the user can choose, but it does not allow the planner to direct the sequence creation process. This possibility is irrelevant if priorities and relative importance of orders can be easily specified and included in the algorithm.

Unfortunately, in many situations it is very difficult to quantify priorities and the relative importance of orders and customers. An alternative approach was therefore implemented, using the same simulator to deal with the timing aspect. This alternative approach simply consists of a *manual* sequence creation done by the user through a graphical interface. Given the orders to be scheduled, the user can simply assign each of them to one of the printing machines, choosing the position in the sequence. After all orders have been assigned and sequenced, the process simulation can be started by the user to compute the timing. This step is hidden if the automatic scheduler is chosen, since sequences are generated by priority rule/tabu search and simulated without the user's intervention.

Once the simulation is completed, a detailed report gives the completion time and the expected delivery delay of each order, together with aggregated information about the total expected delay and the total time

spent on setups on each machine. On the basis of this, the user can decide if the simulated sequence can be released to the shop floor or if it is necessary to modify it, by exchanging the position of some orders or by moving some orders to a different (feasible) printing machine. If some perturbation is needed, it can be done directly on the original sequence through the user interface.

In practice, the interactive scheduler is a kind of local search done manually by the user, who can explore different sequences and change the evaluation criteria to assess the quality of a schedule. Each explored sequence can be stored in a solution pool to compare them and select one at the end.

5. Timing

Once a sequence has been created on the first stage, we simulate how jobs will be processed by using a discrete event algorithm. During the simulation we collect data to assess the schedule/sequence in terms of due date performance and total setup times.

Discrete event simulation is based on the concept of *event*. An event is an instantaneous occurrence that can change the state of the system. In practice, a discrete event simulation models the evolution in time of a system represented by state variables that change at separate and countable points in time.

In our system, events are (a) the arrival of an order in production (i.e., all the materials are available and the order can be sent to production) and (b) the availability of a machine (i.e., the "previous" order on the machine has just finished and then the machine is available to process successive orders). We used the classical *next-event time advance* mechanism to update the *simulation clock* and have the simulated time passing. With this mechanism, the simulated time advances to the time of the most imminent among future events. The state of the system is updated taking into account that the event has occurred and future events generated by that event are determined. Afterwards, the simulation clock is updated again to set the simulated time at the time of the new most imminent future event and so on. For example, if a machine becomes

available, the first job in the queue starts processing and this generates the future event of *end processing* on that machine, that corresponds to an event of *machine availability*.

The same simulation algorithm can be applied both to user defined sequences and to the sequences generated by the tabu search algorithm. In the tabu search algorithm, we use the collected data to drive the search, i.e., simulation is integrated with optimization in an iterative framework (the "best" solution in the neighborhood is evaluated on the basis of delivery delay and total setup time, computed once the sequence has been simulated). The output will be a set of potential good (non dominated) sequences.

Instead, with user defined sequences, the iterative process is *user-driven*, i.e., the number of iterations to reach "good" solutions is not stated a priori, but decided on the fly by the user. The user creates a sequence, simulates it, and assesses objective values. On the basis of those values, and on the details of the solution (e.g., which is the most delayed order or how much delay has an important customer), the user can decide to continue, changing the sequence on the first stage and simulating it from the beginning, or to stop the search.

A flowchart representing how the overall system works is represented in Figure 3.

INSERT FIGURE 3 about HERE

Note that, despite the choice made (automatic sequencing by priority rule, automatic sequencing followed by tabu search or manually sequencing), the timing module (*simulate*) does not change. The flowchart clearly shows the modularity of our architecture: any change in some points of the algorithm (simulation, scheduling or tabu search) does not influence the rest of the architecture. This implies that a modification in the criteria to evaluate a solution (used in the schedule and tabu search module) does not impact the simulation; any change in the production plant, leading to change in the simulation algorithm, does not have any influence on the way sequencing is done or solutions are reported and so on.

6. Computational tests

The scheduling/simulation environment was implemented in Excel/VBA, and it was tested using real-life data. We followed the weekly production planning for one month, using our tool to determine the weekly order sequence. In each week, we planned for a complete month, using a rolling horizon approach. In practice we run four tests, and each test used the data corresponding to the orders for the next four weeks. The number of customer orders, in each of the four tests, were in the interval (50, 70) and this is the order of magnitude also during the other months.

We used the priority rule described in section 4.1.1 to find the set of initial solutions, with parameter α ranging from 0 to 1 with step of 0.1. We chose a step of 0.1, since preliminary tests showed that a bigger step led to a too restricted set of initial solutions, while a smaller step led to very similar solutions. Once the set of initial feasible solutions was found, the tabu search algorithm described in section 4.1.2 was run. We used a *tabu list* of length equal to 20, while the *optimality* and *seed list* are dynamic list, so as to contain possibly any number of solutions. Also for parameter β (ranging from 0 to 1), we used a step of 0.1. Both the length of the *tabu list* and the step for parameter β have been chosen after preliminary tests.

Each sequence was simulated and using the data collected during the simulation, it was evaluated in terms of objective functions which include the criteria established by the user. In our case, the criteria are the same used in the priority rule, that is *total setup time* on the printing stage and *delivery delay* of customer orders with respect to their due dates. While it is easy to measure the total time spent in setups, since it is enough to sum up the setup times of the sequenced orders, we had to define with the user a reasonable measure for the delay. For each sequence, we computed two measures of delay:

- $T_{max,}$ the maximum tardiness;

- $\sum_i w_i T_i$, the total delay, weighted by the customer importance (w_i).

We used the first criterion (T_{max}) to evaluate the quality of sequences, since reducing the maximum delay had the highest priority for the firm. Both the criteria, however, can be used to visualize the results, so as the user can evaluate the tradeoffs involved and decide which sequence to use in production. Moreover, since we built a modular architecture, it is possible to change a single criterion, or add it to the others in evaluating if a solution is non-dominated or the best in the neighborhood.

Each non-dominated sequence is stored both in an aggregated and detailed way, to be examined by the planner.

Tables 1 to 3 report the results of one of the four tests done with the firm's data. They correspond to an instance of 66 orders and the total CPU time was 187 seconds on a PC (clock 500 MHz, ram 256 Mb) with Windows NT operating system. As the firm was unable to assess customers' priorities, we set all w_i to 1. The results obtained by the automatic sequencing algorithm were positively judged in the firm, since the proposed sequences can be easily evaluated by the planner.

INSERT TABLES 1 to 3 about HERE

To evaluate the impact of customer priorities, we ran a second set of experiments using the same real-life data but randomly generating customers' weights w_i from a Uniform distribution between 1 and 5. For each real-life instance, ten sets of weights were generated.

Tables 4 to 6 report the results of the same instance as in tables 1 to 3, but with customers' weights corresponding to one of the ten randomly generated sets.

The total CPU time was 95 seconds, while the average total CPU time, over the ten sets of weights, was 90.6 seconds.

INSERT TABLES 4 to 6 about HERE

Considering the ten different sets of weights for the same real-life instance, the number of final non-dominated solutions and value of the different objectives (i.e., T_{max}, $\sum_i w_i T_i$, *TotSetup*), in each solution, were highly variable with w_i values.

Both these findings, i.e., smaller CPU time and variable solution quality, were expected. On the one hand, in fact, using different weights makes orders more different one another and this helps the algorithm to find good solutions. On the other hand, however, different weights can force the processing of an order even if it is not urgent from a due date point of view, and/or printing machine has to incur a large setup time, hence possibly increasing total delay and/or total setup time.

Despite the weights used, the tabu search algorithm, even if implemented in a very basic way, improves the initial solutions. A more powerful algorithm, with diversification and intensification strategies, tabu list dynamic length, usage of dominance also in solution evaluation, could probably give major improvements, but this is not within the scope of our work.

The most important aspect, and major contribution, of our work lies indeed in its flexibility and modularity. In fact, each module of the software architecture (scheduling module, timing module and also the service modules such as data entry and result reporting) is separated from the others and from the graphic user interface by standardized data structures which makes the overall architecture very flexible. Figure 2 schematically reports how the various modules are connected each other. Examples of such flexibility are

- possibility to change the objective functions used to evaluate the solutions without having to modify the simulation, scheduling, data entry and reporting modules;

- we can change the algorithm used to find the initial solutions, or how they are improved, or we can just decide to give a user defined sequence in input: whatever we choose, it does not impact on the simulation or evaluation modules;

- how sequences are simulated for the timing is independent from data entry. For example, processing times can be given explicitly or computed by other system parameters without impacting on the simulation itself;

- output file can be created according to user's preferences (e.g., solutions aggregated by machine, or by customer order), without having to modify the simulator, or the evaluation module or data entry.

7. Conclusion

In this paper we propose an interactive and modular environment as a support tool for production scheduling in a cardboard company.

The main contribution of our work lies in its modularity, since, due to the standardized data structures separating each module, the proposed software has a high degree of flexibility. This flexibility allows for an easier maintainability and re-customization, in case of changes in the system to manage.

From a user's point of view, there is also a major effectiveness in respect to the commercial scheduling software. In fact, the output of our tool is not a single solution but it is a set of solutions from among which the user can choose using non-quantitative “criteria”, which are difficult (exactly because they are non-quantitative) to encode in a software. Moreover, with the local search logic, it is possible to link the search of new solutions to the result (in terms of objective functions) achieved by the already explored solutions (i.e., sequences).

At the time we are writing, our software has been implemented in the firm and it is currently under evaluation. The first results of its implementation, however, are positive and indicates that it can be helpful not only in the production planning phase but also as a support for quoting due dates to customers. In fact, in the order acquisition phase, the new order is added to the orders already scheduled and the new sequence simulated. The output of the simulation gives the expected completion time of the order that can be used to quote the delivery date.

Despite the good result obtained on real-life data, many things can be added and improved. The tabu search may be improved, both by adding diversification and intensification strategies and by using the non-dominance property also for managing the *seed list* (i.e., storing in the seed list all the non-dominated solution found in the neighborhood). Also other more sophisticated scheduling algorithm can be tested (for example column generation algorithms).

Future improvements should also try to include a production activity control module. Such a module would allow using the tool not only for planning but also for real-time production management.

References

ARMENTANO, V.A., ARROYO, J.E.C., 2004. An Application of a Multi-Objective Tabu Search Algorithm to a Bicriteria Flowshop Problem. *Journal of Heuristics*,10, 463-481.

ARMENTANO, V.A., ARROYO, J.E.C., 2005. Genetic local search for multi-objective flowshop scheduling problems. *European Journal of Operational Research*, 167, 717-738.

BAYKASOGLU, A., OWEN, S., GINDY, N., 1999. A Taboo Search based Approach To Find the Pareto Optimal Set In Multiple Objective Optimization. *Journal of Engineering Optimization*, 31, 731-748.

CHOI, S.W., KIM, Y.D., LEE, C.G., 2005. Minimizing total tardiness of orders with reentrant lots in a hybrid flowshop. *International Journal of Production Research*, 43, 2149-2167.

CHOOBINEH, F.F., MOHEBBI, E., KHOO, H., 2006. A multi-objective tabu search for a single-machine scheduling problem with sequence-dependent setup times. *European Journal of Operational Research*, 175, 318- 337.

CHOU, F., LEE, C., 1999. Two-machine flowshop scheduling with bicriteria problem. *Computers & Industrial Engineering*, 36, 549-564.

EREN, T., GUNER, E., 2006. A bicriteria scheduling with sequence-dependent setup times. *Applied Mathematics and Computation*, 179, 378-385.

GLOVER, F., 1989. Tabu Search: Part I. *ORSA Journal of Computing*, 1, 190-206.

GLOVER, F., 1990. Tabu Search: Part II. *ORSA Journal of Computing*, 2, 4-32.

GUPTA, J.N.D. , TUNC, E.A., 1998. Minimizing tardy job in a two-stage hybris flowshop. *International Journal of Production Research*, 36, 2397-2417.

GUPTA, J.N.D., PALANIMUTHU, N., CHEN, C-L., 1999. Designing a tabu search algorithm for the two-stage flow shop problem with secondary criterion. *Production Planning & Control*, 10, 251-265.

JANIAK, A., KOZAN, E., LICHTENSTEIN, M., OGUZ, C., 2007. Metaheuristic approaches to the hybrid flow shop scheduling problem with a cost-related criterion. *International Journal of Production Economics*, 105, 407-424.

JIN, Z., YANG, Z., ITO, T., 2006. Metaheuristic algorithms for the multistage hybrid flowshop scheduling problem. *International Journal of Production Economic*s, 100, 322-334.

LAW, A.M., 2006. *Simulation Modeling and Analysis*, 4Rev Ed, McGraw-Hill.

LEE C.G., KIM, Y.D., 2004. A branch-and-bound algorithm for a two-stage hybrid flowshop scheduling problem minimizing total tardiness. *International Journal of Production Research*, 42 4731-4743.

LEE C.G., KIM, Y.D., CHOI, S.W., 2004. Bottleneck-focused scheduling for a hybrid flowshop. *International Journal of Production Research*, 42, 165-181.

NEGENMAN, E.G., 2001. Local search algorithms for the multiprocessor flow shop scheduling problem. *European Journal of Operational Research*, 128 147-158.

PILEGAARD, H.M., 1997. Tabu Search for Multiobjective Optimization: MOTS. *MCDM '97*, Cape Town, South Africa.

RAJENDRAN, C., 1995. Heuristics for scheduling in flowshop with multiple objectives. *European Journal of Operational Research ,* 82, 540-555.

SOEWANDI, H., ELMAGHRABY, S.E., 1998. Sequencing on two-stage flowshop with uniform machines to minimize makespan. *IIE Transaction*, 35, 467-477.

YANG, T., KUO, Y., CHANG, I., 2004. Tabu-search simulation optimization approach for flow-shop scheduling with multiple processors - a case study. *Interational Journal Production Research*, 19, 4015-4030.

WAN, G., YEN, B.P.C., 2002. Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. *European Journal of Operational Research*, 142, 271-281.

Table 1. Initial solutions from composite priority rule (all customers equally important).

	T_{max}	$\sum_i w_i T_i$	*TotSetup*
sol1	10 days	42 days	50.5 hours
sol2	4 days	7 days	72.5 hours
sol3	2 days	4 days	76.0 hours
sol4	3 days	5 days	73.0 hours

Table 2. Solutions found by tabu search algorithm (all customers equally important).

	T_{max}	$\sum_i w_i T_i$	*TotSetup*
sol5	8 days	31 days	63.0 hours
sol6	3 days	6 days	70.0 hours
sol7	2 days	5 days	72.5 hours
sol8	2 days	4 days	73.0 hours
sol9	1 days	7 days	69.5 hours

Table 3. Solutions proposed to the user (all customers equally important).

	T_{max}	$\sum_i w_i T_i$	*TotSetup*
sol1	10 days	42 days	50.5 hours
sol5	8 days	31 days	63.0 hours
sol6	3 days	6 days	70.0 hours
sol7	2 days	5 days	72.5 hours
sol8	2 days	4 days	73.0 hours
sol9	1 days	7 days	69.5 hours

Table 4. Initial solutions from composite priority rule (weights uniformly distributed in (1,5)).

	T_{max}	$\sum_i w_i T_i$	*TotSetup*
sol1	10 days	119 days	71.5 hours
sol2	6 days	39 days	74.0 hours
sol3	5 days	33 days	72.5 hours
sol4	4 days	23 days	76.7 hours
sol5	4 days	21 days	76.7 hours
sol6	6 days	15 days	76.5 hours

Table 5. Solutions found by tabu search algorithm (weights uniformly distributed in (1,5)).

	T_{max}	$\sum_i w_i T_i$	*TotSetup*
sol7	8 days	82 days	61.5 hours
sol8	5 days	30 days	73.5 hours
sol9	3 days	20 days	74.0 hours
sol10	4 days	16 days	75.5 hours
sol11	6 days	16 days	70.5 hours

Table 6. Solutions proposed to the user (weights uniformly distributed in (1,5)).

	T_{max}	$\sum_i w_i T_i$	*TotSetup*
sol3	5 days	33 days	72.5 hours
sol6	6 days	15 days	76.5 hours
sol7	8 days	82 days	61.5 hours
sol8	5 days	30 days	73.5 hours
sol9	3 days	20 days	74.0 hours
sol10	4 days	16 days	75.5 hours
sol11	6 days	16 days	70.5 hours

Figure 1: Production stages

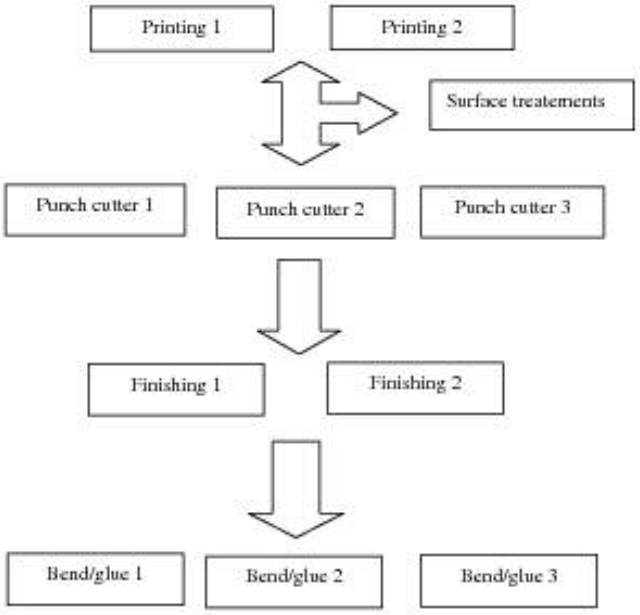


Figure 2: Scheduler/simulator architecture

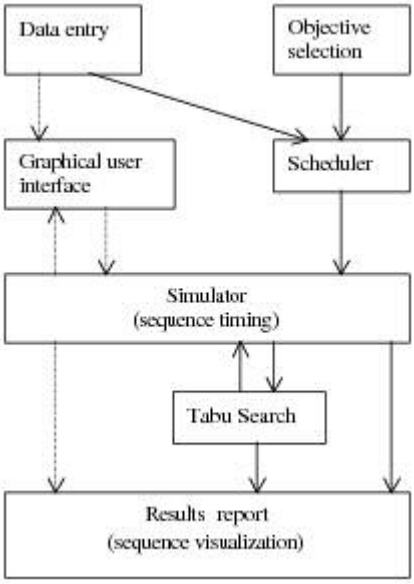
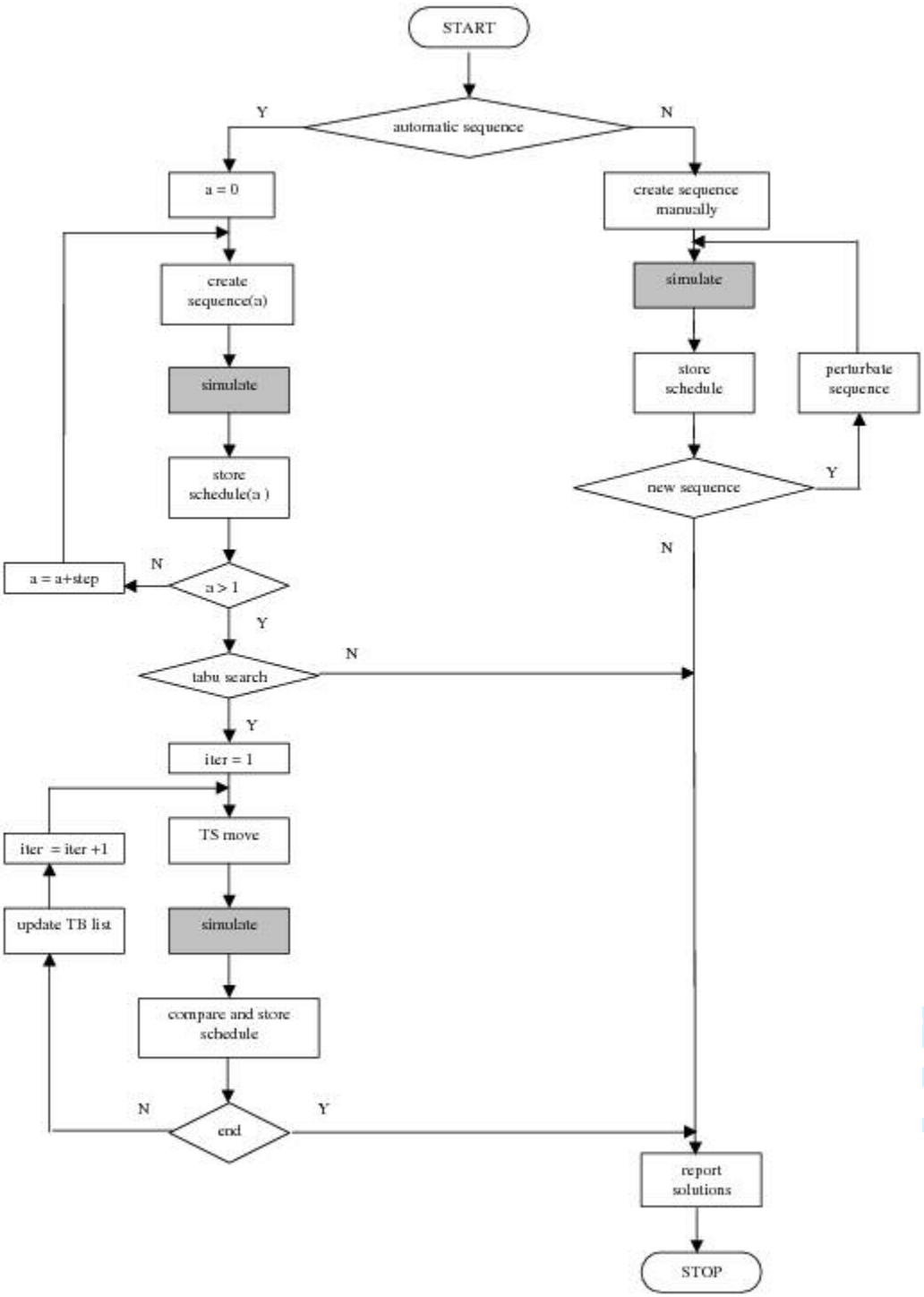
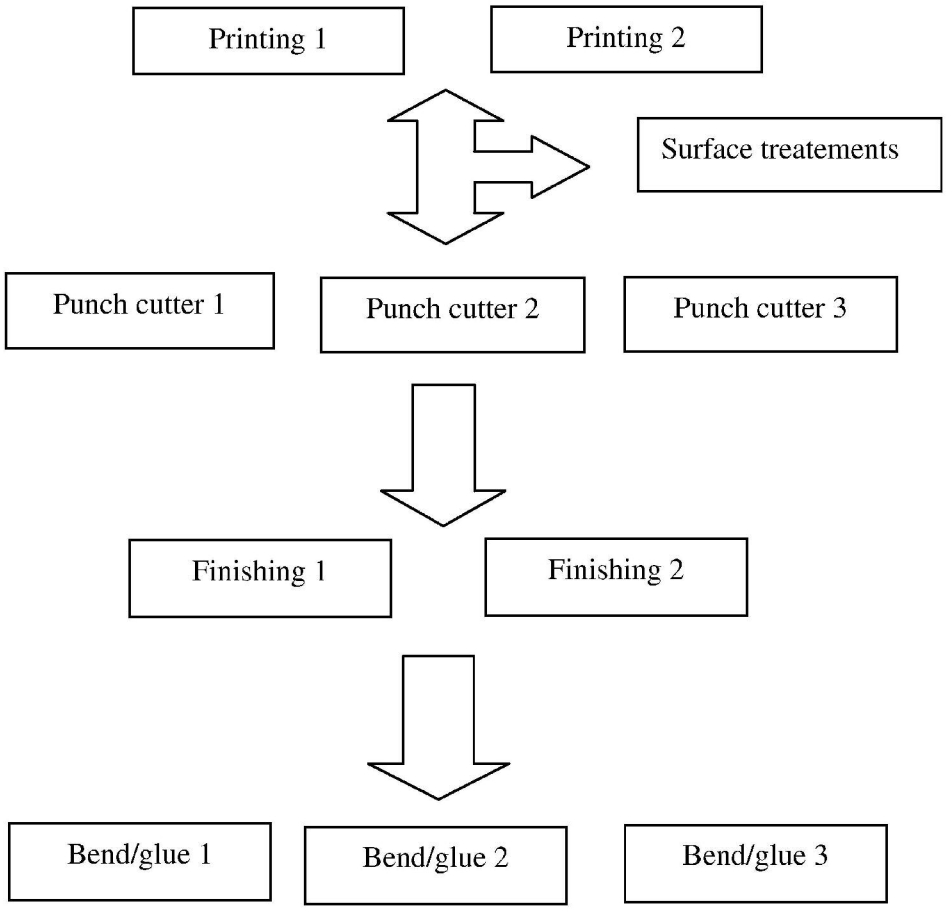
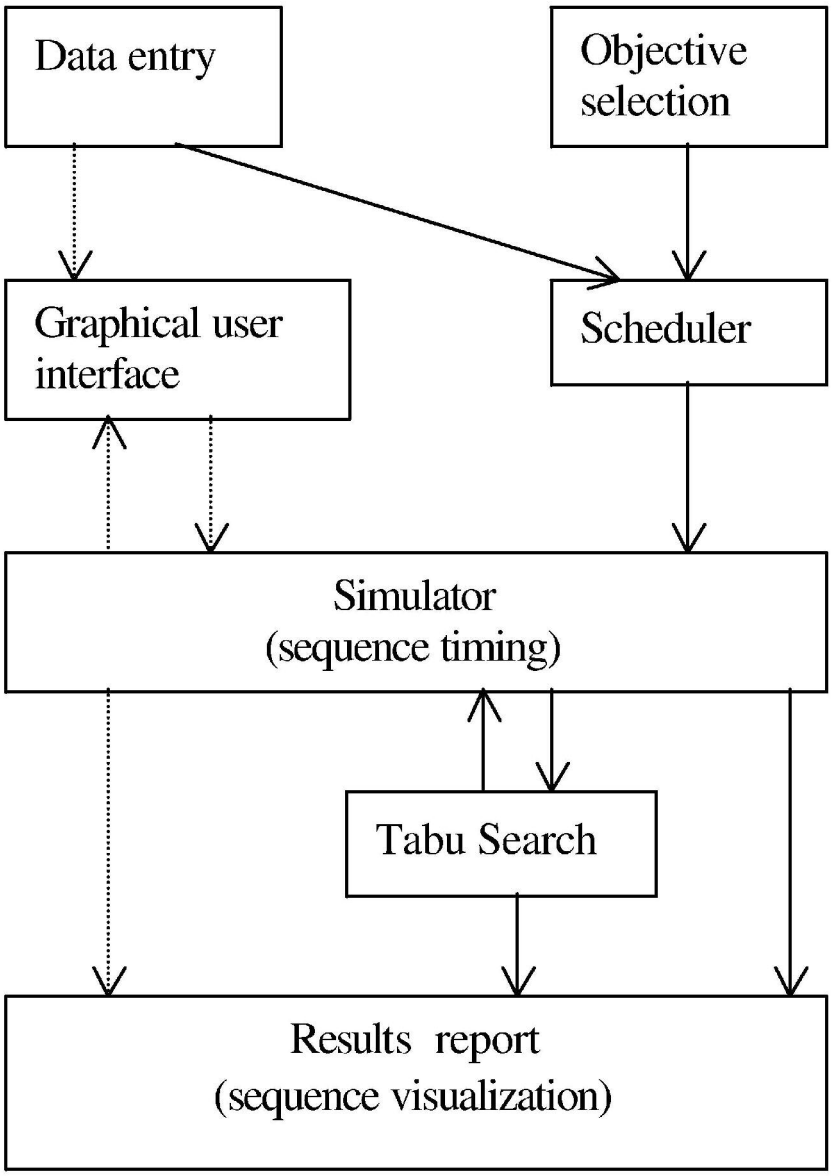


Figure 3: Flowchart of the overall system

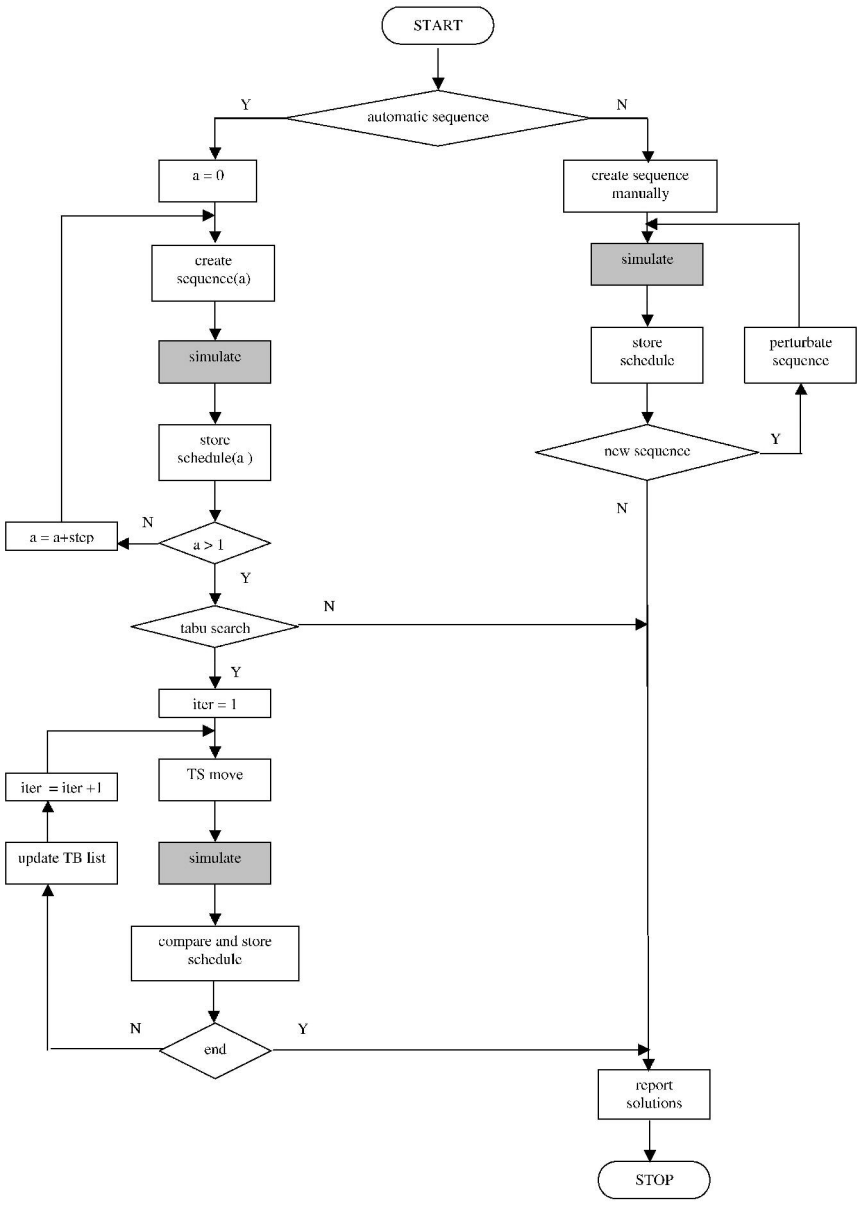




Production stages
117x114mm (600 x 600 DPI)



Scheduler/simulator architecture
80x111mm (600 x 600 DPI)



Flowchart of the overall system
195x273mm (600 x 600 DPI)