

**A two-stage methodology
for short-term batch plant scheduling :
Discrete-Event Simulation and Genetic Algorithm**

Catherine AZZARO-PANTEL, Leonardo BERNAL-HARO, Philippe BAUDET*,

Serge DOMENECH, Luc PIBOULEAU

Laboratoire de Génie Chimique - UMR CNRS 5503

ENSIGC INPT

18, Chemin de la Loge

F-31078-Toulouse Cedex - FRANCE

Tél : +33 (0)5 62 25 23 73

Fax : +33 (0)5 62 25 23 18

* Author to whom all correspondence should be addressed

Abstract

In this paper, a two-stage methodology for solving jobshop scheduling problems is proposed. The first step involves the development of a discrete-event simulation (DES) model to represent dynamically the production system behavior, taking into account the main features inherent to the application field. Since most scheduling problems in batch processing belong to the family of problems classified as NP-complete, probabilistic optimization algorithms (simulated annealing, evolutionary algorithms...) represent a good alternative for solving large-scale combinatorial problems (for instance, traveling salesman problem). In the second step of our approach, we thus investigate Genetic Algorithms (GAs) for solving batch process scheduling problems : a GA has been developed for minimizing the average residence time to produce a set of batches in function of batch order in a multipurpose-multiobjective plant with unlimited storage. The evaluation of the objective function is provided by its coupling with the DES model embedded in the optimization loop. Computational results show that the use of this approach can significantly help improving the efficiency of the production system. This paper is focused on semiconductor application, which is the first example treated in our laboratory, but the general approach adopted in this study is now extended to other fields of applications (e.g. fine chemistry with finite intermediate storage and unstable intermediates).

Keywords : batch plant - scheduling - semiconductor manufacturing - discrete-event simulation - optimization- Genetic Algorithm.

1. Introduction

Batch processes are the prevalent mode of manufacture for specialty products which are of high added value, low volume or which require complex synthesis procedure and close control of process conditions (pharmaceuticals, cosmetics, polymers, biochemicals, food products and electronic materials). The typical features of batch processes involve considerable complexities to the related short-term scheduling problem. A special configuration of a batch facility which is by far the most complex one, is the multipurpose plant or jobshop. It consists of general purpose equipment items used to manufacture a variety of products following different routes through the plant. In this context, scheduling is an important factor governing plant operations, i.e., establishing the order in which the products are to be produced, how time should be used for the production of each batch and how they share common resources (equipment items, operators, storage tanks, raw materials, utilities...) in order to optimize a suitable objective.

Scheduling problems have received considerable attention in the literature (Birewar and Grossman, 1990), (Egli and Rippin, 1986), (Espuna and Puigjaner, 1989), (Grossman and Sargent, 1979), (Karimi and Reklaitis, 1985), (Ku and Karimi, 1988), (Ku and Karimi, 1990), (Ku and Karimi, 1991), (Lazaro et al., 1989), (Mauderli and Rippin, 1979), (Musier and Evans, 1980), (Rippin, 1983a), (Rippin, 1983b), (Rippin, 1993), (Wellons and Reklaitis, 1991a), (Wellons and Reklaitis, 1991b), (Yamalidou et al., 1990). Most of the published work involves simplifying assumptions, such as zero-wait task operations, negligible batch transfer times, labor availability, maximum allowable utility loads. Comprehensive reviews of the literature on batch plants are available in (Reklaitis, 1991) and (Rippin, 1993).

A typical feature of most scheduling problems is that determination of the production sequence is a combinatorial optimization problem in complex plant configurations : the number of candidate solutions grows exponentially with the problem size. Most flowshop and jobshop problems have been shown to be NP-complete problems. Efficient algorithms for obtaining an optimal solution exist only for very special cases of the simplified flowshop (e.g. Johnson's algorithm for two-stage, serial UIS flowshop). For very simple flowshop configurations, the related scheduling problem has been formulated using recurrence relations (Ku and Karimi, 1988). Another way to solve optimally the flowshop sequencing problem for small-size applications is to formulate the problem as a Mixed Integer Linear Program Problem (MILP), in which the problem formulation explicitly involves both integer and real variables. For larger problems, several heuristic algorithms have also been proposed based on the MILP formulation for

obtaining good, but suboptimal, solutions (Ku and Karimi, 1988). Another approach, using only integers as optimization variables, is to use the Branch and Bound (BAB) strategy (Ignall and Schrage, 1965). But as reported in (Ku and Karimi, 1988), for most flowshop sequencing problems, algorithms based on either MILP or BAB are unable to solve a problem larger than about 12 products. In a multipurpose plant, the short-term scheduling problem is considerably more complicated. The existing work in the chemical engineering literature has dealt with two kinds of problems : scheduling of individual batches of products and production runs of identical batches of a product. In the majority of these works, the MILP approach, the BAB strategy or a variant of these methods have been used (Ku and Karimi, 1988). The same conclusion as for flowshop configurations is valid : these approaches become rapidly intractable for practical large-size problems due to a combinatorial explosion.

This is why another approach combining a two-stage methodology for solving jobshop scheduling is proposed in this paper.

First, it can be said that a real plant production environment is dynamic by nature. In that aim, the first stage of our approach is to develop a discrete-event simulation (DES) model to represent dynamically the production system behavior. The model must take into account the main features inherent to the concerned plant and can thus provide a means for measuring the impact of some decisions and, in this mode, can be used as a decision making aid.

The second step is to couple the DES model with a probabilistic optimization algorithm (genetic algorithm, simulated annealing...), as shown in Figure 1. This kind of algorithms has proven to be efficient to solve large-scale combinatorial problems (for instance traveling salesman problem, VLSI layout problems, robotics problems....). Consequently, they could be extremely useful for minimizing a performance criterion of the production system evaluated by the DES model.

This methodology has been used successfully in our laboratory for two kinds of application domains: semiconductor manufacturing and fine chemistry. In both cases, a discrete-event simulation model dedicated to each field was developed and coupled with an evolutionary algorithm to determine the input order of products so as to minimize a given criterion (Peyrol et al. , 1991), (Baudet et al. 1995a), (Baudet et al., 1995b), (Baudet et al. 1995c).

This paper is focused on the semiconductor application, which is the first example treated in our laboratory. It is organized as follows. In section 2, key features of wafer manufacturing are briefly recalled. The basic principles of the discrete-event simulation model dedicated to semiconductor manufacturing (called MELISSA - Abbreviation for MicroElectronique Logiciel Industriel de Simulation et de Suivi d'Ateliers -) and developed in our laboratory will be

shortly presented. An example taking into account the main characteristics of such plants will illustrate our approach. In section 4, the principles of the Genetic Algorithm used in this study will be briefly presented and typical results obtained by use of GAs will illustrate our analysis. In the final section, we draw conclusions and discuss future work.

2. Unit operations of wafer manufacturing

In this part, only the key features of wafer manufacturing which are necessary to have a good idea of all the elements to be taken into account in the simulation are given (the interested reader will find more details in (Sze, 1983)).

First, let us say that problems involved during wafer manufacturing are very similar to those found in *batch chemical processing* although more stages are needed in VLSI. Wafer manufacturing involves a long sequence of processing steps (up to 100) as well as many equipment units (up to 50). Products move through the fab in lots, often of a constant size based on standard containers used to transport wafers. More precisely, the production of semiconductors is achieved in a multistep process involving a variety of complex chemical processes which can be divided into *unit operations*, i.e., deposition, photolithography, etching, ion implantation, photoresist strip.

Each equipment unit is associated with one of the unit operations involved in the global process. It is quite usual that an individual wafer batch visits one or more equipment units repeatedly during its course. In particular, each batch of wafers visits many times the same lithography or ion implantation work station, giving the resulting process a cyclic character. On the contrary, deposition equipment and plasma etchers are often dedicated to a single operation to prevent contamination.

The production of integrated circuits is carried out in a so-called clean-room, commonly referred to as a wafer fab. The wafer fab is typically divided into different zones, requiring different levels of cleanliness.

Since a semiconductor facility is a multipurpose plant, the various products have different flows through the workshop. The processing times are predetermined and different products may have different processing times in a same equipment unit.

Some batches may also be more important than others, which means that they get priority over all others at given stations. Batch starts often occur at regular intervals but these intervals may be changed in the time horizon to be considered.

Another particularity is that equipment units with varying capacity may be found. So, on the one hand, whenever different batches of a same class (i.e. leading to the same final product) arrive at an equipment unit superior in capacity to one batch, they are processed simultaneously (the number of batches to be regrouped is equal to the capacity of the workstation). On the other hand, whenever a wafer batch arrives at a process step inferior in capacity to one batch, the batch is split into different parts. After its traveling through this step, this batch is merged if necessary with another batch of its family.

It is also quite usual to allocate particular products to specific equipment, even if similar equipment are available at the required date. This is often the case for deposition sequences in order to avoid contamination. The wafers, are not generally subjected to any risk of contamination and yield loss and can thus be stored with no problem.

The product entrance frequency or cadence in some sets of equipment may also be different from that corresponding to the total processing time at that step. More simply, a task may be divided into subtasks, so that a product enters the first sub-task before a product has achieved its total treatment.

In addition to these operations, many loading/unloading, cleaning, measurement, inspection operations are performed throughout the fab by operators, who are generally polyvalent, i.e., neither affected to a particular equipment unit nor to a specified zone. Finally, the workshop behavior may be affected by sporadic equipment breakdown or maintenance and personnel shift.

3. Development of the MELISSA discrete-event simulation model

Discrete-event simulation has been chosen as it is a common technique for modeling manufacturing systems in a high degree of detail when the treatment of the queuing network model problem is analytically intractable. General programming languages (FORTRAN, PASCAL, C...) or simulation languages (SLAM, SIMAN, QNAP2 (Bell, 1985)...) can be used to carry out discrete-event techniques. For instance, commercial packages exist in which discrete-event simulation and dispatching rules have been married. Pritsker has developed different software tools in SLAM2 based on this coupling (the « FACTOR » package) (Pritsker, 1997) implemented in various industrial fields and, particularly, in a circuit card facility. But to our knowledge, operators are not taken into account in the model, as well as work schedules and preventive maintenance. Other computation codes based on the same principles have also been developed but are dedicated to specific applications (aeronautics, mechanics...), which render difficult their extension to

other industrial contexts. These reasons explain mainly why we have decided to develop our own code. The case studied in this paper is the result of an industrial collaboration with Motorola, Toulouse (France), which required to use FORTRAN to develop the code.

3. 1. Basic principles

This section describes the basic characteristics and major assumptions underlying the formulation of the MELISSA discrete-event simulation model (Peyrol et al. 1991), (Peyrol, 1992). The model allows to determine the exact chronology of discrete-events occurring in the facility. In our formulation, time evolution occurs by "event jump", i.e. from an event to the following one. Typical events taken into account in the simulation model and managed have been widely presented in (Peyrol, 1992).

The design of the simulation program has taken into account a complete separation between the definition of the workshop, the scheduling module and the generation of final results.

The data required to use the simulation program and the results obtained at the end of each simulation are listed in Table 1. It must be pointed out that idleness refers to the number of times when a step is available but when no wafer batch is available to be processed. The model formulation is based on the following assumptions :

- 1 - Batch size is given (it is generally a standard in microelectronics, i.e. 50 wafers per batch) ;
- 2 - Batch merging/splitting is authorized for products belonging to the same class which enter the fab at the same date ;
- 3 - The storage policy in a semiconductor facility does not impose additional constraints :first, the wafers are not submitted to degradation ; second, they are stored in small containers without generating major spatial problems. An unlimited (infinite number) intermediate storage (UIS) policy has thus been adopted in the model ;
- 4 - The operators are either polyvalent or can concentrate their activities in particular areas ;
- 5 - Priority rules, reflecting the heuristic strategy generally adopted have been used to deal with the problem of conflicts which may occur. When an equipment unit becomes free at an operation, a dispatching rule is used to order the jobs currently waiting at that operation. In our simulation study, the following dispatching rules have been selected and are applied in this order but can easily be modified:

a- When a resource becomes available, the job to do is chosen according to its priority index, reflecting its importance over other products.

b- Then, dispatching rule First In-First Out (FIFO) is selected if the first criterion is not sufficient to choose the job to be carried out : highest priority is given to the waiting operation that arrived at the queue first.

c-If the second criterion is still inefficient, the job that entered first the workshop is selected.

6 - Another source of conflicts may be encountered when a batch arrives at parallel equipment. The resource allocation is accomplished in order to maximize equipment used time. Another question also arises of which operator is to be chosen to carry out a task. This problem has been solved in balancing their workload in the time horizon to be considered.

3.2. Typical results of the discrete-event simulation of a wafer fab

The software developed has been used by two microelectronics facilities : Motorola Inc. (Toulouse, France) and Silmag (Grenoble, France). Due to the confidential nature of these industrial results, only a didactic example will be treated and analyzed. Nevertheless, this example illustrates a basic situation occurring in a wafer fab. Only some significant results provided by the model will be presented. Four products, regrouped into two classes A (products 1 and 2) and B (products 3 and 4) are to be manufactured. The batch size is assumed given (50 wafers per batch) and is held constant throughout the study. The workshop involves six processing steps for both classes (see Figure 2). These are the routes each class of products must follow :

Class A - Steps : **1 - 3 or 4 - 2 - 5 - 3 or 4 - 6**

Class B - Steps : **1 - 2 - 3 or 4 - 5 - 3 or 4 - 6**

Figure 2 exhibits the cyclic character of the flow, where each batch flows through step 3 or 4 twice. The elimination of the cycles leads to a linear processing path for both products (see Figure 3). It can be noted that the steps are renumbered (the actual number of steps is put in brackets). Let us underline that steps 2 and 5, 3 and 7, 4 and 8 are common resources.

Workshop structure, personnel allocation, product specifications are summarized in Table 2. It is important to note that processing time for step 2, i.e.15 min (respectively for step 5, i.e. 20 min) requires three (respectively four)

human interventions, after the product spent 5, 10 and 15 min at this step (respectively, spent 5, 10, 15 and 20 min at this step. It is worth emphasizing that these human interventions have been introduced in the model as masked times.

In the simulation example which will be analyzed, no equipment failure occurs and all the operators are assumed to be available during the different production campaigns. No product has a priority index over the others.

For the practitioner who must determine a viable production system, it is important to characterize system performance in the long run, after the system reaches a steady state. In this simulation run, the workshop area was started from an empty state. The simulation is then repeated several times for several batch starting up campaigns, using the results of the previous simulation run which have modified the state of the workshop. The steady state is defined as a regime in which the number of products that have not been yet completed their treatment remains constant from a run to another. These products will be referred in the following to the number of products in process (PIP).

Tables 3(a), 3(b), 3(c) and 3(d) list information given by the model after the 4th starting up campaign and Gantt chart is shown in Figure 4 for a time horizon of 440 min. Due to the important number of results provided by the model, a full list for all products would be tiresome. So, only information concerning products 1 to 8 are given in Tables 3(a) and 3(b). The waiting times of table (3a) refer to the time spent by the product out of the equipment unit, thus including the inlet and outlet times. Noteworthy is that the number of stored products, the average storage time (see Table 3b) and information concerning both steps and operators are valid for the whole time horizon.

Let us note that products class A enter in equipment item 5(2) before the previous batch leaves the same equipment item since equipment item 5(2) has been identified as a cadence step (see table 2). Step 5(2) is subdivided into three subtasks : a batch can enter equipment item 5(2) five minutes later a first batch entered this step (under the conditions that one operator is available to treat the second batch). A same reasoning is valid for products class B but this case does not explicitly appear in the example treated.

It must also be emphasized that when a product has the choice between parallel equipment, the selection is first based on the criterion of step availability. If this criterion is satisfied, then the choice is made in order to maximize step utilization time. In this example, it is clear that for the first campaign which is represented in this Gantt chart, items 3 are assigned to A while items 4 are assigned to class B. This situation may change when considering a longer horizon time, more products, breakdowns....

The wafer fab was simulated for a set of different batch start rates. The graph of Figure 5 shows the number of batches in process plotted as a function of starting up. For the simulations performed, a starting up campaign

corresponds to two products of both classes entering the fab (4 batches). The products of Class B enter the fab 20 min later than the products of Class A.

Figure 5 shows that two regions can be distinguished for batch start rates inferior to 4 batches per 110 min. In the former region, a linear increase in the number of batches in process vs. batch starting up is observed, which corresponds actually to a transient behavior of the workshop. In the latter region, the number of batches in process is constant, so that the workshop has reached steady-state. Obviously, the interactions between last batches (7,8,9,10) and the ones will follow them (11,12,... that are not represented) have been considered in the establishment of the steady-state regime. It is thus interesting to work with the maximum value of batch start rate, in this case 4 batches per 110 min, for which the workshop can run at a steady-state regime. Beyond this value, the number of batches in process increases regularly and rapidly until no more steady-state behavior can be reached. This rapid increase corresponds to the building of queues, which can be illustrated by the situation occurring at steps 3 (or 7) and 4 (or 8), which are the limiting steps of the production system for 4 batches per 90 min entering the fab. Other possible uses of the simulation, comparing system performance using different resource levels, are widely analyzed in (Peyrol, 1992).

In the approach presented in this paper, the simulation model has been used to evaluate the objective function used in the global optimization loop.

4. Jobshop scheduling by probabilistic optimization algorithms

4. 1. Background work

As mentioned in introduction, except for a few cases, scheduling of batch process operations is a difficult combinatorial optimization problem. Most scheduling problems have been shown to be NP-complete and no efficient algorithm exists for solving them optimally as the size of the problem increases. Suboptimal algorithms have been developed (Rajagolapan and Karimi, 1989) but involve major drawbacks. First, the quality of the solutions deteriorates very rapidly with an increase in problem size. Another drawback but not the least one is that two scheduling problems that seem closely related must often be solved by different algorithms. To tackle this difficult problem, use of probabilistic optimization algorithms (Simulated Annealing (Laarhoven et al., 1992) or Genetic Algorithms (Goldberg, 1989), for instance) is an attractive alternative. This kind of algorithm has been applied successfully to many combinatorial optimization problems in such diverse areas as computer aided design of integrated circuits, image

processing, design of heat exchanger networks and scheduling (Caux et al., 1994). To our knowledge, in the chemical engineering field, several batch plant scheduling problems have been solved using simulated annealing (SA) procedures (Das et al., 1990), (Ku and Karimi, 1991), (Tandom et al., 1995). Investigators have applied SA for minimizing the total time to produce a set of batches in the serial flowshop with unlimited storage (Ku and Karimi, 1991) or under various storage policies (Das et al., 1990), or for minimizing tardiness (difference between completion time of late products and their prior due dates) in a network of single-stage, unrelated parallel units (Tandom et al., 1995). These experiments showed that SA represents a good alternative over heuristic methods most widely used for this purpose, above all for real-world size problems. In the chemical engineering literature, very few papers are concerned with Genetic Algorithms. In (Cartwright and Long, 1993), a GA has been implemented to optimize both chemical feedorder and/or topology in a chemical flowshop, with minimization of the makespan of the products as objective function. In their study, GA has provided a reliable method for finding near optimum feedorder/topology combinations. In this study, we have investigated the potential of GA for solving the problem of optimization of product sequence so as to minimize a criterion based on product average residence time. Use of SA to solve the same problem has been previously treated by Peyrol in (Peyrol, 1993). The main advantage of GA over SA is that it works with a set of potential solutions instead of a single one.

4. 2. Principles of Genetic Algorithms (GAs) (Holland, 1975), (Goldberg, 1989)

Genetic Algorithms (GAs), developed mainly by Holland in the seventies (Holland, 1975), are direct random search algorithms based on the genetic processes of biological organisms. GAs rely on the collective learning process within a population, each of which represents a search point in the space of potential solutions of a given optimization problem. The population evolves towards increasingly better regions of the search space by means of randomized processes of selection, mutation and recombination.

The selection mechanism favors individuals of better objective function value to reproduce more often than worse ones when a new population is formed. Recombination allows for the mixing of parental information when this is passed to their descendants, and mutation introduces innovation in the population. Usually, the initial population is randomly initialized and the evolution process is stopped after a predefined number of iterations.

The use of a population of solutions, rather than a single solution (as with SA), is an essential and interesting feature of GAs as it speeds the search of and convergence to good solutions. Furthermore, it helps to prevent convergence of the calculation onto suboptimal solutions in complex space problems.

On the one hand, the power of GAs comes from the fact that this technique is robust and can deal successfully with a wide range of problem areas, including those which are difficult for other methods to solve which is the case in our problem. On the other hand, GAs are not guaranteed to find the global optimum solution to a problem although theoretical studies on convergence of this method are improving (Eiben et al., 1991), (Cerf,1994).

The jobshop scheduling problem, treated in this study, i.e., determining the input order of products so as to minimize the average residence time of all final products, has been identified to be NP-complete and has therefore been tackled with GAs, as it will be presented in the following. The standard flowchart of GAs is presented in Figure 6.

Before a GA can be run, a suitable *coding* for the problem must be devised. A *fitness function*, which assigns a figure of merit to each coded solution is also required. During the run, parents must be *selected* for reproduction and *recombined* to generate offspring. These aspects are described below for the jobshop scheduling problem treated in this study.

A variant of a standard GA has been implemented and the modifications brought will be presented throughout this paper.

4. 3. Application of Genetic Algorithms (GAs) to jobshop scheduling

4.3.1 Coding :

Although a lot of investigators using GAs report the use of binary-coded strings, it seems more natural in our case to code batch feedorder in permutation representation. The feedorder is coded as a string of numbers which defines the sequence in which wafer batches enter the fab. For instance, let us consider the following chromosome :

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|----|
| 9 | 8 | 4 | 5 | 6 | 7 | 1 | 3 | 2 | 10 |
|---|---|---|---|---|---|---|---|---|----|

Its content, i.e., the different genes given in this order, means that batch 9 enters the fab first, then followed by batch 8 and so on...

4.3.2 Fitness evaluation :

As previously mentioned, the evaluation function was computed using the DES model. Along with the coding scheme used, the fitness function is the most crucial aspect of any GA. In this study, the idea was to minimize the average residence time (ART) of final products in function of product input order. It has been reported in the literature, that the success of a GA computation is intimately related to the discrimination with which the algorithm can choose between solutions of varying effectiveness. In this investigation, since the fittest strings are those with the lowest average residence time of final products, it seems reasonable to relate the fitness to the inverse of this quantity. When using this fitness function, results of simulation runs have shown that the range of values thus generated is rather small, even among the better strings. In order to stretch the range of fitness, an elitism factor ($0 \leq F_{elitism} \leq 1$) has been introduced in the computation of the fitness function as suggested in (Cartwright and Long, 1993) (see the expression of fitness below). The adjustment of this value has been obtained after a series of simulations (see paragraph related to parameter settings). We think it is also important to consider the influence of products in process number (PIP), since we may have a good value for ART with an important number of products in process. The fitness function F_i of each individual i which has finally been chosen in this study takes into account both the average residence time of final products (ART_i) and the number of products in process (PIP_i) :

$$F_i = \frac{1}{((PIP_i + 1) ART_i) - F_{elitism} \text{Min}_{i=1,ni}((PIP_i + 1) ART_i)}$$

where n_i represents the total number of individuals. Let us note that in this expression, the term $(PIP_i + 1)$ and not only PIP_i has been considered to treat the case where all products would have completed all their operating sequence. Coupling of MELISSA with GA is presented in Figure 7.

Of course with this methodology, it is clear that the DES manipulates all the information related with plant performance, including time assignment and resource allocation rules. The objective proposed in this study may, in this sense, be restrictive (only time minimization is considered), since the algorithm performance is constrained by both the plant and process characteristics. To justify the validity of our approach, it can be yet argued that the heuristics

embedded in the DES model reflect the managing policies traditionally used in this industrial context. This situation is now improving in our laboratory and, in that way, perspectives will be suggested in the final section of this paper, taking into account priority/dispatching rules, the choice of events to be selected in the DES core as optimization variables.

4.3.3 Selection :

Selection consists in determining from which individuals the next population will be generated. In this study, the principle of *Goldberg's biased roulette wheel* has been adopted (Goldberg, 1989) : each current string in the population has a roulette wheel slot sized in proportion to its fitness. For each individual i , his relative fitness F_i^{rel} is calculated according the relation :

$$F_i^{rel} = \frac{F_i}{\sum_{i=1}^{ni} F_i}$$

ni represents the total number of individuals and $\sum_{i=1}^{ni} F_i^{rel} = 1$

Then, the cumulated probability of appearance of individual i denoted Q_i is determined as follows :

$$Q_i = \sum_{j=1}^i F_j^{rel} \quad (\text{with } 0 < Q_i \leq 1)$$

To select the new population, ni random choices are done in the following way. A random choice routine returns a uniform random number R between zero and one ($0 < R \leq 1$) ; the selected individual i will verify :

If $R < Q_1$ then Individual 1 is selected;

If $Q_{i-1} < R \leq Q_i$ then Individual i is selected (for $i > 1$);

Using this scheme, the fittest individuals are favored. Good individuals will be probably selected several times in a generation, poor ones may not be at all. Having selected two parents, their chromosomes are recombined using the mechanisms of crossover and mutation.

A variant of this scheme has been used in the GA implemented in this study. At the start of GA runs, it is common to have a few very good individuals in a population of bad individuals. When using Goldberg's biased roulette wheel selection, it means that these very good individuals would take over a significant proportion of the population in a generation, leading to premature local convergence of the algorithm. To prevent this, we have introduced in the

selection procedure a limitation in the number of copies of a same individual (in this study, an individual can not represent more than 50% of the whole population).

A replacement strategy has been introduced in the algorithm to create the next generation from the preceding one before crossover and mutation. When GA runs, it is possible that the best performing individual found so far is lost and replaced by worse solutions. To prevent this loss, we allow it to take part in crossover and mutation only if it will be replaced by a better individual. If not, it is directly placed into the following generation (the number of individuals after the first generation is then equal to $(n_i + 1)$).

4.3.4 Crossover :

Product input order crossover was performed using the well-known PMX (partially mapped crossover), MPX (maximal preservative crossover), OX (order crossover), LOX (linear order crossover) operators, which are well-suited for problems with permutation representation in order to prevent the generation of invalid strings. With a classical crossover operator, strings in which two batches appear twice and other batches fail to appear at all may be generated. Only the case of PMX will be presented in what follows.

Under PMX, two strings are aligned and two crossing sites are picked uniformly at random along the strings. The two crossing points (P1 and P2) define a matching section that is used to effect a cross through position-by-position exchange operations. For the sake of illustration, let us consider two strings :

| | | | | | | | | | | |
|----------|---|---|-----------|---|---|-----------|---|---|---|----|
| | | | P1 | | | P2 | | | | |
| Parent 1 | 9 | 8 | 4 | 5 | 6 | 7 | 1 | 3 | 2 | 10 |
| | | | P1 | | | P2 | | | | |
| Parent 2 | 8 | 7 | 1 | 2 | 3 | 10 | 9 | 5 | 4 | 6 |

PMX proceeds by positionwise exchanges. First, mapping parent 2 to parent 1, the 5 and the 2, the 3 and the 6, and the 10 and the 7 exchange places. Similarly mapping parent 1 to parent 2, the 5 and the 2, the 6 and the 3, and the 7 and the 10 exchange places. Following PMX, we are left with two offspring 1 and 2 where each string contains information partially determined by each of its parents :

P1 **P2**

| | | | | | | | | | | |
|-------------|---|---|---|---|---|----|---|---|---|----|
| offspring 1 | 9 | 8 | 4 | 2 | 3 | 10 | 1 | 3 | 2 | 10 |
|-------------|---|---|---|---|---|----|---|---|---|----|

| | P1 | | | P2 | | | | | | | |
|-------------|----|---|---|----|---|---|---|---|---|---|--|
| offspring 2 | 8 | 7 | 1 | 5 | 6 | 7 | 9 | 5 | 4 | 6 | |

in which duplicate batch numbers appeared in bold. The two new strings then follow another treatment : if a duplicate batch number is found in string 1, it is swapped with the batch number of string 2 in front of the first duplicate batch position of string 1 (for instance 3 in string 1 is exchanged with 6 in string 1). The same procedure applies for duplicate batch numbers of string 2. The process is continued until all duplicates have been removed. Following this process, the offspring finally obtained are :

| | P1 | | | P2 | | | | | | | |
|-------------|----|---|---|----|---|----|---|---|---|---|--|
| offspring 1 | 9 | 8 | 4 | 2 | 3 | 10 | 1 | 6 | 5 | 7 | |

| | P1 | | | P2 | | | | | | | |
|-------------|----|----|---|----|---|---|---|---|---|---|--|
| offspring 2 | 8 | 10 | 1 | 5 | 6 | 7 | 9 | 2 | 4 | 3 | |

Crossover is not applied to all pairs of individuals selected for mating. A random choice is made according to a crossover probability defined with the GA parameters.

4.3.5 Mutation :

While it is generally held that crossover is the main force leading to a thorough search of the problem space, mutation is seen as a "background" operator, responsible for introduction of some element of innovation and random search in the vicinity of the population when it has largely converged. Although mutation probability is generally low, mutation is a very important operator which must not be ignored. As with crossover, different techniques can be used for mutation (Goldberg, 1989) : (i) - permutation of the locus of two randomly chosen genes in a string (M1); (ii) - permutation of a randomly chosen gene in a string and permutation with the following gene (M2); (iii) - randomly selection of two genes and matching of the two sub-strings (M3); transport procedure (M4). Mutation M3 is illustrated as follows :

P1

| | | | | | | | | | | |
|-----------------|---|---|---|---|---|----|---|---|---|----|
| before mutation | 9 | 8 | 4 | 2 | 3 | 10 | 1 | 6 | 5 | 7 |
| after mutation | 1 | 6 | 5 | 7 | 9 | 8 | 4 | 2 | 3 | 10 |

4.3.6 Parameter settings:

The determination of adequate values for the variable parameters required by GAs, such as the population size as well as crossover and mutation probabilities is very important. Convergence of the algorithm is dependent on a suitable choice of these parameters which are related to the complexity of the problem. Tests of performance of the algorithm on the treated example lead us to use the following parameters presented in Table 4.

We tested the algorithm on two scheduling problems of different sizes : a "BIG" example which corresponds to a didactic case (taking into account more products and steps than commonly reported in the literature) which will be analyzed in detail and a "GIANT" example which can be considered of industrial size. We used the same parameter setting for tests on the two problems.

4.3.7 Simulation results for a "BIG" example:

In the simulation example which is presented, 16 products grouped in 6 classes (see Table 5), are to be manufactured during a production campaign of 1500 min. The workshop involves 9 processing steps, with some parallel units (4 and 5, 7 and 8). Steps 4 and 5 can process two batches simultaneously, and the capacity of steps 7 and 8 is one half batch. Steps 2 and 3 are cadence steps with a product input frequency of 20 min.

Concerning the steps with a two-batch capacity, we have considered that only batches belonging to the same class but with the same input date in the plant may be merged at this step. A sequence $\{k_1, k_2, \dots, k_n\}$ where k_i refers to two products of a same class with the same input date, means that k_i is introduced in the i^{th} position in the plant. The total number of possible sequences n_s is given by :

$$n_s = \frac{n!}{n_{clas} \prod_{j=1} (ng_j)!}$$

where :

n represents the number of products with a given input date, ng_j is equal to the number of k_j of class j and n_{clas} to the number of product classes.

The processing sequences of each class of products are the following ones :

Class 1: steps 1, 2, 3, 4 or 5, 6, 7 or 8, 9.

Class 2: steps 1, 4 or 5, 2, 3, 4 or 5, 6, 7 or 8, 9.

Class 3: steps 1, 2, 3, 4 or 5, 2, 3, 4 or 5, 7, 8 or 9.

Class 4: steps 1, 2, 3, 4 or 5, 2, 3, 4 or 5, 6, 7 or 8, 9.

Class 5: steps 1, 2, 3, 4 or 5, 7 or 8, 9.

Class 6: steps 1, 4 or 5, 2, 4 or 5, 7 or 8, 9

After elimination of cycles as explained in Section 3.2, the processing sequences become :

Class 1: Steps 1, 2, 3, 4 or 5, 10, 11 or 12, 13.

Class 2: Steps 1, 4 or 5, 6, 7, 8 or 9, 10, 11 or 12, 13.

Class 3: Steps 1, 2, 3, 4 or 5, 6, 7, 8 or 9, 11 or 12, 13.

Class 4: Steps 1, 2, 3, 4 or 5, 6, 7, 8 or 9, 10, 11 or 12, 13.

Class 5: Steps 1, 2, 3, 8 or 9, 11 or 12, 13.

Class 6: Steps 1, 4 or 5, 6, 8 or 9, 11 or 12, 13.

The operating times of each class of products are presented in Table 6.

A series of experiments was performed to find the best product sequence for various mutation and crossover strategies. Simulation results obtained are presented in Table 7. The percentage deviation of the best fitness dev is the deviation between the true optimal solution obtained by a whole enumerative exploration of the solution space and the best solution of a given combination of mutation and crossover operators. N_{TOT} represents the total number of explored configurations and N_{BEST} the number of times that the optimum value is reached. The average value of a given simulation and the standard deviation are also presented. At first, when we note that the search space contains $8!/4 =$

10080 possible sequences, 5 times greater than the number of evaluations. Second, it is difficult to assess that a strategy overcomes clearly the others, since the performance of a GA depends upon several parameters.

Nevertheless, several comments can be made which can be useful for guiding the choice of the crossover operator for a scheduling problem. If we look at Figure 8 showing generation average results for all the crossover strategies tested in this study and for a given mutation operator, MPX and PMX are the most conservative operators, with a "no jump" evolution. This property can lead them to reach in some cases a local optimum. On the contrary, OX appears to be the most innovative operator. Without going into further details concerning the presentation of this operator, let us say that the offspring generated by OX are strongly different from their parents even with two identical parents (this is not the case with the other operators, two identical parents will give two identical offspring). If the optimum is obtained with OX, it is generally due to the important number of explored configurations (approximately 3000 combinations with OX instead of 2000 with the other crossover policies). In this study, LOX appears to be the most performing for various mutation policies (with respect to both number of explored configurations and best solution). The results of two numerical experiments obtained with LOX are presented in Figure 9, the best-of- (and worst-of) generation and generation average results respectively. Figure 9 shows how the best individuals of each generation converged to a near-optimal solution after about 40 generations (approximately 1000 fitness evaluations). The best solution is obtained from the following sequence : 2 5 1 5 4 6 3 6 similar to that obtained by using a simulated annealing procedure (SA) (Peyrol et al., 1993).

Experiments have also been performed for different mutation operators. Typical average generation results obtained with MPX are presented in Figure 10. Unlike crossover, a "destructive" mutation operator may be useful to come out local optima, if combined with a low mutation probability to preserve the general evolution.

In terms of computational effort as measured by CPU time on a monoprocessor IBM RS6000, the coupling of DES/GA with LOX, PMX, MPX operators required approximately 100 min CPU for a 16 product, 13 step problem. Reports concerning CPU time allocation indicated clearly that the bulk of the time was spent on the simulation of the workshop by MELISSA. The high CPU times are thus a strong reflection of the computational complexity of the simulation model: it is our choice to develop a detailed DES model as explained in the former sections even if it may penalize from an optimization computational effort. Let us note that this work must be considered as a feasibility study, leaving space for improvement. In order to reduce the DES calls in the program and thus to make Genetic Algorithms more competitive with DES from a computational effort point of view, it could be interesting to reduce the search space

by using a learning procedure for parameter setting like crossover/mutation policies and probabilities. We are currently investigating this issue.

4.3.8 Simulation results for a "GIANT" example:

Test performance of the algorithm on the "BIG" example leads us to use the same parameters with LOX crossover and M1 mutation policies. In this "GIANT" example, 48 products (24 groups of 2 batches) are to be manufactured in a workshop involving 70 operating steps. A production campaign equal to a week has been considered. The evolution of the average fitness in a population and the fitness of the best string of the population with generation number is presented in Figure 11. The efficiency of the GA is much more apparent in the case of the GIANT problem. In this case, the total number of possible sequences n_s is equal to $24! = 6.2 \cdot 10^{23}$. In this example, the total number of configurations explored is equal to 5395 and the genetic algorithm displays its force as a search operator. The best individuals of each generation converged to a near-optimal solution after about 100 generations, i.e., approximately 3600 fitness evaluations, as shown in Figure 12. As far as the processing time is concerned, the 100 generations of the "GIANT" problem took approximately 6000 min CPU on the same machine. Although this time is relatively high, such a study can be very useful for a decision's maker who has to cope with such combinatorial problems and has to decide how to group the products to be manufactured. In this example, the maximum discrepancy between an average fitness and the best fitness found is equivalent to approximately 24 hr, which represents an appreciable gain relative to the production campaign, i.e., a week. The same remark as for the BIG example is valid : the high CPU times are determined not only by the problem size but above all by the computational complexity of the DES model.

5. Conclusions and perspectives

In this paper, a two-stage methodology for solving industrial-size scheduling problems has been presented : development of a discrete-event simulation model dedicated to an industrial domain, i.e. semiconductor manufacturing and its coupling with a probabilistic optimization algorithm, i.e. Genetic Algorithms. Our numerical experiments showed that the GA developed represents a good alternative for treating large combinatorial problems.

This approach has been applied to two case studies corresponding to a "BIG" example and a "GIANT" one. It has yielded in both cases to very good solutions, reducing considerably the search space. The search space is further reduced as the size of the problem increases. At that stage of our work, it must be argued that the computational effort needed for "GIANT" problems may be prohibitive. Concerning this point, reports concerning CPU time allocation have indicated clearly that the bulk of the time was spent on the simulation of the workshop by the DES model. This work must be considered as a feasibility study on GA coupled with DES : solutions have already been suggested and tested in some cases to reduce the number of DES calls in the program by a guided choice of GA parameters via a learning procedure of these parameters. They will be soon reported.

The strength of probabilistic optimization algorithms, and consequently of GA, lies in the fact that it can be coupled very easily with any jobshop discrete-event simulation model with minor modifications, in contrast to some heuristic methods which are more suitable for certain types of problems than other. The results already obtained have encouraged us to generalize the approach to other application fields, e.g. fine chemistry which presents more complexity due to the presence of unstable intermediates and the different storage policies involved (Finite Intermediate Storage, Zero Wait, Mixed Intermediate Storage). A DES model dedicated to fine chemistry applications is under development. Concerning the GA procedure, it has also been extended to take into account more decision variables (i.e., the choice of priority/dispatching rules, the selection of an event to be executed in the simulation core, batch sizing), thus to increase the solution space.

Abbreviations

| | | |
|------|---|--|
| ART | : | average residence time of final products |
| BAB | : | branch and bound |
| DES | : | discrete-event simulation |
| GA | : | genetic algorithm |
| LOX | : | linear order crossover |
| MILP | : | mixed integer linear programming |
| MPX | : | maximal preservative crossover |
| OX | : | order crossover |
| PIP | : | number of products in process |
| PMX | : | partially mapped crossover |
| SA | : | simulated annealing |

References

- Baudet P. , Azzaro-Pantel C., Domenech S., Pibouleau L., A discrete-event simulation approach for scheduling batch processes, *Computer Chem. Engng.*, Suppl. Vol. **19**, S633-S638 (1995).
- Baudet P. , Azzaro-Pantel C., Domenech S., Pibouleau L., Simulation d'ateliers discontinus de chimie fine, *Proceedings of the "5ème Congrès de Génie des Procédés*, Lyon, 19-21, **40**, 117-122 (1995)
- Baudet P. , Azzaro-Pantel C., Domenech S., Pibouleau L., A discrete-event simulation model for batch chemical plant scheduling, *ADEDOPS Workshop (workshop on analysis and design of event-driven operations in process systems)*, Imperial College, Londres, 10-11 avril 1995.
- Bell G., Dubois D. Modélisation et simulation des systèmes automatisés, *RAIRO APII*, Vol. 19 (1985).
- Birewar D.B., I.E. Grossman, Simultaneous production planning and scheduling in multiproduct batch plants, *Ind. Eng. Chem. Res.*, **29**, 570 (1990).
- Cartwright H.M., R.A. Long, Simultaneous Optimization of Chemical Flowshop Sequencing and Topology Using Genetic Algorithms, *Ind. Eng. Chem. Res.*, **32**, 2706 (1993).
- Caux C., Pierreval H., Portmann M.C., Les algorithmes génétiques et leur application aux problèmes d'ordonnancement, *Proceedings Journées d'études « ordonnancement et entreprise »*, Toulouse (1994).
- Cerf R., Une théorie asymptotique pour les algorithmes génétiques, *PhD Thesis*, Montpellier, France (1994).
- Das H., P.T. Cummings, M.D. LeVan, Scheduling of serial multiproduct batch processes via simulated annealing, *Comput. Chem. Engng*, Vol. 14, 1351-1362 (1990).
- Egli U.M., Rippin D. W. T., Short term scheduling for multiproduct batch chemical plants, *Comput. Chem. Engng*, **10**, 303 (1986).
- Eiben A.E., Aarts E. H. L., Van Lee K.M., Global convergence of genetics algorithms : a markov chain analysis, *Proceedings of the first workshop Ppsni*, Dortmund (1991).
- Espuna A., Puigjaner L., On the solution of the retrofitting problem for multiproduct batch semi-continuous chemical plants, *Comput. Chem. Engng*, 13, 483 (1989).
- Goldberg D.E., Genetic algorithms in search, optimization and machine learning, *Addison Wesley*, MA (1989).

- Grossman I.E., R.W.H. Sargent, Optimal design of multipurpose chemical plants, *Ind. Eng. Chem. Process. Des. Develop.*, **18**, 343 (1979).
- Holland J.H., *Adaptation in Natural and Artificial Systems*; University of Michigan Press, Ann Arbor, MI (1975).
- Ignall E., Schrage L., *Operations Research*, **13**, 400 (1965).
- Karimi I.A., Reklaitis G.V., Intermediate storage in continuous processes involving stages of parallel units, *AIChE Journal*, **31**, 44 (1985).
- Ku H., Karimi I.A., Scheduling in serial multiproduct batch processes with finite intermediate storage : A mixed integer linear program formulation, *Ind. Eng. Chem. Res.*, **27**, 1840 (1988).
- Ku H., Karimi I.A., Scheduling in serial multiproduct batch processes with due-date penalties, *Ind. Eng. Chem. Res.*, **29**, 580 (1990).
- Ku H., Karimi I.A., An evaluation of simulated annealing for batch process scheduling, *Ind. Eng. Chem. Res.*, **30**, 163 (1991).
- Laarhoven P.J.M., E.H.L. Aarts, j.K. Lenstra, Job Shop Scheduling by simulated annealing, *Operations Research* , **40**, 113 (1992).
- Lazaro M.A., A. Espuna A., L. Puigjaner, A comprehensive approach to production planning in multipurpose batch plants, *Comput. Chem. Engng*, **13**, 1031 (1989).
- Mauderli A.M., D.W.T. Rippin, Production planning and scheduling for multi-purpose batch chemical plants, *Comput. Chem. Engng.*, **3**, 199 (1979).
- Musier R., Evans L., Batch process management, *Chem. Eng. Prog.*, **4**, 37, (1980).
- Peyrol E., L. Pibouleau , J.P. Couderc, S. Domenech, Semiconductor circuit fabrication plant management by discrete simulation, *Simulation of semiconductor devices and processes*, Edited by W. Fichtner, D. Aemmer, September 12-14 (1991).
- Peyrol E., Gestion d'un atelier de fabrication de composants électroniques, *PHD Thesis*, INPT -15 dec. (1992).
- Peyrol E., P. Floquet, L. Pibouleau , S. Domenech, Scheduling and simulated annealing. Application to a semiconductor circuit fabrication plant, *Comput. Chem. Engng*, **17**, Suppl. p. S39-44 (1993).
- Pritsker A. A. B., Introduction to simulation and SLAM II, 4th Edition, *Systems Publishing Company* (1997).
- Rajagolapan D., I.A. Karimi, Completion times in a serial mixed-storage multiproduct processes with transfer and set-up times, *Comput. Chem. Engng*, **13**, 175 (1989)

- Reklaitis G.V., Overview on scheduling and planning of process operations, *NATO Advanced Study Institute on Batch Processing Systems EngineerIng*, Antalya, Turkey, (1992).
- Rippin D.W.T., Simulation of single- and multiproduct batch chemical plants for optimal design and operation, *Comput. Chem. Engng.*, **7**, 137 (1983).
- Rippin D.W.T., Design and operation of multiproduct and multipurpose batch chemical plants, *Comput. Chem. Engng.*, **7**, 463 (1983).
- Rippin D.W.T., Batch process systems engineering : a retrospective and prospective review, *Comput. Chem. Engng*, *17*, Suppl. S1-S536 (1993).
- Sze S.M., V.L.S. I. Technology, *Mc Graw Hill*, Ed. New York (1983).
- Tandom M., P.T. Cummings, M.D. Le Van, Scheduling of multiple products on parallel units with tardiness penalties using simulated annealing, *Comput. Chem. Engng*, **19**, 10, 1069-1076 (1995).
- Wellons M.C., Reklaitis G.V., Scheduling of multipurpose batch chemical plants-Formation of single-product campaigns, *Ind. Eng.Chem. Res.*, **30**, 671 (1991)
- Wellons M.C., Reklaitis G.V., Scheduling of multipurpose batch chemical plants-multiple-product campaign formation and production planning, *Ind. Eng.Chem. Res.*, **30**, 688 (1991)
- Yamalidou E.C., E.P. Patsidou, J.C. Kantor, Modeling discrete-event dynamical systems for chemical process control-a survey of several new techniques, *Comput. Chem. Engng*, **14**, 281 (1990).