A GENETIC ALGORITHM FOR BATCH CHEMICAL PLANT SCHEDULING

P. Baudet, C. Azzaro, L. Pibouleau et S. Domenech

Laboratoire de Génie Chimique - UMR CNRS 5503

ENSIGC INPT
18, Chemin de la Loge
F-31078-Toulouse Cedex - FRANCE
Tél : 62 25 23 00
Fax : 61 25 23 18
email : Catherine.Azzaro@ensigct.fr

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 (D.E.S.) model to represent dynamically the production system behaviour, taking into account the main features inherent to a batch chemical plant. In the second step of our approach, we investigate Genetic Algorithms (GAs) for solving batch process scheduling problems : a GA has been developed for minimising the average residence time to produce a set of batches in function of batch order in a multipurpose-multiproduct batch plant. The evaluation of the objective function is provided by its coupling with the D.E.S. model embedded in the optimisation loop. Computational results show that the use of this approach can significantly help improving the efficiency of the production system.

I. Introduction

Batch processes are the prevalent mode of manufacture for speciality 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 optimise a suitable objective.

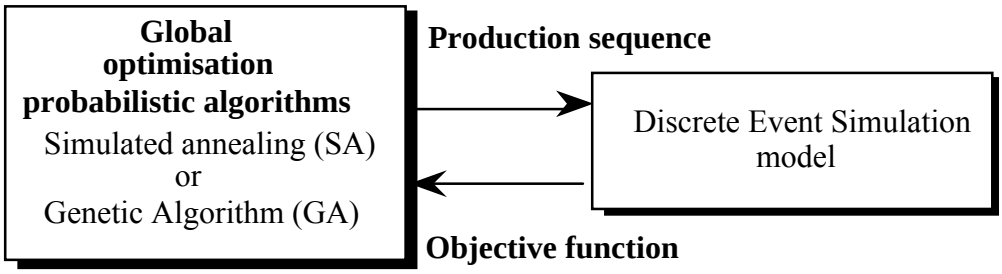
A typical feature of most scheduling problems is that determination of the production sequence is a combinatorial optimisation problem in complex plant configurations : the number of candidate solutions grows exponentially with the problem size. Most flowshop and jobshop problems have been shown to be NP-complete problems. Efficient algorithms for obtaining an optimal solution exist only for very special cases of the simplified flowshop. A 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). For larger problems, several heuristic

algorithms have also been proposed based on the MILP formulation for obtaining good, but suboptimal, solutions [KuK88]. Another approach is to use the Branch and Bound (BAB) strategy. But as reported in [KuK88], 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 and 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 (D.E.S.) model to represent dynamically the production system behaviour. 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 D.E.S. model with a probabilistic optimisation algorithm (genetic algorithm, simulated annealing...). This kind of algorithms has proven to be efficient to solve large-scale combinatorial problems and could be extremely useful for minimising a performance criterion of the production system evaluated by the D.E.S. model.



This paper is organised as follows. In section 2, basic principles of the discrete event simulation model dedicated to batch chemical plants will be shortly presented. In section 3, we will describe the principles of the Genetic Algorithm used in this study and typical results obtained by use of GAs will illustrate our analysis. In the final section, we draw conclusions and discuss future work.

II Discrete Event Simulation Model

The objective of the D.E.S. model is to determine the sequence in which the products should be manufactured and the time when the various production operations for each product should be carried out while satisfying a variety of constraints concerning labor, utilities .... To develop our model, main common resources (equipment items, buffer stocks for specific intermediates, operators...) have been described as finite state automata and all the allowed events have been characterized by their occurrence conditions and by the consequences of their occurrence, especially logical changes on finite state automata and generation of scheduled events. The list of scheduled events is kept by increasing occurrence date on a simulation module, so that when no more event is possible at a time, we look for the next one in the scheduled list. Then the model progresses iteratively from an event to another and the time progression jumps from an occurrence date to the following one till the end of the simulation. In this work, the workshop is divided into different production and/or storage areas containing equipment items and buffer stocks for specific intermediates.

II.1 Processing network and resource restrictions

Equipment items (EI) are modeled as finite state automata; the different logical states of an EI are described in figure 1. With each EI are associated fixed attributes, i.e. production area, maximum capacity, temperature and pressure. In the current version of the model, failure event is considered as a preestablished simple event in a specific calendar. Further studies should allow the model to treat this event differently using a set of data about EI breakdowns (for instance mean time between failure...). Maintenance operations on EI are provided in a specific calendar where the date, duration and number of required operators are registered.

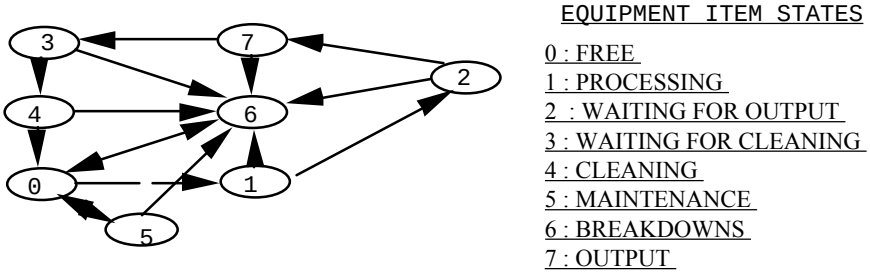


Figure 1: Equipment item finite state automaton

Buffer stocks (BS) for specific intermediates are also defined as finite state automata, as it can be seen in figure 2. Fixed attributes associated with BS are their maximum capacity and type which may be depending on their construction material (glass, steal...). BS maintenance operations are defined as those concerning equipment items. Cleaning operation of BS is defined by its duration and number of required operators to carry it out. There are two key renewable shared resources allowed in the model formulation : operators and utilities. The finite state automaton modeling human resource can be seen in figure 3. The operators are characterized by their polyvalence in one or more areas. An operator declared polyvalent in an area can execute any task in it. Main activities are relative to equipment items (product input/output, cleaning or maintenance). All other operator activities will be referred as secondary activities. Let us note that like EI breakdowns, operator work is often neglected in scheduling studies, although it is often in practice a scarce resource of the system.

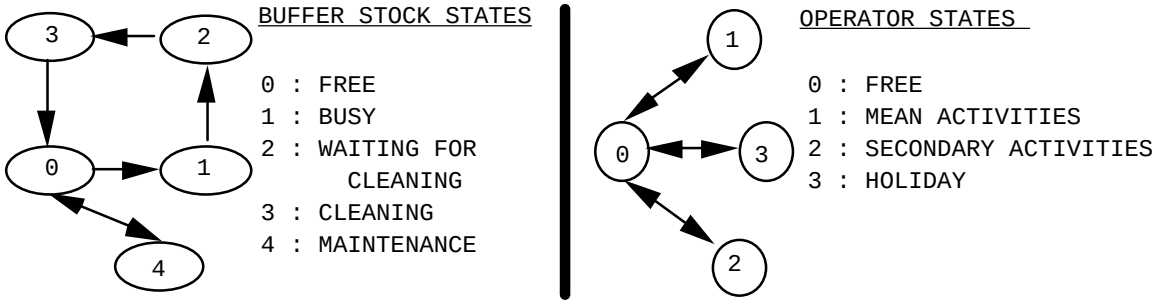


Figure 2 and 3 : Buffer stock and operator finite state automata

Utilities (or shared resources) are renewable resources used as discrete amounts by all the equipment items of the workshop without area consideration (electricity, vapor, hot oil...).

Raw materials (RM) are consumed during the required production elaboration. They are stored in specific storage units assumed to be of unlimited capacity. When a raw material charge is made to a unit, the inventory of that material is reduced and can only be restored by delivery of new

material. Note that RM supply is considered without spatial constraints and is achieved according to a preestablished calendar.

II.2 Products

Final products (FP) have to be manufactured in required quantities often according to delivery dates in order to satisfy customers. FP are also stored in specific storage units which are supposed to be of unlimited capacity.

An intermediate product (IP) is the main product leaving a production stage which is totally consumed during a further operation leading to final products. An IP may be declared stable, unstable or stable during a limited time. In the unstable case, the next processing stage must follow immediately. In the limited stability case, it can be deferred if necessary until appropriate equipment is available whereas in the stable case, the IP can be stored in an appropriate stock. A stable IP may also generate spatial storage constraints but no time storage constraints. It may have a specified storage unit type and is available as soon as it is needed for further operations.

A shared intermediate product (S I P) has a special treatment. It can be a stable intermediate product which is involved during the elaboration of several different final products and is then elaborated separately. It can also be a product which for one part is directly valorized and, for the other, consumed during the elaboration of a higher-added value product. A SIP is then both a final product with its own recipe and its own required quantity to be produced during the campaign, and a raw material consumed during the elaboration of other products. As for raw materials, SIP supply is allowed according to a preestablished calendar. In any case, SIPs are necessarily stable and can be stored in specific storage units of unlimited capacity.

Two kinds of subproducts are taken into account in the model. On the one hand, recycled products (RP) with their own recipe may be treated in situ during the production campaign. On the other hand, the second category of subproducts is not recycled in situ but is stored before being valorized elsewhere. They will be referred in the following part of this article as non recycled in situ products (NRP). Both subproduct types are stable and generate only spatial storage constraints.

Considering the previously described elements, a share of all resources is achieved between three product types during the production campaign : FPs, SIPs, and RPs.

II.3 Recipe and batch definition

A recipe is the adequate term to represent the sequence of operations leading to the elaboration of a required product. To define a recipe, a reference volume which represents the mean product volume to be elaborated or to be recycled has been taken into account. Reactional volumes (RV) of all the operations of the recipe are defined in percentage of this reference volume. In this study, each operation of the recipe is defined as follows:

- Identification of operation type : an operation is said parallel if more than one equipment item is able to process it. It is referred as a multiple input step if more than one intermediate may enter the operation and as a multiple output step if more than one product may leave the processing step.
- Identification of feasible equipment items for each task.
- Definition of operating conditions (pressure, temperature).
- Definition of input products, i.e. identification of RMs, SIPs, or IPs entering an operation step and definition of their volume in percentage of RV.
- Definition of output products, i.e. identification of all products leaving an operation and definition of their volume in percentage of RV. If the operation is not the last one of the recipe, the operation in which the IP will be consumed is then defined. The precedence constraints between the operations of the recipe are so taken into account.

- Different times have to be defined to complete the recipe description (input, output, processing and cleaning times). Associated with these durations is defined the number of required operators, except for process durations.

- Amounts of utilities needed during the processing time have also to be declared for each operation and for each parallel EI of the recipe. The different durations and amounts of needed utilities may be constant or a function of the operation capacity, temperature and/or pressure.

For all the FP, SIP and RP products of the workshop, recipes are defined in the same manner.

The total required production is then converted into a list of batches to be processed, including the production of any needed intermediate. Associated with each batch is a product recipe and a batch size. Of course, several batches of the same product may be required during a production campaign, with different batch sizes. A recycling mode is also needed, either maximum (i.e. recycling of a maximum number of RP batches) or minimum (i.e. maximum storage of RP according to limited storage capacities). In any case, for each RP, we have to define the reference volume of a recycling batch, i.e. the volume of the RP recycled by each batch.

Finally, before computation, the simulation programm achieves an entire treatment of the input data. It implies a global material balance evaluation, first to check if the required production is possible (test about RM or SIP sufficiency for instance) and second, to compute according to the recycling mode, the number of RP batches which will have to be processed during the production campaign. It calculates also the real volume values (from the given reference volume) of all batch operations and determines time and utility amounts functions. Finally, it checks the validity of the EI to process the operation. If a calculated reactional volume is greater than the EI capacity, or if reactional pressure or temperature are greater than the EI admissible maxima, the EI is removed from the batch recipe. If all EI have inadequate capacities, the step is detected and will be made in a multiple pass operation during the computation.

II.4 Simulation Runs and use of heuristics

Under the simulation motor control, using the entire set of previously defined data, the system evolves from an event to another and time from an occurrence date to the following one. However, two problems may occur during the simulation run and have a great impact on the final production performance of the workshop. The former concerns transfer rules. Which EI has to be chosen to process an operation when more than one are available ? What product has to be processed first when more than one are in the free EI waiting queue (note that these two questions may arise simultaneously) ? What buffer stocks have to be chosen between the free ones to store a given intermediate ? Which operation must have the priority for operator assignment and on the contrary, how do we choose an operator to execute a task when more than one are free ? To solve these questions, classical heuristic rules have been used (for instance, FIFO, SIPT, EDD rules are available). It becomes clear that the execution order of several conditional possible events at a given time may have a great impact. The occurrence of an event, associated with the allocation of its needed resources and logical changes (on the concerned finite state automata) may indeed suppress the occurrence possibiblity of some others. In this work, the execution order of conditional events whose occurrence conditions are satisfied has been chosen using the following rules : maintenance operations are first executed because they are supposed to be important to avoid breakdowns. Next, events concerning EI have the priority (input and output of products and after, cleaning events). Finally, the program executes the remaining events (i.e. concerning buffer stocks).

II.5. Simulation results

A lot of results are provided by the model to represent the production system behavior during the campaign. Completion times, total cycle times and total waiting times are available for the required batches. The total waiting time of a batch includes the total time elapsed in the EI waiting queues and the synchronisation time for the multiple output operations. For a multiple output operation, the synchronisation time is the time between the elaboration date of the first and the last entering IP. Each batch elaboration can be studied more precisely from initial to final stage. For each elaboration stage of the batch, the following results are available: concerning input stage description, the input date with the required operators, the chosen EI (if the operation is a parallel one) and the selected buffer stocks for the leaving IP (if the operation is not the last of the recipe) are reported. Concerning output description, the effective output date with the required operators and information about an effective storage of the leaving IP or its immediate utilization for the following stage are presented. Statistics provide general information about finite state automata activities (utilization rate of EI, activity rate of operators, e.t.c...). A Gantt chart offers a global view of the system behavior during the campaign presenting in the same graph EI and products. The level of products in process can also be visualized. For our optimisation problem, we will only use a chosen technical result provided by the model, i.e mean cycle time of products.

III. Jobshop scheduling by probabilistic optimisation algorithms

As mentioned in introduction, except for a few cases, scheduling of batch process operations is a difficult combinatorial optimisation 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 [Raj89] 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 optimisation algorithms (Simulated Annealing [Laa92] or Genetic Algorithms [Gol89], for instance) is an attractive alternative. This kind of algorithm has been applied successfully to many combinatorial optimisation problems in such diverse areas as computer aided design of integrated circuits, image processing, design of heat exchanger network and scheduling. 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 this study, we have investigated the potential of GA for solving the problem of optimisation of product sequence so as to minimise a criterion based on product average residence time. The main advantage of GA over SA is that it works with a set of potential solutions instead of a single one.

III.1. Principles of Genetic Algorithms (GAs) [Hol75], [Gol89]

Genetic Algorithms (GAs), developed mainly by Holland in the seventies [Hol75], 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 optimisation problem. The population evolves towards increasingly better regions of the search space by means of randomised processes of selection, mutation and recombination.

The selection mechanism favours 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 initialised 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.

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. GAs are not guaranteed to find the global optimum solution to a problem but they are generally good at finding "acceptably good" solutions to problems "acceptably quickly".

The jobshop scheduling problem, treated in this study, i.e., determining the input order of products so as to minimise 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 below :

Step 0 : String encoding and setting of parameters : number of individuals, mutation probability, crossover probability and maximum number of generations
Step 1 : $t \leftarrow 0$
Creation of the initial randomly-generated population of n individuals
Step 2 : Calculation of the evaluation function of each individual
Step 3 : Selection of n individuals
Step 4 : Recombination
Crossover
Mutation
Step 5 : $t \leftarrow t + 1$
Update the generation and return to step 2 until the maximum number of generations is reached

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.

III.2. Application of Genetic Algorithms (GAs) to jobshop scheduling

III.2.1 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.

III.2.2 Fitness evaluation :

As previously mentioned, the evaluation function was computed using the D.E.S. 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 minimise 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, a value equal to 90% of the minimum average residence time calculated thus far is subtracted from ART. The adjustment of this value has been obtained after a series of simulations. 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) :

$$Fi : 1/\sum_{i=1}^{N} ARTi$$

III.2.3 Selection :

Selection consists in determining from which individuals the next population will be generated. In this study, the principle of *Goldberg's biased wheel* has been adopted [Gol89] : 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 $\sum_{i=1}^{ni} F_i^{rel} = 1$

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 ($0 \leq R \leq 1$); the selected individual j will verify :

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

Using this scheme, the fittest individuals are favoured. 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, it is

directly placed into the following generation (the number of individuals after the first generation is then equal to (ni + 1) and can also take part to crossover and mutation .

III.2.4 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 offsprings 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 offsprings 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.

III.2.5 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 [Gol89] : (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

III.3 Parameter of GAs

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 1 :

GA parameter	Value
Population size	30
Number of generations	500
Crossover probability	0.8
Mutation probability	0.1

Table 1 : GA parameters used in this study

IV An optimisation example

Our approach was used to determine the release order of 30 different products, each with its own recipe. The production area contains 22 equipment items and production recipes involve almost 6 successive operations. The production campaign includes with many constraints of raw materials deliveries or maintenance operations of equipment items. Population size of the GA has been set up at 30 and the number of generations is limited to 500 (i.e. 15000 evaluation procedures of the fitness using the discrete event simulation model between 30! ($2{,}6\ 10^{32}$) feasible solutions. MPX crossover and mutation M3 have been chosen as genetic operators and the GA parameters are described in table 1.

The best solution is obtained after 427 generations and leads to a mean cycle time of 17887 time units. The required production is achieved at the end of the production campaign. A simulation run, without GA coupling leads to a mean cycle time of 21350 time units (the release order was determined with an heuristic (SIPT)) and the production is not achieved because 2 products are still in process in the production area at the end of the production campaign. On this example, we show that for a little number of evaluations, Genetic Algorithm leads to a good solution of the scheduling problem.

V Conclusions and perspectives

In this paper, the feasibility of a two-stage methodology for solving scheduling problems has been presented : development of a discrete-event simulation model dedicated to batch chemical domain and its coupling with a probabilistic optimisation algorithm, i.e. Genetic Algorithm. Our numerical experiments showed that the GA developed represents a good alternative over other techniques (MILP, BAB) which are inefficient for large combinatorial problems and gives better results than SA.

GAs use to solve our optimisation problem has yielded to very good solutions, reducing considerably the search space.

The strength of probabilistic optimisation 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.

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
SIPT	:	Shortest Imminent Processing Times

Literature cited

[[Bir90] Birewar D.B., I.E. Grossman, Simultaneous production planning and scheduling in multiproduct batch plants, Ind. Eng. Chem. Res., 29, 570 (1990).

[Gol89] Goldberg D.E., Genetic algorithms in search, optimization and machine learning, Addison Wesley, MA (1989)

[[Hol75]Holland J.H., Adaptation in Natural and Artificial Systems; University of Michigan Press, Ann Arbor, MI (1975)

[KuK88]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)

[Laa92]Laarhoven P.J.M., E.H.L. Aarts, j.K. Lenstra, Job Shop Scheduling by simulated annealing, Operations Research , 40, 113 (1992).

[Raj89]Rajagolapan D., I.A. Karimi, Completion times in a serial mixed-storage multiproduct processes with transfer and set-up times, Comput. Chem. Engng, Vol. 13, p. 175 (1989)