F

Flexible Job-Shop Scheduling Problems

Imed Kacem
Université de Technologie de Troyes, France

INTRODUCTION

Planning and scheduling problems in various industrial environments are combinatorial and very difficult. Generally, it is extremely hard to solve these types of problems in their general form. Scheduling can be formulated as a problem of determining the best sequence to execute a set of tasks on a set of resources, respecting specific constraints like precedence or disjunctive constraints (Carlier & Chrétienne, 1988). They consist generally in a simultaneous optimization of a set of non-homogeneous and conflicting goals. Therefore, the exact algorithms such as branch and bound, dynamic programming, and linear programming are not suitable for such problems and need a lot of time and memory to converge.

Because of this difficulty, experts prefer to find not necessary the optimal solution, but a good one to solve the problem. To this end, new search techniques such as genetic algorithms (Dasgupta & Michalewicz, 1997; Sarker, Abbas & Newton, 2001), simulated annealing (Kirkpatrick, Gelatt & Vecchi, 1983), and tabu search (Golver, Taillard & De Werra, 1993) are proposed to reach this aim: construct an approximated solution for a large set of hard optimization problems.

In this article, we are interested in the evolutionary techniques and their application to an important branch of scheduling problems. We aim in particular to present an overview of the recent models proposed to solve flexible job shop scheduling problems using genetic algorithms.

FLEXIBLE JOB SHOP SCHEDULING PROBLEM

The Flexible Job Shop Scheduling Problem is an extension of the classical job shop problem. Indeed, it represents two classical combinatorial optimization problems. The first one is a linear assignment since each task can be carried out on a set of resources according to different processing times. The second problem consists of finding the best sequence of the tasks on the resources by respecting all the problem's constraints (interdiction of pre-emptive execution, disjunctive constraints on the resources, precedence constraints) (Dauzère-Pérès, Roux & Lasserre, 1998; Mastrolilli, 2000).

The structure of the flexible job shop scheduling problem can be described as follows. We have a set of jobs and a set of machines. Each job consists of a certain number of ordered operations. The execution of each operation requires one machine selected from the machines existing in the shop. Therefore, we should find the best assignment of the machines before scheduling. The sequencing of operations is fixed for each job (precedence constraints). All machines are initially available and each job has a release date. For a given machine and a given operation, a processing time is defined as the duration necessary to perform the task on the above machine. The pre-emption condition is not allowed, and each machine can perform operations one after the other (disjunctive resource constraints).

MODELLING FLEXIBLE JOB SHOP

The encoding problem is a main step in the genetic algorithm conception. Such a problem has been intensively studied in the literature (Dasgupta & Michalewicz, 1997). In a simple way, solutions can be represented by binary vectors. Such a model is a judicious method for the problems in which solutions can be reduced to a sequence of zeros and ones. Unfortunately, we are not able to use such a representation for solving real-world problems like planning and scheduling problems (Caux, Pierreval & Portmann, 1995). Corrective procedure should be conceived because some illegal configurations can obviously be observed if we use basic genetic operators (crossover or mutation). The literature presents many different encodings for a large set of problems. We can mainly distinguish two types of codings. The first one is the direct representation in which the chromosome represents the solution itself. Such an approach requires conceiving specific genetic operators. The second one is the indirect representation in which the chromosome does not directly represent a solution, and a transition from the chromosome representation to a feasible solution is needed.

Concerning the application of the genetic algorithms to the planning problems, the literature shows a large set of works. Some of them have been successfully applied for solving the flexible job shop scheduling problem. We aim in this article to present most recent of them.

1. PMR (Parallel Machine Representation) (Mesghouni, 1999)
This coding represents the extension of Kobayashi's representation initially proposed for the classical job shop problem. Therefore, it offers the possibility of considering the assignment property. In fact, each cell is coded by three elements: the operation index (*i*), the corresponding job *j*, and starting time $t_{i,j}$ of operation $O_{i,j}$ on the corresponding machine. This representation can directly describe legal solutions and give all the information to execute the schedule. Unfortunately, it presents some difficulties concerning the generation of some illegal cyclic configurations. To overcome such an inconvenience, Mesghouni has proposed some corrective procedures. Unfortunately, such procedures imply an important cost in terms of computation time and therefore reduce the coding effectiveness.
2. PJsR (Parallel Jobs Representation) (Mesghouni, 1999)
Such a representation is an extension of the coding presented by Yamada for the classical job shop problem. Its particularity consists of the fact that resources can be reassigned to the different tasks (resource flexibility property). The chromosome consists of a list of jobs. Each job is coded by a row in which each case is represented by a couple of terms. The first term indicates the machine assigned to the operation. The second term is the corresponding starting time at which the above operation will be started.
This encoding allows us to obtain feasible solutions without illegal configuration by integrating the precedence constraints. Genetic operators are very simple and can easily be implemented. Unfortunately, such a coding has a reduced search capacity by comparing it to other possible encodings. Crossover and mutation operators have been proposed by Mesghouni for the two preceding representations. Despite their simplicity and fastness, they are completely based on the exchange of assignment choices and are not sufficient to take into account the sequencing property.
3. Ternary Permutation Matrix with Assignment Vector (Portmann, 1996)
This coding was proposed by Portmann initially to consider the precedence constraints for a large range of scheduling problems (Portmann 1996). In fact such a model consists of describing the sequencing property by a matrix *MT* noted "Permutation matrix" such that $MT(i,j)=1$ if *i* precedes *j*, $MT(i,j)=-1$ if *j* precedes *i*, and $MT(i,i)=0$ for every task *i* (see Figure 2). Such a coding allows us to keep the good properties in the generated schedule. In order to make it efficient, Portmann has proposed well-adapted operators that can generate feasible schedules with any corrective procedure (Portmann, 1996). Unfortunately, such a coding is insufficient to deal with the flexible job shop problem because of the assignment property. Nevertheless, Portmann has proposed to associate an assignment vector to the ternary coding. Such a proposition allows us to handle scheduling problems with assignment property but, unfortunately, dissociates their two independent parts.
4. Operations Machines Coding (OMC) (Kacem, Hammadi & Borne, 2003)
This is a direct representation of the schedule. It allows us to obtain all the information on the assignment choices and the different starting times. In addition, it allows us to integrate the schemata notion (Kacem et al., 2003). By such a coding, a schedule is represented in a table. Each case of the table can have one of two value possibilities: 0 (if the operation is not assigned to the machine considered) or [$t_{i,j}, tf_{i,j}$] (with $t_{i,j}$ and $tf_{i,j}$ respectively the starting time and the completion time of operation $O_{i,j}$ on the chosen machine). An illustrative example is presented in Figure 3(a).
5. List Operations Coding (LOC) (Kacem et al., 2003)
This consists of representing the schedule in a table with three columns. The first column is reserved for the operations . The second indicates the assigned machine to execute the corresponding operation, and the third column gives the starting time and the completion time. We can notice a great similarity between OMC and LOC. In fact, exploration assignment and sequencing search spaces have the same size. The only difference consists of the representation form. Such a difference gives more simplicity and more exploration possibilities (vertical crossover) for OMC. An illustrative example is presented in Figure 3(b).
6. Jobs Sequencings List Coding (JSLC) (Kacem et al., 2003)
Although, it is relatively difficult to be designed and difficult to be implemented, this encoding represents an efficient representation. In fact, it presents the same possibilities of the exploration of the assignment space search and offers more possibilities to explore the sequencing of one compared to the two preceding representations. It enables us to consider jointly or separately the assignment and the scheduling problems, and avoid the limited use of the priority rules. This coding is presented in a list of columns. Each column contains the different