A LOCAL SEARCH GENETIC ALGORITHM FOR THE JOB SHOP SCHEDULING PROBLEM

Kebabla Mebarek, Mouss Leila Hayat and Mouss Nadia

Laboratoire d'automatique et productique, Université Hadj Lakhdar -Batna
kebabla@yahoo.fr, hayet_mouss@yahoo.fr, kinzmouss@sahoo.fr

***Abstract*--Scheduling of job-shop is very important in the fields of production management and combinatorial optimization. This paper proposes a method for solving general job-shop scheduling problems based on hybridized algorithm that combines a genetic algorithm with a taboo search in two distinct phases research. In the first phase an operations-coded genetic algorithm is used to find an elite population. The set of elite solutions obtained from the first phase acts as the initial population of the second phase, in which a taboo search algorithm is applied to each one of them to intensify the research. The effectiveness of this algorithm is confirmed by applying it to a set of benchmarks with the makespan as the objective function. The results obtained show that local search applied at the final population can improve greatly the research.**

***Index Terms*-- Job-Shop Scheduling, Hybrid Meta-Heuristic, Genetic Algorithm, Local Search, Taboo Search**.

INTRODUCTION

The job-shop scheduling problem (JSSP) plays an important role in the scheduling theory and finds many practical applications. It deals with the sequencing of a set of jobs on a set of machines, in order to minimize an objective function [9]. For years, job-shop scheduling has attracted the attention of many researchers in the fields of both production management and combinatorial optimization. Efficient methods for solving the JSSP have significant effects on performance of production system. It has been demonstrated that this problem is usually an NP-complete (nondeterministic polynomial time complete) problem [5]. An indication for this is that one 10x10 problem formulated by Muth & Thompson in 1963 [10] remained unsolved for twenty years [12]. For this hardiness, exact methods become quickly inapplicable in practice. Instead, it is often preferred to use approximation algorithms such as heuristics and meta-heuristics e.g. simulated annealing, genetic algorithms, and taboo search.

In recent years, much attention has been devoted to four general heuristics: simulated annealing (SA), taboo search (TS), genetic algorithm (GA), and neural network (NN) [13]. These methods are capable of providing high-quality solutions with reasonable computational effort. However, the problem is hard that cannot be solved efficiently by applying any single technique and a great deal of research have focused on hybrid methods [14]. Several authors pointed out that the performance of genetic algorithm on some combinatorial optimization problems was a bit inferior to that of neighborhood search algorithms (e.g., local search, simulated annealing and taboo search). Hybrid methods of genetic algorithms and those neighborhood search algorithms were proposed, and their high performance was reported [6].

In this paper, we propose a genetic local search algorithm to improve the search process, the proposed algorithm acts in two phases. In the first one, a genetic algorithm is developed to find a set of best solutions. In the second, a local search algorithm with a memory has to intensify the research around each solution to improve it. The performance of the algorithm will be assessed through an experimental analysis with a set of benchmark problems. The remainder of this paper is organized as follows: in the next section we describe the problem of general job-shop scheduling to be solved. In section 3 we describe the proposed algorithm and its process. A numerical experiment is presented in the section 4.

PROBLEM DESCRIPTION

The problem studied in this paper is a deterministic and static n-job, m-machine job-shop scheduling problem (JSSP). The aim behind is to optimally allocate different operations for each job across a set of machines respecting temporal and resource constraints. This problem is formulated as follows:

There are n jobs $J_1,..., J_n$ to be scheduled on m machines $M_1,..., M_m$. Each job j consists of a sequence of n_j operations

O_{ij} ($i = 1, \ldots, nj$) that must be processed on the m machines in a given order O_{1j} - O_{2j} - ... - $O_{nj,j}$ (in our case $n_j = m$). Each operation is characterized by specifying both the required machine $M \in \{M_1,..., M_m\}$, and the fixed processing time $pij > 0$ [4].

Furthermore, several constraints considered on jobs and machines, are listed as follows:

(i) Each job must pass through each machine once and only once.

(ii) Each job should be processed through the machines in a particular order.

(iii) Each operation executed must be uninterrupted on a given machine (no pre-emption is allowed).

(iv) Each machine can only process one operation at a time.

The objective is to determine a feasible schedule with minimal *makespan* (i.e. minimizing the completion time of the last job) which is the most common goal for these problems:

$$C_{max} = \max_{j=1,\ldots,n}\{C_j\},$$

where C_j is the completion time of job Jj .

Table 1 presents an example of a job-shop problem formed of 3 jobs (J_1, J_2, J_3) which processed on 3 machines (M_1, M_2, M_3). For this problem, a solution schedule is presented by Gantt chart in Fig.1.

TABLE I

An example of a job-shop scheduling problem

$\boldsymbol{J_1}$:	M_1:4	M_2:3	M_3:3
$\boldsymbol{J_2}$:	M_1:1	M_3:5	M_2:3
$\boldsymbol{J_3}$:	M_2:2	M_1:4	M_3:1

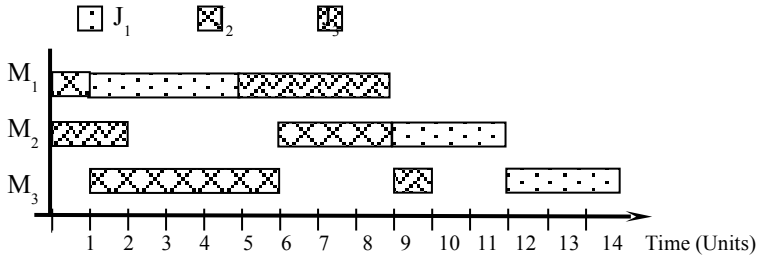

Fig. 1. A Gantt-Chart representation of a solution for the instance in TABLE I.

I. The Proposed Genetic Local Search Algorithm

For solving this problem, we introduced a fast and easily implemented hybrid algorithm. The proposed algorithm combines a genetic algorithm with a local search one. The former has to find a set of best solutions, and the local search procedure is applied to each solution generated by genetic operations to "dig" around for improving it. The global procedure of this algorithm is described briefly as follows:

A. First phase

In the first phase, we apply a genetic algorithm which begin with an initial population and attempt to improve it through successive generations. The process of this algorithm is presented in Algorithm 1.

Algorithm 1: The genetic algorithm

Input A scheduling problem instance P ;
Output A set of best schedules for instance P ;
1. Generate the initial population;
2. Evaluate the population;
while No termination criterion is satisfied **do**
3. Select chromosomes from the current population;
4. Apply the recombination operator to the chromosomes selected at step 3 to generate new ones;
5. Evaluate the chromosomes generated at step 4;
6. Apply the mutation operator to a randomly selected chromosomes;
7. Apply the selection criteria to replace new chromosomes;
return A set of best schedules evaluated so far;

1 Chromosome Representation: To represent the chromosome, we base on an *Operation-Based* representation that uses an unpartitioned permutation with m-repetitions of job numbers [3]. For an *n*-job *m*-machine problem, a chromosome contains *n*×*m* genes. Each job appears in the chromosome exactly *m* times, and each repeating gene does not indicate a concrete operation of a job but refers to an operation which is context-dependent. For example, considering the problem above (TABLE I), one of the chromosomes may be [231213123], which should be interpreted to a schedule as shown in Fig. 2. Each job number is repeated three times because each job has three operations. The first job number represents the first operation of the job, and the second represents the second operation. The order of genes in the chromosome represents the order in which the operations of jobs are scheduled.

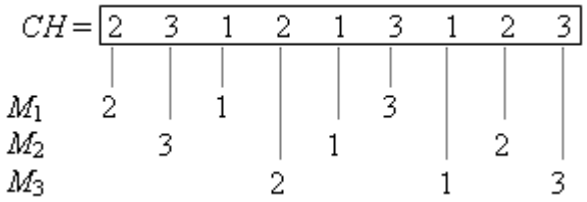

Fig. 2. A schedule building from a chromosome of the problem showed in TABLE I.

2 Initialization: In this algorithm the initial population consists of randomly generated chromosomes.

3 Gentic operators: The genetic evolution is done using three main genetic operators: crossover, mutation, and selection.

Crossover: In this study we adopt the *Generalized Order-Crossover* (GOX) scheme that generates only feasible solutions [3]. In GOX, one chromosome (donor) contributes a substring of length in the range of one third to half of his length. This substring is inserted in the receiver chromosome in the same position of the substring first gene after deleting all genes from the receiver with the same position as the genes in the substring according to their order in the jobs.

Mutation: The mutation operator has to bring a change to the chromosome. In our algorithm, we use a special mutation operator, that we called *Job Based Mutation* (JBM). In this mutation two jobs are randomly chosen. After that, all genes of the considered chromosome corresponding to one job are changed to the other. For example, chromosome [**2** 3 **1** **2** **1** 3 **1** **2** 3] become [**1** 3 **2** **1** **2** 3 **2** **1** 3] if considering jobs 1 and 2 to be swapped.

Selection: The selection mechanism for reproduction in this paper is based on the fitness ranking of the chromosomes. Two chromosomes are chosen with a probability proportional to their fitness for crossover among best individuals (some rate of worst solutions is excluded from being reproduced). Then deleting the worst member of the population.

4 Fitness Function: Solutions in both phases are evaluated according to their fitness. In this study, we use the makespan value of schedules as the fitness function.

B. Second phase

In this phase a local search procedure is applied to the best solutions among final offspring solutions generated by genetic operations. Essentially, local search consists in moving from a solution to another one in its neighborhood. So, we implant a simple taboo search method in order to avoid recycling. For this, two elements are necessary to define: the neighborhood structure and the memory (taboo list). The basic role of the taboo list is to prevent the search process from turning back to solutions visited in previous steps. The taboo list stores the arcs that have recently been reversed rather than the whole solutions. The length of the taboo list is usually of critical importance. Thus, we have used a dynamic taboo list that varying between two values [min, max] as it is proposed in [11]. Its length is decreased by one unit when the current solution is better than the previous one; otherwise it is increased in same amount.

For the neighborhood structure, in our taboo search, we adopt the technique used by Nowocki and Smutnicki [11], In this neighborhood, a critical path composed of *b* blocks is generated. A critical block of operations is defined as a set with the maximum successive operations that belong to the critical path and that are processed on a same machine. If $1 < l < b$ (l: block order), then swap the first two operations on the last block and the last two operations on the first block. However, if $l = 1$ swap only the last two operations in this block, on the other hand if $l = b$ swap the first two operations [11].

Algorithm 2 shows the taboo search algorithm we have considered here. This algorithm is a simple one, in the first step, the initial solution is taken from the set given by the first phase. Then, it iterates over a number of steps. In each iteration, the neighborhood of the current solution is built and one of the neighbors is selected for the next iteration.

The tabu search finishes after a fixed number of iterations that is no so much high. Then it passes to the following solution of the elite set to begin with it again.

Algorithm 2: The local search algorithm

Input A set of best schedules *C* and a problem instance *P*;
Output A schedule for instance *P*;
for all set of best solutions *i* **do**
1. Evaluate schedule *C(i)*;
2. Generate the *NS* neighborhood of the current solution;
If there is a better solution **then**
3. replace the current solution with and return to 1;
4. update the tabu list;
else
5. replace the current solution with best solution among the examined;
6. update the tabu list;
If iterations threshold is not reached **then** go to 2;
return the best schedule evaluated so far;

II. Numerical Experiments

The presented above genetic local search algorithm was programmed in Pascal Object (Delphi environment) and was run on a PC computer with a processor Intel P Dual 2.2 GHz for all the experiments. Fig. 3 shows a screenshot of this program solving the famous FT10 proposed in Muth & Thompson [10].

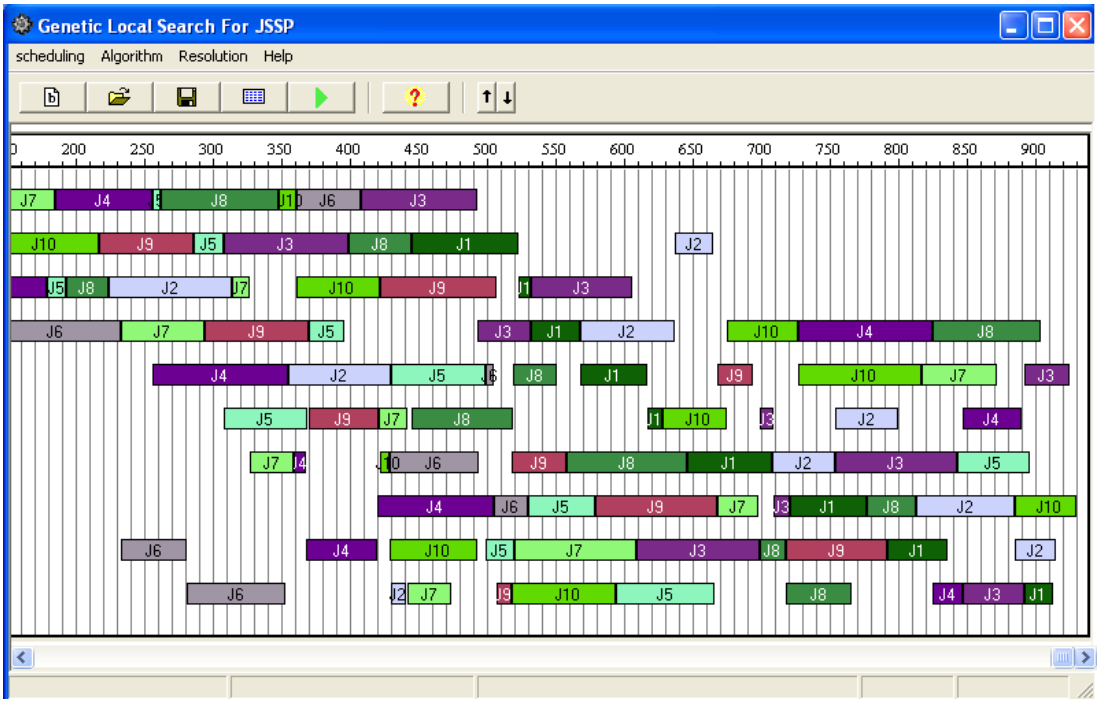

Fig.3. A screenshot of the program showing optimal solution of FT10.

The performance of the algorithm is analyzed on a set of benchmarks on the job-shop scheduling problem instances from literature. The size of the benchmark instances varies from 10 to 20 jobs and from 5 to 20 machines. We consider (FT10, FT20) proposed by Fisher and Thompson [10]; three problems (ABZ5-7) generated by Adams, Balas & Zawack [1]; ten 10 × 10 problems (ORB01-10) generated by Applegate and Cook [2] and 15 problems of different sizes (LA01-15) generated by Lawrence [8].

Table II shows the makespan performance statistics of the proposed GLS algorithm for the selected benchmark problems comparatively with a simple genetic algorithm that constitute the first phase of the whole GLS algorithm. It lists problem name, problem size (number of jobs, number of operations), the best-known solution, the best solution obtained by simple genetic algorithm in five attempts (i.e after first phase processing only), and the best solution obtained by our GLS algorithm in five attempts.

TABLE II

Computational results obtained by the proposed algorithm on benchmark problems

Instance	Size (n,m)	Best known solution	Best generated solution with GA in 5 attempts	Best generated solution with GLS in 5 attempts
FT10	(10,10)	930	938	930
FT20	(20,5)	1165	1173	1165
ABZ5	(10,10)	1234	1238	1234
ABZ6	(10,10)	943	944	943
ABZ7	(15,20)	656	680	667
ORB01	(10,10)	1059	1068	1059
ORB02	(10,10)	888	897	889
ORB03	(10,10)	1005	1011	1008
ORB04	(10,10)	1005	1032	1012
ORB05	(10,10)	887	890	887
ORB06	(10,10)	1010	1010	1010
ORB07	(10,10)	397	397	397
ORB08	(10,10)	899	899	899
ORB09	(10,10)	934	934	934
ORB10	(10,10)	944	944	944
LA01	(10,5)	666	666	666
LA02	(10,5)	655	665	665
LA03	(10,5)	597	597	597
LA04	(10,5)	590	590	590
LA05	(10,5)	593	593	593
LA06	(15,5)	926	926	926
LA07	(15,5)	890	890	890
LA08	(15,5)	863	863	863
LA09	(15,5)	951	951	951
LA10	(15,5)	958	958	958
LA11	(20,5)	1222	1222	1222
LA12	(20,5)	1039	1039	1039
LA13	(20,5)	1150	1150	1150
LA14	(20,5)	1292	1292	1292
LA15	(20,5)	1207	1207	1207

From the computational results of the table 1, it could be concluded that the proposed algorithm produced good solutions on all instances tested. In most of cases, it returns optimal solutions. In the few other cases the solutions are very near of the optimal. It could be concluded also, that the GLS algorithm returns best solutions or at least equal solutions to those returned by the simple genetic algorithm that constitute the first phase of the proposed algorithm. This means that the second phase improves really the research process.

Conclusion

Due to the stubborn nature of job-shop scheduling, much effort shown in the literature has focused on hybrid methods, since most single techniques cannot solve this problem efficiently [7]. In this paper, we have considered a general job-shop problem, and we have proposed a genetic local search algorithm. The proposed algorithm acts in two steps. Firstly, a genetic algorithm with operation-based encoding, GOX crossover, JBM mutation and elitist strategy selection is applied to get a set of best solutions. In

a second step, a tabu search procedure is applied to each solution of the elite set generated by genetic operations hopefully to improve some of them.

An empirical study is carried out to test the proposed strategies on a set of standard JSP instances taken from the literature. The results show that the proposed algorithm is an efficient mean in solving the problem considered. This algorithm gives better results than the two algorithms separately. It's concluded that combination of local search and genetic algorithms is a promising approach for improving resolution of job-shop scheduling problems.

References

[1] Adams, J., E. Balas, and D. Zawack. "The Shifting Bottleneck Procedure for Job Shop Scheduling", *Management Science*, vol.34 (1988), pp. 391–401.

[2] Applegate, D. & Cook, W., "A Computational Study of Job-shop Scheduling". *ORSA J. Computing*. vol. 2 (1991), pp. 149-156.

[3] Bierwirth C., "A generalized permutation approach to job shop scheduling with genetic algorithms", *OR Spektrum*, vol. 17 (1995), pp. 87-92.

[4] French, S., *Sequencing and Scheduling: An introduction to the Mathematics of the Job-Shop*, John Wiley & Sons, Inc., New York (1982).

[5] Garey M.R., Johnson D.S., Sethi R., "The Complexity of Flow-Shop and Job-Shop Scheduling", *Mathematics of Operations Research*, vol. 1 (1976), pp. 117-129.

[6] Ishibuchi H. & Tadahiko M., "Multi-Objective Genetic Local Search Algorithm", *Proceedings of IEEE International Conference on Evolutionary Computation*, 1996, pp. 119-124.

[7] Jain A.S. & Meeran S., "Deterministic job shop scheduling: Past, present, Future", *European Journal of Operation Research*, vol. 113, pp. 390-434, 1999.

[8] Lawrence S., "Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques", *Technical Report*, *Graduate School of Industrial Administration*, Pittsburgh, Carnegie Mellon University, 1984.

[9] Mati Y. & Xie X., "The complexity of two-job shop problems with multi-purpose unrelated machines", *European Journal of Operational Research*, vol. 152 (2004), pp. 159–169.

[10] Muth J. F. & Thompson G., *Industrial scheduling*, Prentice Hall, Englewood Cliffs, NJ, 1963.

[11] Nowicki E. and Smutnicki C., "A fast taboo search algorithm for the job shop scheduling problem" *Management Science*, vol. 42, pp. 797-813, 1996.

[12] Ombuki B. M. & Ventresca M., "Local Search Genetic Algorithms for the Job Shop Scheduling Problem", *Applied Intelligence*, vol. 21, pp. 99-109, 2004.

[13] Ponnambalam S. G., Aravindan P. and Rajesh S. V., "A Tabu Search Algorithm for Job Shop Scheduling", *Int. J. Adv. Manuf. Technol*., vol. 16, pp. 765-771, 2000.

[14] Vazquez M. & Whitley D., "A Comparison of Genetic Algorithms for the Static Job Shop Scheduling Problem", *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, pp.303-312, Sept.2000.

[15] Waiman C. and Hong Z., "Using Genetic Algorithms and Heuristics for Job Shop, Scheduling with Sequence-Dependent Setup Times" *Annals of Operations Research*, vol. 107, pp. 65-81, 2001.