HAL
open science

Proposition of New Genetic Operator for Solving Joint Production and Maintenance Scheduling : Application to the Flow Shop Problem.

Fatima Benbouzid-Sitayeb, Christophe Varnier, Nourredine Zerhouni

To cite this version:

Fatima Benbouzid-Sitayeb, Christophe Varnier, Nourredine Zerhouni. Proposition of New Genetic Operator for Solving Joint Production and Maintenance Scheduling : Application to the Flow Shop Problem.. IEEE International Conference Service System and Service Management, ICSSSM'06., Oct 2006, Troyes, France. pp.607-613, 10.1109/ICSSSM.2006.320531 . hal-00334160

HAL Id: hal-00334160

https://hal.science/hal-00334160v1

Submitted on 24 Oct 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Proposition of New Genetic Operator for Solving Joint Production and Maintenance Scheduling: Application to the Flow Shop Problem

Fatima Benbouzid-Sitayeb[1], Christophe Varnier[2], Nourredine Zerhouni[2]

[1] Laboratoire des Méthodes de Conception de Systèmes (LMCS). BP 68M Oued Smar Algiers – Algeria - f_sitayeb@ini.dz

[2] Laboratoire d'Automatique de Besançon (LAB). 25, rue A. Savary 25000 Besançon – France – cvarnier@ens2m.fr, zerhouni@ens2m.f

Abstract

Genetic algorithms are used in scheduling leading to efficient heuristic methods for large sized problems. The efficiency of a GA based heuristic is closely related to the quality of the used GA scheme and the GA operators: mutation, selection and crossover. In this paper, we propose a Joint Genetic Algorithm (JGA), for joint production and maintenance scheduling problem in permutation flowshop, in which different genetic joint operators are used. We also proposed a joint structure to represent an individual in with two fields: the first one for production data and the second one for maintenance data. We used different Taillard benchmarks to compare the performances of JGA with each proposed operator.

***Keywords* :** GA, Production, Maintenance, joint scheduling, joint crossover, joint mutation**.**

1. INTRODUCTION

Most production scheduling problems are NP-Hard [1]. The joint scheduling of production and maintenance is, in our sense, a complex problem due to the scheduling of two different activities: production jobs and systematic preventative maintenance tasks which use the same resources. Many approximation approaches and algorithms exist in the literature to attempt to solve NP-Hard problems. In this paper, we will focus on the Genetic Algorithms (GA) which gave interesting results for many NP-Hard problems [2, 3]. GAs have also been successfully applied to a variety of scheduling problems including jobshop and flowshop [15, 16, 17].

This paper proposes new genetic operators based on common representation of production and maintenance data. This was done in order to solve preventive maintenance and production joint scheduling in permutation flowshop, where each machine must be maintained periodically within known intervals of times. We chose the genetic algorithms for their proven effectiveness in the production scheduling problems. Moreover we use them for their flexibility on individuals' representation which lends itself perfectly to our problem. The aim is to optimize an objective function which takes into account the criteria of maintenance and production.

This paper is organized as follows: in section 2, a brief definition of joint production and maintenance scheduling is given. Section 3 presents the importance and drawbacks of each step of a standard GA. Section 4 presents the new genetic operators proposed for solving joint production/maintenance scheduling. Some results are given in section 5.

2. JOINT PRODUCTION AND MAINTENANCE SCHEDULING

Maintenance and production are two functions, which act on the same resources. However the scheduling of their respective activities is independent, and does not take account this constraint. These two elements having been established separately, their integration in the operation of the workshop poses a problem that is often solved by negotiation between the respective persons in charge of the two services in a sequential way.

We chose the systematic preventive maintenance, for this study, because it is preset and periodic. These two aspects are suits very well the maintenance scheduling. In that case, the search of a solution for the production scheduling is correlated with the search of a solution for the maintenance scheduling. Every search mechanism is constraint by other one, justifying so the need of a high collaboration between both functions.

Generally we have to schedule the production under the constraint of respect of the deadlines, cost and quality of the products. At the same time, the planning of maintenance is done under the constraints of equipment reliability. So, the execution of both schedules generates conflicts on the use of common resources (fig. 1).

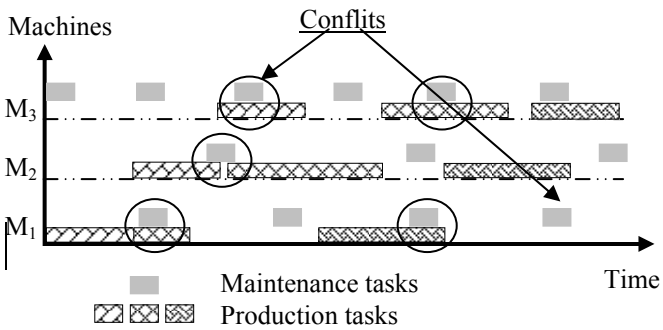

Fig. 1. production and maintenance separate scheduling

The interaction between maintenance and production, particularly their joint scheduling, is rather recent in the literature [4, 5, 6, 7, 8, 9, 22, 23, 24]. The majority of works concerning the interaction between production and maintenance use probabilistic approaches. The aim is to determine the best moment to plan maintenance task according to a compromise between the maintenance cost and the risk of machines unavailability [9, 10, 11].
Binding together these two functions is natural. Indeed, one notices that small maintenance tasks are directly integrated in the production scheduling. The objective is to plan the execution of maintenance tasks, while changing the least possible the production plan and respecting the equipment maintenance periodicity.
One counts in the literature several strategies of joint scheduling. We will describe below the sequential and integrated strategies which aim is to solve the existing conflicts between maintenance and production.
- The sequential strategy consists in planning, at first, one of the two activities (production or maintenance). Then the obtained schedule will be an additional constraint of machine unavailability in the resolution of the problem.
- The integrated strategy consists in generating a joint and simultaneous scheduling of maintenance and production tasks. This planning strategy diminishes the risks of interference between production and maintenance activities and allows optimizing the quality of the schedule.

The qualifier: sequential or integrated, of the above strategies, refers to the resolution method used and not to the obtained result. In both cases, simultaneous scheduling of production and maintenance tasks is generated.

The difference between the integrated strategy and the sequential one lies in the fact that for the first strategy both types of tasks (production and maintenance) are taken into account during the resolution. This implies a definition of an appropriate structure for the presentation of a joint scheduling.

We are interested, in this work, in the resolution of the problem of joint production/maintenance scheduling with the integrated strategy in permutation flowshop. We will develop the following two particular aspects of our problem in section 5: adapted representation for a joint solution (production and maintenance) and adapted operators.

3. GENETIC ALGORITHMS (GA): ADVANTAGES AND DRAWBACKS

GAs are based on the living species evolution phenomena. The evolution is provided by two interesting mechanisms: natural selection and reproduction. These phenomena, when repeated on a large population, after many generations, produce individuals that are well adapted to the environment in which they live.
A standard GA consists of the followings steps:
- Random generation of the initial population
- Selection
- Reproduction (crossover and mutation)
- Replacement of the current population

We will, quickly, go through each step of a standard GA to show its advantages and drawbacks.
The first step consists on generating the initial population. This is done randomly. The process is quick; however, we might not have a diversified population to work on.

Then a selection is performed. The most adapted individuals are selected for reproduction. The best individuals are always selected. This might lead to a premature convergence to a local optimum.

Crossover is then applied, with a certain probability, to selected individuals. This operator combines two solutions to produce two new ones. It is very quick as it does not take into account the nature of the problem. It generates blindly new solutions.

Mutation, an operator not as important as the crossover, is applied to individuals at a defined rate. The role of this operator is to change the characteristics of a solution. As a matter of fact, it tries to diversify the population by introducing new solutions in it. From the current population and the generated one (after crossover and mutation), a set of individuals has to be chosen to form the new generation. This population will enclose the best

individuals (solutions). After few generations, the GA tends to converge rapidly to the same solution.

The crossover operator has been considered to be the central component of GA and makes GA distinctively different from other problem solvers. By using this operator a pair of solutions (parents) generates new solutions (offsprings) by mutually exchanging and recombining information. A simple GA uses a bit string as a genotype representation and a bit manipulation crossover e.g., a 1- or 2- point or a uniform crossover. The increasing complexity of the problems to be solved, however, has made the simple approach difficult to use. Now, more powerful and tailored representations and their corresponding crossover operators are constructed to effectively solve more difficult problems. For example, permutation of the symbols denoting visiting cities might be used for TSP (traveling salesman problem), and permutation of the symbols denoting jobs or Gantt chart representations might be used for a scheduling problem[12, 13].

4. CONTEXT OF STUDY

4.1. Production data

The flow-shop problem can be presented as a set of N jobs $J_1, J_2 \ldots J_N$ to be scheduled on M machines. Machines are critical resources: one machine can be assigned to two jobs simultaneously. Each job J_i is composed of M consecutives tasks $t_{i1}, \ldots, t_{iM}$ where t_{ij} represents the jth task of the job J_i requiring the machine m_j. To each task t_{ij} is associated a processing time p_{ij}. Each job J_i must be achieved before the due date d_i.
In our study, we are interested in permutation flow-shop problems where jobs must be scheduled in the same order on all machines. We have to minimize the total completion time (C_{max}).
The task t_{ij} is scheduled at the time s_{ij}. The objective function can be computed as follows:

$$f_1 = C_{max} = \mathrm{Max}\{s_{iM} + p_{iM} / i \in [1..N]\} \quad (1)$$

In the R.L.Graham&al. notation [25] this problem can be defined by F/perm,di/(C_{max},T).

4.2. Maintenance data

The maintenance used is a systematic preventive one. The tasks are periodic interventions occurring every T^*_{ij} periods (T^*_{ij} indicates the task of maintenance i on the machine j). Each preventive maintenance task is characterized by a range of maintenance pre-established by the maintenance service or the manufacturer of the considered equipment. It consists of a succession of elementary operations which duration p'_{ij} is evaluated with more or less certainty. Moreover, the periodicity T* of these jobs is authorized to vary in a tolerance interval noted **[$Tmin_{ij}$,$Tmax_{ij}$]**. This interval represents a compromise between the maintenance cost and the machine unavailability risk (fig. 2).

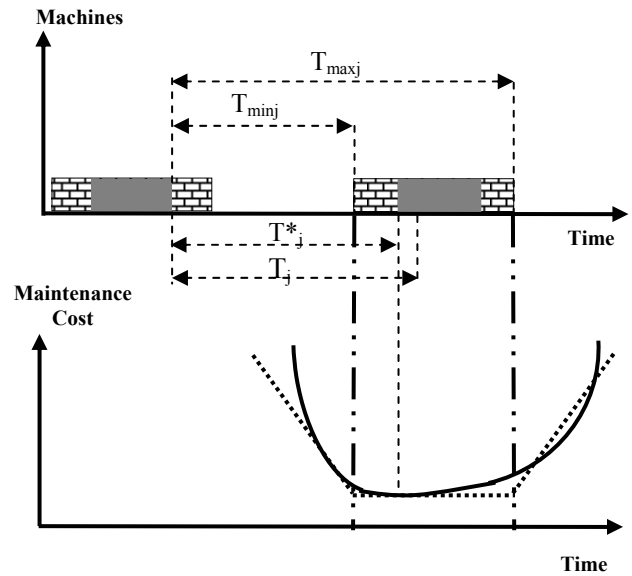

Fig. 2. Tolerance interval of a maintenance task.

we will note:

- M_j **:** periodical maintenance task associated to machine j ;
- p'_j **:** processing time of task M_j (its assumed know and constant) ;
- t'_{ij}: date of beginning of the ith maintenance task M_j ;
- E'_{ij} : Advance of ith maintenance task M_j ;
  $E'_{ij} = Max\ (0, (t'_{i-1j} + p'_j + Tmin_j) - t'_{ij})$;
- L'_{ij} : Delay of the ith maintenance task M_j ;
  $L'_{ij} = Max\ (0, t'_{ij} - (t'_{i-1j} + p'_j + Tmax_j))$.

From the point of view of the supplier, the respect of the maintenance periods influences the constraints of the production system. One will note f2 this objective function, which can be expressed as follows:

$$f2 = \sum_{j=1}^{m} \sum_{k=1}^{\max_{ij}} E'_{ijk} + L'_{ijk} \quad (2)$$

Where $\max_{ij}$ is the effective maintenance task number M_{ij}.

The goal is to propose a method that provides a common planning for the production jobs and maintenance tasks. Thus, the objective of optimization must be a compromise between the target objective maintenance and production functions.

To optimize the two criteria, we take into account the following common objective function:

$$f = \alpha . f_1 + \beta f_2. \quad (3)$$

α and β are chosen by the user.

5. JOINT PRODUCTION AND MAINTENANCE GENETIC ALGORITHM

5.1. Representation of a common production and maintenance solution

Encoding problems have been observed in the GA literature [18], where slightly different problems require completely different genetic encodings for a good solution to be found. Choosing a good representation is important to solve any search problem. However, choosing a good representation for a problem is as difficult as choosing a good search algorithm for a problem. Care must be taken to adopt both representational schemes and the associated genetic operators for an effective genetic search. Traditionally, chromosomes are simple binary vectors. This simple representation is an excellent choice for the problems in which a point naturally maps into a string of zeros and ones. Unfortunately, this approach cannot usually be used for real word engineering problems such as combinatorial ones [19].

Some different encodings are proposed in the literature [20, 21]. These encodings are split into two categories. The first one is the direct chromosome representation. For example, we can represent a scheduling problem by using the schedule itself as a chromosome. This method generally requires developing specific genetic operators. The second category is the indirect chromosome representation. The chromosome does not directly represent a schedule. It should be decoded to obtain a valid schedule prior to evaluation.

In this paper, we chose a direct representation to give viability and legibility to a chromosome and simplicity of utilization for a user. We suggest a direct representation for chromosome with its genetic operators.

Each individual is coded by a structure with two fields: the first field is a sequence S that represents the order of execution of the production jobs. The second is a matrix M that represents the sites of the maintenance tasks insertion. The element M [i,j] represents the insertion of the jth job of maintenance of the ith machine in the sequence S.
Example:

Sequence: S

9	8	5	3	1	2	0	7	4

Matrix M

$$\begin{vmatrix} 0 & 1 & 4 & 6 \\ 1 & 2 & 5 & \\ 0 & 4 & 7 & 8 \end{vmatrix}$$

M [3,2] = 4 means that the second maintenance task on the third machine is inserted in position 4 (after the production job 8) in sequence S. The execution of the tasks on the three machines according of the preceding example is the following:

M1 : $\mathbf{M_{01}}$,P_1,$\mathbf{M_{11}}$,P_9,P_3,P_8,$\mathbf{M_{21}}$,P_5,P_6,$\mathbf{M_{31}}$,P_7,P_4,P_2, P_0
M2 : P_1,$\mathbf{M_{02}}$,P_9,$\mathbf{M_{12}}$,P_3,P_8,P_5, $\mathbf{M_{22}}$, P_6,P_7,P_4,P_2,P_0
M3 : $\mathbf{M_{03}}$,P_1,P_9,P_3,P_8,$\mathbf{M_{13}}$,P_5,P_6,P_7,$\mathbf{M_{23}}$,P_4,$\mathbf{M_{33}}$,P_2,P_0

Where
M_{ij}: the ith maintenance task on the machine j, P_i: production job i.

5.2. Joint crossover

A valid individual should be generated from two parents. This individual should inherit its information from the production and the maintenance of its parents. This led us to define the following crossover operators:

1. The crossover on Production only. They are the classical crossover on production defined for GA [2, 19].
2. The crossover on Maintenance only. It was inspired from the k-points crossover [19] to define two new operators on maintenance: a horizontal crossover with even k-points and a horizontal crossover with odd k-points.

a- Horizontal crossover with even K-points: it consists in generating randomly a number k then performing the classical k-points crossover [19] but only on the maintenance tasks. The principle of this operator is presented in algorithm 1.

Algorithm Horizontal crossover with even K-points
Begin
Generate K points randomly ∈ [0.. a number of machines].
The sites of the maintenance jobs on the machines, which are in even parts of the first parent, are copied in the descendent 1, and the sites of the maintenances jobs that are in the remaining parts (odd) of the descendent 1 are copied from the odd parts of the second parent.
End

Algorithm 1. Horizontal crossover with even k points.

b- Horizontal crossover with odd K-points: It is the same principle as the horizontal crossover with K even points, except that in this case the odd parts are copied from the first parent and the even parts from the second.

The combination of the crossovers of production and those of maintenance enabled to define more operators.

The principle of horizontal crossover with even 1-point is illustrated by the following example with 1 point crossover (k=1), m=3 (3 machines) and 8 maintenance tasks.

	Parent 1		Child 1	
k →	Machine1	1 2 3 4	Machine 1	1 2 3 4
	Machine2	0 1 5	Machine 2	0 1 5
	Machine3	0 2 5 7	Machine 3	*2 3 4*

	Parent 2		Child 2	
k →	Machine 1	*0 3 6*	Machine 1	*0 3 6*
	Machine 2	*1 4 5 6*	Machine 2	*1 4 5 6*
	Machine 3	*2 3 4*	Machine 3	0 2 5 7

5.3. Joint mutation

The mutation can also be done on the production or maintenance. Three operators of mutation are proposed:

- Production mutation: it is done by generating two positions randomly, then changing the sites of the two production jobs which are within these two positions.
- Maintenance mutation*:* We define two types of mutation on maintenance: random mutation and vertical mutation.

a- Random mutation consists in shifting randomly towards the left or the right-hand side one or more maintenance jobs. The principle of this type of mutation is introduced in algorithm 2.

Algorithm Random mutation on maintenance

Local variables t_m: rate of mutation ($0 \leq t_m \leq n$) ; S=1 or 0;

Begin

  For each machine j

  Do Generate a random number r ($r \in [0, 1]$)

   If ($r \leq t_m$)

   Then Choose randomly a maintenance task on this machine

    Generate a random number S (0 or 1).

    If S=0

    Then shift this maintenance task on the left

    Else shift this maintenance task on the right

    End If

   End If

End.

Algorithm 2. Random mutation on maintenance.

b- Vertical mutation: The principle of this operator is to permute the sites of the production jobs, which are in two different parts, by preserving the sites of the maintenance tasks that are inside of each part compared to the production jobs which are in the same part. Its effectiveness increases when the maintenance tasks are well positioned compared to the production jobs.

We defined two operators of vertical mutation: 1-point vertical mutation and 2-points vertical mutation.

The principle of 1-point vertical mutation consists in generating randomly a position p in the sequence of production S, then the sites of both groups of production jobs, which are respectively between position 0 and p (first group) and between p+1 and n (second group) are permuted. This principle is introduced in algorithm 3.

Algorithm Vertical mutation with 1-point

Local variables n: number of machines; p: random number ($0 \leq p \leq n$);

Begin

- Generate a random number p
- Permute the sites of both groups of production jobs which are respectively between position 0 and p (group 1) and between p+1 and n (group 2).
- Update the maintenance tasks matrix M.

End.

Algorithm 3. Horizontal crossover with even 1-point.

The following example illustrates the principle of this operator for 3 machines and 10 maintenance tasks with p = 4

p

S

1	2	3	4	5	6	7	8	9	0

M

0 1 4 6
2 3
0 4 7 8

6	7	8	9	0	1	2	3	4	5

1 5 6 9
6 7 8
2 3 5 9

The principle of the 2-points vertical mutation is similar to the 1-point. Following example illustrates the principle of this operator with 10 maintenance tasks and 2 machines for: p_1=2 and p_2 =7

			p_1					p_2		
S	0	1	2	3	4	5	6	7	8	9

M 0 1 2 4 5 8
3 5 10

7	8	9	3	4	5	6	0	1	2

1 2 4 5 7 8 9
3 5 10

In what follows, we are going to introduce the tests which we performed on benchmarks of Taillard [14] as well as the results of these tests.

6. Tests and results

The production data on which we made the tests are Taillard benchmarks [14]. The data of maintenance are generated randomly, and are coded by Y-1 (Y indicates the number of machine). We use one maintenance task per machine with a constant processing time. The selected objective function is: $f_1 + f_2$ with α=1 and β=1.
GA was executed 100 times. The best results are saved, as well as associated parameters. The following parameters are the same for all the executions of the genetic algorithms:
Crossing rate: 0,7, Mutation rate : 0,01, strategy of renewal is N_best, the n best solutions are selected from the current population and the crossed one. The parameters of Sharing are $\alpha_{sharing}$= 0,99 and $\delta_{sharing}$ = 4.
In all that follows, we will use the following notations: HCE: Horizontal crossover with even 1 point; HCO: Horizontal crossover with odd one-point; VM1: Vertical mutation with 1 point; VM2: Vertical mutation with 2 points, RM: Random mutation.

6.1. JGA with horizontal crossover on maintenance

The classical one-point crossover is quick, as it does not take into consideration the nature of the problem. To overcome this drawback, we use a joint (production and maintenance) 1-point crossover. Table 1 summarizes the simulations performed with the joint crossover operators.

TABLE I
INTEGRATED PRODUCTION/MAINTENANCE GA WITH CROSSOVER ON MAINTENANCE

Benchmarks	PS	IT	MC	Solution
20*5_1	50	506	HCO	1874
20*5_2	50	186	HCE	2165
20*10_1	50	142	HCE	4003
20*10_2	100	326	HCE	3893
20*20	100	94	HCE	3602
50*5	60	493	HCO	12893
50*10	60	591	HCO	21981
50*20	60	580	HCE	33319
100*5	60	777	HCE	26130
100*10	60	246	HCO	80262
100*20	40	999	HCE	86700
200*10	40	197	HCO	328984
200*20	20	197	HCO	460439

PS: Population Size; **IT**: Iteration at which the best solution was found; **MC**: Maintenance Crossover; **HCE**: Horizontal Crossover with Even 1-point; **HCO**: Horizontal Crossover with Odd 1-point.

For the crossover on maintenance, we can assert that HCO is better than HCE. Horizontal crossover with even one-point crossover provided 51 % of the best results in comparison with the operator with odd one-point. We think that both operators are technically interesting.

6.2. JGA with vertical mutation on maintenance

After the selection step in a classical GA (section 3), only the best-fitted solutions are kept and become parents for the next generation. Mutation diversifies the new population by splitting the parent chromosomes to create new ones with crossover. Table 2 summarizes the simulations performed with the proposed joint mutation operators on maintenance.

TABLE II
INTEGRATED PRODUCTION/MAINTENANCE GA WITH MUTATION ON MAINTENANCE

Benchmarks	PS	IT	Mm	Solution
20*5_1	50	506	VM1	2789
20*5_2	50	186	VM1	2063
20*10_1	50	142	VM1	3964
20*10_2	100	326	VM2	3893
20*20	100	94	RM	3602
50*5	60	493	VM1	12930
50*10	60	591	VM2	21789
50*20	60	580	VM1	33430
100*5	60	777	VM1	26130
100*10	60	246	VM1	81760
100*20	40	999	VM2	86860
200*10	40	197	VM1	328796
200*20	20	197	VM2	469823

PS: Population Size; **IT**: Iteration at which the best solution was found; **MM**: Maintenance Mutation, **VM1:** 1-point Vertical Mutation, **VM2**: 2-points Vertical Mutation, **RM**: Random Mutation.

In the case of joint mutation operators, the one-point vertical mutation operator provided the best results. Indeed the principle to permute two groups in sequence S, and to perform the same operation on the associated matrix M is efficient.

For JAG, we performed 100 tests with the same

parameters and each operator defined in section 5. Table 3 summarizes the best results obtained for each operator.

TABLE III
RESULTS OF ALL OPERATORS

	CROSSOVER		MUTATION		
BEST RESULTS	HCE 51%	HCO 49%	RM 11 %	VM1 57 %	VM2 32 %

7. CONCLUSION

In this paper, we have developed a genetic algorithm for the resolution of the joint production and maintenance scheduling problem. Our approach uses the power of genetic algorithms to produce adapted and joint genetic operators for this particular scheme where an individual represents the production data and also the maintenance data.

The experimental results show that the use of horizontal crossover with even k-points as crossover on maintenance and the one-point vertical mutation on maintenance given the best results.

Future research will be to investigate a bi-criterion approach to solve this problem.

REFERENCES

[1] Garey & Johnson, "Computers and Intractability, A guide to the theory of NP-Completeness", *W.H. Freeman and Co., San Francisco*, 1979.

[2] Goldberg D.E., "Genetic algorithms in search", *Optimisation and Machine Learning.* Addison-Wesley, Mass., 1989.

[3] Dorne R., Hao J.K., "A new genetic local search algorithm for graph coloring", *LNCS*, 1498, pp.745-754, Springer Verlag, 1998.

[4] M.Ben-Daya & M.Makhdoum, "Integrated production and quality model under various maintenance policies". *Journal of the Operational Research Society*, 49(8), pp.840-853, 1998.

[5] M.Bennour, C.Bloch & N.Zerhouni, "Modélisation intégrée des activités de maintenance et de production. *3e Conférence Francophone de Modélisation et de SIMulation MOSIM'01* Troyes (France), 2 pp.805-810, 2001.

[6] X.Qi, T.Chen & F.Tu, Scheduling the maintenance on single machine. *Journal of the Operational Research Society*, 50(10), pp.1071-1078 , 1999.

[7] T.D.Rishel & D.P.Christy, Incorporating maintenance activities into production planning ; Intégration at the master schedule versus material requirement level. *International Journal of Production Research*, 34(2), pp.421-446, 1996.

[8] P.Fraternali & S.Paraboschi, Ordering and selecting production rules for constraint maintenance: Complexity and heuristic solution. *IEEE Transactions on Knowledge and Data Engineering*, 9(1), pp. 173-178, 1997.

[9] C.Y.Lee & Z.L.Chen, Scheduling jobs and maintenance activities on parallel machines. *Naval Research Logistics*, 47, p.145-165, 2000.

[10] S.F.Smith, Knowledge-Base Production Management: Approaches, Results and Prospectives. *Production Planning and Control*, 4(3), 1992.

[11] M.Brandolese, M.Fransi & A.Pozzeti, Production and maintenance integrated planning, *International Journal of Production Research*, 34(7), pp. 2059-2075, 1996.

[12] K.Benatchba, M.Koudil, H. Drias, R. Belkacem, "An adapted genetic algorithms for solving Max-Sat problems " *WSEAS transactions on systems*, 4(2), pp. 786-790, 2003.

[13] T. Yamada and R. Nakano. "A genetic algorithm applicable to large-scale job-shop problems". *In 2nd PPSN*, pp. 281–290, 1992.

[14] Taillard E., Benchmarks for basic scheduling problems, *European Journal of Operational Research*, 64, pp. 278-285, 1993.

[15] C. Bierwirth, D. Mattfeld, and H. Kopfer. On permutation representations for scheduling problems. In *4th PPSN*, pp. 310-318, 1996.

[16] T. Yamada and R. Nakano. Scheduling by genetic local search with multi-step crossover. *In 4th PPSN*, pp. 960-969, 1996.

[17] T. Yamada and C.R. Reeves. Permutation flowshop scheduling by genetic local search. *In 2nd IEE/IEEE Int. Conf. on Genetic Algorithms in Engineering Systems (GALESIA '97),* pp. 232-238, 1997.

[18] Dasgupta, D. and Michalewicz, Z. (editors). Evolutionary Algorithms in Engineering Applications. Berlin: Springer-Verlag, 1997.

[19] Portmann M.C., "Genetic algorithms and scheduling: a state of the art and some propositions", *Workshop on Production Planning and Control*, pp. I-XIV, Mons, 1996.

[20] Bagchi, S., Uckun, S., Miyabe, Y., & Kawamura, K. "Exploring Problem-Speci_c Recombination Operators for Job-Shop Scheduling". *Proc. of the Fourth Intl. Conf. on Genetic Algorithms,* pp. 10-17, 1991.

[21] Djerid L, Portmann M-C. "How to keep good schemata using crossover operators for the permutation problems". *International Transactions in Operations Research*, 7(6), pp.637-51, 2000.

[22] Benbouzid F., Bessadi Y., Guebli S., Varnier C. & Zerhouni N., "Résolution du problème de l'ordonnancement conjoint maintenance/production par la stratégie séquentielle", *4e Conférence Francophone de Modélisation et de SIMulation MOSIM'03* Toulouse (France), 2, pp. 627-633, 2003.

[23] Benbouzid F., Varnier C. & Zerhouni N., "Resolution of joint maintenance/production scheduling by sequential and integrated strategies", *7th International Work Conference on Artificial and Natural Neural Networks IWANN2003*, 3-6 June, Spain, Proceedings LNCS 2687, pp. 782-789, 2003

[24] Benbouzid F., Varnier C. & Zerhouni N., "Resolution of joint maintenance/production scheduling with GA", 9th *International Workshop on project management and scheduling PMS2004*, April 26-28, pp. 343-347, Nancy, France, 2004.

[25] Graham R.L., Lawler E.L., Lenstra J.K. and Kan A.H.G.R., "Optimisation and approximation in deterministic sequencing and scheduling: a survey". *In Annals of Discrete Mathematic*, 5, pp. 287-326, 1979.