Approach by Localization and Multiobjective Evolutionary Optimization for Flexible Job-Shop Scheduling Problems

Imed Kacem, Slim Hammadi, *Member, IEEE*, and Pierre Borne, *Fellow, IEEE*

***Abstract*—Traditionally, assignment and scheduling decisions are made separately at different levels of the production management framework. The combining of such decisions presents additional complexity and new problems.**

In this paper, we present two new approaches to solve jointly the assignment and job-shop scheduling problems (with total or partial flexibility).

The first one is the approach by localization (AL). It makes it possible to solve the problem of resource allocation and build an ideal assignment model (assignments schemata).

The second one is an evolutionary approach controlled by the assignment model (generated by the first approach). In such an approach, we apply advanced genetic manipulations in order to enhance the solution quality. We also explain some of the practical and theoretical considerations in the construction of a more robust encoding that will enable us to solve the flexible job-shop problem by applying the genetic algorithms (GAs).

Two examples are presented to show the efficiency of the two suggested methodologies.

***Index Terms*—Approach by localization (AL), assignment, controlled evolutionary algorithm, flexible job-shop, genetic manipulations, scheduling, schemata.**

I. Introduction

DESPITE the diversity of resolution methods and the spectacular evolution of the computing processors technology, scheduling problems remain difficult to solve. This difficulty is due to their combinatorial complexity [1]–[4]. Front to this difficulty, meta-heuristic techniques such as evolutionary algorithms can be used to find a good solution. The literature shows that they could be successfully used for combinatorial optimization, such as wire routing, transportation problems, scheduling problems, etc. [5], [6]. In order to be efficient, a method has to give good results in a reasonable computation time. It therefore has to explore intelligently the search space to avoid useless paths and explore the most suitable zones.

In this paper, we propose an efficient method to solve the assignment and job-shop scheduling problem (with partial or total flexibility). The considered objective is to minimize the overall completion time (makespan) and the total workload of the machines. This multi-objective optimization will be done in a suitable search space that will be determined by a judicious assignment algorithm. Computational experiments will be carried out to evaluate the efficiency of these methods with a large set of representative problem instances based on practical data. Analysis of the methods and their functioning will also be done.

This paper is organized as follows. In Section II, we formulate the problem and define the criteria used to evaluate schedule quality. Then, assignment algorithms are described in Section III. Section IV focuses on the evaluation of the approach by localization (AL) according to the solution quality and computing time. Section V highlights some aspects of the genetic algorithms (GAs) and schemata theory. Section VI shows the efficiency of the controlled evolutionary approach in the case of partial flexibility. An example with a total flexibility is studied in Section VII. Finally, in Section VIII, we give concluding remarks and introduce directions for future works.

Manuscript received March 5, 2001; revised December 10, 2001. This paper was recommended by Associate Editor M. Embrechts.

The authors are with the Laboratoire d'Automatique et d'Informatique de Lille, URA CNRS D 1440, Ecole Centrale de Lille, Cité Scientifique, Villeneuve d'Ascq 59651, France (e-mail: imed.kacem@ec-lille.fr; slim.hammadi@ec-lille.fr; p.borne@ieee.org).

Publisher Item Identifier S 1094-6977(02)04680-1.

II. Problem Formulation

A job-shop scheduling is a process-organized manufacturing facility; its main characteristic is a great diversity of jobs to be performed [7]. A job-shop produces goods (parts); these parts have one or more alternative process plans. Each process plan consists of a sequence of operations; these operations require resources and have certain (predefined) durations on machines. The task of planning, scheduling, and controlling the work is very complex, and perfect knowledge of the problem is necessary to assist in these tasks [7], [8].

The flexible job-shop scheduling problem is to select a sequence of operations together with assignment of start/end times and resources for each operation. The main considerations that need to be taken into account are the cost of having idle machine and labor capacity, the cost of carrying in-process inventory, and the need to meet certain completion due dates.

The problem now is to organize the execution of N jobs on M machines. The set of machines is noted U. Each job j represents a number of n_j nonpreemptable ordered operations. The execution of each operation i of a job j (noted $O_{i,j}$) requires one resource or machine selected from a set of available machines called $U_{i,j} \subseteq U$.

In the case of total flexibility, $U_{i,j} = U$ for each operation $O_{i,j}$. In the case of partial flexibility problem, we note the existence of at least one operation $O_{io,jo}$ such that $U_{io,jo} \subset U$.

The assignment of the operation $O_{i,j}$ to the machine $M_k\,(M_k \in U_{i,j})$ entails the occupation of this machine during a processing time called $d_{i,j,k}$.

1094-6977/02$17.00 © 2002 IEEE

```
Starting from a table D presenting the processing times possibilities on
the various machines, create a new table D' whose size is the same one as
the table D;
create a table S whose size is the same one as the table D (S is going to
represent chosen assignments);
initialize all elements of S to 0 (S_{i,j,k} = 0);
recopy D in D';
FOR (j=1; j ≤ N )
    FOR(i=1; i ≤ n_j)
•     Min= +∞ ;
•     Position=1 ;
•     FOR(k=1; k ≤ M)
            IF (d'_{i,j,k} <Min) Then {Min= d'_{i,j,k}; Position=k;}
            End IF
      End FOR
•     S_{i,j,Position} =1 (assignment of O_{i,j} to the machine M_{Position});
      // updating of D':
•     FOR(i'=i+1; i' ≤ n_j)
            d'_{i',j,Position} = d'_{i',j,Position} + d_{i,j,Position};
      End FOR
•     FOR(j'=j+1; j' ≤ N)
            FOR(i'=1; i' ≤ n_{j'})
            d'_{i',j',Position} = d'_{i',j',Position} + d_{i,j,Position};
            End FOR
      End FOR
    End FOR
End FOR
```

Fig. 1. Assignment_Procedure.

TABLE I
TABLE D

		M1	M2	M3	M4
J1	O1,1	1	3	4	1
	O2,1	3	8	2	1
	O3,1	3	5	4	7
J2	O1,2	4	1	1	4
	O2,2	2	3	9	3
	O3,2	9	1	2	2
J3	O1,3	8	6	3	5
	O2,3	4	5	8	1

In this problem, we make the following hypotheses:

— all machines are available at $t = 0$;
— all jobs can be started at $t = 0$;
— for each job, the order of operations is fixed and cannot be modified (precedence constraints);
— at a given time, a machine can only execute one task: it becomes available to the other tasks once the task which is currently assigned to is completed (resources constraints).

The flexible job-shop scheduling problems present two difficulties. The first one is to assign each operation $O_{i,j}$ to a machine M_k (selected from the set $U_{i,j}$). The second one is the computation of the starting times $t_{i,j}$.

The proposed optimization method will both be based on minimization of the makespan $(C_{\max})$ and balancing of the workloads W_k of machines $(k \in [1 \cdots M])$.

III. Assignment Algorithm

The algorithm is based on the procedure described in Fig. 1.

This procedure enables us to assign each operation to the suitable machine taking into account the processing times and workloads of machines on which we have already assigned operations.

TABLE II
TABLE D' FOR $j = 1$ AND $i = 1$

		M1	M2	M3	M4
J1	O1,1	1	3	4	1
	O2,1	4	8	2	1
	O3,1	4	5	4	7
J2	O1,2	5	1	1	4
	O2,2	3	3	9	3
	O3,2	10	1	2	2
J3	O1,3	9	6	3	5
	O2,3	5	5	8	1

TABLE III
TABLE D' FOR $j = 1$ AND $i = 2$

		M1	M2	M3	M4
J1	O1,1	1	3	4	1
	O2,1	4	8	2	1
	O3,1	4	5	4	8
J2	O1,2	5	1	1	5
	O2,2	3	3	9	4
	O3,2	10	1	2	3
J3	O1,3	9	6	3	6
	O2,3	5	5	8	2

TABLE IV
ASSIGNMENT S1

		M1	M2	M3	M4
J1	O1,1	1	0	0	0
	O2,1	0	0	0	1
	O3,1	1	0	0	0
J2	O1,2	0	1	0	0
	O2,2	0	1	0	0
	O3,2	0	0	1	0
J3	O1,3	0	0	1	0
	O2,3	0	0	0	1

TABLE V
D AFTER PERMUTATION

		M1	M2	M3	M4
J3	O1,3	8	6	3	5
	O2,3	4	5	8	1
J2	O1,2	4	1	1	4
	O2,2	2	3	9	3
	O3,2	9	1	2	2
J1	O1,1	1	3	4	1
	O2,1	3	8	2	1
	O3,1	3	5	4	7

A. Case of Total Flexibility

To explain this algorithm, we choose the following example (with a total flexibility).

The problem is to execute three jobs on four machines according to the processing times possibilities $d_{i,j,k}\ \{1 \le j \le N;\ 1 \le i \le n_j;\ 1 \le k \le M\}$ described in Table I.

The application (step by step) of the assignment procedure yielded the following results.

$j = 1; i = 1$: we assign $O_{1,1}$ to M_1 $(S_{1,1,1} = 1)$ and we add $d_{1,1,1} = 1$ to the elements of the first column of D' [the elements that follow the row (1, 1), see Table II].

$j = 1; i = 2$: we assign $O_{2,1}$ to M_4 $(S_{2,1,4} = 1)$ and we add $d_{2,1,4} = 1$ to the elements of the fourth column of D' [the elements that follow the row (2, 1), see Table III].

By following the same method, we obtain assignment S1 shown in Table IV.

Notice that the result obtained depends on the jobs positions in the table of processing times (the order of jobs) and machines positions (on columns). We illustrate this dependence by the two following examples.

TABLE VI
ASSIGNMENT S2

		M1	M2	M3	M4
J 1	O 1 ,1	0	0	0	1
	O 2 ,1	0	0	0	1
	O 3 ,1	1	0	0	0
J 2	O 1 ,2	0	1	0	0
	O 2 ,2	1	0	0	0
	O 3 ,2	0	1	0	0
J3	O 1 ,3	0	0	1	0
	O 2 ,3	0	0	0	1

D=Random_Exchange(D) (exchange randomly 2 jobs in the table D);
create a new table D' whose size is the same one as the table D;
create a table S whose size is the same one as D (S is going to represent chosen assignments);
initialize all elements of S to 0 ($S_{i,j,k}$ =0);
recopy D in D';
FOR (j=1; j ≤ N)
- **FOR** (i=1; i ≤ n_j)
- Min= +∞ ;
- r=RANDOM(M);
- Position=r;
- **FOR** (k=r; k ≤ M)
  IF ($d'_{i,j,k}$ <Min) **Then** {Min= $d'_{i,j,k}$; Position=k;}
  End IF
  End FOR
- **FOR** (k=1; k ≤ r-1)
  IF ($d'_{i,j,k}$ <Min) **Then** {Min= $d'_{i,j,k}$; Position=k;}
  End IF
  End FOR
- $S_{i,j,Position}$ =1: (assignment of $O_{i,j}$ to the machine $M_{Position}$);
  // Updating of D':
- **FOR** (i'=i+1; i' ≤ n_j)
  $d'_{i',j,Position} = d'_{i',j,Position} + d_{i,j,Position}$;
  End FOR
- **FOR** (j'=j+1; j' ≤ N)
- **FOR** (i'=1; i'≤$n_{j'}$)
  $d'_{i',j',Position} = d'_{i',j',Position} + d_{i,j,Position}$;
  End FOR
  End FOR
  End FOR
End FOR

Fig. 2. Assignment*_Procedure.

1) Example 1: Demonstration of the Influence of the Jobs Order on the Result: We permute the job 1 and the job 3 in the table of processing times. The result obtained is represented in Table V.

The application of the assignment procedure gives assignment S2 (see Table VI).

2) Example 2: Demonstration of the Influence of the Machines Order on the Result: We consider the example in Table I: we remember that we have assigned operation $O_{1,1}$ to the machine M_1 because it presents the minimum of processing times ($d_{1,1,1} = 1$), but we can see that the assignment of the same task on machine M_4 would have been possible (assignment with the same processing time $d_{1,1,4} = 1$). This result is due to the manner of covering rows of the table. In fact, the assignment is referred to machine M_1 by default if the rest of the machines does not present a more interesting solution (even in the case of equality).

To avoid such inconvenience that tends to privilege some configurations as compared to the others, we can randomly choose the machine by default using the function "*RANDOM*" and initializing the variable *Position* to a random value belonging to $[1 \cdots M]$.

Create a new table D' whose size is the same one as D;
create a table S with the same size of D (S will content the assignments);
initialize all elements of S to 0 ($S_{i,j,k}$ =0);
recopy D in D';
WHILE (D' is not empty)
- Min= +∞ ;
- r=RANDOM(M);
- PositionK=r ;
- **FOR** (j=1; j ≤ N)
  FOR(i=1; i ≤ n_j)
- **FOR** (k=r; k ≤ M)
  IF ($d'_{i,j,k}$<Min) **Then**
  {Min=$d'_{i,j,k}$;PositionK=k;PositionI=i;PositionJ=j}
  End IF
  End FOR
- **FOR**(k=1; k ≤ r-1)
  IF ($d'_{i,j,k}$<Min) **Then**
  {Min=$d'_{i,j,k}$;PositionK=k;PositionI=i;PositionJ=j}
  End IF
  End FOR
  End FOR
  End FOR
- $S_{PositionI,PositionJ,PositionK}$ = 1;
- eliminate the row (PositionI, PositionJ) from the table D';
// Updating of D':
- add $d_{PositionI,PositionJ,PositionK}$ to the elements of the column $Col_{PositionK}$ of the table D':
  $d'_{i,j,PositionK} = d_{PositionI,PositionJ,PositionK} + d'_{i,j,PositionK}$ for all i and j;
End WHILE

Fig. 3. Assignment**_Procedure.

TABLE VII
EXAMPLE D

		M1	M2	M3	M4
J 1	O 1 ,1	1	4	5	8
	O 2 ,1	7	5	6	5
J 2	O 1 ,2	2	5	6	2
J 3	O 1 ,3	12	5	4	7
	O 2 ,3	5	6	3	5
	O 3 ,3	2	4	12	5

For the jobs order problem, we have chosen to select two jobs randomly and to permute their positions in the table. Thus, the procedure *Assignment_Procedure* will be replaced by *Assignment*_Procedure* (see Fig. 2).

This procedure will enable us to construct a set of assignments (and balance the machines workloads) by modifying randomly the jobs order and the position of the machine to which we assign by default current operation.

3) Possible Improvement: The set of obtained assignments varies according to the random permutations. Therefore, we have thought to enrich it by other assignments independent of the jobs order. In fact, these assignments are found by entirely exploring the table of processing times. An operation $O_{i,j}$ is assigned to a machine M_k if $d'_{i,j,k}$ represents a global minimum on all the table. The updating of the table D' consists of taking account of the current workloads of machines (the same method as in the preceding procedure) and by eliminating the assigned operation from the search space [we eliminate the row (i, j) from the table D'] as follows *Assignment**_Procedure* in Fig. 3 and the example described in Table VII.

The application of the considered procedure yields the following results.

First iteration:

The global minimum corresponds to $d'_{1,1,1} = 1$; $PositionK = 1$; $PositionI = 1$; $PositionJ = 1$; $S_{1,1,1} = 1$. We eliminate the row (1, 1) and we add 1 to column Col_1 (see Table VIII).

Second iteration:

The global minimum corresponds to $d'_{1,2,4} = 2$; $PositionK = 4$; $PositionI = 1$; $PositionJ = 2$; $S_{1,2,4} = 1$. We eliminate the row (1, 2) and we add 2 to column Col_4 (see Table IX).

Third iteration:

The global minimum corresponds to $d'_{2,3,3} = 3$; $PositionK = 3$; $PositionI = 2$; $PositionJ = 3$; $S_{2,3,3} = 1$. We eliminate the row (2, 3) and we add 3 to column Col_3 (see Table X).

Fourth iteration:

The global minimum corresponds to $d'_{3,3,1} = 3$; $PositionK = 1$; $PositionI = 3$; $PositionJ = 3$; $S_{3,3,1} = 1$. We eliminate the row (3, 3) and we add 2 to column Col_1 (see Table XI).

Fifth iteration:

The global minimum corresponds to $d'_{2,1,2} = 5$; $PositionK = 2$; $PositionI = 2$; $PositionJ = 1$; $S_{2,1,2} = 1$. We eliminate the row (2, 1) and we add 5 to column Col_2 as follows Table XII.

Sixth iteration:

The global minimum corresponds to $d'_{1,3,3} = 7$; $PositionK = 3$; $PositionI = 1$; $PositionJ = 3$; $S_{1,3,3} = 1$. We eliminate the row (1, 3). D' becomes void and we obtain assignment S1 as presented in Table XIII.

Therefore, we obtain a total machine workload of $\sum W_k = 17$ units of time instead of 21 units (for the solution obtained by the first procedure described in Fig. 2).

Remark: The solution given by the first procedure is shown in Table XIV.

TABLE VIII
D' IN THE FIRST ITERATION

		M1	M2	M3	M4
J 1	O 2 ,1	8	5	6	5
J 2	O 1 ,2	3	5	6	2
J 3	O 1 ,3	13	5	4	7
	O 2 ,3	6	6	3	5
	O 3 ,3	3	4	12	5

TABLE IX
D' IN THE SECOND ITERATION

		M1	M2	M3	M4
J 1	O 2 ,1	8	5	6	7
J 3	O 1 ,3	13	5	4	9
	O 2 ,3	6	6	3	7
	O 3 ,3	3	4	12	7

TABLE X
D' IN THE THIRD ITERATION

		M1	M2	M3	M4
J 1	O 2 ,1	8	5	9	7
J 3	O 1 ,3	13	5	7	9
	O 3 ,3	3	4	15	7

TABLE XI
D' IN THE FOURTH ITERATION

		M1	M2	M3	M4
J 1	O 2 ,1	10	5	9	7
J 3	O 1 ,3	15	5	7	9

TABLE XII
D' IN THE FIFTH ITERATION

		M1	M2	M3	M4
J3	O 1 ,3	15	10	7	9

TABLE XIII
ASSIGNMENT S1

		M1	M2	M3	M4
J 1	O 1 ,1	1	0	0	0
	O 2 ,1	0	1	0	0
J 2	O 1 ,2	0	0	0	1
J 3	O 1 ,3	0	0	1	0
	O 2 ,3	0	0	1	0
	O 3 ,3	1	0	0	0

TABLE XIV
ASSIGNMENT S2

		M1	M2	M3	M4
J 1	O 1 ,1	1	0	0	0
	O 2 ,1	0	0	0	1
J 2	O 1 ,2	0	0	0	1
J3	O 1 ,3	0	0	1	0
	O 2 ,3	1	0	0	0
	O 3 ,3	0	1	0	0

B. Case of a Partial Flexibility

In this case, some tasks are only achievable on a part of the available machines set. In the example of Table XV, symbol "X" indicates that the assignment is impossible.

According to some authors [9], this constraint is going to make the problem more difficult, complicate the search space, and increase the computation time. However, we are going to show that such an assignment procedure is applicable in this case. In fact, the algorithm avoids to assign an operation to a machine whose processing time is long. Thus, for each forbidden assignment, we have associated an infinite fictitious processing time, which means we suppose that operation $O_{i0,\,j0}$ (correspondent to the forbidden assignment to machine M_{k0}) is henceforth achievable in an infinite time $d_{i0,\,j0,\,k0} = +\infty$. Thus, this assignment will be automatically rejected by the algorithm since it avoids long durations.

For the same preceding example, we only need to transform it in a job-shop with total flexibility with infinite processing time $d_{infinite} = 999$ for each forbidden state according to the equivalent data shown in Table XVI.

The application of the algorithm has confirmed this idea. In fact, no forbidden assignment has been presented by one of the solutions given by the assignment procedure.

Such a result shows an equivalence between the two types of problems. This equivalence is very interesting to implement a modular conception and construct supple solutions.

IV. TEST OF THE EFFICIENCY OF THE APPROACH BY LOCALIZATION

The objective of the assignment algorithm is to reduce the search space to an area where the probabilities of the balancing

TABLE XV
CASE OF A PARTIAL FLEXIBILITY

		M1	M2	M3	M4	M5	M6	M7	M8
J1	O1,1	5	3	5	3	3	X	10	9
	O2,1	10	X	5	8	3	9	9	6
	O3,1	X	10	X	5	6	2	4	5
J2	O1,2	5	7	3	9	8	X	9	X
	O2,2	X	8	5	2	6	7	10	9
	O3,2	X	10	X	5	6	4	1	7
	O3,4	10	8	9	6	4	7	X	X
J3	O1,3	10	X	X	7	6	5	2	4
	O2,3	X	10	6	4	8	9	10	X
	O3,3	1	4	5	6	X	10	X	7
J4	O1,4	3	1	6	5	9	7	8	4
	O2,4	12	11	7	8	10	5	6	9
	O3,4	4	6	2	10	3	9	5	7
J5	O1,5	3	6	7	8	9	X	10	X
	O2,5	10	X	7	4	9	8	6	X
	O3,5	X	9	8	7	4	2	7	X
	O4,5	11	9	X	6	7	5	3	6
J6	O1,6	6	7	1	4	6	9	X	10
	O2,6	11	X	9	9	9	7	6	4
	O3,6	10	5	9	10	11	X	10	X
J7	O1,7	5	4	2	6	7	X	10	X
	O2,7	X	9	X	9	11	9	10	5
	O3,7	X	8	9	3	8	6	X	10
J8	O1,8	2	8	5	9	X	4	X	10
	O2,8	7	4	7	8	9	X	10	X
	O3,8	9	9	X	8	5	6	7	1
	O4,8	9	X	3	7	1	5	8	X

TABLE XVI
EQUIVALENT OF TABLE XV

		M1	M2	M3	M4	M5	M6	M7	M8
J1	O1,1	5	3	5	3	3	999	10	9
	O2,1	10	999	5	8	3	9	9	6
	O3,1	999	10	999	5	6	2	4	5
J2	O1,2	5	7	3	9	8	999	9	999
	O2,2	999	8	5	2	6	7	10	9
	O3,2	999	10	999	5	6	4	1	7
	O3,4	10	8	9	6	4	7	999	999
J3	O1,3	10	999	999	7	6	5	2	4
	O2,3	999	10	6	4	8	9	10	999
	O3,3	1	4	5	6	999	10	999	7
J4	O1,4	3	1	6	5	9	7	8	4
	O2,4	12	11	7	8	10	5	6	9
	O3,4	4	6	2	10	3	9	5	7
J5	O1,5	3	6	7	8	9	999	10	999
	O2,5	10	999	7	4	9	8	6	999
	O3,5	999	9	8	7	4	2	7	999
	O4,5	11	9	999	6	7	5	3	6
J6	O1,6	6	7	1	4	6	9	999	10
	O2,6	11	999	9	9	9	7	6	4
	O3,6	10	5	9	10	11	999	10	999
J7	O1,7	5	4	2	6	7	999	10	999
	O2,7	999	9	999	9	11	9	10	5
	O3,7	999	8	9	3	8	6	999	10
J8	O1,8	2	8	5	9	999	4	999	10
	O2,8	7	4	7	8	9	999	10	999
	O3,8	9	9	999	8	5	6	7	1
	O4,8	9	999	3	7	1	5	8	999

of machines workloads and the minimization of the makespan are raised. To test the efficiency of this algorithm, it is therefore necessary to check the regularity of machines workloads and to see the impact of the choice of the assignments on the makespan value.

This test has been made by applying a scheduling algorithm after choosing the assignments. This algorithm calculates starting times $t_{i,j}$ by taking account of machines availabilities and precedence constraints. Conflicts are solved by a heuristic using different priority rules (SPT, LPT, LIFO, FIFO, RIFO, etc. [10], see Fig. 4).

In this following, we explain this scheduling algorithm, we then evaluate our results with an example of the literature to conclude on the assignments efficiency.

Beginning Scheduling Algorithm
initialize the vector of machines availabilities Dispo_Machine[k]=0 for each machine M_K ($k \leq M$);
initialize the vector of jobs availabilities Dispo_Job[j]=0 for each job j ($j \leq N$);
FOR ($i=1, i \leq \text{Max}_j(n_j)$)
- construct the set E_i of operations to schedule from S:
  $E_i = \{O_{i,j} / S_{i,j,k}=1,\ 1 \leq j \leq N\}$;
- classify the operations of E_i according to the chosen priority rule;
- **FOR** ($j=1; 1 \leq j \leq N$)
  - calculate starting times by following the same order given by the classification of E_i according to the formula:
    $t_{i,j} = \text{Max}(\text{Dispo\_Machine}[k], \text{Dispo\_Job}[j])$ such that $S_{i,j,k}=1$;
  - updating of the vector of machine availabilities:
    $\text{Dispo\_Machine}[k] = t_{i,j} + d_{i,j,k}$;
  - updating of the vector of job availabilities:
    $\text{Dispo\_Job}[j] = t_{i,j} + d_{i,j,k}$;

  End FOR
End FOR
End Scheduling Algorithm

Fig. 4. Scheduling algorithm.

A. Scheduling Algorithm

Example: We consider the example presented in Table I and we choose assignment S2 (already presented in Table VI). The application of "scheduling algorithm" using short processing time (SPT) rule yielded the following results.

— Iteration 1: $i = 1$:
Construction of $E_1 = \{O_{1,1}; O_{1,2}; O_{1,3}\}$: the operations of E_1 are respectively achievable during $d_{1,1,4} = 1$, $d_{1,2,2} = 1$ and $d_{1,3,3} = 3$ units of time, therefore they keep the same order $(d_{1,1,4} \leq d_{1,2,2} \leq d_{1,3,3})$. The starting times are computed by following the same order what gives: $t_{1,1} = t_{1,2} = t_{1,3} = 0$. The updating of machines and job availabilities gives the following vectors:

$Dispo\_Machines$: (0, 1, 3, 1); $Dispo\_Jobs$: (1, 1, 3).

— Iteration 2: $i = 2$:
Construction of $E_2 = \{O_{2,1}; O_{2,2}; O_{2,3}\}$: the operations of E_2 are respectively achievable during $d_{2,1,4} = 1$, $d_{2,2,1} = 2$ and $d_{2,3,4} = 1$ units of time. E_2 becomes therefore: $\{O_{2,1}; O_{2,3}; O_{2,2}\}$, in fact, $d_{2,1,4} \leq d_{2,3,4} \leq d_{2,2,1}$. The starting times are computed by following the same order what gives: $t_{2,1} = 1$; $t_{2,3} = 1$; $t_{2,2} = 3$. The update of machines and jobs availabilities gives the following vectors:

$Dispo\_Machines$: (3, 1, 3, 4); $Dispo\_Jobs$: (2, 3, 4).

— Iteration 3: $i = 3$:
Construction of $E_3 = \{O_{3,1}; O_{3,2}\}$: the operations of E_3 are respectively achievable during $d_{3,1,1} = 3$ and $d_{3,2,2} = 1$ units of time. E_3 becomes therefore: $\{O_{3,2}; O_{3,1}\}$, in fact, $d_{3,2,2} \leq d_{3,1,1}$. The starting times are computed by following the same order what gives: $t_{3,2} = 3$; $t_{3,1} = 3$. The update of machines and job availabilities yields the following vectors:

$Dispo\_Machines$: (6, 4, 3, 4); $Dispo\_Jobs$: (6, 4, 4).

TABLE XVII
A SCHEDULE GIVEN BY THE AL

	Ope_1	Ope_2	Ope_3
J1	4, 0, 1	4, 1, 2	1, 3, 6
J2	2, 0, 1	1, 1, 3	2, 3, 4
J3	3, 0, 3	4, 3, 4	******

Finally, the schedule is shown in Table XVII using the following representation:

[Machine, starting time, completion time].

Machines workloads (W_k): $\{W_1 = 5, W_2 = 2, W_3 = 3, W_4 = 3\}$.

The sum of workloads of machines $W = \sum W_k = 13$.

The workload of the most loaded machine $= \mathrm{Max}(W_k) = 5$.

The makespan $= C_{\max} = 6$.

B. Evaluation of Results: Case of Job-Shop With Partial Flexibility

To evaluate the efficiency of our approach, we have chosen an application on the example introduced in Table XV. Such an example has been already processed in the literature by two methods. It is therefore going to serve us to compare the quality of our solutions and solutions of the literature.

The first method is the temporal decomposition [11]. The obtained schedule in this case is characterized by the following values:

$$W = 91, \quad \mathrm{Max}(W_k) = 19, \quad C_{\max} = 19.$$

The second method has been developed by our team and it consists of applying classic GAs [9]. The best schedule obtained by this technique is characterized by the following values:

$$W = 77, \quad \mathrm{Max}(W_k) = 11, \quad C_{\max} = 16.$$

Such values show the efficiency of the genetic approach as compared to that of the temporal decomposition. In fact, the second method enables us to reduce the total machine workload (77 instead of 91) and to obtain a gain of more than 15% in terms of makespan. However, on the opposite, such a method is expensive in computation time.

Concerning the AL, the best schedule is obtained for the assignment shown in Table XVIII.

The obtained schedule is presented in Table XIX.

This obtained schedule is characterized by the following values: $W = 75$, $\mathrm{Max}(W_k) = 13$, and $C_{\max} = 16$.

The AL enables us to decrease the total machines workloads to 75 units of the time keeping a makespan of 16 units. The maximal workload is 13 units, which presents the same efficiency compared with the GAs.

C. Conclusion

Such results show that the AL enables us to construct solutions as interesting as solutions obtained using the classic GA (same value of makespan, similar workloads). The large advantage of this method is a significant reduction of the computation time. In fact, the assignment algorithm localizes most of the interesting zones of the search space. Thus, the scheduling is increasingly easy and becomes more efficient (another example, in Section VI, with total flexibility, confirms these results).

TABLE XVIII
ASSIGNMENT TABLE

		M1	M2	M3	M4	M5	M6	M7	M8
J1	O1,1	0	1	0	0	0	0	0	0
	O2,1	0	0	0	0	1	0	0	0
	O3,1	0	0	0	0	0	1	0	0
J2	O1,2	0	0	1	0	0	0	0	0
	O2,2	0	0	0	1	0	0	0	0
	O3,2	0	0	0	0	0	0	1	0
	O4,2	0	0	0	0	1	0	0	0
J3	O1,3	0	0	0	0	0	0	1	0
	O2,3	0	0	0	1	0	0	0	0
	O3,3	1	0	0	0	0	0	0	0
J4	O1,4	0	1	0	0	0	0	0	0
	O2,4	0	0	0	0	0	1	0	0
	O3,4	0	0	1	0	0	0	0	0
J5	O1,5	1	0	0	0	0	0	0	0
	O2,5	0	0	0	0	0	0	1	0
	O3,5	0	0	0	0	0	1	0	0
	O4,5	0	0	0	0	0	0	1	0
J6	O1,6	0	0	1	0	0	0	0	0
	O2,6	0	0	0	0	0	0	0	1
	O3,6	0	1	0	0	0	0	0	0
J7	O1,7	0	0	1	0	0	0	0	0
	O2,7	0	0	0	0	0	0	0	1
	O3,7	0	0	0	1	0	0	0	0
J8	O1,8	1	0	0	0	0	0	0	0
	O2,8	0	1	0	0	0	0	0	0
	O3,8	0	0	0	0	0	0	0	1
	O4,8	0	0	0	0	1	0	0	0

TABLE XIX
OBTAINED SCHEDULE

	Ope_1	Ope_2	Ope_3	Ope_4
J1	2, 1, 4	5, 4, 7	6,13,15	******
J2	3, 3, 6	4, 6, 8	7,11,12	5,12,16
J3	7, 0, 2	4, 8,12	1,12,13	******
J4	2, 0, 1	6, 1, 6	3, 6, 8	******
J5	1, 2, 5	7, 5,11	6,11,13	7,13,16
J6	3, 0, 1	8, 1, 5	2, 8,13	******
J7	3, 1, 3	8, 5,10	4,12,15	******
J8	1, 0, 2	2, 4, 8	8,10,11	5,11,12

In general, the solutions of the previously mentioned approach are equally acceptable and satisfactory. However, the solutions of many real problems are not necessarily of the same value in the eyes of decision makers and optimum or high-quality solutions according to the desired criterion are preferred [12]. Therefore, it is worthwhile to investigate possible gains from hybridizing the AL with the GA which have been used to produce appropriate solutions for many problems while they do not guarantee the optimality of the final solution.

V. THE APPROACH BY LOCALIZATION AND CONTROLLED GENETIC ALGORITHM: NOTION OF ASSIGNMENT SCHEMATA

In this section, we show how the AL can contribute to a multiobjective optimization by combining it with GAs.

A. Genetic Algorithms (GAs)

GAs enable us to make an initial set of solutions evolve to a final set of solutions bringing a global improvement according to a criterion fixed at the beginning [5], [13]–[18].

These algorithms function with the same usual genetic mechanisms (crossover, mutation, and selection).

Here, we explain the important concepts of GAs. The solutions set is called *population*. Each population is constituted of chromosomes which each represent a particular coding of a so-

M 1	(i, J_j, $T_{i,j,1}$)	…
M 2		…
M 3	(i', $J_{j'}$, $T_{i',j',3}$)	
….		
M n	…	…

Fig. 5. Parallel machine representation.

J 1	(M_1, T_{M1})	(M_2, T_{M2})	…
J 2	(M_5, T_{M5})	(M_1, T_{M1})	(M_2, T_{M2})
J 3			
…			
J n	…	…	

Fig. 6. Parallel jobs representation.

lution. The chromosome consists of a sequence of genes that can take some values called *alleles*. These values are taken from an alphabet that has to be judiciously chosen to suit the studied problem.

The classic coding corresponds to the binary alphabet: $\{0, 1\}$. In this case, the chromosome represents simply a table of 0 and 1.

The operators that intervene in the GAs are selection, crossover, and mutation.

The implementation difficulty of these algorithms consists of conceiving the genes content in order to describe all data of the problem.

Concerning evolutionary algorithms and flexible job-shop scheduling problems, the literature presents many interesting propositions. Some of them can be used to solve the considered optimization problem. As examples, we present the following coding possibilities.

1) Parallel Machine Representation (PMR) [9]: The chromosome is a list of machines placed in parallel (see Fig. 5). For each machine, we associate operations to execute. Each operation is coded by three elements:

1) $i =$ operation index;
2) $J_j =$ corresponding job;
3) $T_{i,j,k} =$ starting time of $O_{i,j}$ on the machine M_k.

2) Parallel Jobs Representation (PJsR) [9].: The chromosome is represented by a list of jobs. Each job is represented by the corresponding row where each case is constituted of two terms. The first term represents the machine that executes the operation. The second term represents the corresponding starting time (Fig. 6).

In [6], we can find other coding possibilities for scheduling problems with or without assignment. As an example, Portmann *et al.* have proposed to use "ternary permutation matrix" with "assignment vector" already proposed by Vignier *et al.* [19] to deal with hybrid flow shop problem. However, these codings are not specified for the flexible job-shop problem.

A GA is an algorithm that represents a special architecture. It operates on data without using preliminary knowledge on the problem processed. In fact, it consists of the following stages.

1) *Genesis:* This is the generation phase of the initial population.
2) *Evaluation:* In this stage, we compute the value of criterion for each individual of the current population.
3) *Selection:* After the evaluation, we choose better adapted elements for the reproduction phase.
4) *Reproduction:* We apply genetic operators (crossover and mutation) on the selected individuals.
5) *Test:* In this phase, we evaluate the improvement and decide if the solution is efficient. If the criterion reaches a satisfactory value, we take the current solution. If the result is insufficient, we return to the second phase and we repeat the same process until reaching the maximal iterations number.

B. The Schemata Theorem

This notion was introduced in GAs by Holland [20]. It consists of conceiving a model of chromosomes that suits the problem. This model will serve in the construction of new individuals in order to integrate the good properties contained in the schemata [21].

Example: In the case of a binary coding, a schemata is a chromosome model where some genes are fixed and the others are free (see the following example S):

$$S = 100 * 1 * 00.$$

Positions 4 and 6 are occupied by the symbol: "$*$." This symbol indicates that considered genes can take "0" or "1" as value. Thus, chromosome C1 and C2 respect the model imposed by the schemata S

$$\text{C1} = 10\,001\,100$$
$$\text{C2} = 10\,011\,100.$$

The objective of the schemata theory is to make GAs more efficient and more rapid in constructing the solution by giving priority to the reproduction of individuals respecting model generated by the schemata and not from the whole set of chromosomes.

In the case of scheduling problems, the difficulty of the implementation of this technique is higher. In fact, it necessitates the elaboration of a well particular coding that enables us to describe the problem data and exploit the schemata theory.

Here, we show how the AL enables us to overcome this difficulty and we introduce this notion to solve flexible job-shop scheduling problems.

We are reminded that the AL enables us to construct a set E of assignments by minimizing the sum of machines workloads. The idea is to generate, from the set E, an assignment schemata that will serve us to control the GA. This schemata is therefore going to represent a constraint which newly created individuals must respect. Thus, it would enable us to optimize makespan in a search area where assignments minimize the workloads of the machines (optimization by phase).

The construction of this schemata consists of collecting the assignments $S^z(1 \leq z \leq \text{cardinal}(E))$ given by the procedure *Assignment*_Procedure* and by the procedure *Assignment**_Procedure* and to determine (for each operation)

J 1	(M_1, T_{M1})	(M_2, T_{M2})	…
J 2	(M_5, T_{M5})	(M_1, T_{M1})	(M_2, T_{M2})
J 3			
…			
J n	…	…	

Fig. 7. Schemata generation algorithm.

TABLE XX
ASSIGNMENT SCHEMATA S^{ch}

		M1	M2	M3	M4	M5	M6	M7	M8
J 1	O 1 ,1	*	*	0	*	*	0	0	0
	O 2 ,1	0	0	*	0	*	0	0	*
	O 3 ,1	0	0	0	0	0	*	*	0
J 2	O 1 ,2	*	0	*	0	0	0	0	0
	O 2 ,2	0	0	*	*	*	0	0	0
	O 3 ,2	0	0	0	0	0	0	1	0
	O 4 ,2	0	0	0	0	1	0	0	0
J 3	O 1 ,3	0	0	0	0	0	0	1	0
	O 2 ,3	0	0	*	*	0	0	0	0
	O 3 ,3	1	0	0	0	0	0	0	0
J 4	O 1 ,4	*	*	0	0	0	0	0	0
	O 2 ,4	0	0	*	0	0	*	*	0
	O 3 ,4	*	0	*	0	*	0	*	0
J 5	O 1 ,5	1	0	0	0	0	0	0	0
	O 2 ,5	0	0	*	*	0	0	*	0
	O 3 ,5	0	0	0	0	*	*	0	0
	O 4 ,5	0	0	0	0	0	*	*	*
J 6	O 1, 6	0	0	1	0	0	0	0	0
	O 2, 6	0	0	0	0	0	0	*	*
	O 3, 6	0	1	0	0	0	0	0	0
J 7	O 1 ,7	*	*	*	0	0	0	0	0
	O 2 ,7	0	0	0	0	0	0	0	1
	O 3 ,7	0	0	0	*	0	*	0	0
J 8	O 1 ,8	*	0	0	0	0	*	0	0
	O 2 ,8	0	*	*	0	0	0	0	0
	O 3 ,8	0	0	0	0	*	0	*	*
	O 4 ,8	0	0	*	0	*	*	0	0

the set of possible machines in S^{ch} according to the algorithm shown in Fig. 7.

For each operation $O_{i,j}$, this algorithm associates the frequency $S^{ch}_{i,j,k}$ to be assigned to a machine M_k then, in function of chosen thresholds α and β, it reduces the set $U_{i,j}$ to a subset where the probabilities of having a good schedule are raised.

As an example, for the problem introduced in Table XV, we obtain the schemata S^{ch} shown in Table XX (for $\alpha = 0.03$ and $\beta = 0.95$).

The value $S^{ch}_{i,j,k} = 0$ indicates that the assignment of the operation $O_{i,j}$ to the machine M_k is *forbidden*.

The value $S^{ch}_{i,j,k} = 1$ indicates that the assignment of the operation $O_{i,j}$ to the machine M_k is obligatory, in this case, all values of the rest of the row (i, j) are inevitably equal to "0."

The symbol "$*$" indicates that the assignment is possible, in this case, we cannot have the value "1" in all the rest of the row (i, j).

In conclusion, this schemata covers the totality of the interesting assignment possibilities and expensive prohibitions in terms of machine workloads. The form of this schemata has forced us to develop a well-adapted coding that is presented in the following paragraph.

C. New Coding: Operations Machines Coding (OMC)

1) Coding: It consists in representing the schedule in the same assignment table S. We replace each case $S_{i,j,k} = 1$ by the couple $(t_{i,j}, tf_{i,j})$ where $t_{i,j}$ is the starting time and $tf_{i,j}$ is the completion time. The cases $S_{i,j,k} = 0$ are unchanged.

TABLE XXI
CODING OMC

		M1	M2	M3	M4
J 1	O 1 ,1	0	0	0	0, 1
	O 2 ,1	0	0	0	1, 2
	O 3 ,1	3, 6	0	0	0
J 2	O 1 ,2	0	0, 1	0	0
	O 2 ,2	1, 3	0	0	0
	O 3 ,2	0	3, 4	0	0
J 3	O 1 ,3	0	0	0, 3	0
	O 2 ,3	0	0	0	3, 4

Select randomly 2 parents S^1 and S^2;
select randomly 2 integers j and j' such that $j \leq j' \leq N$;
select randomly 2 integers i and i' such that $i \leq n_j$ and $i' \leq n_{j'}$ (in the case where j=j', $i \leq i'$);
the individual e^1 receives the same assignments from the parent S^1 for all operations between the row (i, j) and the (i', j');
the rest of assignments for e^1 is obtained from S^2;
the individual e^2 receives the same assignments from the parent S^2 for all operations between the row (i, j) and the row (i', j');
the rest of assignments for e^2 is obtained from S^1;
calculate the starting and completion times according to the algorithm "Scheduling Algorithm".

Fig. 8. Crossover algorithm.

TABLE XXII
PARENT S^1

		M1	M2	M3	M4
J 1	O 1 ,1	0	0	0	0, 1
	O 2 ,1	0	0	0	1, 2
	O 3 ,1	3, 6	0	0	0
J 2	O 1 ,2	0	0, 1	0	0
	O 2 ,2	1, 3	0	0	0
	O 3 ,2	0	3, 4	0	0
J 3	O 1 ,3	0	0	0, 3	0
	O 2 ,3	0	0	0	3, 4

TABLE XXIII
PARENT S^2

		M1	M2	M3	M4
J 1	O 1 ,1	0, 1	0	0	0
	O 2 ,1	0	0	0	1, 2
	O 3 ,1	3, 6	0	0	0
J 2	O 1 ,2	0	0	0, 1	0
	O 2 ,2	1, 3	0	0	0
	O 3 ,2	0	3, 4	0	0
J 3	O 1 ,3	0	0	1, 4	0
	O 2 ,3	0	0	0	4, 5

To explain this coding, we present the same schedule introduced in Table XVII (under the PJsR coding) using the operations machines coding (OMC) one as follows Table XXI.

Remark: We use the following example to define genetic operator associated to this coding in all the continuation.

This new coding presents several advantages. On the one hand, it integrates the notion of the assignment schemata that represents the "skeleton" of an optimized scheduling. On the other hand, it enables us to exchange information contained in current good solutions and make fine crossovers (the elementary level is the operation when it was the job in the case of the coding PJsR). Also, this coding presents an easy form to interpret. In fact, by looking at the rows, we observe the execution of the operations and by looking at the columns, we get the tasks of each machine with the starting and completion times.

2) Crossover: This operator is described in Fig. 8 and illustrated by the following example (see Tables XXII–XXVII):

TABLE XXIV
e^1 IN CONSTRUCTION

		M1	M2	M3	M4
J 1	O 1 ,1	?,?	0	0	0
	O 2 ,1	0	0	0	?,?
	O 3 ,1	?,?	0	0	0
J 2	O 1 ,2	0	?,?	0	0
	O 2 ,2	?,?	0	0	0
	O 3 ,2	0	?,?	0	0
J 3	O 1 ,3	0	0	?,?	0
	O 2 ,3	0	0	0	?,?

TABLE XXV
e^2 IN CONSTRUCTION

		M1	M2	M3	M4
J 1	O 1 ,1	0	0	0	?,?
	O 2 ,1	0	0	0	?,?
	O 3 ,1	?,?	0	0	0
J 2	O 1 ,2	0	0	?,?	0
	O 2 ,2	?,?	0	0	0
	O 3 ,2	0	?,?	0	0
J 3	O 1 ,3	0	0	?,?	0
	O 2 ,3	0	0	0	?,?

TABLE XXVI
e^1 FIRST OFFSPRING

		M1	M2	M3	M4
J 1	O 1 ,1	0, 1	0	0	0
	O 2 ,1	0	0	0	1, 2
	O 3 ,1	3, 6	0	0	0
J 2	O 1 ,2	0	0, 1	0	0
	O 2 ,2	1, 3	0	0	0
	O 3 ,2	0	3, 4	0	0
J 3	O 1 ,3	0	0	0, 3	0
	O 2 ,3	0	0	0	3, 4

TABLE XXVII
e^2 SECOND OFFSPRING

		M1	M2	M3	M4
J 1	O 1 ,1	0	0	0	0, 1
	O 2 ,1	0	0	0	1, 2
	O 3 ,1	3, 6	0	0	0
J 2	O 1 ,2	0	0	0, 1	0
	O 2 ,2	1, 3	0	0	0
	O 3 ,2	0	3, 4	0	0
J 3	O 1 ,3	0	0	1, 4	0
	O 2 ,3	0	0	0	4, 5

— construction of offsprings (copying of the assignments, see Tables XXIV and XXV).
— computation of starting and completion times (See Tables XXVI and XXVII).

3) Mutation: The objective of our search is to minimize the makespan and workloads of machines. It would therefore be interesting to make genetic operators able to contribute in this optimization. In such a context, we propose two operators of artificial mutation.

a) Operator of mutation reducing the effective processing time (EPT_j) of a job j (Fig. 9): Let us consider the example S shown in Table XXVIII. The job 1 has the most raised value of the EPT's ($EPT_1 =$ six units of time). We therefore have to cover the list of its operations to reduce this duration. For operation $O_{1,1}$ the processing time $d_{1,1,4}$ corresponds to the minimum of processing times. On the other hand, operation $O_{2,1}$, can be assigned to the machine M_4 instead of the machine M_3 (because $d_{2,1,4} < d_{2,1,3}$) and thereafter we reduce the EPT_1 to five units of time and the makespan to six units instead of eight. The obtained chromosome is shown in Table XXIX.

Select randomly an individual S;
choose the job j whose Effective Processing Time is the most long:
$(\text{Max}_j \{ EPT_j \text{ such that } EPT_j = \sum_i \sum_k S_{i,j,k} \cdot d_{i,j,k} \});$
i=1; r = 0;
WHILE ($i \leq n_j$ and r = 0)
- find K_0 such that $S_{i,j,K0} = 1$;
- **FOR** (k=1, $k \leq M$)
  IF ($d_{i,j,k} < d_{i,j,k0}$) **Then** {$S_{i,j,K0} = 0$; $S_{i,j,K} = 1$; r=1;}
  End IF
  End FOR
- i=i+1;

End WHILE
calculate starting and completion times according to the algorithm "Scheduling Algorithm";

Fig. 9. First mutation algorithm.

TABLE XXVIII
S BEFORE MUTATION 1

		M1	M2	M3	M4
J 1	O 1 ,1	0	0	0	0, 1
	O 2 ,1	0	0	3, 5	0
	O 3 ,1	5, 8	0	0	0
J 2	O 1 ,2	0	0, 1	0	0
	O 2 ,2	0	1, 4	0	0
	O 3 ,2	0	0	0	4, 6
J 3	O 1 ,3	0	0	0, 3	0
	O 2 ,3	0	0	0	3, 4

TABLE XXIX
S AFTER MUTATION 1

		M1	M2	M3	M4
J 1	O 1 ,1	0	0	0	0, 1
	O 2 ,1	0	0	0	1, 2
	O 3 ,1	2, 5	0	0	0
J 2	O 1 ,2	0	0, 1	0	0
	O 2 ,2	0	1, 4	0	0
	O 3 ,2	0	0	0	4, 6
J 3	O 1 ,3	0	0	0, 3	0
	O 2 ,3	0	0	0	3, 4

Select randomly an individual S;
find the most loaded machine M_{k1}:
$W_{k1} = \text{Max}_k \{ W_k / W_k = \sum_j \sum_i S_{i,j,k} \cdot d_{i,j,k} \};$
find the less loaded machine M_{k2} ($\text{Min}_k \{ W_k \}$);
choose randomly an operation $O_{i,j}$ such that $S_{i,j,k1} = 1$;
assign this operation to the less loaded machine: $S_{i,j,k1} = 0$; $S_{i,j,k2} = 1$;
calculate the starting and completion times according to the algorithm "Scheduling Algorithm";

Fig. 10. Second mutation algorithm.

b) Operator of mutation balancing workloads of machines (Fig. 10): In the example of Table XXX, the workload of the critical machine is $W_4 =$ five units of time (M_4). The less loaded machine is M_1 ($W_1 =$ three units). We suppose than the operation $O_{1,1}$ has been chosen randomly among operations executed on M_4. This operation will therefore be assigned to M_1 (see Table XXXI).

Workloads are therefore balanced, and the two machines M_1 and M_4 work during the same working time $W_1 = W_4 =$ four units of time.

These operators of mutation present a new way of application of evolutionary algorithms: it is the way of "genetic manipulations." In genetic biology, these manipulations enable us to generate genetically modified organisms (GMOs). Our method is inspired of this principle and intervenes in the construction

TABLE XXX
S BEFORE MUTATION 2

		M1	M2	M3	M4
J1	O1,1	0	0	0	0, 1
	O2,1	0	0	0	1, 2
	O3,1	2, 5	0	0	0
J2	O1,2	0	0, 1	0	0
	O2,2	0	1, 4	0	0
	O3,2	0	0	0	4, 6
J3	O1,3	0	0	0, 3	0
	O2,3	0	0	0	3, 4

TABLE XXXI
S AFTER MUTATION 2

		M1	M2	M3	M4
J1	O1,1	0, 1	0	0	0
	O2,1	0	0	0	1, 2
	O3,1	2, 5	0	0	0
J2	O1,2	0	0, 1	0	0
	O2,2	0	1, 4	0	0
	O3,2	0	0	0	4, 6
J3	O1,3	0	0	0, 3	0
	O2,3	0	0	0	3, 4

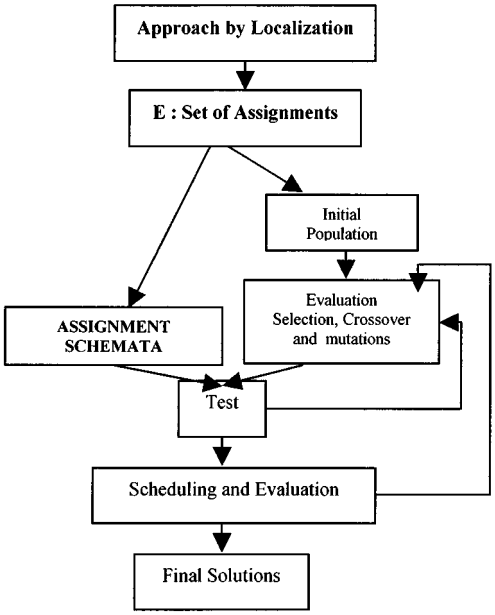

Fig. 11. Controlled genetic algorithm.

phase of the new chromosomes by applying the "artificial mutations" in order to accelerate the convergence and ensure a high quality of final solutions.

D. Controlled Genetic Algorithm (CGA)

The objective considered is to imbricate a double control in the evolutionary approach:

— a control ensured by the schemata assignment S^{ch} (obtained by applying the AL);
— a control ensured by the artificial mutations (genetic manipulations).

(see the flowchart described in Fig. 11).

We can notice that the crossover operator preserves the membership of the new individuals to the assignment schemata. Therefore, the GA will be only controlled by testing new individuals after their mutation in order to reduce the computation time and make the search more efficient.

TABLE XXXII
CGA SOLUTION: MONO-CRITERION EVALUATION

		M1	M2	M3	M4	M5	M6	M7	M8
J1	O1,1	0	0	0	0	0, 3	0	0	0
	O2,1	0	0	0	0	3, 6	0	0	0
	O3,1	0	0	0	0	0	11, 13	0	0
J2	O1,2	0	0	1, 4	0	0	0	0	0
	O2,2	0	0	0	4, 6	0	0	0	0
	O3,2	0	0	0	0	0	0	9,10	0
	O4,2	0	0	0	0	10,14	0	0	0
J3	O1,3	0	0	0	0	0	0	0, 2	0
	O2,3	0	0	0	6, 10	0	0	0	0
	O3,3	10,11	0	0	0	0	0	0	0
J4	O1,4	0	0, 1	0	0	0	0	0	0
	O2,4	0	0	0	0	0	4, 9	0	0
	O3,4	0	0	9, 11	0	0	0	0	0
J5	O1,5	0, 3	0	0	0	0	0	0	0
	O2,5	0	0	0	0	0	0	3, 9	0
	O3,5	0	0	0	0	0	9, 11	0	0
	O4,5	0	0	0	0	0	0	11,14	0
J6	O1,6	0	0	0, 1	0	0	0	0	0
	O2,6	0	0	0	0	0	0	0	1, 5
	O3,6	0	9, 14	0	0	0	0	0	0
J7	O1,7	0	1, 5	0	0	0	0	0	0
	O2,7	0	0	0	0	0	0	0	5,10
	O3,7	0	0	0	10,13	0	0	0	0
J8	O1,8	0	0	0	0	0	0, 4	0	0
	O2,8	0	5, 9	0	0	0	0	0	0
	O3,8	0	0	0	0	0	0	0	10,11
	O4,8	0	0	11,14	0	0	0	0	0

VI. RESULTS GIVEN BY THE CONTROLLED GENETIC APPROACH

We have considered the example of Table XV and we have used the AL to construct the schemata assignment (already presented in Table XX) and generate the initial population. Then, we have applied the controlled genetic algorithm (CGA) with the following parameters:

— population size: $N_{ind} = Cardinal(E) = 100$;
— mutation probability: $P_m = 0.12$;
— crossover probability: $P_c = 0.88$;
— number of generations: $N_g = 500$.

The function "evaluation" used in the CGA has been applied in two different manners.

1) First manner: We use a monocriterion evaluation, so we consider only the makespan according to the following function F1:

$$F1 = \text{makespan}.$$

2) Second manner: We use a multicriteria evaluation according to the following function F2:

$F2(gen) =$ makespan, IF gen is even (gen is the generation index);
$F2(gen) =$ sum of machines workloads, IF gen is odd.

In other words, by evolving a generation to the next, we alternate the two optimization criteria (the makespan and the sum of machines workloads).

A. Results

The application of our controlled evolutionary approach yielded the following results:

1) First Case: Monocriterion Evaluation With the Function F1: The best solution is presented in Table XXXII.

Thus, it enables us to reduce the makespan to 14 units after five generations.

TABLE XXXIII
MONOCRITERION EVALUATION

	Temporal Decomposition	Classic Genetic Method	Approach by Localization	AL + CGA
makespan	19	16	16	14

TABLE XXXIV
MULTICRITERIA EVALUATION

	Temporal Decomposition	Classic Genetic Method	Approach by Localization	AL + CGA	
makespan	19	16	16	15	16
W	91	77	75	79	75

2) Second Case: Multicriteria Evaluation With the Function F2: The best obtained solutions have the same criteria values comparing to the solution given by the AL, we can present the following two solutions.

1) First solution (characterized by the following values):

$$\text{makespan} = 15, \qquad W = 79.$$

This enables us to keep a correct value of machines workloads ($W = 79$) and decline the makespan to 15 units of time.

2) Second solution (characterized by the following values):

$$\text{makespan} = 16, \qquad W = 75.$$

This is the same solution given by the AL.

B. Comparisons

Here, we present the different results obtained by our approach and we show its efficiency compared with the other methods (see Tables XXXIII and XXXIV).

VII. OTHER RESULTS: JOB-SHOP WITH TOTAL FLEXIBILITY

To evaluate the efficiency of our approach, we have chosen to apply it on the example shown in Table XXXV.

This example has been already processed in the literature by two methods:

The first method is the temporal cecomposition [11], [15]. The schedule obtained in this case is characterized by the following values:

$$W = 59, \qquad \text{Max}(W_k) = 16, \qquad C_{\max} = 16.$$

The second method has been developed by our team and it consists to apply classic GAs [9]. The best schedule obtained by this technique is characterized by the following values:

$$W = 53, \qquad \text{Max}(W_k) = 7, \qquad C_{\max} = 7.$$

These values show the efficiency of the genetic approach as compared to that of the temporal decomposition. In fact, the second method enables us to reduce the total machines workloads (53 instead of 59) and to obtain a gain of more than 56% in term of makespan. However, on the opposite, this method is expensive in computation time. Concerning the new AL, the best schedule is obtained for the solution S_{AL} presented in Table XXXVI.

This schedule is characterized by the following values:

Machines workloads (W_k):

$$\{W_1 = 4,\ W_2 = 6,\ W_3 = 5,\ W_4 = 5,\ W_5 = 5,$$
$$W_6 = 5,\ W_7 = 5,\ W_8 = 3,\ W_9 = 4,\ W_{10} = 4\}$$
$$W = 46, \quad \text{Max}(W_k) = 6, \quad C_{\max} = 8.$$

TABLE XXXV
TOTAL FLEXIBILITY

		M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
1	O 1 ,1	1	4	6	9	3	5	2	8	9	5
	O 2 ,1	4	1	1	3	4	8	10	4	11	4
	O 3 ,1	3	2	5	1	5	6	9	5	10	3
2	O 1 ,2	2	10	4	5	9	8	4	15	8	4
	O 2 ,2	4	8	7	1	9	6	1	10	7	1
	O 3 ,2	6	11	2	7	5	3	5	14	9	2
3	O 1 ,3	8	5	8	9	4	3	5	3	8	1
	O 2 ,3	9	3	6	1	2	6	4	1	7	2
	O 3 ,3	7	1	8	5	4	9	1	2	3	4
4	O 1 ,4	5	10	6	4	9	5	1	7	1	6
	O 2 ,4	4	2	3	8	7	4	6	9	8	4
	O 3 ,4	7	3	12	1	6	5	8	3	5	2
5	O 1 ,5	7	10	4	5	6	3	5	15	2	6
	O 2 ,5	5	6	3	9	8	2	8	6	1	7
	O 3 ,5	6	1	4	1	10	4	3	11	13	9
6	O 1, 6	8	9	10	8	4	2	7	8	3	10
	O 2, 6	7	3	12	5	4	3	6	9	2	15
	O 3, 6	4	7	3	6	3	4	1	5	1	11
7	O 1 ,7	1	7	8	3	4	9	4	13	10	7
	O 2 ,7	3	8	1	2	3	6	11	2	13	3
	O 3 ,7	5	4	2	1	2	1	8	14	5	7
8	O 1 ,8	5	7	11	3	2	9	8	5	12	8
	O 2 ,8	8	3	10	7	5	13	4	6	8	4
	O 3 ,8	6	2	13	5	4	3	5	7	9	5
9	O 1 ,9	3	9	1	3	8	1	6	7	5	4
	O 2 ,9	4	6	2	5	7	3	1	9	6	7
	O 3 ,9	8	5	4	8	6	1	2	3	10	12
10	O1,10	4	3	1	6	7	1	2	6	20	6
	O2,10	3	1	8	1	9	4	1	4	17	15
	O3,10	9	2	4	2	3	5	2	4	10	23

TABLE XXXVI
S_{AL}

		M1	M2	M3	M4	M5	M6	M7	M8	M9	M 10
1	O 1 ,1	0, 1	0	0	0	0	0	0	0	0	0
	O 2 ,1	0	1, 2	0	0	0	0	0	0	0	0
	O 3 ,1	0	0	0	5, 6	0	0	0	0	0	0
2	O 1 ,2	2, 4	0	0	0	0	0	0	0	0	0
	O 2 ,2	0	0	0	0	0	0	4, 5	0	0	0
	O 3 ,2	0	0	5, 7	0	0	0	0	0	0	0
3	O 1 ,3	0	0	0	0	0	0	0	0, 3	0	0
	O 2 ,3	0	0	0	3, 4	0	0	0	0	0	0
	O 3 ,3	0	0	0	0	0	0	6, 7	0	0	0
4	O 1 ,4	0	0	0	0	0	0	0, 1	0	0	0
	O 2 ,4	0	2, 4	0	0	0	0	0	0	0	0
	O 3 ,4	0	0	0	6, 7	0	0	0	0	0	0
5	O 1 ,5	0	0	0	0	0	0	0	0	0, 2	0
	O 2 ,5	0	0	0	0	0	2, 4	0	0	0	0
	O 3 ,5	0	4, 5	0	0	0	0	0	0	0	0
6	O 1, 6	0	0	0	0	0	0, 2	0	0	0	0
	O 2, 6	0	0	0	0	0	0	0	0	2, 4	0
	O 3, 6	0	0	0	0	0	0	7, 8	0	0	0
7	O 1 ,7	1, 2	0	0	0	0	0	0	0	0	0
	O 2 ,7	0	0	2, 3	0	0	0	0	0	0	0
	O 3 ,7	0	0	0	7, 8	0	0	0	0	0	0
8	O 1 ,8	0	0	0	0	0, 2	0	0	0	0	0
	O 2 ,8	0	0	0	0	0	0	0	0	0	2, 6
	O 3 ,8	0	6, 8	0	0	0	0	0	0	0	0
9	O 1 ,9	0	0	0, 1	0	0	0	0	0	0	0
	O 2 ,9	0	0	0	0	0	0	5, 6	0	0	0
	O 3 ,9	0	0	0	0	0	6, 7	0	0	0	0
10	O1,10	0	0	1, 2	0	0	0	0	0	0	0
	O2,10	0	0	0	4, 5	0	0	0	0	0	0
	O3,10	0	0	0	0	5, 8	0	0	0	0	0

TABLE XXXVII
S_{CGA}

		M1	M2	M3	M4	M5	M6	M7	M8	M9	M 10
1	O 1 ,1	0, 1	0	0	0	0	0	0	0	0	0
	O 2 ,1	0	0	1, 2	0	0	0	0	0	0	0
	O 3 ,1	0	0	0	2, 3	0	0	0	0	0	0
2	O 1 ,2	2, 4	0	0	0	0	0	0	0	0	0
	O 2 ,2	0	0	0	0	0	0	4, 5	0	0	0
	O 3 ,2	0	0	5, 7	0	0	0	0	0	0	0
3	O 1 ,3	0	0	0	0	0	0	0	0	0	0, 1
	O 2 ,3	0	0	0	0	0	0	0	1, 2	0	0
	O 3 ,3	0	0	0	0	0	0	0	2, 4	0	0
4	O 1 ,4	0	0	0	0	0	0	0, 1	0	0	0
	O 2 ,4	0	0	0	0	0	0	0	0	0	1, 5
	O 3 ,4	0	0	0	5, 6	0	0	0	0	0	0
5	O 1 ,5	0	0	0	0	0	0	0	0	0, 2	0
	O 2 ,5	0	0	0	0	0	0	0	0	2, 3	0
	O 3 ,5	0	0	0	6, 7	0	0	0	0	0	0
6	O 1, 6	0	0	0	0	0	1, 3	0	0	0	0
	O 2, 6	0	0	0	0	0	0	0	0	3, 5	0
	O 3, 6	0	0	0	0	0	0	6, 7	0	0	0
7	O 1 ,7	1, 2	0	0	0	0	0	0	0	0	0
	O 2 ,7	0	0	2, 3	0	0	0	0	0	0	0
	O 3 ,7	0	0	0	0	0	3, 4	0	0	0	0
8	O 1 ,8	0	0	0	0	0, 2	0	0	0	0	0
	O 2 ,8	0	2, 5	0	0	0	0	0	0	0	0
	O 3 ,8	0	5,7	0	0	0	0	0	0	0	0
9	O 1 ,9	0	0	0	0	0	0,1	0	0	0	0
	O 2 ,9	0	0	0	0	0	0	5, 6	0	0	0
	O 3 ,9	0	0	0	0	0	6, 7	0	0	0	0
10	O1,10	0	0	0, 1	0	0	0	0	0	0	0
	O2,10	0	0	0	1,2	0	0	0	0	0	0
	O3,10	0	0	0	0	2, 5	0	0	0	0	0

TABLE XXXVIII
COMPARISONS

	Temporal Decomposition	Classic Genetic Method	Approach by Localization	Localization + CGA
makespan	16	7	8	7
W	59	53	46	45
Max (W_k)	16	7	6	5

The AL enables us to decrease the total machines workloads to 46 units of the time and yields a makespan of eight units of time. The maximal workload is reduced to six units, which presents a satisfactory efficiency compared with the GAs.

Finally, the application of our controlled evolutionary approach (by the assignment schemata and the artificial genetic mutations) has given the solution S_{CGA} (Table XXXVII) characterized by the following values:

$$W = 45, \quad \text{Max}(W_k) = 5, \quad C_{\max} = 7.$$

All results can be summarized in Table XXXVIII.

Values of the different criteria given by each method show the efficiency of the controlled genetic approach. In fact, this method enables us to have good results (the optimal value of makespan in this case) in a polynomial computation time. This efficiency is explained by the judicious choice of the search zone (AL) and by the contribution of artificial mutations in the optimization of solutions.

VIII. CONCLUSION

In this paper, we have proposed an efficient methodology for flexible job-shop scheduling problems.

This method enables us to construct solutions with good quality in a reasonable computation limit.

Our first step was to apply the AL to solve the resource allocation problem and generate the assignment schemata.

Our second step was to apply a controlled evolutionary algorithm. The initial population is constructed starting from the set of assignments found in the first stage.

The evolution of generations will be controlled in the two following levels.

— First level: we test if the new individual respects the model imposed by the assignment schemata.

— Second level: we apply artificial mutations (genetic manipulations) in order to reduce the blind aspect of genetic operators.

As research continues into CGA and AL, the idea of applying the schemata theory becomes more important. In fact, this idea will be effective for a wide range of problem instances for which it was designed. Also, it will amenable to the modification or extension to solve others different types of problems. The fundamental building scheme used in this paper represents the problem at the correct level of abstraction ranging from a completely specified point in the problem types or a set of aggregate qualities that describe a family of a possible good solutions. Also, the study of the other considerations in the multiobjective optimization (such as Pareto optimization [22] [23] or fuzzy evaluation [24]) seems an interesting subject that can enrich the proposed approach and give scientific benefits.

ACKNOWLEDGMENT

This work is integrated in the group “TACT” of a regional program entitled “MOST” (research group on integrated manufacturing and man–machine systems). This program is supported in France by the “Conseil Régional du Nord Pas de Calais” and the “FEDER.” It involves several laboratories in the north of France. One of the goals of the program is to increase the competitiveness of industries by designing new tools and methods.

The authors would like to thank the referees for their constructive comments.

REFERENCES

[1] Ghotha, “Scheduling problems” (in French), *Oper. Res.*, vol. 27, no. 1, pp. 77–150, 1993.

[2] M. R. Garey, R. L. Graham, and D. S. Johnson, “Performance guarantees of scheduling algorithms,” *Oper. Res.*, vol. 26, pp. 3–21, 1978.

[3] J. Turek, J. L. Wolf, K. R. Pattipati, and P. S. Yu, “Scheduling parallelizable tasks: Putting it all on the shelf,” in *Proc. ACM Sigmetrics Performance Evaluation Review*, Newport, RI, June 1–5, 1992.

[4] J. Carlier, “An algorithm for solving the job-shop problem,” *Manage. Sci.*, vol. 35, pp. 164–176, 1989.

[5] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer-Verlag, 1992.

[6] M.-C. Portmann, “Genetic algorithms and scheduling: A state of the art and some proposition,” in *Proc. Workshop Production Planning and Control*, Mons, Belgium, Sept. 9–11, 1996, pp. i–xxiv.

[7] F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research*. San Francisco, CA: Holden-Day, 1967.

[8] M. S. Fox, *Constraint-Directed Search: A Case Study of Job-Shop Scheduling*. San Mateo, CA: Morgan Kaufmann, 1987.

[9] K. Mesghouni, “Application des algorithmes évolutionnistes dans les problèmes d’optimization en ordonnancement de production,” Ph.D. dissertation, USTL 2451, France, 1999.

[10] P. Lopez and F. Roubellat, *L'ordonnancement de la production*, France, 2001.
[11] F. Chetouane, "Ordonnancement d'atelier à tâches généralisées, perturbations, réactivité," DEA Rep., Polytech. Nat. Inst. Grenoble, Grenoble, France, 1995.
[12] M. T. Isaai and M. G. Singh, "An object-oriented constraint-based Heuristic for a class of passengers train scheduling problems," *IEEE Trans. Syst., Man, Cybern. C*, vol. 30, pp. 12–21, Feb. 2000.
[13] T. Yamada and R. Nakano, "A genetic algorithm application to large-scale job-shop problems," in *Parallel Problem Solving From Nature II*, R. Manner and B. Manderick, Eds. Amsterdam, The Netherlands: Elsevier, 1992, pp. 281–290.
[14] L. Davis, *Handbook of Genetic Algorithm*. New York: Van Nostrand Reinhold, 1990.
[15] K. Mesghouni, S. Hammadi, and P. Borne, "Evolution programs for job-shop scheduling," in *Proc. IEEE Syst., Man, Cybern. Conf.*, vol. 1, Orlando, FL, Oct. 12–15, 1997, pp. 720–725.
[16] F. Glover, J. Kelly, and M. Laguna, "Genetic algorithms and tabu search: Hybrid for optimization," Grad. Sch. Business, Univ. Colorado, Boulder, July 1992.
[17] L. Davis, "Job-shop scheduling with genetic algorithms," in *Proc. Int. Conf. Genetic Algorithms and Their Applications*. San Mateo, CA: Morgan Kaufmann, 1985, pp. 136–140.
[18] Ono, "A genetic algorithms for job-shop scheduling problems using job-based order crossover," in *Proc. ICEC*, 1996, pp. 574–552.
[19] A. Vignier and G. Venturini, "Resolution of hybrid flow shop with a parallel genetic algorithm," in *Proc. EURO Working Group on PMS*, Scientific Publisher Own Pan, Posnan, Poland, Apr. 11–13, pp. 258–261.
[20] I. Charon, A. Germinated, and O. Hudry, *Méthodes d'Optimization Combinatoires*. Paris, France: Masson, 1996.
[21] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[22] R. Sarker, H. A. Abbas, and C. Newton, "Solving multiobjective optimization problems using evolutionary algorithm," in *Proc. CIMCA Int. Conf.*, Las Vegas, NV, July 9–11, 2001.
[23] E. Zitzler, K. Deb, L. Thiele, C. Coello, and D. Corne, *Evolutionary Multi-Criterion Optimization*, ser. Lecture Notes in Computer Science. New York: Springer-Verlag, 2001, vol. 1993.
[24] I. Kacem, S. Hammadi, and P. Borne, "Pareto-optimality approach for flexible job-shops scheduling problems: Hybridization of evolutionary algorithms and Fuzzy Logic," *J. Math. Comput. Simul.*, 2002.

Imed Kacem was born in Eljem, Tunisia, in 1976. He received the Eng.Dipl. degree from the ENSAIT, France, and the M.Sc. degree in control and computer sciences from the University of Lille 1, Villeneuve d'Ascq, France, both in 2000. He is currently pursuing the Ph.D. degree in automatic and computer science at "Laboratoire d'Automatique et Informatique de Lille" of Ecole Centrale de Lille, Villeneuve d'Ascq.

His research is related to the evolutionary optimization methods for discrete events system, computer science, and operational research.

Mr. Kacem was selected among the young Tunisian engineers of the Grandes Ecoles to receive the Tunisian Presidential Prize for 2001. He has served as a referee for the International CIMCA'01, the International LWATIC'01, and IEEE/SMC'02 Conferences and IEEE/SMC Transactions.

Slim Hammadi (M'92–SM'01) received the Ph.D. degree from Ecole Centrale de Lille, Villeneuve d'Ascq, France, in 1991.

Currently, he is an Associate Professor of production planning and control at Ecole Centrale de Lille. His research is related to production control, production planning, computer science, and computer integrated manufacturing.

Dr. Hammadi has served as a referee for numerous journals including IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS. He was co-organizer of a Symposium (IMS) of the IMACS/IEEE SMC Multiconference CESA'98 held in Hammamet, Tunisia, in April 1998. He has organized several invited sessions in different SMC conferences where he was session chairman.

Pierre Borne (F'96) is with Ecole Centrale de Lille, Villeneuve d'Ascq, France, where he is "Professor de Classe Exceptionnelle," Director of Research, and Head of the Automatic Control Department. His activities concern automatic control, robust control, and optimization in planning and scheduling, including implementation of fuzzy logic, neural nets, and genetic algorithms. He is author or coauthor of more than 250 journal articles, book chapters, a scientific dictionary and communications in international conferences, and 14 books on automatic control.

Dr. Borne is a Fellow of the Russian Academy of Non-Linear Sciences. He has been President of the IEEE/SMC Society (2000–2001) and has been IMACS Vice President (1988–1994). He is Chairman of the IMACS Technical Committee on Robotics and Control Systems. He received the IEEE Norbert Wiener Award in 1998. He is listed in *Who's Who in the World*. In 1997, he was nominated at the "Tunisian National Order of Merit in Education" by the President of the Tunisian Republic and in 1997, he was nominated Honorary Member of the IMACS board of directors. In 1999, he was promoted to "Officier dans l'ordre des Palmes Académiques" in France. In 2000, he received the IEEE Third Millennium Medal.