A new manner of crossing in the genetic algorithm for resolving Job Shop Problem (JSP)

SAID BOURAZZA
Mathematics Department
Faculty of Science, Jazan University,
P. O. Box 2097 Jazan City,
KINGDOM OF SAUDI ARABIA.

Abstract: - The minimization of the makespan of the job shop problem with J jobs and M machines is NP-hard problem. To resolve it we apply a genetic algorithm [1,2]. We use a real coding for the representation of chromosomes. The originality of our variant lies by the choice of two effective crossover operators and one for mutation, As well as their respective probability, determined by numeric simulations on several examples known in the literature. We compare our results on "OR library" benchmarks [3] with Adamas and al. [4], Ombuki and al [5] and Yamada and al [6] results.

Key-Words: - Job Shop Problem, Genetic Algorithm.

Received: January 2, 2020. Revised: April 8, 2020. Accepted: April 23, 2020. Published: April 30, 2020.

1 Introduction

The job shop problem of type (J, M) contains M machines serving for producing J jobs. Every job consists of a number of operations which can be executed in a prescribed order. Every operation can be executed only on a single machine and every machine can realize only one operation at the same moment. The objective is to find a practicable scheduling which minimizes the total duration of the production of all the jobs (the minimization of "Makespan"). It is the difference between the date of the end of the last operation of all the jobs and the date of the beginning of the first operation of this set. It is a problem of optimization NP-hard. To resolve it we apply a genetic algorithm [1, 2]. We use a real coding for the representation of chromosomes. The originality of our variant lies by the choice of two effective crossover operators and one for mutation, As well as their respective probability, determined by numeric simulations on several examples known in the literature. We compare our results on "OR library" benchmarks [3] with Adamas and al. [4], Ombuki and al [5] and Yamada and al [6] results.

2 Problem Formulation

First, The Job-Shop is a scheduling problem in workshops. it contains two classical problems of the combinatorial optimization : the problem of affectation and the problem of scheduling. The problem can be characterized as follows:

- A set of M machines and a set of J jobs.
- Each job must be processed on each machine in the order given in a predefined technological sequence of machines. Each job consists of a fixed predefined technological operations $O_{i,j}$.
- $O_{i,j}$ representing the jth operation of the job i . This operation requires a processing time $P_{i,j}$.
- $O_{i,j}$ require to be realized on one machine.

It is subject to the following constraints:

- The machines are independents some of the others;
- Each machine can process only one job at a time;
- Each operation started cannot be stopped in its processing time ;
- The jobs are independents some of the others.

The time required to complete all jobs is called the makespan which is denoted as Cmax . A schedule is a set of completion times for each operation that satisfies above constraints. the purpose of the scheduling is to minimize Cmax.

2.1 Subsection

When including a subsection you must use, for its heading, small letters, 12pt, left justified, bold, Times New Roman as here.

2.1.1 Sub-subsection

When including a sub-subsection you must use, for its heading, small letters, 11pt, left justified, bold, Times New Roman as here.

3 Genetic algorithm

The genetic algorithm belongs to the family of the evolution algorithm. They are inspired by the creed of the nature: the survival for the most adapted individuals.

They aroused the interest of the researchers, started by Holland [7] who developed the fundamental principles, passing by Goldberg [8] who used them to resolve concrete problems of optimization. Other researchers followed this way notably Davis [9], Mahfoud ([10] and [11]), Michalewicz ([12] and [13]), etc.

3.1. Our Genetic algorithm

The general framework for our genetic algorithm expressed as:

```
Generate the initial population called P1 of size N
and calculate the fitter of P1
Repeat
P2 and P3 are empties set
for 1 to N do
Select randomly one individual from P1
Crossing this individual with the fitter
Add the two child into P2
od
for 1 to N do
Select randomly one individual from P1
Add Mutating individual into P3
od
Ranking P1 and P2
P1=(N/2 of fitter individuals of P2) U (N/2 of fitter
individuals of P2)
Calculate fitter of P1
Until termination, where all individuals of P1 have
the same fitness
```

3.1.1. Genetic encoding of a schedule

An example of a job shop problem with 3 jobs and 3 machines is given in Table [1]. The data include the technological sequence of machines for each job with the processing times in parentheses. In the table, operations for job 0, for example, are processed in the order of O00 , O_{01}, O_{02} ; i.e., job 0 is first processed on machine 2 with its processing time 3, and then processed on machine 1 with its processing time 6, and then processed on machine 0 with its processing time 2.

Job i	O_{ij} ➔ Machine (processing time)		
Job 0	O_{00} ➔ 2 (3)	O_{01}➔1 (6)	O_{02}➔0 (2)
Job 1	O_{10} ➔0 (2)	O_{11} ➔1 (5)	O_{12}➔2 (2)
Job 2	O_{20} ➔1(1)	O_{21} ➔0 (2)	O_{22}➔2 (1)

Table1: An example of the job shop scheduling problem with 3 jobs and 3 machines.

Each arrow represents the technological sequence of machines for each job with the processing times in parentheses. In the first, the sum of all operations in this example is 9. We grant to every operation a number between 0 and 8 by respecting the order of the operations for every job. So, three operations of the job 0 will have for code 0, 1 and 2. For the job 1, the operations are numbered 3, 4 and 5. And finally, the operations 6, 7 and 8, in this order, constitute the job 2.

A chromosome is coded by a table of integers from 0 to 8 which respects the order between the operations of the same job. So 0 corresponds to the operation O_{00}, 5 corresponds at O_{11}.... The table of integers [0, 3, 1, 4, 6, 7, 2, 5, 8] corresponds to a feasible scheduling of the operations on machines. It is not the case for [0, 3, 1, 5, 4, 2, 6, 7, 8]. Indeed, the operation 5 precedes the operation 4 what does not respect the order predefined by the technological sequence of the machines for the job 1.

3.1.2. Initial population

The initial population is randomly generated. A chromosome is constituted by M*J genes corresponding to the operations. We choose at random one job. In this job, we select one operation not yet marked. This operation will be placed in the chromosome. This operation will be distinguished and marked. If the same job is chosen after that, the successor of the old operation will be placed in the chromosome and will be marked. If all the operations of a job were placed in the chromosome then this job will be marked so that it is not any more chosen afterward. The procedure which generates a chromosome is clarified below:

```
integers j, k, l, idx, tmp, compteur
integers J, M
Tab, chrom : two table of integers with length J * M
job : Table of integers with length J
for j from 0 to (J * M)
   | Tab[j] = j
end for j
```

```
for j from 0 to J
   | job[j] = j
end for j
j = 0
While (j < J * M) do
| tic: l = random number between 0 and (J 1)
| If(job[l]==1) goto tic
| End if
| compteur = 0
| for idx from l * M to ((l+1) * M)1
| | if ( Tab[idx] != 1 ) chrom[j] = Tab[idx]
| | | Tab[idx] = 1
| | | j = j+1
| | | exit from this lap of idx
| | |
| | else compteur = compteur +1
| | | if(compteur == M) job[l] = 1
| | | | goto tic
| | | end if
| | end if
| end for
end while
```

3.1.3.Calcul of Makespan

The makespan is the duration operating total between the date of the end of the last operation of the scheduling and the date of the beginning of its first operation. To calculate the makespan, it is necessary to know the date of the beginning of every operation on its machine of production. We define respectively P_{ij} , T_{ij} and F_{ij} , processing time, the date of the beginning and the end of every operation O_{ij} . P_{ij} is obtained from the problem; F_{ij} is given by the follow formula : $F_{ij} = T_{ij} + P_{ij}$; T_{ij} is given by : $T_{ij} = \max \{ F_{i(j-1)} , T_{hl} + P_{hl} \}$ With O_{hl} is the previous operation of O_{ij} on the machine M_{ij} and $F_{i(j-1)}$ is the date of the end of the operation, which precedes O_{ij} , in the job i . We found two cases:

If O_{ij} is the first operation of the job i then $F_{i(j-1)} = 0$.

If O_{ij} is the first operation executed on the M_{ij} machine then $F_{hl} = T_{hl} + P_{hl}$ is replaced with 0.

The makespan is the maximum of the dates of the end of the last operations for every job:

$\text{Cmax} = \max \{F_{hl}\}$.

3.1.4. Selection method

The purpose of this method is to choose the individuals subject to the crossover operator and for mutation operator. For applying the crossover operator, we take the best individual and we picked randomly one from the former population. And for mutation, we choose randomly one individual from the former population.

3.1.5. Crossover operator

We used two operators of crossover in the same time. The first is edrx crossover given by Whitley [15], and the second is Cedrx.

3.1.5.1.Edge recombination operator (edrx by Whitley [15]): From the two parents, we construct a table that gives for each gene his adjacent neighbors. For example, the gene 1 has for neighbors in parent 1 the genes 2 and 7, and in the parent 2 the genes 5 and 3.

		*					
parent1	1	2	3	4	5	6	7
parent2	7	5	1	3	2	6	4

Neighbor table	**1**	**2**	**3**	**4**	**5**	**6**	**7**
	2	1	2	3	4	5	6
	7	3	4	5	6	7	1
	5	6	1	6	7	2	4
	3			7	1	4	5

child1	2	3	1	5	4	7	6
child2	5	1	3	2	6	4	7

TABLE 2: EDGE RECOMBINATION OPERATOR (EDRX).

To get the child1, we choose randomly the first gene of the child1 from the parent 1. His successor will be selected from the table of neighbors of the first gene with respecting the following rules:

The gene that will be elected must have the smallest number of distinct neighbors.

If we find several genes that verify the condition above, then we will choose between them one randomly.

The elected gene is placed in the child1 and is removed immediately from the table of neighbors. We restart until the total construction of the child1.

3.1.5.2. Complementary Edge recombination operator (Cedrx): This operator is performed as like as edrx in the construction of the neighbor table. But, for making child, we choose the successor of one gene outside of his column in the neighbor table and not yet used in child.

3.1.6. JSP mutation operator:

It avoids to establishing uniform populations unable to evolve. It consists in modifying the values of the genes of chromosomes from those of a single parent. If one chromosome is choosing for the mutation operator, we determined a machine which Fij equal to

Cmax. On this machine, we looking for job corresponding to the last operation. We choose randomly another job to exchange the locations of all operations belong in this two jobs with saving the order defined by technological sequence of the two jobs. This operator is described below:

```
integers machcritiq, tmp,job, job1, k1, k2
integers indice1, indice2
integers dernierjob, dernierop
integers i, j
machcritiq = the critical machine who realized the
    makespan
dernierop = the last operation executed on
    machcritiq machine
dernierjob = dernierop /M
Do job = random between 0 and (J 1)
While(job == dernierjob)
For i from 0 to (M 1)
| k1 = job * i
| k2= dernierjob*i
| For j from 0 to (J*M 1)
| | if (k1 == chrom[j]) indice1 = j
| | endif
| | if (k2 == chrom[j]) indice2 = j
| | endif
| end for j
| tmp = chrom[indice1]
| chrom[indice1] = chrom[indice2]
| chrom[indice2] = tmp
end for i
```

3.2. Mechanism to repair the generated individuals:

The chromosomes obtained by the crossover operator are not always feasible. We used a repair machanism of this childs to remove this anomalies. This mechanism respects the technological sequence of all jobs. It's rectifies this chromosomes.

```
Chrom : table of integers with length M * J
integers : tmp, i, j, k
tab : table of integers with length J * M
for i from 0 to J * M
| tmp = chrom[i]
| tab[i] = The job container the operation tmp
end for i
for i from 0 to (J1 )
| k = i * M
| for j from 0 to ( J*M 1)
| | if (tab[j] == i) chrom[j] = k
| | | k = k+1
| | endif
| end for j
end for i
```

4. Numerical results

We write our program in the C language on the Intel(R) Pentium(R) 4 PC machine with CPU 2593.547 Mhz. This machine used Linux Mandrake operating system. We use benchmarks [3].

Our genetic algorithm is based on:

- Size of the population 200,
- Two Crossover operator,
- The mechanism of repair,
- Jsp mutation operator.

We recapitulate the results, in the following table 2:

Inst. [3]	**J**	**M**	**M.S.C.**	**sGA**	**gkGA**	**LSGA [5]**	**our GA**
La01	1	5	666	667	677	666	666
La05	1	5	593	593	593	593	593
La06	1	5	926	926	926	926	926
La07	1	5	890	891	890	890	890
La08	1	5	863	863	863	863	863
La09	1	5	951	951	951	951	951
La10	1	5	958	958	958	958	958
La11	2	5	1222	1222	124	1222	1222
La12	2	5	1039	1039	103	1039	1039
La13	2	5	1150	1150	115	1150	1150
La14	2	5	1292	1292	129	1292	1292
La15	2	5	1207	1256	130	1207	1207

The results obtained by Ombuki et al. and our results based on benchmarks obtained from "OR library"

The column M.S.C. give the best fitness knowing for this benchmarks ;

The column sGA give thes values obtained by Ombuki et al [5] using genetic algorithm based only on mutation operator ;

The column gkGA makes references to the values found by Ombuki et al. [13] ;

The column LSGA give the new values obtained by Ombuki et al [5].

The results find by our algorithm are equal with LSGA and better than the other algorithms "sGA" and "gkGA".

5. Conclusion

In this paper, we present a new variant of genetic algorithm. The peculiarity of this variant lies in: The way of coding the chromosomes (the representation of the individuals) which contributed effectively to the decrease of the fitness on one hand and to respect the scheduling of the operations of every job on the other hand; Cedrx is new operator of crossover not allowed to lost quickly the diversity of individuals in population. We used in the same time two operators of crossover, We use two intermediate populations, one for the individuals obtained with crossing and second for this obtained with mutation. Its allowed to take out of local minimum and to obtain the best results, The numerical experiments showed the efficiency and the robustness of our algorithm. Indeed, in most of the cases, he gives the best values of the objective function till now published.

References:

[1] S. Bourazza: “Evaluation of the Operators of the Genetic Algorithm in Application of the Traveling Salesman Problem”. Int. J. of Adv. Res. 6 (2). 42-52, 2018. (ISSN 2320-5407)

[2] S. Bourazza, A. Yassine, and S. Gueye : Un algorithme génétique pour un problème d'ordonnancement des véhicules dans une usine, 12èmes Journées MODE SMAI, Université du Havre, Mars 2004..

[3] OR Library site : http://www.brunel.ac.uk/depts/ma/research/jeb / info.html.

[4] J. Adams, E. Balas and D. Zawack : The shifting bottleneck procedure for job shop scheduling. Journal Manage. Sci., vol. 34, n 3, 1988.

[5] B. Ombuki and M.Ventresca : Local search algorithms for job shop scheduling problem. Technical report, november 2002.

[6] T. Yamada and R. Nakano : Job Shop Scheduling. Genetic Algorithms in Engineering Systems, IEE control Engineering series 55, pp. 134160, 1997.

[7] J. H. Holland. Adaptation in Natural and Artificial Systems. The University of Michigan, 1975.

[8] D. Goldberg. Genetic Algorithm in Search, Optimization, and Machine Learning. Addison Wesley, 1989.

[9] L. Davis, D. Orvosh, A. Cox and Y. Qiu. A Genetic Algorithm for Survivable Network Design. ICGA 1993: 408415

[10] S.W. Mahfoud and D.E. Goldberg. A Genetic Algorithm for Parallel Simulated Annealing. eds., Parallel Problem Solving from Nature 2, Elsevier,1992, 301310.

[11] S.W. Mahfoud and D.E. Goldberg. Parallel recombinative simulated annealing: a genetic algorithm. Parallel Computing 21, 1995, 128.

[12] Z. Michalewicz and D.B. Fogel. How to solve it: Modern heuristics. Springer Verlag, 2000.

[13] B.Ombuki, M. Nakamura, and K.Onaga, An Evolutionary Scheduling Scheme Based on gkGA Approach for the Job Shop Scheduling Problem, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science, Vol. E81A, N0.6, 1988.

[14] L. Davis. Applying Adaptive Algorithms to Epistatic Domains.In Proc. International Joint Conference on Artificial Intelligence, 1985.

[15] D. Whitley, T. Starkweather, and D. Fuquay. Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator. In Proc. Third Int'l. Conference on Genetic Algorithms and their Applications. J. D. Shaeffer, ed. Morgan Kaufmann, 1989.