Springer

KSME

Journal of Mechanical Science and Technology 29 (3) (2015) 1273~1281
www.springerlink.com/content/1738-494x
DOI 10.1007/s12206-015-0242-7

A new genetic algorithm for flexible job-shop scheduling problems †

Imen Driss*, Kinza Nadia Mouss and Assia Laggoun

Automatics & Manufacturing Engineering Laboratory, University of Batna, 1st Street Chahid Boukhlouf, Batna, Algeria

(Manuscript Received August 18, 2014; Revised November 25, 2014; Accepted December 2, 2014)

Abstract

Flexible job-shop scheduling problem (FJSP), which is proved to be NP-hard, is an extension of the classical job-shop scheduling problem. In this paper, we propose a new genetic algorithm (NGA) to solve FJSP to minimize makespan. This new algorithm uses a new chromosome representation and adopts different strategies for crossover and mutation. The proposed algorithm is validated on a series of benchmark data sets and tested on data from a drug manufacturing company. Experimental results prove that the NGA is more efficient and competitive than some other existing algorithms.

Keywords: FJSP; Scheduling; Genetic algorithm; Chromosome representation

1. Introduction

Job-shop scheduling problem (JSP) is a branch of production scheduling and combinatorial optimization problems. The classical JSP consists of scheduling a set of jobs on a set of machines under constraint; each job has a specified processing order. The flexible job-shop problem (FJSP) is an extension of the classical job-shop problem in which each operation must be processed on a machine chosen among a set of available ones.

In FJSP, the problem of scheduling jobs involves the following sub-problems: assigning the operation to a machine (routing problem) and sequencing the operations on machines (sequencing problem) to minimize the performance indicators. Therefore, the combination of two decisions presents additional complexity. Thus, FJSP is NP-hard because it is an extension of JSP that is proven to be NP-hard [1].

Over the years, metaheuristics has been used to solve FJSP, specifically through tabu search, simulated annealing, genetic algorithm, and particle swarm optimization.

In this research work, we propose a new genetic algorithm (NGA) to solve FJSP to minimize the makespan. We create a new chromosome representation called "permutation job". This method enables us to find a new coding scheme for individual jobs, which considers all constraints of FJSP. At the same time, we employ different strategies for crossover and mutation operators. Computational results show that the proposed algorithm is effective.

*Corresponding author. Tel.: +213 33803396, Fax.: +213 33803396
E-mail address: Idrissamina@hotmail.fr
† Recommended by Editor Haedo Jeong

This paper is organized as follows. The problem definition and formulation are presented in Sec. 2. Related studies are reviewed in Sec. 3. The NGA is explained in Sec. 4. Computational results are presented in Sec. 5. Conclusion and future research directions are provided in Sec. 6.

2. Definition and formalization problem

2.1 Problem description

We focus on FJSP composed of the following elements:

- Jobs. $J = \{ J_1 \ldots\ldots J_n \}$ is a set of *n* jobs to be scheduled. Each job J_i consists of a predetermined set of operations. O_{ij} is the operation *j* of job J_i. All jobs are released at time 0.
- Machines. $M = \{ M_1 \ldots\ldots M_m \}$ is a set of *m* machines. Each machine can process only one operation at a given time, and each operation can be processed without interruption. All machines are available at time 0.
- Flexibility. FJSP is classified into two types as follows [2]:
  - Total FJSP (T-FJSP): Each operation can be processed on any machine among *M* existing machines on the shop floor.
  - Partial FJSP (P-FJSP): Each operation can be processed on one machine of a subset of *M* existing machines on the shop floor.
- Constraints: Rules that limit the possible assignments of the operations. They can be classified mainly into the following conditions:
  - Each operation can be processed by only one machine at a time (disjunctive constraint).

Table 1. Processing time table of an instance of P-FJSP.

Job	Operation	M_1	M_2	M_3	M_4	M_5
J_1	O_{11}	2	6	5	3	4
	O_{21}	-	8	-	4	-
J_2	O_{21}	3	-	6	-	5
	O_{22}	4	6	5	-	-
	O_{23}	-	7	1	5	8

- Each operation, which has started, runs to completion (non-preemption condition).
- Each machine performs operations one after another (capacity constraint).
- Although no precedence constraints exist among operations of various jobs, the predetermined sequence of operations for each job forces each operation to be scheduled after all predecessor operations (precedence/conjunctive constraint).
- The machine constraints emphasize that the operations can be processed only by the machine from the given set (resource constraint).

• Objective: To find a schedule that requires minimum time to complete all operations (minimum makespan).

To simplify the presentation of the algorithm, we designed a sample instance of FJSP that is used in this paper. Table 1 provides the dataset of P-FJSP, including two jobs operated on five machines where rows correspond to operations and columns correspond to machines. Each cell denotes the processing time of that operation on the corresponding machine.

2.2 Problem formulation

Some symbols used in our paper are listed as follows:

Ω : Set of all machines
n : Number of total jobs
m : Number of total machines
i : Index of i^{th} job
j : Index of j^{th} operation of job j_i
J_{io} : Number of total operations of job J_i
O_{ij} : j^{th} Operation of job j_i
Ω_{ij} : Set of available machines of O_{ij}
p_{ijk} : Processing time of operation O_{ij} on machine k
S_{ijk} : Start time of operation O_{ij} on machine k
E_{ijk} : End time of operation O_{ij} on machine k
$L=\sum_1^n J_{io}$: Sum of all operations of all jobs
H : Very large positive integer

Objective function

$$\text{Minimize}\left[Max\left(C_{1J_1}, C_{2J_2}, \ldots C_{nJ_n}\right)\right]. \quad (1)$$

Subject to

$$C_{ij} - S_{ij} - \sum_{k:O_{ij}\in\Omega_{ij}}\left(P_{ijk}.X_{ijk}\right)=0 \quad \forall i,j \quad (2)$$

$$C_{i`j`} - C_{ij} + H\left(1-Y_{ij i`j`k}\right) + H\left(1-X_{ijk}\right) + \left(1-X_{i`j`k}\right) \geq P_{i`j`k} \quad (3)$$

$$C_{i`j} - C_{ij} + H\left(Y_{ij i`j`k}\right) + H\left(1-X_{ijk}\right) + \left(1-X_{i`j`k}\right) \geq P_{ijk} \quad (4)$$

$$S_{ij} \geq 0 \ \forall i,j \quad (5)$$

$$S_{ij+1} - C_{ij} \geq 0 \forall i,j \ , j = 1 \ldots J_{i-1} \quad (6)$$

$$\sum_{k:O_{ij}\in\Omega_{ij}}\left(X_{ijk}\right)=1, \quad \forall i,j \quad (7)$$

$$X_{ijk} = \begin{cases} 1 \ if \ operation \ O_{ij} is \ assigned \ to \ machine \ k \\ 0 \ otherwise \end{cases} \quad (8)$$

$$Y_{ij i`j`k} = \begin{cases} 1 \ if operation \ O_{ij} procede \ O_{i`j} on \ machine \ k \\ 0 \ otherwise \end{cases} \quad (9)$$

Objective Eq. (1) minimizes. The constraint set Eq. (2) imposes that the difference between the completion time and the starting time of an operation is equal to its processing time on the assigned mach. Constraint sets Eqs. (3) and (4) ensure that no two operations can be processed simultaneously on the same machine. This disjunctive constraint Eq. (3) becomes inactive when $Y_{ij i`j`k}= 0$ and the disjunctive constraint Eq. (4) becomes inactive when $Y_{ij i`j`k}= 1$. Constraint set Eq. (5) ensures that the start time of an operation is always positive. Constraint set Eq. (6) represents the precedence relationship among various operations of a job. Constraint set Eq. (7) imposes that an operation can be assigned to only one machine [16].

3. Related works

Bruker and Schlie [3] were the first to consider this problem by developing a polynomial algorithm to solve the FJSP with two jobs. Brandimarte [4] was the first to apply the decomposition approach to the FJSP by solving the routing subproblem using some existing dispatching rules and then focusing on the scheduling sub-problem using a tabu search heuristic. Paulli [5] applied a hierarchical approach and Mesghouni et al. [6] were the first to model GA known as parallel job representation to solve FJSP. Chen et al. [7] proposed a GA that uses an A-B string representation to solve FJSP for minimum makespan time criterion. Kacem, Hammadi, and Borne [8] proposed a genetic algorithm controlled by the assignment model generated by the approach of localization (AL) to mono-objective and multi-objective FJSP. Ho and Tay [9] proposed a GA based tool called GENACE to solve the FJSP for minimum makespan time criterion. Zhang and Gen [10] proposed a method called multistage-based genetic algorithm to solve FJSP. Mehrabad and Fattahi [11] presented a mathematical model and tabu search algorithm to solve FJSP with a sequence dependent on setups to minimize the makespan. Ho et al. [12] proposed an architecture for learning and evolving of flexible job-shop schedules for minimum makespan criterion called learnable genetic architecture (LEGA), a generalization of their previous approach GENACE. Gao and Gen

[13] developed a hybrid optimization strategy for multi-objective FJSP (min makespan, min maximal machine workload, and min total workload) by combining the genetic algorithm and bottleneck shifting. Tay and Ho [14] proposed a genetic programming-based approach for evolving effective composite dispatching rules to solve the multi-objective FJSP. Girish and Jawahar [15] proposed a GA for the FJSP for minimum makespan time criterion. Ponnambalam et al. [16] employed a genetic algorithm (GA)-based heuristics that have adopted the Giffler and Thompson (GT) procedure, an efficient and active feasible schedule for makespan time criterion. Giovanni and Pezzella [17] proposed an improved GA to solve the distributed FJSP to minimize the makespan. Sun et al. [18] examined the FJSP based on a modified GA. Motaghedi et al. [19] presented an effective hybrid genetic algorithm to solve the multi-objective FJSP. Guohui et al. [20] proposed an effective genetic algorithm to solve the FJSP to minimize the makespan criteria. Zhang et al. [21] proposed a GA with tabu search procedure for FJSP with transportation constraints and bounded processing times to minimize the makespan and the storage of solutions. Chen et al. [22] proposed a GA and grouping GA for JSP with parallel machines and reentrant process.

4. A new genetic algorithm for flexible job-shop scheduling problem

4.1 Basis of genetic algorithm

GAs are search methods based on principles of natural selection and genetics [23, 24]. The interest in heuristic search algorithms with underpinnings in natural and physical processes began as early as the 1970s when Holland [25] first proposed genetic algorithms. The fundamental underlying mechanism to start the search GAs is initialized with a population of individuals. The individuals are encoded as chromosomes in the search space. GAs use two main operators, namely, crossover and mutation, to direct the population to the global optimum. Crossover allows exchanging information between different solutions (chromosomes), and mutation increases the variety in the population. After the selection and evaluation of the initial population, chromosomes are selected and the crossover and mutation operators are applied. Then, the new population is formed. This process is continued until a termination criterion is met [26, 27].

4.2 Chromosome representation

Our chromosome representation has one component, which is job permutation (JP). We use an array of integers with length equal to *L*, and each integer value equals the index of array of corresponding jobs. Our chromosome depends on the following two components:

Concerning the operation sequence part, we use an array of integers with a length equal to *L*, and each integer value equals the index of array of operation for the corresponding job sequence. For the machine sequence part, we also use the same length equal to *L*. For instance, M1 is selected to process operation O_{21} because the value in the array of alternative machine set is 1. The value in the array is also equal to 1 because operation O_{21} can be processed on two machines, M1 or M3, and the valid values are 1 and 3, as shown in Fig. 1. For the structure of the proposed PJ chromosome.

The procedures of the JP encoding chromosome and decoding used in this study are described in Figs. 3 and 4.

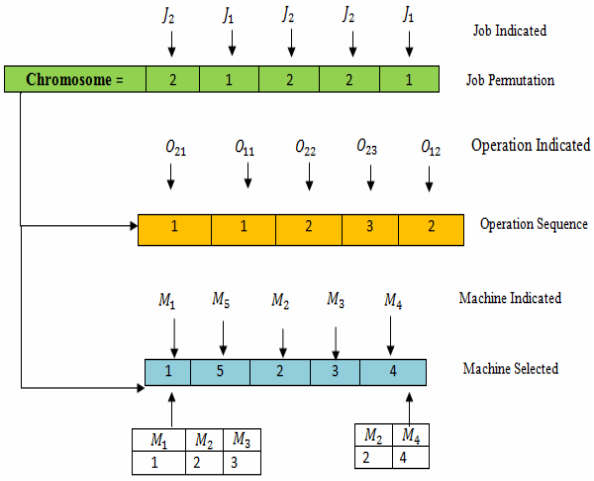

Fig. 1. Structure of proposed PJ chromosome.

Chromosome 1=	2	2	1	2	1
Chromosome 2=	1	2	1	2	2

Fig. 2. Two different permutations for chromosome.

Procedure: Job permutation encoding

Input: Total number of state (machine) **m**, total number of jobs **nj**, total number of operations for each job $\mathbf{no_j}$, processing time matrix **Time**.

Output: Digit permutation chromosome CHROM; assign machine, processing time

Step 1: Generate randomly a vector CHROM of sequencing operations.

Step 2: Set index l =1.

Step 3: Choose randomly from Time matrix a machine such that two operations cannot be processed simultaneously on the same machine.

Set machine (l)

Set processing (l)

Permutation; assignment machine and processing time for each operation is built and read from left to right.

Set l = l + 1.

step 4: Repeat step 3 until l = K

Output CHROM, machine, processing.

End

Fig. 3. Procedure of job permutation encoding.

Procedure: Job permutation decoding

Input: Total number of state (machine) **m**, total number of jobs **nj**, total number of operations for each job $\mathbf{no_j}$, digit permutation **CHROM**; assign machine **Machine**; processing time matrix **Time**.
Output: Best schedule, makespan
Step 1: Deduct from CHROM operation order
Step 2: Choose machine for each operation
Step 3: Calculate makespan
Best schedule, makespan
End

Fig. 4. Procedure of job permutation decoding.

Procedure: Wheel selection job

Input: Total number of state (machine) **m**, total number of jobs **nj**, total number of operations for each job **noj**, total number of individuals in one population **np**, makespan for each individual **Mspan(np)**, population
Output: Selected population Newpopulation
Step 1: Sort Mspan; set index l =1.
Step 2: Prob(l) = Mspan(l)/(sum(Mspan))
l = l + 1.
Step 3: Repeat step 2 until l = np
Step 4: Probp= cumsun(Prob)
Step 5: Select best individuals
Perc = np*30%
Keep 50% of best individuals
rdn random probability
Set index l = 1, repeat
Set index k = 1, repeat
If rdn < Probp(k)
Newpopulation (l) = Population (k)
Break
Endif
k = k+1
Until k = np
l = l+1
Until l = Perc
Output Newpopulation with 30% of best individuals
End

Fig. 5. Procedure of wheel selection job.

4.3 Genetic operators

Selection operator

Choosing individuals for reproduction is the task of selection [28]. The chosen selection approach is adopted. Detailed steps are shown in Fig. 5.

Crossover operator

The goal of the crossover is to obtain better chromosomes to improve the result by exchanging information contained in the current good ones. In this study, we conducted two kinds of crossover operator for the chromosomes.

In accordance with the adopted representation, two crossover operators are used in this study. Uniform crossover and a precedence preserving order-based crossover (POX). The uniform crossover operation is described in Fig. 6.

Procedure: Uniform crossover job

Input: Total number of individuals in one population **np**, Makespan for each individual **Mspan (np)**, Newpopulation
Output: Newpopulation
Step 1: Choose randomly 2 individuals, parent 1 and parent2
Step 2: Generate randomly a binary vector unif(np)
Step 3: Set index l = 1.
Step 4: If unif(l) = 1
Offset1(l) = parent1(l)
Offset2(l) = parent1(1)
with Precedence preserving order
Endif
l = l + 1.
Step 5: Repeat step 3 until l = 50% of population
Replace parent 1,2 with offset 1,2
Newpopulation
End

Fig. 6. Procedure of uniform crossover job.

Procedure: Values mutation job

Input: Total number of individuals in one population **np**, Makespan for each individual **Mspan(np)**, Newpopulation
Output: Newpopulation
Step 1: Set index l = 1
Step 2: Choose randomly an individual from Newpopulation
Step 3: Generate randomly 2 values in an individual
Permute these values if precedence order
Step 4: Repeat step 1 until l 50% of population
Output Newpopulation
End

Fig. 7. Procedure of values mutation job.

Mutation operator

Mutation introduces extra variability into the population to enhance the population diversity. Usually, mutation is applied with small probability. Large probability may destroy the good chromosome. In our study, we conducted one kind of mutation operator, which is the mutation by values for the chromosome PJ (values mutation job is described in Fig. 7).

The framework of the proposed NGA is illustrated in Fig. 8.

Table 2. Results of Brandimart's data.

Problem	n*m	M&G	GENACE	Zhang et al	Chen et al	Pezzella et al	HGTS	NGA
		C_m	C_m	C_m	C_m	C_m	C_m	C_m
Mk01	10*6	40	40	40	40	40	40	37
Mk02	10*6	26	32	26	29	26	26	26
Mk03	15*8	204	N/A	204	204	204	204	204
Mk04	15*8	60	67	60	63	60	60	60
Mk05	15*4	173	176	173	181	173	172	173
Mk06	10*15	58	67	58	60	63	57	67
Mk07	20*5	144	147	144	148	139	139	148
Mk08	20*10	523	523	523	523	523	523	523
Mk09	20*10	307	320	307	308	311	307	307
Mk10	20*15	198	229	198	212	212	198	212

Procedure: New genetic algorithm

Input: Total number of state (machine) **m**, total number of job **nj**, total number of operations for each job no_j, total number of individuals **np**, total number of generations **ng**

Output: Schedule, makespan

Step 1: Generate randomly initial generation, Set index g = 1

Step 2: Calculate makespan

Step 3: Keep best solution

Step 4: Wheel selection operation

Step 5: Uniform crossover operation

Step 6: Values mutation operation

Step 7: Repeat steps from step 2 until g = ng

Output Schedule, makespan

End

Fig. 8. Framework of proposed new GA.

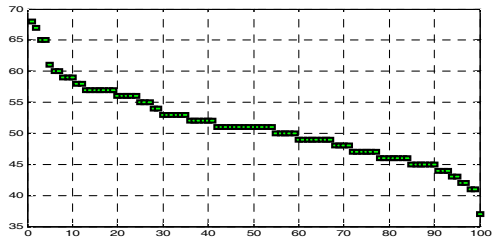

Fig. 9. Decrease of makespan (MK01).

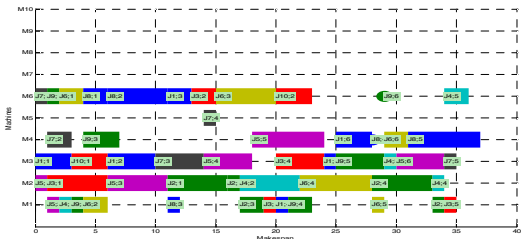

Fig. 10. GANTT chart of MK01.

4.4 Performance verification of NGA

The proposed new GA is tested on Brandimarte's data set (BR data). The data set consists of 10 problems with number of jobs ranging from 10 to 20, number of machines ranging from 4 to 15, and number of operations for each job ranging from 5 to 15.

The proposed new algorithm is compared to the following algorithms [17, 20, 29]:

- M&G: approach proposed by Mastrolilli and Gambardella (2000).
- GENACE: approach proposed by Ho and Tay (2004).
- Zhang et al.: approach proposed by Guohui Zhang and Ling Gao (2011).
- Chen et al.: approach proposed by Chen H and Ihlow J (1999).
- Pessella et al.: approach proposed by Pezzelle and Morganti (2008).
- HGTS: approach proposed by J. J. Palacios and A. González (2014).

The proposed NGA algorithm for FSJ problem was coded in Matlab and run on a 2.3 GHz PC with 4 GO memory with the following parameters: popsize =100, P_c = 0.8, P_m = 0.05, selection percentage = 30%.

Table 2 summarizes the experimental results. It lists problem names, problem dimension (number of jobs × number of machines), best known solution (C_m), solution obtained by our algorithm (NGA), and solution obtained by each of the other algorithms. The computational results show that the proposed genetic algorithm is speed so far of searching optimal solution. Among the 10 test problems, Mk01 could gain better solution among all the approaches. Mk03 and Mk08 could obtain the optimal solution in the first generation by using NGA. Mk02, Mk04, Mk05, and Mk09 (four problems) could have the same good results as the M&G approach. One problem, Mk06, could gain the same results as GENACE and two problems (Mk07, Mk10) could obtain the same results as Chen et al. In Figs. 9 and 10, we draw the decrease of

Table 3. Drugs involved in jobs.

Job	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}
Product	Suipuren 180 ml	Encofluide adult 180 ml	Encofluide enfant 180 ml	Pentussil 180 ml	Eupnex 180 ml	Salbutamol-saidal 150 ml	Ferracur 125 ml	Bromhexine-saidal 60 ml	Allertine 60 ml	Valkine 60 ml

Table 4. Example of job assignment by three shops and different machines.

Product / Job	Shop	Operation	Machines	P_{ij} (ms)
Suipuren 180 ml j1	Weinghing room	O11	BP1	57600
			BP2	0
		O12	BA1	72000
			BA2	600
		O13	B1	43200
			B2	50400
			B3	0
		O14	IM1	7200
			IM2	0
			IM3	9000
	Fabrication	O15	CF1	0
			CF2	230400
			CF3	648000
		O16	GB1	1320
			GB2	60
		O17	NP1	72000
			NP2	0
			NP3	120
	Conditionnement	O18	TE	6000
		O19	BV	80160
		O110	RE-SE	3720
		O111	TA1	5760
		O112	ET	56
		O113	TA2	6420
		O114	EC-PN	4500
		O115	TP	32
		O116	ID	5520
		O117	VN1	0
			VN2	52
		O118	MC1	7200
			MC2	10800

makespan and Gantt chart for the Mk01 test problem with 10 jobs and 6 machines from the BR data.

5. Case study problem

Our research work considers a real-world application: the scheduling problem of a drug company. Saidal Group is one of the leading pharmaceutical manufacturers in Algeria. The company produces a variety of drugs. Each product may be considered as a job. Thus, we consider 10 jobs in this problem as shown in Table 3. The parts are produced in machines. The machine set size is 31.

The processing time is the time required for a machine to be processed in different stages. The processing time taken by each machine is measured many times and the average time is taken in this work. The processing time (in 10^{-3} seconds) of all 10 jobs in different machines are presented in Table 4.

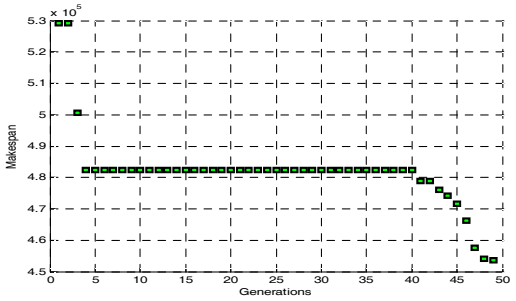

Fig. 11. Decrease of makespan (weighing room).

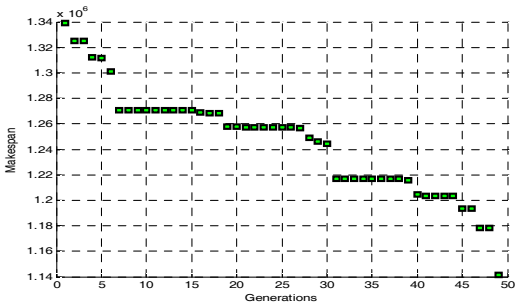

Fig. 12. Decrease of makespan (fabrication shop).

5.1 Results

To obtain meaningful results, we ran our algorithm five times on the same instance. The parameters used in the NGA were chosen experimentally to obtain a satisfactory solution within an acceptable time span. According to the complexity of the problems, the population size of the effective GA ranged from 50 to 150.

Initially, we checked the performance of the approach in Figs. 11-13. Then, we drew the decrease of the average best makespan over five runs for the weighing room with (10 jobs and 10 machines) production shop with (10 jobs and 8 machines), and conditionnement shop with (10 job and 13 ma-

Table 5. Makespan Comparison between SAIDAL Company And NGA.

Shop	n*m	SAIDAL (10^3)(ms)	NGA (ms)	Gap (ms) x10^3
Weighing room	10*10	27000.00	452300.00	265.47700
Fabrication	10*8	108000.00	1134180.00	1068.65820
Conditionnement	10*13	3000.00	699020.00	23.00980
Company	10*31	138000.00	1570140.00	136429860.00

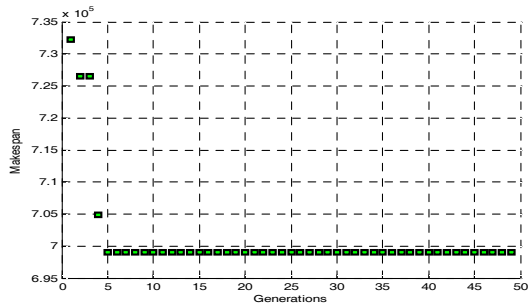

Fig. 13. Decrease of makespan (conditionnement shop).

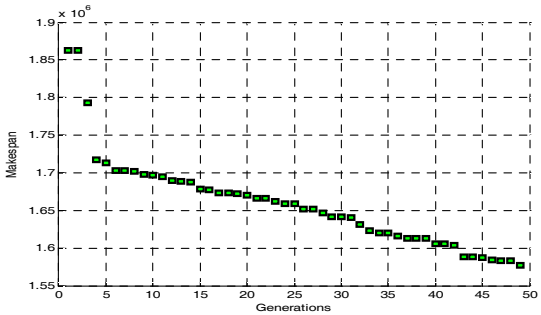

Fig. 14. Decrease of makespan for company.

chines).

In Fig. 14, we show the decrease of the average best makespan over five runs for the entire company's problem with 10 jobs and 31 machines.

Figs. 15-17 show the Gantt chart of optimal solution for weighing room, fabrication shop and conditionnement shop of this company.

Finally, the Gantt chart of the entire company is shown in Fig. 19.

5.2 Additional performance evaluation

We conduct a set of experiments to evaluate the performance of the proposed NGA algorithm. The different factor levels for the design of experiments and the best makespan comparison is presented in Table 5. According to the results, the proposed GA algorithm provides optimal results with minimum computational time. A graphical representation of the gap between our approach and the company's processing time is presented in Fig. 18).

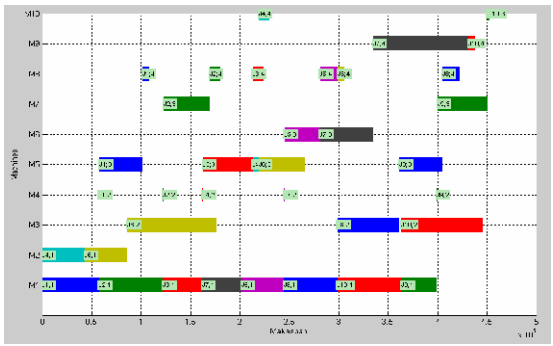

Fig. 15. Gantt chart of weighing room.

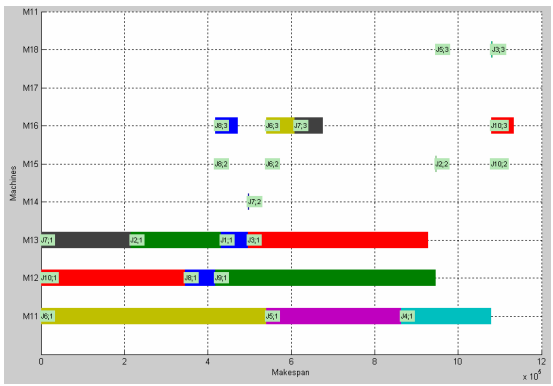

Fig. 16. Gantt chart of fabrication shop.

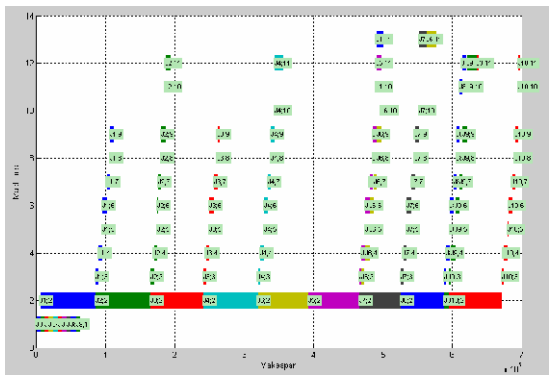

Fig. 17. Gantt chart of conditionnement shop.

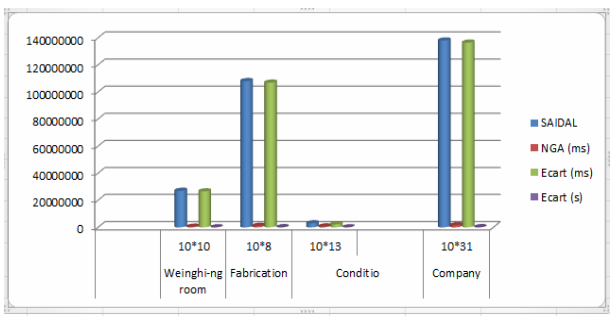

Fig. 18. Comparison of processing time between SIADAL company and NGA.

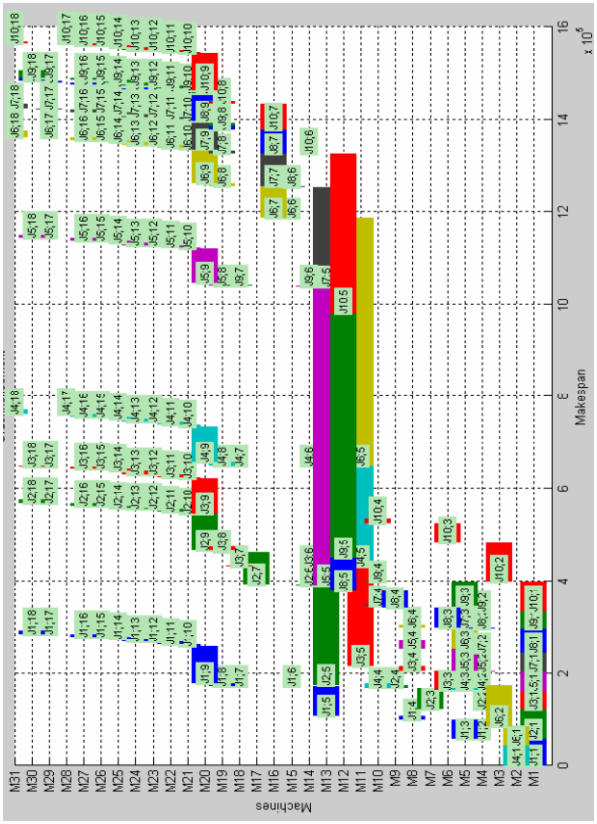

Fig. 19. The Gantt chart of the drug manufacturing company.

6. Conclusion and future study

In this paper, we proposed an NGA to solve FJSP. Thus, we proposed a new chromosome representation scheme and various strategies for crossover and mutation operators. Besides, the proposed algorithm has been tested on instances issued from benchmark literature, and we checked this algorithm with real word application data that belonged to a drug manufacturing company. The computational results show that the proposed new genetic algorithm (NGA) efficiently solves FJSP.

In our future research, we intend to optimize this algorithm and the solution procedure to consider multiple objectives such as due dates, mean flow-time requirements, and other constraints such as changing tools.

Acknowledgment

The study has been generously supported by Batna University. The authors express their sincere appreciation for all support provided.

References

[1] M. R. Garey, D. S. Johmson and R. Sethi, The complexity of flow shop and job shop scheduling, *Mathematics of Operational Research*, 1 (1976) 117-129.

[2] I. Kacem, S. Hammadi and P. Borne, Pareto-optimality approach for flexible job shop scheduling problems, Hybridization of evolutionary algorithms and fuzzy logic, *Matimatic and Computers in Simulation*, 60 (2002) 245-276.

[3] P. Brucker and R. Schile, Job-shop scheduling with multi-purpose machines, *Computing*, 45 (4) (1990) 369-375.

[4] P. Brandimarte, Routing and scheduling in a flexible job shop by tabu search, *Annals of Operations Research*, 41 (1993) 157-183.

[5] J. A. Paulli, Hierarchical approach for the FMS scheduling problem, *European Journal of Operational Research*, 86 (1) (1995) 32-42.

[6] K. Mesghani, S. Hammadi and P. Borne, *On modeling genetic algorithms for flexible job shop scheduling problems* (1998).

[7] H. Chen, J. Ihlow and C. A. Lehmann, Genetic algorithm for flexible Job shop scheduling, *IEEE International Conferenceon Robotics and Automation*, Detroit, 2 (1999) 1120-1128.

[8] I. Kacem, S. Hammadi and P. Borne, Approche by localization and multiobjective evolutionary and optimization for flexible job shop scheduling problems, *IEEE Transations Man and Cybernetrics*, 32 (1) (2002) 1-13.

[9] N. B. Ho and J. C. Tay, GENACE : An efficient cultural algorithm for solving the flexible job shop problem, *Proceeding of IEEE Congress on Evolutionary Computation*, 1 (2004) 1759-1766.

[10] H. P. Zhang and M. Gen, Multistage-based genetic algorithm for flexible job shop scheduling problem, *Journal of Complexity International*, 48 (2005) 409-425.

[11] P. Fattahi, M. S. Mehrabad and F. Joli, Mathematical modeling and heuristic approaches to flexible job shop scheduling problems, *Journal of Inteligent Manufacturing,* 18 (2007) 331-342.

[12] N. B. Ho, J. C Tay, M. Edmund and K. Lai, An effective architecture for learning and evolving flexible job shop schedules, *European journal of Operational Research*, 179 (2007) 316-333.

[13] J. Gao, M. Gen and L. Sun, A hybrid of genetic algorithm and bottleneck shifting for multi objective flexible job shop scheduling problems, *Computers and Industrial Engineering,* 53 (2007) 149-162.

[14] J. C. Tay and N. B. Ho, Evolving dispatching rules using genetic programming for solving multi- objective flexible jobshop problems, *Computers and Industrial Engineering,* 54 (2008) 453-473.

[15] B. S. Girish and N. Jawahar, *Scheduling job shops associated with multiple routings with genetic and ant colony heuristics* (2008).

[16] S. G. Pannanbalam, N. Jawahar and B. S. Girish, Giffler and Thampson Procedure based genetic algorithms for scheduling job shops, *Springer-Verbag Berlin Heidelberg* (2009) 229-259.

[17] F. Pezzela, G. Margenti and G. Ciaschetti, A genetic algorithm for flexible job shop scheduling problem, *Computers and Operations Research,* 35 (10) (2007) 3202-3212.

[18] W. Sun, Y. Pan, X. Lu and Q. Ma, Research on flexible job shop scheduling problem based on a modified genetic algorithm, *Journal of Mechanical Science And Technology,* 24 (10) (2010) 2115-2119.

[19] A. Motaghedi, K. Sabri-Laghare and M. Heydari, solving flexible job shop scheduling problem with multi objectives, *International Journal of Industrial Engineering and Production Research,* 21 (2010)197-209.

[20] G. Zhang, L. Gao and Y. Shi, An effective genetic algorithm for the flexible job shop scheduling problem, *Expert System with Application*, 38 (2011) 3563-3573.

[21] Q. Zhang, H. Manier and A. Manier, A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constants and bounded proceding times, *Computers and operations Research*, 39 (2012) 1713-1723.

[22] J. C. Chen, C. C. Wu and C. W. Chen, Flexible job shop with parallel machines using genetic algorithm and grouping genetic algorithm, *Expert Systems with Application,* 39 (2012) 10016-10021.

[23] N. Kim, H. Kim and J. Lee, Damage detection of truss structures using two stage optimization based on micro genetic algorithm, *Journal Of Mechanical Science And Technology,* 28 (9) (2014) 3687-3695.

[24] R. N. Yadar, V. Yadar and G. H. Singh, Application of non dominated sorting genetic algorithm for multi objective optimization of electrical discharge diamond face grinding process, *Journal of Mechanical Science And Technology*, 28 (6) (2014) 2299-2306.

[25] L. N. Xing, Y. U. Chen and K. W. Yang, Multi population interactive coevolutionnary algorithm for flexible job shop scheduling problems, *Comput. Optim. Appl.* (2009), DOI 10.1007//S 10589-009-9244-7.

[26] F. N. Defersha and M. Chen, A coase-grain parallel genetic algorithm for flexible job shop scheduling with lot streaming, *In IEEE International Conference on Computational Science and Engineering* (2009).

[27] Y. K. Park and J. M. Yang, Optimization of mixed casting processes considering discrete ingot sizes, *Journal Of Mechanical Science and Technology*, 23 (2009) 1899-1910.

[28] S. F. Hwang, Y. Hsu and Y. Chen, A genetic algorithm for the optimization of fiber angles in composite laminates, *Journal of Mechanical Science and Technology*, 28 (8) (2014) 3163-3169.

[29] J. J. Palacios, A. González and C. R. González, Genetic tabu search for the fuzzy flexible job shop problem, *Computers & Operations Research* (2014) 5474-89.

Imen Driss received her Master's degree from the Department of Industrial Engineering, University of Batna, Algeria, in 2010. She is currently a doctoral student at the Department of Industrial Engineering, University of Batna, Algeria. Her research interests include producti degree on scheduling, production engineering, ma nufacturing, and others.
E-mail : idrissamina@hotmail.fr.

Kinza Nadia Mouss was born in Batna, Algeria, in 1960. She received the B.Sc. degree in Electrical Engineering in 1983 from the National Polytechnic School of Algiers, Algeria; the M.Sc. degree in Electrical and Computer Engineering in 1984 from the ENSERB, France; and the Ph.D. also in Electrical and Computer Engineering in 1986 from Bordeaux University, France. After graduation, she joined the University of Batna, Algeria, where she is a Professor of Electrical and Computer Engineering. Dr. Mouss is the head of the Computer Integrated Manufacturing and Supply Chain Management Group. Her current research interests include industrial supply chain management, production systems, and computer integrated manufacturing.
E-mail : kinzmouss@yahoo.fr.