International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 11 – No. 3, August 2016 – www.ijais.org

Genetic Algorithm Tuning Applied to the Open Shop Scheduling Problem

Chaouqi Mohsine
OSIL Team LRI, ENSEM,
KM7, BP 8118 Route El Jadida
Casablanca, Morocco

Benhra Jamal
OSIL Team LRI, ENSEM,
KM7, BP 8118 Route El Jadida
Casablanca, Morocco

My Ali El Oualidi
OSIL Team LRI, ENSEM,
KM7, BP 8118 Route El Jadida
Casablanca, Morocco

ABSTRACT

The present paper deals with the open-shop scheduling problem using a manual tuning of a genetic algorithm's parameters. A comparison has been performed between Taillard's Benchmarks for 60 instances, 2 dispatching rules and 198 variants from the GA algorithm obtained by changing the population size, the generation's number, the crossover probability, and the mutation probability. Interesting results were obtained leading to some conclusions for the best choice of the parameters.

General Terms

Scheduling, Genetic Algorithms

Keywords

Scheduling, Open shop, Genetic Algorithms, Tuning, Benchmarks

1. INTRODUCTION

Scheduling belongs to the most important features of productivity improvement. It is a decision-making's form that plays a crucial role in industries. [1]

In shop scheduling problem, a set of jobs has to be processed on a set of machines by defining the time intervals in which the operations have to be processed.

There are three basic types of shops: a flow-shop (each job is characterized by the same technological route), a job-shop (each job has a specific route) and an open-shop (no technological route is imposed on the jobs). [2]

However there exist different objectives in scheduling optimization. The most known objective is the makespan or Cmax minimization which is the time's span required to process all the jobs, i.e. the time from the beginning of the first operation until the end of the last operation. The second one is to minimize the flowtime, denoted by $\sum$Cj, which is the sum of completion times of all the jobs. Other objectives are the tardiness's minimization, the number of tardy jobs, etc.

In the last decades, many researchers, engineers and mathematicians have been interested by solving the shop scheduling problem. Complex problems were first developed by mathematical models. Because of the time consuming of these models, most of researchers developed heuristic methods [3]. However those heuristics are limited to some specific problems and can't be generalized to all shop scheduling problems. Actually the best way to resolve this kind of complex problem is the use of metaheuristics. These stochastic methods are inspired by analogies from nature like evolutionary algorithms, tabu search, simulated annealing, ant colony optimization, particle swarms, etc. [4]

The genetic algorithm proposed in [5] is used in this article to resolve the problem of minimizing the makespan in an open-shop O||Cmax [6] by tuning its parameters manually. And compared the results obtained with Taillard's benchmarks in 60 instances where the jobs' number is equal to the machines' number.

This paper contains four sections organized as follow: Some works related to this subject have been reviewed in the first section, in section two the method used to resolve this problem has been presented, then the gotten results are given with some discussions in section three. Finally at the end of this article some conclusions and some perspectives for future works are given.

2. LITERATURE REVIEW

There are several works that treated the resolution of the open-shop scheduling by genetic algorithms. Among them one can find those of Liaw [7] who proposed a hybrid genetic algorithm (HGA) to resolve the open shop scheduling problem. The hybrid algorithm incorporates a local improvement procedure based on tabu search (TS) into a basic GA. The algorithm developed has been tested on randomly generated instances and on the benchmarks sets by Taillard [8] and Brucker et al. [9]. It has been found that this HGA outperformed other existing algorithms from the literature, and some benchmarks' instances have been solved to optimality for the first time.

Fang et al. [10] suggested an algorithm which combines a GA with heuristic rules for the schedule construction. The algorithm has been tested on the benchmark instances from [8] using ten runs for each instance. By their tests, they discovered one new best known solution for a problem with 7 machines and 7 jobs and a problem with 10 machines and 10 jobs instance.

Khuri and Miryala [11] presented three GA's variants for the problem O||Cmax : a permutation GA (PGA), a hybrid GA (HGA) and a selfish gene (FGA) algorithm. The three algorithms have been tested on the benchmark instances from [8]. Prins [12] gave a GA for the problem O|| Cmax. He made some tests for the benchmarks instances from [8]. The GA, he proposed, yielded competitive results.

Senthilkumar and Shahabudeen [13] presented a GA for the problem O||Cmax. The authors used a chromosome with length equal to the operations' number which is split into sub-chromosomes each of them representing the machine sequence of a job. In their article they compared the GA with a particular heuristic on small instances. Andresen et al. [5] presented a GA for the problem O||$\sum$Ci. They used the rank matrix to encode an individual on their algorithm.

There is a detailed survey for shop scheduling problems using GA on [2].

Another metaheuristic usually used in scheduling problem is the simulated annealing algorithm. This algorithm has shown good results in different optimization problems e.g. Chaouqi et al [3] used the SA in hybridization within the intuitive heuristic to perform a joint scheduling of production and maintenance in the job shop problem. On the same topic the authors used the Johnson's algorithm combined with a genetic algorithm and the intuitive heuristic to optimize three objectives of the flow shop problem. [14]

Recently Bai et al [15] used a heuristic called general dense scheduling to solve the static and dynamic versions of the flexible open shop scheduling problem. The heuristic proposed brings forth some interesting results.

Naderi et al [16] studied the scheduling open shop problem with no intermediate buffer, called no-wait open shops under makespan minimization. They developed three mathematical models and proposed metaheuristics based on genetic and variable neighborhood search algorithms. The results they got show that the models and metaheuristics are effective to deal with the no-wait open shop problems.

3. GENETIC ALGORITHM AND PARAMETERS TUNING

Genetic algorithms belong to the evolutionary algorithms' class. There exist four kind of evolutionary algorithms which are based on the natural evolution's principles: genetic algorithms (GA), genetic programming, evolution strategies and evolutionary programming. These algorithms have been applied to many optimization problems or learning, and perform good results.[4]

Genetic algorithms (GA) are metaheuristics which generate from a population of starting solutions new solutions by means of random changes. Indeed, they imitate the biological concepts of two individuals' mutation and recombination. They usually start from a population of randomly generated individuals, and follow an iterative process. In each iteration a new population also called a generation is created. In general, a population contains individuals where each individual is usually identified with a chromosome which is further split into a number of genes. These new individuals are included into the next generation if, according to a fitness function, they are not less fit than their corresponding predecessors. This process terminate when a maximum number of generations has been produced, or a satisfactory fitness level has been reached.

A basic genetic algorithm can be presented by the following organigram (figure 1):

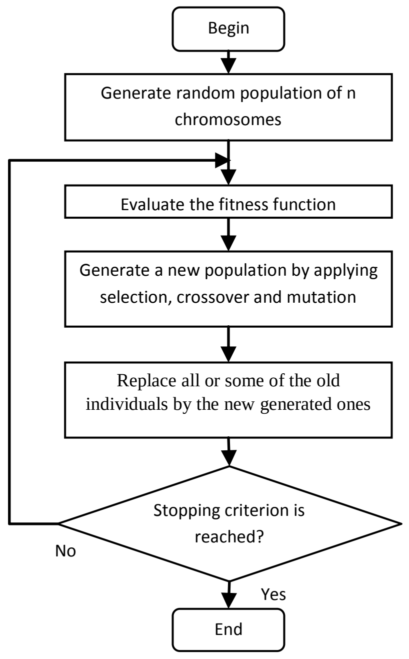
Begin
Generate random population of n chromosomes
Evaluate the fitness function
Generate a new population by applying selection, crossover and mutation
Replace all or some of the old individuals by the new generated ones
Stopping criterion is reached?
No
Yes
End

Figure 1: Genetic algorithm

Before starting a population's generation, a representation of its individuals must be done. There exist different representations also known as chromosome's encoding. In this study the representation proposed by Andersen et al. [5] is reused. Thus the individuals are encoded by a Rank matrix R= (r_{ij}) that describes a sequence graph G(MO, JO) which is a feasible combination of machine orders and job orders. The rank r_{ij} is the maximal number of operations on a longest path ending in operation (i, j) [2]

The mutation used in this algorithm is performed as follow: Let r_{ij}=l from R the rank of a random operation (i,j). In the first place the mutation operator changes this rank by a new value k* where $k^* \epsilon \{1, 2, \dots, k, k+1\} \backslash \{l\}$ and k is the maximal rank in row i and column j of R. Then it puts the operations on a linear order to construct a new correct rank matrix. In the case of ties, a lexicographical order is applied with one exception: The mutated operation gets the smallest number in the linear order among all operations with the same rank. [5][2]

The figure 2 illustrates an example of the mutation operator.

Parent	Proto-child	Linear order of operations	Offspring
$\begin{pmatrix} 2 & 1 & 4 \\ 1 & 2 & 3 \\ 3 & 4 & 1 \end{pmatrix}$	$\begin{pmatrix} 2 & 1 & 4 \\ 1 & 2 & 1 \\ 3 & 4 & 1 \end{pmatrix}$	$\begin{pmatrix} 5 & 2 & 8 \\ 3 & 6 & 1 \\ 7 & 9 & 4 \end{pmatrix}$	$\begin{pmatrix} 3 & 1 & 4 \\ 2 & 3 & 1 \\ 4 & 5 & 2 \end{pmatrix}$

Figure 2: Mutation operator

In the crossover step, two individuals from the current population are chosen. Then the ranks of random operations from those individuals are exchanged. Usually the two gotten proto-children are infeasible. Thus one can use the similar method as the mutation described above: The relative order of the remaining operations from the parent is maintained, and both parts are now combined to a feasible rank matrix. To this end, a linear order is put on the operations in such a way that a smaller number in the linear order indicates that the rank of this operation is not greater than the rank of an operation with a larger number. In the case of ties, a lexicographical order is applied with the following exception: The operations for which the exchange of the ranks takes place get the smallest possible numbers among all operations with the same rank[11][12]. The following example (figure 3) illustrates the crossover operator.

The selection: In this paper the adopted selection scheme is a combination of two strategies; the 2/4 selection developed by Shi [17] and an elitist strategy. [7]

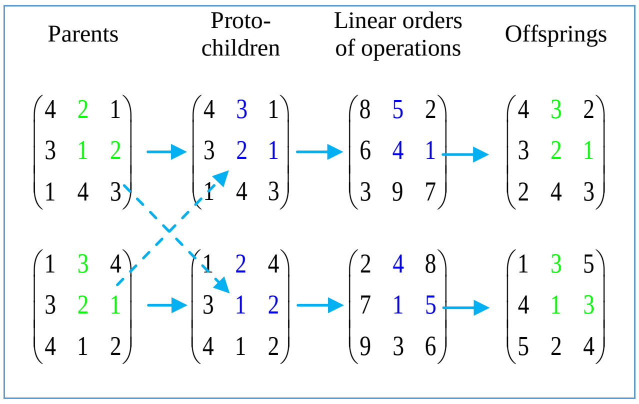

Figure 3: Crossover operator

The selection scheme is described by the following steps:

- Step 1: From the current population select the best solution and insert it directly into the next population.
- Step 2: With an equal probability choose randomly two different solutions from the current population.
- Step 3: Two new offsprings will be generated from those two solutions by applying crossover and mutation operators to them.
- Step 4: Replace each offspring with a local optimum solution by applying local improvement.
- Step 5: Select the better two solutions with different makespans if possible from the four solutions generated in the previous steps (the two old ones and the two new ones).
- Step 6: Insert the two solutions selected in Step 5 into the next generation.
- Step 7: Repeat Steps 2-6 until the next population is full. [7]

4. RESULTS AND DISCUSSION

The computational results for the genetic algorithm is presented in this section. One can note that the parameters used in this study are: num_gene{100, 200}, pop_sizeϵ{50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000}, c_probϵ{0.1, 0.2, 0.5, 0.8} and m_probϵ{0, 0.1, 0.2}. The cases where (num_gen, pop_size, m_prob, c_prob) ϵ{(-, -, 0.2, 0.2), (-, -, 0.1, 0.1), (-, -, 0, 0.1)} are avoided here. Thus the total variants' number is 198. The time limit is set to 10s as a stopping criterion. The generation of the first population was randomly.

4.1 Case n° 1: Cmax = Optimum

Here is described the frequencies of variants which solved a certain number of problems out of the sixty instances given by Taillard, i.e. the final makespan calculated for these variants has reached the optimum of the corresponding problem.

Table. 1: number of variants by the number of solutions where Cmax=Opt

Variants' number	**1**	**3**	**3**	**8**	**8**	**12**
Solutions' number	14	13	12	11	10	9
Variants' number	23	28	24	27	21	23
Solutions' number	8	7	6	5	4	3

The table n°1 presents the results of this first case.

As shown in figure 4, the graph corresponds to a positive skew with a small tail on the right and three pics. This effect is undesirable, i.e. the tuning is not as good as expected. The best configuration which gives the most number of optimums (i.e. 14 out of 60) is GA1 where (pop_size=800, num_gen=200, m_prob=0, c_prob=0.2). Also 13 optimal solutions out of 60 are found for the three following configurations: (1000, 100, 0, 0.2), (700, 200, 0, 0.2) and (1000, 100, 0.1, 0.5).

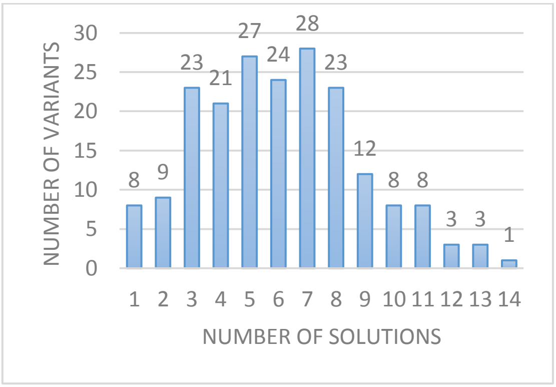

Figure 4: Variants' number by the solutions' number where Cmax=Opt

To avoid any confusion two other cases of a deep analyze of these data are given in the following sections. In one case the ratio Optimum/Cmax is greater than or equal to 0.99 and in the other one it is greater than or equal to 0.98.

4.2 Case n° 2: Optimum/Cmax>=0.99

According to the figure 5 one can notice a huge modification of the distribution. The highest number of solutions where Optimum/Cmax>=0.99 is 38 out of 60. This number was reached for two variants followed by 37 solutions out of 60 for 2 configurations. These four variants are: (900, 100, 0.2, 0.5) denoted by GA2, (700, 200, 0, 0.2), (1000, 200, 0, 0.2) and (700, 200, 0.2, 0.1).

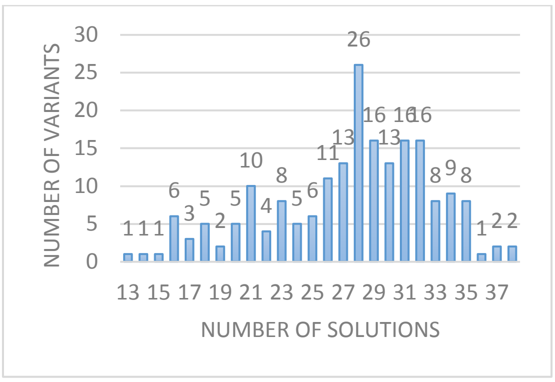

Figure 5: Number of variants by the number of solutions where Optimum/Cmax>=0.99

4.3 Case n° 2: Optimum/Cmax>=0.98

As showen in the figure 6, the graph obtained for this case contains multiple pics. However one can observe that the results here are more interesting than before.

Most of variants give solutions with Optimum/Cmax>=0.98. The maximal number of solutions with Opt/Cmax>=0.98 is 56 out of 60 reached by one variant: (1000, 200, 0.2, 0.5) denoted by GA3, followed by 54 out of 60 obtained for (900, 100, 0.2, 0.5), (1000, 100, 0.2, 0.1) and (700, 200, 0.2, 0.1).

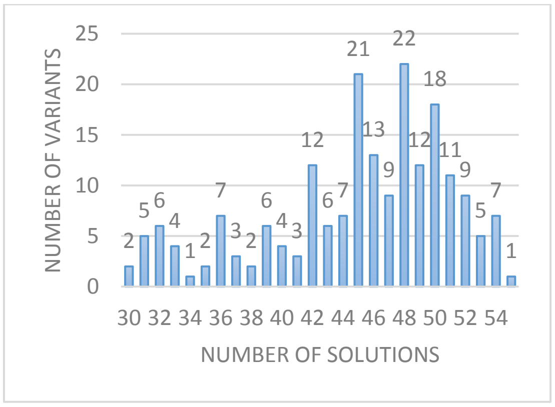

Figure 6: Number of variants by the number of solutions where Optimum/Cmax>=0.98

All the results of GA1, GA2 and GA3 for the benchmark problem from Taillard, are regrouped on table n° 2 with those given by SPT and LPT rules. The abbreviation LB means the lower bound and Opt stands for the optimum found in the literature.

Table 2: Results for the benchmark problems from Taillard

	i	*LB*	*Opt*	*SPT*	*LPT*	*GA1*	*GA2*	*GA3*
(4x4)	1	186	193	228	219	193	196	195
	2	229	236	276	256	247	247	239
	3	262	271	304	299	272	272	272
	4	245	250	307	260	254	253	254
	5	287	295	348	317	298	301	299
	6	185	189	225	239	189	189	189
	7	197	201	247	218	201	203	203
	8	212	217	233	248	217	217	217
	9	258	261	282	282	267	261	261
	10	213	217	235	225	221	221	221
(5x5)	11	295	300	333	344	303	303	302
	12	255	262	297	297	269	265	266
	13	321	323	404	364	335	335	335
	14	306	310	317	369	316	316	316
	15	321	326	392	358	330	330	330
	16	307	312	353	360	320	320	318
	17	298	303	340	357	308	308	308
	18	292	300	369	343	304	304	304
	19	349	353	372	418	362	362	362
	20	321	326	375	371	333	329	328
(7x7)	21	435	435	507	465	445	440	436
	22	443	443	485	526	449	454	454
	23	468	468	538	527	484	474	473
	24	463	463	492	527	466	466	466
	25	416	416	461	443	416	416	420
	26	451	451	518	494	455	455	455
	27	422	422	464	451	432	432	429
	28	424	424	482	494	427	427	427
	29	458	458	520	488	458	460	459
	30	398	398	435	445	402	403	403
(10x10)	31	637	637	685	661	646	642	647
	32	588	588	658	643	592	591	594
	33	598	598	679	672	600	605	606
	34	577	577	632	591	582	580	580
	35	640	640	693	701	640	644	650
	36	538	538	559	556	541	538	540
	37	616	616	672	637	626	620	622
	38	595	595	651	686	602	601	604
	39	595	595	655	621	602	597	595
	40	596	596	633	636	606	604	605
(15x15)	41	937	937	987	972	939	939	943
	42	918	918	937	972	922	922	920
	43	871	871	891	878	873	874	873
	44	934	934	975	965	934	935	934
	45	946	946	959	999	952	950	946
	46	933	933	981	952	935	935	936
	47	891	891	919	955	896	896	897

	48	893	893	928	929	893	894	893
	49	899	899	990	927	911	909	918
	50	902	902	922	943	905	905	905
(20x20)	51	1155	1155	1194	1200	1158	1162	1161
	52	1241	1241	1296	1296	1249	1248	1249
	53	1257	1257	1304	1258	1257	1258	1258
	54	1248	1248	1312	1274	1248	1248	1248
	55	1256	1256	1277	1262	1256	1257	1256
	56	1204	1204	1219	1215	1206	1208	1206
	57	1294	1294	1407	1317	1302	1301	1299
	58	1169	1169	1205	1216	1183	1189	1184
	59	1289	1289	1306	1293	1289	1289	1289
	60	1241	1241	1272	1265	1241	1241	1241

These data are summarized in table n°3. One can observe that the problem's size has an impact on the number of solutions. For example optimum solutions are found for 5 out of 10 where n=m=20 and none where n=m=5 for GA1.

Table n°3: Summary of the results in table2

(nxm)_i	SPT	LPT	GA1	GA2	GA3
(4x4)	-	-	4	3	3
(5x5)	-	-	-	-	-
(7x7)	-	-	2	1	-
(10x10)	-	-	1	1	1
(15x15)	-	-	2	-	3
(20x20)	-	-	5	4	4

5. CONCLUSION

In this paper a computation has been performed between Taillard's Benchmarks for 60 instances in the open shop problem, the SPT and the LPT dispatching rules and 198 variants from the GA algorithm obtained by changing the population size, the generations' number, the crossover probability, and the mutation probability.

The results are interesting in the most cases. However they still far away from those obtained by Liaw [7] where he finds an optimum solution for 58 out of 60 problems using HGA. This difference is maybe due to the time limit set as a stopping criterion or it is due to the chosen GA implementation which needs some improvements.

As a perspective on future studies, one can use some tools on tuning the GA's parameters like neural networks or Bayesian networks and a hybridization in between different algorithms then try new tests to get an optimum makespan for all instances.

6. REFERENCES

[1] M. L. Pinedo, *Scheduling*, vol. 1. Boston, MA: Springer US, 2012.

[2] F. Werner, "Genetic algorithms for shop scheduling problems: A survey," *Preprint*, 2011.

[3] M. Chaouqi and J. Benhra, "Recuit simulé hybride pour un ordonnancement conjoint de la production et de la maintenance dans un atelier job-shop," *Int. Work. Theory Appl. Logist. Transp. TALT15*, 2015.

[4] P. S. J. Dréo, A. Pétrowski, E. Taillard, *Metaheuristics for Hard Optimization*, vol. 53, no. 9. Berlin/Heidelberg: Springer-Verlag, 2006.

[5] M. Andresen, H. Bräsel, M. Mörig, J. Tusch, F. Werner, and P. Willenius, "Simulated annealing and genetic algorithms for minimizing mean flow time in an open shop," *Math. Comput. Model.*, vol. 48, no. 7–8, pp. 1279–1293, Oct. 2008.

[6] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling : a survey," 1979.

[7] C.-F. Liaw, "A hybrid genetic algorithm for the open shop scheduling problem," *Eur. J. Oper. Res.*, vol. 124, no. 1, pp. 28–42, Jul. 2000.

[8] Taillard E., "Benchmarks for basic scheduling problems," *Eur. J. Oper. Res.*, vol. 64, pp. 1–17, 1993.

[9] P. Brucker, J. Hurink, B. Jurisch, and B. Wöstmann, "A branch & bound algorithm for the open-shop problem," *Discret. Appl. Math.*, vol. 76, no. 1–3, pp. 43–59, Jun. 1997.

[10] H. Fang and P. Ross, "A Promising Hybrid GA/Heuristic Approach for Open-Shop Scheduling Problems In Proceedings of the 11th European Conference on Arti cial Intelligence, John Wiley and Sons, 1994, pages 590{594.," no. 699, 1994.

[11] S. Khuri and S. R. Miryala, "Genetic Algorithms for Solving Open Shop Scheduling Problems," in *Proceedings of the 9th Portuguese Conference on Artificial Intelligence: Progress in Artificial Intelligence (EPIA '99)*, 1999, pp. 357–368.

[12] C. Prins, "Competitive genetic algorithms for the open-shop problem," *Math. Methods Oper. Res.*, vol. 52, no. 3, pp. 389–411, 2000.

[13] P. Senthilkumar and P. Shahabudeen, "GA based heuristic for the open job shop scheduling problem," *Int. J. Adv. Manuf. Technol.*, vol. 30, no. 3–4, pp. 297–301, 2006.

[14] M. Chaouqi, J. Benhra, and A. Zakari, "Agile Approach for Joint Scheduling of Production and Maintenance in Flow Shop," *Int. J. Comput. Appl.*, vol. 59, no. 11, pp. 29–36, Dec. 2012.

[15] D. Bai, Z.-H. Zhang, and Q. Zhang, "Flexible open shop scheduling problem to minimize makespan," *Comput. Oper. Res.*, vol. 67, pp. 207–215, Mar. 2016.

[16] B. Naderi and M. Zandieh, "Modeling and scheduling no-wait open shop problems," *Int. J. Prod. Econ.*, vol. 158, pp. 256–266, Dec. 2014.

[17] G. SHI, "A genetic algorithm applied to a classic job-shop scheduling problem," *Int. J. Syst. Sci.*, vol. 28, no. 1, pp. 25–32, Apr. 2007.