Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 52-13 (2019) 2519–2524

An Efficient Two-Stage Genetic Algorithm for Flexible Job-Shop Scheduling ⋆

Danial Rooyani, Fantahun M. Defersha *

* *School of Engineering, University of Guelph, Guelph, Ontario, Canada (e-mail: fdefersh@uoguelph.ca).*

Abstract: Flexible job shop Scheduling Problem (FJSP) is considered as an expansion of classical Job-shop Scheduling Problem (JSP) where operations have a set of eligible machines, unlike only a single machine at JSP. FJSP is classified as non-polynomial-hard (NP-hard) problem. Researchers developed different techniques including Genetic Algorithm (GA) that is widely used for solving FJSP. Regular GAs for FJSP determine both operation sequencing and machine assignment through genetic search. In this paper, we developed a highly efficient Two-Stage Genetic Algorithm (2SGA) that in the first stage, GA coding only determines the order of operations for assignment. But machines are assigned through an evaluation process that starts from the first operation in the chromosome and chooses machines with the shortest completion time considering current machine load and process time. At the end of the first stage, we have a high-quality solution population that will be fed to the second stage. The second stage follows the regular GA approach for FJSP and searches the entire solution space to explorer solutions that might have been excluded at the first stage because of its greedy approach. The efficiency of proposed 2SGA has been successfully tested using published benchmark problems and also generated examples of different sizes. The quality of the 2SGA solutions greatly exceeds regular GA, especially for larger size problems.

Keywords: Flexible Job Shop Scheduling Problem (FJSP); Genetic Algorithm (GA); Two Stage Genetic Algorithm (2SGA); Scheduling.

1. INTRODUCTION

A classic Job shop Scheduling Problem (JSP) consists of scheduling a given number of jobs on a given number of dissimilar machines. Each job needs to be processed through a predefined sequence of operations (routing), while each operation can only be processed on a single designated machine. So JSP solution only consist of sequencing and starting time of operations on each machine (sequencing problem) (Defersha and Chen (2010)). However, in Flexible Job shop Scheduling Problem (FJSP), operations have a set of eligible machines. This flexibility means, FJSP solution must include assignment of operations to one of their alternative machines (routing problem) and simultaneously prioritizing processing of operations on every machine (sequencing problem). Many JSPs are non-polynomial-hard (NP-hard) (Wang et al. (2008)) that means exact algorithms cannot guarantee the optimum solution in finite time (Kachitvichyanukul and Sitthitham (2011)). The flexibility of FJSP improves scheduling efficiency (i.e. shorter makespan), while it makes solving FJSPs more complex than JSPs. (Defersha and Movahed (2018)).

From 1990 that Brucker and Schlie (1990) for the first time proposed an algorithm to solve a FJSP with only two jobs, many different approaches have been developed by researchers to address FJSPs. Techniques like mathematical programming, dispatching rules, simulation-based methods, artificial intelligence-based methods and metaheuristics (Kachitvichyanukul and Sitthitham (2011)). Among those approaches, metaheuristics have been proven to be a very promising method for solving FJSP as a NP-Hard problem. Chaudhry and Khan (2016) performed a survey on different methods that have developed to solve FJSP. They reviewed 191 published journal papers between 1990 to 2014 and reported Genetic Algorithm (GA)as the most popular technique that has been used in 34% of researches either in form of pure GA or hybrid. One of the advantages of GA over other approaches is utilizing a population of solutions to explore solution space more diversely (Al-Hinai and ElMekkawy (2011)).

Amjad et al. (2018) reviewed a total of 190 research articles published between 2001 to 2017 which solved FJSP with GA and concluded GA become extremely popular for solving FJSP specially in the last seven years of the research. GA belongs to a larger class of evolutionary algorithms that initially was proposed by Holland in 1975 (H.Zhang and M.Gen (2005)). It is based on Darwin's evolution theory and Mendel's genetic theory and combines Darwin's "natural selection, survival of the fittest" principle with a structured and randomized offspring creation. It starts from an initial population and applies a series of genetic operators (selection, crossover and mutation) to populate a new generation of chromosomes (solution) which are more fit (i.e. better makespan in FJSP) than their parents. Chromosome encoding (the way each possible solution is

⋆ This research is funded by the Natural Science and Engineering Research Counsel (NSERC) of Canada

10.1016/j.ifacol.2019.11.585

represented), fitness function (suitability of each solution), reproduction operators (selection, crossover and mutation) and GA parameters (like population size, stop condition, mutation and crossover probability) are essential parts of any standard GA.

2. LITERATURE REVIEW

For the first time Brucker and Schlie (1990) proposed a polynomial algorithm to solve a FJSP with two jobs. Since then a large number of techniques ranging from mathematical to various metaheuristics approaches such as Evolutionary Algorithms (EAs), Tabu Search (TS), Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) have been developed to address FJSP. Among all these techniques, GA is the first common and TS is the second-most popular technique to solve FJSP (Chaudhry and Khan (2016)). Researchers also have combined components of these metaheuristics algorithmic and proposed more powerful hybrid algorithms.

In this paper we proposed an efficient two-stage GA. Based on qiu Shi et al. (2018) hybrid and multistage structures are the best strategies to improve GA performance in solving FJSP. They also pointed out unlike hybridization strategy that is the most popular improvement GA strategy, there are only few papers with multistage structures in the FJSP literature despite its effectiveness. H.Zhang and M.Gen (2005) was one of the early researchers who proposed a multistage GA to solve multi-objective FJSP that in fact is a multistage operation of K stages (the total number of operations for all jobs) and M states (total number of machines) that simplified GA encoding and time calculation. Wang et al. (2008) proposed a two-stage GA that finds the fittest GA parameters (number of population, probability of crossover and mutation) using optimal computing budget allocation method and then applies optimum GA parameters to find FJSP optimal solution. They demonstrated efficiency of their method by solving benchmark FJSP instances. Gen et al. (2009) developed a multi-staged GA with two vectors to represent each solution candidate. One vector for initialization and mutation and the other one for crossover operation. In order to strengthen the local search ability, they utilized bottleneck shifting which interchanges sequences and assignments of operations on the critical path. Another two-stage GA is proposed by Kachitvichyanukul and Sitthitham (2011) to solve multi-objective FJSPs. They addressed a FJSP with three objectives of minimizing makespan, earliness and tardiness. At the first stage, three parallel GAs find the best solution for each objective function. Then all three populations are combined to form initial population of the second stage. Second stage uses a weighted and aggregated fitness function to find the final solution. The algorithm performance has been successfully tested using published benchmark problems for both single objective and multi-objective cases. Another two-stage GA is proposed by Al-Hinai and ElMekkawy (2011) to find a robust and stable FJSP solution with random machine breakdowns. At the first stage, GA solves the FJSP based on the main objective function (minimizing makespan) considering all data are deterministic with no expected disruptions (machine breakdown). At the second stage (after a given number of generations) objective function switches to bi-objective of robustness and stability measure with the random machine breakdown. Solution representation and GA parameters will remain same at both stages.

2,1	3,1	1,1	2,2	2,3	1,2	3,2

Stage 1 (j,o)

2,1,3	3,1,1	1,1,2	2,2,1	2,3,3	1,2,2	3,2,1

Stage 2 (j,o,m)

Fig. 1. Solution representation example for stage 1 and 2

3. FJSP AND TWO-STAGE GENETIC ALGORITHM

A regular FJSP includes M different machines (m = 1, 2, ..., M) and J independent jobs (j = 1, 2, ..., J) that job j has o_j number of operations. Each operation has a set of eligible machines with different process time (process time of operation o of job j on machine m will be T_{jom}).

In this paper we proposed a new and efficient two-stage GA to solve FJSP. At the first stage, GA only determines the order in which operations will be assigned. But machine assignment is determined through a greedy approach that for each operation chooses a machine with the shortest completion time based on machine current load and process time. The first stage provides a high quality initial population for the second stage. Second stage follows a regular GA for FJSP. The overall structure of proposed 2SGA is described below:

Elements of Two-Stage GA: (1) Encoding: Stage one uses gene (j, o) with no machine assignment but stage two utilizes (j, o, m) gene structure; (2) Initial Population: At the first stage, initial population is formed randomly but initial population of stage two is the final population of stage one; (3) Fitness evaluation: at both stages is makespan of each chromosome; (4) Selection: at both stages is k-way tournament; (5) Reproduction: at stage one sequencing crossover and operation swap mutation, stage two all operators of stage 1 plus assignment mutation and assignment crossover; (6) Stop criteria: Stage one after 3000 iterations switches to stage two that runs for another 7000 iterations (total of 10,000 iterations)

We will describe each element of algorithm in more details in section 3.1 to 3.5

3.1 Solution Representative

As illustrated in figure 1, solution representative at stage one, (j,o), only indicates order of operations that in which they will be assigned. But gene structure at stage two, (j,o,m), following solution encoding presented by Pezzella et al. (2008) indicates assignment of operation o of job j to an alternative machine m and operation sequencing is determined by order of genes in the chromosome. At both stages number of genes is equal to total number of operations of all jobs. Figure 1 shows solution representatives for 3 jobs with total of 7 operations on 3 machines at both stages.

3.2 Initial Population

GA needs an initial population to start. To form initial population of stage one, operations are randomly posi-

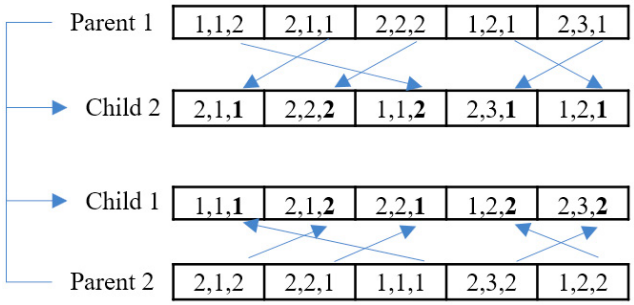

Fig. 2. Assignment crossover operator (only at stage 2)

tioned in the chromosome without machine assignment. Then feasibility of this order is checked to make sure precedence constraint among operations of the same job is met and switch their positions otherwise, i.e., $(j,\ o)$ cannot be after $(j,\ o+1)$. This will provide the order of operation in which they can be assigned. So starting from the first operation (the first gene), every eligible machine is checked and the one with the minimum completion time will be selected. To find a machine with minimum completion time, process time (T_{jom}) and current machine load (the operations that have been already assigned on the machine) are being considered. As it was mentioned, initial population of stage two is the final population of stage one.

3.3 Fitness Evaluation

Makespan or completion time is the time when the last operation of every job is completed. Minimizing the completion time of the last work order is one of the primary measures of scheduling problems and fitness evaluation function for many FJSP problems including our proposed 2SGA.

3.4 Genetic Operators

GA uses selection, mutation and crossover operators to produce offsprings from initial population and improves them until reaches to an acceptable solution.

In proposed 2SGA, we utilized a k-way tournament as selection operator. At k-way tournament, k-portion of population is randomly selected and chromosome with the best fitness is copied into reproduction pool. This process will be repeated until number of chromosomes in the reproduction pool is equal to population size.

After forming reproduction pool, GA uses crossover and mutation operators for producing offspring. Crossover operators start with two parents and generate two children while mutation operators has one parent and produces only one offspring. In this research we developed two types of crossover operators, assignment and sequencing crossovers. At our assignment crossover operator, illustrated in Figure 2, each child inherits its sequencing from one parent and machine assignment from the other one. Sequencing crossover operator, illustrated in Figure 3, selects a crossover point and switches genes before that point, then completes the chromosome in the same order of original sequencing.

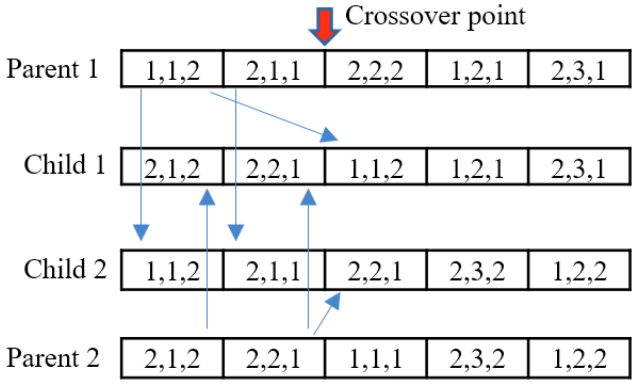

Fig. 3. Sequencing crossover operator

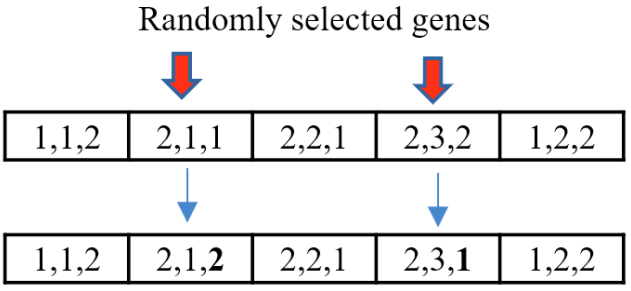

Fig. 4. Assignment mutation operator (only at stage 2)

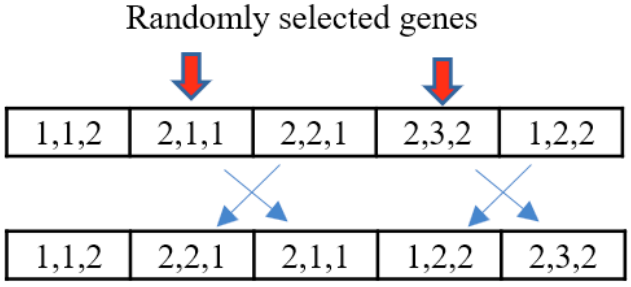

Fig. 5. Swapping mutation operator

We also have two mutation operators, assignment and swapping mutation. At our assignment mutation, illustrated in Figure 4, several genes are randomly selected and their machine assignments will be changed to another alternative. Swapping mutation operator, illustrated in Figure 5, randomly selects several genes and swaps their sequence with the ones right after them if they both do not belong to a same job. Since there is no assignment in stage one, assignment mutation and assignment sequencing are only used in stage two. This is the only difference in utilizing operators in two stages.

3.5 Genetic Parameters

We performed several experiments and found following GA parameters will result in better solutions. Usually mutation are applied with small probability.

population size: 1000

Number of iterations: 3000 for phase one and 7000 for phase two

K-way tournament factor: 0.002

Assignment and sequencing crossover probability: 0.85

Swapping and assignment mutation probability: 0.15

4. NUMERICAL EXAMPLE

The proposed 2SGA has been coded by C++ using Visual Studio on a PC with core i5 2.67 GHz processor and 8GB RAM. To compare the performance of our proposed 2SGA to literature, we have selected two sets of benchmark problems. Brandimarte (1993) introduced a set of 10 problems with 10 to 20 jobs, 5 to 15 operations for each job and 4 to 15 machines. Another benchmark problem set that have been tested was introduced by Hurink et al. (1994) with 10 to 20 jobs, 5 to 10 machines. We also solved theses benchmark problems with a regular GA to illustrate how 2SGA improves GA performance. To have a comparable regular GA, we skipped the first stage of 2SGA (greedy approach stage) and solved the problems with only the second stage that is a regular GA for FJSP.

For solving MK benchmark problem series introduced by Brandimarte (1993), we followed Li and Gao (2016) reporting format who compared result of 21 researches (including their results) who solved these problems with different approaches. Table 1 shows number of job (N) and number of machines (M) of each problem and lists the best makespan reported by each of those 21 researches. Reported researches and their solution approaches listed in column 1 to column 21 are: Tabu Search(TS) by Mastrolilli and Gambardella (2000), Learnable Genetic Architecture(LEGA) by Ho et al. (2007), Hierarchical Optimization(HO) by Zribi et al. (2007), Genetic Algorithm(GA) by Pezzella et al. (2008), Multi-Agent Tabu Search Systems(MAS) by Ennigrou and Ghedira (2008), Hybrid genetic and Variable Neighborhood Descent Algorithm (HGVNA)by Gao et al. (2008), Artificial Immune Algorithm (AIA) by Bagheri et al. (2010), Variable Neighbourhood Search (VNS) by Amiri et al. (2010), Parallel Variable Neighborhood Search (PVNS) by Yazdani et al. (2010), Hybrid Tabu Search (HTS) by Li et al. (2011), Discrepancy Search (DS) by Hmida et al. (2011), Hybrid Chaos Particle Swarm Optimization and GA (HAT) Tang et al. (2011), Artificial Bee Colony (ABC) by Wang et al. (2012), Evolutionary Algorithm (EA) by Chiang and Lin (2013), Hybrid Harmony Search (HHS) by Yuan et al. (2013), Hybrid Harmony and Large Neighborhood Search (HS) by Yuan and Xu (2013), Heuristic by Ziaee (2014), Two-Stage Artificial Bee Colony (TABC) by Gao et al. (2015), Hybrid GA and Tabu Search (HGTS) by Palacios et al. (2015), Multi-objective Memetic Algorithms (MA2) by Yuan and Xu (2015) and Hybrid GA and Tabu Search (HA) by Li and Gao (2016). The last two columns show results of our regular GA and 2SGA approaches.

In order to summarize and make the comparison easier, table 2 shows minimum and maximum makespan values of all 21 reported researches along with our regular GA and 2SGA makespan and conversion CPU time. The last two columns show deviation of 2SGA makespan from min and max values of all other approaches, defined as:

dev = [(2SGA - max or min)/ max or min] * 100

As it can be seen, for every problem, both our regular GA and 2SGA outperformed some other approaches and

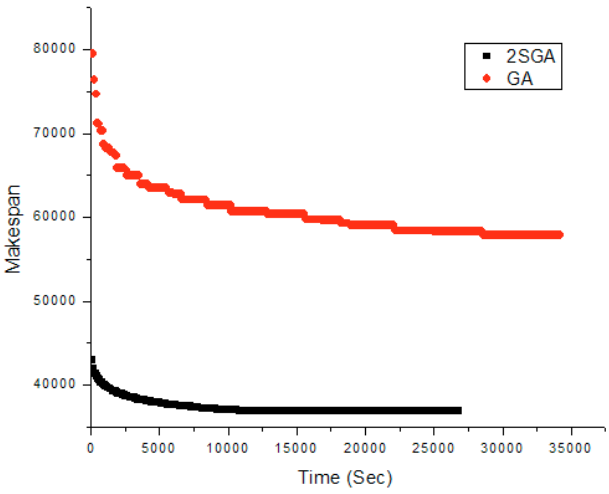

Fig. 6. GA and 2SGA performance comparison to solve a 100x40 FJSP example

2SGA has much better performance than regular GA in terms of conversion time and final result. Also deviation numbers indicates 2SGA results are far better than the maximum values in every case, also they either reached the minimums or very close to it.

Another benchmark data that we utilized to test our algorithm has been introduced by Hurink et al. (1994) and comes in three different instances of "edata" (only few operations with more than one alternative machine), "rdata" (most of operations with more than one alternative machine) and "vdata" (all operations have more than one alternative machine). We selected vdata set as a good example of FJSP. Table 3 contains number of jobs and machines of each problem, reported lower bounds (LB) and best known upper bounds (UB) values from Behnke and Geiger (2012), best results and conversion times of our regular GA and 2SGA. UB values with asterisk have been proven to be optimal solutions. Similar to other bench mark sets, 2SGA outperformed regular GA in every problem at both conversion time and makespan. The last column shows relative deviation of 2SGA results from UB values that indicates 2SGA reaches to optimal solutions in 11 problems and is quite close in the rest. The relative deviation defines as:

dev = [(2SGA-UB)/UB] * 100

In order to have a better comparison of 2SGA and regular GA performances, we also developed a random problem generator to create 5 sample FJSPs of different sizes described in Table 4. Table 5 shows results of proposed 2SGA and regular GA (through skipping the first stage of 2SGA). Results indicate 2SGA significantly outperformed regular GA specially when problem size increases. Figure 6 illustrates performance of 2SGA and GA for solving a FJSP with 100 jobs and 40 machines.

5. DISCUSSION AND CONCLUSION

In this paper we proposed a two-stage GA (2SGA) to solve FJSP that at the first stage GA only determines the order of operations for assignment. Machines are assigned using a greedy approach that chooses an eligible machine

Table 1. Reported solutions for benchmark problems by Li and Gao (2016)

Problem	N	M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	Reg GA	2SGA
MK01	10	6	40	40	41	40	40	40	40	40	40	40	40	40	40	40	40	40	42	40	40	40	40	42	41
MK02	10	6	26	29	28	26	32	26	26	26	26	26	26	26	26	26	26	26	28	26	26	26	26	27	27
MK03	15	8	204	N/A	204	204	N/A	204	204	204	204	204	204	204	204	204	204	204	204	204	204	204	204	204	204
MK04	15	8	60	67	67	60	67	60	60	60	60	62	60	60	60	61	60	60	75	60	60	60	60	68	63
MK05	15	4	173	176	177	173	188	172	173	173	173	172	173	173	172	173	172	172	179	173	172	172	172	175	173
MK06	10	15	58	67	61	63	85	58	63	59	60	65	58	60	60	65	59	58	69	60	57	59	57	71	64
MK07	20	5	144	147	154	139	154	139	140	140	141	140	139	140	139	140	139	139	149	139	139	139	139	144	141
MK08	20	10	523	523	523	523	523	523	523	523	523	523	523	523	523	523	523	523	555	523	523	523	523	559	523
MK09	20	10	307	320	321	311	N/A	307	312	307	307	310	307	307	307	311	307	307	342	307	307	307	307	336	311
MK10	20	15	198	229	219	212	N/A	197	214	207	208	214	197	205	208	225	202	205	242	202	198	202	197	240	205

Note Column (method) labels: 1 = TS; 2 = LEGA; 3 = HO; 4 = GA; 5 = MAS; 6 = HGVNS; 7 = AIA; 8 = VNS; 9 = PVNS; 10 = HTS; 11 = DS; 12 = HAT; 13 = ABC; 14 = EA; 15 = HHS; 16; HS; 17 = Heuristic; 18 = TABC; 19 = HGTS; 20 = MA2; 21 = HA;

Table 2. Comparison between proposed 2SGA, regular GA and min and max of other methods

			Reg						Dev. from	
Problem	N	M	Min	Max	GA	GA t.	2SGA	2SGA t.	Min	Max
MK01	10	6	40	42	42	00:02	41	00:02	2.5%	-2.4%
MK02	10	6	26	32	27	00:44	27	00:01	3.8%	-15.6%
MK03	15	8	204	204	204	00:15	204	00:01	0%	0%
MK04	15	8	60	75	68	02:30	63	00:50	5%	-16%
MK05	15	4	172	188	175	02:06	173	00:08	0.6%	-8%
MK06	10	15	57	85	71	06:34	64	00:12	12.3%	-24.7%
MK07	20	5	139	154	144	03:00	141	00:04	1.5%	-8.4%
MK08	20	10	523	555	523	04:46	523	00:06	0%	-5.8%
MK09	20	10	307	342	336	15:01	311	02:19	1.3%	-9.1%
MK10	20	15	197	242	240	13:03	205	05:53	4.1%	-15.3%

with shortest completion time considering process time and current load of the machine. The second stage is a regular GA to search entire solution space that may be missed at the first greedy stage.

We solved two sets of benchmark problems using proposed 2SGA and regular GA and compared their performance with literature. The results show 2SGA outperforms regular GA in every benchmark problem by archiving a better makespan and better conversion time, also 2SGA performs better than or close to several reported approaches in the literature.

We also generated five sample FJSPs using a random problem generator code to compare performance of 2SGA with regular GA at different problem sizes. Results show 2SGA outperforms regular GA significantly and is a promising method for solving FJSP specially for larger size problems.

Table 3. Best known LB and UB, as well as GA and 2SGA solutions for vdata problems reported by Behnke and Geiger (2012)

Problem	NxM	LB	UB	Reg GA	GA t.	2SGA	2SGA t.	Dev.
LA1	10x5	570	570*	574	01:17	571	00:08	0.2%
LA2	10x5	529	529*	531	00:50	531	00:01	0.4%
LA3	10x5	477	477*	480	01:11	480	00:01	0.6%
LA4	10x5	502	502*	505	00:47	504	00:02	0.4%
LA5	10x5	457	457	465	01:40	458	00:07	0.2%
LA6	15x5	799	799*	802	02:02	800	00:02	0.1%
LA7	15x5	749	749*	752	01:34	750	00:03	0.1%
LA8	15x5	765	765*	768	01:10	766	00:02	0.1%
LA9	15x5	853	853*	855	01:25	853*	00:04	0%
LA10	15x5	804	804*	807	01:22	805	00:03	0%
LA11	20x5	1071	1071*	1076	02:46	1071*	00:05	0%
LA12	20x5	936	936*	938	02:14	936*	00:03	0%
LA13	20x5	1038	1038*	1040	00:30	1038*	00:03	0%
LA14	20x5	1070	1070*	1074	00:38	1070*	00:03	0%
LA15	20x5	1089	1089*	1096	01:34	1089*	00:06	0%
LA16	10x10	717	717*	717	00:31	717*	00:01	0%
LA17	10x10	646	646*	646	01:02	646*	00:01	0%
LA18	10x10	663	663*	663	00:52	663*	00:01	0%
LA19	10x10	617	617*	681	02:58	617*	00:03	0%
LA20	10x10	756	756*	756	00:37	756*	00:01	0%
LA21	15x10	800	806	885	07:19	813	06:05	0.9%
LA22	15x10	733	739	806	03:46	751	00:51	1.4%
LA23	15x10	809	815	882	08:05	824	00:22	1.1%
LA24	15x10	773	777	860	07:01	786	00:08	1.2%
LA25	15x10	751	756	819	05:42	769	02:55	1.7%
LA26	20x10	1052	1054	1059	00:08	1058	00:53	0.4%
LA27	20x10	1084	1085	1169	10:41	1089	00:51	0.4%
LA28	20x10	1069	1070	1145	07:00	1076	02:06	0.6%
LA29	20x10	993	994	1071	08:37	999	02:27	0.5%
LA30	20x10	1068	1069	1162	10:24	1080	08:05	1%

Table 4. Characteristics of example problems

			A		O	
Problem	J	M	Max	Min	Min	Max
5x10	5	10	4	2	4	2
20x10	20	10	4	2	5	3
40x20	40	20	6	3	10	6
50x20	50	20	6	2	16	8
100x40	100	40	8	4	25	12

Note: J = No of Jobs; M = No of Machines; A = Number of alternative machines per operation; O = Number of Operations

REFERENCES

Al-Hinai, N. and ElMekkawy, T. (2011). Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm. *Int. J. Production Economics*, 132, 279–291.

Amiri, M., M.Zandieh, M.Yazdani, and A.Bagheri (2010). A variable neighbourhood search algorithm for the flexible job shop scheduling problem. *Int. J of Production Research*, 48, 5671–5489.

Amjad, M.K., Butt, S.I., Kousar, R., Ahmad, R., Agha, M.H., Faping, Z., Anjum, N., and Asgher, U. (2018). Recent research trends in genetic algorithm based flexible job shop scheduling problems. *Mathematical Problems in Engineering*, 2018, 1–32.

Bagheri, A., M.Zandieh, Mahdavi, I., and M.Yazdani (2010). An artificial immune algorithm for the flexible job shop scheduling problem. *Future Generation Computer. Systems*, 26, 533–541.

Behnke, D. and Geiger, M. (2012). Test instances for the flexible job shop scheduling problem with work centers. *Helmut-Schmidt-University Research Report.*

Table 5. Comparison of 2SGA with regular GA

	2SGA			Regular GA		
Problem	C_{max}*	Iter.	Time	C_{max}	Iter.	Time
5x10	4609	15	00:00	4609	790	00:07
20x10	5854	1590	00:44	6832	5390	02:09
40x20	12552	2265	06:57	17123.5	2615	06:30
50x20	22969	2990	26:45	30982.5	8825	1:08:06
100x40	36923.5	2865	03:08:26	57925	8315	7:55:48

*C_{max} = Makespan; Iter. = Number of iteration

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

Brucker, P. and Schlie, R. (1990). Job-shop scheduling with multi-purpose machines. *Computing*, 45, 369–375.

Chaudhry, I.A. and Khan, A.A. (2016). A research survey: review of flexible job shop scheduling techniques. *Int Transactions in Operational Research*, 23, 551–591.

Chiang, T. and Lin, H. (2013). A simple and effective evolutionary algorithm for multi objective flexible job shop scheduling. *Int. J. of Production Economics*, 141, 87–98.

Defersha, F. and Chen, M. (2010). A parallel genetic algorithm for a flexible job-shop scheduling problem with sequence dependent setups. *Int J of Advanced Manufacturing Technology*, 49, 263–279.

Defersha, F. and Movahed, S. (2018). Linear programming assisted (not embedded) genetic algorithm for flexible jobshop scheduling with lot streaming. *Computers and Industrial Engineering*, 117, 319–335.

Ennigrou, M. and Ghedira, K. (2008). New local diversification techniques for flexible job shop scheduling problem with a multi-agent approach. *Autonomous Agents and Multi-Agent Systems*, 17, 270–287.

Gao, J., Sun, L., and Gen, M. (2008). A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computers and Operations Research*, 35, 2892–2907.

Gao, K., Suganthan, P., T.J. Chua, C.S. Chong, T.C., and Pan, Q. (2015). A two-stage artificial bee colony algorithm scheduling flexible job shop scheduling problem with new job. *Expert Systems with Applications*, 42, 7652–7663.

Gen, M., Gao, J., and Lin, L. (2009). Multistage-based genetic algorithm for flexible job-shop scheduling problem. *Intelligent and Evolutionary Systems*, 187, 183–196.

Hmida, A.B., Haouari, M., Huguet, M., and Lopez, P. (2011). Discrepancy search for the flexible job shop scheduling problem. *Computers and Operations Research*, 37, 2192–2201.

Ho, N.B., Tay, J.C., and M.-K.Lai, E. (2007). An effective architecture for learning and evolving flexible job-shop schedules. *EU J. of Operational Research*, 179, 316–333.

Hurink, J., Jurisch, B., and Thole, M. (1994). Tabu search for the job-shop scheduling problem with multi-purpose machines. *OR Spektrum*, 15.

H.Zhang and M.Gen (2005). Multistage-based genetic algorithm for flexible job-shop scheduling problem. *Journal of Complexity International*, 11, 223–232.

Kachitvichyanukul, V. and Sitthitham, S. (2011). A two-stage genetic algorithm for multi-objective job shop scheduling problems. *Journal of Intelligent Manufacturing*, 22, 355–365.

Li, J., Pan, Q.K., Suganthan, P., and Chua, T. (2011). A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem. *Int. J. of Advanced Manufacturing Technology*, 52, 683–697.

Li, X. and Gao, L. (2016). An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. *Int. J. Production Economics*, 174, 93–110.

Mastrolilli, M. and Gambardella, L.M. (2000). Effective neighbourhood functions for the flexible job shop problem. *Journal of Scheduling*, 3, 3–20.

Palacios, J., Gonzalez, M., C.R.Vela, I.R., and Puente, J. (2015). A genetic tabu search for the fuzzy flexible job shop problem. *Computers and Operations Research*, 54, 74–89.

Pezzella, F., Morganti, G., and Ciaschetti, G. (2008). A genetic algorithm for the flexible job-shop scheduling problem. *Computers and Operations Research*, 35, 3202–3212.

qiu Shi, X., Long, W., Yan-yan Li, Y.l.W., and shan Deng, D. (2018). Different performances of different intelligent algorithms for solving fjsp: A perspective of structure. *Computational Intelligence and Neuroscience*, 2018(doi:10.1155/2018/4617816).

Tang, J.C., Zhang, G., Lin, B., and Zhang, B. (2011). A hybrid algorithm for flexible job shop scheduling problem. *Procedia Engineering*, 15, 3678–3683.

Wang, L., Zhou, G., Xu, Y., Wang, S., and Liu, M. (2012). An effective artificial bee colony algorithm for the flexible job shop scheduling problem. *Int. J. of Advanced Manufacturing Technology*, 60, 303–315.

Wang, Y., Xiao, N., Yin, L., Hu, E., Zhao, C., and Jiang, Y. (2008). A two-stage genetic algorithm for large size job shop scheduling problems. *Int. J. of Advanced Manufacturing Technology*, 39, 813–820.

Yazdani, M., Amiri, M., and Zandieh, M. (2010). Flexible job shop scheduling with parallel variable neighborhood search algorithm. *Expert Systems with Applications*, 37, 678–687.

Yuan, Y. and Xu, H. (2013). An integrated search heuristic for large-scale flexible job shop scheduling problems. *Computers and Operations Research*, 40, 2864–2877.

Yuan, Y. and Xu, H. (2015). Multi objective flexible job shop scheduling using memetic algorithms. *IEEE Transactions on Automation Science and Engineering*, 12, 336–353.

Yuan, Y., Xu, H., and Yang, J. (2013). A hybrid harmony search algorithm for the flexible job shop scheduling problem. *Applied Soft Computing*, 13, 3259–3272.

Ziaee, M. (2014). A heuristic algorithm for solving flexible job shop scheduling problem. *Int. J. of Advanced Manufacturing Technology*, 71, 519–528.

Zribi, N., Kacem, I., Kamel, A., and Borne, P. (2007). Assignment and scheduling in flexible job shops by hierarchical optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 37, 652–661.