ISSN 1392–1207. MECHANIKA. 2017 Volume 23(4): 610–616

A novel algorithm for solving job-shop scheduling problem

A. Muthiah*, R. Rajkumar**
**Department of Mechanical Engineering, P.S.R. Engineering College, Sivakasi, India E-mail: amuthiah68@gmail.com*
***Department of Mechanical Engineering, Mepco Schlenk Engineering College, Sivakasi, India*

crossref http://dx.doi.org/10.5755/j01.mech.23.4.14055

1. Introduction

The job shop scheduling problem (JSSP) is to decide a schedule of jobs that is endowed with pre-set operation series in a multi-machine atmosphere. In the traditional JSSP, n-jobs are processed to the finishing point on m-unrelated machines. For each and every task, technology limitations spell out an absolute and distinctive routing which is set and identified earlier. In addition, processing periods are set and identified previously [1]. A number of production resources are available which are utilized according to diverse methods by the task for processing. In fact, each job has its own specific route in the production site. Every stage of the routing is a unitary function performed by one/multiple resource (s) in the course of a specified processing interval. Nevertheless, a resource is incompetent to carry out two operations simultaneously, in other words resources are non-cumulative [2]. The problem of scheduling jobs in FJSP was divided into two subproblems. The first one was the routing that allocates every task to a machine chosen from among a set of competent machine, while the next was the scheduling that includes sequencing the allocated tasks on all machines so as to achieve a viable schedule to reduce the predefined objective task [3]. For each task there is a job to which it belongs, a machine on which it has to be processed, a prearranged processing interval on that machine as well as a preset processing order on the machines. The issue is to reduce the makespan simultaneously making certain that multiple jobs cannot be processed together on the same machine, and also ensuring that when a job is commenced, it has to be somehow finished [4]. But, FJSSP is rather complicated than the traditional JSP, as it brings in an additional decision level in addition to the sequencing one such as the job routes. Decision on the job routes includes determining, for every task, what machine must be used to process it, from among the existing ones [5]. In the contemporary flexible manufacturing systems endowed with computer-controlled robots, hoists, cranes, and other material handling mechanisms, setup and transport intervals are important and have to be given due consideration. Such transporting devices are overtly integrated into the analogous scheduling models [6]. A solution was entrusted with the task of evaluating the operation series on the machines to meet with certain restrictions. In this document, the reduction of makespan was deemed as the prominent target. It is characterized as the overall time between the commencement of the initial task and the end of the final task in all jobs [7]. In the makespan reduction issue under linear deterioration of the two-machine flow shop issue is powerfully NP-hard. It was demonstrated that for the three-machine flow shop problem with simple linear deterioration, there was no polynomial-time approximation algorithm with a worst-case ratio bounded by a constant [8]. A single-machine scheduling problem with uneven release intervals for optimizing the makespan in which the learning effect and the deteriorating jobs were simultaneously taken in to account. Various dominance standards and the lower limits are set up to allow the branch-and-bound algorithm for achieving the optimal solution [9]. In predictive-reactive scheduling technique a production schedule is created initially and thereafter revised in the reaction to a disruption of real time events so as to reduce its influence on system excellence, whereas in reactive scheduling scheme only fractional schedules are produced when needed for the instantaneous future in accordance with the existing system status and restricted data and limitations [10]. The dimension robustness is essentially categorized into two types such as quality robustness and solution robustness. The former is generally employed to specify the insensitivity of the schedule excellence under ambiguity in respect of the objective value, like makespan and tardiness while the latter generally signifies the insensitivity of task commencement or completion intervals to the ambiguity. From the job shop scheduling, the optimization algorithm is employed to estimate the least makespan interval within the optimal schedule [11].

2. Abbreviation

AGA	– Adaptive Genetic Algorithm
BKS	– Best Known Solution
FJSP	– Flexible Job Shop Problem
FJSSP	– Flexible Job Shop Scheduling Problem
GA	– Genetic Algorithm
JSSP	– Job Shop Scheduling problem
OGA	– Opposite Genetic Algorithm

3. Related works

Sureshkumar et al. [12] proposed the unordered subsequence exchange crossover in genetic algorithm for reducing the makespan interval in the JSSP. To reduce the makespan duration and locate the optimal schedule special crossover technique, unordered subsequence exchange crossover was proposed in genetic algorithm (GA). Using the special cross over technique unordered subsequence exchange crossover the majority of the standard outcomes were assessed and contrasted and the outcomes achieved were more or less close to optimal value of the standard issues. From the encouraging outcomes, it was clear that the innovative algorithm had achieved optimal upshots of the standard issues and several of the realized outcomes were observed in proximity to optimal and with the employment of the unordered subsequence exchange crossover several benchmark issues outcomes were realized well within the minimum number of iterations.

Antonin Ponsich et al. [13] have proposed a hybrid differential evolution-tabu search algorithm for the resolution of JSSP. The Differential Evolution (DE) had established its unique position as a highly proficient method for incessant optimization, though it failed miserably to yield a superb performance when applied to transformation issue. Therefore, an innovative hybridizing DE with Tabu Search (TS) was employed to tackle the issues of the JSSP. The computational investigations in respect of cases exceeding 100 JSSP instances illustrated the unquestioned fact that the novel hybrid DE–TS algorithm was competent to yield stunning performance when compared with those of the several high-tech approaches. The amazing outcomes have illustrated that the width of confidence intervals was less than 0.7% of the reference makespan value, establishing the fact that low dispersion of the solutions was realized.

Chen et al .[14] exhibited an innovative flexible job shop scheduling with parallel machines by means of genetic algorithm and grouping genetic algorithm (GGA). This paper possessed two modules such as the machine selection module (MSM) and operation scheduling module (OSM). The MSM extended a helping hand to an operation to choose one of the matching machines to process it and the OSM was thereafter employed to organize the series of the entire operations allocated to each and every machine. The outcomes upheld the preeminence of the combination of MSM employing GGA and OSM using GA which went on zooming as and when the number of orders augmented. The result demonstrated the innovative method using GGA and GA were well-equipped to allocate a machine to an operation and organize the series of operations at each machine to bring in lesser tardiness, machine idle interval, and makespan.

DarrellLochtefeld et al. [15] used the helper-objective optimization strategies for the JSSP. The Multiple Objectives Evolutionary Algorithms (MOEAs) was executed on the JSSP and it was proved that they performed superior to the single objective GA. Helper-objectives, characterizing segments of the fundamental goal, were exceedingly helpful in guiding MOEAs in the search procedure. The series in which helper-objectives were employed which corroborated the fact that problem-specific skills could be integrated to decide a helper-objective sequence. Computational outcomes well exhibited the manner by which cautiously sequenced helper objectives were competent to step up search excellence. This resulted in the summary rejection of the traditional practice of picking helper sequence based on arbitrary order on account of the deficiency of comprehension about optimal sequencing.

Lin et al. [16] were credited with introducing an innovative technique of particle swarm optimization for job-shop scheduling. In this paper, a hybrid swarm intelligence algorithm integrating particle swarm optimization, simulated annealing technique and multi-type individual enhancement scheme was introduced to find viable solutions to the JSSP. It was evident from the outstanding outcomes that the difference between the MPSO's Average and the BKS were limited to less than 2% and MPSO were competent to achieve the optimal area in the search space with lesser population dimension and get better solutions by exploiting the superior individual enhancement aptitude.

4. Proposed methodologies

A novel algorithm, Opposition Genetic Algorithm (OGA) is developed to find the minimum makespan time within the optimal scheduling. The minimum makespan value is found by using the fitness computation of the algorithm. In opposition based algorithm, the chromosomes of the algorithm i.e., solution sets are generated and its opposition chromosomes are calculated for finding the best chromosome of the solution by using the fitness value. The opposition chromosomes are finding the two extremes solutions of the chromosomes to find the best solutions. Then, the cross over and mutation process of the normal genetic algorithm process is applied on it for finding the best optimal value for the minimum makespan time. Finally the results are compared with Genetic Algorithm (GA) and Adaptive Genetic Algorithm (AGA).

4.1. Constraints of the proposed algorithm to solve JSSP

- Each job must be processed in the allocated time.
- Each job can be processed with the shortest time such that it completes within short period of time.
- At a given time, a machine can execute at most one operation.

4.2. Genetic Algorithm (GA)

Fig 1 depicts the GA. In genetic algorithm optimization technique the chromosome in the hidden layer and neurons are considered and then the fitness function calculated. Based on fitness value, the cross over is created. These individuals reproduce the offspring and after that the offspring are mutated randomly. Then the fitness value is found and checked with the other solutions and the minimum error value obtained.

4.3. Adaptive Genetic Algorithm (AGA)

In usual genetic algorithm, the mutation rate or probability is stable for all chromosomes to unearth the fitness function. So there is no altering in any values for the best fitness values for the best chromosomes. The adoptive genetic algorithm is used to tune the value near the best value with the less time. AGA process crossover and mutation based on the rate value is shown in Eqs. (1) and (2).

Cross over rate:

$$P_C = \left(f_{max} - f\right) / \left(f_{max} - f_{avg}\right), \quad f \geq f_{avg} . \tag{1}$$

Mutation rate

$$P_C = 0.5\left(f_{max} - f\right) / \left(f_{max} - f_{avg}\right), \quad f \geq f_{avg} , \tag{2}$$

where f_{max} represents the maximum fitness value of the population, f_{avg} the average fitness value of the population, *f* the fitness value of the solution undergone mutation. *f* the larger of the fitness values of the solutions are to be crossed.

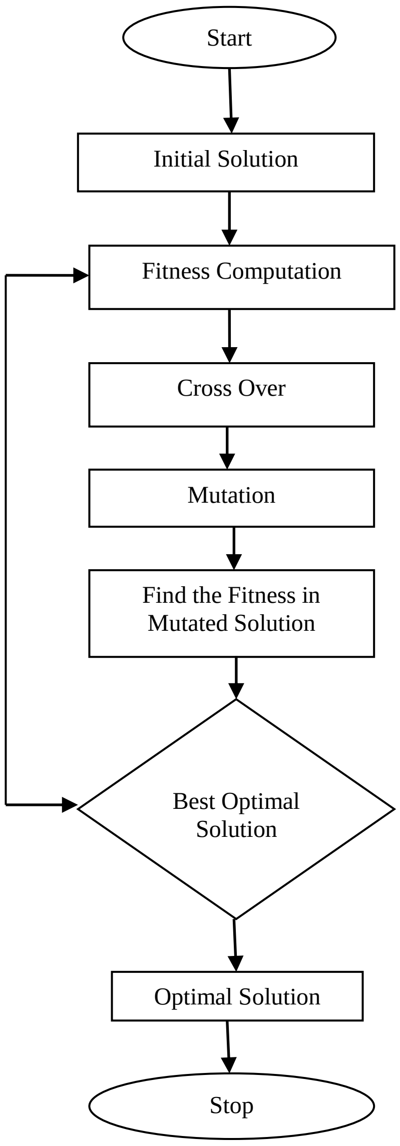

Fig. 1 Genetic algorithm

4.4. Opposition Genetic Algorithm (OGA)

In OGA, first a population is created with a group of individuals randomly. The individuals in the population are then evaluated. The evaluation function is provided by the user. Next two individuals are selected based on their fitness (higher fitness individuals are selected). Then these individuals will reproduce to create one or more offspring. After that the offspring are mutated randomly. This process continues until a suitable solution has been found or a certain number of generations have passed, depending on the needs of the user.

4.4.1. Making of chromosomes

The initial solutions are generated randomly and each solution is called gene. The individual gene is combined as chromosome and it is called the solution set. The numbers of genes are combined the chromosomes and created the solution set for the population. The population of genetic algorithm is composed of chromosomes and the population size is initialized as fixed. The numbers of solutions are initialized based on the normal genetic algorithm.

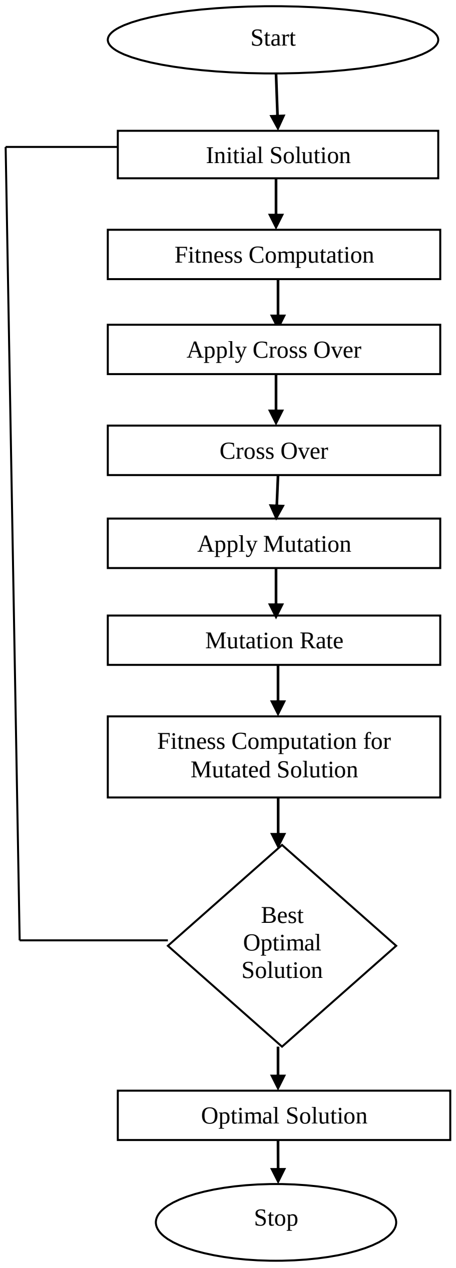

Fig. 2 Adaptive Genetic Algorithm

The Opposition based genetic algorithm is varied from the normal genetic in this part and calculate the opposition chromosome solution set by using the following formula.

$$Y_j = p_j + q_j - y_j \,, \tag{3}$$

where *Y* is the opposition vector, *y* is within the interval of [*p*, *q*]. The *p* and *q* are the boundary conditions. The ‘*j*’ is the number of solutions. The randomly generated solution sets are generating its opposition chromosomes by using the Eq. (3).

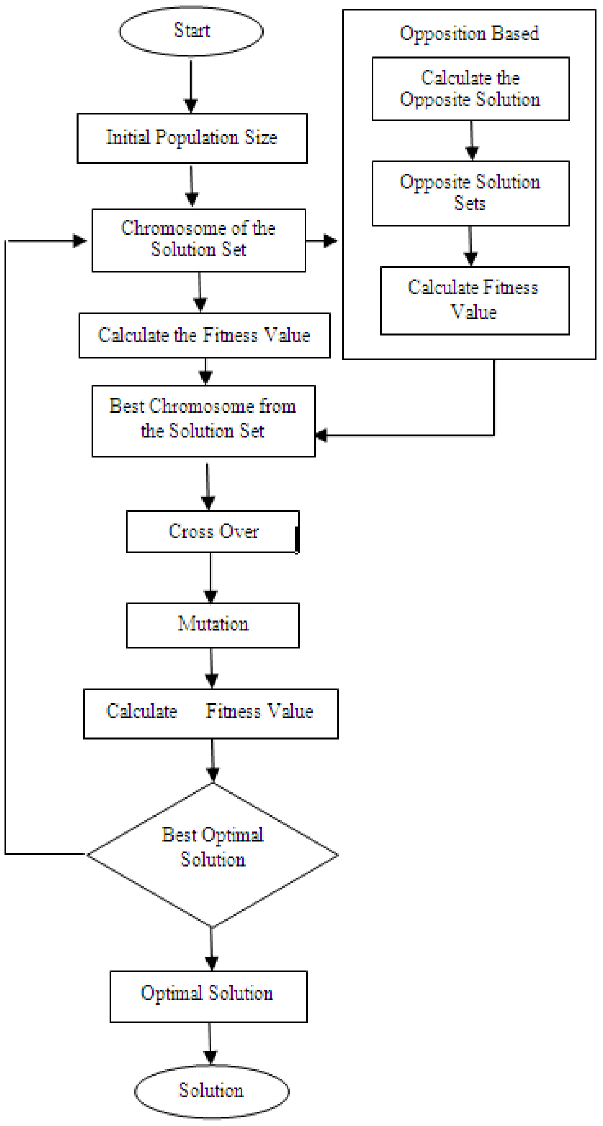

Fig. 3 Flow chart for opposition genetic algorithm

The maximum and minimum value of the each solution is known and it is used for generating the opposition based chromosome solution sets. Its population size is same for its randomly generated population size. For example, the population size is 10 and its opposition based solution set is 10 because the opposition based solutions are determined by using the its initial solution sets.

4.4.2. Fitness function

The following formula is used to find the fitness function for calculating the makespan time and it is:

$$Fitness = min\left(Y\left(t\right)\right), \tag{4}$$

where, the $Y(t)$ is the makespan time:

$$Y\left(t\right) = max\left(T_{i-1,j}, T_{i,j-1}\right) + T_{i,j}, \tag{5}$$

where the T is the processing time, the 'i' is the job order and j is the machine order. The fitness function is calculated for the initial solution sets of the chromosomes and the opposition based solution sets for finding the best chromosomes. After the solutions sets are evaluated, the cross over and mutation function are applied on the chromosomes of the solutions sets.

4.4.3. Cross over

In our proposed method, the cross over function is based on the job order and the parent chromosomes are shown in the Fig. 4. In cross over, the two parent chromosomes are taken to exchange their genes within them. The following Fig. 4 denotes the parent chromosomes parent 1 and parent 2. The unordered crossover is used for the crossover operation. In parent 1 and 2 chromosomes, the bold lettered job orders are unchanged in their positions and remaining gene of the chromosomes is exchanged between the parent chromosomes. After the crossover, the chromosome is shown below in the Fig. 5.

3	**2**	**1**	3	1	2	3	2	1

Parent-1

1	2	3	**1**	**3**	**2**	3	1	2

Parent-2

Fig. 4 Parent chromosomes

3	1	2	**1**	**3**	**2**	3	2	1

New Chromosome-1

3	**2**	**1**	1	2	3	3	1	2

New Chromosome-2

Fig. 5 New Chromosomes

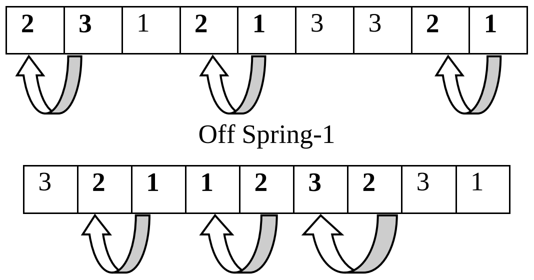

Fig. 6 Mutated off spring

4.4.4. Mutation

After the crossover, the child chromosome is mutated for increasing the efficiency of the solution and the bold letter shows the mutated gene of the chromosome. In our proposed method mutation, the same job order is selected within the offspring and it is interchanged from its position to other place for giving the best optimal solution The shift changing mutation method is used in the mutation operation and the job orders are shift to left one step and replacing by the new job order. After the shift changing within the off-spring is shown in Fig. 5. From the Fig. 6, the gene of the off spring is shifted one step left and the optimized new solution is getting by the mutation process. The optimal solution is obtained after the mutation function and it shows the final output of the result with their minimal optimized time and it gives the minimum makespan time.

4.4.5. Optimal solution

At end of the mutation, the new chromosomes are generated for the new solution sets. Then, the fitness value is finding for the new solutions. From that, which solution is given the minimized makespan time and it is used as optimal solution otherwise the above steps are processed for the new solution sets.

5. Result and discussion

The different benchmarks problems are used to find the minimum make span time and its operation are shown below with the minimum make span time. The following FT20 (20X10), LA01 (10X10), LA06 (15X10), LA16 (10X20), LA21 (15X20), LA26 (20X20), LA22 (15X20), LA27 (20X20), LA28 (20X20) and LA31 (30X20) benchmark problems are used to find the minimum make span time.

In the tabular columns processing times and completion times of the each job is separately shown. In completion time tabular columns, the process sequence is shown as the operation sequence which is denoted as the O11, O12….up to its end of the number of operations. In all benchmark problem minimum make span time produced in Opposition Genetic Algorithm (OGA). The following tables are shows that the processing time of each problem size with their jobs and machines. The minimum make span time calculating processing matrix for GA, AGA and OGA and minimum time attained in OGA. In the above tables are shown the different benchmark problem processing time durations and the minimum make span time of the OGA process. The job sequences furnish the minimum make span time for all the jobs and they have the input possessing time durations which are added with those of the other jobs. The minimum make span time is given at the end of the complete operation of the all the jobs. After all the jobs are finished, the opposite genetic algorithm (OGA) yields the minimum time with the corresponding job sequence. These job sequences are processed to give the minimum make span time, which is added to their corresponding job sequence input time. Here, if the problem size varies, the processing time is also found to vary based on the number of the machine operated. For FT20 (20X10) the minimum make span time of the process viz. 1147 is attained in 9 jobs performed in 20 machines in the OGA algorithm and is compared to the other two algorithms as shown in Table 1. For LA01 (10X10) the target and predicted values are 666 and 658 respectively for 5 machines. The corresponding job sequence is different for 10X10 and it is brought along with the minimum number of iterations for all the problems in Fig. 7. The minimum make span time is obtained at the above job sequence and added with their corresponding job sequence input times. For LA22 (15X20) and LA06 (15X10) the minimum make span time values are attained in the OGA technique in 15 jobs performed in 9 machines and they are 868 and 920 respectively. These job sequences are processed to yield the minimum make span time. Similarly the minimum make span time is obtained in the OGA technique compared to the GA and AGA processes.

Table 1
Comparison of bench mark problems - GA, AGA and OGA

Sl.No	Instance	Size $n \times m$	BKS	GA	AGA	OGA
1	ft06	6 × 6	55	59	62	56
2	ft10	10 × 10	930	1105	1087	925
3	**ft20**	**20 × 5**	**1165**	**1170**	**1147**	**1035**
4	**la01**	**10 × 5**	**666**	**698**	**671**	**658**
5	**la02**	**10 × 5**	**655**	**742**	**826**	**653**
6	la03	10 × 5	597	647	682	597
7	la04	10 × 5	590	692	702	590
8	la05	10 × 5	593	685	715	593
9	**la06**	**15 × 5**	**926**	**928**	**876**	**868**
10	la07	15 × 5	890	987	847	890
11	la08	15 × 5	863	965	947	863
12	la09	15 × 5	951	1219	1248	951
13	la10	15 × 5	958	1325	1352	958
14	la11	20 × 5	1222	1398	1402	1222
15	la12	20 × 5	1039	1210	1185	1039
16	la13	20 × 5	1150	1324	1344	1150
17	la14	20 × 5	1292	1457	1482	1292
18	la15	20 × 5	1207	1420	1455	1207
19	**la16**	**10 × 10**	**945**	**1057**	**1083**	**941**
20	la17	10 × 10	784	962	987	784
21	la18	10 × 10	848	991	1020	848
22	la19	10 × 10	842	984	1014	842
23	la20	10 × 10	902	1065	1074	902
24	**la21**	**15 × 10**	**1046**	**1276**	**1352**	**1044**
25	**la22**	**15 × 10**	**927**	**1254**	**1284**	**920**
26	la23	15 × 10	1032	1287	1302	1032
27	la24	15 × 10	935	1112	1144	935
28	la25	15 × 10	977	1025	1035	977
29	**la26**	**20 × 10**	**1218**	**1766**	**1592**	**1211**
30	**la27**	**20 × 10**	**1235**	**1692**	**1657**	**1227**
31	**la28**	**20 × 10**	**1216**	**1715**	**1612**	**1215**
32	la29	20 × 10	1152	1345	1247	1152
33	la30	20 × 10	1355	1457	1366	1355
34	**la31**	**30 × 10**	**1784**	**1998**	**1952**	**1772**
35	la32	30 × 10	1850	2014	1989	1850
36	la33	30 × 10	1719	1857	1892	1719
37	la34	30 × 10	1721	1951	1976	1721
38	la35	30 × 10	1888	2089	1988	1888
39	**la36**	**15 × 15**	**1268**	**1931**	**1893**	**1118**
40	la37	15 × 15	1397	1549	1452	1397
41	la38	15 × 15	1196	1246	1168	1196
42	la39	15 × 15	1233	1412	1386	1233
43	la40	15 × 15	1222	1397	1287	1222

Table 1 illustrates the minimum make span time for the various benchmark problems along with their make span time. The minimum make span time is estimated only after the entire operations are finished. It is clear from the graph that the best minimum make span time is realized in the OGA technique vis-à-vis the AGA and GA. The optimal make span time is subjected to analysis and comparison with the identified optimal makespan time. It is gratifying to note that the innovative technique has ushered in an optimal makespan time value which goes hand in hand with the identified optimal value.

Fig. 7 elegantly exhibits that the proposed OGA performs well than GA, and AGA for several benchmark problems. In respect of FT20 (20X10) the minimum make span time is 1035 which is attained in the OGA in the 42-th iteration. When it is compared with the GA, the time difference is 135 and the minimum time of the GA is attained in the 41-th iteration. In the case of the AGA also the minimum time is attained in the 45-th iteration. For this problem the maximum time is 1170 in the GA. In respect of the problem size LA01 (10X10), the minimum makespan time achieved for the GA is 698. Here, the original time is 666 and the minimum time is attained in the 39-th iteration in the OGA process. In LA06, the original time is found to be 926, and for GA and AGA the difference is 2 and 50 while for OGA is 58. Similarly for LA16 (10X20) also, the minimum processing time is 941 for OGA and when compared with the original processing time it is the minimum time for the job scheduling process. Finally for the problem size LA21 (15x20), when the make span time of the OGA is compared with the GA the difference is found to be 232. Similarly for the AGA the minimum time of 308 is attained in the 44th iteration of the AGA process. For the problem size LA22 (15X20). It is crystal clear from the graph that the minimum make span time achieved is 920 for the AGA in the 47 iteration. When it is compared with that of the (OGA) the difference is 73.

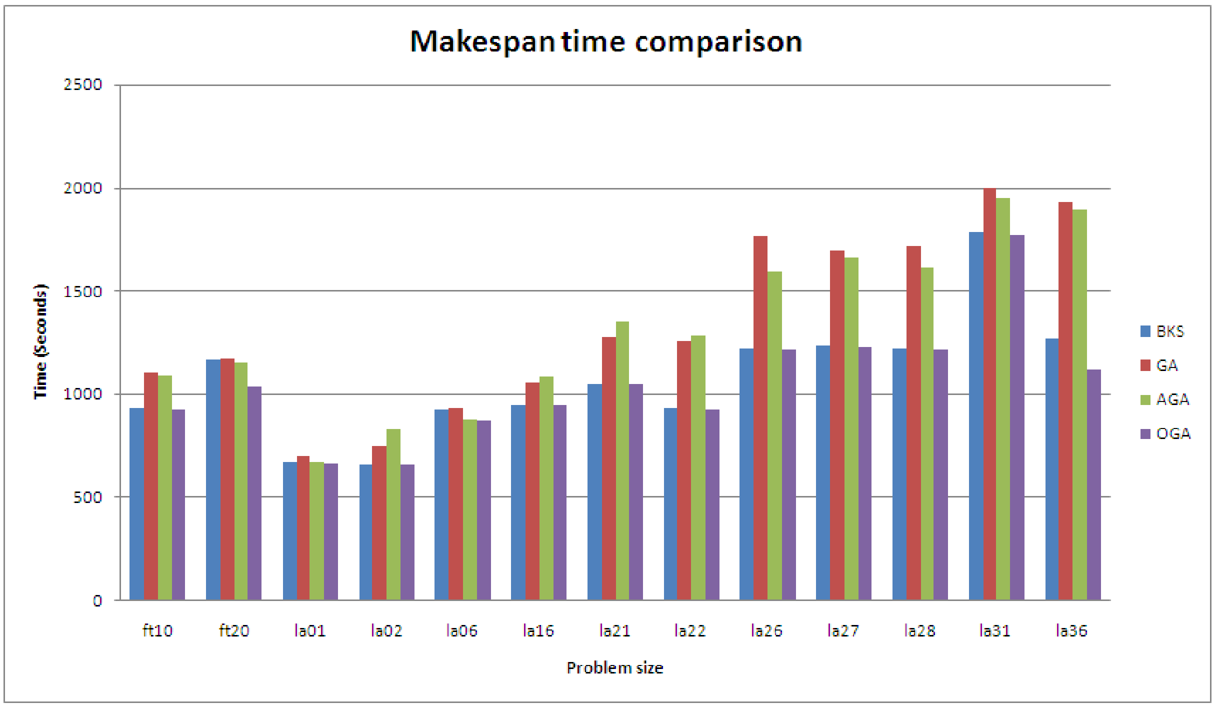

Fig. 7 Makespan time comparison for bench mark problems

In respect of the problem size LA26 (20X20) the maximum time for OGA is estimated as 1211. As far as the problem size LA27 (20X20) is concerned ,the original time and the minimum make span time are 1235 and 1227 respectively and the time difference in the OGA with the original time is found to be 99.23%. For LA28 (20X20) the minimum makespan time is estimated as 1215 for the OGA technique which is achieved in the 45-th iteration. In respect of the LA31 (20X20) also, the minimum make span time of 1772 is realized in OGA technique.

6. Conclusion

The proposed opposite genetic algorithm has achieved the least make-span interval vis-à-vis those of the rivals such as the adaptive genetic algorithm and genetic algorithm. The make-span interval is estimated for various standard challenges like FT20 (20X10), LA01 (10X10), LA06 (15X10), LA16 (10X20), LA21 (15X20), LA26 (20X20), LA22 (15X20), LA27 (20X20), LA28 (20X20) and LA31 (30X20) and they are assessed and contrasted with those of the rival methods. The fascinating results substantiate the fact the ground-breaking opposite genetic algorithm has come out with flying colors by ushering the least make-span interval for all standard issues addressed. The makespan time is minimized by using the different algorithms such as genetic algorithm, adaptive genetic algorithm. The different benchmark problems are applied for finding the minimum makespan time. The developed algorithms can be integrated with other heuristics algorithms to develop hybrid algorithms in future.

References

1. **Fattahi, P.; Jolai, F.; Arkat, J.** 2009**.** Flexible job shop scheduling with overlapping in operations, Applied Mathematical Modelling 33: 3076-3087. http://dx.doi.org/10.1016/j.apm.2008.10.029.

2. **Prot, D.; Bellenguez-Morineau, O.** 2012**.** Tabu search and lower bound for an industrial complex shop scheduling problem, Computers and Industrial Engineering 62: 1109-1118. http://dx.doi.org/10.1016/j.cie.2012.01.003.
3. **Guohui Zhang.; Liang Gao.; Yang Shi.** 2011**.** An effective genetic algorithm for the flexible job-shop scheduling problem, Expert Systems with Applications 38: 3563-3573. http://dx.doi.org/10.1016/j.eswa.2010.08.145.
4. **Heinonen, J.; Pettersso, F.** 2007**.** Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem, Applied Mathematics and Computation 187: 989-998. http://dx.doi.org/10.1016/j.amc.2006.09.023.
5. **Pezzella, F.; Morganti, G.; Ciaschetti, G.** 2008**.** A genetic algorithm for the Flexible Job-shop Scheduling Problem, Computers & Operations Research 35: 3202-3212. http://dx.doi.org/10.1016/j.cor.2007.02.014.
6. **Levner, E.; Kats, V.; Alcaide D.; De Pablo, L.; Cheng, T.C.E.** 2010**.** Complexity of cyclic scheduling problems: A state-of-the-art survey, Computers & Industrial Engineering 59: 352-361. http://dx.doi.org/10.1016/j.cie.2010.03.013.
7. **Liang Gao; Guohui Zhang; Liping Zhang; Xinyu Li** 2011**.** An efficient memetic algorithm for solving the job shop scheduling problem, Computers & Industrial Engineering 60: 699-705. http://dx.doi.org/10.1016/j.cie.2011.01.003.
8. **Xiao-Yuan Wang; Ming-Zheng Wang; Ji-Bo Wang** 2011**.** Flow shop scheduling to minimize makespan with decreasing time-dependent job processing times, Computers & Industrial Engineering 60: 840-844. http://dx.doi.org/10.1016/j.cie.2011.02.003.
9. **Yu-Hsiang Chung; Lee-Ing Tong** 2012**.** Bi-criteria minimization for the permutation flow shop scheduling problem with machine-based learning effects, Computers & Industrial Engineering 63: 302-312. http://dx.doi.org/10.1016/j.cie.2012.03.009.
10. **Li Nie; Liang Gao; Peigen Li; Xinyu Shao** 2013**.** Reactive scheduling in a job shop where jobs arrive over time, Computers & Industrial Engineering 66: 389-405. http://dx.doi.org/10.1016/j.cie.2013.05.023.
11. **Jian Xiong.; Li-ning Xing.; Ying-wu Chen.** 2013**.** Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns, International Journal of Production Economics 141: 112-126. http://dx.doi.org/10.1016/j.ijpe.2012.04.015.
12. **Antonin Ponsich; Carlos A. Coello** 2013**.** A hybrid Differential Evolution—Tabu Search algorithm for the solution of Job-Shop Scheduling Problems, Applied Soft Computing 13: 462-474. http://dx.doi.org/10.1016/j.asoc.2012.07.034.
13. **James C. Chen; Cheng-Chun Wu; Chia-Wen Chen; Kou-Huang Chen** 2012**.** Flexible job shop scheduling with parallel machines using Genetic Algorithm and Grouping Genetic Algorithm, Expert Systems with Applications 39: 10016-1002. http://dx.doi.org/10.1016/j.eswa.2012.01.211.
14. **Darrell F. Lochtefelda; Frank W. Ciarallo** 2011**.** Helper-objective optimization strategies for the Job-Shop Scheduling Problem, Applied Soft Computing 11: 4161-4174. http://dx.doi.org/10.1016/j.asoc.2011.03.007.
15. **Tsung-Lieh Lin; Shi-Jinn Horng; Tzong-Wann Kao; Yuan-Hsin Chen; Ray-Shine Run;Rong-Jian Chen; Jui-Lin Lai; I-Hong Kuo** 2010**.** An efficient job-shop scheduling algorithm based on particle swarm optimization, Expert Systems with Applications 37: 2629-2636.
16. **Amjad Iqbal; Naveed Kazim Khan; Arfan Jaffar; Ramzan; Rauf Bai** 2010**.** Opposition based Genetic Algorithm with Cauchy Mutation for Function Optimization, in proceedings of IEEE Information Science and Applications, p.1-7.

A. Muthiah, R. Rajkumar

A NOVEL ALGORITHM FOR SOLVING JOB-SHOP SCHEDULING PROBLEM

S u m m a r y

Of late, Scheduling optimization is the highly significant hassles in the job shop. The supreme advantage of the job shop scheduling is that the conclusion period of the entire tasks is cutback to the minimum possible. The job shop scheduling problem takes its origin from the conventional job shop scheduling problem, which is equipped with extensive accessibility of machines for all the entire tasks to be completed. Taking into account the two phases of the issue, two diverse definitions such as total flexibility and partial flexibility are envisaged to segregate the diverse accessibility data of machines. It is cheering that the efficiency has been scaled up by means of the opposite genetic algorithm which has proved its mettle as an effectual and proficient mechanism for successfully addressing the pointed issue of job-shop scheduling. Moreover, it is effectively employed to find out the least optimal makespan time with the optimal solution.

Key words: Job shop scheduling, makespan time, adaptive genetic algorithm, opposite genetic algorithm.

Received January 28, 2016
Accepted August 04, 2017