Improved Genetic Algorithm Integrated with Scheduling Rules for Flexible Job Shop Scheduling Problems

Muhammad Kamal Amjad[1,*], *Shahid* Ikramullah Butt[1] and *Naveed* Anjum[1]

[1]School of Mechanical and Manufacturing Engineering (SMME), National University of Science and Technology (NUST), Islamabad, Pakistan

Abstract. This paper presents optimization of makespan for Flexible Job Shop Scheduling Problems (FJSSP) using an Improved Genetic Algorithm integrated with Rules (IGAR). Machine assignment is done by Genetic Algorithm (GA) and operation selection is done using priority rules. Improvements in GA include a new technique of adaptive probabilities and a new forced mutation technique that positively ensures the generation of new chromosome. The scheduling part also proposed an improved scheduling rule in addition to four standard rules. The algorithm is tested against two well-known benchmark data set and results are compared with various algorithms. Comparison shows that IGAR finds known global optima in most of the cases and produces improved results as compared to other algorithms.

1 Introduction

Manufacturing scheduling deals with the decision making problem of allocating a predetermined set of resources to perform a set of activities such that the activities can be completed in an optimal manner with regards to a certain performance parameter.

Job Shop Scheduling Problem (JSSP) deals with the sequencing of a certain number of operations on several fixed machines with predefined task sequences and processing times. Thus, operations can be performed on a predefined set of machines only; however, different sequences offer to optimize the overall schedules. The Flexible Job Shop Scheduling (FJSSP) consists of two sub-problems i.e. assignment and scheduling [1]. In assignment part of problem, operations are assigned to available machines, whereas, in scheduling part, assigned operations are scheduled / sequenced on all available machines. Scheduling objectives are used to assess the optimality of a reference schedule.

Section 2 of this paper provides relevant literature review for the FJSSP using GA. Section 3 deals with the problem formulation. Section 4 provides description of the proposed GA used for solving the FJSSP. Section 5 deals with the analysis of computational results and Section 6 concludes the paper.

2 Literature review

Scheduling problems are known to be NP-hard [2], whereby, it becomes virtually impossible for the exact methods to provide the solution in a reasonable time. Thus, approximate algorithms have been used traditionally to acquire acceptable solutions.

Genetic Algorithms (GAs) mimic the evolutionary intelligent behaviour of nature for evolution of generations based upon the rule of 'survival of the fittest'. GA has been used most commonly to solve the JSSP out of the available artificial intelligence based techniques [3]. Similarly, it has been pointed out that pure GA based applications constitute 54.69% of the GA based FJSSP literature [4]. This clearly identifies the popularity and adaptability of the algorithm for solving the FJSSP.

Makespan minimization has been conducted mostly in the literature as single objective optimization [4]. Qiao et al [5] used adaptive crossover and mutation with roulette wheel selection to minimize the makespan. Yang et al [6] used Non-dominating Sorting Genetic Algorithm (NSGA) for multi-objective optimization of makespan, maximal workload, total workload and total tardiness. They used an adaptive mutation method, where they used smaller mutation rates at start of evolution and bigger mutation rates at the end. Pan et al [7] used an adaptive GA with two part operations and machine representation chromosome for minimization of makespan. Mutation, crossover and selection parameters are adaptable according to linear interpolation mechanism which generates the respective values for each generation.

Kaweegitbundit et al [8] use hybrid scheduling rules for machine selection part for minimization of mean tardiness. Doh et al [9] conducted a comparative study on 36 priority rule combinations (separately for machine selection and operation sequencing) for the optimization of makespan and other objectives in FJSSP environment. The scheduling technique was tested against an industrial system to show that Shortest Processing Time (SPT) provided best results in all scenarios.

* Corresponding author: kamal.amjad@smme.edu.pk

In this paper, new adaptive strategies have been proposed for recombination operators and a modified Most Work Remaining (mMWR) scheduling rule is also developed. The algorithm is tested against two benchmark problem sets of Kacem [10] and Fattahi [11]. Makespan (C_{max}) is the optimization objective used in this paper. Table 1 presents the notations used in this paper.

Table 1. Notations

Symbol	Description
N	Total number of jobs
M	Total number of machines
L	Total number of sequences
i	Index of i^{th} job out of N
J_{io}	Total number of operations of i^{th} job
j	Index of j^{th} operation out of J_{io}
O_{ij}	j^{th} operation of i^{th} job
Ω_{ij}	Set of available machines for O_{ij}
k	Index of k^{th} machine out of Ω_{ij}
M_k	k^{th} machine out of M
P_{ijk}	Process time of O_{ij} on M_k
r_{ijk}	Release time of O_{ij} on M_k
t_{ijk}	Start time of O_{ij} on M_k
E_{ijk}	End time of O_{ij} on M_k
C_k	Completion time of current process on M_k
S_k	Set of operations on M_k
n_{ij}	Sequence number of O_{ij}
U_{ij}	Number of machines available for O_{ij}

3 Problem formulation

FJSSP can be represented as a set of N jobs to be scheduled on a set of M machines such that $J = [J_1, J_2, J_3, \ldots J_N]$, $M = [M_1, M_2, M_3, \ldots, M_M]$. Each job J_i consists of predefined operations O_{ij}. Total number of operations for job J_i can be denoted by J_{io}. Each operation O_{ij} can be performed on any of machines $M_k \in M$. P_{ijk} is the processing time of operation O_{ij} on machine M_k. In this scenario, the total number of operations of all jobs can be represented as $L = \sum$ Jio [12]. Now, a sequence number n_{ij} is assigned to every operation O_{ij} such that:

$$n_{ij} = \sum_{x=1}^{i-1} J_{xo} + j \quad (1)$$

Other constraints of FJSSP are as follows:

i. Each resource is available at the start of the problem search.
ii. Only a single operation can be performed on each machine on a given time.
iii. Operations are executed in a predefined sequence.
iv. There is no interruption once an operation is started.

4 Proposed algorithm

An Improved Genetic Algorithm integrated with Rules (IGAR) is proposed to solve the assignment part and scheduling is done using priority rules. Algorithm takes problem as input; solves it using genetic algorithm and priority rules and displays Gantt chart with optimized makespan as output. Fig. 1 shows the flow chart of complete algorithm.

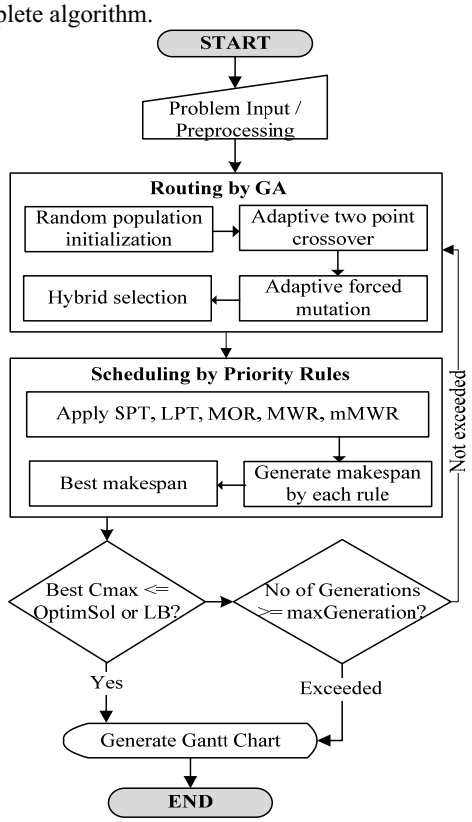

Fig. 1. Improved Genetic Algorithm integrated with Rules (IGAR)

4.1 Routing by Genetic Algorithm

GA has been used for routing of jobs on available machines. Following two improvements have been incorporated in GA part of the algorithm.

i. Proposed GA uses an adaptive technique to calculate crossover and mutation probabilities based on complete population fitness.
ii. An improved forced mutation technique has been proposed that positively ensures the generation of new chromosome during mutation process.

Chromosome representation proposed by Zhang [13] has been used in which every gene contains information of machine assignment for a particular process. Two-Point Crossover (TPX) technique has been used whereby two random chromosomes are selected from previous population as parents. Crossover points are randomly generated in range of [1, L]. Conventionally, crossover probability is kept high to explore more and more search space. Here, it is an adaptive parameter that depends upon the convergence of population to any local optima. As population converges, crossover probability starts increasing to maintain the diversity of algorithm around

the local optima by using the relation $(\overline{Ft}/\max Ft)$; where Ft is the fitness of the population.

A new forced mutation technique has been proposed based upon the concept of flexible genes that ensures the diversity of population in search space in a better way. A flexible gene can be defined as the gene that can have more than one possible value i.e. $U_{ij} > 1$. Flexibility index (*f*) can be defined as the average number of machines per operation as shown below.

$$f = \frac{\sum_{i=1}^{N}\sum_{j=1}^{J_{io}} U_{ij}}{L} \quad (2)$$

Considering the problem flexibility level, following two rules are proposed to ensure the forced mutation.

i. Mutation shall be performed on any random flexible gene instead of any random gene.
ii. While changing the value of gene, selected for mutation; new value shall be selected from remaining available options for that gene instead of all possible options.

The probability of mutation is also adaptive as of crossover. When GA starts converging to local minima; values of minimum fitness and maximum fitness become closer and their ratio increases. Thus, mutation probability increases to maintain the diversity of population in search space. High value of mutation probability may generate too much instability in the convergence of GA, but stability of convergence has been managed by improving the selection process to 40% elitism and 60% roulette wheel. Table 2 presents the pseudocode for forced mutation algorithm.

Table 2. Forced Mutation Algorithm

function Forced-Mutation ($chrom$,U_{ij}) returns $mutChrom$
$N \leftarrow$ count $U_{ij} > 1$ i.e. no. of flexible genes
$R \leftarrow$ random natural number from $1 \sim N$
$n \leftarrow Ind(R)$ i.e. index of desired flexible gene in $chrom$
$g_{ij} \leftarrow chrom(n)$ i.e. value of desired gene
$options \leftarrow 1 \sim U_{ij}(n)$ i.e. available options at n^{th} gene
$options \leftarrow$ Remove-Value($options$,g_{ij}), remove already availed option from $options$
$R \leftarrow$ a random natural number from $1 \sim length(options)$
$chrom(n) \leftarrow options(R)$, random selection from remaining options
$mutChrom \leftarrow chrom$
Return $mutChrom$

4.2 Scheduling by Priority Rules

Fitness function generates schedules using following priority rules.

i. Shortest Processing Time (SPT)
ii. Longest Processing Time (LPT)
iii. Most Operations Remaining (MOR)
iv. Most Work Remaining (MWR)
v. Modified Most Work Remaining (mMWR)

SPT and LPT use P_{ijk} for deciding the priority of any job. MOR schedules the job with most operations remaining on priority. Remaining operations (OR_{ij}) after any operation O_{ij} of job J_i can be calculated as $J_{io} - j + 1$. In MWR, job with most work remaining is scheduled on priority. Remaining work for any operation O_{ij} of job J_i can be calculated as $\sum_{x=j}^{J_{io}} P_{ixk'}$. Here, k' is the machine index assigned to O_{ix} in given chromosome.

In mMWR, it has been proposed to use average of all processing times of an operation; available for different machines, thus making it independent of machine assignment. Moreover, MWR is inclined to a schedule based on predefined assignment; whereas, mMWR defined priority of a process or job independently.

$$avgP_{ij} = \frac{1}{U_{ij}}\sum_{x=1}^{U_{ij}} P_{ij(\Omega_{ij}(x))} \quad (3)$$

$$mWR_{ij} = \sum_{x=j}^{J_{io}} avgP_{ix} \quad (4)$$

Here mWR_{ij} is the modified work remaining for operation O_{ij} of job J_i. Table 3 presents the pseudocode for scheduling.

Table 3. Scheduling Algorithm

function Get-Schedule($chrom$,$Problem$) returns $schedule$
$buffer \leftarrow$ Initialize an empty array of length M. Operations to be scheduled on M_k
$C_k \leftarrow$ Initialize with zeros, an array of length M. Time of availability of k^{th} machine.
for each O_{i1} do
$n \leftarrow \sum_{x=1}^{i-1} J_{xo} + 1$, operation sequence number
$g_n \leftarrow chrom(n)$, gene value at n^{th} index
$k \leftarrow \Omega_{ig_n}, g_n^{th}$ machine id out of machines for O_{ij}
$buffer(k) \leftarrow O_{i1}$
while !empty($buffer$)do
$cbuffer \leftarrow buffer$, take a copy of $buffer$
for each non-empty $cbuffer$ index $k$do
$List \leftarrow cbuffer(k)$, list of operations on M_k
$List' \leftarrow$ list of previous operations of $List$
$k' \leftarrow$ machine assignment for $List'$ in $chrom$
$C_{k'} \leftarrow$ availability time of $M_{k'}$
$O_{ij} \leftarrow$ operations from $List$ for which $C_{k'} < C_k$
if $empty(O_{ij})$ do
$O_{ij} \leftarrow$ Apply-Priority-Rule($List$)
if O_{ij} is 1st operation of J_i then do
$t_{ijk} \leftarrow C_k$
else do
$t_{ijk} \leftarrow \max(C_k, E_{ij'k'})$
$E_{ijk} \leftarrow t_{ijk} + P_{ijk}$
$C_k \leftarrow E_{ijk}$
$List \leftarrow$ remove O_{ij} form $List$
$buffer(k) \leftarrow List$
If!equal(j, J_{io})for O_{ij} do
$n \leftarrow \sum_{x=1}^{i-1} J_{xo} + j + 1$, operation sequence number
$g_n \leftarrow chrom(n)$, gene value at n^{th} index
$k \leftarrow \Omega_{ig_n}, g_n^{th}$ machine id out of machines for O_{ij}
$buffer(k) \leftarrow O_{i(j+1)}$
$cbuffer \leftarrow buffer$, take a copy of $buffer$
$schedule \leftarrow [t_{ijk} \quad E_{ijk}]$
return $schedule$

Makespan is calculated from all these five schedules. The algorithm terminates if any of the following conditions is fulfilled.

i. Best C_{max} among the population is less than or equal to optimum solution (optimum solutions of benchmark problems according to Behnke [14] have been used as termination condition).
ii. Maximum number of generations is achieved.
iii. Best Cmax among the population is less than or equal to lower bound (LB) of the problem, calculated using equation (5) to (7).

$$P'_{ij} = min(P_{ijk}) \quad \forall M_k \epsilon \Omega_{ij} \tag{5}$$

$$P'_i = \sum_{x=1}^{J_{io}} P'_{ix} \quad \forall O_{ij} \tag{6}$$

$$LB = \max_{1 \le i \le N} P'_i \tag{7}$$

5 Results and discussion

The efficacy of the algorithm is tested against the benchmark data sets of Kacem and Fattahi. Table 4 shows close coherence of results of proposed algorithm with ILOG Constraint Programming Engine [14] for Kacem instances. A limited comparison is carried out here, as the results of ILOG are already reported as optimal minima. Percentage deviation is calculated for each comparison by using the following equation.

$$Dev(\%) = \frac{Reference - Achieved}{Reference} \times 100 \tag{}$$

Table 4. Results for Kacem Instances

Problem	IGAR	CP	
		C_{max}	Dev(%)
Kacem1	11	11	0
Kacem2	11	11	0
Kacem3	7	7	0
Kacem4	14	12	-16.7

The results of IGAR for Fattahi problem set are compared with results obtained by two search based algorithms [11], a selected GA approach [15], a Mixed-Integer Linear Programming (MILP) approach [16] and ILOG Constraint Programming Engine [14] as reported in Table 5. Fig. **2** presents the Gantt Chart of MFJS7. It is evident that IGAR produces superior results as compared to other algorithms and remains comparable with ILOG.

Table 5. Comparison of IGAR with other algorithms

Problem	IGAR	H/TS [11]	H/SA [11]	GA [15]	MILP [16]	CP [14]
SFJS1	66	66	66	66	66	66
SFJS2	107	107	107	107	107	107
SFJS3	221	221	221	221	221	221
SFJS4	355	355	355	355	355	355
SFJS5	119	119	119	119	119	119
SFJS6	320	320	320	320	320	320
SFJS7	397	397	397	397	397	397
SFJS8	253	253	256	253	253	253
SFJS9	210	210	210	210	210	210
SFJS10	516	516	516	516	516	516
MFJS1	468	469	469	468	468	468
MFJS2	448	482	468	448	446	446
MFJS3	468	533	538	466	466	466
MFJS4	554	634	618	554	564	554
MFJS5	514	625	625	514	514	514
MFJS6	634	717	730	634	635	634
MFJS7	881	964	947	881	935	931
MFJS8	884	970	922	891	905	884
MFJS9	1097	1105	1105	1094	1192	1070
MFJS10	1275	1404	1384	1286	1276	1208

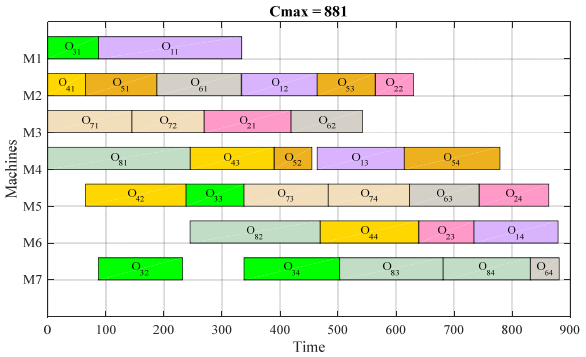

Fig. 2. Gantt Chart of MFJS7

Fig. **3** shows distribution of makespan for initial population and 20th generation to evaluate the convergence scheme of the algorithm. It is observed that population is randomly dispersed at the initial stage and then gradually some of the individuals start converging at best makespan due to elitism selection criteria. Remaining individuals follow Gaussian distribution due to roulette wheel selection criteria based on cumulative fitness of all chromosomes.

Fig. **4** shows the convergence pattern of minimum, maximum and average makespan of each generation for Kacem-4 problem up to 80 generations. It also shows the best rule that generated minimum makespan in each generation.

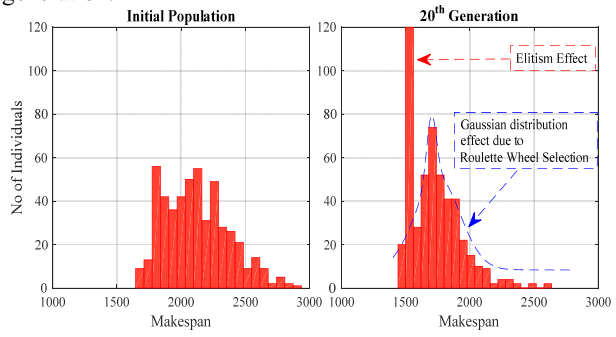

Fig. 3. Population Distribution

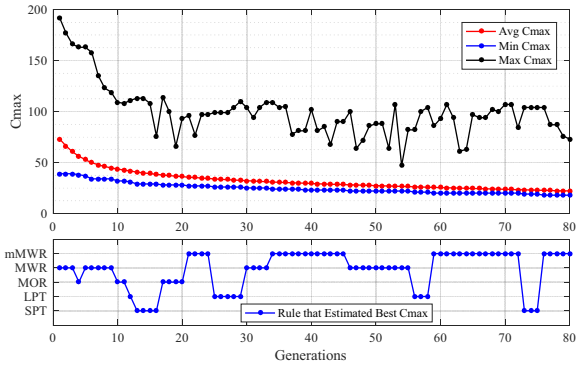

Fig. 4. Population Convergence for Instance-4 of Kacem

6 Conclusion

An improved genetic algorithm integrated with scheduling rules has been proposed in this study for optimization of makespan in the FJSSP environment. The GA uses an adaptive methodology for crossover and mutation probabilities to ensure maximum search space evaluation and restrain premature convergence. An improved mutation technique is also proposed to ensure generation of new chromosomes. Simulation studies are conducted on benchmark data of and results are compared. A positive mean percentage improvement has been obtained for IGAR as compared to other algorithms which confirm the adequacy of the algorithm. Future research may address the inclusion of other rules in the algorithm and implementation of the algorithm on industrial setups.

References

1. I.A. Chaudhry and A.A. Khan, International Transactions in Operational Research, *A research survey: review of flexible job shop scheduling techniques.* 41, (2015).
2. M. Chen and J.-L. Li. *Genetic Algorithm Combined with Gradient Information for Flexible Job-shop Scheduling Problem with Different Varieties and Small Batches*. in *MATEC Web of Conferences*. 2017. EDP Sciences.
3. B. Çaliş and S. Bulkan, Journal of Intelligent Manufacturing, *A research survey: Review of AI solution strategies of job shop scheduling problem.* **26**, 961-973, (2015).
4. M.K. Amjad, S.I. Butt, R. Kousar, R. Ahmad, M.H. Agha, Z. Faping, N. Anjum, and U. Asgher, Mathematical Problems in Engineering, *Recent Research Trends in Genetic Algorithm Based Flexible Job Shop Scheduling Problems.* **2018**, 32, (2018).
5. W.L. Qiao, Qiaoyun, *Solving the Flexible Job Shop Scheduling Problems Based on the Adaptive Genetic Algorithm*, in *International Forum on Computer Science-Technology and Applications IFCSTA '09)*, Z. Qihai, Editor. 2009. p. 97-100.
6. J.J.J. Yang, L. Y.; Liu, B. Y., *The improved genetic algorithm for multi-objective flexible job shop scheduling problem*, in *Applied Mechanics and Materials*. 2011. p. 870-875.
7. Y.Z. Pan, W. X.; Gao, T. Y.; Ma, Q. Y.; Xue, D. J., *An adaptive Genetic Algorithm for the Flexible Job-shop Scheduling Problem*, in *IEEE International Conference on Computer Science and Automation Engineering (CSAE)*. 2011. p. 405-409.
8. P. Kaweegitbundit and T. Eguchi, Journal of Advanced Mechanical Design, Systems, and Manufacturing, *Flexible job shop scheduling using genetic algorithm and heuristic rules.* **10**, (2016).
9. H.-H. Doh, J.-M. Yu, J.-S. Kim, D.-H. Lee, and S.-H. Nam, International Journal of Production Research, *A priority scheduling approach for flexible job shops with multiple process plans.* **51**, 3748-3764, (2013).
10. I.H. Kacem, Slim; Borne, Pierre, Mathematics and Computers in Simulation, *Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic.* **60**, 245-276, (2002).
11. P. Fattahi, M. Saidi-Mehrabad, and F. Jolai, Journal of Intelligent Manufacturing, *Mathematical modeling and heuristic approaches to flexible job shop scheduling problems.* **18**, 331-342, (2007).
12. M.K. Amjad, S.I. Butt, N. Anjum, I.A. Chaudhry, Z. Faping, and M. Khan, Advances in Production Engineering & Management, *A layered genetic algorithm with iterative diversification for optimization of flexible job shop scheduling problems.* **15**, 377-389, (2020).
13. G.G. Zhang, Liang; Shi, Yang, Expert Systems with Applications, *An effective genetic algorithm for the flexible job-shop scheduling problem.* **38**, 3563-3573, (2011).
14. D. Behnke and M.J. Geiger, *Test Instances for the Flexible Job Shop Scheduling Problem with Work Centers*. 2012, Universitätsbibliothek der Helmut-Schmidt-Universität: Hamburg.
15. M. Zandieh, I. Mahdavi, and A. Bagheri, Journal of Applied Sciences, *Solving the Flexible Job-Shop Scheduling Problem by a Genetic Algorithm.* **8**, 4650-4655, (2008).
16. C.Ö. Özgüven, Lale; Yavuz, Yasemin, Applied Mathematical Modelling, *Mathematical models for job-shop scheduling problems with routing and process plan flexibility.* **34**, 1539-1548, (2010).