Volume 64, Issue 1, 2020
Journal of Scientific Research
Institute of Science,
Banaras Hindu University, Varanasi, India.

Genetic Algorithms with Feasible Operators for Solving Job Shop Scheduling Problem

Yoginath R. Kalshetty[1], Amol C. Adamuthe[2*], and S. Phani Kumar[1]

[1]Department of CSE, GITAM University, Hyderabad. yrkalshetty@gmail.com, phanikumar.singamsetty@gitam.edu
[2]Dept. of CS & IT, RIT, Rajaramnagar, Sangli, MS, India. amol.admuthe@gmail.com

Abstract: **Job scheduling is one of the key activities performed in industries for manufacturing planning. In job scheduling, each job that contains various operations is allocated to one of the available machines for processing. Each job has a duration and each machine can handle only one operation at a time. An efficient allocation of jobs is mandatory for decreasing the makespan and idle time of the machines. In Job Shop Scheduling (JSS), the operations of the jobs are ordered. Genetic algorithm (GA) is a popular heuristic algorithm investigated to solve different scheduling problems. This paper presents feasibility preserving solution representation, initialization and operators for solving job shop scheduling problem. Proposed GA obtained best known results with good success rate for Lawrence (1984) datasets. Experiments show fast convergence of GA towards best solution. Hybridization of GA with local search or repair operator is required to obtain best solution with better success rate.**

Index Terms: **Genetic algorithm, generalized order crossover, job shop scheduling problem, scheduling problem.**

I. Introduction

Planning and scheduling are the two main activities performed in industries for manufacturing the products. The planning activity estimates the actions that have to be done and the restrictions on how to do it. The scheduling is the process of estimating the time and resources for each activity. Further, the scheduling process estimates the precedence relationship between the activities and the constraints. The complete execution of a plan demands temporal assignment of tasks and activities. Optimal scheduling provides the following advantages,

- Improved on-time delivery
- Decreased inventory
- Cut lead times
- Increased bottleneck resource utilization

But, as the scheduling problems are combinatorial, it is often difficult to estimate the optimal schedules. Further, the scheduling problem is considered as NP-complete because the schedule of ‘a’ number of jobs in ‘b’ number of machines demands optimal exploitation of resource and time. The traditional Job Shop Scheduling Problem (JSSP) considers the jobs as activities and machines as resources. According to the technique constraint of JSSP an operation of a job can be processed only after processing its precedent operations. The resource constraint of the JSSP states that each job should be processed on each machine only once and only one at a time. Further, the jobs are expected to be scheduled with minimal makespan and without any interruption (Chaudhary et al., 2013). To satisfy the constraints of the traditional JSSP, various scheduling techniques are used.

Single machine scheduling is defined as the process of assigning the group of tasks to a single machine for processing. The considerations of the single machine scheduling are as follows:

- The machine is always available during the scheduling period.
- The machine can process only one job at a time
- The processing time of the jobs on a machine is previously known.
- The information related to the jobs such as due date of the job, and release date of the job are known before.
- In case of the non-preemptive scheduling, the jobs complete their processing without any interruption. Whereas in the pre-emptive scheduling, the jobs are removed from the machine without finishing the operation.

In JSS, the routing information and processing time of the jobs are exploited for providing an efficient allocation of jobs to the machines. The assumptions of the job shop scheduling are as follows,

- Each job has a chain of operations.

DOI: http://dx.doi.org/10.37398/JSR.2020.640157

- Each machine has the ability to handle only one operation at a time.
- Every operation is expected to be processed without interruption.
- The main objective of the job shop scheduling is to allocate the operations to the machines that has minimal time interval.

In order to achieve an optimal allocation of jobs to the machines in JSS, the Swarm-based Optimization Algorithms (SOAs) such as GA, BCO, ACO, and PSO are used. When compared to the direct search algorithms, the SOA provides a population of solution for every iteration (Yuce et al., 2013).

The flow shop scheduling is a special case of JSS where only one operation in each job is deployed in every machine. When all the jobs pass between the machines in the same order it results in Flow Shop Scheduling Problem (FSSP). The FSSP can be categorized into two types such as static and dynamic. In the static FSS, the optimal sequence of the jobs on the machines is determined. Whereas, in the dynamic FSS, the jobs arrive continuously over time.

From the survey results, it is clear that the existing optimization techniques do not consider the parameters such as size of the task and delay time for the job completion. Further, the reduction of the makespan is not satisfactory. This paper presents Genetic algorithms for solving job shop scheduling problem. The objective of this paper is to minimize makespan of job shop scheduling problem.

This paper is organized as follows, Section II illustrates the background and related work for scheduling and job shop scheduling. Section III describes the job shop scheduling problem. Section IV is about Genetic algorithms, it's operators and parameters. Section V describes dataset and results. The paper is concluded in section VI.

II. Background and Related Work

Scheduling is the process of allocating the optimal resource for executing a task. Job scheduling can be classified into three types namely single machine scheduling, flow shop scheduling and job shop scheduling.

A. *Single machine scheduling*

In single machine scheduling, multiple jobs are assigned to a single machine for execution. The machine to which the jobs are allocated can be classified into two types such as,

- Dependent
- Independent

If the set-up time of the jobs is independent, then the problem is named as single machine scheduling problem with independent jobs or it is named as single machine scheduling problem with dependent jobs. The performance of the single machine scheduling problem is measured using the following metrics.

- Mean flow time
- Maximum lateness
- Total hardiness
- Number of tardy jobs

If the number of machines is more than one, the single machine scheduling is called as single machine scheduling with parallel machines. The parallel machine scheduling problem is classified into three types such as identical parallel machine scheduling problem, proportional or uniform parallel machines scheduling problem and unrelated parallel machine scheduling problem.

1) *Identical parallel machine scheduling problem*
In identical parallel machine scheduling problem, the machines that are parallel have identical speed. The jobs that are allocated to the parallel machines consume the same amount of processing time. A branch and bound algorithm is proposed in (Lee & Kim, 2015) for reducing the tardiness of the jobs in identical parallel machine scheduling problem. Once the specified numbers of jobs are processed, each machine demands a preventive maintenance task.

2) *Proportional parallel machine scheduling problem*
In this type of single machine scheduling, the parallel machines have different speeds. Among the available machines, the first machine is considered to be the slowest machine and the last machine is considered to be the fastest machine. The issues related to the scheduling of jobs that has similar due date and proportional early and tardy penalties of the identical parallel machines are analyzed (Sun & Wang, 2003). The analysis results show that the scheduling is a NP-hard problem. Further, the issues in the scheduling are addressed using dynamic programming problem.

3) *Unrelated parallel machine scheduling problem*
In this type of scheduling there will not be any relationship between the processing times of the jobs on the parallel machines. The difference in the technology can be due to the factors such as different machine and different job features. An iterated greedy algorithm is proposed for addressing the large-scale unrelated parallel machines scheduling problem (Abdelmaguid, 2015). The suggested algorithm by iterating over the constructive heuristic using destruction and construction phase provides a sequence of solutions. When compared to the traditional metaheuristic approach, the proposed approach provides optimal performance. A multi-objective PSO (MOPSO) optimization is proposed for estimating the optimal approximation of Pareto frontier (Torabi, 2013). By exploiting the selection regimes the personal and global best

solutions are obtained. When compared to the Conventional Multi-Objective Particle Swarm Optimization (CMOPSO) algorithm, the suggested MOPSO provides optimal quality, diversity and spacing.

B. *Flow shop scheduling*

In this type of scheduling, the jobs can be scheduled in various machines. Each job follows a process sequence. The process sequences of all the jobs are same. The performance of the flow shop scheduling is measured using the following metrics,

- Mean flow time
- Maximum lateness
- Total hardiness
- Number of tardy jobs
- Makespan

A Memetic algorithm named Opposition-based Differential Evolution (ODDE) is suggested for addressing the Permutation Flow Shop Problem (PFSSP) (Li & Yin, 2013). Initially, the ODDE is made suitable for the PFSSP using Largest-Ranked Value (LRV) rule. The LRV rule converts the continuous position of Direct Evolution (DE) into discrete job permutation. The Nawaz-Enscore-Ham (NEH) is combined with the random initialization to the population with certain quality and diversity. By exploiting the global optimization property of DE, the crossover rate is tuned. By deploying the opposition based learning for the initialization and generation jumping for the global optimum solution enhancement, the convergence rate of the DE is enhanced. The individuals with certain probability are enhanced using fast local search. The pairwise based local search is used for enhancing the global optimum solution. Further, it prevents the algorithm from local minimum. An Effective estimation of Distributed Algorithm (EDA) is suggested for addressing the Distributed Permutation Flow-shop Scheduling problem (DPFSP) (Wang, 2013). The optimal schedules are generated by deploying completion factory rule. The probability distribution of the solution space is illustrated using probability model.

C. *Job shop scheduling*

Job shop scheduling is an optimization problem that allocates suitable jobs to the machines for execution. In the job shop scheduling, the jobs are scheduled based on two factors such as routing of the jobs and processing time of the jobs. The scheduling issues of the flexible job shops are illustrated in (Sobeyko & Mönch, 2016). The Shifting Bottleneck Heuristic (SBH) is hybridized using local search approach and Variable Neighborhood Search (VNS) approach. The increase in the processing flexibility decreases the improvement of the advanced techniques.

An agent-based local search GA is used for efficiently handling the job shop scheduling problem. The suggested GA exploits a multi-agent system for deploying the local search genetic algorithm (Asadzadeh, 2015).

A novel hybrid island model is proposed for handling the job shop scheduling problem. The suggested model exploits a self-adaptation phase strategy for maintaining an optimal balance between diversification and intensification of the search process. The suggested self-adaptation phase strategy selects the optimal individuals based on the local search using tabu search (Kurdi, 2015).

III. Job Shop Scheduling Problem

The classical Job shop scheduling problem (JSSP) is one of the important and difficult problems in computer science and operations research and received an enormous amount of attention in the research literature.

The JSP problem is to determine the total completion time of set of operation/tasks on a set of machines. Following is the constraints that must be followed.

1. All jobs are available at time zero.
2. Each machine can process at most one operation at any time.
3. Each operation can be processed only at one machine at a time.
4. Operations of each job must be processed in a given order.
5. Processing time ti,j of each operation Oi,j is defined where ith operation of job j.
6. All the set of operation must be completed on set of machine.

The objective of the scheduling task is to optimize a certain criterion. These criterions are used as performance measure of the schedule.

Makespan: The makespan means the time needed to complete all the jobs and can be defined as $C_{max} = max_{1 \leq i \leq n}(C_i)$, where C_i is the completion time of job J_i.

IV. Genetic Algorithms

A. *Basic Genetic Algorithms*

Genetic algorithm is based on the Darwin's theory of evolution. According to Darwin's theory only the fittest individual survives in the next generation. By exploiting the information in solution population, new solutions with better performance are obtained.

Table I. Literature review for JSSP and FJSP

Reference	Performance	Quality measurement/dataset	Merits and demerits
Chong et al., 2007	A honey bees foraging model was suggested for addressing the job shop scheduling problems	• Makespan • Computation time • 82 job shop problems were considered for the experimental analysis	• As the proposed model modified the previous solution instead of constructing the new solution from scratch the makespan and computation time are optimal. • When compared to the probabilistic-based approaches the local optimums were avoided.
Wang, 2012	A hybrid genetic algorithm is proposed for enhancing the local search ability of GA.	• Benchmark problems were used for the experimental analysis	• The complete characteristics of the problem was exploited • The diversity of the population is increased using mixed selection operator • The local search ability of GA was greatly enhanced.
Gao et al., 2015	A Hybrid Island Model Genetic Algorithm (HIMGA) was proposed for addressing the job shop scheduling problem.	• Quality of the solution. • Effectiveness • 76 benchmark datasets with self-adaptation strategy is as the dataset.	• Achieved a balance between diversification and intensification of the search process. • Among the 76 benchmark datasets, the optimal solution was estimated for almost 71%. • The average relative deviation was from 0.3% to 0.75%.
Amirghasemi & Zamani, 2015	An effective asexual genetic algorithm was proposed for addressing the job shop scheduling problem	• Search space coverage • The dataset was extracted from ORLIB site of Brunel University.	• Solution with highest quality was chosen from the pool. • Replaced the lowest quality solution with modified solution. • Balanced the exploitation versus exploration. • The value of 10x10 instance was obtained as 0.06s. • For larger problems, the solution with precision of less than one percent was chosen as the optimal solution.
Gao et al., 2015	An optimal Two stage Artificial Bee Colony (TABC) was suggested for scheduling and rescheduling the new inserting jobs.	• Performance of scheduling stage • Performance of rescheduling stage • Fifteen benchmark instances that includes eight manufacturing instances were used for the experimental analysis.	• Minimized the makespan • Enhanced the TABC performance using ensemble local search • Produced optimal results for both scheduling and rescheduling stage.
Saidi-Mehrabad et al., 2015	An Ant Colony Algorithm (ACA) was proposed for composed of two components such as Conflict-Free Routing Problem (CFRP) and Job Shop Scheduling Problem (JSSP)	• Efficiency • Completion time • 13 test problems and sensitivity analysis were used for the experimental analysis.	• The objective function minimized the completion time. • Experimental analysis proved that the ACA was an effective meta-heuristic.
Zhao et al., 2018	A two-generation parent ant colony algorithm was suggested for generating a feasible scheduling solution	• The NSGA-II was compared with the proposed two-generation parent ant colony algorithm	• The father ant colony addressed the flexible processing route decision problem. • Produced optimal results than NSGA-II.

Zhang et al., 2009	A hybrid PSO algorithm was suggested for addressing the multi-objective Flexible Job-Shop scheduling problem (FJSP)	Efficiency in handling multi-objective FJSP	• Increased search accuracy • Efficiently addressed the multi-objective FJSP
Xing et al., 2010	A Knowledge-Based Ant Colony Optimization (KBACO) algorithm is suggested for performing the Flexible Job Shop Scheduling Problem (FJSSP). The ACO model was integrated with the knowledge model	• Quality of the schedules • Own benchmark instances were used for the performance evaluation	• By exploiting the ant colony optimization model, the knowledge information was obtained. • When compared to the traditional approaches the proposed algorithm increased the quality of the schedules.
Li et al., 2011	A hybrid parento-based discrete artificial bee colony algorithm was proposed for addressing the multi-objective flexible job shop scheduling problem.	Efficiency	• The available information was obtained using crossover operator. • The exploration and exploitation was balanced using local search approaches.
Nouiri et al., 2013	The PSO algorithm was proposed for addressing the FJSP.	• The partial FJSP and total FJSP were used as the benchmark data.	• Minimized the completion time • Efficiently solved the FJSP
Yuan & Xu, 2013	A novel memetic algorithm is proposed for addressing the Multi-Objective Flexible Job Shop Scheduling Problem (MO-FJSP)	• Makespan • Total workload • Critical workload	• Minimized the makespan • Reduced the workload • Minimized the critical workload
Chang et al. 2015	A Hybrid Taguchi-Genetic algorithm (HTGA) is suggested for addressing the flexible job shop scheduling problem with makespan optimization	• The Brandimarte MK1-MK10 benchmarks was used for the experimental analysis • Convergence speed	• Efficiently addressed the limitations of the Traditional Genetic Algorithm (TGA) • Prevented the unfeasible solutions that has increased convergence speed
Teekeng et al., 2016	An Evolutionary PSO (EPSO) algorithm was suggested for addressing the FJSP	• Particle life cycle with the following four features was considered as one of the feature of EPSO • Discrete position update mechanism is considered as another feature of EPSO. • 20 Benchmark instances are used for the experimental analysis	Reduced the makespan
Li & Gao, 2016	An efficient Hybrid Algorithm (HA) that integrates GA and Tabu Search (TS) was proposed for Flexible Job Shop scheduling Problem (FJSP)	• Computational time • Six famous benchmark instances including 201 open problems was used as the dataset	• Provides optimal balance between intensification and diversification • Integrates the advantages of both the evolutionary algorithm and LS method • Does not provide optimal result for all benchmark instances

Godberg (1989) described the steps in the GA. Initially, random number of chromosomes are collected for generating a population, then the fitness value of each chromosome in the population is computed. Among the existing chromosomes two chromosomes that have higher fitness value is selected. The selected chromosomes are then applied the crossover probability for generating new off springs. With the mutation probability new off springs are mutated at each locus then the mutated off springs are placed in the new population. The entire process is repeated till the termination criterion is met. If the end condition is satisfied the optimal solution from the current population is returned.

B. Genetic Algorithms for JSSP

This subsection presents the solution representation, genetic operators used for solving job shop scheduling problem.

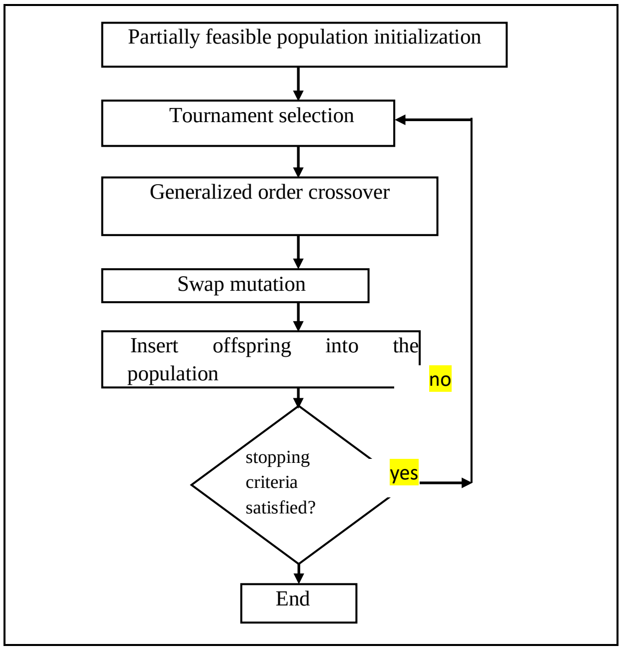

Fig. 1. Genetic algorithm flowchart

I. Solutions Representation

A 1D array is used for solution representation as shown in figure 2. The total number of operations of the JSSP problem defines the size of array. The chromosome is formed in such as way that it covers all the operation and there is no chance of formation of invalid chromosome. The integer values in the array indicate the job number. The repeated values of job number indicates the different operation of same job number. The operations number is measured from left to right direction in increasing order.

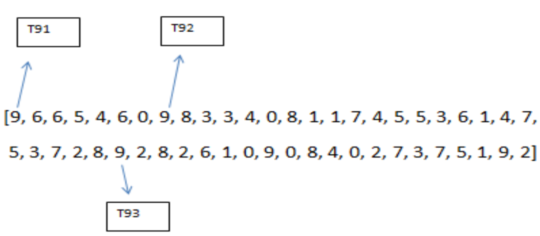

Fig. 2. Solutions representation

For example, in figure 2, the T91 number (9) indicates the first operation of Job number 9, and T92 denotes the second operation of job number 9 and so on. The second number is 6 indicates the first operation of job number 6 etc. The sequences of this number define the solutions of the JSSP problem. The solution which is minimum makespan is called an optimal solution.

II. Solution Initialization Process

Initial population is generated randomly considering the total tasks in each job. Randomness in chromosome is maintained using numpy.shuffle method from python. In JSSP, the number of machine generally indicates the number of operation of each job. It is important to ensure that all the operation of all the job are executed. The chromosome size implicitly validates number of operations to be performed. The design of chromosome take care that all the operation are considered and the further process of crossover or mutation does not invalidate the chromosome.

III. Genetic Operator

Table II presents selection, crossover, mutation operators and experimental setup of genetic algorithms experimented for job shop scheduling problem.

Table II. Experimental setup

Operator/Parameter	**Name/Value**
Population size	25
Elitism	5
Selection operator	Tournament
Tournament size	5
Crossover operator	GOX
Mutation operator	swap
Iterations	250
Crossover probability	0.8
Mutation probability	0.1

IV. Selection Operator

The selection strategy of chromosomes for the next generation is equally important to find the solutions of the problem. The tournament selection operator is used to select the chromosome.

V. Crossover Operator

GOX (generalized order crossover) (Bierwirth, 1995; Bierwirth et al., 1996) is used to produce the valid permutation while preserving the order of the operation within the parent chromosome. The order of genes and valid genes sequence of the chromosome is important as far as crossover is considered. In GOX, two parent chromosomes are divided in such as way that new offspring formed is valid and some sequences (fixed_list) is maintained in new offspring.

Example: Consider two chromosomes P1 and P2 as shown in figure 3. P1 and P2 represents the two chromosome used to produce the new offspring. P1 and P2 have 9 task of three jobs. The genes from chromosome represent the task of the Job and same gene number represents the next task of same Job. P1's first entry '1' indicates the first task of Job No. 1, next entry '2' indicates first task of Job No.2, third entry '2' indicates second task of Job No. 2 and so on. The index is used for references as index of the elements starting from 1.

The idea behind this is to use part of one chromosome (P1) as fixed and mix with another chromosome (P2) to form new offspring. The random number of i, j, k is shown in figure 3 as i=6, j=3 and k=3, The portion of chromosome P1 is fixed from location j to j+i elements. If j+i crosses the length of chromosome then remaining task are taken by considering chromosome as circular. i.e. task from starting of P1 are considered. In our example fixed_list of i=6 is formed as shown rectangle in figure 4 as fixed_list. The value of k is used as divider for forming left_list and righ_list as shown in P2 of figure 4. The left_list [3,1] and right_list=[2,3,3,2,1,2,1] is formed. The task which are in fixed_list is used as it is in final chromosome, so the task from left_list and right_list are removed which are in fixed_list, which is shown in figure 5. At last all the task from left_list is used followed by fixed_list and then remaining from the right_list. The new offspring formed is shown in figure 5.

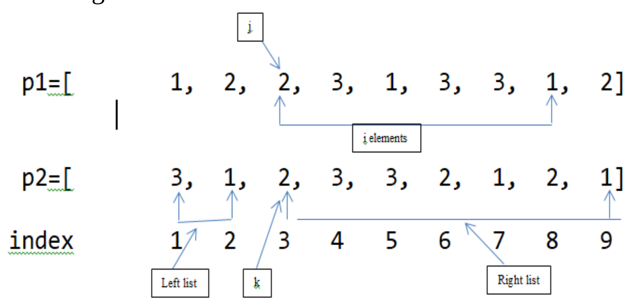

Fig. 3. GOX crossover

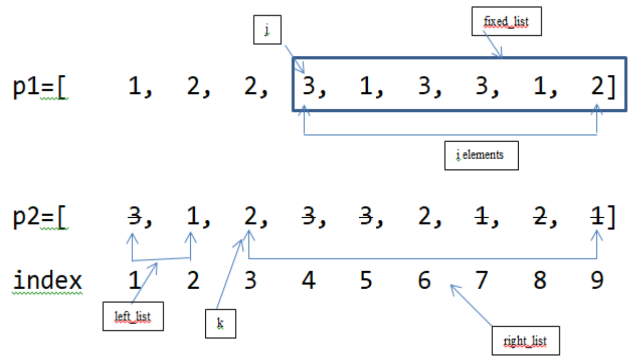

Fig. 4. Random i, j, k value of GOX chromosome

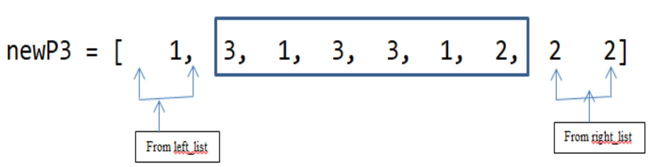

Fig. 5. Generation of new offspring

VI. *Mutation Operator*

The mutation operator probability is maintained in between 0.1 to 0.2 for better exploration of solutions space.

VII. *Objective Function*

The Job scheduling problem is treated as minimization problem. The objective is minimization of total unit time of the schedule.

V. Data Set and Experimental Results

This section gives a detailed explanation about the datasets used and the results obtained. The proposed algorithm is implemented using "Python" programming language and tested on a computer with the following specifications: Windows 7 Professional, Intel core i5 8250U CPU 2.5 @1.60 GHz 1.80 GHz and 8 GB RAM. For every dataset, Genetic algorithm was executed for 10 times.

A. *Dataset*

Dataset 1: Lawrence (1984) presented 40 problem instances of JSSP starting from la01 to la40 (Lawrence, 1984). Lawrence problem instances can be divided into 8 types depending on number of job operations and number of machines. The datasets, la01 to la05 datasets are with size 10x5, means 10 jobs and 5 machines. Datasets, la06 to la10 are 15x5, la11 to la15 are 20 x 5, la16 to la20 are 10x10 problem, la21 to la25 are 15x10, la26 to la30 are 20x10, la31 to la35 are 30x10 and la36 to la40 are 15x15.

Dataset 2: Applegate & Cook (1991) presented problem instances from orb01 to orb10. All the problem instances are with 10 jobs and 10 machines.

Dataset 3: Fisher (1963) defined ft06, ft10, and ft20 dataset instances with size 6x6, 10x10 and 20x5 respectively.

Dataset 4: Adams et al. (1988) have problem instances from abz6 to abz9 ranging from 10x10 to 20x15.

Below is the example of Lawrence dataset la01.

```
10 5
1 21 0 53 4 95 3 55 2 34
0 21 3 52 4 16 2 26 1 71
3 39 4 98 1 42 2 31 0 12
1 77 0 55 4 79 2 66 3 77
0 83 3 34 2 64 1 19 4 37
1 54 2 43 4 79 0 92 3 62
3 69 4 77 1 87 2 87 0 93
2 38 0 60 1 41 3 24 4 83
3 17 1 49 4 25 0 44 2 98
4 77 3 79 2 43 1 75 0 96
```

All the JSSP instances are given in standard formats which are combination of numbers. The location of number and its values give us the detail information of the problem. In the above example, the first line of JSSP problem instance have two integer numbers which denotes the number of job and number of machines respectively.

The second line onward gives information about the sequences of operations to be performed on which machine and time unit of the processing on each machine. The second line onward information is as follows.

1. Each line from second line onward indicates operations of single job.
2. The job number starts from 0 to N-1 jobs from second line to N+1 lines. In above example of 10 job X 5 machines, job number 0 on second line, job number 1 on third line, job number 2 on fourth line .. and job number 9 on 11th line.
3. Each line should be read from left to right to read the sequences of operation of any particular Job. Each line (except first line) should be read in pair to get complete information of single operation of particular Job. Consider line number four - 3 39 4 98 1 42 2 31 0 12 , which is job number 2 and should be read as (3, 39), (4, 98), (1, 42), (2,31),(0,12).

   This sequence of pair defines the sequences of operation of job no 2 on different machines and its time unit. The first number from the pair denotes the machine number (first machine starts with 0). In our example, (3,39) indicates that job number 2 have first operation to be processed on machine number 3 having processing time unit of 39, second operation (4,98) is on machine number 4 with processing time unit of 98, and so on.
4. Continue the above 3 step until the last time to read all the Job's operation sequences.

B. Results

The chromosome sample of Lawrence la01 problem is given as shown in figure 6.

[9, 6, 5, 6, 1, 1, 5, 4, 4, 7, 3, 7, 9, 6, 5, 0, 4, 1, 8, 0, 7, 0, 8,
7, 1, 3, 8, 9, 3, 5, 2, 1, 4, 2, 7, 8, 2, 8, 4, 9, 0, 6, 6, 5, 2, 3, 3, 2, 9, 0]

Fig. 6. Chromosome of instance la01 (Lawrence, 1984)

The job sequences indicates the Job's task sequences with triplet (machine_no, starting_time, end_time) for example the first list indicates the first job and entry is (1,131,152) indicates that the first task of job no 1 is executed on machine no 2 (as machine no starts with 0) at timing slot from 131 to 152 and so on.

Similarly, the machine sequences gives information of machines in triplet form (job_no, starting_time, end_time). The whole row indicates all the task the particular machine executes. i.e. The fifth row entry (9,0,77) indicates that the first task of job no 9 is executed on machine no 4 from time slot 0 to 77 time unit and so on.

Job sequence:

[[(1, 131, 152), (0, 164, 217), (4, 249, 344), (3, 345, 400), (2, 619, 653)],

[(0, 0, 21), (3, 69, 121), (4, 233, 249), (2, 249, 275), (1, 331, 402)],

[(3, 306, 345), (4, 448, 546), (1, 546, 588), (2, 588, 619), (0, 619, 631)],

[(1, 54, 131), (0, 217, 272), (4, 369, 448), (2, 511, 577), (3, 577, 654)],

[(0, 21, 104), (3, 121, 155), (2, 155, 219), (1, 402, 421), (4, 629, 666)],

[(1, 0, 54), (2, 54, 97), (4, 154, 233), (0, 272, 364), (3, 400, 462)],

[(3, 0, 69), (4, 77, 154), (1, 154, 241), (2, 318, 405), (0, 413, 506)],

[(2, 0, 38), (0, 104, 164), (1, 241, 282), (3, 282, 306), (4, 546, 629)],

[(3, 234, 251), (1, 282, 331), (4, 344, 369), (0, 369, 413), (2, 413, 511)],

[(4, 0, 77), (3, 155, 234), (2, 275, 318), (1, 421, 496), (0, 506, 602)]]

Machine sequence:

[[(1, 0, 21), (4, 21, 104), (7, 104, 164), (0, 164, 217), (3, 217, 272), (5, 272, 364), (8, 369, 413), (6, 413, 506), (9, 506, 602), (2, 619, 631)],

[(5, 0, 54), (3, 54, 131), (0, 131, 152), (6, 154, 241), (7, 241, 282), (8, 282, 331), (1, 331, 402), (4, 402, 421), (9, 421, 496), (2, 546, 588)],

[(7, 0, 38), (5, 54, 97), (4, 155, 219), (1, 249, 275), (9, 275, 318), (6, 318, 405), (8, 413, 511), (3, 511, 577), (2, 588, 619), (0, 619, 653)],

[(6, 0, 69), (1, 69, 121), (4, 121, 155), (9, 155, 234), (8, 234, 251), (7, 282, 306), (2, 306, 345), (0, 345, 400), (5, 400, 462), (3, 577, 654)],

[(9, 0, 77), (6, 77, 154), (5, 154, 233), (1, 233, 249), (0, 249, 344), (8, 344, 369), (3, 369, 448), (2, 448, 546), (7, 546, 629), (4, 629, 666)]]

Figure 7 represent the schedule of all the task of each job on machines. The horizontal line indicates the time unit of the execution of task whereas the vertical line indicates the machine numbers starting with 0. Each job is shown with difference color and number on the task indicates the Job number and length of the bar indicates the time period of task execution on that machine.

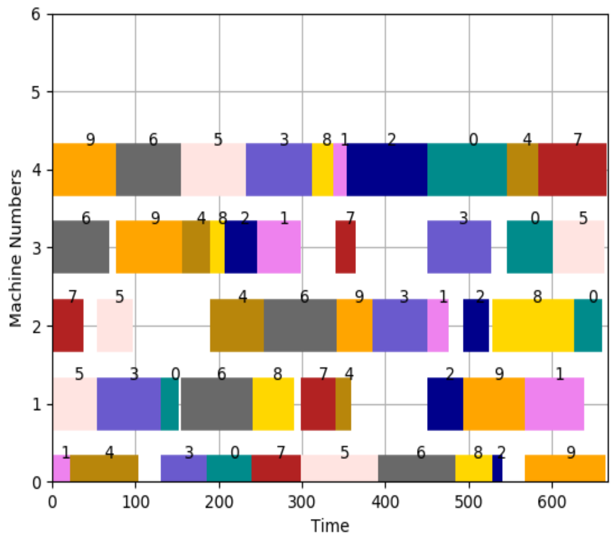

Fig. 7. Result of la01 schedule makespan = 666

Table III. GA result for dataset instances from (Lawrence, 1984)

Instance	Best Known	Best obtained	Difference	Deviation (%)	Success Rate (%)
la01	666	666	0	0	100
la02	655	663	8	1.22	-
la03	597	603	6	1.01	-
la04	590	604	14	2.37	-
la05	593	593	0	0	100
la06	926	926	0	0	100
la07	890	890	0	0	80
la08	836	863	27	3.23	-
la09	951	951	0	0	100
la10	958	958	0	0	100
la11	1222	1222	0	0	100
la12	1039	1039	0	0	100
la13	1150	1150	0	0	100
la14	1292	1292	0	0	100
la15	1207	1216	9	0.75	-
la16	945	973	28	2.96	-
la17	784	797	13	1.66	-
la18	848	869	21	2.48	-
la19	842	883	41	4.87	-
la20	902	918	16	1.77	-
la21	1053	1159	106	10.07	-
la22	927	1029	102	11	-
la23	1032	1091	59	5.72	-
la24	935	1032	97	10.37	-
la25	977	1060	83	8.5	-
la26	1218	1354	136	11.17	-
la27	1235	1402	167	13.52	-
la28	1216	1382	166	13.65	-
la29	1152	1343	191	16.58	-
la30	1355	1481	126	9.3	-
la31	1784	1842	58	3.25	-
la32	1850	1951	101	5.46	-
la33	1719	1789	70	4.07	-
la34	1721	1850	129	7.5	-
la35	1888	1962	74	3.92	-
la36	1268	1395	127	10.02	-
la37	1397	1580	183	13.1	-
la38	1196	1406	210	17.56	-
la39	1233	1382	149	12.08	-
la40	1222	1384	162	13.26	-

The result of Lawrence (1984) dataset is shown in Table III. The obtained results, best known results, deviation and success rate is presented. The result shows that for ten datasets, GA obtained optimal values.

Table IV. GA result for dataset instances from (Applegate & Cook, 1991)

Instance	Best Known solution	Best obtained	Difference	Deviation (%)
orb01	1059	1163	104	9.82
orb02	888	911	23	2.59
orb03	1005	1113	108	10.75
orb04	1005	1056	51	5.07
orb05	887	916	29	3.27
orb06	1010	1076	66	6.53
orb07	397	426	29	7.3
orb08	899	982	83	9.23
orb09	934	981	47	5.03
orb10	944	1035	91	9.64

Table V. GA result for dataset instances from (Fisher, 1963)

Instance	Best Known solution	Best obtained	Difference	Deviation (%)
ft06	55	55	0	0
ft10	930	1009	79	8.49
ft20	1165	1263	98	8.41

Table VI. GA result for dataset instances from (Adams et al., 1988)

Instance	Best Known Solution	Best Obtained	Difference	Deviation (%)
abz5	1234	1334	100	7.5
abz6	943	979	36	3.68
abz7	656	781	125	16.01
abz8	645	800	155	19.38
abz9	661	824	163	19.78

Table IV represents the results for Applegate & Cook (1991) 8 instances from 0rb1 to orb8. The values in deviation column shows the percentage of deviation is below 10 for all the instances and minimum deviation is 2.59%. Table V shows the result for Fisher (1963) instances where ft06 deviation is 0% and remaining are below 10%. Table VI shows the result for Adams et al., (1988) instances where the result of deviation is below 20% for three instance and below 10% for two instances.

Figure 8, 9 and 10 shows the convergence of proposed GA for datasets from Lawrence (1984), Fisher (1963) and Adams et al., (1988). Figure 8 show the chart of result of dataset instance of la01, la04 and la12 JSSP problems. The vertical line (Y-Axis) represents the makespan time unit value that needs to be minimized whereas the horizontal line(X- Axis) represents the number of iteration.

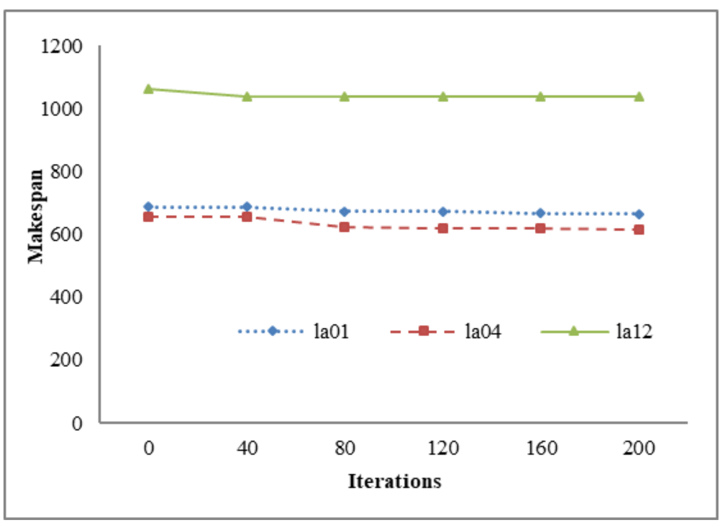

Fig. 8. Performance of GA, makespan verses iteration (Lawrence, 1984)

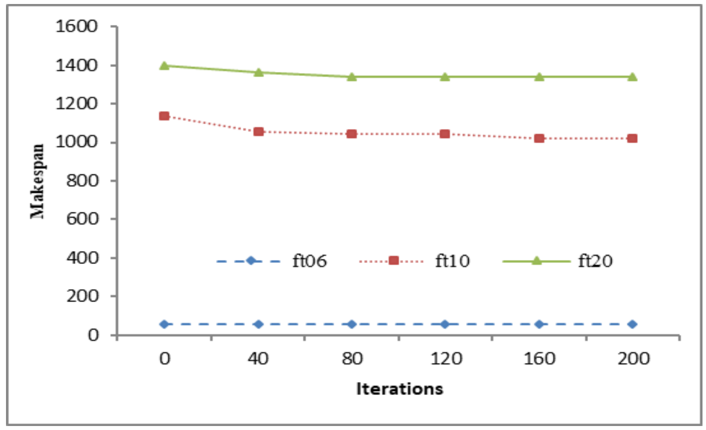

Fig. 9. Performance of GA, makespan verses iteration (Fisher, 1963)

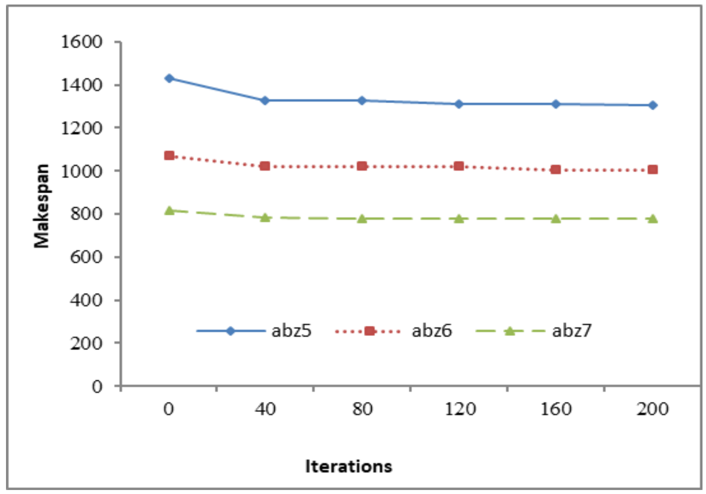

Fig. 10. Performance of GA, makespan verses iteration (Adams et al., 1988)

CONCLUSIONS

Job shop scheduling problem is NP problem. In literature different heuristic algorithms are investigated to solve different variations of job shop scheduling problem. From the survey on the various JSS optimization techniques it is observed that the existing techniques do not have the ability to handle variations in constraints and objectives.

This paper presents genetic algorithms for solving job shop scheduling problem. Proposed one dimensional solution representation and initialization process creates partially feasible solution. Used generalized order crossover and swap mutation maintains the feasibility in the solution. Performance of GA is tested on four benchmark datasets. Proposed GA with feasible representations and operators shows fast convergence towards best solution.

Future work: In many instances genetic algorithms stops close to the best known solution. There is scope to improve the performance with hybridization of local search algorithm.

References

Abdelmaguid, T. F. (2015). A neighborhood search function for flexible job shop scheduling with separable sequence-dependent setup times. *Applied Mathematics and Computation*, *260*, 188-203.

Adams, J., Balas, E., & Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management science*, *34*(3), 391-401.

Amirghasemi, M., & Zamani, R. (2015). An effective asexual genetic algorithm for solving the job shop scheduling problem. *Computers & Industrial Engineering*, *83*, 123-138.

Applegate, D., & Cook, W. (1991). A computational study of the job-shop scheduling problem. *ORSA Journal on computing*, *3*(2), 149-156.

Asadzadeh, L. (2015). A local search genetic algorithm for the job shop scheduling problem with intelligent agents. *Computers & Industrial Engineering*, *85*, 376-383.

Bierwirth, C. (1995). A generalized permutation approach to job shop scheduling with genetic algorithms. *Operations-Research-Spektrum*, *17*(2-3), 87-92.

Bierwirth, C., Mattfeld, D. C., & Kopfer, H. (1996, September). On permutation representations for scheduling problems. In *International Conference on Parallel Problem Solving from Nature* (pp. 310-318). Springer, Berlin, Heidelberg.

Chang, H. C., Chen, Y. P., Liu, T. K., & Chou, J. H. (2015). Solving the flexible job shop scheduling problem with makespan optimization by using a hybrid Taguchi-genetic algorithm. *IEEE access*, *3*, 1740-1754.

Chaudhary, D., Singh, D., Rawat, N., Agarwal, P., & Kumar, P. (2013). A Survey on Different Techniques used for Solving Job Shop Scheduling Problem. *International Journal of Emerging Trends in Electrical and Electronics*, 3, 64-67

Chong, C. S., Low, M. Y. H., Sivakumar, A. I., & Gay, K. L. (2007). Using a bee colony algorithm for neighborhood search in job shop scheduling problems. In *21st European conference on modeling and simulation (ECMS 2007)*.

Fisher, H. (1963). Probabilistic learning combinations of local job-shop scheduling rules. *Industrial scheduling*, 225-251.

Gao, K. Z., Suganthan, P. N., Chua, T. J., Chong, C. S., Cai, T. X., & Pan, Q. K. (2015). A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion. *Expert systems with applications*, *42*(21), 7652-7663.

Gao, L., Li, X., Wen, X., Lu, C., & Wen, F. (2015). A hybrid algorithm based on a new neighborhood structure evaluation method for job shop scheduling problem. *Computers & Industrial Engineering*, *88*, 417-429.

Godberg, D. E. (1989). Genetic algorithms in search. *Optimization, and Machine Learning*.

Kurdi, M. (2015). A new hybrid island model genetic algorithm for job shop scheduling problem. *Computers & Industrial Engineering*, *88*, 273-283.

Lawrence, S. (1984). Resouce constrained project scheduling: An experimental investigation of heuristic scheduling techniques (Supplement). *Graduate School of Industrial Administration, Carnegie-Mellon University*.

Lee, J. Y., & Kim, Y. D. (2015). A branch and bound algorithm to minimize total tardiness of jobs in a two identical-parallel-machine scheduling problem with a machine availability constraint. *Journal of the Operational Research Society*, *66*(9), 1542-1554.

Li, J. Q., Pan, Q. K., & Gao, K. Z. (2011). Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems. *The International Journal of Advanced Manufacturing Technology*, *55*(9-12), 1159-1169.

Li, X., & Gao, L. (2016). An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. *International Journal of Production Economics*, *174*, 93-110.

Li, X., & Yin, M. (2013). An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure. *Advances in Engineering Software*, *55*, 10-31.

Nouiri, M., Jemai, A., Ammari, A. C., Bekrar, A., & Niar, S. (2013, October). An effective particle swarm optimization algorithm for flexible job-shop scheduling problem. In *Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (IESM)* (pp. 1-6). IEEE.

Saidi-Mehrabad, M., Dehnavi-Arani, S., Evazabadian, F., & Mahmoodian, V. (2015). An Ant Colony Algorithm (ACA) for solving the new integrated model of job shop scheduling and conflict-free routing of AGVs. *Computers & Industrial Engineering*, *86*, 2-13.

Sobeyko, O., & Mönch, L. (2016). Heuristic approaches for scheduling jobs in large-scale flexible job shops. *Computers & Operations Research*, *68*, 97-109.

Sun, H., & Wang, G. (2003). Parallel machine earliness and tardiness scheduling with proportional weights. *Computers & Operations Research*, *30*(5), 801-808.

Teekeng, W., Thammano, A., Unkaw, P., & Kiatwuthiamorn, J. (2016). A new algorithm for flexible job-shop scheduling problem based on particle swarm optimization. *Artificial Life and Robotics*, *21*(1), 18-23.

Torabi, S. A., Sahebjamnia, N., Mansouri, S. A., & Bajestani, M. A. (2013). A particle swarm optimization for a fuzzy multi-objective unrelated parallel machines scheduling problem. *Applied Soft Computing*, *13*(12), 4750-4762.

Wang, S. Y., Wang, L., Liu, M., & Xu, Y. (2013). An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem. *International Journal of Production Economics*, *145*(1), 387-396.

Wang, Y. (2012). A new hybrid genetic algorithm for job shop scheduling problem. *Computers & Operations Research*, *39*(10), 2291-2299.

Xing, L. N., Chen, Y. W., Wang, P., Zhao, Q. S., & Xiong, J. (2010). A knowledge-based ant colony optimization for flexible job shop scheduling problems. *Applied Soft Computing*, *10*(3), 888-896.

Yuan, Y., & Xu, H. (2013). Multiobjective flexible job shop scheduling using memetic algorithms. *IEEE Transactions on Automation Science and Engineering*, *12*(1), 336-353.

Yuce, B., Packianather, M., Mastrocinque, E., Pham, D., & Lambiase, A. (2013). Honey bees inspired optimization method: the bees algorithm. *Insects*, *4*(4), 646-662.

Zhang, G., Shao, X., Li, P., & Gao, L. (2009). An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers & Industrial Engineering*, *56*(4), 1309-1318.

Zhao, B., Gao, J., Chen, K., & Guo, K. (2018). Two-generation Pareto ant colony algorithm for multi-objective job shop scheduling problem with alternative process plans and unrelated parallel machines. *Journal of Intelligent Manufacturing*, *29*(1), 93-108.

***