Makespan Optimization in Job Shop Scheduling Problem using Differential Genetic Algorithm

Arshdeep Kaur
Department of Computer Science & Engineering
BBSBEC
Fatehgarh Sahib, Punjab, India

Baljit Singh Khehra
Department of Computer Science & Engineering
BBSBEC
Fatehgarh Sahib, Punjab, India

Ishpreet Singh Virk
Department of Computer Science & Engineering
BBSBEC
Fatehgarh Sahib, Punjab, India

ABSTRACT
Job shop scheduling problem belongs to a class of NP-Hard problems. Hence, finding an optimal solution for this problem is a difficult task. In this study, a hybrid method consisting of Genetic Algorithm (GA) and Differential Evolution (DE) algorithm has been proposed for solving the Job Shop Scheduling problem (JSSP). These algorithms are evolutionary algorithms for solving optimization problems which refine the candidate solutions iteratively. The results of previous studies show that the application of genetic algorithm and differential evolution algorithm individually for this problem yield results close to the upper bounds. The proposed algorithm implemented in MATLAB R2013a uses minimization of makespan as the objective function. This algorithm has been tested on 50 instances of Taillard series (TA01-50) benchmark problem. The simulation results obtained by the proposed algorithm are better than those obtained by the IPSO-TSAB algorithm.

General Terms
Production scheduling, Evolutionary Algorithm.

Keywords
Combinatorial optimization; Job Shop Scheduling; Genetic Algorithm (GA); Differential Evolution (DE); Makespan.

1. INTRODUCTION
Scheduling problems have gained significant attention in recent years, due to their increasing demand in industrial applications particularly manufacturing. Scheduling can be considered as an optimization problem which defines the manner in which various jobs are ordered in accordance with the available resources. An optimization problem involves maximizing or minimizing some function, relative to an available data set, often representing a range of choices available in a certain situation. These problems may be either continuous or discrete, depending upon the type of variables. The problems having continuous and discrete variables are classified as constrained and combinatorial optimization problems, respectively. In combinatorial optimization problem, an optimal solution is obtained from a discrete set of feasible solutions [1]. Shop scheduling is an interesting area of research in the field of scheduling, and is considered to be a combinatorial optimization problem.

In shop scheduling problems, there is a set of n jobs and each job consists of a certain number of operations which are to be scheduled on a given set of machines. There may be certain dispatching rules for scheduling jobs, in order to achieve the desired objective [2]. Depending upon certain conditions for execution of jobs, there are three basic variants of shop scheduling problems: flow shop, job shop and an open shop problem [3]. In a flow shop problem, there is a strict order of operations for all jobs [4]. In a job shop problem, there are some precedence constraints for each job, according to which the jobs are completed [5]. In an open shop problem, the operations of all the jobs may be executed in any order [6]. This study is focused on Job Shop Scheduling Problem (JSSP), under the category of NP-Hard problems, in which the solution space increases exponentially as the number of jobs increase, and thus, the number of operations also increase [1]. Metaheuristic algorithms have proved to be beneficial in solving such problems by exploring large solution spaces.

Many metaheuristic algorithms have been used for solving the JSSP including Simulated Annealing [7], Tabu Search [8], Genetic Algorithm [9], Particle Swarm Optimization [10], Ant Colony Optimization [11], Bacteria Foraging Optimization [12], and Differential Evolution [13]. Gao *et al.* [14] employed a hybrid Particle Swarm Optimization algorithm for JSSP followed by Tabu Search, to further refine the solutions. Real-Integer encoding and decoding scheme was used to exchange the solutions between these algorithms. This IPSO-TSAB algorithm was tested on benchmarks of Lawrence (LA36-40) and Taillard (TA01-50) series and it resulted in optimal makespan as compared to Tabu Search, Improved Tabu Search and Hybrid PSO algorithm.

Chang and co-workers [15] developed a hybrid Genetic Algorithm (GA) and embedded the Taguchi method behind mating to increase the effectiveness of GA. The experiment on Brandimarte (MK01-MK10) benchmarks led to feasible solutions as compared to Effective & Distributed PSO, Effective GA, and hybrid of PSO & Tabu Search. Goncalves *et al.* [16] presented a GA for JSSP which involved random key representation, schedule construction using priority rules and embedding local search to refine the solution. The algorithm was tested on some Fisher & Thompson (1963) and Lawrence (1984) instances. The near-to-optimal solutions with average relative deviation of 0.39% from the best known solution were obtained. Wang and Tang [17] applied an improved adaptive Genetic Algorithm (IAGA) for solving JSSP using adaptive crossover and mutation probability, and a new operator was devised for crossover. The computational results indicated that IAGA is robust and resulted in better solutions than other algorithms. Qing-dao-er-ji and Wang [18] designed a hybrid Genetic Algorithm (HGA) for JSSP, consisting of a mixed selection operator based on fitness and concentration value, new crossover operator based on machine, mutation operator based on critical path, and used local search at the end. HGA was tested on Fisher & Thompson and Lawrence benchmark instances and yielded better results as compared to other algorithms.

Zhang *et al.* [19] proposed a Chaotic Differential Evolution Algorithm (CDEA) for flexible JSSP and used logistic mapping for initialization, machine based encoding, double mutation scheme and an elitist strategy in selection operation. It helped in minimizing the makespan by reducing relative error. Ponsich and co-workers [20] hybridized Differential Evolution (DE) with Tabu Search to solve JSSP. The experiments on more than 100 instances proved that the results obtained by the proposed hybrid algorithm are comparable with those of other algorithms. Yuan *et al.* [21] presented a DE algorithm for JSSP in which local search was embedded to improve the exploration and exploitation ability. Various computations showed that proposed algorithm provided optimal results in comparison to standard GA and DE. Vesterstrom and Thomson [22] evaluated the applicability of Differential Evolution (DE), Particle Swarm Optimization (PSO) and Evolutionary Algorithms (EAs) as numerical optimization techniques. The experiment was performed on 34 benchmark problems for set number of evaluations & random seeds. It showed that DE outperformed EA & PSO, except on two noisy functions, in which EA outperformed DE & PSO.

Thus, it can be concluded from the above discussion that Genetic Algorithm (GA) and Differential Evolution (DE) algorithm can yield optimal results when used for JSSP. Both GA and DE are metaheuristic algorithms. Genetic Algorithm can efficiently find global minimum in minimization problems [23]. Traditional GA has powerful exploration ability, but gets trapped in the local optima. Hence, GA suffers from premature convergence [17, 24]. On the other hand, DE has certain parameters (e.g. scaling factor F and crossover ratio Cr) to control the exploration-exploitation balance. Also, DE can find global minimum for benchmark problems with smaller function calls as compared to PSO and it is more robust [23]. So, the combination of both these algorithms may help in finding optimal makespan. The proposed hybrid algorithm is tested on 50 benchmark problem instances of Taillard series from 15×15 to 30×20 [25].

2. JOB SHOP SCHEDULING PROBLEM (JSSP)

A standard Job Shop Scheduling Problem (JSSP) may be defined as: There are 'n' different jobs required to be processed on 'm' machines, which execute one of the 'm' operations per job [14, 26]. The number of operations that each job consists of, is equal to the number of machines. The basic assumptions of JSSP include [24, 27]:

1. Each job consists of a sequence of operations.
2. All operations must go to each machine only once.
3. Each machine handles only one operation at a time.
4. An operation cannot be interrupted until it is complete i.e. non-preemption.
5. There are precedence constraints among the operations of same job, but not among different jobs.
6. There are no time delays.

The main aim of JSSP is to find a schedule having minimum length [28]. A schedule may be defined as the mapping of jobs to machines, and the processing times of operations to minimize the makespan i.e. the completion time of the last job. It can be denoted by $C_{max}(\pi)$ of the last job in an order $\pi = \{j_1, j_2, \ldots, j_n\}$. The makespan is said to be optimal if it is close to the upper bound i.e. the upper limit of the makespan or the minimum makespan [25].

Consider an example of a JSSP. Suppose there are 3 jobs and each job consists of 3 operations, which are to be executed on 2 machines. A JSSP consists of the dimension n×m where n is the number of jobs and m is the number of machines. Here, the given problem is a 3×2 problem. The processing times of operations and the order of execution of operations is given below:

Table 1. Example of 3×2 JSSP

Jobs	Processing Times	Execution Sequence
Job 1	O_{11}=3; O_{12}=3; O_{13}=3	O_{11}-O_{12}-O_{13}
Job 2	O_{21}=2; O_{22}=4; O_{23}=3	O_{21}-O_{22}-O_{23}
Job 3	O_{31}=2; O_{32}=3; O_{33}=1	O_{31}-O_{32}-O_{33}

There are many schedules available for the above example. One of the feasible schedules is shown in Figure 1 in the form of a Gantt chart. The makespan in this case is 13. The various tasks are denoted by the operation numbers. O_{11} implies the first operation of job 1, O_{12} implies second operation of job 2 and so on.

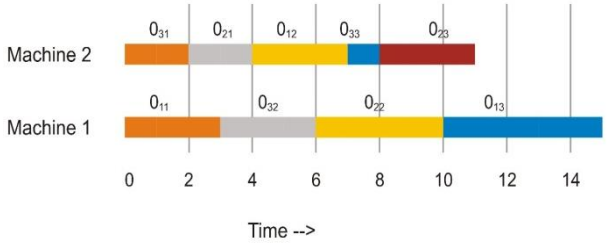

Figure 1: Gantt chart for 3×2 JSSP

3. DIFFERENTIAL GENETIC ALGORITHM (DGA)

3.1 Genetic Algorithm (GA)

A Genetic Algorithm is a population based search mechanism involving the phenomenon of natural genetics [24]. It follows Darwin's Theory of survival of the fittest which states that: Only fittest individuals survive and reproduce. In this, chromosomes are used for initialization of generations and fitness function is evaluated. The fitness function is the objective function which is to be maximized or minimized, depending upon the problem.

Basic steps of Genetic Algorithm are as below:

i. Encoding: It is used to describe the chromosome. The representation may be direct or indirect [1, 26]. In direct representation, the chromosomes represent the schedules for scheduling problem whereas in indirect representation, the chromosome does not directly represent a schedule. Hence, a decoding mechanism is used to convert the chromosomes into the respective schedules [29].

ii. Selection: It selects the chromosomes for crossover on the basis of fitness function. There are various schemes for performing selection in GA i.e. Roulette wheel selection, Rank based selection, Tournament selection, Boltzmann selection, Stochastic Universal Sampling and so on [30, 31].

iii. Crossover: The two selected chromosomes exchange their genes with one another on the basis of some random number or crossover rate. New chromosomes contain features from both the parents. Some variants of crossover operation are PMX crossover, order crossover, cyclic crossover and so on [32].

iv. Mutation: After the new chromosomes have been generated, mutation is induced to preserve the genetic diversity of the population. Various mutation mechanisms have been proposed i.e. random mutation, flipping, and swap mutation etc. [33, 34].

3.2 Differential Evolution (DE)

Differential Evolution (DE) is a population based stochastic optimization algorithm [35] which uses real values for performing various operations. The operations in DE are same as GA, but their order is reverse. The mutation is induced followed by crossover, and the selection is performed at the end. DE works with old and new generation individuals and the fittest ones are taken to the next generation.

The Differential Evolution algorithm proceeds as follows:

i. Initialization: The parameters used i.e. mutation factor (F), crossover factor (CR), population size, number of generations etc. are initialized.

ii. Mutation: Each individual is set as the target vector, one by one and mutation is performed using mutation factor (F) and some other individuals. There are certain variants for performing mutation [36]. The individual generated after mutation is called the mutant vector or donor vector.

iii. Crossover: It is performed between the target vector and the mutant or donor vector, to produce the trial vector. Crossover rate (CR) is used to perform exponential or binomial crossover operation [37].

iv. Selection: The target and trial vectors are compared according to their fitness values and the individual having better value of fitness function is placed in the next generation.

3.3 Proposed Hybrid Algorithm

The DE algorithm consists of various phases in the sequence: mutation, crossover and selection whereas GA consists of these phases in reverse sequence. In the proposed approach, Differential Evolution (DE) algorithm is embedded in the Genetic Algorithm (GA). The phases proceed as in GA, but the operators used are the same as in DE. The parameters are set to yield good results. GA suffers from premature convergence but the scaling factor less than 0.6 results in slower convergence, thus improving the results [38]. Similarly, the value of crossover factor (Cr) is set accordingly.

Pseudo code: **Proposed-DGA:**

Step 1: Begin

Step 2: Initialize the parameters: Mutation Factor, F= 0.5; Crossover Factor, Cr=0.4; Population Size= 30; Maximum Iterations= 100

Step 3: Randomly generate two initial populations of chromosomes of size 30 lying b/w 0 & 1

Step 4: while (the termination criterion is not satisfied)

Step 5: do

Step 6: for (i=1 to Population Size)

Use real-integer encoding approach to transform the real solutions into integer solutions, to calculate their fitness function i.e. makespan using Eq. 1.

Step 7: Perform Selection: The population with minimum makespan is selected for crossover.

Step 8: Perform Crossover: Binomial crossover is applied on initial population without encoding.

Step 9: Mutation: rand/1/bin approach as in Eq. 2 is used for mutation.

Step 10: Perform Greedy selection to choose the best schedule between the solutions, obtained after mutation and crossover on the basis of makespan (C_{max}) value.

Step 11: Place the selected individual in the next generation.

Step 12: end for

Step 13: end do

Step 14: End

The various operators used in above algorithm are described below:

i. Initial population: Two initial populations of size 30 with values lying between 0 & 1 are generated randomly. For example, population with size 4 and 6 genes can be represented as:

Individual 1:	0.68	0.89	0.04	0.94	0.52	0.17
Individual 2:	0.92	0.22	0.33	0.61	0.58	0.06
Individual 3:	0.22	0.14	0.40	0.02	0.04	0.07
Individual 4:	0.12	0.09	0.05	0.91	0.57	0.32

ii. Encoding: The real-integer encoding approach [14] is used to convert the real values into integers. It can be explained as: Consider the first chromosome with genes 0.68, 0.89, 0.04, 0.94, 0.52 and 0.17. These are first sorted in ascending order according to their position in the chromosome: C= (3, 6, 5, 1, 2, 4). Now, apply the formula: C1= ceil(C/no. of machines). Suppose number of machines is 2. Then, C1= (2, 3, 3, 1, 1, 2). The elements in C1 depict the serial numbers of the machines.

iii. Fitness Function: The fitness function used here is makespan. It refers to total completion time of jobs, which is to be minimized. It is represented by C_{max} and the basic model with makespan objective is described as follows [14]:

$$\pi^* = arg\{C_{max}(\pi)\} \rightarrow \min \forall \pi \in \prod \qquad \text{(Eq. 1)}$$

iv. Crossover: The crossover method used in the proposed algorithm is Binomial crossover, same as in Differential Evolution (DE) algorithm. In this, components are taken from the two selected individuals. One is taken with probability Cr and the other with probability 1-Cr. The value of Cr is taken to be 0.4 in this case.

A random value less than 1 is generated. If random value<= Cr, then that gene is added from second individual, otherwise from first individual [37]. For example, suppose Cr= 0.4 and random value is generated every time for each gene, then a trial individual is obtained as:

Table 2. Generation of trial individual

Individual 1	Individual 2	Random value	Trial Individual
0.68	0.92	0.8	0.68
0.89	0.22	0.2	0.22
0.04	0.33	0.3	0.33
0.94	0.61	0.6	0.94
0.52	0.58	0.4	0.58
0.17	0.06	0.7	0.17

v. Mutation: The random mutations are done first. Then, rand/1/bin approach of Differential Evolution (DE) is used. In this approach, the weighted difference between two randomly chosen individuals is added to a third randomly chosen individual according to the equation [36]:

$$X_{new} = x_{r1} + F\,(x_{r2} - x_{r3}) \qquad \text{(Eq. 2)}$$

Here, F is the mutation factor which is taken to be 0.5 and x_{r1}, x_{r2} and x_{r3} are three randomly chosen individuals.

For example, suppose there are 2 randomly chosen individuals i.e. Individual 2 and 4. The difference between genes of Individuals 2 and 4 is calculated. It is then converted into a weighted difference by multiplying the difference with mutation factor (F) as shown below:

Table 3. Calculation of weighted difference

Individual 2	Individual 4	Difference	Weighted difference
0.92	0.12	0.8	0.4
0.22	0.09	0.13	0.07
0.33	0.05	0.28	0.14
0.61	0.91	Mod(-0.3)	0.28
0.58	0.57	0.01	0.05
0.06	0.32	Mod(-0.26)	0.13

This weighted difference obtained is added to a third randomly chosen individual (i.e. Individual 3 in this case) to form a mutant individual as follows:

Table 4. Generation of mutant individual

Weighted difference	Individual 3	Mutant Individual
0.4	0.22	0.62
0.07	0.14	0.21
0.14	0.40	0.54
0.28	0.02	0.3
0.05	0.04	0.09
0.13	0.07	0.2

At the end, the fitness value of trial individual is compared with the fitness of mutant individual, and the individual with minimum fitness value is placed in next generation.

vi. Termination criteria: The termination criterion is maximum number of generations, which is 100 in this case. The algorithm continues for the set number of iterations and then terminates while returning the best schedule and its makespan.

4. SIMULATION RESULTS & COMPARISONS

The results obtained by the proposed algorithm i.e. DGA for JSSP are close to the upper bounds of optimal makespan. The DGA is compared with IPSO-TSAB [14] for 50 problem instances of Taillard Series (TA01-TA50). It is observed that the results obtained by DGA are comparable to those obtained by IPSO-TSAB. The Best Known Solution (BKS) and makespan values obtained by IPSO-TSAB and the Proposed DGA are listed in Table 5. The makespan obtained by DGA is very close to the optimal makespan for large problems too, thus, proving its applicability for large sized problems.

Table 5. Comparative results of algorithms for TA01-TA50 instances

Problem	**Size**	**BKS (or upper bound)**	**IPSO-TSAB**	**Proposed-DGA**
TA01		1231	1231	1231
TA02		1244	1244	1244
TA03		1218	1218	1218
TA04		1175	1175	1175
TA05	15×15	1224	1224	1224
TA06		1238	1238	1238
TA07		1227	1228	1227
TA08		1217	1217	1217
TA09		1274	1274	1274
TA10		1241	1241	1241
TA11		1359	1362	1359
TA12		1367	1370	1367
TA13		1342	1347	1342
TA14		1345	1345	1345
TA15	20×15	1339	1342	1340
TA16		1360	1362	1360
TA17		1462	1468	1462
TA18		1396	1401	1396
TA19		1335	1335	1335
TA20		1348	1352	1349
TA21		1644	1647	1645
TA22		1600	1600	1600
TA23		1557	1557	1557
TA24		1646	1647	1646
TA25	20×20	1595	1597	1595
TA26		1645	1651	1646
TA27		1680	1686	1680
TA28		1603	1617	1604
TA29		1625	1625	1625
TA30		1584	1584	1584
TA31		1764	1764	1764
TA32		1795	1817	1798
TA33		1791	1795	1791
TA34		1829	1830	1829
TA35	30×15	2007	2007	2007
TA36		1819	1819	1819
TA37		1771	1791	1772
TA38		1673	1674	1673
TA39		1795	1795	1795
TA40		1674	1686	1676
TA41		2018	2032	2018
TA42		1949	1962	1949
TA43		1858	1880	1858
TA44		1983	2001	1983
TA45	30×20	2000	2000	2000
TA46		2015	2027	2016
TA47		1903	1921	1904
TA48		1949	1965	1950
TA49		1967	1981	1968
TA50		1926	1950	1930

The comparison of the performance of IPSO-TSAB and the proposed DGA algorithm is given in Figures 2-6 below. It is observed that the performance of the proposed algorithm is similar to IPSO-TSAB for all instances of 15×15 problems.

But the results start improving while going towards larger problems till 30×20 dimensions.

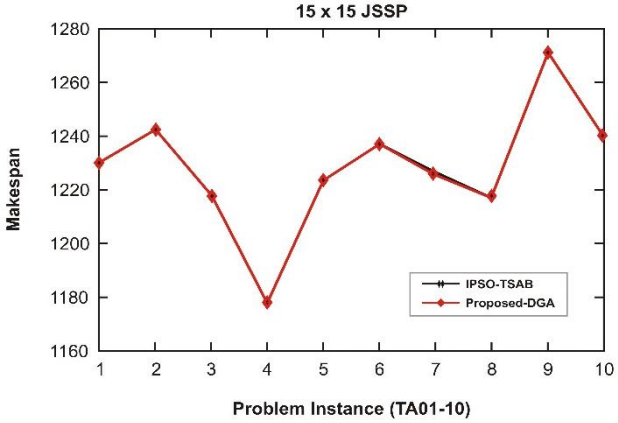

Figure 2: Results of IPSO-TSAB & Proposed DGA for 15×15 JSSP

It is clear from Figure 2 that the value of makespan obtained by IPSO-TSAB and Proposed-DGA are equal to the upper bound, except for an instance TA07, for which the proposed algorithm provides better result.

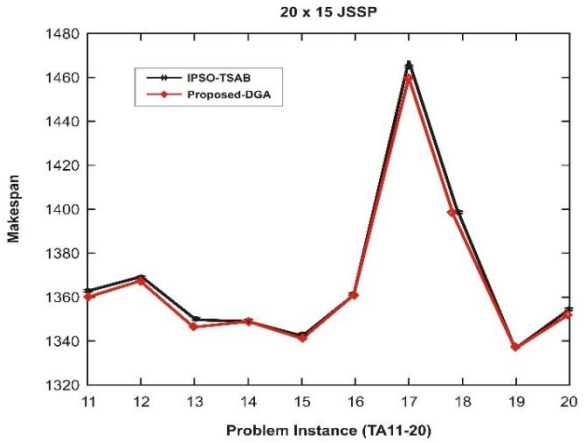

Figure 3: Results of IPSO-TSAB & Proposed DGA for 20×15 JSSP

It is clear from Figure 3 that the proposed algorithm provides good results in almost all the instances for problems with dimension 20×15.

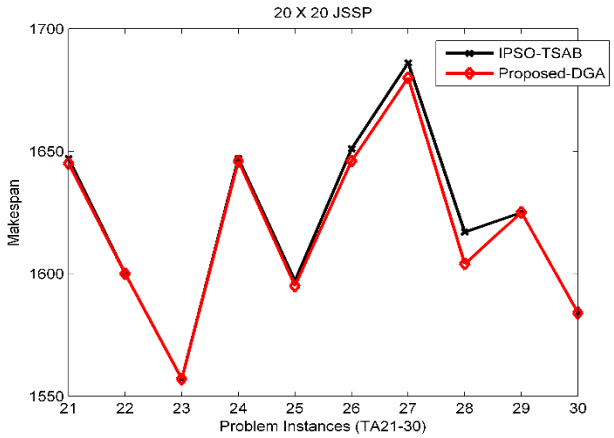

Figure 4: Results of IPSO-TSAB & Proposed DGA for 20×20 JSSP

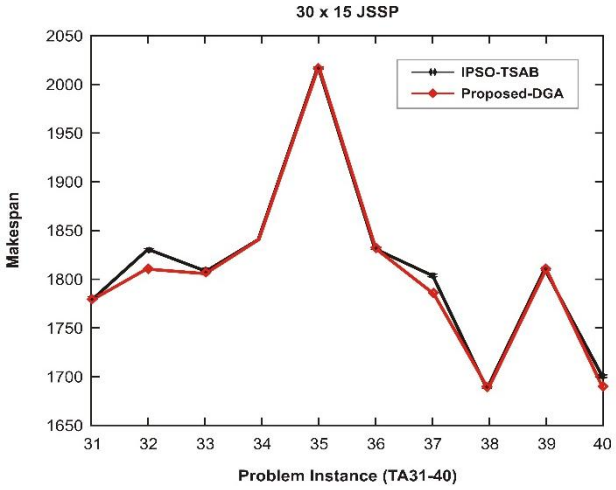

Figure 5: Results of IPSO-TSAB & Proposed DGA for 30×15 JSSP

Similarly, the results for many problem instances of dimension 20×20 and 30×15 are obtained close to the upper bounds of optimal makespan as shown in Figures 4 and 5.

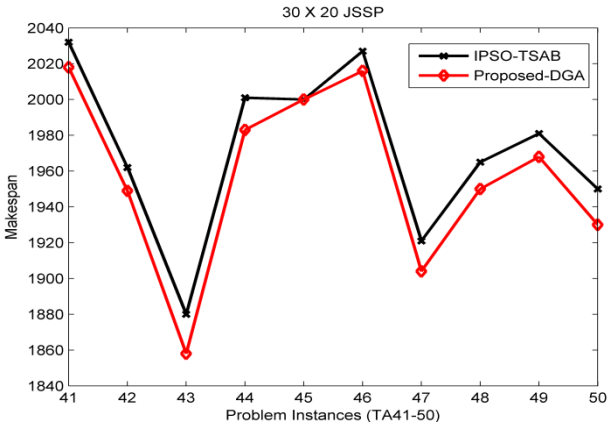

Figure 6: Results of IPSO-TSAB & Proposed DGA for 30×20 JSSP

The difference between the makespan values obtained from both the algorithms is clearly visible in Figure 6. The proposed algorithm improved the makespan value for all the problem instances, except for TA45 for which the IPSO-TSAB also obtained the optimal makespan.

5. CONCLUSIONS & FUTURE SCOPE

In this study, the Job Shop Scheduling Problem (JSSP) has been solved using two evolutionary algorithms, namely Genetic Algorithm (GA) and Differential Evolution (DE). Genetic Algorithm helped in finding global optimal solution and Differential Evolution algorithm was useful in overcoming the problem of premature convergence, by adjusting the parameters in such a way that exploration and exploitation abilities are balanced. The algorithm was run 10 times on Intel Core i3-2330M system for each of the 50 problem instances. It resulted in makespan values close to the Best Known Solution (BKS) for all benchmark instances. The computational results achieved demonstrate the effectiveness of the evolutionary algorithms for solving such problems.

Further, the DGA can be tested on various other benchmark instances like Lawrence, Fisher & Thompson, Applegate & Cook and the remaining instances of Taillard series. It can

also be used for other scheduling problems like Flow Shop problem, Open Shop problem and Mixed shop scheduling problems. The operators of Differential Evolution algorithm can be varied and used on some other problems also. The future research may be focused on developing more efficient meta-heuristic algorithms to find more optimal solutions for the Job Shop Scheduling Problem.

6. ACKNOWLEDGMENTS

The authors gratefully acknowledge Computer Science and Engineering Department, Baba Banda Singh Bahadur Engineering College, Fatehgarh Sahib, Punjab (India) for providing support and computer lab facilities to carry out this research work.

7. REFERENCES

[1] Milosevic, M., Lukic, D., Durdev, M., Antic, A., and Borojevic, S., "An overview of Genetic Algorithms for Job Shop Scheduling problems", Journal of Production Engineering, vol. 18, no. 2, pp. 11-15, 2015.

[2] Korytkowski, P., Rymaszewski, S., and Wisniewski, T., "Ant Colony Optimization for Job Shop Scheduling using multi-attribute dispatching rules", International Journal of Advanced Manufacturing Technology, vol. 67, 2013.

[3] Nguyen, V. and Bao, H. P., "An Efficient Solution to the Mixed Shop Scheduling Problem Using a Modified Genetic Algorithm", In: Procedia Computer Science, vol. 95, pp. 475-482, 2016.

[4] Yenisey, M. M. and Yagmahan, B., "Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends", Omega, vol. 45, pp. 119-135, 2014.

[5] Huang, K. L. and Liao, C. J., "Ant colony optimization combined with taboo search for the job Shop scheduling problem", Computers & operations research, vol. 35, no. 4, pp. 1030-1046, 2008.

[6] Dorndorf, U., Pesch, E. and Phan-Huy, T., "Solving the open shop scheduling problem", Journal of Scheduling, vol. 4, no. 3, pp. 157-174, 2001.

[7] Zhang, R. and Wu, C., "A hybrid immune simulated annealing algorithm for the job shop scheduling problem", Applied Soft Computing, vol. 10, no. 1, pp. 79-89, 2010.

[8] Li, J. Q., Pan, Q. K., Suganthan, P. N. and Chua, T. J., "A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem", The International Journal of Advanced Manufacturing Technology, vol. 52, no. 5, pp. 683-697, 2011.

[9] Watanabe, M., Ida, K. and Gen, M., "A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem", Computers & Industrial Engineering, vol. 48, no. 4, pp. 743-752, 2005.

[10] Lin, T. L., Horng, S. J., Kao, T. W., Chen, Y. H., Run, R. S., Chen, R. J., Lai, J. L. and Kuo, I. H., "An efficient job-shop scheduling algorithm based on particle swarm optimization", Expert Systems with Applications, vol. 37, no. 3, pp. 2629-2636, 2010.

[11] Zhou, R., Nee, A. Y. C. and Lee, H. P., "Performance of an ant colony optimisation algorithm in dynamic job shop scheduling problems", International Journal of Production Research, vol. 47, no. 11, pp. 2903-2920, 2009.

[12] Wu, C., Zhang, N., Jiang, J., Yang, J. and Liang, Y., "Improved bacterial foraging algorithms and their applications to job shop scheduling problems", In: International Conference on Adaptive and Natural Computing Algorithms, Springer Berlin Heidelberg, 2007. pp. 562-569.

[13] Wisittipanich, W. and Kachitvichyanukul, V., "Differential evolution algorithm for job shop scheduling problem", Industrial Engineering and Management Systems, vol. 10, no. 3, pp. 203-208, 2011.

[14] Gao, H., Kwong, S., Fan, B. and Wang, R., "A hybrid particle-swarm tabu search algorithm for solving job shop scheduling problems", IEEE Transactions on Industrial Informatics, vol. 10, no. 4, pp. 2044-2054, 2014.

[15] Chang, H. C., Chen, Y. P., Liu, T. K. and Chou, J. H., "Solving the flexible job shop scheduling problem with makespan optimization by using a hybrid Taguchi-genetic algorithm", IEEE, vol. 3, pp. 1740-1754, 2015.

[16] Goncalves, J. F., de Magalhaes Mendes, J. J. and Resende, M. G. C., "A hybrid genetic algorithm for the job shop scheduling problem", European journal of operational research, vol. 167, no. 1, pp. 77-95, 2005.

[17] Wang, L. and Tang, D. B., "An improved adaptive genetic algorithm based on hormone modulation mechanism for job-shop scheduling problem", Expert Systems with Applications, 2011.

[18] Qing-dao-er-ji, R. and Wang, Y., "A new hybrid genetic algorithm for job shop scheduling problem", Computers & Operations Research, vol. 39, no. 10, pp. 2291-2299, 2012.

[19] Zhang, H., Yan, Q., Zhang, G. and Jiang, Z., "A Chaotic Differential Evolution Algorithm for Flexible Job Shop Scheduling", In: Asian Simulation Conference, Springer, pp. 79-88, 2016.

[20] Ponsich, A. and Coello, C. A. C., "A hybrid differential evolution—tabu search algorithm for the solution of job-shop scheduling problems", Applied Soft Computing, vol. 13, no. 1, pp. 462-474, 2013.

[21] Yuan, Y. and Xu, H., "Flexible job shop scheduling using hybrid differential evolution algorithms", Computers & Industrial Engineering, vol. 65, no. 2, pp. 246-260, 2013.

[22] Vesterstrom, J. and Thomsen, R., "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems", Evolutionary Computation, CEC2004, IEEE, vol. 2, pp. 1980-1987, 2004.

[23] Kitayama, S., Arakawa, M. and Yamazaki, K., "Differential evolution as the global optimization technique and its application to structural optimization", Applied Soft Computing, vol. 11, no. 4, pp. 3792-3803, 2011.

[24] Ying, W. and Bin, L., "Job-shop scheduling using genetic algorithm", In: Systems, Man, and Cybernetics, IEEE International Conference, 1996. vol. 3, pp. 1994-1999.

[25] Taillard, E., "Benchmarks for basic scheduling problems", European journal of operational research, vol. 64, no. 2, pp. 278-285, 1993.

[26] Chen, J. C., Wu, C. C., Chen, C. W. and Chen, K. H., "Flexible job shop scheduling with parallel machines using Genetic Algorithm and Grouping Genetic Algorithm", Expert Systems with Applications, vol. 39, no. 11, pp. 10016-10021, 2012.

[27] Blazewicz, J., Domschke, W. and Pesch, E., "The job shop scheduling problem: Conventional and new solution techniques", European journal of operational research, vol. 93, no. 1, pp. 1-33, 1996.

[28] Cheng, R., Gen, M. and Tsujimura, Y., "A tutorial survey of job-shop scheduling problems using genetic algorithms—I. Representation", Computers & industrial engineering, vol. 30, no. 4, pp. 983-997, 1996.

[29] Mesghouni, K., Hammadi, S. and Borne, P., "Evolutionary algorithms for job-shop scheduling", International Journal of Applied Mathematics and Computer Science, vol. 14, no. 1, pp. 91-104, 2004.

[30] Ranjini, A. and Zoraida, B. S. E., "Analysis of selection schemes for solving job shop scheduling problem using genetic algorithm", International Journal of Research in Engineering and Technology (IJRET), vol. 2, no. 11, pp. 2319-1163, 2013.

[31] Goldberg, D. E. and Deb, K., Foundations of Genetic Algorithms, vol. 1, CA: Morgan Kaufmann Publishers, 1991, A comparative analysis of selection schemes used in genetic algorithms, pp. 69-93.

[32] Aickelin, U. and Dowsland, K. A., "An indirect genetic algorithm for a nurse-scheduling problem", Computers & Operations Research, vol. 31, no. 5, pp. 761-778, 2004.

[33] Yu, J. and Buyya, R., "Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms", Scientific Programming, vol. 14, no. 3-4, pp. 217-230, 2006.

[34] De Falco, I., Della Cioppa, A. and Tarantino, E., "Mutation-based genetic algorithm: performance evaluation", Applied Soft Computing, vol. 1, no. 4, pp. 285-299, 2002.

[35] Mohanty, B., Panda, S., and Hota, P. K., "Differential evolution algorithm based automatic generation control for interconnected power systems with non-linearity", Alexandria Engineering Journal, vol. 53, no. 3, pp. 537-552, 2014.

[36] Storn, R., "On the usage of differential evolution for function optimization", Fuzzy Information Processing Society, NAFIPS, 1996 Biennial Conference of the North American, IEEE. pp. 519-523.

[37] Zaharie, D., "A comparative analysis of crossover variants in differential evolution", In Proceedings of International Multiconference on Computer Science and Information Technology (IMCSIT), 2007. pp. 171-181.

[38] Gao, A., and Zhao, C., "Parameter Controlling and Adjusting Strategy of Differential Evolution Algorithm", Technical Journal of the Faculty of Engineering (TJFE), vol. 39, no. 5, pp. 351-357, 2016.