A Bi-Criteria Approach to the M-machine Flowshop Scheduling Problem

R. Rajkumar [1#], P. Shahabudeen [2], P. Nagaraj [1], S. Arunachalam [3] and T. Page [4]

[1] Department of Mechanical Engineering, Mepco Schlenk Engineering College, Sivakasi, Tamil Nadu, INDIA. Email: rrkumarau@gmail.com

[2] Department of Industrial Engineering, Anna University, Chennai, INDIA.

[3] School of Computing and Technology, University of East London, London.

[4] Department of Design and Technology, Loughborough University, Loughborough, UK.

[#] Corresponding author

Abstract: This paper considers the problem of permutation flowshop scheduling with the objectives of minimising the makespan and total flowtime of jobs, and presents an Improved Genetic Algorithm (IGA). The initial population of the genetic algorithm is created using the popular NEH constructive heuristic (Nawaz *et al.*, 1983). In IGA, multi-crossover operators and multi-mutation operators are applied randomly to subpopulations divided from the original population to enhance the exploring potential and to enrich the diversity of the crossover templates. The performance of the proposed algorithm is demonstrated by applying it to benchmark problems available in the OR-Library. Computation results based on some permutation flowshop scheduling benchmark problems show that the IGA gives better solution when compared with the earlier reported results.

Keywords: Flowshop scheduling, Makespan, Total flowtime, Genetic Algorithm

Dr. R. Rajkumar is currently Assistant Professor at Mepco Schlenk Engineering college, in the Department of Mechanical Engineering. His areas of specialisation are Operation research, Manufacturing scheduling, Meta-Heuristics, etc. He had his Bachelor degree in Mechanical Engineering and Master Degree in Industrial Engineering from Madurai Kamaraj University and Ph.D from Anna University, Chennai. Dr.R.Rajkumar is grateful to the Management, the Principal and the Head of Mechanical Engineering Department of MEPCO Schlenk Engineering College, Sivakasi, Tamilnadu, India for their support of this paper.

Dr. P. Shahabudeen is currently Professor & Head of Industrial Engineering Department at College of Engineering, Anna University, Chennai, India. His areas of specialisation are Simulation, Optimisation, Information Systems, Design of Manufacturing systems etc. He had his Bachelor degree in Mechanical Engineering from Madurai Kamaraj University Madurai, Master Degree as well as Doctoral degree in Industrial Engineering from Anna University. Currently he is guiding doctoral students in areas like Supply Chain Management, Manufacturing Scheduling, Kanban systems, Taguchi Methods in Manufacturing, E commerce applications etc.

Dr. P. Nagaraj is currently Professor & Head in the Department of Mechanical Engineering, Mepco Schlenk Engineering College, Sivakasi, India. His areas of specialisation are Operation Research, Fuzzy logic, Inventory control etc. He had his Bachelor degree in Mechanical Engineering and Master Degree in Industrial Engineering from Madurai Kamaraj University and Ph.D from Bharathiyar University, Coimbatore, India.

Dr. Subramaniam Arunachalam is a senior lecturer in Manufacturing Systems Engineering in university of East London, UK. He has many years of experience in teaching and developing course materials in manufacturing and operations management disciplines. His fields of research interest include lean concepts, manufacturing systems engineering, strategy and management, quality management, supply chain management, manufacturing simulation, production management, total quality management and project management.

Dr T. Page is a lecturer in Electronic Product Design in the Department of Design and Technology at Loughborough University UK. Tom is an external examiner on Engineering and Manufacturing programmes at Sheffield Halllam University. Dr Page is a visiting scholar at Iceland University and the University of Lapland in Finland and has been an external examiner on undergraduate fields in Product Design and Manufacturing Engineering at the University of East London. Dr Page is co-founder of the European Society for Virtual Reality Learning Environment Research and Development.

1. Introduction and Literature Review

The problem of scheduling in permutation flow shops has been extensively investigated by many researchers. Its aim is to determine the sequence for processing jobs on a given set of machines. The need for scheduling arises from the limited resources available to the decision-maker and it is an important aspect of operational level shop floor decisions. Its importance and relevance to industry has prompted researchers to study it from different perspectives. Flowshop is the classical and most studied manufacturing environment in scheduling literature. A general flowshop in which n jobs to be processed through m machines has been considered. The processing times are fixed and non-negative. Further assumptions are that each job can be processed on only one machine at a time, the operations are not pre-emptable, the jobs are available for processing at time zero and setup times are sequence independent. Here we consider the permutation flowshop shop problem, the same job order is chosen on every machine. The objective then is to find a sequence, i.e., a permutation of the numbers 1,…,n that minimizes the makespan and total flow time. Makespan and total flow-time are two commonly used performance measures in flowshop scheduling literature (Baker, 1974; Garey et al., 1976; Nagar et al., 1995). Flowtime is defined as the time spent by each job in the system and makespan is the time at which the last job completes its processing on the last machine. Minimising makespan is important in situations where a simultaneously received batch of jobs is required to be completed as soon as possible, for example, a multi-item order submitted by a single customer that must be delivered in the minimal possible time. The makespan criterion increases the use of resources. There are other real-life situations in which each completed job is required as soon as it is processed. In such situations, we are interested in minimising the mean or sum of flowtimes of all jobs rather than minimising makespan. This objective is particularly important in real-life situations in which reducing inventory or maintaining cost is of primary concern. We have so far seen the development of various heuristic methods that consider a single measure of performance, viz., makespan. However, the desirability of a schedule being evaluated by more than one performance measure is often cited in the literature. Apart from the makespan objective, other significant objectives in flowshop scheduling problem is the minimisation of total (or mean) flowtime of all jobs.

Optimisation algorithms for the two- and three-machine flowshop problems with respect to different objectives have been developed by Johnson (1954) and Ignall and Schrage (1965). The NP- completeness of various scheduling problems has been discussed widely in the literature (Garey et al., 1976, Pinedo, 2002). As the vast majority of flowshop scheduling problems is NP-complete, research is mostly directed towards the development of heuristic or near-optimal methods. Some of the heuristic procedures developed so far are due to Campbell et al., (1970), Dannenbring (1977), King and Spachis (1980), Nawaz et al., (1983), Widmer and Hertz (1989), Osman and Potts (1989), Ogbu and Smith (1990), Ishibuchi et al., (1995), and Ben-Daya and Al-Fawzan (1998). Nagar et al., (1995) gave a survey of the existing multicriteria approaches of scheduling problems. Nagar et al., (1995) were the first to address the two-machine flowshop problem using the weighted sum of makespan and flow-time criteria. They presented a branch-and-bound algorithm that works well for special cases. Yeh (1999) developed additional branch-and-bound algorithms for the same problem. For the problem of scheduling in a flowline-based manufacturing cell with missing operations for jobs in a part-family, Logendran and Nudtasomboon (1991), Sridhar and Rajendran (1993) and Ravindran et al. (2005) have proposed heuristics to minimise makespan and total flowtime, respectively.

In this paper an attempt is made to present an efficient Improved Genetic Algorithm (IGA) to solve the Bi-criteria flowshop scheduling problem for minimising makespan and total Flowtime of jobs. As discussed in the literature review, we have identified that among the flowshop scheduling heuristics, the algorithm by Rajendran (1995) and Ravindran et al. (2005) aims at minimising not only makespan, but also total flowtime of jobs. Ravindran et al. (2005) claimed that their heuristic procedure giving better results than Rajendran (1995). Hence in this work, we have chosen to compare the performance of the proposed algorithm IGA with that of Ravindran et al. (2005) algorithm. The proposed algorithm is coded in C and run on a PC Pentium 2.80 GHz under the Windows operating environment. The results of the evaluation of performance of the proposed algorithm and that of Ravindran et al. (2005) are presented for benchmark problems. An extensive experimental investigation has been conducted to relatively evaluate the performances.

2. Problem Description and Notation

Consider an *m*-machine flowshop where there are *n* jobs to be processed on the *m* machines in the same order. We only consider the permutation schedules, i.e., the same job order on each machine. The objective of this paper is to develop an Improved Genetic Algorithm and hence to find the optimal or near optimal sequence in flowshop environment by minimising makespan and total flowtime. The processing time $t(i,j)$ is given for any job *i* on any machine *j*. Given a permutation (processing sequence) of the jobs $\{J_1, J_2, \ldots, J_n,\}$, the completion times of jobs on the machines, makespan and the total flowtime TFT of the jobs in the flowshop can be calculated as follows:

$C(J_1,1) = t(J_1,1)$

$C(J_i,1) = C(J_{i-1},1)+ t(J_i,1), \quad i = 2, \ldots, n$

$C(J_1,1) = C(J_1,J-1)+ t(J_1,j), \quad j = 2, \ldots, m$

$C(J_1,1) = \max\{ C(J_{i-1},j), C(J_i,j-1)\}+t(J_i,j)$

$i = 2, \ldots, n; j = 2, \ldots, m$

(i) Makespan $C_{\max}$: the length of time required to finish processing all jobs, i.e.

$$C_{\max} = \max\{C_1, C_2, \ldots, C_n\}.$$

Where C_n is the completion time of job *n*.

(ii) Total flowtime (TFT): the total amount of time that all jobs spend in the production system, i.e. **Error!**

Where $C(i,m)$ is the completion time of i^{th} job on last machine *m*.

The weighted sum of the above two objective values are taken as the combined objective function.

Combined objective function,

$$COF = w_1 \times f_1(x) + w_2 \times f_2(x)$$

Where $w_1=0.5$ and $w_2=0.5$.

The objective is to find a permutation of jobs so as to minimize the makespan and total flowtime of jobs. Since flow-shop scheduling problem has been shown to be NP–complete problem, for practical purposes, it is often more appropriate to apply an approximation method which generates a near optimal solution effectively. In this paper an attempt is made to improve the existing genetic algorithm procedures to apply to permutation flowshop scheduling problems to give better results. The following notations are used in this paper.

n	number of jobs
m	number of machines
p_{ij}	processing time of job i on machine j
$C_{\max}$	Makespan
TFT	Total Flow time of jobs
k	current generation number
p_s	population size
p_c	probability of crossover
p_m	probability of mutation
$P(k)$	population at k^{th} generation
r	real random number between 0 and 1
N_g	maximum generation

G_p generation limit for inserting new randomly generated chromosomes

G_m generation limit constant for changing mutation probability

3. Simple Genetic Algorithm

Genetic Algorithms are a class of optimisation algorithms that seek improved performance by sampling areas of the solution space that have high probability of leading to a good solution. They imitate the natural evolutionary process in that, in each generation, the fittest individuals have a better chance to produce off-springs by mixing features of the parents or by altering one or more of the parent characteristics whereas, the worst individuals are most likely to die. Since their introduction, genetic algorithms have been applied to a wide variety of combinatorial optimization problems including the well known Travelling Salesman Problem (Goldberg and Lingle 1985) and the Scheduling Problem (Biegal and Davem 1990, Vempati et al 1993, Neppalli et al 1994 and Chen et al 1995). A survey on applications of GAs can be found in Goldberg (1989). Researchers have used successfully this GA in a wide variety of applications including packing, scheduling, neural networks, traveling salesman, and transport problems (Yenlay 2001, Jaszkiewicz 2002 and Ruben Ruiz et al 2003).

Decisions that have to be made for applying GA include individual or chromosome representation, method of crossover, probability of crossover, method of mutation, probability of mutation, and population size. GA is naturally parallel and exhibits implicit parallelism, which does not evaluate and improve a single solution, but analyses and modifies a set of solutions simultaneously (Goldberg, 1989). The ability of a GA to operate on many solutions simultaneously and gather information from all current solutions to direct search reduces the possibility of being trapped in a local optimum.

The GA attempts to simulate nature's genetic processes by representing a solution to the problem as a string of genes that can take on some value from a specified finite range or alphabet. This string of genes, which represents a solution, is known as a chromosome. Then an initial population of legal chromosomes is constructed at random. At each generation, the fitness of each chromosome in the population is measured. The fitter chromosomes are then selected to produce offspring for the next generation, which inherit the best characteristics of both the parents. After many generations of selection for the fitter chromosomes, the result is hopefully a population that is substantially fitter than the original.

In general, Simple Genetic Algorithm (SGA) consists of the following steps:

Step 1: Initialize a population of chromosomes.

Step 2: Evaluate the fitness of each chromosome.

Step 3: Create new chromosomes by applying genetic operators such as crossover and mutation to current chromosomes.

Step 4: Evaluate the fitness of the new population of chromosomes.

Step 5: If the termination condition is satisfied, stop and return the best chromosome; otherwise, go to Step 3.

4. The Bi-criteria Improved Genetic Algorithm for Flowshop Scheduling

4.1 Framework of the Multi-objective Improved Genetic Algorithm

Genetic algorithms have aroused intense interest in the past few years because of their flexibility, versatility, and effectiveness in solving problems in which traditional optimisation methods are insufficient. GAs need no simplifying assumptions of linearity, continuity, etc., and thus can solve highly complex real-world problems. A GA may lose solutions and substructures because of disruptive effects of genetic operators, and it is not easy to regulate a GA's convergence and hence a pure GA may easily produce premature and poor results (Leung et al., 1997). To enhance the performance of genetic searches and to avoid premature convergence, an Improved Genetic Algorithm is proposed. To enhance the performance of genetic searches and to avoid premature convergence, IGA is incorporated with the following changes.

4.1.1 Initial population

NEH heuristic's (Nawaz et al., 1983) sequence is incorporated in the generation of initial population as the NEH sequence can generate suboptimal solution rapidly. The diversity of the initial population can be maintained to a certain extent, because the other solutions are still generated randomly.

4.1.2 Selection procedure

One of the simplest methods to combine multiple objective functions into a scalar fitness solution is the following weighted sum approach (Venkata Ranga Neppalli 1996 and Murata et al 1996).

$$f(x) = \frac{1}{1 + (w_1 \times f_1(x) + w_2 \times f_2(x))} \quad (1)$$

where $f(x)$ is a combined fitness function (CFF),

$f_1(x)$ is the makespan objective function,

$f_2(x)$ is the total flowtime objective function,

w_i is a weight assigned for each objective function (w_1=0.5 and w_2=0.5)

4.1.3 Crossover operation

In the GA, a single type of crossover operator is applied to the whole population from start to finish, which is not good for retaining useful information and maintaining diversity if the evolution tends to be premature. In IGA, a set of crossover operators two point crossover, partially mapped crossover, similar job order crossover and linear order crossover are used each with certain probabilities. Multiple crossover operators ensure that the diversity can be enhanced and the search region can be extended.

4.1.4 Mutation operation

Mutation generates an offspring solution by randomly modifying the parent's feature. It helps to preserve a reasonable level of population diversity, and provides a mechanism to escape from local optima. For each child obtained from crossover, the mutation operator is applied independently with a probability pm. In this work, three types of mutation operators arbitrary three-job change, arbitrary two-job change and shift change are used. A probability is attached to each type of mutation and each time any one type is selected using Monte arlo simulation.

4.1.5 Elite preserve strategy

Computation shows that when the best solution keeps unimproved for a certain number of generations in the GA process, the solution quality will be difficult to be improved further even if the generation continues. Therefore, in this IGA the following changes are incorporated to avoid premature convergence.

a. Elitist strategy: In multi-objective optimisation problems, a solution with the best value of each objective can be regarded as an elite individual. Therefore we have n elite individuals for an n objective problem. It is natural to think that such solutions are to be preserved to the next generation in genetic algorithms. Therefore, worst chromosomes are removed from the current population and the best chromosome is added into that population.

b. Hypermutation: When the number of generations with out improving the best solution is greater than a pre-specified constant, premature convergence can be assumed then increase the probability of mutation (hypermutation) and continue the search (Venkata Ranga Neppalli et al., 1996). The purpose of increasing the probability is to diversify the population of MOIGA.

Thus, we provide a framework for the Bi-Criteria Improved Genetic Algorithm. IGA preserves the generality of SGA and can be easily implemented and applied to any kind of optimisation problems.

4.2 Steps of Improved Genetic Algorithm

Step 0 (Initialisation): Generate an initial population containing N_{pop} strings where N_{pop} is the number of strings in each population. One chromosome in the initial population is arrived from NEH heuristic and others are generated randomly.

Step 1 (Evaluation): Calculate the combined fitness function (Makespan and total flowtime) of the objective functions for the generated strings.

Step 2 (Selection): Select a pair of strings from the current population by tournament selection and roulette wheel selection with a certain probability attached with each. This step is repeated $N_{pop}/2$ times to produce N_{pop} offspring by the crossover operation in Step 3.

Step 3 (Crossover): For each selected pair, apply a crossover operation viz, Two point, Partially Mapped Crossover or Similar Job Order crossover, to generate an offspring with the crossover probability p_c. A probability is attached to each type of crossover operators and each time any one type is selected using monte carlo simulation. N_{pop} strings should be generated by the crossover operation.

Step 4 (Mutation): For each string generated by the crossover operation, apply a mutation viz, Two-job change mutation or Shift change mutation with a pre specified mutation probability p_m. A probability is attached to each type of mutation operators and each time any one type is selected using monte carlo simulation.

Step 5 (Elitist strategy): Adopt Elitist Strategy. Insert the two best chromosomes into the current population by removing two worst chromosomes (having minimum combined objective function value).

Step 6: (Termination test): If a pre specified stopping condition is not satisfied, return to Step 1.

IGA1: The sequences obtained in all the iterations are compared and the sequence which is having minimum makespan is selected and corresponding total flow time is taken for the resultant sequence. If more sequences having same minimum makespan then the sequence with minimum total flow time among all the sequence of minimum makespan is taken as the resultant sequence.

IGA2: The sequences obtained in all iterations are compared and the sequence which is having minimum total flow time is selected and corresponding makespan time is taken for the resultant sequence. If more sequences having the same minimum total flow time then the sequence with minimum makespan time among all the sequence of minimum total flow time is taken as the resultant sequence.

IGA3: The resultant sequence from the last iteration is taken as the final sequence.

5. Computational Results and Comparisons

To test the performance of the IGA, benchmark problems proposed by Taillard (1993) are selected. Various sizes of the problems with 20 jobs and 5, 10, 20 machines were tested using Improved Genetic Algorithm.

5.1 Result and analysis

Based on the implementation discussed in Section 4, the IGA algorithms are coded in C and run on a personal computer with Pentium 2.8 GHz CPU and 512 Mb RAM. The performance of the IGA for the benchmark problems are compared with the result reported by other heuristics of earlier reported results with the following notations.

IGA1 – Best makespan sequence of proposed Improved Genetic Algorithm

IGA2 – Best total flowtime sequence of Improved Genetic Algorithm

IGA3 – Last iteration sequence of Improved Genetic Algorithm

HAMC1– Best makespan sequence of *Hybrid*

Algorithm for Multi Criterion 1 of Ravindran *et al.* (2005)

HAMC2– Best total flowtime sequence of *Hybrid Algorithm for Multi Criterion* 2 of Ravindran *et al.* (2005)

HAMC3– Last iteration sequence of Hybrid Algorithm for Multi Criterion 3 of Ravindran *et al.* (2005)

5.1.1 Percentage Improvement

To evaluate our IGA, experiments were conducted to make a comparison with existing HAMC algorithms (Ravindran *et al.*, 2005) on the benchmark problem provided by Taillard (1993).

A comparison of makespan and total flowtime obtained using IGA algorithms with HAMC algorithms were made. The Percentage improvement is calculated as follows.

$$\text{Percentage improvement} = \frac{C_X - C_{IGA}}{C_X} \quad (2)$$

where

C_X = Performance measures reported by Ravindran *et al.* (2005)

C_{IGA} = Performance measures obtained using IGA algorithms

Percentage improvement of IGA algorithms over the earlier literature results was found. It was observed that for all the problems quite consistent results were obtained. It is observed that IGA1 provides a 10.20 % average improvement in Makespan and 8.69 % average improvement in total flowtime with respect to HAMC1. IGA2 provides an 8.74 % average improvement in Makespan and 8.02 % average improvement in total flowtime with respect to HAMC2. IGA3 provides a 9.52 % average improvement in Makespan and 8 % average improvement in total flowtime with respect to HAMC3. Also it can be observed that IGA provides overall average improvement percentage as 8.86 with HAMC.

The average difference of makespan and total flow time values between IGA and earlier reported result of HAMC are 186.26 and 2087.45 respectively. For example, IGA1 provides 882 difference in makespan time and 8665 difference in total flowtime with respect to HAMC1 for the problem TA022. Hence, in this work considerable reduction of makespan time and in total flowtime are achieved from IGA for all the Benchmark problems considered.

Figure 1 to 3 illustrates graphically the overall performance of each one of the IGA algorithms over the HAMC's algorithms. Each histogram in Figure 1 to Figure 3 corresponds to the percentage improvement from the result achieved by HAMC algorithm for the benchmarks. Thus, the effectiveness of the IGA versus the HAMC is clearly illustrated in the Figure 1 to 3.

The above discussions show that the proposed IGA is better than the HAMC for all the benchmark problems and it gives superior results. It can be observed that the proposed IGA algorithm is more effective than the existing one for scheduling with the dual objectives of minimising makespan & total flowtime and also this work may be extended for minimising other suitable objectives either combined or individually.

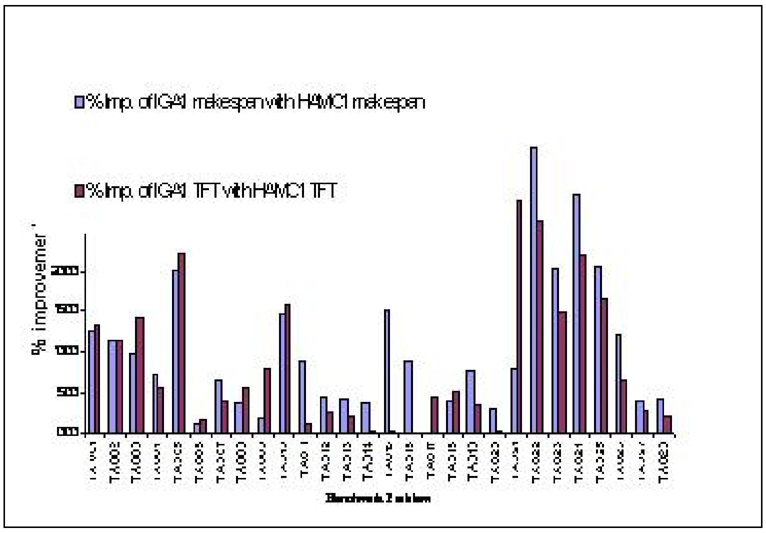

Figure 1. Comparison of IGA1 with HAMC1

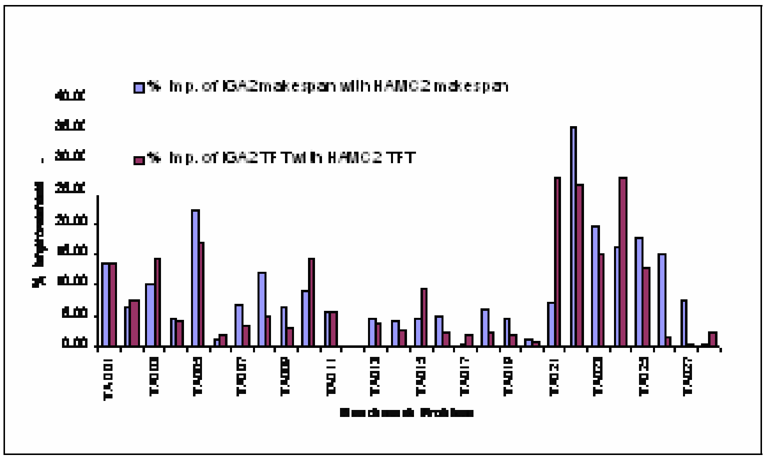

Figure 2. Comparison of IGA2 with HAMC2

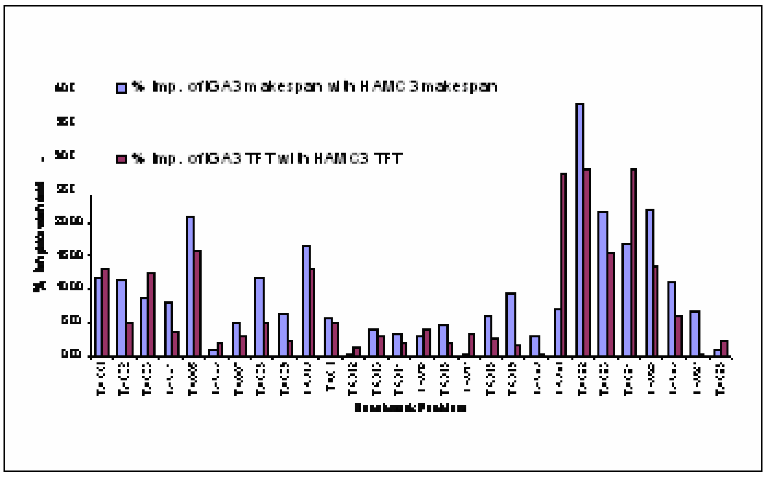

Figure 3. Comparison of IGA3 with HAMC3

6. Conclusion

This paper has addressed the problem of scheduling in flowshop manufacturing systems with the objective of minimizing makespan and total flowtime. A correct formulation of makespan and total flowtime in a flowshop environment is first presented. Then the Improved Genetic Algorithm approach is highlighted under the multi objective criteria. Finally, the proposed methods are applied to a number of multi machine and multi machine combinations which are available as benchmark problems in OR Library. The results show that Improved Genetic Algorithm gives better result than the results of earlier literature. Further this IGA can be extended to handle other types of objectives like minimizing total tardiness, number of tardy job, machine idle time etc. in various manufacturing environments.

REFERENCES

1. BAKER, K. R. (1974) **Introduction to sequencing and scheduling**. Wiley, New York.
2. BEN-DAYA, M. and AL-FAWZAN, M. (1998). **A tabu search approach for the flow shop scheduling problem**. European Journal of Operational Research, Vol. 109, pp. 88-95.
3. BIEGAL J.E. and DAVEM J.J. (1990), **Genetic algorithms and job shop scheduling,** Computers and Industrial Engineering Vol. 19, pp. 81-91.
4. CAMPBELL, H. G., DUDEK, R. A. and SMITH, M. L. (1970) **A heuristic algorithm for the n-job, m-machine sequencing problem**. Management Science, Vol. 16, Part B, pp. 630-637.
5. CHEN, C. L., VEMPATI, V. S., and ALJABER, N. (1995), **An application of genetic algorithms for flow shop problems**, European Journal of Operational Research, Vol. 80, No. 2, pp. 389–396.
6. DANNENBRING, D. G. (1977), **An evaluation of flow-shop sequencing heuristics**. Management Science, Vol. 23, pp. 1174-1182.
7. GAREY M. R., JOHNSON D. S. and SETHI R. (1976), **Complexity of flowshop and jobshop scheduling**, Mathematics of Operations Research; Vol.1. pp. 117-129.
8. GOLDBERG, D. E. (1989), **Genetic algorithms in search, optimisation and machine learning**, Addison Wesley, Reading, MA.
9. GOLDBERG D.E. and LINGLE JR. R. (1985), **Alleles, loci and the Travelling Salesman Problem**, Proceedings of International Conference on Genetic Algorithms and their Applications, Carnegie-Mellon University, Pittsburgh, PA, pp. 154-159.
10. IGNALL, E. and SCHRAGE, L. (1965), **Application of the branch-and-bound technique to some flowshop scheduling problems**, Operations Research. Vol. 13, pp. 400-412.
11. ISHIBUCHI, H., MISAKI, S. and TANAKA H. (1995), **Modified simulated annealing algorithms for the flow shop sequencing problems.** European Journal of Operational Research, Vol. 81, pp. 388-398.
12. JOHNSON, S. M. (1954), **Optimal two and three-stage production schedules with setup times included**, Nav Res Log Vol. 1, pp. 61-68.
13. KING, J. R. and SPACHIS, A. S. (1980), **Heuristics for flow-shop scheduling**. International Journal of Production Research, Vol. 18, pp. 345-357.
14. LEUNG, Y., GAO, Y. and XU Z. B. (1997), **Degree of population diversity – a perspective on premature convergence in genetic algorithms and its Markov-chain analysis**, IEEE Transaction Neural Networks Vol. 8, No. 5, pp. 1165-1176.
15. LOGENDRAN, R. and NUDTASOMBOON, N. (1991), **Minimising the makespan of a group scheduling problem: a new heuristic**. International Journal of Production Research, Vol. 22, pp. 217-230.

16. MURATA T., ISHIBUCHI H. and TANAKA H. (1996), **Multi-objective genetic algorithm and its application to flowshop scheduling,** Computers Ind. Eng., Vol. 30, No. 4, pp. 957-968.

17. NAGAR, A., HADDOCK, J. and HERAGU, S. (1995), **Multiple and bicriteria scheduling: A Literature survey**, European Journal of Operational Research Vol. 81, pp. 88-104.

18. NAWAZ, M., ENSCORE, E. and HAM, I. (1983), **A heuristic algorithm for the m- machine, n- machine flow shop sequencing problem**. Omega Vol. 11, No.1, pp. 91-95.

19. NEPPALLI V.R. CHEN C.L. and ALJABER N.J. (1994), **An effective heuristic for the flow shop problem with weighted tardiness**, Proceedings of the Third Industrial Engineering Research Conference, pp. 634-638.

20. OGBU, F. A. and SMITH D. K. (1990), **The applications of the simulated annealing algorithm to the solution of the n/m/Cmax flowshop problem**. Computers and Operations Research Vol. 17. No. 3, pp. 243-253.

21. OSMAN, I. and POTTS, C. (1989), **Simulated annealing for permutation flow-shop scheduling**. OMEGA, The International Journal of Management Science, Vol. 17, No. 6, pp. 551–557.

22. PINEDO, M. (2002), **Scheduling: Theory, Algorithms, and Systems**, Second Edition, Prentice Hall.

23. RAJENDRAN C. (1995), **Heuristic for scheduling in flow shop with multiple objectives**. European Journal of operational Research, Vol. 82: pp. 540-555.

24. RAVINDRAN, D., NOORUL HAQ, A., SELVAKUMAR, S. J. AND SIVARAMAN, R., (2005), **Flow shop scheduling with multiple objective of minimising makespan and total flow time**. International Journal of Advanced Manufacturing Technology; Vol. 25, pp. 1007-1012.

25. RUBEN RUIZ, CONCEPCION MAROTO and JAVIER ALCARAZ (2003), **New genetic algorithms for the permutation flowshop scheduling problem**, Proceedings of the Fifth Metaheuristics International Conference, Kyoto, Japan, August 25-28, pp 63.1-63.8.

26. SRIDHAR, J. and RAJENDRAN, C. (1993), **Scheduling in a cellular manufacturing system - a simulated annealing approach**. International Journal of Production Research; Vol. 31, pp. 2927-2945.

27. TAILLARD, E. (1993), **Benchmarks of basic scheduling problems**. European Journal of Operational Research Vol. 64, pp. 278–285.

28. NEPPALLI, V. R., CHEN, C-L, and GUPTA, J. N. D. (1996), **The Two-Stage Bicriteria Flowshop Problem - A Genetic Algorithms Approach,** European Journal of Operational Research, Vol. 95, pp. 356-373.

29. VEMPATI V.S., CHEN C.L. and BULLINGTON S.F. (1993), **An application of genetic algorithms to the Flow Shop Problem**, Proceedings of 15th International Conference of Computers and Industrial Engineering.

30. WIDMER, M. and HERTZ, A. (1989), **A new heuristic method for the flow shop sequencing problem**. European Journal of Operational Research, Vol. 41, No. 2, pp. 186–193.

31. YEH, W. C. (1999), **A new branch-and-bound approach for the n/2/flowshop/ $\alpha F+\beta C_{max}$ flowshop scheduling problem**, Computers and Operations Research, Vol. 26, pp. 1293–1310.

32. YENLAY O. (2001), **A comparison of the performance between a Genetic Algorithm and the Taguchi method over artificial problems**, Turkish Journal of Engineering Environmental Science, Vol. 25, pp. 561-568.