A hybrid genetic algorithm for a complex cost function for flowshop scheduling problem

Debraj Bhowmick* and P. Maniyan

National Institute of Industrial Engineering (NITIE),
Vihar Lake, Mumbai 400087, India
E-mail: devraj.007@gmail.com
E-mail: pmaniyan@gmail.com
*Corresponding author

Anjali Saxena

Indian Institute of Foreign Trade (IIFT),
IIFT Bhawan, B-21, Qutab Institutional Area,
New Delhi 110016, India
E-mail: dhatrigwl@gmail.com

Yves Ducq

University of Bordeaux,
IMS UMR 5218 CNRS, 351 Cours de la Libération,
33405 Talence Cedex, France
E-mail: yves.ducq@laps.ims-bordeaux.fr

Abstract: Supply chain excellence has a real impact on business strategy. Manufacturing is an integral part of this strategy represents one of the most exciting opportunities to create value and one of the most challenging tasks for the policy makers. In this paper, we consider a performance criterion for the flowshop scheduling problem that aims to minimise a complex cost function, i.e., the sum of weighted tardiness and weighted flow-time costs. A heuristic and hybrid genetic algorithms are proposed and experimental results are provided. We address this trade-off and propose solution techniques that are easy for the shop-floor manager to implement. As scheduling function is an integral part of supply chain, the proposed solution minimises the opportunity losses and improves cost based supply chain performance. This paper addresses this interesting and challenging domain.

Keywords: flowshop scheduling; hybrid genetic algorithm; complex objective; heuristics.

Reference to this paper should be made as follows: Bhowmick, D., Maniyan, P., Saxena, A. and Ducq, Y. (2011) 'A hybrid genetic algorithm for a complex cost function for flowshop scheduling problem', *Int. J. Electronic Transport*, Vol. 1, No. 1, pp.64–75.

Biographical notes: Debraj Bhowmick is a PhD scholar working in the area of industrial engineering and operations management in NITIE, Mumbai, India. His research interests include flowshop scheduling, meta-heuristics, genetic algorithms and supply chain management.

P. Maniyan is a PhD scholar working in the area of industrial engineering and operations management in NITIE, Mumbai, India. His research interests include scheduling, meta-heuristics transportation and logistics management, and supply chain management.

Anjali Saxena is currently a PhD scholar at Indian Institute of Foreign Trade (IIFT), India. She holds a Master in Operation management and Bachelor in Electrical Engineering. Her research interests include supply chain management, decision support system, service system, and operation management.

Yves Ducq obtained his PhD from the University of Bordeaux 1 in 1999. He is currently a Professor at University Bordeaux 1. He has extensive interest in enterprise modelling. He worked as a Consultant to develop GRAI applications for industry for several years. He has worked on several European projects. As a Researcher, he has more than 80 publications in many renowned journals and conferences. He is guiding many PhD students. His research areas include enterprise modelling, performance evaluation, CIM system, supply chains and service systems. He acts also as a Quality Manager for University Bordeaux 1.

1 Introduction

Today's market can be considered as global and competitive due to dynamic nature of customer demands and frequently changeovers in the business scenarios and continuous evolutions of the new technologies. In order to retain customer goodwill and market share, firms need to manufacture quality products at reduced costs maintaining high responsiveness, thus a transition phase is visualised. Hence, there is a hunt for best compromise manufacturing system. To realise such targets, we consider a performance criterion for the flowshop scheduling problem that aims to minimise a complex cost function, i.e., the sum of weighted tardiness and weighted flow-time costs. The objective was first introduced by Gelders and Kleindorfer (1974) for single machine job shop problem. Later, Gelders and Sambandam (1978) provided heuristics for the *m*-machine permutation flowshop problem. Even though the complex cost function was introduced three decades ago, few researchers worked on it due to its complex nature. The cost function has two main components. The first, weighted tardiness, works in favour of the customer and ensures timely delivery of the job order. The second component, weighted flow-time, considers the shareholder point of view. Thus, this cost function is able to capture two important aspects that any business faces in today's competitive global business environment. This paper aims to provide solution techniques for the complex cost function that are easy to understand and implement on a factory shop floor.

The general flowshop scheduling problem consists of a set, N, of jobs $(1, 2,\dots,n)$ to be processed on a set, M, of machines $(1, 2,\dots,m)$ in series. The flow of jobs is unidirectional since all jobs follow the same technological routing through the machines. Processing of all jobs happen sequentially on multiple machines in the same order. Additionally, each job can be processed on only one machine at a time and each machine can process only one job at a time respectively (Sule, 1997; Gupta and Stafford, 2006). Furthermore, all operations are assumed to be non-preemptable.

The problem is to find a job ordering when a given set of n jobs is to be processed on *m* machines with an assumption that all jobs are simultaneously available. Processing starts on the first machine then on second machine and so on. All jobs follow the same processing order (identical routing). It is further assumed that the job sequences on all machines are the same (no passing allowed). Therefore, only *n*! permutation schedules need to be examined. For each job, there is an assigned due date. If the job is completed on or before its due date, then there is no tardiness penalty incurred. Otherwise, it incurs tardiness penalty at certain rate. In addition, it is assumed that there will be job holding cost for each job. Therefore, the performance criterion is to minimise a complex cost function, i.e., the sum of weighted tardiness and weighted flow time costs.

In this paper, we present a heuristic and hybrid genetic algorithms for the complex cost function. First, we provide a brief literature review. This is followed by the proposed heuristic based on the insertion technique popularised by Nawaz et al. (1983). This is followed by the existing heuristic for the complex cost function given by Gelders and Sambandam (1978) is given. Then, workings of the hybrid genetic algorithm are given. Experimental results are given, which is followed by conclusion.

2 Literature review

Since it was introduced in three decades back, very few researchers have worked on the complex cost function. The weighted tardiness component has made it a difficult problem to deal with. In this section, we present research work done by few authors on this cost function. The complex objective that attempts to minimise the sum of weighted flow-time and weighted tardiness of jobs was first introduced by Gelders and Kleindorfer (1974). The authors minimised the complex objective subject to a given capacity plan for the single machine case. They analysed various lower bounding structures and gave an outline of a branch-and-bound algorithm.

Gelders and Sambandam (1978) presented four heuristics that attempt to minimise the complex objective for the flow shop environment. The heuristics work by building up complete sequences from a null sequence by appending jobs one by one and by considering the sum of idle time of machines, the idle time of the last machine, lower bound on makespan, and the holding and tardiness costs and due-date of jobs (Rajendran and Ziegler, 2003). The authors noted that any algorithm that performs well with tighter due dates would also perform well under slack conditions.

Rajendran and Ziegler (1999) proposed three heuristics for the complex objective and compared them with Gelders and Sambandam (1978). They developed heuristic preference relations by considering lower bounds on the completion times, operation due dates, and weights for holding and tardiness of jobs. They developed one heuristic based on the above heuristic preference relations and proposed two more heuristics to improve the solution given by the previous heuristic. They observe that all three proposed heuristics perform better than the existing heuristics in giving a solution of superior quality and that the first proposed heuristic yields a good solution by requiring a negligible computer time. They also give a simulated annealing algorithm.

Rajendran and Ziegler (2003) gave heuristics for scheduling jobs in a static flowshop with jobs having sequence-dependent setup times. The heuristic starts by constructing a schedule using two heuristic preference relations. To improve the quality of the solution an improvement scheme was used twice. The results were compared with an existing heuristic, a random search procedure, and a greedy local search method. The author found that the proposed heuristic was computationally faster and more effective in yielding solutions of better quality than the benchmark procedures.

The difficulty of solving this problem can be seen from the fact that none of the authors discussed above have proposed exact solution methods like branch-and-bound etc, for the flowshop or multi machine scheduling problem. In this paper we compare the solutions obtained from the proposed heuristic to that given by Gelders and Sambandam (1978). We do not compare our results with Rajendran and Ziegler (1999) as their paper lacked details regarding implementation and experimental set up.

3 Proposed heuristic method

Nawaz et al. (1983) proposed a heuristic for the makespan problem. Henceforth, the Nawaz et al. (1983) heuristic is called NEH in the remaining parts of this paper. NEH heuristic is based on job insertion (insertion neighbourhood) and is one of the most used techniques in flowshop scheduling (Vallada and Ruiz, 2010). Also, Kalczynski and Kamburowski (2007) noted that the NEH heuristic has been commonly regarded as the best heuristic for minimising the makespan in permutation flowshops. Other researchers like Taillard (1990), Park et al. (1984), and Turner and Booth (1987) have confirmed the superiority of NEH over other heuristics. Later, Rajendran and Ziegler (1999) and Woo and Yim (1998) successfully applied NEH type job insertion technique to other objectives. Thus, a heuristic based on job insertion technique used in NEH was developed and applied to the objective function under consideration in this study. Suitable modifications as mentioned above were incorporated in the heuristic to suit the complex objective.

The proposed heuristic follows the same steps used by Nawaz et al. (1983) in their heuristic for the makespan objective function. The NEH heuristic is modified to suit the complex objective function. The notable change is in the evaluation of the objective function. Instead of evaluating makespan, the proposed heuristic evaluates the complex cost function. The steps of the heuristic are reproduced below. It may be noted that in Step 3 the complex cost function is evaluated.

- *Step 1:* For each job i calculate $T_i = \sum_{j=1}^{m} t_{i,j}$, where $t_{i,j}$ is the process time of job i on machine j.
- *Step 2:* Arrange the jobs in descending order of T_i.
- *Step 3:* Pick the two jobs from the first and second position of the list of Step 2, and find the best sequence for these two jobs by calculating the complex cost function for the two possible sequences. Do not change the relative positions of these two jobs with respect to each other in the remaining steps of the algorithm. Set $i = 3$.
- *Step 4:* Pick the job in the i^{th} position of the list generated in Step 2 and find the best sequence by placing it at all possible i positions in the partial sequence found in the previous step, without changing the relative positions to each other of the already assigned jobs.

- *Step 5:* If $n = i$, STOP, otherwise set $i = i + 1$ and go to Step 4.

4 GS heuristic

Gelders and Kleindorfer (1974) were the first to introduce the complex cost function for single machine job shop problem. They proposed a performance criterion to minimise a complex cost function, i.e., the sum of weighted tardiness and weighted flow time costs. Later, Gelders and Sambandam (1978) provided four heuristics for the m-machine permutation flowshop problem. A description of the heuristic provided by Woollam and Sambandam (1985) is given below.

A dispatching index is generated, given by $R_i = M_i / D_i$; where M_i is the relative measure of lateness of job i and D_i is the total idle time of job i on all machines. M_i in turn is found by $M_i = T_i / (T_i - S_\sigma - P_i)$ where T_i is the lower bound on the makespan of job i; S_σ is the finishing time of the partial sequence σ on the last machine m and P_i is the processing time plus the calculated idle time of job i on the last machine m. $D_i = \sum_{j=1}^{m} D_{ij}$ where D_{ij} is the delay or idle time of job i on machine j. The jobs are added to the sequence with respect to the largest R_i. The heuristic follows three principles,

1 uses a dynamic job dispatching rule

2 gives priority to the item which is most expensive to hold

3 fits the jobs together so that idle time is minimal.

5 Hybrid GA

The genetic algorithm (GA) starts by generating initial population randomly. Then parents are selected based on fitness and crossover is performed on them to generate off-springs. Mutation is applied with a probability to the off-springs before it is passed to the hybrid procedure. Hybrid procedure is applied with a probability. If the stopping criterion is not met, some of the better off-springs are included in the new population and replace less fit individuals. A brief description of the hybrid GA is given below.

In this work, for encoding or chromosomal representation we choose to employ the natural representation of a sequencing problem, which is a permutation of jobs. This type of representation is also called as non-linear encoding. This natural representation of the problem has been frequently used by researchers (Reeves, 1995; Chen et al., 1995; Murata et al., 1996). In other words, the string used in this genetic algorithm is a permutation of given jobs.

Initialisation or the initial population of the genetic algorithm is generated randomly. The population is also seeded with good solutions obtained in subsequent runs of the genetic algorithm. Reeves (2003) had stated that 'seeding' the initial population with known good solutions can help the genetic algorithm find better solutions more quickly compared to random start. Hence, initial population of every generation was seeded with solutions from heuristics by Campbell et al. (1970) and Nawaz et al. (1983).

Elitism strategy is used here. According to Reeves (2003), elitism ensures the survival of the best individual so far by preserving it. Hence, subsequent populations were seeded with the best solution found in the previous runs. The remaining members of the population were generated randomly (Reeves, 1995).

The basic idea of selection is that it should be related to fitness (Reeves, 2003). Selection is done as described in Murata et al. (1996). The selection probability was obtained by dividing probability of selecting the fittest string by probability of selecting the average string.

The reproduction compromises crossover and mutation. Crossover replaces some of the genes of one parent by corresponding genes of the other. In mutation, genes of a string are interchanged to promote variety in the population and slowdown the premature convergence in the population.

Three types of crossover operators were used namely, one-point, two-point and uniform crossover. One point and two-point crossovers used here are described in Murata et al. (1996) and uniform crossover is described in Reeves (2003).

Two types of mutation operators were used. The purpose of mutation is to help preserve a reasonable level of population diversity and enables the process to escape from sub-optimal regions of the solution space (Reeves, 2003). Adjacent two-job change and Shift change mutations described in Murata et al. (1996) are used in this work.

5.1 Hybridisation procedures

Hybridisation of the genetic algorithm with local search methods makes the search more effective and more efficient (Tseng and Lin, 2010). In this work, seven local search procedures are given including four novel local search schemes. These procedures intend to search the neighbourhood of the child for better sequences. The sequence with the lowest objective function value is then reinserted into the population. Except the scheme based on adjacent pairwise interchange, which is applied to all child sequences, other hybrid procedures mentioned below are applied with a probability after mutation. Thus, these procedures are not applied to all individuals in the population as proposed by Ruiz et al. (2006). This low probability enables the GA to run fast and prevents it from getting trapped in local optimum.

5.1.1 Hybrid 1: random job insertion search

In this version of local search, a job is selected at random from the given sequence and is inserted in all possible positions back into the sequence. The resulting $(n - 1)$ sequences are evaluated and the best objective and its corresponding sequence is returned. A brief description of the steps in Hybrid procedure described above are given below.

1 Receive sequence σ

- *Step 1:* From the given sequence, remove a job randomly.
- *Step 2:* Place the job in all possible positions in the sequence and generate $(n - 1)$ sequences.
- *Step 3:* Evaluate every sequence for the given objective function and return the best objective function value and sequence. STOP.

5.1.2 Hybrid 2: random block insertion search

In this novel local search scheme, a block of jobs is chosen randomly and removed from the given sequence. These jobs are then reinserted into the sequence one by one. A detailed description of the scheme is as follows. A random point is selected and from this point onwards, a few jobs equal to a random number are selected. These jobs are removed from the sequence and placed in a set of unscheduled jobs in the order in which they were removed. From the set of unscheduled jobs, the first job is removed and inserted in all possible positions in the sequences. The partial sequence that gives the lowest objective function value is selected. Using this sequence, the algorithm proceeds. The remaining unscheduled jobs are inserted in the same manner as described above until the set of unscheduled jobs is empty. The best sequence is chosen and returned. A brief description of the steps used in the procedure are given below.

1 Receive sequence σ

- *Step 1:* From the given sequence σ remove a set of jobs chosen randomly and place them in a set of unscheduled jobs σ'.
- *Step 2:* Remove the first job from the set of unscheduled jobs σ' and place it in all possible positions in the partial sequence σ.
- *Step 3:* Evaluate all the partial sequences generated in Step 2.
- *Step 4:* Choose the partial sequence that gives the best objective function value and using it repeat Steps 1 to 4 until σ' is empty. STOP.

Two termination conditions were set for the genetic algorithm. First, number of generations was restricted to 500. Second, if there were no improvements in the last 100 generations then GA would stop. Termination would take place for any condition that occurred earlier. The upper limit of generations is similar to the stopping criteria proposed by Figielska (2009). Other authors, like Vallada and Ruiz (2010) propose a stopping criterion set to maximum elapsed CPU time of n(m/2) 120 ms, which allows for increased number of observations in order to allow for a more exhaustive analysis. Compared to later, stopping criterion used in this study is restrictive and permits a time bound evaluation of the problem at hand.

5.2 Experimental design and computational results

Experiments were conducted using randomly generated problems to evaluate the proposed heuristics and hybrid genetic algorithm. Problems with randomly generated processing times, due dates, due date penalty and holding cost were used to carry out experiments. As mentioned in Gelders and Sambandam (1978) and Rajendran and Ziegler (1999), the processing times were randomly generated and ranged from 1 to 20, the due date for each job was generated by the sum of the processing time plus a random number up to 5n as given below.

$$\sum_{j=1}^{m} p_{ij} \text{ and } 5.5n.$$

where p_{ij} is the processing time of the i^{th} job on the j^{th} machine. Rajendran and Ziegler (1999) used the same procedure to generate due-dates and found these due dates tight. They commented that such tight due-date setting would bring out the better or poorer relative performance of a heuristic with respect to other heuristics under evaluation. Due date penalty costs per unit time and holding costs per unit time were randomly generated between 0 to 10.

The relative evaluation of results is done as follows: Suppose the schedule given by the i^{th} GA (i = 1 to 10 corresponding to the ten problems) is π_i with its objective function value denoted by Z_i. The relative percentage error of the schedule given by the i^{th} GA solution is:

$$\frac{Z_i - \min\left(Z_i, 1 \le i \le 10\right)}{\min\left(Z_i, 1 \le i \le 10\right)} \times 100$$

Table 1 Comparison of heuristics for small randomly generated problems

Jobs	*Machines*	*Mod NEH*	*GS*
5	5	0.0000	2.6697
	10	0.6714	6.8955
	15	1.1160	0.6674
	20	4.6292	1.3920
	25	1.7999	6.6183
6	5	0.7732	4.5126
	10	2.9261	4.3277
	15	2.1454	6.1984
	20	1.2718	3.0051
	25	1.7039	2.8549
7	5	4.4008	6.0295
	10	2.0887	7.4601
	15	3.5818	4.2052
	20	2.7170	6.6419
	25	0.9005	3.4287
8	5	4.8657	9.7053
	10	4.9057	7.5625
	15	2.8374	7.2616
	20	1.4929	3.4940
	25	3.8995	4.5026
9	5	5.8859	5.9810
	10	2.1871	7.8736
	15	1.7652	9.8444
	20	2.9223	4.2773
	25	1.3050	5.3088
Average		2.5117	5.3087

Experiments were carried out on an Intel P4, 2.5 GHz computer with 256 MB RAM. Experiments were conducted to determine generation size, crossover type, mutation type and mutation probability. Two levels of generation size, i.e., 20 and 40 and two mutation probability settings, i.e., 0.001 and 0.0005 were tested in addition to crossover type and mutation type. In the pilot runs, it was found that generation size 40, uniform crossover, adjacent two-job change mutation with probability of 0.001 performed well. These settings were used to carry out further experiments.

Experiments were carried out for small problems. Jobs ranged from 5 to 9 and machines from 5 to 25. The results from the modified NEH heuristic and the GS (Gelders and Sambandam, 1978) heuristic were compared with results from an adapted enumeration routine given by Sedgewick (1983). The results of this experiment are given in Table 1. It can be seen that the modified NEH heuristic is giving better results as compared to the other heuristic.

Experiments were carried out on large problems. 10, 20 and 30 jobs were considered and 5 to 25 machines were considered. For large problems, the results of heuristics and hybrid genetic algorithms were compared with the lowest result found during that experiment. The results are given in Table 2. It was found that for large problems, the modified NEH performed well and genetic algorithm with random job insertion search hybrid procedure performed well.

Table 2 Comparison of heuristics and hybrid genetic algorithms using randomly generated problems

Jobs	*Machines*	*Mod NEH*	*GS*	*GA Hy 1*	*GA Hy 2*
10	5	4.7224	11.4403	0.0000	0.0000
	10	1.9063	10.2269	0.0000	0.0000
	15	5.2032	8.9749	0.0000	0.0000
	20	2.1493	10.0282	0.0000	0.0000
	25	1.8091	8.1506	0.0000	0.0000
20	5	7.2706	19.3877	0.2835	0.3564
	10	4.7282	18.9907	0.4755	0.5492
	15	5.7012	18.4909	0.4349	0.3458
	20	5.5059	18.7401	0.4115	0.3865
30	5	10.0945	25.9308	1.4771	1.4480
	10	9.4437	24.5958	1.6983	1.7928
	15	8.5349	30.4563	1.5707	1.5742
Average			5.5891	17.1178	0.5293

Experiments were conducted using benchmark problems given by Taillard (1993). The benchmark problems for flowshop have the processing times and all other values required for this work were generated as mentioned above. The results are given in Table 3. It was found that the GS heuristic performed better on Taillard's problems as compared to randomly generated problems with randomly generated processing times. In case of GA, GA with random block insertion search hybrid procedure performed well. It can be noted that, even though the modified NEH has performed well as compared to GS heuristic, the performance of GS heuristic has improved for Taillard benchmark problems. These problems are known to be difficult to solve. It can be concluded that, the GS heuristic may perform well when the problems are difficult to solve.

Table 3 Comparison of heuristics and hybrid genetic algorithm using Taillard (1993) benchmark problems

Problem	*No.*	*Mod NEH*	*GS*	*GA Hy 1*	*GA Hy 2*
tai_20 × 5	1	5.4204	10.9802	0.2786	0.3297
tai_20 × 5	2	3.7423	18.0838	0.5922	0.3276
tai_20 × 5	3	4.8909	24.1708	0.1839	0.0865
tai_20 × 5	4	1.6216	22.7271	0.1222	0.1956
tai_20 × 5	5	5.3788	15.3742	0.0000	0.0000
tai_20 × 5	6	6.7026	18.6937	0.4059	0.6241
tai_20 × 5	7	11.2529	9.8567	0.5595	0.3516
tai_20 × 5	8	6.8598	12.0632	0.1508	0.4120
tai_20 × 5	9	8.0721	19.1364	1.1768	0.8666
tai_20 × 5	10	6.9130	14.6651	0.8021	0.8653
tai_20 × 20	1	5.6319	10.4552	0.1014	0.1427
tai_20 × 20	2	3.2486	26.3637	0.1698	0.1278
tai_20 × 20	3	2.7987	10.8161	0.6060	0.4486
tai_20 × 20	4	5.4970	18.2247	0.1836	0.1224
tai_20 × 20	5	4.9844	9.4495	0.1860	0.1010
tai_20 × 20	6	3.6799	15.2860	1.0198	1.0308
tai_20 × 20	7	4.0197	17.2734	0.5100	0.5565
tai_20 × 20	8	6.3355	8.5175	0.3942	0.2686
tai_20 × 20	9	5.7741	18.4422	0.5992	0.4895
tai_20 × 20	10	8.5947	10.5505	0.3916	0.3585
Average		5.5709	15.5565	0.4217	0.3853

6 Academic and managerial implications

The complex objective is one of the few objective functions that can entice the interest of a practicing manager and scientific community. The objective tries to find a balance between the shareholders and customers points of view, which is relevant to the manager on the shop floor. In addition to this, researchers got further interested when they found that the objective poses computational challenges, and stretches the limits of existing solution methods.

From the managerial perspective, a heuristic solution method proposed in this paper is easy to understand and implement for small problems. On this note, the authors would like to mention that in order to avoid using sophisticated solution methods like genetic algorithms, the shop floor manager can reduce the time horizon of problem at hand and break the problem into smaller problems. These smaller problems can then be solved using the proposed heuristic method. Even though this may not be a scientifically sound way to address the problem at hand, but contribute to the understanding of the nature of the problem.

From the operations management researchers' point of view, it is an irony that such a useful objective function that has practical implications on the shop floor is difficult to solve and has stretched the limits of available solution methods. This objective would prove to be challenging in future as researchers would be drawn back to it when new advances in solution methods would be found.

7 Conclusions

This paper deals with the complex cost function comprising the sum of weighted tardiness and weighted flow-time costs. This unique objective takes care of the customer and the share holder of the company. The importance of such objectives increases in today's global business environment where often meeting customer expectations is a tough and demanding challenge and most of the times companies do so at the expense of shareholder returns. In this paper we try to address this trade-off and propose solution techniques that are easy for the shop-floor manager to implement. Efforts can be directed towards a focus on the study of alternative measures of performances instead of cost.

In this paper, we presented a heuristic and two hybrid genetic algorithms for the complex cost function. Experiments were conducted to evaluate the performance of the heuristics and hybrid GA. Small, medium, large and benchmark problems were used to conduct the experiments. Among the heuristics, it was found that NEH heuristic developed for makespan objective outperformed GS (Gelders and Sambandam, 1978) heuristic. It was also found that the GS (Gelders and Sambandam, 1978) heuristic performed well for difficult to solve benchmark problems given by Taillard (1993) as compared to randomly generated problems. It can be noted that the famous NEH heuristic for makespan objective based on job insertion technique was modified and applied to this problem with promising results. This heuristic is simple to use and understand. It can be used by shop-floor managers to solved small problems in a short time. Since the NEH heuristic minimises makespan, it might tend to reduce in-process holding costs as well as tardiness costs. This might be a major contributing factor to the success of the modified NEH heuristic in solving the complex cost function. Our research is continuing in this direction.

In scope for further research, it can be seen from the literature review, the complex objective has proved to be a challenging problem for the flowshop researchers. Even though some effort has been made by researchers to deal with problem, there is a need to develop branch and bound schemes. This would help in deepen the understanding of the behaviour of the problem. On the genetic algorithms front, adaptive population sizing schemes and dynamic parameter setting strategies can be developed to suit the complex objective.

References

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, No. 10, pp.630–637.

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.

Figielska, E. (2009) 'A genetic algorithm and a simulated annealing algorithm combined with column generation technique for solving the problem of scheduling in the hybrid flowshop with additional resources', *Computers and Industrial Engineering*, Vol. 56, No. 1, pp.142–151.

Gelders, L.F. and Kleindorfer, P.R. (1974) 'Coordinating aggregate and detailed scheduling in the one-machine job shop: I – theory', *Operations Research*, Vol. 22, No. 1, pp.46–60.

Gelders, L.F. and Sambandam, N. (1978) 'Four simple heuristics for scheduling a flow-shop', *International Journal of Production Research*, Vol. 16, No. 3, pp.221–231.

Gupta, J.N.D. and Stafford, E.F. (2006) 'Flowshop scheduling research after five decades', *European Journal of Operational Research*, Vol. 169, No. 3, pp.699–711.

Kalczynski, P.J. and Kamburowski, J. (2007) 'On the NEH heuristic for minimizing the makespan in permutation flow shops', *Omega*, Vol. 35, No. 1, pp.53–60.

Murata, T., Ishibuchi, H. and Tanaka, H. (1996) 'Genetic algorithms for flowshop scheduling problems', *Computers and Industrial Engineering*, Vol. 30, No. 4, pp.1061–1071.

Nawaz, M., Enscore, E.E. and Ham, I. (1983) 'A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem', *Omega*, Vol. 11, No. 1, pp.91–95.

Park, Y.B., Pegden, C.D. and Enscore, E.E. (1984) 'A survey and evaluation of static flowshop scheduling heuristics', *International Journal of Production Research*, Vol. 22, No. 1, pp.127–141.

Rajendran, C. and Ziegler, H. (1999) 'Heuristics for scheduling in flowshops and flowline-based manufacturing cells to minimize the sum of weighted flowtime and weighted tardiness of jobs', *Computers and Industrial Engineering*, Vol. 37, No. 4, pp.671–690.

Rajendran, C. and Ziegler, H. (2003) 'Scheduling to minimize the sum of weighted flowtime and weighted tardiness of jobs in a flowshop with sequence-dependent setup times', *European Journal of Operational Research*, Vol. 149, No. 3, pp.513–522.

Reeves, C. (2003) 'Genetic algorithms', in Glover, F. and Kochenberger, G.A. (Eds.): *Handbook of Metaheuristics*, Chapter 2, pp.55–82, Kluwer Academic Publishers.

Reeves, C.R. (1995) 'A genetic algorithm for flowshop sequencing', *Computers and Operations Research*, Vol. 22, No. 1, pp.5–13.

Ruiz, R., Maroto, C. and Alcaraz, J. (2006) 'Two new robust genetic algorithms for the flowshop scheduling problem', *Omega*, Vol. 34, No. 5, pp.461–476.

Sedgewick, R. (1983) *Algorithms*, Addison-Wesley, Reading, Massachusetts.

Sule, D.R. (1997) *Industrial Scheduling*, PWS Pub. Co Boston.

Taillard, E. (1990) 'Some efficient heuristic methods for the flow shop sequencing problem', *European Journal of Operational Research*, Vol. 47, No. 1, pp.65–74.

Taillard, E. (1993) 'Benchmarks for basic scheduling problems', *European Journal of Operational Research*, Vol. 64, No. 2, pp.278–285.

Tseng, L.Y. and Lin, Y.T. (2010) 'A hybrid genetic algorithm for no-wait flowshop scheduling problem', *International Journal of Production Economics*, Vol. 28, No. 1, pp.144–152.

Turner, S. and Booth, D. (1987) 'Comparison of heuristics for flow shop sequencing', *Omega*, Vol. 15, No. 1, pp.75–85.

Vallada, E. and Ruiz, R. (2010) 'Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem', *Omega*, Vol. 38, Nos. 1–2, pp.57–67.

Woo, H.S. and Yim, D.S. (1998) 'A heuristic algorithm for mean flowtime objective in flowshop scheduling', *Computers & Operations Research*, Vol. 25, No. 3, pp.175–182.

Woollam, C.R. and Sambandam, N. (1985) 'The flow shop problem with a composite cost function', *Computers and Industrial Engineering*, Vol. 9, No. 1, pp.83–89.