Sādhanā Vol. 37, Part 6, December 2012, pp. 695–707. © Indian Academy of Sciences

Minimizing makespan for a no-wait flowshop using genetic algorithm

IMRAN ALI CHAUDHRY* and ABDUL MUNEM KHAN

National University of Sciences and Technology, H-12 Islamabad 46000, Pakistan
e-mail: imran_chaudhry@yahoo.com; munem-cae@nust.edu.pk

MS received 5 May 2012; revised 31 July 2012; accepted 6 September 2012

Abstract. This paper explains minimization of makespan or total completion time for n-jobs, m-machine, no-wait flowshop problem (NW-FSSP). A spread sheet based general purpose genetic algorithm is proposed for the NW-FSSP. The example analysis shows that the proposed approach produces results are comparable to the previous approaches cited in the literature. Additionally, it is demonstrated that the current application is a general purpose approach whereby the objective function can be tailored without any change in the logic of the GA routine.

Keywords. No-wait; flowshop; scheduling; genetic algorithm (GA).

1. Introduction

Flowshop scheduling problems have been extensively studied due to their application in industry and have thus attracted attention from many researchers. For flowshop scheduling problem, most of the researchers assume that there is an unlimited storage available between machines. However, in many practical cases there is no longer a need for blocking between stages or intermediate storage. In this case all the operations of a particular job are required to be processed from start to finish without any interruptions either on or between machines. The primary reason for the occurrence of a no-wait or blocking production environment lies in the production technology itself. In some processes, for example, the temperature or other characteristics (such as viscosity) of the material require that each operation follow the previous one immediately. Such situations commonly arise in the production of steel, where molten steel undergoes a series of operations such as moulding into ingots, unmoulding, reheating, soaking, and preliminary rolling. Similarly, to prevent degradation of product in the plastic moulding and silverware production, a series of processes must follow one another immediately. Further examples occur in the pharmaceutical and chemical industries. Similarly, the canning operation in the food processing industry must immediately follow cooking to ensure freshness. The second reason for the occurrence of a no-wait or blocking production environment is lack of storage (or buffer) capacity between intermediate work stations or machines. Two different environments arise here. In no-wait scheduling, a job must leave a machine immediately after processing is completed.

*For correspondence

Garey & Johnson (1979) show that the no-wait FSSP is NP-hard even for two machine environment. Similarly, Rock (1984) proved that no-wait FSSP with makespan criterion is NP-hard. A comprehensive survey of the research and applications of no-wait flow shop scheduling problem has been given by Hall & Sriskandarajah (1996).

In this paper, we present a GA implemented within a spreadsheet environment to minimize makespan for a no-wait flowshop scheduling problem. Makespan is defined as the completion time at which all jobs complete the processing or equivalently may be termed as maximum completion time of jobs. A review of flowshop scheduling with makespan criterion has been given by Hejazi & Saghafian (2005).

The paper is organised as follows. Review of papers related to no-wait flowshop scheduling with respect to makespan criterion is given in section 2. Section 3 describes the problems and assumptions underlying the paper. Brief introduction to GAs and its components as implemented in this paper are given in section 4. Section 5 gives the implementation details while experimental analysis and conclusions are given in sections 6 and 7, respectively.

2. Literature review

First reported instance in literature for no-wait flowshop scheduling although in German has been presented by Piehler (1960). The author showed that NW-FSSP is equivalent to a travelling salesman problem (TSP). Wismer (1972); Reddi & Ramamoorthy (1972) also modelled the NW-FSSP as a TSP and then employ known solution techniques to minimize the makespan. Bonney & Gundry (1976); King & Spachis (1980) developed heuristic method for minimizing the makespan. Szwarc (1983) presented a heuristic method based on Gilmore & Gomory (1964) algorithm. Gangadharan & Rajendran (1993); Rajendran (1994) presented simple heuristic algorithms. These heuristics gave superior performance as compared to the two heuristics by Bonney & Gundry (1976); King & Spachis (1980). Aldowaisan & Allahverdi (2003) proposed two heuristics that are based on simulated annealing (SA) and Genetic Algorithm (GA) techniques. The authors show that the SA performs better than the GA and heuristics by Gangadharan & Rajendran (1993); Rajendran (1994). Schuster & Framinan (2003) presented a hybridized genetic algorithm and simulated annealing called GASA. This method is superior to Rajendran (1994). Aldowaisan & Allahverdi (2004) proposed several new heuristics and show that its performance is better than the GA technique (Aldowaisan & Allahverdi 2003).

Grabowski & Pempera (2005) developed and compared different local search algorithms for the no-wait flow-shop problem with makespan minimization and present variants of descending search and tabu search algorithms. The authors show that the proposed algorithms are found to be relatively more elective in finding better quality solutions than existing algorithms. Kalczynski & Kamburowski (2007) identify networks whose longest path lengths represent the makespans. The networks lead to a natural reduction of the no-wait flow shop problem to the travelling salesman problem. Liaw (2008) considered the problem to minimize makespan for a two-machine no-wait job shops. The authors developed a two-phase heuristic where initially the problem is transformed into a no-wait flow shop problem and then solving it with the well-known Gilmore & Gomory algorithm (1964) while in the phase 2, TS algorithm is used to improve the solution. Liu *et al* (2007) proposed the particle swarm optimization based algorithms. They develop local search methods based on simulated annealing and Nawaz–Enscore–Ham (NEH) heuristic. Both the local search methods are then hybridized, which they call their first algorithm, while the PSO and simulated annealing based local search method was called as their second algorithm.

Pan *et al* (2008a) proposed a hybrid discrete particle swarm optimization algorithm to solve the no-wait flow shop scheduling problems with minimization of makespan as the objective function. Pan *et al* (2008b) presented a discrete particle swarm optimization algorithm to solve NW-FSSP with both makespan and total flowtime criteria. The authors demonstrate that the algorithm generated either competitive or better results than those already reported in the literature.

Pan *et al* (2008c) presented an improved iterated greedy algorithm to minimize makespan for the no-wait flowshop scheduling problem. The superiority of the proposed improved iterated greedy algorithm (IIGA) in terms of effectiveness and efficiency is demonstrated. Framinan & Nagano (2008) proposed a new heuristic based on travelling salesman problem. They conclude that with respect to the computational effort, there are no significant differences between the proposed heuristic and the other heuristics under comparison. Li & Wu (2008) presented a fast composite heuristic (FCH). Experimental results show that the heuristic requires far less computation time without compromising the effectiveness of the FCH and is similar to that of the best methods. FCH can also be applied in real time scheduling and rescheduling for no-wait flow shops. Li *et al* (2008) proposed a composite heuristic to minimize makespan for no-wait flowshop. An objective increment method is then used to judge whether a new schedule is better than its parent or not. The authors demonstrated that the proposed heuristic is suitable for large-scale practical no-wait flow shops. Laha & Chakraborty (2009) presented a new constructive heuristic, based on the principle of job insertion, to minimize makespan in no-wait flow shop scheduling problems. Bożejko & Makuchowski (2009) described a hybrid tabu search algorithm in a no-wait job shop problem with a makespan criterion. Qian *et al* (2009) proposed an effective hybrid differential evolution (HDE) for the NW-FSSP with the makespan as the objective function. Tseng & Lin (2010) proposed a hybrid genetic algorithm to solve the NW-FSSP with the makespan objective. The authors demonstrated the advantage of combining the two local search methods. Jarboui *et al* (2011) also proposed a hybrid genetic algorithm for the no-wait flowshop scheduling. The results indicate that the solutions obtained by the proposed algorithm are efficient both in terms of time requirement and quality of solution.

3. Problem and assumptions

No-wait flowshop scheduling problem can be described as follows (Zhu *et al* 2008): given the processing time t_{ij}, for job i and machine j, each of n jobs will be sequenced through m machines. Each job i has a sequence of m operations. To satisfy the no-wait restriction, the completion time of the operation o_{ij} must be equal to the earliest time to start of the operation $o_{i,j+1}$. There must not be any waiting time between the processing of any consecutive operations of each of n jobs. The problem is to minimize the makespan or total completion time of all the jobs without waiting between successive operations of a job. The following assumptions are taken in no-wait flowshops: (i) All jobs are available at zero time; (ii) there is no breakdown of the machines and are thus always available; (iii) the processing time of each job is known and constant; (iv) removal and setup times are included in processing times; (v) transportation times are negligible; (vi) jobs follow a pre-determined ordered sequence of operations; (vii) at a given time only one job can be processed on one machine; (viii) once operation has started on a machine, it cannot be interrupted before completion, either on or between the machines.

4. Genetic algorithms

Genetic algorithms (GAs) are search heuristics that mimic the process of natural evolution. GAs were initially introduced by Holland (1975) in United States in the 1970s at the University of Michigan. To use a genetic algorithm, we first represent the solution to our problem as a chromosome or genome. The algorithm then randomly generates a population of solutions. The genetic operators, namely, crossover and mutation are then applied to evolve the solutions in order to find the best solution. Detailed introduction of GAs is given by Goldberg (1989). GAs find application in computational science, economics, engineering, phylogenetics, chemistry, bioinformatics, manufacturing, physics, manufacturing and many other fields. Detailed analysis is provided in Davis (1991).

Davis (1985) reported the earliest application of GA in scheduling. Chaudhry & Drake (2008); Chaudhry (2009) and Nanvala (2011) have presented a recent review of GAs applied to production scheduling.

In this paper, we employ a commercial GA Evolver (1998), which functions as an add-in Microsoft Excel^{TM} spreadsheet. No-wait flowshop model is built in the spreadsheet using the built in functions. Evolver generates a number of trial solutions and uses genetic algorithms to continually improve results of each trial. Each possible solution becomes an independent 'organism' that can 'breed' with other organisms. The spreadsheet model acts as an environment for the organisms, determining which are 'fit' enough to survive based on their results. The adjustable cells (variables), constraints, and the objective function are specified within the corresponding locations of the spreadsheet. Figure 1 illustrates the Evolver-spread sheet architecture.

Various benefits were accrued from the proposed approach. It has found that that the program runs in the background, thus freeing the user to work in the fore ground. Moreover, due to the familiar layout of the spread sheet software helps the user to use the software easily and carry out what-if analysis.

4.1 *Chromosome representation*

We use permutation representation for the no-wait flowshop scheduling problem. Permutation representation can be described from the following example: consider five jobs A-B-C-D-E, one possible chromosome according to permutation representation would be A-C-D-E-B, while another could be A-D-B-E-C.

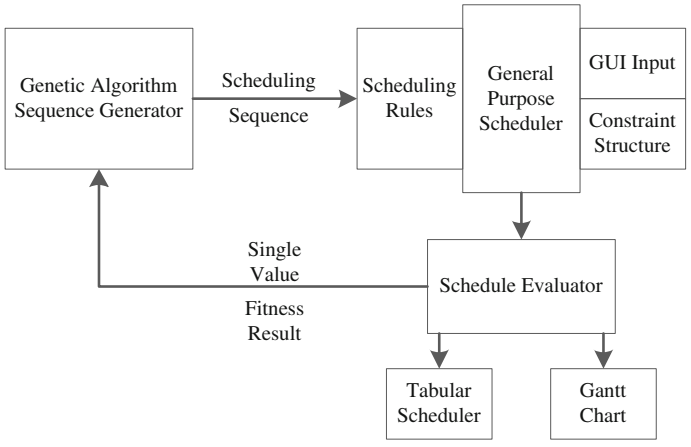

Figure 1. Evolver-spreadsheet architecture.

4.2 *Reproduction and selection*

For reproduction and selection, steady state approach is employed. In this approach, after the application of crossover, only one child is produced as compared to other approaches where, two children are produced. The resulting child solution is put back into the population. If the fitness of the child is better than the other members of the population then least fit member of the population is discarded, otherwise the child solution is discarded.

As far as parent selection is concerned, in Evolver uses a rank-based mechanism to choose the parents. In this method, relative position or rank of a chromosome in the population defines the selection probabilities. A smoother selection probability curve is offered by rank-based approach. The fitness assigned to each individual depends only on its position in the individual rank and not on the actual objective value.

4.3 *Crossover operator*

The crossover operator is the primary operator in GA. During the crossover process recombination of bit strings are formed via an exchange of segments between a pairs of chromosomes. For the no-wait FSSP, permutation representation was best suited to represent the jobs thus the 'Order Solving Method' of the software was used. In this research, order crossover operator (1991) is used to perform crossover operation. This builds offspring by choosing a subsequence from one parent and preserving the relative order of jobs from the other. For example, the two parents

$$[\mathrm{P1}] = (5\ 7\ |9\ 1\ 2\ 3|\ 6\ 8\ 4) \text{ and}$$
$$[\mathrm{P2}] = (8\ 6\ |3\ 7\ 1\ 5|\ 4\ 9\ 2)\,.$$

First, the genes between the two cut points are copied into the child solution

$$[\mathrm{C1}] = (\_\ \_\ 9\ 1\ 2\ 3\ \_\ \_\ \_) \text{ and}$$
$$[\mathrm{C2}] = (\_\ \_\ 3\ 7\ 1\ 5\ \_\ \_\ \_)\,.$$

Next, starting from the second cut point of [P1] parent, the jobs from the second parent are copied, except which are already in child [C1], we get 4-8-6-7-5. This sequence is placed in the first child.

$$[\mathrm{C1}] = (7\ 5\ 2\ 1\ 9\ 3\ 4\ 8\ 6)\,.$$

Similarly, we get the other child solution: [C2] = (9 2 3 7 1 5 6 8 4).

4.4 *Mutation operator*

During crossover operation the search is constrained to genes which exist in the population. This phenomenon is overcome in the mutation operation by simply randomly selecting any bit position in a string and changing it. This is useful since crossover and inversion may not be able to produce new alleles if they do not appear in the initial generation. The 'Order Solving Method' performs order-based mutation as we need to preserve all the original values. In this type of mutation, two positions are randomly selected, and two characters (genes) in these positions are interchanged. For example, in a parent [1 2 3 4 9 6 7 8 5], with selected positions 2 and 5, the resultant child would be [1 9 3 4 2 6 7 8 5]. As the mutation rate setting (from 0 to 1) increased or decreased, the number of swaps performed is increased or decreased proportionately.

Table 1. Job data for example no-wait FSSP.

	Processing time on machine		
Job	*1*	*2*	*3*
1	3	2	4
2	4	5	3
3	1	4	5
4	1	3	2
5	4	3	7

5. Implementation details

Consider the $5/3/F/C_{max}$ example given in table 1.

A Gantt chart for the sequence of jobs 5-3-2-1-4 produced with wait times between the jobs is shown in figure 2.

The flowchart in figure 3 shows the working of GA for NW-FSSP within the spreadsheet environment.

The 'no-wait' spreadsheet model works in two stages. In the first stage, the delay in the start of the job p after q is calculated. In the second stage, the job is delayed by that many units as calculated in stage one to produce a valid no-wait schedule. The delay of starting job q after p is calculated by the equation given by Reddi & Ramamoorthy (1972). Let $D(p, q)$ be the minimum delay time between the completion of job J_p and the initiation of J_q, then $D(p, q)$ is given by:

$$\begin{aligned} D(p,q) &= \max\left(t_{p,2} - t_{q,1},\ t_{p,2} + t_{p,3} - \left(t_{q,1} + t_{q,2}\right), \ldots\ldots\ldots,\right. \\ &\qquad \left. t_{p,2} + t_{p,3} + t_{p,4} + \ldots\ldots\ldots + t_{p,m} - \left(t_{q,1} + t_{q,2} + \ldots\ldots\ldots + t_{q,m-1}\right), 0\right) \\ &= \max_k \left(\sum_{i=2}^{k} t_{p,i} - \sum_{i=1}^{k-1} t_{q,i}, 0\right), \qquad 2 \le k \le m, \end{aligned}$$

where

$t_{p,i}$ = process time for job p on machine i
m = number of machines.

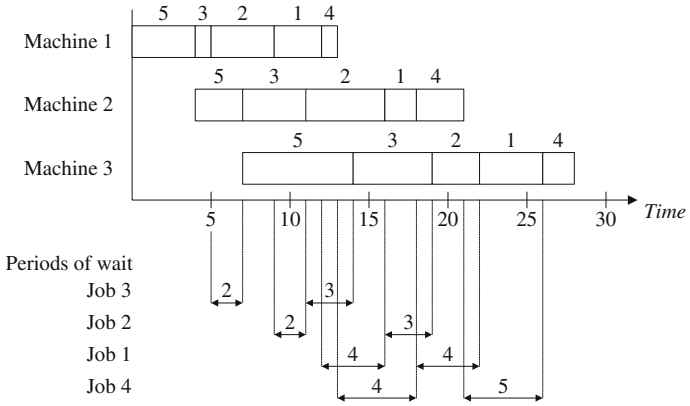

Figure 2. Gantt chart showing job sequence 5-3-2-1-4 and the periods of wait for the jobs.

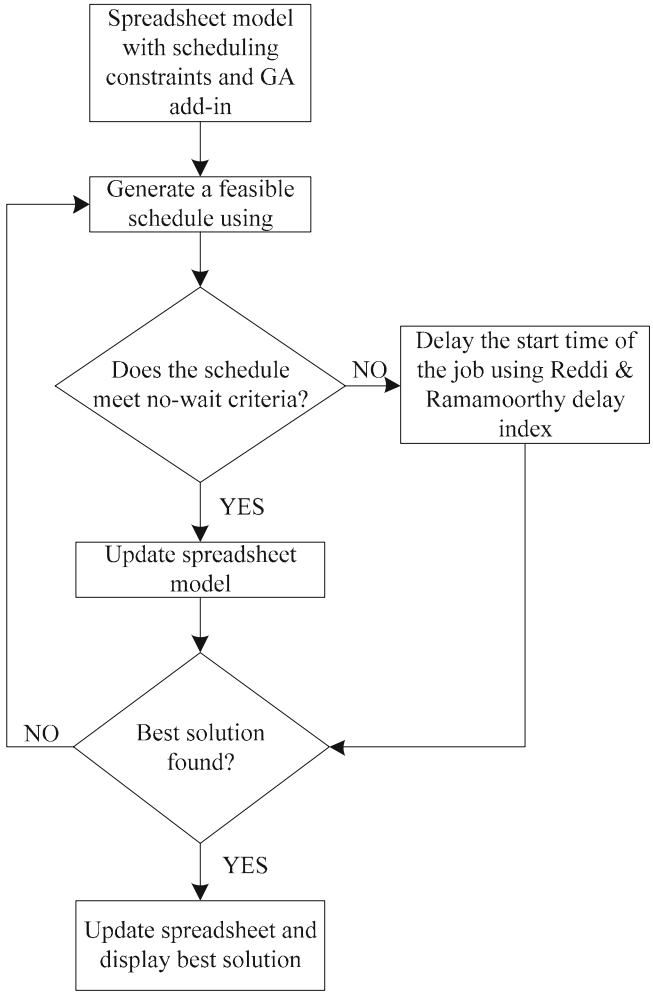

Figure 3. Flowchart for the working of GA in spreadsheet environment.

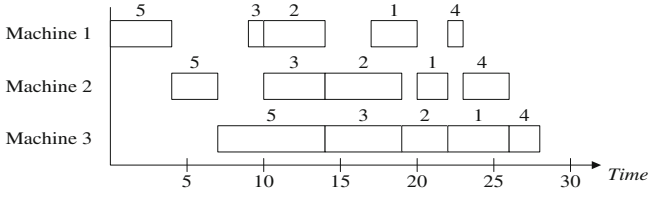

Figure 4. Gantt chart showing job sequence 5-3-2-1-4 for no intermediate storage.

Table 2. Performance comparison of different meta-heuristics.

		Opt /best		VNS		GASA		Tabu search		HPSO		HGA		Proposed GA	
Instance	n × m	known	RAJ	Min	% diff	Min	% diff	Min	% diff	Min	% diff	Min	% diff	Min	% diff
Problem 1	12 × 6	6921	–	–	–	–	–	–	–	–	–	–	–	6850	−1.03
Problem 2	7 × 5	683	–	–	–	–	–	–	–	–	–	–	–	565	−17.28
Problem 3	6 × 5	565	–	–	–	–	–	–	–	–	–	–	–	471	−16.64
car1	11 × 5	8142	8288	8201	0.72	8142	0	8142	0	8142	0	8142	0	8142	0
car2	13 × 4	8242	8610	8256	0.17	8242	0	8242	0	8242	0	8242	0	8242	0
car3	12 × 5	8866	9226	8866	0.00	8866	0	8866	0	8866	0	8866	0	8866	0
car4	14 × 4	9195	10,119	9348	1.66	9195	0	9195	0	9195	0	9195	0	9195	0
car5	10 × 6	9159	10,039	9496	3.68	9159	0	9159	0	9159	0	9159	0	9159	0
car6	8 × 9	9690	10,161	9690	0	9690	0	9690	0	9690	0	9690	0	9690	0
car7	7 × 7	7705	7903	7705	0	7705	0	7705	0	7705	0	7705	0	7705	0
car8	8 × 8	9372	9515	9372	0	9372	0	9372	0	9372	0	9372	0	9372	0
rec01	20 × 5	–	1590	1546	−2.77	1527	−3.96	1526	−4.03	1530	−3.77	1526	−4.03	1532	−3.65
rec03	20 × 5	–	1457	1394	−4.32	1392	−4.46	1361	−6.59	1361	−6.59	1361	−6.59	1364	−6.38
rec05	20 × 5	–	1637	1522	−7.03	1524	−6.90	1511	−7.70	1516	−7.39	1511	−7.70	1520	−7.15
rec07	20 × 10	–	2119	2070	−2.31	2046	−3.45	2042	−3.63	2042	−3.63	2042	−3.63	2088	−1.46
rec09	20 × 10	–	2141	2090	−2.38	2045	−4.48	2042	−4.62	2043	−4.58	2042	−4.62	2058	−3.88
rec11	20 × 10	–	1946	1916	−1.54	1881	−3.34	1881	−3.34	1881	−3.34	1881	−3.34	1906	−2.06

After delaying the start of job q after job p by duration $D(p, q)$, the schedule in figure 2 would be a no-wait schedule and is as given in figure 4.

6. Experimental analyses

To check the performance of the proposed GA, we took number of problems already published in the literature. The problems have been simulated on a Dual Core 2.88 GHz computer having 1 GB RAM. Initially, repeated test were conducted to find the best set of values for the GA parameters, i.e., crossover and mutation rate and population size. The best values for the crossover rate, mutation rate and population size were determined as 0.65, 0.09 and 60, respectively. Hence, for each run, same set of parameter settings have been used, which correspond to 80 sec on a Dual Core 2.88 GHz computer having 1 GB RAM.

The performance of the proposed GA approach was compared with the performances of meta-heuristic methods proposed by Schuster & Framinan (2003), Grabowski & Pempera (2005), Liu *et al* (2007) and Tseng & Lin (2010). Summary of the results makespan criterion is given in table 2. First three problems, i.e., Problems 1, 2 and 3 have been taken from Zhu *et al* (2008), Li & Wu (2008) and Li *et al* (2008), respectively. Eight problems with varying job and machine sizes (car1 to car8) have been taken from Carlier (1978) while six problems having 20 jobs (rec01, rec03, rec05, rec07, rec09 and rec11) have been taken from Reeves (1995). In table 2, 'Instance' represents the problem name, '$n \times m$' denotes the 'number of jobs and number of machines, 'Opt/Best Known' gives the makespan of the optimal solution of Carlier's benchmark problems, 'RAJ' represents the results by RAJ heuristic (Rajendran 1994), 'VNS' and 'GASA' represent the results of Schuster & Framinan (2003), 'Tabu Search' represents the results of Grabowski & Pempera (2005), 'HPSO' represents the results of Liu *et al* (2007), and 'HGA' represents the results of hybrid GA by Tseng & Lin (2010) while the results of the proposed approach are given in the last column. 'Min' represents the makespan of the best solution found. '% diff' represents the percentage relative difference $(C_{\min} - C^*)/C^* \times 100\%$, where C^* is the makespan of the known optimal solution given by Carlier (1978), however C^* is the makespan of the best solution found by RAJ heuristic (Rajendran 1994) for benchmark problems given by Reeves (1995).

It is observed from table 3 that the proposed approach found better solution than the previous approach for the first three problems. Similarly, it produced optimal solution for all eight Carlier (1978) benchmark problems. For Reeves (1995) benchmark problems, the proposed approach found better solutions than those found by RAJ heuristic (Rajendran 1994). The details of the simulation results are given in table 2.

Table 3. Performance comparison for different objective functions.

Prob	Reference	Objective function	Approach	Result	GA result
1	Wismer (1972)	Makespan	Branch and bound	42	42
2	van Deman & Baker (1974)	Average flow time	Branch and bound	28	28
3	Szwarc (1983)	Makespan	Gilmore-Gomory Algorithm (1964)	88	88
4	Rajendran & Chaudhri (1990)	Total flow time	Heuristic approach	503	501[a]
5	Rajendran (1994)	Makespan	Heuristic approach	20	20

[a]This is shown to be an optimal solution given by Rajendran & Chaudhri (1990)

Table 4. Performance comparison of different meta-heuristics for larger size problems.

			VNS		GASA		Tabu search		HPSO		HGA		Proposed GA	
Instance	$n \times m$	RAJ	Min	% diff	Min	% diff	Min	% diff	Min	% diff	Min	% diff	Min	% diff
rec13	20 × 15	2709	2553	−5.76	2556	−5.65	2545	−6.05	2545	−6.05	2545	−6.05	2565	−5.32
rec15	20 × 15	2691	2532	−5.91	2529	−6.02	2529	−6.02	2529	−6.02	2529	−6.02	2565	−4.68
rec17	20 × 15	2740	2599	−5.15	2590	−5.47	2587	−5.58	2587	−5.58	2587	−5.58	2599	−5.15
rec19	30 × 10	3157	2918	−7.57	2895	−8.30	2850	−9.72	2868	−9.15	2850	−9.72	2952	−6.49
rec21	30 × 10	3015	2888	−4.21	2948	−2.22	2823	−6.37	2843	−5.70	2829	−6.17	2891	−4.11
rec23	30 × 10	3030	2704	−10.76	2827	−6.70	2700	−10.89	2703	−10.80	2700	−10.89	2783	−8.15
rec25	30 × 15	3835	3626	−5.45	3732	−2.69	3593	−6.31	3616	−5.71	3593	−6.31	3662	−4.51
rec27	30 × 15	3655	3442	−5.83	3560	−2.60	3432	−6.10	3431	−6.13	3431	−6.13	3565	−2.46
rec29	30 × 15	3583	3324	−7.23	3440	−3.99	3291	−8.15	3303	−7.81	3291	−8.15	3383	−5.58
rec31	50 × 10	4631	4413	−4.71	4757	2.72	4343	−6.22	4357	−5.92	4334	−6.41	4528	−2.22
rec33	50 × 10	4770	4515	−5.35	4998	4.78	4466	−6.37	4507	−5.51	4458	−6.54	4711	−1.24
rec35	50 × 10	4718	4458	−5.51	4891	3.67	4427	−6.17	4434	−6.02	4424	−6.23	4634	−1.78
rec37	75 × 20	8979	8081	−10.00	9508	5.89	8127	−9.49	8181	−8.89	8121	−9.56	9515	8.16
rec39	75 × 20	9158	8671	−5.32	9964	8.80	8517	−7.00	8536	−6.79	8505	−7.13	9885	7.94
rec41	75 × 20	9344	8652	−7.41	9978	6.79	8520	−8.82	8602	−7.94	8505	−8.98	9888	5.82

In order to demonstrate the general purpose and customization feature of the proposed approach, five example problems with different objective function were taken from the literature to demonstrate this approach. A summary of the results is given in table 3. It is worth mentioning here that the problems were formulated within the same model as was built for finding the makespan. No change was made to either the spreadsheet model or the GA routine, thus emphasizing the general purpose feature of the proposed approach.

In tables 2 and 3 only small to medium-sized problems have been attempted. Varying the problem size allows us to access algorithm scalability, therefore we ran the algorithm on bigger size problems already published in the literature. Fifteen different problems with varying job and machine numbers were solved to see the performance of the proposed approach. All instances have been taken from Reeves (1995). The performance of the proposed method is compared with the same meta-heuristics as given in table 2. The results of the simulations for each of the instance are given in table 4.

From table 4 we can see that the proposed approach produces better solutions than RAJ heuristic (Rajendran 1994) except 75×20 problem size (instances rec37, rec39 and rec41), where the performance of the algorithm was even worse than RAJ heuristic (Rajendran 1994). Although the proposed approach does not get better solutions than the five meta-heuristics, keeping in view the general purpose nature of the proposed approach, familiar spreadsheet environment and the ability to tackle any objective function without changing the base algorithm, the performance of the algorithm can be said to be robust thus making it a general purpose scheduling approach. The robustness and general purpose nature of the algorithm has been the key advantage of the proposed approach.

7. Conclusions

In this paper, we considered the problem of scheduling jobs in a no-wait flowshop. The problem is known to be NP-hard. A general purpose GA methodology implemented in a spreadsheet environment was presented. The spreadsheet based GA approach has been found to be simple yet very effective to implement to cater for peculiarities of any shop environment. Furthermore, what if analysis can also be performed easily as most of the practitioners are familiar with the spread sheet environment. The spread sheet model is easily customizable to include additional workers, machines or jobs without changing the logic of the GA routine thus making it a general purpose scheduling approach. We also demonstrate that any objective function can be used without changing the logic of the GA routine.

The proposed approach was able to find optimal or near optimal solution for all the problems with different objective function thus highlighting the robustness of the proposed GA approach. The experimental results reveal that the proposed approach has the capability to solve large problems with a reasonable accuracy.

References

Aldowaisan T and Allahverdi A 2003 New heuristics for no-wait flowshops to minimize makespan. *Comput. Oper. Res*. 30(8): 1219–1231

Aldowaisan T and Allahverdi A 2004 New heuristics for m-machine no-wait flowshop to minimize total completion time. *Omega-Int. J. Manage. Sys*. 32(5): 345–352

Bonney M C and Gundry S W 1976 Solution to the constrained flowshop sequencing problem. *Oper. Res. Q*. 27(4): 869–883

Bożejko W and Makuchowski M 2009 A fast hybrid tabu search algorithm for the no-wait job shop problem. *Comput. Ind. Eng*. 56(4): 15029–1509

Carlier J 1978 Ordonnancements a contraintes disjonctives. *Rairo-Rech Oper*. 12(4): 333–351

Chaudhry I A 2009 Minimizing flow time for the worker assignment problem in identical parallel machine models using GA. *Int. J. Adv. Manuf. Technol*. 48(5–8): 747–760

Chaudhry I A and Drake P R 2008 Minimizing total tardiness for the machine scheduling and worker assignment problems in identical parallel machines using genetic algorithms. *Int. J. Adv. Manuf. Technol*. 42(5): 581–594

Davis L 1985 Job Shop Scheduling with Genetic Algorithms. J J Grefenstette (eds) In: *Proceedings of the 1st International Conference on Genetic Algorithms*, Pittsburgh, USA: PA, Lawrence Erlbaum Associates, p. 136–140

Davis L 1991 *Handbook of genetic algorithms*, New York, USA: Van Nostrand Reinhold

Evolver (1998) *The genetic algorithm super solver*, Version 4, New York, USA: Palisade Corporation

Framinan J M and Nagano M S 2008 Evaluating the performance for makespan minimization in no-wait flowshop sequencing. *J. Mater. Process. Tech*. 197(1–3): 1–9

Gangadharan R and Rajendran C 1993 Heuristic algorithms for scheduling in the no-wait flowshop. *Int. J. Prod. Econ*. 32(3): 285–290

Garey M R and Johnson D S 1979 *Computers and intractability: A guide to the theory of NP-completeness*, San Francisco, CA, USA: Freeman

Gilmore P C and Gomory R E 1964 Sequencing a one-state variable machine: A solvable case for the travelling salesman problem. *Oper Res*. 12(5): 655–679

Goldberg D E 1989 *Genetic algorithms in search, optimization and machine learning*. Boston, USA: Addison-Wesley

Grabowski J and Pempera J 2005 Some local search algorithms for no-wait flow-shop problem with makespan criterion. *Comput. Oper. Res*. 32(8): 2197–2212

Hall N G and Sriskandarajah C 1996 A survey of machine scheduling problems with blocking and no-wait in process. *Oper. Res*. 44(3): 510–525

Hejazi S R and Saghafian S 2005 Flowshop-scheduling problems with makespan criterion: a review. *Int. J. Prod. Res*. 43(14): 2895–2929

Holland J 1975 *Adaptation in natural and artificial systems*. USA: University of Michigan Press

Jarboui B, Eddaly M and Siarry P 2011 A hybrid genetic algorithm for solving no-wait flowshop scheduling problems. *Int. J. Adv. Manuf. Technol*. 54(9–12): 1129–1143

Kalczynski P J and Kamburowski J 2007 On no-wait and no-idle flow shops with makespan criterion. *Eur. J. Oper. Res*. 178(3): 677–685

King J R and Spachis A S 1980 Heuristics for flow-shop scheduling. *Int. J. Prod. Res*. 18(3): 345–357

Laha D and Chakraborty U K (2009) A constructive heuristic for minimizing makespan in no-wait flow shop scheduling. *Int. J. Adv. Manuf. Technol*. 41(1–2): 97–109

Li X and Wu C 2008 Heuristic for no-wait flow shops with makespan minimization based on total idle-time increments. *Sci. China Ser. F*. 51(7): 896–909

Li X, Wang Q and Wu C 2008 Heuristic for no-wait flow shops with makespan minimization. *Int. J. Prod. Res*. 46(9): 2519–2530

Liaw C-F 2008 An efficient simple metaheuristic for minimizing the makespan in two-machine no-wait job shops. *Comput. Oper. Res*. 35(10): 3276–3283

Liu B, Wang L and Jin Y-H 2007 An effective hybrid particle swarm optimization for no-wait flowshop scheduling. *Int. J. Adv. Manuf Technol*. 31(9–10): 1001–1011

Nanvala H 2011 Use of Genetic algorithm based approaches in scheduling of FMS: A Review. *Int. J. Eng. Sci. Technol*. 3(3): 1936–1942 [open access journal: http://www.ijest.info/docs/IJEST11-03-03-164.pdf]

Pan Q-K, Tasgetiren M F and Liang Y-C 2008a A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. *Comput. Oper. Res*. 35(9): 2807–2839

Pan Q-K, Wang L, Tasgetiren M F and Zhao B-H 2008b A hybrid discrete particle swarm optimization algorithm for the no-wait flow shop scheduling problem with makespan criterion. *Int. J. Adv. Manuf. Technol*. 38(3–4): 337–347

Pan Q-K, Wang L and Zhao B-H 2008c An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion. *Int. J. Adv. Manuf. Technol*. 38(7–8): 778–786

Piehler J 1960 Ein Beitrag Zum Reihenfolgeproblem. *Unternehmensforschung*. 4: 138–142 (in German)

Qian B, Wang L, Hu R, Huang D X and Wang X 2009 A DE-based approach to no-wait flow-shop scheduling. *Comput. Ind. Eng*. 57(3): 787–809

Rajendran C 1994 A no-wait flowshop scheduling heuristic to minimize makespan. *J. Oper. Res. Soc*. 45(4): 472–478

Rajendran C and Chaudhri D 1990 Heuristic for continuous flow-shop problem. *Nav. Res. Log*. 37(5): 695–705

Reddi S S and Ramamoorthy C V 1972 On the flowshop sequencing problem with no-wait in process. *Oper. Res. Q*. 23(3): 323–331

Reeves C R 1995 A genetic algorithm for flowshop sequencing. *Comput. Oper. Res*. 22(1): 5–13

Rock H 1984 The three-machine no-wait flowshop problem is NP-complete. *J. Assoc. Comput. Mach*. 31(2): 336–345

Schuster C J and Framinan J M 2003 Approximative procedures for no-wait job shop scheduling. *Oper. Res. Lett*. 31(4): 308–318

Szwarc W 1983 Solvable cases of the flow-shop problem without interruptions in job processing. *Nav. Res. Log Q*. 30(1): 179–183

Tseng L-Y and Lin Y-T 2010 A hybrid genetic algorithm for no-wait flowshop scheduling problem. *Int. J. Prod. Econ*. 128(1): 144–152

van Deman J M and Baker K R 1974 Minimizing mean flowtime in the flow shop with no intermediate queues. *AIIE Trans*. 6(1): 28–34

Wismer D A 1972 Solution of the flowshop-scheduling problem with no intermediate queues. *Oper. Res*. 20(3): 689–697

Zhu X, Li X and Wang Q 2008 Hybrid heuristic for m-machine no-wait flowshops to minimize total completion time. W Shen, J Luo, Z Lin, J-PA Barthès, Q Hao, (eds) In: *Proceedings of the 11th International Conference on Computer Supported Cooperative Work in Design*, LNCS 5236, Melbourne, Australia, p. 192–203