An improved heuristic for permutation flowshop scheduling

Uday Kumar Chakraborty*

Department of Mathematics and Computer Science,
University of Missouri,
St. Louis, MO 63121, USA
Fax: +1-314-516-5400
E-mail: uday@cs.umsl.edu
*Corresponding author

Dipak Laha

Department of Mechanical Engineering,
Jadavpur University,
Calcutta 700032, India
E-mail: dipaklaha_jume@yahoo.com

Abstract: Flowshop scheduling deals with determination of optimum sequence of jobs to be processed on some machines in a fixed order so as to satisfy certain scheduling criteria. The general problem of scheduling has been shown to be NP-complete. Exact algorithms, such as integer programming and branch-and-bound, guarantee optimality but do not yield the optimum solution in polynomial time even for problems of small size. Heuristics have been shown to yield good working solutions (not necessarily optimal) in reasonable time. Although much research on the flowshop problem has been done over several decades starting from Johnson's algorithm, only a few good algorithms exist. The Nawaz-Enscore-Ham heuristic, used for minimisation of makespan, continues to be the most popular algorithm because of its simplicity, solution quality and time-complexity. In the present paper we have modified the NEH algorithm, achieving significant improvement in the quality of the solution while maintaining the *same* algorithmic complexity. The proposed approach derives its strength from the use of a population-based technique. Experimental comparisons have been made on a large number of randomly generated test problems of varying problem sizes. Our approach is shown to outperform both the original NEH and NEH's best-known competitor to date, the HFC heuristic. Statistical tests of significance are performed to substantiate the claims of improvement.

Keywords: flowshop scheduling; heuristics; makespan.

Reference to this paper should be made as follows: Chakraborty, U.K. and Laha, D. (2007) 'An improved heuristic for permutation flowshop scheduling', *Int. J. Information and Communication Technology*, Vol. 1, No. 1, pp.89–97.

Biographical notes: Uday Kumar Chakraborty is an Associate Professor of Computer Science at the University of Missouri, St. Louis and has previously held positions at CMC Limited (India), Jadavpur University (India) and

GMD Forschungszentrum Informationstechnik (Germany). His research interests include evolutionary computation, soft computing, heuristics and computer graphics. He has (co)authored one book and over 70 papers. He is an Area Editor of *New Mathematics and Natural Computation* and an Editor of the *Journal of Computing and Information Technology*. He has guest-edited special issues of the *Journal of Systems Architecture* and *Information Sciences*.

Dipak Laha is a Reader in the Department of Mechanical Engineering at Jadavpur University, Calcutta, India. He received a BE from Bengal Engineering College, Shibpur, an MTech from the Indian Institute of Technology, Kharagpur and a PhD from Jadavpur University. His research interests are in the areas of manufacturing scheduling, evolutionary optimisation and artificial intelligence.

1 Introduction: the problem

The problem of the assignment of times to a set of jobs for processing through a series of machines has long received the attention of researchers. A great deal of research has been carried out in manufacturing scheduling. The practical importance of such problems is great, as scheduling plays a significant role in successful production, planning and control. A variety of scheduling algorithms have been developed over the past years to address different production systems. Two common problems that appear regularly in the scheduling literature of the past 40 years are flowshop scheduling and jobshop scheduling. In flowshop scheduling it is generally assumed that the jobs must be processed on the machines in the same technological or machine order. In jobshop scheduling, on the other hand, jobs are usually processed following different machine orders.

In the flowshop scheduling problem, n jobs are to be processed on m machines. The order of the machines is fixed. We assume that a machine processes one job at a time and a job is processed on one machine at a time without preemption. Let $t_p(i, j)$ denote the processing time of job j on machine i and $t_c(i, j)$ denote the completion time of job j on machine i. Let J_j denote the jth job and M_i be the ith machine. The completion times of the jobs are obtained as follows. For $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$.

$$t_c(M_1, J_1) = t_p(M_1, J_1)$$
$$t_c(M_i, J_1) = t_c(M_{i-1}, J_1) + t_p(M_i, J_1)$$
$$t_c(M_1, J_j) = t_c(M_1, J_{j-1}) + t_p(M_1, J_j)$$
$$t_c(M_i, J_j) = \max\{t_c(M_{i-1}, J_j), t_c(M_i, J_{j-1})\} + t_p(M_i, J_j)$$

Makespan is defined as the completion time of the last job, that is, makespan is denoted by $t_c(M_m, J_n)$. We obtain the n-job sequence that minimises the makespan.

The search space consists of $n!$ possible job sequences. The problem is NP-complete and exhaustive enumeration of all $n!$ sequences is computationally prohibitive.

In the present paper we present an efficient deterministic heuristic for solving the n-job, m-machine flowshop scheduling problem. The algorithm seeks to improve upon the famous NEH (Nawaz et al., 1983) method. The remainder of this paper is organised as follows: in Section 2 we provide the background information necessary to understand the present paper, Section 3 explains why the NEH algorithm continues to be important till today, Section 4 discusses the proposed approach, Section 5 derives the algorithmic complexity, Section 6 presents comparative experimental results and finally, Section 7 presents the conclusions.

2 Previous work

Currently available heuristics for solving flowshop scheduling problems can be broadly divided into two categories: constructive heuristics and improvement heuristics. A constructive heuristic generates a schedule of jobs so that once a decision is taken it cannot be changed for improvement. An improvement heuristics starts with an initial sequence of jobs and an attempt is made to improve the objective function by amending the sequence. The scheduling approach generally cited as the foundation technique is the one developed by Johnson (1954) who presented a simple constructive method to minimise the makespan for n-job, two-machine scheduling problems. In Johnson's method, job J_i is scheduled first or last according to whether t_{1i} or t_{2i} is $\min\{t_{ji}\}$. After removing the scheduled job from the list, the next job is scheduled to the next available position (starting at the beginning or at the end of the schedule) following the same $\min\{t_{ji}\}$ criterion. The process continues till the last job is included. The simplicity and guaranteed optimality of Johnson's algorithm led many researchers to extend his idea to the general case of n-jobs, m-machine problems, but without success. Palmer (1965) proposed a solution to the general (n, m) problem by computing a slope index to give priorities to jobs to proceed from one machine to another and then sequencing the jobs in descending order of the slope index. Campbell et al. (1970) (or CDS for short) proposed a generalisation of Johnson's method. They developed $m - 1$ artificial two-machine problems from the original m-machine problem and solved them using Johnson's algorithm. The best sequence is taken as the final solution. Dannenbring (1977), too, used Johnson's algorithm and proposed two neighbourhood schemes to be implemented for exploring (possibly) improved solutions. With the 'rapid access with close order search (RACS)' neighbourhood, $n - 1$ new sequences are examined based on adjacent job interchanges and the best one is selected; the other neighbourhood, the 'rapid access with extensive search (RAES)', uses the best immediate neighbour and was shown to be superior to RACS. Ignall and Schrage (1965) used branch-and-bound to develop an optimisation algorithm for three-machine flowshop problems. Other examples of the use of the branch-and-bound technique include Lomnicki (1965), Brown and Lomnicki (1966) and Bestwick and Hastings (1976).

Nawaz et al. (1983) proposed a heuristics (refereed to as the NEH) that gave priority to jobs with large total processing times. A heuristic method based on minimising the idle time of the last machine was proposed by Sarin and Lefoka (1993). Koulamas (1998) presented a simple constructive heuristics capable of producing non-permutation schedules. The algorithm works in two phases: the first phase generates a permutation schedule which then is fed to the second phase; the second phase improves upon the sequence obtained in the first phase by generating a non-permutation schedule.

3 Relevance of NEH

Despite the existence of a plethora of flowshop scheduling heuristics, the NEH method continues to be the best constructive heuristic method because of its simplicity, solution quality and time complexity. Johnson's two-machine scheme gives the optimal makespan, but fails to generalise to m-machine problems. Park's (1981) study, comparing CDS, NEH and other heuristics, demonstrates that NEH is the 'least biased and best-operated' of the heuristics tested on both small-sized ($n = 3 - 9$, $m = 4 - 20$) and medium-sized ($n = 15 - 30$, $m = 4 - 20$) problems and that the CDS comes next. As far as computation time is concerned, for large problems NEH outperformed the CDS but was outperformed by the Gupta (1971) algorithm (the NEH times were not unacceptably large, though). Nawaz et al. argue that their algorithm will continue to perform well for large problem sizes ($m, n \geq 100$). They also point out that for large problems where the number of machines greatly exceeds the number of jobs, CDS is likely to outperform NEH, because the former's effectiveness is dependent on the number of machines while the latter's on the number of jobs. Now, Park's study did not include Dannenbring's (1977) work and as mentioned by Turner and Booth (1987), Dannenbring's RAES is superior to CDS. Turner and Booth also observed that NEH proved to be more efficient than RAES on both measures of performance (makespan and CPU time). Sarin and Lefoka (SL) have shown that NEH is more effective than their SL when the number of machines is ≤ 100 but is inferior for larger m.

4 The proposed algorithm

The proposed algorithm builds the n-job sequence incrementally and thus is a constructive method. What leads to its improved performance is its use of a group of promising partial solutions at each stage (i.e. as each new job is added to the sequence). Here is an outline of the method:

1 For each job i, find the total processing time T_i which is given by

$$T_i = \sum_{j=1}^{m} t_p(j, i)$$

where $t_p(j, i)$ is the processing time of job i on machine j.

2 Sort the n jobs on descending order of their total processing times.

3 Take the first four jobs from the sorted list and form $4! = 24$ partial sequences (each of length 4). The best k (k is a parameter of the algorithm) out of these 24 partial sequences are selected for further processing. The relative positions of jobs in any partial sequence is not altered in any later (larger) sequence.

4 Set $z = 5$.

5 The zth job on the sorted list is inserted at each of the z positions in each of the k $(z - 1)$-job partial sequences, resulting in $z \times k$ z-job partial sequences.

6 The best k out of the $z \times k$ sequences are selected for further processing.

7 Increment z by 1.

8 If $z > n$, accept the best of the k n-job sequences as the final solution and stop. Otherwise go to step 5.

5 Computational complexity

In this section we derive the algorithmic complexity of the proposed scheme. Step 1 of our algorithm computes a sum of m terms for each of the n jobs and is thus of complexity

$\Theta(mn)$. Step 2 involves sorting n items and can be implemented using any good algorithm from the literature. Quicksort (Cormen et al., 1990), with an average-case complexity of $\Theta(n \lg n)$, is a natural choice. Step 3 takes a constant amount of time. Steps 5 – 8 together take a total time given by

$$\sum_{z=5}^{n} k \times z \times \mathrm{TMS}(z)$$

where $\mathrm{TMS}(z)$ denotes the time to compute the makespan for a z-job partial sequence.

For a direct comparison with NEH, we note that the total number of enumerations (of partial and complete sequences) in the present method is

$$4! + \sum_{z=5}^{n} k \times z = 4! + k \times \sum_{z=5}^{n} z$$
$$= \Theta(n^2)$$

The number of enumerations in NEH was shown (Nawaz et al., 1983) to be

$$\frac{n(n+1)}{2} - 1$$

which, clearly, is $\Theta(n^2)$. Thus the asymptotic time complexity of our method is the same as that of NEH.

6 Experimental results

The algorithm was run on 28 different problem sizes ($n = 12, 18, 24, 30, 40, 50, 100$ and $m = 5, 10, 15, 20$). For each problem size, 15 independent problem instances were created. Each problem instance corresponds to a new t_p matrix: each processing time ($t_p(.,.)$ value) was independently obtained from a uniform random u(1,99) discrete distribution. Table 1 shows makespan values (averaged over 15 independent instances) obtained by the original NEH and the proposed algorithm for two values of k: 6 and 24. These values of k are only representative. We did not attempt any tuning for the parameter k. These results bring out the superiority of the proposed approach.

Tables 2 and 3 show results of statistical tests of significance for two separate cases ($k = 6$ and $k = 24$). Each test suite (recall that one test suite comprises 15 independent instances) gives us 15 pairs of makespan values and we thus have a paired comparison. For each test suite, the mean and the standard deviation of the 15 differences in makespan are easily obtained. The difference in each instance is obtained by subtracting the makespan of the proposed scheme from the NEH makespan. We now test the hypothesis that the population corresponding to the differences has mean, μ, zero. Specifically, we test the (null) hypothesis $\mu = 0$ against the alternative $\mu > 0$. We assume that the makespan difference is a normal random variable, and choose the significance level $\alpha = 0.5$. If the hypothesis is true, the random variable

$$t = \sqrt{N}\frac{\bar{x} - \mu}{s}$$

has a t-distribution with $N - 1$ degrees of freedom (Kreyszig, 1972). The critical value c is obtained from the relation

Table 1 Performance comparison between NEH and the proposed heuristic (H)

Test suite #	*# of jobs*	*# of machines*	*# of instances*	*Average makespan*		
				NEH	*Proposed method*	
					H (k = 6)	*H(k = 24)*
1	12	5	15	838	834	831
2	12	10	15	1215	1201	1189
3	12	15	15	1529	1498	1490
4	12	20	15	1904	1888	1866
5	18	5	15	1142	1132	1127
6	18	10	15	1540	1527	1521
7	18	15	15	1843	1829	1810
8	18	20	15	2196	2161	2151
9	24	5	15	1440	1440	1438
10	24	10	15	1863	1859	1849
11	24	15	15	2173	2147	2126
12	24	20	15	2530	2492	2477
13	30	5	15	1819	1816	1811
14	30	10	15	2149	2129	2116
15	30	15	15	2566	2537	2519
16	30	20	15	2892	2842	2823
17	40	5	15	2274	2272	2272
18	40	10	15	2689	2673	2657
19	40	15	15	3125	3076	3052
20	40	20	15	3459	3419	3394
21	50	5	15	2817	2814	2814
22	50	10	15	3157	3135	3115
23	50	15	15	3615	3595	3572
24	50	20	15	4040	3975	3946
25	100	5	15	5473	5470	5470
26	100	10	15	5820	5796	5794
27	100	15	15	6243	6209	6193
28	100	20	15	6660	6593	6580

$$\text{Prob}(t > c) = \alpha = 0.05$$

From the standard tables of t-distribution, we have for 14 degrees of freedom $c = 1.76$. For example, the first entry in Table 1 corresponds to sample size = $N = 15$, $\mu = 0$, sample mean = $\bar{x} = 3.467$, sample standard deviation = $s = 5.678$ and the sample $t = \sqrt{15}(3.467 - 0)/5.678 = 2.33$. Since $t > 1.76$, we conclude that the difference is statistically significant.

Two additional metrics have been used to quantify the improvement. These are the Average Relative Percentage Deviation (ARPD) and the Maximum Percentage Deviation (MPD). These metrics are defined as follows (MS in the equations below stands for makespan and H represents the proposed heuristic):

$$\text{ARPD}_{\text{NEH}} = \frac{100}{15} \sum_{i=1}^{15} \frac{\text{MS}_{\text{NEH},i} - \min(\text{MS}_{\text{NEH},i}, \text{MS}_{H,i})}{\min(\text{MS}_{\text{NEH},i}, \text{MS}_{H,i})}$$

$$\mathrm{ARPD}_H = \frac{100}{15}\sum_{i=1}^{15}\frac{\mathrm{MS}_{H,i} - \min(\mathrm{MS}_{\mathrm{NEH},i}, \mathrm{MS}_{H,i})}{\min(\mathrm{MS}_{\mathrm{NEH},i}, \mathrm{MS}_{H,i})}$$

$$\mathrm{MPD}_{\mathrm{NEH}} = \max_i \left\{\frac{\mathrm{MS}_{\mathrm{NEH},i} - \min(\mathrm{MS}_{\mathrm{NEH},i}, \mathrm{MS}_{H,i})}{\min(\mathrm{MS}_{\mathrm{NEH},i}, \mathrm{MS}_{H,i})}\right\} \times 100$$

$$\mathrm{MPD}_H = \max_i \left\{\frac{\mathrm{MS}_{H,i} - \min(\mathrm{MS}_{\mathrm{NEH},i}, \mathrm{MS}_{H,i})}{\min(\mathrm{MS}_{\mathrm{NEH},i}, \mathrm{MS}_{H,i})}\right\} \times 100$$

Clearly, the best possible performance corresponds to both ARPD and MPD being zero.

The experimental results show that our method is superior to NEH. To our knowledge, the HFC heuristic (Koulamas, 1998) is the only competitor of NEH to date, but as the author of HFC admits, the HFC is no better than NEH for permutation flowshop problems (HFC is better than NEH only for non-permutation problems). Thus the proposed heuristics is better than HFC, too.

Table 2 Results of statistical tests. The proposed method H is run with $K = 6$

Test suite #	*Difference in 15 instances*		*t-statistic*	*ARPD*		*MPD*	
	Mean	*SD*		*NEH*	*H(K = 6)*	*NEH*	*H(K = 6)*
1	3.467	5.768	2.33	0.533	0	2.179	0
2	13.867	16.22	3.31	1.199	0	3.715	0
3	30.733	23.903	4.98	2.067	0	5.181	0
4	15.267	45.74	1.3	1.361	0.187	4.311	1.371
5	9.867	19.96	1.91	1.133	0.215	4.785	3.231
6	13.2	32.57	1.57	0.958	0.399	2.971	4.196
7	14.4	22.79	2.45	1.001	0.199	3.231	2.071
8	34.67	20.72	6.48	1.603	0	3.555	0
9	0.67	18.33	0.14	0.412	0.162	2.557	2.424
10	4.4	31.49	0.54	0.169	0.301	0.724	2.857
11	25.67	25.54	3.89	1.232	0.048	3.294	0.588
12	38.6	26.2	5.70	1.602	0.051	3.203	0.756
13	3	16.36	0.71	0.378	0.181	2.617	1.327
14	20.07	29.7	2.62	1.028	0.089	3.212	0.883
15	28.8	34.91	3.195	1.372	0.214	3.496	2.045
16	50	20.92	9.26	1.766	0	3.130	0
17	2	10.198	0.76	0.184	0.088	1.376	0.545
18	15.53	20.07	2.3	0.505	0.022	2.33	0.259
19	48.4	48.16	3.89	1.521	0.128	3.725	1.085
20	40.07	29.76	5.21	1.181	0.025	2.131	0.179
21	2.8	8.53	1.27	0.109	0.007	1.204	0.072
22	21.93	35.636	2.38	0.889	0.169	2.314	1.767
23	19.667	43.747	1.74	0.795	0.245	2.846	2.199
24	64.4	53.91	4.63	1.711	0.077	3.824	0.571
25	3.13	7.17	1.69	0.063	0.005	0.418	0.073
26	24.333	38.565	2.44	0.512	0.075	1.771	0.733
27	33.467	39.599	3.036	0.587	0.048	1.698	0.381
28	66.2	63.55	4.03	1.066	0.068	2.308	0.623

Table 3 Results of statistical tests. The proposed heuristic (H) is run with $k = 24$

Test suite #	*Difference in 15 instances*		*t-statistic*	*ARPD*		*MPD*	
	Mean	*SD*		*NEH*	*H(K = 24)*	*NEH*	*H(K = 24)*
1	6.8	7.103	3.71	0.853	0	2.570	0
2	26.6	20.378	5.05	2.255	0	6.269	0
3	38.733	22.176	6.76	2.611	0	5.181	0
4	37.4	27.92	5.19	2.061	0.061	4.311	0.914
5	14.93	14.14	4.09	1.348	0	4.175	0
6	19.33	24.77	3.02	1.522	0.256	3.226	3.636
7	33.27	15.97	8.07	1.844	0	3.914	0
8	45.2	25.26	6.93	1.798	0.003	3.554	0.046
9	1.667	25.68	0.25	0.685	0.529	2.557	3.949
10	14.267	28.767	1.92	0.986	0.175	5.242	1.554
11	47	23.71	7.68	2.197	0	4.325	0
12	52.93	21.96	9.335	1.837	0	3.955	0
13	8	13.62	2.27	0.479	0.037	2.617	0.549
14	33.4	42.998	3.01	1.905	0.314	4.182	4.708
15	46.73	45.55	3.97	1.505	0.166	3.681	2.045
16	68.867	15.343	17.38	2.152	0	3.016	0
17	2.733	7.55	1.4	0.139	0.008	1.376	0.128
18	31.267	39.845	3.04	1.353	0.143	4.151	1.883
19	72.4	39.37	7.12	2.441	0.049	3.638	0.745
20	65.33	26.95	9.39	1.804	0	3.262	0
21	2.867	8.76	1.26	0.119	0.014	1.204	0.107
22	41.93	31.92	3.17	1.386	0.021	2.661	0.321
23	42.867	44.25	3.752	1.342	0.150	2.817	2.144
24	86.87	44.33	7.59	2.374	0	3.681	0
25	2.8	7.37	1.47	0.073	0.007	0.607	0.110
26	25.267	38.98	2.51	0.556	0.095	1.658	0.887
27	49.667	53.28	3.61	0.794	0.096	2.008	0.842
28	79.53	58.51	5.26	1.034	0.028	2.415	0.341

7 Conclusions

This paper presented a new deterministic heuristic for permutation flowshop scheduling by modifying the classic NEH algorithm (Nawaz et al., 1983). The proposed algorithm is elegant, easy to implement, yields solutions of better quality than NEH does and yet has the *same* order of computational complexity as NEH. Our results have been shown to be statistically significantly better than those produced by the best deterministic method known to date. Given the fact that the NEH is still the best deterministic heuristic for this class of problems and that the proposed method outperforms NEH, our algorithm should serve as a framework for further research into permutation flowshop sequencing.

Acknowledgement

The contribution of the first author was supported in part by a UMSL Research Award.

References

Bestwick, P.F. and Hastings, N.A.J. (1976) 'A new bound for machine scheduling', *Operational Research Quarterly*, Vol. 27, pp.479–490.

Brown, A.P.G. and Lomnicki, Z.A. (1966) 'Some applications of the branch and bound algorithm to the machine scheduling problem', *Operational Research Quarterly*, Vol. 17, pp.173–182.

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, pp.630–637.

Cormen, T.H., Leiserson, C.E. and Rivest, R.L. (1990) 'Introduction to Algorithms', Cambridge, MA: MIT Press.

Dannenbring, D.G. (1977) 'An evaluation of flowshop sequencing heuristics', *Management Science*, Vol. 23, No. 11, pp.1174–1182.

Gupta, J.N.D. (1971) 'A functional heuristic algorithm for the flow-shop scheduling problem', *Operational Research Quaterly*, Vol. 22, No. 1.

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.

Johnson, S.M. (1954) 'Two and three stage production schedules with set-up times included', *Naval Research Logistics Quarterly*, Vol. 1, pp.61–68.

Koulamas, C. (1998) 'A new constructive heuristic for the flowshop scheduling problem', *European Journal of Operations Research*, Vol. 105, pp.66–71.

Kreyszig, E. (1972) 'Advanced Engineering Mathematics', New York: John Wiley.

Lomnicki, Z.A. (1965) 'A branch-and-bound algorithm for the exact solution of the three-machine scheduling problem', *Operational Research Quarterly*, Vol. 16, pp.89–100.

Nawaz, M., Enscore Jr., E.E. and Ham, I. (1983) 'A heuristic algorithm for the m-machine n-job flowshop sequencing problem', *OMEGA International Journal of Management Science*, Vol. 11, pp.91–95.

Palmer, D.S. (1965) 'Sequencing jobs through a multi-stage process in the minimum total time - a quick method of obtaining a near-optimum', *Operational Research Quarterly*, Vol. 16, No. 1, pp.101–107.

Park, Y.B. (1981) 'A simulation study and an analysis for evaluation of performance-effectiveness of flowshop sequencing heuristics: a static and dynamic flowshop model', Master's Thesis, Pennsylvania State University.

Sarin, S. and Lefoka, M. (1993) 'Scheduling heuristics for the n-job, m-machine flowshop', *OMEGA*, Vol. 21, pp.229–234.

Turner, S. and Booth, D. (1987) 'Comparison of heuristics for flowshop sequencing', *OMEGA*, Vol. 15, pp.75–78.