An Improved Heuristic for Permutation Flow Shop Scheduling (NEH ALGORITHM)

[1]Ekta Singhal, [2]Shalu Singh, [3]Aneesh Dayma
(Department of Software Engineering),
[3] (Department of Computer Science), Suresh Gyan Vihar University,Jaipur

Abstract- Flowshop Scheduling is used to determine the optimal sequence of n jobs to be processed on m machines in the same order. Permutation Flowshop Scheduling Problems (PFSP) require same job sequence on all the machines with the constraint that machines can only process one job at a time and jobs can be processed by only one machine at a time. No machine is allowed to remain idle when a job is ready for processing. Such problems are NP-Complete and hence optimal solutions are not guaranteed but heuristics have been shown to produce good working solutions.

NEH (Nawaz, Enscore, Ham) Algorithm is an efficient algorithm that works by minimizing the makespan for Permutation Flowshop Scheduling Problems PFSP. The proposed algorithm is obtained by modifying the NEH algorithm and produces improved quality solutions with algorithmic complexity same as the original algorithm.

***Keywords*:** Flowshop Scheduling, heuristics, makespan

I. Introduction

1.1 Important Definitions

1.1.1 Heuristics

Heuristic refers to experience-based techniques for problem solving, learning, and discovery. Heuristic methods are used to speed up the process of finding a good enough solution, where an exhaustive search is impractical[6].

1.1.2 Flowshop Scheduling

Flowshop Scheduling determines an optimum sequence of n jobs to be processed on m machines in the same order i.e. every job must be processed on machines *1,2,...,m* in this same order[10].

1.1.3 Permutation Flowshop Scheduling

Permutation Flowshop Scheduling is a special case of FSPs where same job sequence is followed in all machines i.e. processing order of the jobs on the machines is the same for every machine.

1.1.4 NP-Complete

A problem *L* is NP-complete if it has two properties:

- It is in the set of NP (nondeterministic polynomial time) problems: Any given solution to *L* can be *verified* quickly (in polynomial time).
- It is also in the set of NP-hard problems: Any NP problem can be converted into *L* by a transformation of the inputs in polynomial time. Although any given solution to such a problem can be verified quickly, there is no known efficient way to locate a solution in the first place; indeed, the most notable characteristic of NP-complete problems is that no fast solution to them is known. That is, the time required to solve the problem using any currently known algorithm increases very quickly as the size of the problem grows.

1.1.5 Makespan

Makespan is the completion time of last job on last machine.

1.1.6 Constructive Heuristics

In constructive heuristics once a decision is taken it cannot be changed for improvement[10].

1.1.7 Improvement Heuristics

Pawel J. Kalczynski, Jerzy Kamburowski [3] used improvement heuristics, we start with an initial sequence and attempt to improve it as we proceed.

1.2 Problem Definition

The permutation flow shop problem requires to find the order in which *n* jobs are to be processed on *m* consecutive machines. The jobs are processed in the order machine 1, machine 2, . . . machine *m*.

Machines can only process one job at a time and jobs can be processed by only one machine at a time without preemption.

No job can jump over any other job, meaning that the order in which jobs are processed in machine 1 is maintained throughout the system. Moreover, no machine is allowed to remain idle when a job is ready for processing. All jobs and machines are available at time 0.

II. OBJECTIVE

For each job two parameters are computed:
tp (i, j) processing time of job j on machine i
tc (i, j) completion time of job j on machine i

The completion time of all jobs is can be computed as:

tc (M1, J1) = tp (M1, J1)
tc (Mi, J1) = tc (Mi-1, J1) + tp (Mi, J1)
tc (M1, Jj) = tc (M1, Jj-1) + tp (M1, Jj)
tc (Mi, Jj) = max {tc (Mi-1, Jj), tc (Mi, Jj-1)} + tp (Mi, Jj)

The objective is to find an n-job sequence so as to minimize the makespan i.e. tc (Mm, Jn).

III. Neh Algorithm (Nawaz Enscore Ham)

It is a constructive heuristic.
Step 1: Sort the *n* jobs in non-increasing order of their total processing times

Step 2: Take the first two jobs and schedule them in order to minimise the partial makespan as if there were only these two jobs

Step 3: For *k*= 3 to *n* do Step 4

Step 4: Insert the *k*th job at the place, which minimises the partial makespan among the *k* possible ones.

IV. Improved Heuristic

Step 1: Sort the *n* jobs in non-increasing order of their total processing times

***Step 2*:** 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

***Step 3*:** Set *z* = 5

***Step 4*:** The *z*th 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* × *k*) *z*-job partial sequences

***Step 5*:** The best *k* out of the *z* × *k* sequences are selected for further processing

***Step 6*:** Increment *z* by 1

***Step 7*:** If *z* > *n*, accept the best of the *k* *n*-job sequences as the final solution and stop.

Otherwise go to step 4.

V. Comparision (EXAMPLE)

Comparison (Example)

	Machine 1	Machine 2	Machine 3	Machine 4	Machine 5
Job 1 (J1)	5	9	8	10	1
Job 2 (J2)	9	3	10	1	8
Job 3 (J3)	9	4	5	8	6
Job 4 (J4)	4	8	8	7	2
Job 5 (J5)	3	5	6	3	7

Total processing times of jobs
Job 1= 5+9+8+10+1= 33
Job 2= 9+3+10+1+8= 31
Job 3= 9+4+5+8+6= 32
Job 4= 4+8+8+7+2= 29
Job 5= 3+5+6+3+7= 24

Sorting in non-increasing order of total processing times
J1, J3, J2, J4, J5

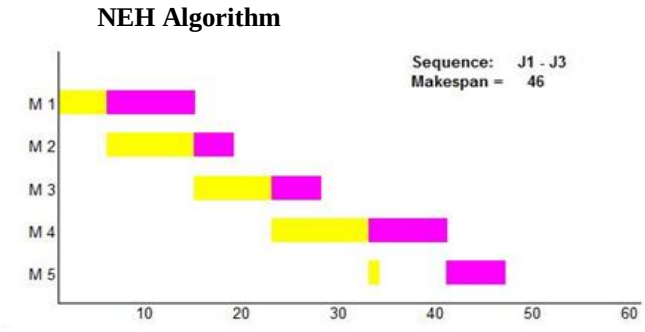

Fig: 1

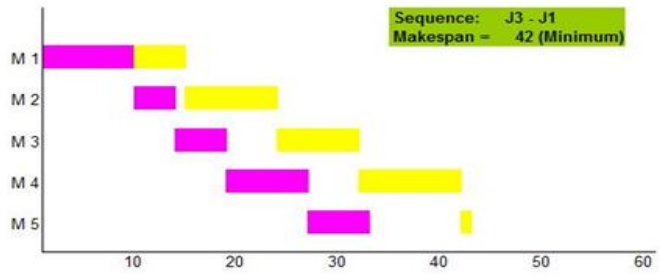

Fig: 2

Sequence: J1-J3 Makespan: 46
Sequence: J3-J1 Makespan: 42
Select sequence J3-J1

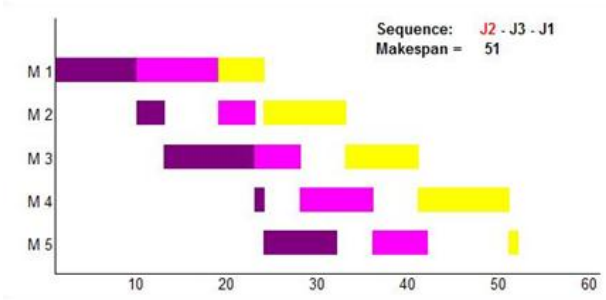

Fig: 3

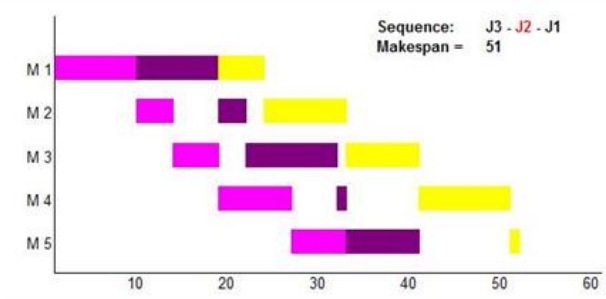

Fig: 4

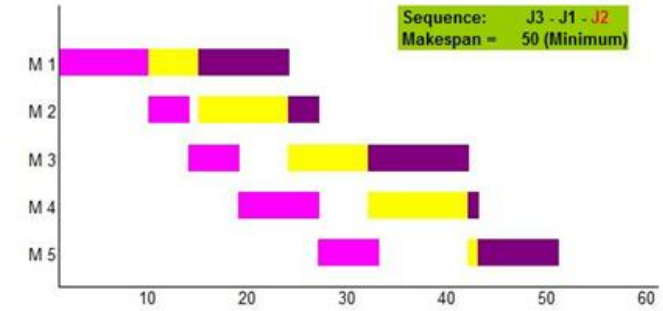

Fig: 5

Sequence: **J2**-J3-J1 Makespan: 51
Sequence: J3-**J2**-J1 Makespan: 51
Sequence: J3-J1-**J2** Makespan: 50
Select sequence J3-J1-J2

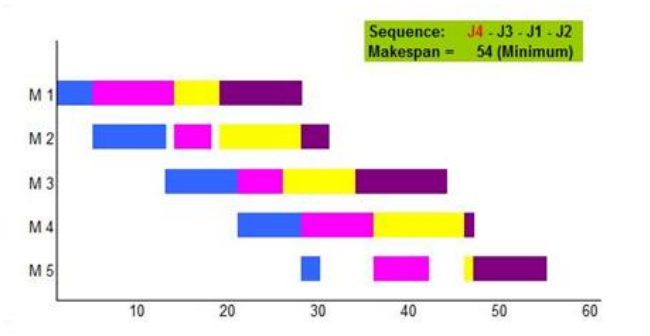

Fig: 6

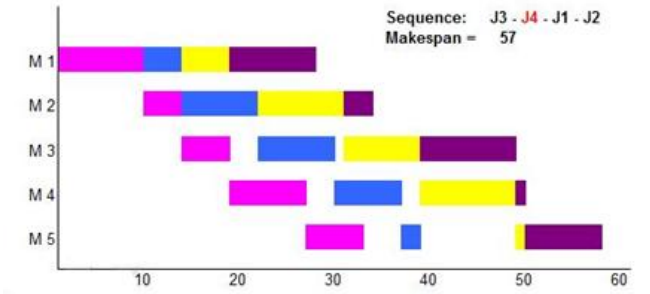

Fig: 7

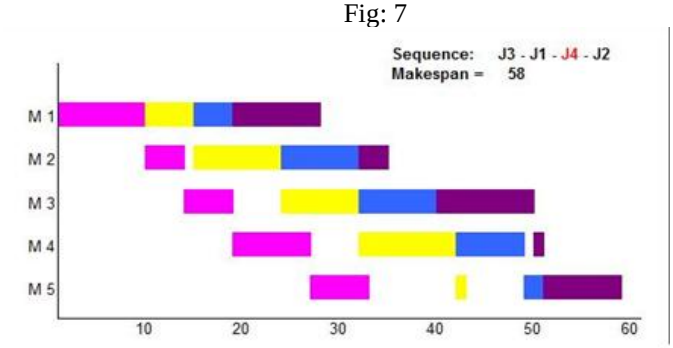

Fig: 8

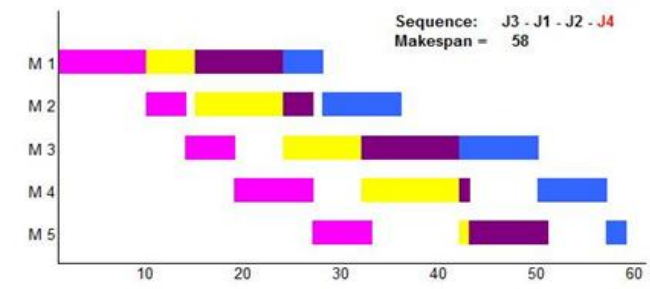

Fig: 9

Sequence: **J4**-J3-J1-J2 Makespan: 54
Sequence: J3-**J4**-J1-J2 Makespan: 57
Sequence: J3-J1-**J4**-J2 Makespan: 58
Sequence: J3-J1-J2-**J4** Makespan: 58
Select sequence J4-J3-J1-J2

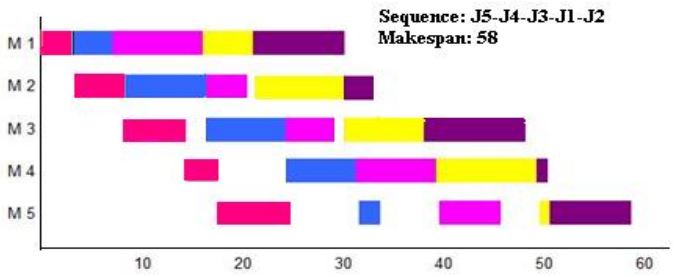

Fig: 10

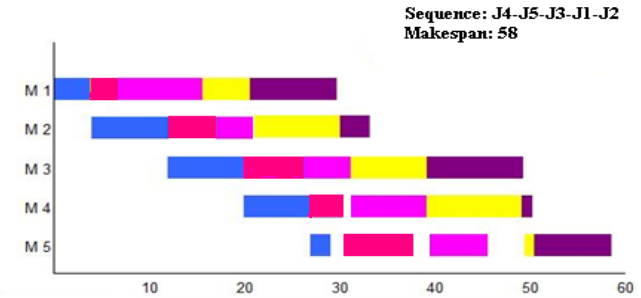

Fig: 11

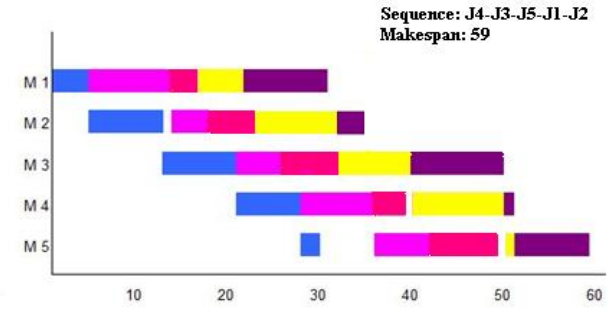

Fig: 12

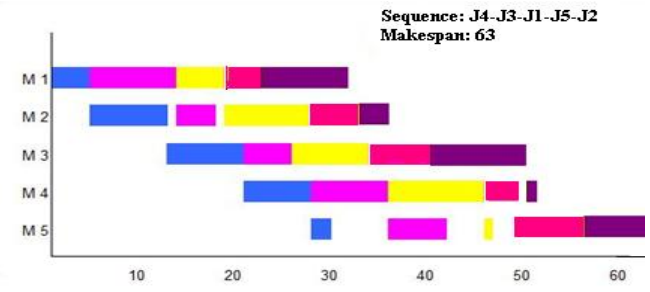

Fig: 13

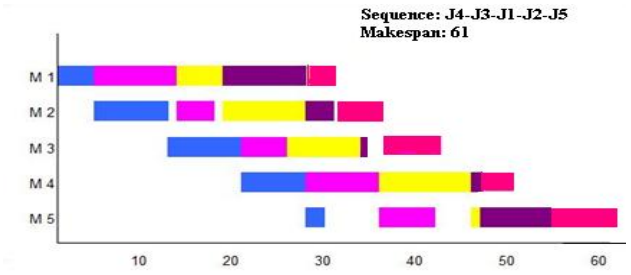

Fig: 14

J5- **Sequence**: J4-J3-J1-J2 Makespan: 58

J5- **Sequence**: J4-J3-J1-J2 Makespan: 58

J3- Sequence: J4-**J5**-J1-J2 Makespan: 59

Sequence: J4-J3-J1-**J5**-J2 Makespan: 63

Sequence: J4-J3-J1-J2-**J5** Makespan: 61

Thus, the best sequence is **J5-J4-J3-J2-J1** and **J4-J5-J3-J1-J2** with makespan of 58.

VI. Improved Heuristic

Taking first four jobs from the sorted order to form 24 partial sequences.

Sequence: J1-J2-J3-J4	Makespan: 59
Sequence: J1-J4-J2-J3	Makespan: 59
Sequence: J1-J3-J2-J4	Makespan: 57
Sequence: J1-J4-J3-J2	Makespan: 61
Sequence: J3-J1-J2-J4	Makespan: 58
Sequence: J3-J4-J1-J2	Makespan: 57
Sequence: J2-J1-J3-J4	Makespan: 58
Sequence: J2-J4-J1-J3	Makespan: 62
Sequence: J2-J3-J1-J4	Makespan: 58
Sequence: J2-J4-J3-J1	Makespan: 56
Sequence: J3-J2-J1-J4	Makespan: 58
Sequence: J3-J4-J2-J1	Makespan: 58
Sequence: J1-J2-J4-J3	Makespan: 61
Sequence: J4-J1-J2-J3	Makespan: 58
Sequence: J1-J3-J4-J2	Makespan: 58
Sequence: J4-J1-J3-J2	Makespan: 63
Sequence: J3-J1-J4-J2	Makespan: 58
Sequence: J4-J3-J1-J2	Makespan: 54
Sequence: J2-J1-J4-J3	Makespan: 62
Sequence: J4-J2-J1-J3	Makespan: 63
Sequence: J3-J2-J4-J1	Makespan: 58
Sequence: J4-J3-J2-J1	Makespan: 55
Sequence: J4-J2-J3-J1	Makespan: 55
Sequence: J2-J3-J4-J1	Makespan: 57

The parameter of the algorithm k is taken as 7.
Selecting the best 7 sequences for further processing.

J4-J3-J1-J2

Sequence: **J5**-J4-J3-J1-J2	Makespan: 58
Sequence: J4-**J5**-J3-J1-J2	Makespan: 58
Sequence: J4-J3-**J5**-J1-J2	Makespan: 59
Sequence: J4-J3-J1-**J5**-J2	Makespan: 63
Sequence: J4-J3-J1-J2-**J5**	Makespan: 61

J4-J3-J2-J1

Sequence: **J5**-J4-J3-J2-J1	Makespan: 58
Sequence: J4-**J5**-J3-J2-J1	Makespan: 60
Sequence: J4-J3-**J5**-J2-J1	Makespan: 60
Sequence: J4-J3-J2-**J5**-J1	Makespan: 60
Sequence: J4-J3-J2-J1-**J5**	Makespan: 64

J4-J2-J3-J1

Sequence: **J5**-J4-J2-J3-J1	Makespan: 58
Sequence: J4-**J5**-J2-J3-J1	Makespan: 60
Sequence: J4-J2-**J5**-J3-J1	Makespan: 60
Sequence: J4-J2-J3-**J5**-J1	Makespan: 60
Sequence: J4-J2-J3-J1-**J5**	Makespan: 64

J2-J4-J3-J1

Sequence: **J5**-J2-J4-J3-J1	Makespan: 60
Sequence: J2-**J5**-J4-J3-J1	Makespan: 62
Sequence: J2-J4-**J5**-J3-J1	Makespan: 60
Sequence: J2-J4-J3-**J5**-J1	Makespan: 60
Sequence: J2-J4-J3-J1-**J5**	Makespan: 65

J2-J3-J4-J1

Sequence: **J5**-J2-J3-J4-J1	Makespan: 62
Sequence: J2-**J5**-J3-J4-J1	Makespan: 61
Sequence: J2-J3-**J5**-J4-J1	Makespan: 63
Sequence: J2-J3-J4-**J5**-J1	Makespan: 63
Sequence: J2-J3-J4-J1-**J5**	Makespan: 67

J1-J3-J2-J4

Sequence: **J5**-J1-J3-J2-J4	Makespan: 59
Sequence: J1-**J5**-J3-J2-J4	Makespan: 60
Sequence: J1-J3-**J5**-J2-J4	Makespan: 63
Sequence: J1-J3-J2-**J5**-J4	Makespan: 63
Sequence: J1-J3-J2-J4-**J5**	Makespan: 63

J3-J4-J1-J2

Sequence: **J5**-J3-J4-J1-J2	Makespan: 60
Sequence: J3-**J5**-J4-J1-J2	Makespan: 62
Sequence: J3-J4-**J5**-J1-J2	Makespan: 62
Sequence: J3-J4-J1-**J5**-J2	Makespan: 66
Sequence: J3-J4-J1-J2-**J5**	Makespan: 64

Thus, the best sequences are

J5-J4-J3-J1-J2

J4-J5-J3-J1-J2

J5-J4-J3-J2-J1

J5-J4-J2-J3-J1

with makespan of 58.

VII. Complexity

Complexity of NEH Algorithm

The total number of enumerations in Neh is given by

$$n(n+1)/2$$

which clearly states that the complexity of this algorithm is

$$\Theta(n^2).$$

Complexity of improved heuristic

The total number of enumerations in case of the improved heuristic is given by[18]

$$4! + \sum_{z=5}^{n} k * z$$
$$= 4! + k * \sum_{z=5}^{n} z$$

Where, k denotes the algorithm parameter,
And n is the number of jobs.

Hence, the algorithmic complexity of this approach is

$$\Theta(n^2).$$

VIII. Conclusions

The improved heuristic proposed for PFSP yields better result than original NEH algorithm while maintaining the same algorithmic complexity.

As shown using an example, the improved heuristic generates more number of minimum makespan sequences as compared to the NEH algorithm and hence we have more options of job sequences that can be implemented for greater production.

Experimental studies show that the improved heuristic for PFSP results in sequences with lower makespan as compared to those obtained from NEH algorithm in case of medium sized (n=12 – 30) and large sized (n>70) problems.

IX. Future Scope

NEH is considered to be the best known heuristic for PFSPs. But the proposed heuristic has been proved to outperform NEH.

Hence, this heuristic has a great scope in industry where n jobs are required to be scheduled on m machines for greater production, efficient planning of resources and maintaining proper control over the industry.

References

[1] Dipak Laha , Uday Kumar Chakraborty, (2007), Int. J. Information and Communication Technology, Vol. 1, No. 1

[2] Nawaz, M., Enscore Jr, E. and Ham, I., (91-95), The International Journal of Management Science. v11. 91-95.

[3] Pawel J. Kalczynski, Jerzy Kamburowski, (2008), Journal Computers and Operations Research, Volume 35, Issue9

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

[5] 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.

[6] 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.

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

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

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

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

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

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

[14] 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.

[15] 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.

[16] 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.

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

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