Improvement Tools for NEH Based Heuristics on Permutation and Blocking Flow Shop Scheduling Problems

Ramon Companys*, Imma Ribas, and Manel Mateo

Dpto. de Organización de Empresas, Escuela Técnica Superior de Ingeniería Industrial de Barcelona, Universidad Politécnica de Cataluña, Av. Diagonal 647, 08028 Barcelona
{ramon.companys,imma.ribas,manel.mateo}@upc.edu

Abstract. In this paper, two tools to improve the performance of the NEH-based heuristics for the flow shop problem with and without buffer constraints are proposed. The first tool is the use of the reversibility property of the problems considered and the second one is a new tie-breaking strategy to be use in the insertion phase of the NEH heuristic. In addition, we have analyzed the behavior of five initial solution procedures for both problems. The analysis of results confirms the effectiveness of the measures proposed and allows us to recommend the best ordering procedure for each one of the problems.

Keywords: Scheduling, heuristic algorithms, permutation flow shop, blocking flow shop.

1 Introduction

This work deals with the permutation flow shop scheduling problem with and without storage space between stages. If there is enough storage space between successive machines, when a job finishes its operation can leave the machine. But, if there is no storage space between stages, intermediate queues of jobs waiting in the system for their next operation are not allowed. If operation on machine j for a job i is finished and the next machine, j+1, is still busy on the previous job, the completed job i has to be blocked into machine j. For simplicity purposes we shall refer to these problems as PFSP (permutation) and BFSP (blocking), respectively.

The most common criterion, here considered, is the minimization of the makespan or maximum completion time. Using the proposed notation by Graham et al. [1], these two problems are denoted by $Fm \mid prmu \mid C_{max}$ and $Fm \mid block \mid C_{max}$.

The PFSP has become one of the most intensively investigated topics in scheduling since the publication of the paper of Johnson [2]. For $m \geq 3$ the problem is shown to be strongly NP-hard [3]. Nawaz et al. [4] proposed a NEH heuristic which is considered one of the best heuristics for the PFSP. The NEH procedure can be divided into two steps: (1) the generation of an initial order for the jobs applying the Largest Processing Time (LPT) rule and (2) the iterative insertion of jobs, in a partial sequence, in accordance with the initial order obtained in the first step. Given its efficiency, this procedure has been widely

* Corresponding author.

B. Vallespir and T. Alix (Eds.): APMS 2009, IFIP AICT 338, pp. 33–40, 2010.

studied and different variants of it have been proposed in the literature. Nagano and Moccellin [5] proposed a different initial ordering; Framinan et al. [6] examined 176 rules used to obtain the initial sequence and showed that the ordering proposed initially in the NEH heuristic is the one which obtains the best results when the objective is to minimize the makespan. However among these rules, the Nagano and Moccelin's [5], the trapezium [7], the Pour's [8] and the *Profile Fitting* procedures are not included.

Considering now the BFSP, Reddi and Ramamoorthy [9] showed that exist a polynomial algorithm for $F2 \mid block \mid C_{max}$, which gives an exact solution. Hall and Sriskandarajah [10] showed, using a result from Papadimitriou and Kanellakis [11], that $Fm \mid block \mid C_{max}$ problem for $m \geq 3$ machines is strongly NP-hard.

In this paper we propose two tools to improve the performance of the NEH-based heuristics: the use of the reversibility property of problems considered and a new tie-breaking strategy to be used in the step 2. The computational results show the effectiveness of these tools. In addition, we have compared the performance of four initial sequencing procedures with the NEH's sequencing rule (LPT).

2 Problem Statement

At instant zero there are *n* jobs which must be processed, in the same order, in *m* machines. Each job goes from machine 1 to machine *m*. The process time for each operation is $p_{j,i}$, being $p_{j,i} > 0$, where $j \in \{1,2,...,m\}$ indicates the machine and $i \in \{1,2,...,n\}$ the job. The setup times are included in the processing time. The objective function considered is the minimization of the makespan, which is equivalent to maximizing the use of the machines.

Given a permutation, P, of the *n* jobs, [k] indicates the job in position *k* in the sequence. Given a feasible schedule associated to a permutation, $e_{j,k}$ is defined as the initial instant in which the job that occupies position *k* starts to be processed in the machine *j* and $f_{j,k}$ is defined as the instant when the job that occupies position *k* in machine *j* is finished. The PFSP can be expressed with the following formulation:

$$e_{j,k} + p_{j,[k]} \leq f_{j,k} \qquad j=1,2,...,m \qquad k=1,2,...,n \qquad (1)$$

$$e_{j,k} \geq f_{j,k-1} \qquad j=1,2,...,m \qquad k=1,2,...,n \qquad (2)$$

$$e_{j,k} \geq f_{j-1,k} \qquad j=1,2,...,m \qquad k=1,2,...,n \qquad (3)$$

$$C_{max} = f_{m,n} \qquad (4)$$

being, $f_{j,0} = 0 \quad \forall j$, $f_{0,k} = 0 \quad \forall k$, the initial conditions.

The program is semi-active if the equation (1) is written as $e_{j,k} + p_{j,[k]} = f_{j,k}$ and the equations (2) and (3) are summarized as $e_{jk} = \max\{f_{j,k-1}; f_{j-1,k}\}$.

As it is said, when there is no storage space between stages, BFSP case, if a job *i* finishes its operation on machine *j* and the next machine, *j*+1, is still processing the previous job, the completed job *i* has to remain in machine *j* until machine j+1 is free. This condition requires an additional equation (5) in the formulation of the problem.

$$f_{j,k} \geq f_{j+1,k-1} \qquad j=1,2,...,m \qquad k=1,2,...,n \tag{5}$$

The initial condition $f_{m+1,k} = 0 \quad k=1,2,...,n$ must be added.

The schedule obtained is semi-active if equation (1) and (5) is summarized as (5'):

$$f_{j,k} = \min\left\{r_{j,k} + p_{j,[k]}, f_{j+1,k-1}\right\} \quad . \tag{6`}$$

Consequently, the $Fm \mid prmu \mid C_{max}$ problem can be seen as a relaxation of the $Fm \mid block \mid C_{max}$ problem.

Both problems, $Fm \mid prmu \mid C_{max}$ and $Fm \mid block \mid C_{max}$, are reversible. Given an instance I, which can be called direct instance, with processing times $p_{j,i}$, one can determine another instance I', which can be called inverse instance, with processing times $p'_{j,i}$ calculated as (6) :

$$p'_{j,i} = p_{m-j+1,i} \qquad j = 1, 2, ..., m \qquad i = 1, 2, ..., n \tag{7}$$

For a permutation P, the value C_{max} in I is the same as the one given in I' for the inverse permutation P'. So, the minimum of maximum completion time is the same for I and I', and the permutations associated to both instances are inverse one each other. Therefore, it does not matter to solve I or to solve I'.

3 Heuristics

As it is mentioned above, the NEH heuristic is considered one of the best heuristic for the PFSP and it has also a good performance for the BFSP [12]. It consists of two steps:

Step 1: ordering jobs according the LPT rule.

Step 2: in accordance with the order established in step 1, take the first two jobs and schedule them in such a way that they minimize the partial makespan, considering an instance with only two jobs. Then for k=3 up to n, insert the k-th job into one of the possible k positions of the partial sequence. The objective is to minimize the C_{max} with k jobs.

For step 1, we have considered four initial sequencing procedures, from the literature: the proposed by Nagano and Moccellin's [5], (NM), the *Trapezium* [7] (TR), the proposed by Pour [8] (PO) and the *Profile Fitting* [13] (PF). The three first procedures were designed for the PFSP, whereas the last one was specially designed for the BFSP.

For step 2, we propose a new tie breaking method when two different positions give the same makespan. This method consists in calculating the total idle time (IT) for each position as (7):

$$\sum_{j=1}^{m} IT(j) = \sum_{j=1}^{m}\left(f_{j,n} - e_{j,1} - \sum_{i=1}^{n} p_{j,i} \right) \tag{8}$$

If there is a tie between two positions the job is inserted in the position with lower total idle time. If there is still a tie, the procedure defined in [14] for NEHKK1 is used.

Finally, as a second way to improve the solutions, we apply the procedures on the direct and inverse instance retaining the best of both solutions.

4 Computational Experience

The objective of the computational experience is to analyze the effectiveness of the proposed tools and to compare the performance of the sequencing procedures considered in step 1 to find the best procedure for each problem. We shall refer as NME2 to the variant of the proposed algorithm with NM rule [5] in step 1, as TRE2 to the variant with TR rule [7], as POE2 to the one with the PO procedure [8] and, finally, as PLE2 to the variant with PF rule [13]. As in PF rule, there is no efficient criterion for determining which job is the most suitable for the first position; therefore, we have chosen the job with the largest processing time (named as PL).

The test has been done running the algorithms on two different problem sets: the Taillard's benchmarks (1993) and nine generated sets of a thousand instances, with 3, 4 and 5 machines and 13, 14 and 15 jobs. For these instances, namely LOI instances, the optimal solutions, for the $Fm|prmu|C_{max}$ and for the $Fm|block|C_{max}$ problem, were obtained by the LOMPEN algorithm [15]. The experiments were carried out on an Intel Core 2 Duo E8400 CPU, 3GHz and 2GB of RAM memory.

To analyze the experimental results we have used the index I_{hi} calculated as (9):

$$I_{hi} = \frac{Heur_{hi} - Best_i}{Best_i} x100. \quad (9)$$

Where $Heur_{hi}$ is the average of the makespan values obtained by the heuristic *h*. and $Best_i$ is optimum or the lowest makespan known for instance *i*.

4.1 The Fm|prmu|C_{max} Problem

The Taillard instances are arranged on 12 sets of ten instances each, the sets differs by the values of the couple (*n*, *m*). Optimal or best known solutions are found in http://ina2.eivd.ch/collaborateurs/etd/problemes.dir/ordonnancement.dir/flowshop.dir/best_lb_up.txt.

Table 1. Average of I_{hi}, on Taillard instances, for $Fm|prmu|C_{max}$ for each tie breaking method

	NEH0	NEH1	NEH_KK1	NEH_IT
20x5	3.3	2.69	2.73	**2.52**
20x10	4.6	4.35	**4.31**	4.32
20x20	3.73	3.68	**3.41**	3.54
50x5	0.73	0.87	**0.59**	0.6
50x10	5.07	5.08	4.87	**4.83**
50x20	6.66	6.51	6.42	**5.77**
100x5	0.53	0.48	0.4	**0.35**
100x10	2.21	2.1	**1.77**	2.08
100x20	5.34	**5.28**	**5.28**	5.43
200x10	1.26	1.19	1.16	**1.02**
200x20	4.42	4.42	4.25	**4.19**
500x20	2.06	1.98	2.03	**1.96**
All	3.33	3.22	3.1	**3.05**

First, to analyze the effectiveness of the tie breaking method proposed, we have compared the average values of index I_{hi} for the original NEH procedure (denoted as NHE0), the two tie breaking methods proposed in Kalcynski and Kamburowski [14] (denoted as NEH1and NEH_KK1) and the tie breaking here proposed (denoted as NEH_IT). The values corresponding to each version of the NEH procedure are shown in Table 1. To carry out this comparison we have applied the procedures only on the direct instances.

As it can be seen in Table 1, the minimum overall average value of index I_{hi} is obtained when our tie breaking method is applied. Therefore, we can conclude that it is an effective tool to improve the performance of the NEH heuristic.

Table 2. Average of I_{hi}, for Taillard instances for $Fm \mid prmu \mid C_{max}$ problem

n x m	NEH	NEH2	NME	NME2	TRE	TRE2	POE	POE2	PLE	PLE2
20x5	2.69	2.32	2.58	2.29	2.71	1.77	3.51	2.40	2.82	2.14
20x10	4.42	3.80	4.20	3.05	6.18	4.29	4.99	4.39	5.31	4.22
20x20	3.71	3.43	4.23	3.62	5.01	4.44	5.12	4.09	4.01	3.57
50x5	0.57	0.56	0.92	0.67	1.04	0.64	1.06	0.79	0.96	0.55
50x10	5.05	4.34	4.91	4.72	6.01	5.53	5.96	5.24	5.00	4.44
50x20	5.78	5.76	5.82	5.60	8.05	7.14	6.93	6.71	6.68	6.15
100x5	0.36	0.35	0.45	0.35	0.67	0.39	0.66	0.44	0.52	0.38
100x10	2.06	1.65	2.18	1.71	3.17	2.46	2.76	2.44	2.26	1.88
100x20	5.43	5.00	5.47	5.03	6.68	6.39	6.43	6.07	5.24	4.84
200x10	1.07	0.97	1.12	0.96	1.42	1.13	1.57	1.18	1.12	0.96
200x20	4.11	3.96	4.19	3.94	5.34	4.98	4.94	4.81	4.29	4.16
500x20	2.02	1.80	1.91	1.82	3.17	2.85	2.75	2.70	2.21	1.96
All	3.11	2.83	3.16	2.81	4.12	3.50	3.89	3.44	3.37	2.94

Next, to compare the improvement obtained when the reversibility property is used we have reported (Table 2) the average values of I_{hi} for each set of Taillard instances and procedure. Number 2 is omitted from the name of variants when procedures are applied only on the direct instance. It can be observed that the performance increases, between 10 and 20%, when the procedures are applied on the direct and inverse instances retaining the best of both solutions

Table 3. Average CPU time, in seconds, required by each variant on each instance

n x m	NEH2	NME2	TRE2	POE2	PLE2
20x5	0.02	0.01	0.01	0.07	0.02
20x10	0.03	0.02	0.03	0.13	0.02
20x20	0.06	0.04	0.06	0.23	0.04
50x5	0.13	0.13	0.25	1.21	0.27
50x10	0.25	0.26	0.25	1.51	0.26
50x20	0.50	0.52	0.81	3.44	0,494
100x5	1.01	1.03	1.00	13.95	1.08
100x10	2.00	2.03	2.91	20.60	2.44
100x20	3.94	4.04	4.84	32.21	4.12
200x10	16.01	15.99	16.01	283.57	19.17
200x20	31.47	33.33	31.44	437.35	38.03
500x20	500.95	491.01	493.04	15147.06	588.98

Regarding the performance of the proposed variants, we can say, according to the overall average value of index I_{hi} (Table 2), that NME2 is slightly better than NEH2. It is worth noting that PLE2, with an initial procedure designed for the blocking problem, has better results than TRE2 and POE2.

To analyze the efficiency of these procedures, we have reported in Table 3, the CPU time required by each. It can be observed that all variants require a similar time except POE 2 which requires much more time.

The results obtained for the second set of problems, LOI instances, confirm the little advantage of NME2. Table 4 show the average value of I_{hi} for each set of instances and procedure. In all cases, the CPU time required to solve each instance is lower than 0.014 seconds.

Table 4. Average values of I_{hi} for each set of LOI instances and each heuristic

nxm	NEH2	NME2	TRE2	POE2	PLE2
13×3	0.24	0.17	0.27	0.26	0.23
13×4	0.88	0.76	0.86	0.88	0.82
13×5	1.59	1.52	1.59	1.60	1.41
14×3	0.24	0.18	0.27	0.32	0.21
14×4	0.79	0.70	0.85	0.91	0.71
14×5	1.59	1.46	1.69	1.71	1.43
15×3	0.17	0.13	0.24	0.23	0.18
15×4	0.75	0.65	0.84	0.84	0.67
15×5	1.49	1.44	1.65	1.68	1.47
All	0.86	0.78	0.92	0.94	0.79

4.2 The Fm|block|C_{max} Problem

First, we analyze the improvement obtained by applying the procedures on the direct and inverse instance obtaining the best of both solutions. For the BFSP problem, the best known solutions are found in Companys [16]. The average values of index I_{hi} by set and procedure are shown in Table 5. It can be seen that the improvement reached with the reversibility property tool is between 4 and 12%, less than for the permutation problem but significant.

Table 5. Average values of I_{hi} for Taillard instances for Fm | block | C_{max} problem

nxm	NEH	NEH2	NME	NME2	TRE	TRE2	POE	POE2	PLE	PLE2
20x5	5.09	4.94	6.32	4.85	7.02	5.05	5.95	5.20	5.27	4.57
20x10	5.45	5.23	5.83	4.27	5.75	5.05	6.24	4.76	4.68	4.16
20x20	3.39	3.37	4.03	2.80	4.11	3.66	4.25	3.70	4.02	3.12
50x5	8.76	8.42	8.43	7.81	9.30	8.32	7.68	7.27	8.62	8.02
50x10	7.96	7.15	8.11	7.64	7.43	6.91	7.25	6.23	7.14	6.75
50x20	7.10	6.87	6.69	6.26	5.97	5.34	5.55	4.85	5.17	5.03
100x5	7.52	7.25	7.92	7.63	8.33	8.01	7.17	6.73	7.63	7.19
100x10	6.81	6.72	6.70	6.27	6.38	6.07	6.16	5.62	6.69	6.58
100x20	5.55	5.30	5.70	5.18	5.42	4.70	4.54	4.41	4.75	4.48
200x10	6.95	6.74	6.80	6.59	6.82	6.54	6.43	5.93	6.73	6.48
200x20	4.36	4.12	4.49	4.28	3.74	3.47	3.54	3.20	3.82	3.70
500x20	3.62	3.46	3.58	3.41	2.90	2.83	2.87	2.69	3.25	3.04
All	6.05	5.80	6.22	5.58	6.10	5.50	5.63	5.05	5.65	5.26

Regarding the performance of the variants, we can see that POE2 has the lowest overall average value of index I_{hi} followed by PLE2. But, POE2 requires much more time than the others, as can be seen in Table 6. Therefore it is more recommendable PLE2 than POE2.

Table 6. Average CPU times, in second, required by each variant on each instance

nxm	NEH2	NME2	TRE2	POE2	PLE2
20x5	0.02	0.02	0.02	0.04	0.02
20x10	0.03	0.03	0.04	0.16	0.03
20x20	0.09	0.10	0.09	0.30	0.07
50x5	0.18	0.36	0.35	1.99	0.37
50x10	0.70	0.73	0.70	2.69	0.45
50x20	1.38	1.43	1.40	4.05	0.89
100x5	2.35	2.36	2.36	15.02	1.74
100x10	3.77	3.80	3.76	22.62	3.46
100x20	6.56	6.67	6.60	36.34	6.90
200x10	26.95	23.95	23.03	300.99	27.73
200x20	45.46	46.03	46.44	472.30	54.49
500x20	839.49	863.26	714.87	18713.45	841.75

The overall average of index I_{hi} obtained in the second test (Table 7), with the LOI instances, confirm this little advantage of POE2. But, due to the large CPU time required we recommend PLE2 to be used for the BFSP.

Table 7. Average values of I_{hi} for each set of LOI instances and each heuristic

nxm	NEH2	NME2	TRE2	POE2	PLE2
All	3.98	3.61	3.76	3.44	3.50

5 Conclusions

The computational experience shows the effectiveness of the tie breaking procedure and the use of the reversibility property as a way to improve the performance of the NEH-based heuristic. Regarding the performance of the variants implemented, we have seen that the behaviour of the algorithms on the permutation and the blocking cases are different. For $Fm \mid prmu \mid C_{max}$ problem, NME2 and NEH2 are the best procedures are. However, for $Fm \mid block \mid C_{max}$ problem, the best ones are POE2 and PLE2, but due to the CPU time necessary by POE2, PLE2 is more recommendable.

References

1. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: A survey. In: Annals of Discrete Mathematics, pp. 5287–5326 (1979)

2. Johnson, S.M.: Optimal two- and three-stage production schedules with set up times included. Naval Research Logistics Quarterly 1, 61–68 (1954)
3. Garey, M.R., Johnson, D.S.: Computers and intractability: A guide to the theory of NP-Completness. Freeman, San Francisco (1979)
4. Nawaz, M., Enscore Jr., E.E., Ham, I.: A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega 11, 91–95 (1983)
5. Nagano, M.S., Moccellin, J.V.: A high quality constructive heuristic for flow shop sequencing. Journal of the Operational Research Society 53, 1374–1379 (2002)
6. Framinan, J.M., Leisten, R., Ramamoorthy, B.: Different initial sequences for the heuristic of Nawaz, Enscore and Ham to minimize makespan, idletime or flowtime in the static permutation flowshop sequencing problem. International Journal of Production Research 41, 121–148 (2003)
7. Companys, R.: Métodos heurísticos en la resolución del problema del taller mecánico. Estudios Empresariales 5, 7–18 (1966)
8. Pour, H.D.: A new heuristic for the n-job, m-machine flow shop problem. Production Planning & Control 12, 648–653 (2001)
9. Reddi, S.S., Ramamoorthy, B.: On the flow-shop sequencing problem with no wait in process. Opers Res. Q. 23, 323–331 (1972)
10. Hall, N.G., Sriskandarajah, C.: A survey of machine scheduling problems with blocking and no wait in process. Operations Research 44, 510–525 (1996)
11. Papadimitriou, C.H., Kanellakis, P.C.: Flowshop scheduling with limited temporary storage. Journal of the ACM 27, 533–549 (1980)
12. Leisten, R.: Flowshop sequencing problems with limited buffer storage. Int. J. Prod. Res. 28, 2085–2100 (1990)
13. McCormick, S.T., Pinedo, M.L., Shenker, S., Wolf, B.: Sequencing in an Assembly Line with Blocking to Minimize Cycle Time. Operations Research 37, 925–936 (1989)
14. Kalczynski, P.J., Kamburowski, J.: An improved NEH heuristic to minimize makespan in permutation flow shops. Computers & Operations Research 9, 3001–3008 (2008)
15. Companys, R., Mateo, M.: Different behaviour of a double branch-and-bound algorithm on Fm|prmu|C_{max} and Fm|block|C_{max} problems. Computers & Operations Research 34, 938–953 (2007)
16. Companys, R.: Note on the blocking flowshop problem. Working paper (2009), http://upcommons.upc.edu/e-prints/handle/2117/420