Available online at www.sciencedirect.com
SCIENCE DIRECT®
PERGAMON
Omega 31 (2003) 311–317
omega
The International Journal of Management Science
www.elsevier.com/locate/dsw

An efficient constructive heuristic for flowtime minimisation in permutation flow shops

J.M. Framinan[a,*], R. Leisten[b]

[a] Industrial Management, School of Engineering, University of Seville, Seville 41092, Spain
[b] Production Management, Faculty of Business Administration and Economics, University Duisburg-Essen in Duisburg, Germany

Received 17 August 2001; accepted 28 March 2003

Abstract

In this paper, we propose a heuristic for mean/total flowtime minimisation in permutation flow shops. The heuristic exploits the idea of 'optimising' partial schedules, already present in the NEH-heuristic (Omega 11 (1983) 91) with respect to makespan minimisation. We compare the proposed heuristic against the ones by Rajendran and Ziegler (Eur. J. Oper. Res. 32 (1994) 2541), and Woo and Yim (Comput. Oper. Res. 25 (1998) 175), which are considered the best constructive heuristics for flowtime minimisation so far. The computational experiments carried out show that our proposal outperforms both heuristics with respect to the quality of the solutions. Moreover, our heuristic can be embedded in an improvement scheme to build a composite heuristic in the manner suggested by Allahverdi and Aldowaisan (Int. J. Prod. Econom. 77 (2002) 71) for the flowtime minimisation problem. The so-constructed composite heuristic also improves the best results obtained by the original composite heuristics by Allahverdi and Aldowaisan.

Keywords: Flow shop; Sequencing; Heuristics; Flowtime

1. Introduction

A flowshop is a conventional manufacturing system where the machines are arranged in the order of job processing [22]. Within a flowshop sequencing problem, a sequence of jobs which optimises certain performance measures has to be determined. One of these performance measures is mean flowtime or, equivalently, total flowtime, which is known to lead to stable or even utilisation of resources, rapid turn-around of jobs and minimisation of in-process inventory [3,4].

Since the problem of flowtime minimisation has shown to be NP-complete [5,6], most research has been devoted to developing heuristic algorithms to obtain good (but not necessarily optimal) solutions within a limited amount of computation time. The proposed heuristic methods for the above problem correspond to Gupta [7], Miyazaki et al. [8], Rajendran and Chaudhuri [9], Rajendran [10], Ho [11], Rajendran and Ziegler [12], Wang et al. [13], Woo and Yim [14] and Allahverdi and Aldowaisan [1].

Rajendran and Chaudhuri develop three heuristics that prove to perform better than those due to Gupta, and Miyazaki et al., both in terms of the quality of the solutions and the computational effort. Rajendran [10] proposes a new heuristic and shows that it performs better than the heuristics by Rajendran and Chaudhuri in terms of the quality of the solution, at the expense of higher computational burden.

Ho proposes a heuristic composed of several iterations of an improvement scheme based on finding a local optimum by adjacent pairwise interchange of jobs, and improving the solution by insertion (or shift) movements. This heuristic performs significantly better than the previous, although its main disadvantage is that it employs much higher CPU-time [1,15,16]. In fact, this heuristic seems to be closer to local

* Corresponding author. Tel.: +34-95-448-7204; fax: +34-95-448-73-29.
E-mail address: jose@esi.us.es (J.M. Framinan).

0305-0483/03/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/S0305-0483(03)00047-1

search techniques like simulated annealing or taboo search, and it has to be discarded for large problem sizes and/or in those environments where sequencing decisions are required in very short time intervals. Therefore, this heuristic is usually excluded from the comparisons among heuristics for the flowtime minimisation problem [1,15].

Woo and Yim [14] develop a heuristic—WY in the following—for mean flowtime minimisation. They compared its performance with Rajendran's heuristic, as well as with the adaptation for flowtime minimisation of other well-known heuristics for makespan minimisation such as NEH [17] and CDS [18]. They found that their algorithm outperforms the others, particularly the one by Rajendran.

Rajendran and Ziegler [12] propose the RZ heuristic, which consists of two phases. In the first phase, a seed sequence is generated by successively including the machines using a priority rule similar to the shortest weighted total processing time rule. The second phase consists of an improvement scheme based on the sequential insertion of each job in the seed sequence at each possible different position of a generated sequence.

Wang et al. [13] develop two heuristics named LIT (less idle time) and SPD (smallest process distance). The core idea for both heuristics is trying to keep the machine idle time at minimum, since idle time increases, in general, job completion times. Their heuristics are not compared with previous existing ones, but with the lower bound proposed for the problem by Ahmadi and Bagchi [19].

The heuristics by Woo and Yim, Rajendran and Ziegler, and Wang et al. were developed independently, and no comparison among the performance of these three is provided in the original papers where the heuristics are presented. In [1,15], the RZ and the WY heuristics are compared. In both studies it is found that RZ outperforms WY for small problem instances, while the relative performance of WY is improving with the number of jobs until it finally outperforms RZ. With respect to the heuristics by Wang et al. heuristics, it is stated in [1] that they are clearly outperformed by both heuristics, results that we have confirmed in a series of preliminary experiments.

Allahverdi and Aldowaisan [1] develop a set of heuristics for flowtime minimisation that constitute composite heuristics in the sense defined, e.g. in [2], that is: they employ one or several constructive heuristics (i.e. the flowtime-adaptation of the NEH heuristic, the WY heuristic, and the RZ heuristic) as starting solution and/or for improving the current solution. Among these heuristics, the best performance is achieved by the IH7 heuristic, which clearly outperforms WY and RZ as well as the rest of composite heuristics suggested in [1] for all problem sizes. The IH7 heuristic consists of three phases. In the first phase, an initial solution is obtained by applying the WY heuristic to the problem under consideration. This sequence is employed in a second phase as the seed sequence for the RZ heuristic. Finally, the so-obtained sequence is improved by a greedy general pairwise interchange until it reaches a local optimum.

The remainder of the paper is organised as follows: in Section 2 we review the Woo and Yim heuristic and give its precise complexity. In Section 3, a new heuristic exploiting the principle of partial flowtime minimisation is presented. The complexity order of the heuristic and a numerical example of the procedure are also given. In Section 4, the suggested heuristic is compared with the best existing constructive heuristics for the problem, i.e., the WY heuristic and the RZ heuristic. Additionally, we use the suggested heuristic to replace the WY heuristic in the first phase of the IH7 heuristic, and compare the results of this new composite heuristic with those obtained by IH7. Finally, the conclusions and points for future research are presented in Section 5.

2. The complexity of the Woo and Yim heuristic

The Woo and Yim heuristic can be described as follows:

Step 1: Set $K = 1$.

- Pick every job $J_i (i = 1, 2, \ldots, n)$ and evaluate its mean flowtime. Since no preceding jobs exist, the mean flowtime is the sum of processing times in all machines.
- Select the best one with the least partial mean flowtime among n jobs and set the partial schedule as a current solution S.
- Store the remaining $(n - 1)$ jobs into remaining set R.

Step 2: Update $K = K + 1$.

Step 3: Pick a job J, $J \in R$ and insert it into K possible slots in the current solution.

- Among the K sequences, select the best one with the least partial mean flowtime. Set the partial sequence as a candidate solution.
- Remove job J from remaining set R.

Step 4: If remaining set $R \neq \emptyset$ then go to Step 3, else go to Step 5.

Step 5: Among $(n - K + 1)$ candidate solutions, select the best one with the least partial mean flowtime. Set the best one as a current solution S.

- Store all jobs (except the $(n - K)$ jobs of current solution S) into remaining set R.

Step 6: If $K = n$ then stop, else go to Step 2.

Regarding complexity issues, Woo and Yim claim their algorithm to be $O(n^3)$, based on the observation that the NEH-heuristic is $O(n^2)$. However, a more precise complexity consideration yields a different result: By examining the steps of the heuristic, it is easy to see that Step 3 is the step

that largely determines the complexity of the algorithm. It can also be noted that Step 3 must be executed under the loop of K (Step 2), and for all jobs in the remaining set R (Step 5).

Step 3 consists of inserting a job J in all K possible slots of the current sequence. For each insertion, flowtime is calculated. The classical equation for calculating flowtime for a given sequence in a flowshop with n jobs and m machines is as follows (e.g. [2]):

$$C_{i0} = 0, \quad C_{0j} = 0,$$

$$C_{ij} = \max\{C_{i,j-1}, C_{i-1,j}\} + t_{ij}$$

$$(i = 1,\ldots,n), \quad (j = 1,\ldots,m),$$

where C_{ij} is the completion time of job i on machine j. From that, flowtime F is obtained by summing all C_{im}, i.e.

$$F = \sum_{i=1}^{n} C_{im}.$$

From the above equations, it is obvious that flowtime calculation in an n-job, m-machine flowshop for a given sequence is of complexity $O(nm)$. Thus, since K flowtimes in a K-job, m-machine flowshop have to be calculated in Step 3, we can conclude that Step 3 is of complexity $O(K^2m)$. As mentioned before, Step 3 is accomplished within the loop of K, and for all jobs in the remaining set $(n-K)$, so the complexity of the overall algorithm is $O((n-K)K^3m)$. It can be concluded then that the Woo and Yim algorithm has complexity $O(n^4m)$.

The statement made by Woo and Yim that their approach requires $O(n^3)$ is based in the assumption that the complexity of the NEH heuristic is $O(n^2)$. Nevertheless, the original complexity of NEH is $O(n^3m)$, as noted in [20,21], although its complexity can be improved when calculating makespan according to the procedure described by Taillard [20]. Using this procedure, NEH can be executed in $O(n^2m)$. However, Taillard's procedure is solely designed for makespan calculation and not for flowtime calculation. Such a modification, or any other procedure that allows computing the K flowtimes of inserting a job into the $K+1$ possible slots of a given schedule of length K in $O(Km)$ has not been reported. As conclusion, it is more precise to claim a complexity of $O(n^4m)$ for the Woo and Yim heuristic.

3. The proposed heuristic

The proposed heuristic is based on the idea of the NEH heuristic, which is designed for makespan minimisation. The NEH heuristic obtains a solution by first sorting the jobs in descending order of the sum of the jobs' processing times and then, iteratively taking one job after the other from this sequence, constructing partial schedules by inserting a non-scheduled job in all possible slots between each two consecutive jobs of the best partial schedule obtained in the previous iteration. This heuristic has reported excellent results in terms of makespan minimisation. It has been shown that this performance is due to the number of evaluations of the partial schedules rather than to the initial order of the initial sequence [16]. However, this heuristic is not the best one for flowtime optimisation, as shown, e.g. in [1,14,15]. Therefore, we propose performing, in every iteration, a general pairwise interchange to improve the quality of the partial schedules.

More specifically, the heuristic can be described as follows:

Step 1: Set $K = 1$.

- Pick every job $J_i (i = 1,2,\ldots,n)$ and evaluate its mean flowtime. Since no preceding jobs exist, the mean flowtime is the sum of processing times in all machines. Order the jobs in ascending order of the partial flowtime. Select the best one with the least partial flowtime among n jobs and set the partial schedule as a current solution S.
- Store the remaining $(n-1)$ jobs into remaining set R ordered by ascending order of their sum of processing times.

Step 2: Update $K = K + 1$.

Step 3: Pick a job J, $J \in R$ and insert it into K possible slots in the current solution.

- Among K sequences, select the best one with the least partial flowtime. Set the partial sequence C as a candidate solution.
- Set $S := C$.
- Remove job j from remaining set R.

Step 4: If $K > 2$, then for all i, j $(1 \leqslant i < K,\ i < j \leqslant K)$, obtain all possible sequences by interchanging jobs in positions i and j of the partial sequence C. If, among all $K \cdot (K-1)/2$ partial sequences, one sequence C' holds a lower partial flowtime than S, then set $S := C'$.

Step 5: If the remaining set $R = \emptyset$ (or, alternatively, if $K = n$) then stop, else go to Step 2.

It is easy to see that the complexity order of the proposed heuristic is $O(n^4m)$. By simple inspection, it may be seen that Steps 3 and 4 are those that determine the complexity of the algorithm. In Step 3, K flowtime calculations have to be accomplished, while $K \cdot (K-1)/2$ flowtime calculations should be carried out in Step 4. In total, $K \cdot (K+1)/2$ flowtime calculations should be performed. As Steps 3 and 4 are executed under the loop of K, and taking into account that every flowtime calculation is done on a partial sequence of K jobs and hence its complexity is $O(Km)$, then the overall complexity of the heuristic is $O(K^3 \cdot m \cdot (K+1)/2)$, or $O(n^4m)$. This is the same complexity as the Woo and Yim heuristic, which is, as previously stated, the best constructive heuristic for big problem sizes.

In order to illustrate the proposed heuristic, let us consider the following 5-jobs, 5-machines problem instance:

	M_1	M_2	M_3	M_4	M_5
J_1	55	51	62	77	26
J_2	22	24	34	57	14
J_3	61	4	94	22	81
J_4	63	71	73	55	57
J_5	17	33	87	82	61

According to Step 1, the initial ordering of the jobs is {2–3–1–5–4}. Therefore, the initial partial sequence is {2}. The rest of the jobs are stored in R.

The next job to be inserted in the partial sequence is job 3, which is inserted in the two possible slots according to Step 3 and obtaining the partial sequences {3–2} and {2–3}. The flowtime of the first is 538, while the flowtime of the second is 435. Therefore, {2–3} is selected as partial schedule for the next iteration. Note that, in this iteration, there is no need to perform Step 4 since this would lead to the re-evaluation of {3–2} and {2–3}.

For $K = 3$, the job to be tried is 1. The partial sequences {1–2–3}, {2–1–3}, and {2–3–1} yield a partial flowtime of 992, 831, and 789, respectively. Accordingly, {2–3–1} is selected as a result of Step 3. In Step 4, a general pairwise interchange is operated on {2–3–1}. The resulting partial sequences are {3–2–1}, {1–3–2}, and {2–1–3}, obtaining a flowtime of 896, 1015, and 831, respectively. Since no one of these improves the result obtained by the seed sequence, {2–3–1} is retained as best partial sequence.

In Step 3 for $K = 4$, the following partial sequences are tried: {5–2–3–1}, {2–5–3–1}, {2–3–5–1}, and {2–3–1–5}. The flowtime of the partial sequences are 1383, 1278, 1299, and 1270. Therefore, the selected partial sequence is {2–3–1–5}. In Step 4, the general interchange applied to the selected partial sequence gives the following sequences: {3–2–1–5}, {1–3–2–5}, {5–3–1–2}, {2–1–3–5}, {2–5–1–3}, and {2–3–5–1}. The flowtime of these sequences are 1381, 1541, 1478, 1345, 1246, and 1299, respectively. The partial sequence {2–5–1–3} is the best among these sequences and improves the seed solution; therefore, it is chosen as best partial sequence. In an identical manner, for $K = 5$ the final sequence {2–5–4–1–3} with a flowtime of 1744 is reached.

4. Computational experiments

In order to compare the performance of the suggested heuristic with the existing ones, two test-beds based on the problem instances designed by Woo and Yim [14] have been developed. The first test-bed is composed of small problems that can be optimally solved by enumeration. Hence, comparison of the heuristic results against the optima can be obtained. This test-bed is composed of problem instances of 5–9 jobs, and 5, 10, 15, and 20 machines. The second test-bed is composed of large problem instances including 10, 20, 30, 40, 50, 60, 70, and 80 jobs, and 5, 10, 15, and 20 machines.

For both test-beds, the processing times at each machine are generated randomly based on a uniform distribution between 1 and 99. One hundred replications are conducted for each combination of machines and jobs. Hence, the first test-bed contains 2000 instances, while the second consists of 3200 instances.

The constructive heuristics to be compared with our proposed heuristic are the RZ heuristic and the WY heuristic. As mentioned above, these heuristics have been shown to be the best so far constructive heuristic for the flowtime minimisation problem [1,15]. Additionally, we include for comparison the IH7 heuristic, which is the heuristic that obtains the best quality of solutions among the set of composite heuristics designed by Allahverdi and Aldowaisan [1]. Finally, we include a new composite heuristic (named IH7-proposed in the following) into our comparison consisting of replacing the WY algorithm by our heuristic in the first phase of the IH7 heuristic. Note that the complexity of the IH7-proposed heuristic is the same as that of the one of IH7, since we are merely replacing the WY heuristic used as initial phase of the IH7 heuristic by another heuristic of the same complexity.

In the same manner as Woo and Yim, two measures are computed to compare the performance of both heuristic methods: the average relative percentage deviation (ARPD) from the best-known solution, and the number of problem instances for which the heuristic under consideration produces the best-known solution. For the first test-bed (small problems), the best-known solution is the optimal solution of the instances obtained by complete enumeration. For the test-bed with the big problems, the best-known solution is the minimum flowtime obtained among the compared heuristics. The ARPD is defined as follows:

$$\sum_{i=1}^{100}\left(\frac{\text{heuristic}_i - \text{best known}_i}{\text{best known}_i}\right)/100.$$

Results in Table 1 indicate that, for all tested numbers of jobs and machines, the proposed heuristic improves the performance of the WY heuristic. This happens in both the quality of the solutions (as measured in terms ARPD) and the robustness (as measured in terms of the number of instances in which, for a given size of the problem, the algorithm reaches the optimal solution). In none of the problem sizes, the average deviation from the optima exceeds 1%. Finally, it is to note that, for a given number of jobs, the performance of the algorithm does not decrease with the number of machines.

Regarding the comparisons of our proposed heuristic with the RZ heuristic, there is not a clear order between both. Although the overall results for the small test-bed show that our suggested heuristic outperforms RZ in terms of quality of the results, the latter obtains the best solutions for seven out of the 20 problem sizes. In terms of the number of optima

Table 1
Comparison of heuristics for the small problems

		WY		RZ		Proposed		IH7		IH7-proposed	
n	*m*	ARPD	No. optimal	ARPD	No. optimal	ARPD	No. optimal	ARPD	No. optimal	ARPD	No. optimal
5	5	0.700	60	0.323	75	0.331	74	0.090	92	0.076	91
	10	0.612	62	0.192	85	0.289	79	0.098	92	0.081	91
	15	0.389	61	0.093	93	0.171	78	0.107	87	0.029	94
	20	0.303	60	0.026	92	0.098	81	0.069	89	0.042	93
6	5	0.581	51	0.472	62	0.346	67	0.127	82	0.086	86
	10	0.724	43	0.429	60	0.317	66	0.172	78	0.070	84
	15	0.563	45	0.164	73	0.279	65	0.088	83	0.088	86
	20	0.403	49	0.136	73	0.203	66	0.056	86	0.030	89
7	5	1.067	32	0.639	43	0.509	59	0.242	66	0.172	79
	10	0.663	38	0.539	49	0.438	56	0.171	76	0.177	73
	15	0.738	36	0.429	50	0.334	52	0.201	73	0.128	76
	20	0.692	28	0.353	54	0.328	50	0.140	69	0.148	73
8	5	1.226	16	0.984	33	0.926	41	0.432	54	0.398	62
	10	1.087	26	0.918	34	0.603	39	0.300	62	0.286	64
	15	1.123	18	0.655	34	0.547	39	0.277	62	0.172	74
	20	0.744	21	0.423	45	0.447	35	0.246	54	0.210	63
9	5	1.331	19	1.099	26	0.877	26	0.414	51	0.393	53
	10	1.441	12	1.120	17	0.932	23	0.439	40	0.426	49
	15	1.217	13	0.813	25	0.895	21	0.328	47	0.364	42
	20	0.970	17	0.810	23	0.580	30	0.320	47	0.234	54
Total		0.829	707	0.531	1046	0.473	1047	0.216	1390	0.180	1476

reached, both heuristics seem to behave similarly for the small test-bed.

It can be also seen from Table 1 that the suggested heuristic is clearly outperformed for all problem sizes by the composite IH7 heuristic, both in terms of quality of solutions and number of optima. However, when embedding our heuristic in the IH7 schema (IH7-proposed), we obtain a powerful composite heuristic that yields the best results among all heuristics (both constructive and composite) under comparison. In average, IH7-proposed obtains solutions that do not exceed 0.2% from the optima, and these optima are reached for roughly three quarters of the instances.

The results for the big test-bed offered in Table 2 confirm those already presented. With respect to the constructive heuristics under comparison (WY, RZ, and proposed), our proposal obtains the best results in terms of ARPD for all problem sizes. Note that the average improvement obtained by our proposal with respect to WY and RZ is about 50%. Regarding the average number of times that each heuristic reaches the best-known solution for a given problem size, our proposal is best for most of the problem sizes and in the total average results for all problem sizes. From Table 2 it can be seen as well that the performance of the RZ heuristic decreases with the number of jobs, and it is outperformed not only by our proposal, but also by WY.

Regarding the two versions of the composite heuristic, IH7-proposed outperforms IH7 in terms of ARPD for all problem sizes. In average, the results obtained by IH7-proposed improve those by IH7 by almost 50%. In terms of the average number of times that the best-known solution is reached, IH7-proposed outperforms IH7 for all but one problem size. In total, the best-known solution is reached by IH7-proposed in nearly 50% more problems as compared with employing IH7.

5. Conclusions

In this paper, we have presented a constructive heuristic for flowtime minimisation in permutation flow shops with the same complexity as the Woo and Yim heuristic, which is the best so far existing constructive heuristic for large problem instances. The computational results indicate that the proposed heuristic outperforms the current best constructive heuristics. Additionally, we find that we can also improve the best composite heuristic for flowtime minimisation—i.e.

Table 2
Comparison of heuristics for the big problems

		WY		RZ		Proposed		IH7		IH7-proposed	
n	*m*	ARPD	No. best	ARPD	No. best	ARPD	No. best	ARPD	No. best	ARPD	No. best
10	5	1.556	14	1.155	18	0.915	29	0.419	57	0.206	72
	10	1.237	20	1.173	17	0.860	21	0.327	59	0.289	62
	15	1.240	7	0.890	22	0.528	24	0.375	46	0.210	63
	20	1.033	20	0.866	21	0.541	30	0.271	52	0.173	62
20	5	2.025	0	1.977	2	1.123	4	0.510	37	0.263	55
	10	2.019	1	2.051	5	1.173	4	0.431	44	0.326	46
	15	1.993	1	1.863	7	1.136	3	0.481	44	0.310	46
	20	1.666	0	1.550	4	0.931	5	0.400	34	0.317	50
30	5	1.995	1	2.080	1	0.937	3	0.467	38	0.232	61
	10	2.075	0	2.008	2	1.257	1	0.554	38	0.289	57
	15	1.993	0	1.989	3	1.070	2	0.576	34	0.263	60
	20	1.852	0	1.881	4	0.965	1	0.418	36	0.284	56
40	5	2.148	0	2.572	1	0.934	2	0.476	36	0.213	63
	10	2.217	0	2.259	1	1.286	0	0.509	48	0.328	51
	15	1.940	0	2.123	3	1.141	1	0.461	47	0.357	51
	20	1.860	0	1.877	0	1.082	1	0.446	37	0.200	60
50	5	1.902	0	2.216	0	0.763	0	0.464	37	0.185	63
	10	1.958	0	2.512	2	1.125	0	0.449	37	0.281	61
	15	1.993	0	2.163	2	1.112	1	0.478	43	0.262	54
	20	1.874	0	1.995	1	1.110	1	0.535	34	0.194	65
60	5	1.920	0	2.265	0	0.742	1	0.432	29	0.134	71
	10	2.096	0	2.309	1	1.051	0	0.475	37	0.262	62
	15	1.890	0	2.263	1	1.098	0	0.365	52	0.344	46
	20	1.832	0	1.892	0	1.144	0	0.404	44	0.234	55
70	5	1.787	0	2.362	0	0.712	1	0.332	37	0.203	63
	10	1.930	0	2.326	0	1.080	0	0.475	38	0.226	62
	15	1.925	0	2.143	1	0.983	1	0.503	34	0.195	65
	20	1.907	0	2.159	0	1.032	0	0.496	33	0.238	67
80	5	1.718	0	2.466	0	0.600	1	0.344	41	0.214	59
	10	1.961	0	2.292	0	0.934	0	0.448	31	0.156	69
	15	1.825	0	2.207	0	0.987	0	0.360	44	0.275	56
	20	1.792	0	1.776	0	0.851	0	0.451	44	0.206	56
Total		1.849	64	1.989	119	0.975	137	0.442	1302	0.246	1889

the IH7 heuristic—by replacing the Woo and Yim heuristic used in the first phase of IH7 by our proposal. The result is a powerful composite heuristic that obtains the best results in terms of quality of the solutions and average number of times that it reaches the best-known (or optimal) solution.

Given the strength of the proposed heuristic, future research lines may investigate new composite heuristics or improvement schemes in order to improve the results. Besides, it is clear that the suggested heuristic may be employed for other objective functions relevant to flow shop scheduling. However, this has to be tested, since the mere adaptation of heuristics to a different objective—as in the case of the application of the NEH heuristic to flowtime minimisation—does not necessarily yield similar results to those obtained for the original problem.

Acknowledgements

The authors wish to thank the referees for their ameliorating review. The core idea of this paper was conceived while

the first author was in the University of Duisburg (Germany) as recipient of a grant funded by the German Academic Exchange Service (DAAD).

References

[1] Allahverdi A, Aldowaisan T. New heuristics to minimize total completion time in *m*-machine flowshops. International Journal of Production Economics 2002;77:71–83.
[2] Pinedo M. Scheduling: theory, algorithms, and systems. Englewood Cliffs, NJ: Prentice-Hall, 1995.
[3] French S. Sequencing and scheduling: an introduction to the mathematics of the job-shop. Chichester: Ellis Horwood, 1982.
[4] Rajendran C. A heuristic for scheduling in flowshop and flowline-based manufacturing cell with multi-criteria. International Journal of Production Research 1994;32: 2541–58.
[5] Garey MR, Johnson DS, Sethi R. The complexity of flowshop and jobshop scheduling. Mathematics of Operations Research 1976;1:117–29.
[6] Rinnooy Kan AHG. Machine scheduling problems: classification, complexity, and computations. The Hague: Nijhoff, 1976.
[7] Gupta JND. Heuristic algorithms for multistage flowshop scheduling problem. AIIE Transactions 1972;4:11–8.
[8] Miyazaki S, Nishiyama N, Hashimoto F. An adjacent pairwise approach to the mean flowtime scheduling problem. Journal of the Operations Research Society of Japan 1978;21: 287–99.
[9] Rajendran C, Chaudhuri D. An efficient heuristic approach to the scheduling of jobs in a flowshop. European Journal of Operational Research 1991;61:318–25.
[10] Rajendran C. Heuristic algorithm for scheduling in a flowshop to minimise total flowtime. International Journal of Production Economics 1993;29:65–73.
[11] Ho JC. Flowshop sequencing with mean flowtime objective. European Journal of Operational Research 1995;81:571–8.
[12] Rajendran C, Ziegler H. An efficient heuristic for scheduling in a flowshop to minimize total weighted flowtime of jobs. European Journal of Operational Research 1997;103:129–38.
[13] Wang C, Chu C, Proth JM. Heuristic approaches for $n/m/F/\sum C_i$ scheduling problems. European Journal of Operational Research 1997;96:636–44.
[14] Woo DS, Yim HS. A heuristic algorithm for mean flowtime objective in flowshop scheduling. Computers and Operations Research 1998;25:175–82.
[15] Framinan JM, Leisten R, Rajendran C. 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 2003;41:121–48.
[16] Framinan JM, Leisten R, Ruiz-Usano R. Efficient heuristics for flowshop sequencing with the objectives of makespan and flowtime minimisation. European Journal of Operational Research 2002;141:561–71.
[17] Nawaz M, Enscore EE, Ham I. A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem. Omega 1983;11:91–5.
[18] Campbell HG, Dudek RA, Smith ML. A heuristic algorithm for the *n*-job, *m*-machine sequencing problem. Management Science 1970;16:B630–7.
[19] Ahmadi RH, Bagchi U. Improved lower bounds for minimizing the sum of completion times of *n* jobs over *m* machines. European Journal of Operational Research 1990;44:331–6.
[20] Taillard E. Some efficient methods for the flow shop sequencing problem. European Journal of Operational Research 1990;47:65–74.
[21] Turner S, Booth D. Comparison of heuristics for flowshop sequencing. Omega 1987;15:75–85.
[22] Baker KR. Introduction to sequencing and scheduling. New York: Wiley, 1974.