Comparison of heuristics for flowtime minimisation in permutation flowshops

Technical report IO-2003/01

Version 0.5
Last version: 26/07/2003

Jose M. Framinan
Industrial Management, School of Engineering, University of Seville

Rainer Leisten
Production Management, University Duisburg-Essen in Duisburg

A revised, enhanced version of this Report is to be published in Computers & Operations Research

ABSTRACT

We focus our attention on the problem of sequencing jobs in a permutation flow shop with the objective of minimising the sum of completion times or flowtime. This objective is considered to be more relevant and meaningful for today's dynamic production environment, and therefore it has attracted the attention of researchers during the last years. As a result, a number of different types of heuristics have been recently developed, each one claiming to be the best for the problem. However, these heuristics have been independently developed and only partial comparisons among them exist. Consequently, there are no conclusive results on their relative performance. Besides, some of these types of heuristics are of a different nature and could be combined in order to obtain composite heuristics. In this paper we first conduct an extensive comparison among the existing heuristics. Secondly, based on the results of the experiments, we suggest two new composite heuristics for the problem. The subsequent computational experience shows these two heuristics to be efficient for the problem under consideration.

1. Introduction

A flowshop consists of *n* jobs that must be processed on *m* machines in the same order. The scheduling problem in flow shops is finding a sequence of jobs for each machine according to certain performance measure(s). Additionally, for many situations, it is assumed that the job sequences will be the same on every machine (permutation flowshops). Other hypotheses common in scheduling research are, e.g. the simultaneous availability of all jobs and of all stations, deterministic processing times, etc. For a complete list of these assumptions, see e.g. [6].

Among the objectives to be considered when scheduling in a flowshop, the minimisation of the total completion time or makespan is shown to be directly related to the maximisation of the throughput and the usage of the resources. Therefore, it is not surprising that most of the research during the last decades has concentrated in the minimisation of the makespan.

More recently, the amount of research devoted to the minimisation of the sum of the completion times of the jobs (or equivalently mean flow time or mean completion time) has increased. This objective is also known as (total or average) flow time minimisation, and it is denoted as the $F|prmu|\Sigma C_j$ problem, according to the notation by [11]. Flow time minimisation leads to stable or even use of resources, a rapid turn-around of jobs and the minimisation of in-process inventory [21]. Therefore, it is considered to be more relevant and meaningful for today's dynamic production environment [14].

Given the NP-hard nature of the $F|prmu|\Sigma C_j$ problem, most of the research on this topic is devoted to propose heuristics that yield good approximate solutions. Prior to year 2000, the best constructive heuristics available for the problem were the heuristics by Woo and Yim [25], and by Rajendran and Ziegler [21].

More recently, a number of heuristic approaches for the problem have been published, i.e. the heuristic by Liu and Reeves [14], the set of composite heuristics by Allahverdi and Aldowaisan [1], the work by Framinan et al. [9] on the application of the NEH heuristic [17] to the $F|prmu|\Sigma C_j$ problem, and the heuristic by Framinan and Leisten [7].

Most of these works have been independently developed and only a partial comparison between the heuristics by Allahverdi and Adowaisan and that of Framinan and Leisten is reported [7]. Hence, there are no conclusive results on the comparison of these suggested heuristics. Besides, some of these heuristics are of a different nature: on one hand, the heuristic by Liu and Reeves consists of a first phase were the jobs are sorted according to an index taking into account the idle time followed by a local improvement phase. On the other hand, the set of heuristics by Allahverdi and Aldowaisan can be considered as composite heuristics in the sense defined e.g. in Pinedo [18], i.e. they contain other heuristics – namely the Woo and Yim, and the Rajendran and Ziegler ones. Therefore, it will be worth to investigate whether different combinations of heuristics could be employed for constructing new composite heuristics, and whether there exist other local improvement schemes that could be applied in order to improve their performance.

The paper is organised as follows: In the next section we review the main contributions regarding heuristic solutions for the problem under consideration. Additionally, we develop a framework to classify the different phases of these heuristics. This framework will be useful when searching for new combinations of the existing heuristics. In Section 3 we present the experimental design for the comparison of heuristics, while the results of these comparisons are presented in Section 4. This section is divided in two parts: first the simple heuristics are compared, and secondly new combinations of heuristics are tested. Finally, Section 5 is devoted to conclusions.

2. Review of existing heuristics

The problem of minimising flowtime in permutation flowshops has been addressed by a number of researchers. Prior to 1998, there are some heuristics that have been clearly outperformed by the new developments on the problem. These 'old' heuristics are presented by Gupta [12], Miyazaki *et al.* [16], Rajendran and Chaudhuri [19], Rajendran [19], Ho [13], and Wang *et al.* [24]. Since there exists a number of papers proving that all these heuristics are outperformed by the new heuristics that were appearing since 1997 (see e.g. [1], or [14]), we do not elaborate further on these. Additionally, it should be mentioned that in [14] it is shown that several of these heuristics perform worse than random choice of a sequence for the usual parameter combinations set in most numerical studies. The only exception is the Ho heuristic, which consists of several iterations of an improvement scheme based on finding a local optimum by adjacent pairwise interchange of jobs, and then improving the solution by insertion (or shift) movements. This heuristic seems closer to local search techniques such as simulated annealing or tabu search than to constructive heuristics and its CPU time requirements do not make it suitable for large problem sizes and/or in those environments where sequencing decisions are required in short time intervals (see the evolution of the CPU times required for this heuristic e.g. in [10]). As a consequence, the Ho heuristic has not been included in previous comparisons of heuristics for the problem under consideration (see e.g. [1], or [14]), and it is neither included in this study.

The oldest heuristic that we discuss in this review is the Rajendran and Ziegler [21] heuristic – RZ in the following –. This heuristic consists in, first, obtaining a seed sequence by sorting the jobs according to a priority rule similar to the shortest weighted total processing time. Secondly, the seed sequence is improved by sequential insertion of each job, according to the seed sequence, into the best sequence found so far.

Independently developed, the Woo and Yim (WY) heuristic [25] exploits the idea presented in the NEH heuristic [17] for makespan minimisation. The NEH heuristic consists of two phases: First the jobs are ranked according to the descending sum of their processing times. In a second phase, a solution is constructed in the following manner: Starting from a partial sequence constructed by taking the first job of the rank, then, for $k = 2,\dots,n$, k partial sequences are constructed by inserting the kth job of the rank in all k slots of the partial sequence. These k partial sequences are evaluated with respect to makespan and the one obtaining the lowest value is retained as partial sequence for step $k + 1$. Obviously, a straightforward adaptation of this heuristic to the $F|prmu|\Sigma C_j$ problem consist of evaluating the partial sequences in the second phase with respect to flowtime. In the following, we name this adaptation NEH-flowtime.

In the WY heuristic, jobs are initially sorted according to the ascending sum of their processing times. Then, in an iterative process, a partial sequence is constructed by trying to insert all non-scheduled jobs in all possible slots of the partial sequence. More specifically, for iteration k of this heuristic, all non scheduled $n - k + 1$ jobs are inserted into all k possible slots of the partial sequence of size $k - 1$. Several comparisons carried out between WY and RZ heuristics (e.g. [1] and [9]) indicate that, for a small number of jobs, the latter outperforms the former, while the opposite occurs when the number of jobs is growing. With respect to computation times, WY is much slower than RZ.

Liu and Reeves [14] present a heuristic based on developing an index function to sort the jobs. The index function is based on two terms: the weighted total machine idle time, and the artificial total flow time. The so-obtained solution is then improved by employing several kinds of greedy pairwise interchanges named FPE and BPE (we will detail both procedures later on). The heuristic can be enhanced by generating x different schedules consisting in using each of the first x jobs (according to the above-described index) and constructing a sequence by selecting jobs one by one using the index function. The sequence obtaining the minimum flow time is selected and the pairwise interchange schemes are then applied. Adapting Liu and Reeves' notation, we label this set of heuristics as LR(x), meaning x the number of schedules generated. The LR(x) heuristics are shown to clearly outperform both WY and RZ [14].

Allahverdi and Aldowaisan [1] present seven heuristics – named IHx – which are constructed by combining several heuristics such as WY and RZ. These heuristics are compared with the original WY and the RZ heuristic, and it turns out that the so-called IH7 is the best in terms of the quality of the solutions, outperforming both heuristics. The IH7 heuristic consists of employing the solution provided by the WY heuristic as initial solution, then applying the second phase of the RZ heuristic to construct a schedule, and finally using a local search method based on general pairwise interchange in order to improve the solution. The mechanism of the other six IHx heuristics is summarised in table 1.

The idea of extending the mechanism of solution construction of the NEH heuristic to the flowtime minimisation problem is extensively investigated by Framinan et al. [9]. In this study, the authors study 177 different approaches for ranking the jobs in the first phase of the NEH heuristic. They evaluate a large test-bed for all possible five-tupels among the 177 approaches and select the best of them (named B5FT in the following), i.e. the five-tupel which on average yields the best solution by taking the minimum flowtime of the five solutions generated by the second phase of NEH for all five initial sequences for the respective five-tupel. According to their results, the B5FT heuristic outperforms both WY and RZ in terms of quality of the solutions.

Framinan and Leisten [7] present a heuristic – FL in the following – based on constructing a partial schedule in the manner of the NEH heuristic. However, here the jobs are initially ordered according to the ascending sum of processing times. As in NEH, the *k*th job of this index is inserted into the *k* possible slots, and the partial schedule with lowest flow time is retained as best solution. After that, a general pairwise interchange is applied to this partial schedule, and the best partial solution with respect to flow time is retained. The process is repeated until all jobs are scheduled. This heuristic is shown to outperform the WY and the RZ heuristic. Finally, in order to compare their heuristic with the composite heuristics of Allahverdi and Aldowaisan, a version of the IH7 heuristic is presented consisting of employing the FL heuristic as an initial solution instead of the WY heuristic as in the original IH7. The experiments show that the latter heuristic – named IH7-FL– clearly improves the original IH7.

As a summary of the review, within the last years a number of heuristics for the flow time minimisation problem have been developed independently and therefore no extensive comparison among them has been carried out. Furthermore, some of the heuristics presented are of a very different nature (i.e. some are composite heuristic based on one or more simple heuristics). Since no classification and comparison of the existing simple heuristics has been carried out, the process of building composite heuristics has not been completed.

In order to classify the different relevant heuristics for the problem under consideration, we follow the scheme employed by Framinan et al. [8] when classifying simple heuristics for makespan minimisation. They divide the existing heuristics into three phases, namely index development, solution construction, and solution improvement. Each heuristic can use one or more of these phases. During the index development phase, jobs are sorted by employing some characteristic of the problem (e.g. processing times) or some analogy (e.g. the Travelling Salesman Problem). The solution construction phase consists of developing a solution starting from a given order of jobs (possibly employing the order in the index development phase). Note that, in this phase, there is no guarantee that the final solution is better than the initial solution. Finally, the solution improvement phase consists of employing some local search mechanism in order to improve the current solution.

The term 'composite heuristic' itself is not always clear. For instance, [14] classify the heuristics into constructive heuristics and improvement heuristics. A heuristic is then labelled as composite if it is a combination of a constructive heuristic plus an improvement heuristic. From this point of view, heuristics such as IH6 in [1] will be classified together with constructive heuristics such as the WY heuristic, being the latter the first phase of the former.

Extending the above classification in [8] to include composite heuristics, we define the term composite heuristic in the following manner: a heuristic is regarded as composite if it employs another heuristic for one or more of the three above-mentioned phases. Consequently, a heuristic is regarded as simple if it does not contain another heuristic within any of the three phases. According to the above presented classification, among the simple heuristics mentioned there are several strategies for phase I (sorting jobs). These are:

- Descending sum of processing times (for NEH-flowtime)
- Weighted descending sum of processing times (for RZ)
- Five different indices based on functions depending on the processing times of the jobs (for B5FT, see [9] for details)
- Ascending sum of processing times (for FL)
- An index function based on the weighted total machine idle time and the artificial total flow time (for LR)

With respect to the solution construction phase, several strategies are employed:

- NEH-C. This strategy consists of inserting the *k*th job of the initial solution in each of the *k* + 1 possible slots of the best partial solution. Each of these so-obtained partial sequences is evaluated, and the best partial sequence is retained as the best partial solution. The process is repeated until a whole schedule is constructed.

- RZ-C. This strategy consists of inserting each of the jobs of the initial solution into each of the possible slots.
- FL-C. This strategy consists of inserting the *k*-th job of the initial solution in each of the $k + 1$ possible slots of the best partial solution. Each of these so-obtained partial sequences is evaluated, and the best partial sequence is retained as the best partial solution. The chosen partial solution is tried to be improved by performing a pairwise interchange. If a better result is obtained, the new partial solution is retained as the best partial sequence. The process is repeated until a whole schedule is constructed.
- WY. As mentioned before, this strategy can be seen as an extension of the NEH strategy, where all non-scheduled ($n - k$) jobs are tried into all possible k slots. The partial schedule yielding the lowest flow time is retained as the best partial schedule for the next iteration.

Finally, with respect to the solution improvement phase, several strategies have been adopted in the reviewed heuristics:

- FPE – Forward Pairwise Interchange ([14]). It consists of starting from an initial sequence and then tries to exchange each job with each of a certain number of jobs following it in the sequence. For each trial, if the new sequence obtained is better, then the exchange is actually made. Otherwise, the two jobs are kept in their positions. After all the exchanges are tried, the process starts over again from the first job in the sequence, and the procedure continues until no improvement can be made for a whole round of trials.
- BPE – Backward Pairwise Interchange ([14]). This strategy is similar to FPE, but now the exchanges start from right to left.
- FPE-R – Forward Pairwise Interchange (Restart). This strategy is followed in [1] under the name of ‘pairwise interchange’. It is identical to FPE, but when the exchange is actually made because the new solution improves the current one, the process of exchange starts from the first job in the new sequence.

Additionally, there is one strategy that follows directly from the ideas expressed above: BPE-R or Backward Pairwise Interchange (Restart). This is similar to BPE, but when the exchange is actually made because the new solution improves the current one, the process of exchange starts from the last job in the new sequence.

		Phase I	Phase II	Phase III
Simple heuristics	NEH-flowtime	ASC(sum(pt))	NEH-C	
	B5FT	Best of 5 initial sequences	NEH-C	
	WY		WY	
	RZ	ASC(w sum(pt))	RZ-C	
	LR	LR (3 versions)		
	FL	DESC(sum(pt))	FL-C	
Composite heuristics	IH1	NEH-flowtime		FPE-R
	IH2	NEH-flowtime (best of 5 runs)		
	IH3	NEH-flowtime (best of 5 runs)		FPE-R
	IH4	WY		FPE-R
	IH5	RZ		FPE-R
	IH6	WY	RZ-C	
	IH7	WY	RZ-C	FPE-R
	IH7-FL	FL	RZ-C	FPE-R

Table 1. Summary of simple and composite heuristics for the flowtime minimisation problem. In this table, the composite heuristics are expressed in terms of the simple heuristics (appearing in shadowed boxes)

The four above-mentioned strategies for the improvement phase exploit the pairwise interchange neighbourhood. Although there are no studies specifically on the case of flow time minimisation, the insertion neighbourhood is considered to be the most efficient neighbourhood scheme for local search with makespan minimisation objective.

Additionally, the RZ-C strategy is based on insertion movements. Therefore, it may be of interest to see whether these results can be extended to the case of flow time minimisation. In Section 4.2 we will introduce four insertion strategies equivalent to the four presented here for the exchange neighbourhood.

Table 1 summarises the phases employed by the heuristics described in this section.

3. Experimental design

In order to compare the heuristics described in the previous section, it is required to obtain or develop a set of problem instances. Perhaps the most well-known test-bed for permutation flowshop problems is the one developed by Taillard [22]. This is a testbed available in the OR-Library [2] and consists of a set of 120 problem instances of various problem sizes (10 instances for each problem size). However, there are several problems for employing this test-bed here: on one hand Taillard's test-bed was specifically designed for makespan minimisation and not for flow time minimisation, since the problem instances were chosen so that the distance between their lower bound (referring to makespan) and the results obtained by a lengthy tabu search was high. On the other hand, we try to perform an exhaustive comparison including hypothesis testing, and the number of instances for each problem size in the test bed (10) is not sufficient to draw concluding results.

For almost all of the reviewed heuristics, the test beds in which these heuristics are compared generate the processing times using a discrete rectangular distribution [1,99] (see e.g. [1], [9], or [25]). Although this distribution is not likely to be found in practice, it is known to produce difficult type of problem instances (see e.g. [4] or [5]) and therefore constitutes an excellent benchmark for the comparison of heuristics.

With respect to the problem sizes, after reviewing the papers describing heuristics for flowtime minimisation, we try to cover a high range of problem sizes. Therefore, *n* the number of jobs is chosen as $n \in \{10,20,30,40,50,60,70,80,100,200\}$, and *m* the number of machines is chosen as $m \in \{5, 10, 15, 20\}$. For each problem size, 100 instances are generated. This relatively high number of problem instances of the same size is chosen such that the results with respect to hypotheses contrasts are as high as possible. In total, the number of problem instances of the test-bed is 4,000.

4. Comparison of heuristics

In this section, we carry out the comparison of the heuristics presented in section 2 by employing the test-bed designed in section 3. In order to obtain a clear picture of the results, we first compare the simple heuristics among them in terms of the quality of the solutions and their computational effort. According to these results, we then construct two composite heuristics which are compared with existing composite heuristics.

4.1. Simple heuristics

The heuristics to be compared are WY, RZ, B5FT, LR (in the three versions described in [15]), and FL. The results of the comparison are presented in tables 3 and 4. In table 3, the Average Relative Percentage Deviation (ARPD) and the CPU time (in seconds) of each heuristic for each combination of n and m are presented. The ARPD has been obtained with respect to the best known solution for each instance.

In order to check the statistical significance of the results, we test a number of hypotheses using a one-sided test for the differences of means of paired samples (see e.g. [2] or [14]) for every combination of m and n. More specifically, we test all combinations among the WY, RZ, LR(x), B5FT, and FL heuristics. In table 2, we summarize the hypotheses H0 and the corresponding hypotheses H1 in the symmetric position of the matrix.

The results are shown in table 4, where ps are given as the maximum level of significance to reject H0 (p represents the limit value to reject hypothesis H0 resulting from a t-test, i.e. for every level of significance $\alpha \leq p$, H0 would have to be rejected, whereas for every $\alpha > p$, H0 would not be rejected. A high p indicates that H0 can be rejected with high level of significance and therefore H1 can be accepted.)

H0

H1

	WY	RZ	H(*n*)	B5FT	FL
WY		F(WY) ≤ F(RZ)	F(WY) ≤ F(H(*n*))	F(WY) ≤ F(B5FT)	F(WY) ≤ F(FL)
RZ	F(WY) > F(RZ)		F(RZ) ≤ F(H(*n*))	F(RZ) ≤ F(B5FT)	F(RZ) ≤ F(FL)
H(*n*)	F(WY) > F(H(*n*))	F(RZ) > F(H(*n*))		F(H(*n*)) ≤ F(B5FT)	F(H(*n*)) ≤ F(FL)
B5FT	F(WY) > F(B5FT)	F(RZ) > F(B5FT)	F(H(*n*)) > F(B5FT)		F(B5FT) ≤ F(FL)
FL	F(WY) > F(FL)	F(RZ) > F(FL)	F(H(*n*)) > F(FL)	F(B5FT) > F(FL)	

Table 2. Hypotheses to be tested for the simple heuristics

In view of the results of both tables, the following statements can be made:

- Regarding the overall results, there are three heuristics that are outperformed by the others: these are the WY, RZ, and the LR(1) heuristic. WY performs better than RZ for most problem sizes except the smallest instances, where the latter outperforms WY. LR(1) is better than both for the combination of m = 5 and $n \geq 40$, and better than RZ for $m \geq 10$ and $n \geq 100$. For the rest of the combinations LR(1) is outperformed by both WY and RZ. However, the small CPU requirements consumed by LR(1) in order to obtain results that are only around 5% of the best have to be taken into account.
- With respect to the rest of the heuristics, the FL heuristic is best in terms of the quality of the solutions obtained. However, it is outperformed by LR(*n*) for the combination of a big number of jobs and a small number of machines (i.e. for $n \geq$ 60 and m = 5, and for n =200 and m =10), and also by B5FT for a small number of jobs combined with a large number of machines.
- The B5FT heuristic is second in terms of quality of the results, being outperformed by LR(*n*) and LR(*n*/10) – third and four in quality of results – only for the combination of small number of machines and big number of jobs. Additionally, it is a very fast heuristic, and therefore it seems to be particularly suitable for

environments where scheduling decisions should be taken in very small time intervals.

- As mentioned before, LR(*n*) is third in quality of the results. However, the CPU requirements are much higher than for the rest of the heuristics. From this point of view, it seems that the relative performance of LR(*x*) deteriorates as *x* increases: in the whole test-bed, LR(*n*) takes more than 2,800 seconds to obtain an ARPD of 3.150, while LR(1) takes less than 18 seconds to obtain 4.217. This may speak for the suitability of the approach by Liu and Reeves to obtain fast, good solutions, but not for the approach of trying different initial job orders.

4.2. Composite heuristics.

In this subsection, we try to combine some of the heuristics tested in the previous section. Specifically, we are interested in combining two or more heuristics in order to obtain a new (composite) heuristic that covers at least the phases I and II depicted in table 1. After that, for phase III we employ some of the local search methods described above in order to improve the obtained solutions. Note that the process of obtaining composite heuristics can be nested into several loops, i.e. we can obtain a ‘second generation’ of composite heuristics consisting of combining several composite heuristics. However, here we restrict our attention in obtaining ‘first generation’ composite heuristics whose computation times do not differ very much from existing (simple or composite) heuristics.

With respect to the first phase – index development – it seems clear that LR obtains reasonably good results in negligible CPU time. In contrast, for the solution construction phase (phase II), the so called FL-C seems to obtain very good results. Therefore, it seems reasonable to combine both phases into a single heuristic. We therefore build a composite heuristic named C1, consisting of applying the FL solution construction scheme (labelled FL-C in table 1) starting from the solution provided by LR.

n	m	RZ		B5FT		WY		FL		LR(1)		LR(n/10)		LR(n)	
		ARPD	CPU(s)	ARPD	CPU(s)	ARPD	CPU(s)	ARPD	CPU(s)	ARPD	CPU(s)	ARPD	CPU(s)	ARPD	CPU(s)
10	5	1.435	0.00	0.681	0.00	1.836	0.00	1.194	0.00	2.208	0.00	2.208	0.00	1.696	0.00
10	10	1.357	0.00	0.487	0.00	1.423	0.00	1.044	0.00	2.362	0.00	2.362	0.00	1.743	0.00
10	15	1.005	0.00	0.356	0.01	1.355	0.00	0.642	0.00	1.815	0.00	1.815	0.00	1.471	0.00
10	20	3.081	0.00	2.359	0.01	3.260	0.06	2.749	0.00	3.821	0.00	3.821	0.00	3.298	0.00
20	5	5.504	0.01	4.745	0.02	5.553	0.00	4.614	0.00	6.132	0.00	5.662	0.00	5.353	0.05
20	10	5.268	0.01	3.885	0.02	5.246	0.05	4.383	0.06	6.355	0.00	5.697	0.00	5.090	0.06
20	15	4.063	0.02	2.855	0.04	4.192	0.00	3.321	0.05	4.988	0.00	4.529	0.00	4.223	0.11
20	20	3.517	0.02	2.471	0.05	3.631	0.06	2.882	0.05	4.395	0.00	4.009	0.00	3.732	0.16
30	5	5.311	0.03	4.809	0.05	5.237	0.11	4.140	0.11	5.372	0.00	4.836	0.00	4.735	0.17
30	10	4.460	0.04	3.575	0.08	4.534	0.16	3.697	0.17	6.070	0.00	5.110	0.05	4.614	0.27
30	15	4.150	0.06	2.983	0.11	4.149	0.17	3.213	0.22	5.076	0.05	4.369	0.06	4.150	0.50
30	20	3.567	0.07	2.424	0.14	3.534	0.22	2.633	0.28	4.565	0.00	3.647	0.05	3.492	0.60
40	5	5.717	0.07	5.066	0.12	5.282	0.27	4.032	0.38	4.726	0.06	4.247	0.06	4.185	0.50
40	10	4.658	0.10	3.668	0.18	4.621	0.39	3.674	0.55	5.092	0.00	4.278	0.11	4.045	0.87
40	15	4.051	0.14	2.932	0.26	3.866	0.54	3.045	0.71	5.252	0.00	4.068	0.11	3.923	1.37
40	20	3.711	0.17	2.637	0.34	3.691	0.61	2.893	0.93	4.780	0.06	3.897	0.22	3.586	1.81
50	5	5.186	0.12	5.016	0.24	4.862	0.61	3.692	0.83	4.467	0.05	3.934	0.11	3.898	1.10
50	10	5.166	0.20	4.105	0.36	4.601	0.93	3.745	1.32	5.730	0.05	4.411	0.22	4.268	2.20
50	15	3.988	0.26	2.943	0.49	3.814	1.27	2.921	1.81	5.042	0.11	3.778	0.33	3.549	3.30
50	20	3.742	0.34	2.650	0.63	3.620	1.53	2.843	2.30	5.068	0.11	3.794	0.44	3.606	4.39
60	5	4.788	0.21	4.684	0.38	4.439	1.15	3.230	1.75	3.833	0.06	3.230	0.27	3.207	2.25
60	10	5.148	0.33	4.445	0.64	4.946	1.87	3.869	2.69	6.017	0.11	4.544	0.44	4.359	4.45
60	15	4.577	0.46	3.449	0.85	4.204	2.47	3.388	3.63	5.482	0.11	4.180	0.66	3.973	6.70
60	20	3.455	0.59	2.458	1.09	3.391	3.19	2.697	4.67	4.657	0.16	3.128	0.93	3.018	8.90
70	5	4.586	0.32	4.430	0.59	3.997	2.30	2.901	3.08	2.984	0.06	2.613	0.39	2.597	4.06
70	10	5.059	0.53	4.427	0.94	4.657	3.30	3.782	4.89	5.661	0.16	4.317	0.82	4.218	8.13
70	15	4.113	0.74	3.238	1.30	3.890	4.56	2.932	6.70	4.748	0.17	3.376	1.26	3.215	12.14
70	20	3.647	0.93	2.605	1.67	3.386	5.76	2.506	8.56	4.795	0.22	3.236	1.60	3.046	16.20
80	5	4.819	0.50	4.495	0.89	4.063	3.63	2.917	5.44	3.262	0.11	2.779	0.66	2.758	6.87
80	10	5.332	0.78	4.817	1.43	4.976	5.71	3.929	8.18	5.588	0.17	4.435	1.37	4.324	13.73
80	15	4.161	1.06	3.261	1.91	3.769	7.75	2.919	11.26	4.859	0.28	3.358	2.03	3.259	20.54
80	20	3.481	1.38	2.674	2.46	3.499	9.83	2.545	14.39	4.586	0.33	3.128	2.75	2.956	27.41
100	5	3.301	0.94	2.574	1.71	2.154	8.79	1.064	12.63	1.149	0.16	0.840	1.65	0.832	16.53
100	10	3.172	1.51	2.713	2.75	2.559	13.40	1.570	20.49	2.891	0.33	1.801	3.35	1.710	32.90
100	15	2.772	2.09	2.098	3.68	2.435	18.62	1.515	27.30	3.788	0.50	2.386	4.94	2.260	49.27
100	20	2.610	2.68	1.768	4.73	2.355	23.73	1.463	34.99	3.539	0.66	2.014	6.65	1.880	65.53
200	5	2.961	7.51	1.729	13.17	1.533	137.04	0.658	193.83	0.613	1.27	0.404	25.70	0.401	255.51
200	10	2.810	12.01	2.428	20.93	2.081	208.55	1.024	309.23	1.581	2.63	0.831	51.36	0.792	510.15
200	15	2.649	16.44	2.171	28.70	1.873	285.12	1.073	424.08	2.320	3.90	1.290	76.95	1.183	764.29
200	20	2.426	21.15	1.815	36.89	1.787	365.75	1.046	543.87	3.029	5.22	1.472	102.61	1.347	1019.36
Avg:		**3.870**		**3.073**		**3.608**		**2.710**		**4.217**		**3.346**		**3.150**	

Table 3. Comparison of the simple heuristics for the flowtime minimisation problem

		Hypotheses to be tested									
n	m	WY $\le$ RZ	WY $\le$ LR(n)	WY $\le$ B5FT	WY $\le$ FL	RZ $\le$ LR(n)	RZ $\le$ B5FT	RZ $\le$ FL	LR(n) $\le$ B5FT	LR(n) $\le$ FL	B5FT $\le$ FL
10	5	100.000	100.000	100.000	100.000	0.000	100.000	100.000	100.000	100.000	0.000
10	10	100.000	0.000	100.000	100.000	0.000	100.000	100.000	100.000	100.000	0.000
10	15	100.000	0.000	100.000	100.000	0.000	100.000	100.000	100.000	100.000	0.000
10	20	100.000	0.307	100.000	100.000	0.000	100.000	100.000	100.000	100.000	0.000
20	5	99.559	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
20	10	4.133	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	0.000
20	15	100.000	7.243	100.000	100.000	0.000	100.000	100.000	100.000	100.000	0.000
20	20	100.000	0.000	100.000	100.000	0.000	100.000	100.000	100.000	100.000	0.000
30	5	0.000	100.000	100.000	100.000	100.000	100.000	100.000	0.045	100.000	100.000
30	10	100.000	0.000	100.000	100.000	0.000	100.000	100.000	100.000	100.000	0.000
30	15	71.357	73.027	100.000	100.000	52.814	100.000	100.000	100.000	100.000	0.000
30	20	3.643	99.981	100.000	100.000	100.000	100.000	100.000	100.000	100.000	0.000
40	5	0.000	100.000	100.000	100.000	100.000	100.000	100.000	0.000	100.000	100.000
40	10	0.396	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	73.729
40	15	0.000	0.002	100.000	100.000	100.000	100.000	100.000	100.000	100.000	0.000
40	20	6.801	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	0.000
50	5	0.000	100.000	0.000	100.000	100.000	100.000	100.000	0.000	100.000	100.000
50	10	0.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
50	15	0.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	99.537
50	20	0.000	88.509	100.000	100.000	100.000	100.000	100.000	100.000	100.000	0.000
60	5	0.000	100.000	0.000	100.000	100.000	100.000	100.000	0.000	8.318	100.000
60	10	0.000	100.000	100.000	100.000	100.000	100.000	100.000	0.000	100.000	100.000
60	15	0.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
60	20	0.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	0.000
70	5	0.000	100.000	0.000	100.000	100.000	100.000	100.000	0.000	0.000	100.000
70	10	0.000	100.000	100.000	100.000	100.000	100.000	100.000	0.000	100.000	100.000
70	15	0.000	100.000	100.000	100.000	100.000	100.000	100.000	3.198	100.000	100.000
70	20	0.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
80	5	0.000	100.000	0.000	100.000	100.000	100.000	100.000	0.000	0.000	100.000
80	10	0.000	100.000	100.000	100.000	100.000	100.000	100.000	0.000	100.000	100.000
80	15	0.000	100.000	100.000	100.000	100.000	100.000	100.000	37.711	100.000	100.000
80	20	92.531	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
100	5	0.000	100.000	0.000	100.000	100.000	100.000	100.000	0.000	0.000	100.000
100	10	0.000	100.000	0.000	100.000	100.000	100.000	100.000	0.000	100.000	100.000
100	15	0.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
100	20	0.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
200	5	0.000	100.000	0.000	100.000	100.000	100.000	100.000	0.000	0.000	100.000
200	10	0.000	100.000	0.000	100.000	100.000	100.000	100.000	0.000	0.000	100.000
200	15	0.000	100.000	0.000	100.000	100.000	100.000	100.000	0.000	100.000	100.000
200	20	0.000	100.000	0.000	100.000	100.000	100.000	100.000	0.000	100.000	100.000

Table 4. Maximum level of significance (*p*) for the rejection of H0 for the simple heuristics

With respect to the local improvement method to employ within the third phase, it has been previously mentioned that, in addition to the existing three strategies employed in the literature (FPE, FPE-R, and BPE) and the BPE-R strategy that follows straight from the previous, the insertion strategy can be also considered. This strategy seems to be interesting because, although there are no studies on the efficiency of this neighbourhood for the problem under consideration, the insertion is considered to be the most efficient neighbourhood with respect to the makespan minimisation problem (see e.g. [22]). Additionally, the insertion neighbourhood constitutes the basis for the improvement phase of the RZ heuristic, which produces acceptable solutions (particularly if integrated into a composite heuristic, as in the IH7 heuristic). Therefore, we also consider the following four neighbourhood strategies: FIE (Forward Insertion Exchange), FIE-R (Forward Insertion Exchange – Restart), BIE (Backward Insertion Exchange), and BIE-R (Backward Insertion Exchange – Restart). Their definition is analogous to the strategies presented in section 2, so we omit their explicit description.

Obviously, the efficiency of the improvement strategy may depend on the quality of the initial (starting) solution and on whether the heuristic is connected or not to a specific kind of neighbourhood (i.e., the FL heuristic is connected to the FPE neighbourhood and therefore, it may seem foreseeable that this insertion neighbourhood may explore different solutions and thus be more effective). In order to check these assumptions, we obtain the ARPD values for all eight neighbouring strategies starting from the solutions provided by several heuristics (i.e. Random solution, RZ, WY, FL, and C1). The results are shown in table 5 in terms of the relative ARPD improvement of each strategy with respect to the starting solution, i.e.: $\frac{\mathrm{ARPD(H)} - \mathrm{ARPD(N(H))}}{\mathrm{ARPD(H)}} \times 100$

where ARPD(H) is the ARPD obtained by applying a specific heuristic H and ARPD(N(H)) is the ARPD obtained after applying the neighbouring strategy N to the solution obtained by H. The higher the value of the relative ARPD improvement, the greater the improvement obtained when applying that specific neighbouring strategy to the solution provided by the heuristic. In table 5, the ranks of these relative ARPD improvements are shown within brackets. In view of this table, the following statements can be made:

- With the exception of the case with the random solution as starting solution, the forward strategy always produces better results than the corresponding backward strategy.
- For most of the cases, re-starting the search after finding a new solution improves the results with respect to not doing so.
- For random, WY, and RZ, the pairwise interchange neighbourhood seems to be more efficient than the insertion neighbourhood. However, for FL and Comp_1, the best results are achieved for the insertion neighbourhood.

With all the information gathered from the above results, we built a second composite heuristic C2, consisting of employing C1 to obtain an initial solution, then applying the RZ-C construction phase, and finally applying FIE-R to the resulting solution . These two heuristics – C1 and C2 – are compared with the most effective composite heuristics, i.e. the IH7 heuristic by [1], and FL-IH7 [7]. Additionally, we include the FL heuristic in order to check whether C1 outperforms FL or not. The results are presented in table 6.

Heuristic	Pairwise Exchange (PE)				Insertion Exchange (IE)			
	Forward (F)		Backward (B)		Forward (F)		Backward (B)	
	FPE-R	FPE	BPE-R	BPE	FIE-R	FIE	BIE-R	BIE
Random	91.548 (1)	87.753 (2)	84.491 (4)	85.632 (3)	76.856 (7)	74.353 (8)	77.819 (5)	77.263 (6)
RZ	36.099 (2)	32.293 (3)	31.558 (5)	31.293 (6)	32.231 (4)	44.403 (1)	27.293 (7)	27.236 (8)
WY	28.991 (2)	37.661 (1)	24.822 (6)	25.016 (5)	28.721 (3)	27.284 (4)	23.189 (8)	23.717 (7)
FL	3.589 (5)	3.381 (6)	3.178 (8)	3.271 (7)	16.505 (1)	15.870 (2)	12.894 (4)	13.345 (3)
Comp_1	22.176 (5)	22.082 (6)	21.870 (8)	21.905 (7)	23.821 (1)	23.625 (2)	22.531 (4)	22.693 (3)

Table 5. Average relative improvement over each heuristic solution obtained for the different neighbouring strategies. The number within brackets is the rank that each neighbouring strategy occupies.

Again, we test a number of hypotheses using a one-sided test for the differences of means of paired samples for every combination of *m* and *n*. More specifically, we tested the combinations of the best simple heuristics B5FT and FL, and the IH7, FL-IH7, C1, and C2 composite heuristics. We summarize in table 7 the hypotheses H0 and the corresponding hypotheses H1 in the symmetric position of the matrix. The only exception is the hypothesis FT(B5FT) ≤ FT(FL), which has been already tested in table 4.

	B5FT	FL	C1	IH7	IH7-FL	C2
B5FT		**H0**	F(B5FT) ≤ F(C1)	F(B5FT) ≤ F(IH7)	F(B5FT) ≤ F(IH7-FL)	F(B5FT) ≤ F(C2)
FL	**H1**		F(FL) ≤ F(C1)	F(FL) ≤ F(IH7)	F(FL) ≤ F(IH7-FL)	F(FL) ≤ F(C2)
C1	F(B5FT) > F(C1)	F(FL) > F(C1)		F(C1) ≤ F(IH7)	F(C1) ≤ F(IH7-FL)	F(C1) ≤ F(C2)
IH7	F(B5FT) > F(IH7)	F(FL) > F(IH7)	F(C1) > F(IH7)		F(IH7) ≤ F(IH7-FL)	F(IH7) ≤ F(C2)
IH7-FL	F(B5FT) > F(IH7-FL)	F(FL) > F(IH7-FL)	F(C1) > F(IH7-FL)	F(IH7) > F(IH7-FL)		F(IH7-FL) ≤ F(C2)
C2	F(B5FT) > F(C2)	F(FL) > F(C2)	F(C1) > F(C2)	F(IH7) > F(C2)	F(IH7-FL) > F(C2)	

Table 7. Hypotheses to be tested for the composite heuristics

The results are shown in table 8, where *p*s are given as the maximum level of significance to reject H0. From the results of tables 7 and 8, the following statements can be made:

- The best simple heuristics (FL and B5FT) are outperformed by all the composite heuristics for most of the problem sizes. More specifically, C1 outperforms FL for all except two problem sizes (note that both heuristics only differ in the initial job ordering). Therefore, employing the scheme by Liu and Reeves as starting order seems to be of interest. C1 is also better than B5FT on the overall results, although it is outperformed by the latter for one third of the settings (those involving the smallest number of jobs). Finally, C1 is outperformed by the rest of the heuristics.
- Regarding the quality of the results, C2 is the best of the heuristics under consideration. It outperforms the rest for all problem sizes, with the exception of IH7-FL for three specific settings. It is also the most time-consuming heuristic, although in the order of magnitude of the heuristics under comparison.

- With respect to the rest of heuristics, IH7-FL performs only slightly better than IH7 in the overall results. However, this difference is consistent for all problem sizes.

n	M	FL		C1		IH7		FL-IH7		C2	
		ARPD	CPU(s)	ARPD	CPU(s)	ARPD	CPU(s)	ARPD	CPU(s)	ARPD	CPU(s)
10	5	1.182	0.00	0.970	0.00	0.685	0.00	0.472	0.00	0.405	0.00
10	10	1.042	0.00	0.693	0.00	0.508	0.01	0.470	0.01	0.307	0.00
10	15	0.619	0.00	0.718	0.00	0.466	0.01	0.301	0.01	0.303	0.00
10	20	2.749	0.00	2.649	0.00	2.477	0.01	2.375	0.01	2.333	0.01
20	5	4.614	0.00	4.496	0.01	3.983	0.03	3.730	0.04	3.824	0.04
20	10	4.383	0.06	4.089	0.02	3.622	0.05	3.512	0.06	3.198	0.07
20	15	3.312	0.05	3.275	0.02	2.636	0.07	2.468	0.08	2.392	0.08
20	20	2.876	0.05	2.793	0.03	2.336	0.09	2.252	0.10	2.025	0.11
30	5	4.140	0.11	3.924	0.05	3.664	0.16	3.416	0.18	3.277	0.18
30	10	3.694	0.17	3.518	0.08	2.979	0.23	2.705	0.28	2.699	0.28
30	15	3.210	0.22	3.230	0.11	2.703	0.30	2.385	0.35	2.280	0.39
30	20	2.632	0.28	2.608	0.14	2.076	0.37	1.939	0.44	1.896	0.46
40	5	4.032	0.38	3.699	0.15	3.559	0.50	3.287	0.54	3.196	0.53
40	10	3.674	0.55	3.371	0.25	2.867	0.67	2.688	0.79	2.529	0.84
40	15	3.044	0.71	2.837	0.34	2.358	0.87	2.249	1.06	2.008	1.10
40	20	2.887	0.93	2.901	0.43	2.243	1.07	1.989	1.34	1.890	1.42
50	5	3.692	0.83	3.503	0.35	3.384	1.16	3.099	1.31	3.060	1.15
50	10	3.745	1.32	3.539	0.59	3.054	1.56	2.879	1.87	2.697	2.07
50	15	2.919	1.81	2.756	0.81	2.272	2.11	2.055	2.44	1.900	2.65
50	20	2.836	2.30	2.779	1.02	2.250	2.52	1.907	3.12	1.882	3.34
60	5	3.230	1.75	2.887	0.72	2.917	2.59	2.612	2.56	2.482	2.34
60	10	3.869	2.69	3.659	1.19	3.281	3.26	3.063	4.00	2.921	3.72
60	15	3.388	3.63	3.246	1.64	2.641	4.19	2.617	5.05	2.394	5.28
60	20	2.697	4.67	2.466	2.08	1.944	5.19	1.775	6.33	1.573	6.49
70	5	2.901	3.08	2.648	1.31	2.514	5.07	2.382	4.58	2.275	4.08
70	10	3.782	4.89	3.443	2.17	3.162	6.10	2.906	7.14	2.732	6.99
70	15	2.932	6.70	2.689	2.99	2.440	7.75	2.131	9.01	1.891	9.38
70	20	2.506	8.56	2.367	3.81	1.958	9.29	1.698	11.39	1.577	12.11
80	5	2.917	5.44	2.606	2.20	2.657	8.55	2.523	7.53	2.331	7.13
80	10	3.929	8.18	3.707	3.65	3.430	10.76	3.128	11.68	3.032	11.74
80	15	2.919	11.26	2.717	5.05	2.282	13.29	2.194	14.95	2.021	15.62
80	20	2.545	14.39	2.416	6.43	2.138	15.54	1.887	18.67	1.669	20.25
100	5	1.064	12.63	0.784	12.63	0.812	22.71	0.617	18.89	0.541	16.40
100	10	1.570	20.49	1.286	20.05	1.065	28.53	0.977	27.12	0.710	27.78
100	15	1.515	27.30	1.295	27.87	1.046	32.50	0.831	34.99	0.633	36.67
100	20	1.463	34.99	1.374	35.86	0.993	39.18	0.713	44.51	0.637	49.61
200	5	0.658	193.83	0.358	191.41	0.364	527.39	0.364	307.96	0.231	236.80
200	10	1.024	309.23	0.701	309.93	0.794	579.93	0.556	448.52	0.404	401.93
200	15	1.073	424.08	0.892	424.77	0.778	617.73	0.549	578.91	0.397	584.52
200	20	1.046	543.87	0.937	548.32	0.760	687.57	0.472	716.56	0.381	750.81
Avg.		**2.708**		**2.521**		**2.202**		**2.004**		**1.873**	

Table 6. Comparison of the composite heuristics for the problem

n	m	B5FT ≤ C1	B5FT ≤ IH7	B5FT ≤ IH7-FL	B5FT ≤ C2	FL≤C1	FL≤IH7	FL≤ IH7-FL	FL≤C2	C1≤IH7	C1≤IH7-FL	C1≤C2	IH7 ≤ IH7-FL	IH7≤C2	IH7-FL≤C2
10	5	0.000	24.137	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
10	10	0.000	0.004	83.111	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	99.998	100.000	100.000
10	15	0.000	0.000	100.000	100.000	0.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	15.875
10	20	0.000	0.000	0.020	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
20	5	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	0.000
20	10	0.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
20	15	0.000	100.000	100.000	100.000	99.861	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
20	20	0.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
30	5	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
30	10	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	63.348
30	15	0.000	100.000	100.000	100.000	7.877	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
30	20	0.000	100.000	100.000	100.000	99.266	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
40	5	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
40	10	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
40	15	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
40	20	0.000	100.000	100.000	100.000	16.833	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
50	5	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	99.993
50	10	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
50	15	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
50	20	0.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	99.864
60	5	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	0.000	100.000	100.000	100.000	100.000	100.000
60	10	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
60	15	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	98.941	100.000	100.000
60	20	24.095	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
70	5	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
70	10	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
70	15	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
70	20	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
80	5	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	0.000	100.000	100.000	100.000	100.000	100.000
80	10	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
80	15	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
80	20	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
100	5	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	0.002	100.000	100.000	100.000	100.000	100.000
100	10	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
100	15	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
100	20	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
200	5	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	1.514	10.223	100.000	64.500	100.000	100.000
200	10	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	0.000	100.000	100.000	100.000	100.000	100.000
200	15	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
200	20	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000

Table 8. Maximum level of significance (*p*) for the rejection of H0 for the composite heuristics

5. Conclusions

In this paper, we have reviewed and compared the most noteworthy heuristics for the problem of flowtime minimisation in permutation flowshops. For this comparison, we have separated the simple heuristics from the composite heuristics (being the latter formed by one or more simple heuristics). With respect to the simple heuristics, two heuristics should be highlighted: the FL heuristic and the B5FT heuristic. The first is best in terms of quality of the results (particularly as the problem size grows), and the latter obtains very good results in short computation times. The fastest heuristic is the LR(1), which allows obtaining reasonably good solutions in negligible computation times.

In view of the results obtained from the comparison of the simple heuristics, we have built a composite heuristic (C1) which is a combination of the FL heuristic using the result of LR(1) as starting order. Since most of the composite heuristics incorporate a local search phase for improving the solution obtained, we have studied several local search schemes and selected the first one to build a second composite heuristic. This heuristic – named C2 – employs the solution obtained by C1, the best constructive scheme for composite heuristics, and the best local search mechanism. The experimental results carried out show that C2 outperforms the existing heuristics.

6. References

[1] Allahverdi, A, and Aldowaisan, T., 2002, New heuristics to minimize total completion time in *m*-machine flowshops, *International Journal of Production Economics*, 77, pp. 71-83.

[2] Bamberg, G., and Baur F., 1998, *Statistik*, 10th edn (Munich: Oldenbourg)

[3] Beasley, J.E., 1990, OR-Library: Distributing test problems by electronic mail, *Journal of the Operational Research Society*, 41, pp. 1069-1072.

[4] Campbell H. G., Dudek R. A. and Smith M. L., 1970, A heuristic algorithm for the n-job, m-machine sequencing problem. *Management Science*, 16, B630-B637.

[5] Dannenbring D. G., 1977, An evaluation of flow-shop sequence heuristics. *Management Science*, 23, pp. 1174-1182.

[6] Dudek, R. A. and Teuton O.F., 1964, Development of m stage decision rule for scheduling n jobs through m machines, *Operations Research*, 12.

[7] Framinan, JM and Leisten R, 2003, An efficient heuristic for flowtime minimisation in permutation flowshops, *OMEGA*, 31, pp. 311-317.

[8] Framinan, JM, Leisten, R., and Gupta JND, 2001, A review and classification heuristics for permutation flow shop scheduling with makespan objective, Technical Report 2001/02, Industrial Management, University of Seville (under revision in the Journal of the Operational Research Society).

[9] Framinan JM, Leisten R, and Rajendran C, 2003, Different initial sequences for the heuristic of Nawaz, Enscore and Ham to minimize makespan, idletime or flowtime in the static permutation flowshop, *International Journal of Production Research*, 41, pp. 121-148.

[10] Framinan JM, Leisten R, and Ruiz-Usano R, 2002, Efficient heuristics for flowshop sequencing with the objectives of makespan and flowtime minimisation, *European Journal of Operational Research*, 141, pp. 561-571.

[11] Graham, R. L, Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G., 1979, Optimisation and approximation in deterministic sequencing and scheduling: a survey, *Annals of Discrete Mathematics*, 5, pp. 287-326.

[12] Gupta, J.N.D., 1972, Heuristic algorithms for multistage flowshop scheduling problem, *AIIE Transactions*, 4, pp. 11-18.

[13] Ho, J.C., 1995, Flowshop sequencing with mean flowtime objective, *European Journal of Operational Research*, 81, pp. 571-578.

[14] Kenkel, J.L., 1996, *Introductory Statistics for Management and Economics*, 4th edn (Belmont: Duxbury)

[15] Liu, J., and Reeves, C.R., 2001, Constructive and composite heuristic solutions to the $P||\Sigma C_i$ scheduling problem, *European Journal of Operational Research*, 132, pp. 439-452.

[16] Miyazaki, S., Nishiyama, N. and Hashimoto, F., 1978, An adjacent pairwise approach to the mean flowtime scheduling problem, *Journal of the Operations Research Society of Japan*, 21, pp. 287-299.

[17] Nawaz, M., Enscore, E.E., and Ham, I., 1983, A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem, *OMEGA*, 11, pp. 91-95.

[18] Pinedo, M., 1995, *Scheduling: Theory, algorithms, and Systems*. Englewood Cliffs, New Jersey: Prentice Hall.

[19] Rajendran, C., 1993, Heuristic algorithm for scheduling in a flowshop to minimise total flowtime, *International Journal of Production Economics*, 29, pp. 65-73.

[20] Rajendran, C., and Chaudhuri, D., 1991, An efficient heuristic approach to the scheduling of jobs in a flowshop, *European Journal of Operational Research*, 61, pp. 318-325.

[21] Rajendran, C., and Ziegler, H. 1997, An efficient heuristic for scheduling in a flowshop to minimize total weighted flowtime of jobs, *European Journal of Operational Research*, 103, pp. 129-138.

[22] Taillard E., 1990, Some efficient heuristic methods for the flow-shop sequencing problem, *European Journal of Operational Research*, 47, pp. 65-74.

[23] Taillard, E., 1993, Benchmark for basic scheduling problems, *European Journal of Operational Research*, 64, pp. 278-285.

[24] Wang, C., Chu, C., and Proth, J.M., 1997, Heuristic approaches for $n|m|F|\Sigma C_i$ scheduling problems, *European Journal of Operational Research*, 96, pp. 636-644.

[25] Woo, D.S., and Yim, H.S., 1998, A heuristic algorithm for mean flowtime objective in flowshop scheduling, *Computers and Operations Research*, 25, pp. 175-182.