Ann Oper Res (2008) 164: 29–40
DOI 10.1007/s10479-007-0257-2

Scheduling jobs on a k-stage flexible flow-shop

Carlos D. Paternina-Arboleda · Jairo R. Montoya-Torres · Milton J. Acero-Dominguez · Maria C. Herrera-Hernandez

Published online: 9 November 2007

Abstract We consider the problem of makespan minimization on a flexible flow shop with k stages and m_s machines at any stage. We propose a heuristic algorithm based on the identification and exploitation of the bottleneck stage, which is simple to use and to understand by practitioners. A computational experiment is conducted to evaluate the performance of the proposed method. The study shows that our method is, in average, comparable with other bottleneck-based algorithms, but with smaller variance, and that it requires less computational effort.

Keywords Flexible flow-shop · Heuristic · Computational experiments

1 Introduction

We consider the flexible flow shop (FFS) machine environment with k stages in series; at stage $s, s = 1, \ldots, k$, there are m_s identical machines in parallel. There is unlimited intermediate storage between two successive stages. Job $j, j = 1, \ldots, n$ has to be processed at each stage on any one machine. The processing times of job j at the various stages are $p_{1j}, p_{2j}, \ldots, p_{kj}$. The goal is to find a schedule without preemption that minimizes the maximum completion time of all jobs (makespan). The flexible flow shop scheduling problem

C.D. Paternina-Arboleda · M.J. Acero-Dominguez · M.C. Herrera-Hernandez
Departamento de Ingeniería Industrial, Universidad del Norte, Km 5 via Puerto Colombia, Barranquilla, Colombia

C.D. Paternina-Arboleda
e-mail: cpaterni@uninorte.edu.co

J.R. Montoya-Torres (✉)
Escuela Internacional de Ciencias Económicas y Administrativas, Universidad de La Sabana, Km. 21 Autopista Norte de Bogotá, D.C., Chia, Colombia
e-mail: jrmontoy@yahoo.com

J.R. Montoya-Torres
e-mail: jairo.montoya@unisabana.edu.co

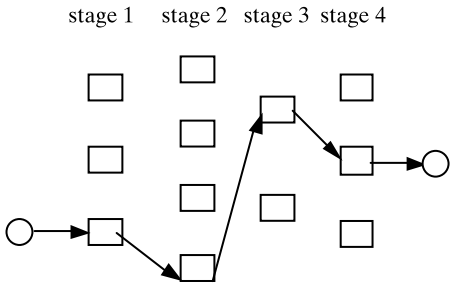

Fig. 1 Flexible (hybrid) flow-shop

is often found in electronic manufacturing environment such as IC packaging and make-to-stock wafer manufacturing. Figure 1 shows an example of job routing in a FFS with 4 stages and variable number of machines at each stage.

Since the two-stage FFS is strongly NP-hard (Gupta 1988), our problem is at least that difficult. In the literature, most of the work is on two-stage problems, for which several exact methods, involving mathematical programming formulations and branch-and-bound algorithms, as well as approximation algorithms and dedicated heuristics have been proposed, e.g. (Brah and Hunsucker 1991; Chen 1995; Haouari and M'Hallah 1997; Dessouky et al. 1998; Moursli and Pochet 2000). State-of-the-art surveys are proposed in (Chen 1994; Linn and Zhang 1999).

While interesting results have been obtained for the two-stage FFS problems, there has been less work on the k-stage ($k \geq 3$) ones to minimize the makespan. Lee and Vairaktarakis (1994) developed heuristics by extending results for a two-stage FFS and aggregating machines at each stage. Soewandi and Elmaghraby (2001) proposed heuristic algorithms for the three-stage FFS and derived lower bounds on their performance. Riane et al. (1998) considered a three-stage FFS with two machines at the second stage and one at each of the other stages, and developed a dynamic programming-based heuristic. Vignier et al. (1997) proposed a branch-and-bound approach for the makespan criterion on a k-stage FFS. The Shifting Bottleneck (SB) heuristic for the jobshop problem can be adapted to solve the FFS problem (Pinedo 1994). Other configurations of the FFS with various sets of constraints or objective functions have also been studied: availability constraints (Allaoui and Artiba 2006), flow time minimization or total completion time (Azizoglu et al. 2001; Guinet and Solomon 1996), job recirculation (Bertel and Billaut 2004), precedence constraints (Botta-Genoulaz 2000; Tang et al. 2006), state-dependent processing times (Sriskandarajah and Wagneur 1991), limited intermediate storage (Sawik 2002), pre-emption in job processing (Djellab and Djellab 2002), or even batch processing of jobs (Xuan and Tang 2007). Integrated planning and scheduling approach for shops with multiple processors in parallel has also been proposed (Riane et al. 2001). Metaheuristics algorithms have also been analyzed for various versions of the FFS problem (Portmann et al. 1998; Jin et al. 2006; Ruíz and Maroto 2006).

In this paper, we are interested in studying the general case with k stages and m_s machines at station s, where the number m_s may be different for each one of the stages, as illustrated in Fig. 1. We are proposing a heuristic algorithm based on the identification and exploitation of the bottleneck station in order to optimize the performance of the whole system (instead of local optima).

The remainder of this paper is organized as follows. Section 2 presents a mathematical formulation of the FFS using linear program. The bottleneck-based heuristic is presented in Sect. 3 as well as an illustrative example of its implementation. Section 4 is devoted to the computational study. Finally, some concluding remarks are given in Sect. 5.

2 Mathematical formulation

This section presents a linear program of the flexible flow shop problem in order to formalize the relation between the different parameters and decision variables. An interesting mixed-integer program was proposed by Jungwattanakit et al. (2006) for a special case with unrelated parallel machines and set-up times. The formulation presented here is based on the following hypothesis:

- All n jobs are independent and available for processing at the initial time.
- The production cell (stage) has sufficient capacity to store and manage the work-in-process (WIP) inventory generated during the execution of the complete set of jobs. That is, we suppose infinite storage capacity at each stage.
- One machine can process only one job at a time and one job can be processed by only one machine at any time.
- The ready times for all machines (times when the machines become available to process the set of jobs to be scheduled) are known.
- For all the jobs, the processing times at each stage are known and deterministic.
- Job set-up times are sequence-independent and are included in the job processing time at the corresponding stage.
- Travel time between consecutive stages is negligible. In instances where this assumption does not hold true, inter-stage travel can be treated as a processing step with the process time equal to the travel time.
- Pre-emption is not allowed, that is no interruption of a job processing is allowed.

To present the mathematical, the following notation is needed:

X_{jis}: a binary variable that is equal to 1 if job j is assigned to machine i at the stage s, 0 otherwise,
Y_{hjs}: a binary variable that is equal to 1 if job h precedes job j when processing at stage s, 0 otherwise,
R_j: release time of job j,
S_{js}: starting time of job j at stage s,
C_{js}: completion time of job j at stage s,
M: is a large constant ($M \to \infty$).

The problem can now be formulated as:

$$\text{Minimize } C_{\max}, \tag{1}$$

$$\text{Subject to: } C_{\max} \geq C_{js}, \quad \text{for all } s = 1, \ldots, k,\ j = 1, \ldots, n, \tag{2}$$

$$C_{js} = S_{js} + p_{sj}, \tag{3}$$

$$\sum_{i=1}^{m_s} X_{jis} = 1, \quad \text{for all } s = 1, \ldots, k,\ j = 1, \ldots, n, \tag{4}$$

$$C_{js} \leq S_{j(s+1)}, \quad \text{for } s = 1, \ldots, k-1, \tag{5}$$

$$S_{hs} \geq C_{js} - MY_{hjs}, \quad \text{for all job pairs } (h, j), \tag{6}$$

$$S_{js} \geq C_{hs} - (1 - M)Y_{hjs}, \quad \text{for all job pairs } (h, j), \tag{7}$$

$$S_{j1} \geq R_j \quad \text{for all } j = 1, \ldots, n, \tag{8}$$

$$X_{jis} \in \{0, 1\}, Y_{jhj} \in \{0, 1\} \quad \text{for all } j = 1, \ldots, n,\ i = 1, \ldots, m_s, \text{ and } s = 1, \ldots, k. \tag{9}$$

The objective function (1) considers the minimization of the makespan and constraints (2) ensure that the makespan is at least equal to the completion times of the last job. Because the objective is to minimize the makespan, constraints in this set will be tight at optimality whenever $C_{\max}$ is positive. The set of constraints (3) correspond to the computation of the completion time of job p_{sj} being the processing time of job j at stage s. Constraint (4) ensures that each job is assigned to exactly one machine at each stage. The set of constraints (5) forces to start the processing of each job only when it has been completed at the precedent stage. The set of constraints (5) and (6) ensure that only one job is on a machine of a stage at any one time. When $Y_{hjs} = 1$, and job h is before job j, the constraint in (6) is trivially satisfied. Equation (7) requires that the starting time of job j at stage s must be after the completion time for job h. When $Y_{hjs} = 0$, indicating that job j is before job h, the constraint in (7) is trivially satisfied and the starting time of job h at stage s must be the completion time for job j at stage s to satisfy (6). Constraints (8) bound the job starting times to be after job release times in the system. Finally, constraint (9) forces both variable X_{jis} and Y_{hjs} to assume binary values 0 or 1.

3 The heuristic algorithm

This section presents the proposed bottleneck-based heuristic for the flexible flow shop scheduling problem. A simple example with 7 jobs and 5 stages, each one containing 3 or 2 identical parallel machines, is also given in order to illustrate the implementation of the algorithm.

3.1 Algorithm description

The proposed method is inspired by the Theory of Constraints (TOC), which is an easy-to-understand manufacturing philosophy already implemented in various manufacturing environments. TOC states that a local optimum is not an optimum at all, and that the overall system performance is governed by the bottleneck resource (Goldratt and Cox 1992). The idea is thus to find the bottleneck stage and to optimize the whole system performance by exploiting it. The basic approach of our heuristic consists on three steps: (i) bottleneck identification, (ii) times windows and scheduling of jobs at the bottleneck stage, and (iii) scheduling of jobs at non-bottleneck stages. The algorithm is now described more in detail.

TOC-based heuristic for flexible flow shop scheduling

1. Identifying the bottleneck stage
   - For each stage s, compute the flow ratio $FR_s = \sum_{j=1}^{n} \frac{p_{sj}}{m_s}$, between the workload and the total available capacity.
   - Select the stage with max FR_s. Let denote S^B such stage.
   - Compute release times R_i^B for each job j from the bottleneck stage S^B as the sum of processing times in the stations before S^B.
   - Compute the estimated flow for each job j as the sum of flows for all the stages.
   - Compute *tails* to the bottleneck station D_i^B for each job j as the difference between the estimated flow and the sum of processing times in the stations before S^B.

2. Sequencing on the bottleneck stage S^B
   - Schedule jobs in a list by increasing order of R_j^B. If there is more than one job with the same R_j^B, then rank them by increasing value of D_j^B. If there is more than one job satisfying the criterion, then prioritize them by increasing order of processing times.
   - Schedule jobs on the machines according with the precedent ranking. If there is more than one machine available at time t, then assign the next job to the machine with less workload until time t. Compute starting and completion times for each job on the bottleneck stage.
3. Sequencing on non-bottleneck stages
   - *Stages after the bottleneck station*: proceed in a similar way as for S^B, but job j may be scheduled as soon as it is completed by the precedent stage.
   - *Stages before the bottleneck station*: in order to respect the delivery time to the bottleneck station, jobs are scheduled according to D_j^B and R_j^B, and processing times criteria explained in step 2.

3.2 Illustrative example

In order to illustrate the use of the proposed TOC-based heuristic, this section presents an simple example of a hybrid flowshop with 7 jobs and 5 stages, each one containing 3 or 2 identical parallel machines. Processing times of jobs on each stage are presented in Table 1.

The first step when applying the procedure is the computation of the mean flow ratio for each stage. Results are presented in the last row of Table 1. We can observe the stage 3 is the bottleneck stage. Hence, release times R_j^B are computed for each job j at this bottleneck stage: $R_j^B = \{12, 17, 20, 15, 14, 13, 12\}$. Tail times D_j^B of job j at this bottleneck stage are also computed: $D_j^B = \{105.5, 100.5, 102.5, 104.5, 103.5, 104.5, 105.5\}$. Now, jobs are assigned to the bottleneck station, as shown in Table 2.

So, for job $j = 1$, which is the first to be executed, we compute $R_1 = 12$ because $(p_{11} + p_{12} = 12)$, and its delivery time is computed as $\{\text{mean flow time} - (p_{14} + p_{15})\} = 117.5 - (5 + 7) = 105.5$.

The next step in the procedure is to schedule jobs on the stage 2, which is the immediate predecessor of stage 3. Table 3 presents such schedule.

Following the procedure, jobs are now scheduled on stages preceding the bottleneck stage until arriving to the first stage. Then, jobs are scheduled on stages following the bottleneck stage until arriving to the last station. In order to adjust the global schedule, all jobs are

Table 1 Processing times for the example

Stages		1			2		3		4			5	
Machines		1	2	3	1	2	1	2	1	2	3	1	2
Jobs	1	7	7	7	5	5	10	10	7	7	7	5	5
	2	10	10	10	7	7	10	10	10	10	10	7	7
	3	12	12	12	8	8	12	12	7	7	7	8	8
	4	8	8	8	7	7	8	8	8	8	8	5	5
	5	7	7	7	7	7	8	8	7	7	7	7	7
	6	5	5	5	8	8	15	15	5	5	5	8	8
	7	9	9	9	3	3	10	10	9	9	9	3	3
Mean flow		19.33			22.50		36.50*		17.67			21.50	

Table 2 Schedule for stage 3, the bottleneck station

j	M1	M2	R_j	D_j	M1	M2	M1		M2		T
	[j]				P_{j1}	P_{j2}	[s,	c]	[s,	c]	
1	[1]		12	105.5	10	10	12	22			12
2	[6]		17	100.5	10	10	37	47			37
3		[7]	20	102.5	12	12			38	50	38
4		[5]	15	104.5	8	8			30	38	30
5		[4]	14	103.5	8	8			22	30	22
6	[3]		13	104.5	15	15	22	37			22
7		[2]	12	105.5	10	10			12	22	12

Table 3 Schedule for stage 2, the immediate predecessor of the bottleneck station

j	M1	M2	R_j	D_j	M1	M2	M1		M2		T
	[j]				P_{j1}	P_{j2}	[s,	c]	[s,	c]	
1		[7]	7	12	5	5			7	12	12
2		[2]	10	37	7	7			30	37	37
3	[1]		12	38	8	8	30	38			38
4	[3]		8	30	7	7	23	30			30
5	[4]		7	22	7	7	15	22			22
6		[5]	5	22	8	8			14	22	22
7	[6]		9	12	3	3	9	12			12

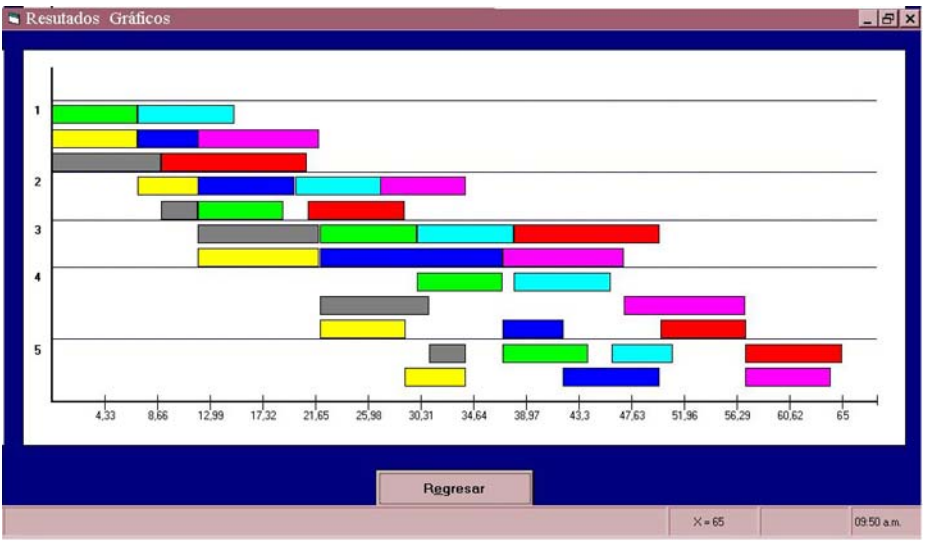

Fig. 2 Gantt chart for the data of the illustrative example

shifted to the right so that the first job is started at time zero. Finally, the makespan in computed. For this example, the makespan is 65. The Gantt chart is presented in Fig. 2.

4 Computational study

For comparison purpose, an experimental study was performed in order to compare the behaviour of the proposed bottleneck-based heuristic algorithm with other well-known heuristics for this scheduling problem, as well as with optimal solutions for small instances and with a lower bound for large instances. The first experiment was performed using the same instances proposed in Acero et al. (2004). The second study was performed using the same instances proposed in Sivrikaya Serifoglu and Ulusoy (2004). The description of the experiments in presented next.

4.1 Comparison of the bottleneck heuristic with the optimal solutions and other heuristics for small instances

A computational experiment was conducted in order to evaluate the performance of the TOC-based method, and to compare with both the famous shifting bottleneck heuristic (SBG), and a hybrid shifting bottleneck-local search (SB-LS) heuristic (Pinedo 1994). Algorithms were implemented in C++ language, and tests were performed on a PC Pentium 4 (750 MHz). A total of 7 sets of instances, with number of jobs ranging from 5 to 9, number of stages ranging from 3 to 5, and number of machines within stages ranging from 2 to 4, have been generated and solved by the three heuristics algorithms as well as an optimal one. The processing times of the jobs were randomly generated from a uniform distribution from 1 to 100. These algorithms were evaluated on the basis of the deviation from the optimal solution using the formula:

$$\text{dev} = \frac{C_{\max}^{\text{heur}} - C_{\max}^{\text{opt}}}{C_{\max}^{\text{opt}}} \times 100. \tag{10}$$

We were interested in looking into the mean and the interval of deviation from the optimal solution obtained for instance using the 0-1 linear program presented in Sect. 2. It is to notice that, because of the NP-completeness of the flexible flow shop problem, finding optimal solutions for large instances is very computationally expensive. For that reason, the set of instances is limited to a small number of jobs and machines within each stage. These results are shown in Table 4, together with the worst-case obtained for the tested instances. Our heuristic outperformed the SBG and SB-LS in 71.4% of the cases. Some makespan values obtained with the proposed heuristic are a little higher than to the one of the Shifting Bottleneck algorithm (SBG). When computing the interval of the deviation form the optimal solution, we obtain, respectively, makespans of [0, 5.23]; [0, 6.98] and [0, 21.5] for the proposed method, the SBG and SB-LS heuristics. So that, the proposed method has a smaller interval of deviation over the total set of instances tested. These results allow us to conclude that the proposed algorithm produces results comparable with those given by other bottleneck-based methods proposed by other researchers.

An evaluation of the CPU time requirements of these algorithms is also presented in Table 4. The highest value obtained for our heuristic was 0.23 seconds, while those for the SBG and SB-LS heuristics were 11 and 45 seconds, respectively. It is to note that the computational requirement of our algorithm increases with the number of stages, but it always takes less time than the others (with a factor between 17.4 and 256.4 times lower).

Table 4 Comparison with the optimum and with other heuristics for small instances

Instance	This paper (TOC)		SBG		SB-LS		Optimum
	$C_{\max}^{\text{TOC}}$	CPU sec.	$C_{\max}^{\text{SBG}}$	CPU sec.	$C_{\max}^{\text{SB-LS}}$	CPU sec.	$C_{\max}^{\text{OPT}}$
I1	172	0.22	172	9	209	45	172
I2	145	0.22	140	9	161	4	137
I3	181	0.23	172	10	172	4	172
I4	557	0.22	579	11	578	4	541
I5	378	0.17	380	11	451	43	373
I6	198	0.16	202	8	194	3	189
I7	36	0.16	36	8	36	4	34
Average		0.20		9.43		15.29	
Max.		0.23		11		45	

4.2 Comparison of the bottleneck heuristic with a lower bound and other heuristics for large instances

In order to compare our algorithm with the Shifting Bottleneck-based heuristics for large instances, a set of experiments was performed using a similar experiment to the one proposed by Oguz et al. (2004) and Sivrikaya Serifoglu and Ulusoy (2004). The actual values of data used for this test can be found on the website at http://www.benchmark.ibu.tr/mpt-hfsp. Problem data files given in such website are named as follows: "*nm*ha'InstNo'.dat", where 'h' stands for 'hundred' and denotes that processing time requirements are from [1,100], "a" denotes the data set name (which implies that the number of machines at the stages may not be the same), "InstNo" denotes the number of the problem instance. For example, 52ha1.dat denotes the first problem of the data set of type "a" with 5 jobs and 2 stages. The results of this experiment are presented in Table 5. These results were obtained treating instances including 5, 10 and 20 jobs, as well as 2, 5, 8, and 10 stages for each case. These larger instances were experimented under same operational conditions than for small ones.

As indicated previously, since the hybrid flow shop problem is NP-complete, obtaining optimal solutions for large instances in extensive time consuming. Instead of computing optimal solutions, Santos et al. (1995) developed the lower bound shown below, where the set of jobs is denoted by J and the set of stages is denoted by M:

$$LB = \max_{s\in M}\left\{\min_{j\in J}\sum_{i=1}^{s-1} p_{ij} + \frac{1}{m_s}\sum_{j\in J} p_{kj} + \min_{j\in J}\sum_{i=s+1}^{k} p_{ij}\right\}. \tag{11}$$

According to each heuristic's capability, our algorithm is able to process 20 jobs top as well as SB-LS one, meanwhile SBG is able to process up to 30 jobs more (50 jobs top in total), but taking much more time than 5200 CPU seconds. Therefore, it clearly becomes computationally expensive. In general terms, computational requirements of our algorithm for treating larger instances increases with the number of stages, as well as in case of small ones, highlighting it goes on taking less time than SB-based heuristics for delivering $C_{\max}$ value for each treatment. The same performance was shown after analyzing results in Table 4 previously. Considering the lower bound computation, last column in Table 5 presents the values obtained for this lower bound. Comparing the proposed heuristic, we can observe an

Table 5 Comparison among heuristics for large instances

Instance	This paper (TOC)		SBG		SB-LS		Stage-based lower bound
	$C_{\max}^{\mathrm{TOC}}$	CPU sec.	$C_{\max}^{\mathrm{SBG}}$	CPU sec.	$C_{\max}^{\mathrm{SB\text{-}LS}}$	CPU sec.	
52ha1	200	0.016	189	1	231	2	182
102ha1	623	0.031	627	1	627	3	623
202ha1	567	0.031	541	4	553	5	541
55ha1	298	0.062	298	1	298	2	293
105ha1	473	0.125	408	4	464	5	374
205ha1	1063	0.125	950	40	1037	14	972
58ha1	579	0.125	478	1	553	3	393
105ha1	707	0.219	734	5	783	6	652
208ha1	1541	0.219	1475	88	1513	28	1397
510ha1	899	0.235	923	7	892	9	824
1010ha1	1475	0.234	1326	128	1367	32	1210
Average		0.129		25.45		9.91	
Max.		0.235		128		32	

average deviation of 13% from the LB value, having a maximum of 47% and obtained the same value for one instance.

Motivated by those results, additional experiments were performed in order to analyze the average deviation of the makespan for large instances given by the TOC-heuristic against the lower bound in comparison with the other heuristics. The experiment was performed for 20-jobs instances with processing times ranging from 1 to 100 and 2, 5, 8 and 10 stages. The number of machines at each stage was the same as considered for 20-jobs instances in Table 5. For each combination of parameters, 10 replications were performed, thus giving a total of 40 instances for this particular experiment. Results are summarized in Tables 6 and 7. Table 6 presents the average deviation of the corresponding heuristic computed against the lower bound, similarly as described by (10). The CPU time (in seconds) is also given in average. With respect to the absolute makespan value, our algorithm gave a better makespan in more than 42% of the instances, compared with SB-LS heuristic, and almost 8% compared with SBG. In addition, as observed in Table 6, the maximal deviation of the makespan given by the proposed procedure, against the lower bound, is about 10%. It is to note, however, that our heuristic is quite competitive: its makespan is only 6% and 1.8% higher than the one respectively given by SBG and SB-LS. Besides, a highlight compared with those heuristic, is that our procedure requires less amount of computational effort for given quite competitive solutions. The TOC-procedure is, in average, more 99% faster than the other two heuristics. In addition, it can be seen in Table 7, which presents the results of the hypothesis test for mean differences, that there is no significant difference between heuristics' performance with a 99% confidence level. These results show the competitiveness of our procedure compared with other that already exist in the literature.

Using this last experiment, the performance of our heuristic can be compared with estimated optima. As stated by Sivrikaya Serifoglu and Ulusoy (2004), the discussion of the statistical estimation of optimal values for combinatorial optimization problems was proposed by Rardin and Uzsoy (2001) as a way to evaluate the performance of heuristics. The basis of the estimation method applied here is a result by Fisher and Tippett (1928) about the distribution of least values, which briefly states that the least of M random variables with

Table 6 Average values the 10-replications experiment among heuristics for large instances

No. of stages	This paper (TOC)		SBG		SB-LS		LB (avg.)
	% dev	CPU sec.	% dev	CPU sec.	% dev	CPU sec.	
2	2.6%	0.017	0.1%	5.5	0.5%	6	495.1
5	9.9%	0.0793	0.5%	59	8.0%	21	1075.6
8	8.2%	0.1424	3.5%	155.6	7.4%	36	1299.6
10	10.7%	0.15	6.3%	239.1	8.7%	51	1309

Table 7 Summary of results of the hypothesis test for the 10-replications large instances experiment

	Mean values for makespan			99% confidence intervals			
	This paper (TOC)	SBG	SB-LS	TOC *versus* SBG		TOC *versus* SB-LS	
				Lower value	Upper value	Lower value	Upper value
2 stages	507.4	480.9	498	−7701.92	7754.917	−7156.179	7174.979
5 stages	1183.2	1081.1	1161	−19636.6	19840.75	−19743.54	19787.94
8 stages	1406.2	1345.7	1394.4	−24263.6	24384.61	−20125.18	20148.78
10 stages	1449.4	1392.4	1423.1	−21252.8	21366.75	−19494.33	19546.93

a common distribution on real numbers greater than or equal to a is asymptotically distributed Weibull with a being the location parameter. For the estimation process, K independent samples each consisting of T_k objective values are obtained. Let z_i be the minimum value of sample $i, i = 1, \ldots, K$. These sample minima, assumed to be independent, are sorted so that $z_{(1)} \leq z_{(2)} \leq \cdots \leq z_{(K)}$. As shown by Zanakis (1979), the location parameter a of the Weibull distribution can then be estimated as:

$$\hat{a} = \frac{z_{(1)}z_{(K)} - z_{(2)}^2}{z_{(1)} + z_{(K)} - 2z_{(2)}}. \tag{12}$$

This estimate of a also provides an estimate of the optimum solution value.

Ghashghai and Rardin (1998) make use of the solutions generated by their GA algorithm to estimate the optima for the problem of finding sub-graphs that meet survivability requirements. Ovacık et al. (2000) integrate such an estimation procedure with a simulated annealing heuristic and apply it on a single machine maximum lateness problem with sequence dependent set-up times. Sivrikaya Serifoglu and Ulusoy (2004) used the estimation technique to analyze a particular multiresource flowshop problem with resource utilization constraints.

Here, estimated optima analysis is applied to the 10-replication 20-jobs instances with processing times ranging from 1 to 100, as described previously. Table 8 summarizes the results. The column % dev corresponds to the deviation from the estimate computed using (10) as previously explained. These few experiments show that the deviation, in percentage, of the proposed TOC heuristic from the estimated optimum makespan values is ranging from 15% to 27%, with being 20% the average.

Table 8 Comparison of TOC procedure against estimated optima for some large instances

Stages	Avg. $C_{\max}^{\mathrm{TOC}}$ (this paper)	Estimated optima ($\hat{C}_{\max}$)	% dev
2	507.4	415.4	22%
5	1183.2	928.5	27%
8	1406.2	1226.6	15%
10	1449.4	1253.5	16%
Average	1136.5	956	20%

5 Concluding remarks

The proposed bottleneck-based algorithm for the flexible flowshop problem produces results comparable with those proposed by other researchers. The performance of the proposed procedure was shown to be statistically similar to the Shifting Bottleneck-based procedures of the literature. Since it uses a strategy based on well-known principles of the Theory of Constraints (TOC), the proposed heuristic is simple to use and simple to understand by practitioners. In particular, by exploiting the bottleneck stage, our algorithm looks at a global system optimization driving to a better utilization of production resources, instead to use ranking rules normally employed from the parallel machine environment, as suggested by other researchers (i.e. LPT-based). Further work can be performed on to relax some constraints of the conventional FFS problem (i.e. including limited inter-stage storage or considering explicit transportation times) in order to adapt the algorithm to more real-life production scheduling problems.

Acknowledgements Part of this work was performed while the second author was visiting the Department of Industrial Engineering at Universidad del Norte, in Barranquilla, Colombia. Authors want to thanks the anonymous referees for their comments that allow improving the presentation of this paper.

References

Acero, M., Montoya-Torres, J. R., & Paternina-Arboleda, C. D. (2004). Scheduling jobs on a *k*-stage flexible flow shop. In A. Oulamara & M. C. Portmann (Eds.), *Proceedings of the ninth international workshop on project management and scheduling* (pp. 242–245).

Allaoui, H., & Artiba, A. (2006). Scheduling two-stage hybrid flow shop with availability constraints. *Computer and Operation Research*, *33*, 1399–1419.

Azizoglu, M., Cakmak, E., & Kondakci, S. (2001). A flexible flowshop problem with total flow time minimization. *European Journal of Operational Research*, *132*, 528–538.

Bertel, S., & Billaut, J. C. (2004). A genetic algorithm for an industrial multiprocessor flow shop scheduling problem with recirculation. *European Journal of Operational Research*, *159*, 651–662.

Botta-Genoulaz, V. (2000). Hybrid flow shop scheduling with precedence constraints and time lags to minimize maximum lateness. *International Journal of Production Economics*, *64*, 101–111.

Brah, S. A., & Hunsucker, J. L. (1991). Branch and bound algorithm for the flow shop with multiple processors. *European Journal of Operation Research*, *51*, 88–99.

Chen, B. (1994). Scheduling multiprocessor flowshops. In *Advances in optimization and approximation*. Dordrecht: Kluwer.

Chen, B. (1995). Analysis of classes of heuristics for scheduling a two-stage flow shop with parallel machines at one stage. *Journal of the Operational Research Society*, *46*, 234–244.

Dessouky, M., Dessouky, M., & Verma, S. (1998). Flowshop scheduling with identical jobs and uniform parallel machines. *European Journal of Operational Research*, *109*, 620–631.

Djellab, H., & Djellab, K. (2002). Preemptive hybrid flowshop scheduling problem of interval orders. *European Journal of Operational Research*, *137*, 37–49.

Fisher, R., & Tippett, L. (1928). Limiting forms of the frequency distribution of the largest or smallest member of a sample. *Proceedings of the Cambridge Philosophical Society*, *24*, 180–190.

Ghashghai, E., & Rardin, R. L. (1998). Using a hybrid of exact and genetic algorithms to design survivable networks. In *Working paper school of industrial engineering*, Purdue University.

Goldratt, E., & Cox, J. (1992). *The goal*. North River Press.

Guinet, A., & Solomon, M. (1996). Scheduling hybrid flowshops to minimize maximum tardiness or maximum completion time. *International Journal of Production Research*, *34*, 1643–1654.

Gupta, J. (1988). Two stage hybrid flow shop scheduling problem. *Journal of the Operations Research Society*, *38*, 359–364.

Haouari, M., & M'Hallah, R. (1997). Heuristic algorithms for the two-stage hybrid flowshop problem. *Operations Research Letters*, *21*, 43–53.

Jin, Z., Yang, Z., & Ito, T. (2006). Metaheuristic algorithms for the multistage hybrid flowshop scheduling problem. *International Journal of Production Economics*, *100*, 322–334.

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., & Werner, F. (2006). Sequencing heuristics for flexible flow shop problems with unrelated parallel machines and setup times. In *Proceedings of the 2006 IE network national conference*, Bangkok, 18–19 December 2006 (Session F53, pp. 1–8).

Lee, C. Y., & Vairaktarakis, G. (1994). Minimizing makespan in hybrid flowshops. *Operations Research Letters*, *16*, 149–158.

Linn, R., & Zhang, W. (1999). Hybrid flow shop scheduling: A survey. *Computers and Industrial Engineering*, *37*, 57–61.

Moursli, O., & Pochet, Y. (2000). A branch and bound algorithm for the hybrid flow shop. *International Journal of Production Economics*, *64*, 113–125.

Oguz, C., Zinder, Y., Dob, V. H., Janiak, A., & Lichtenstein, M. (2004). Hybrid flow shop scheduling problems with multiprocessor task systems. *European Journal of Operational Research*, *152*, 115–131.

Ovacık, I. M., Rajagopalan, S., & Uzsoy, R. (2000). Integrating interval estimates of global optima and local search methods for combinatorial optimisation problems. *Journal of Heuristics*, *6*, 481–500.

Pinedo, M. (1994). *Scheduling: Theory, algorithms, and systems*. Englewood Cliffs: Prentice-Hall.

Portmann, M. C., Vignier, A., Dardilhac, D., & Dezalay, D. (1998). Branch and bound crossed with GA to solve hybrid flowshops. *European Journal of Operational Research*, *107*, 389–400.

Rardin, R. L., & Uzsoy, R. (2001). Experimental evaluation of heuristic optimisation algorithms: A tutorial. *Journal of Heuristics*, *7*, 261–304.

Riane, F., Artiba, A., & Elmaghraby, S. E. (1998). A hybrid three-stage flexible flowshop problem: Efficient heuristics to minimize makespan. *European Journal of Operational Research*, *109*, 321–329.

Riane, F., Artiba, A., & Iassinovski, S. (2001). An integrated production planning and scheduling system for hybrid flowshop organizations. *International Journal of Production Economics*, *74*, 33–48.

Ruíz, R., & Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research*, *169*, 781–800.

Santos, D. L., Hunsucker, J. L., & Deal, D. E. (1995). Global lower bounds for flowshops with multiple processors. *European Journal of Operational Research*, *80*, 112–120.

Sawik, T. (2002). An exact approach for batch scheduling in flexible flow lines with limited intermediate buffers. *Mathematical and Computer Modelling*, *36*, 461–471.

Sivrikaya Serifoglu, F., & Ulusoy, G. (2004). Multiprocessor task scheduling in multistage hybrid flow-shops: a genetic algorithm approach. *Journal of the Operational Research Society*, *55*, 504–512.

Soewandi, H., & Elmaghraby, S. E. (2001). Sequencing three-stage flexible flowshops with identical machines to minimize makespan. *IIE Transactions*, *33*, 985–993.

Sriskandarajah, C., & Wagneur, E. (1991). Hierarchical control of the two processor flow-shop with state dependent processing times: Complexity analysis and approximate algorithms. *INFOR, Canadian Journal of Information Systems and Operational Research*, *29*, 193–205.

Tang, L., Xuan, H., & Liu, J. (2006). A new Lagrangian relaxation algorithm for hybrid flowshop scheduling to minimize total weighted completion time. *Computers and Operations Research*, *33*, 3344–3359.

Vignier, A., Commandeur, P., & Proust, C. (1997). New lower bound for the hybrid flowshop scheduling problem. In *Proceedings of the 1997 IEEE conference on emerging technologies and factory automation* (pp. 446–451).

Xuan, H., & Tang, L. (2007). Scheduling a hybrid flowshop with batch production at the last stage. *Computers and Operations Research*, *34*, 2718–2733.

Zanakis, S. H. (1979). A simulation study of some simple estimators of the three-parameter Weibull distribution. *Journal of Statistical Computing and Simulation*, *9*, 419–428.