International Journal of Production Research
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tprs20

A comparison of metaheuristic procedures to schedule jobs in a permutation flow shop to minimise total earliness and tardiness

Jeffrey Schaller [a] & Jorge M.S. Valente [b]

[a] Department of Business Administration, Eastern Connecticut State University, Willimantic, CT, USA

[b] Faculdade de Economia, Universidade do Porto, Porto, Portugal

Version of record first published: 02 Apr 2012.

To cite this article: Jeffrey Schaller & Jorge M.S. Valente (2013): A comparison of metaheuristic procedures to schedule jobs in a permutation flow shop to minimise total earliness and tardiness, International Journal of Production Research, 51:3, 772-779

To link to this article: http://dx.doi.org/10.1080/00207543.2012.663945

International Journal of Production Research
Vol. 51, No. 3, 1 February 2013, 772–779

Taylor & Francis
Taylor & Francis Group

A comparison of metaheuristic procedures to schedule jobs in a permutation flow shop to minimise total earliness and tardiness

Jeffrey Schaller[a]* and Jorge M.S. Valente[b]

[a]*Department of Business Administration, Eastern Connecticut State University, Willimantic, CT, USA;*
[b]*Faculdade de Economia, Universidade do Porto, Porto, Portugal*

(*Received 17 February 2011; final version received 1 February 2012*)

This paper considers the problem of scheduling jobs in a permutation flow shop with the objective of minimising total earliness and tardiness. A genetic algorithm is proposed for the problem. This procedure and five other procedures were tested on problem sets that varied in terms of number of jobs, machines and the tightness and range of due dates. It was found that the genetic algorithm procedure was consistently effective in generating good solutions relative to the other procedures.

Keywords: scheduling; heuristics; flow shop; earliness and tardiness

1. Introduction

The widespread adoption of lean production methods has caused early delivery of products as well as tardy delivery to be viewed as undesirable. Early deliveries result in unnecessary inventory that ties up cash as well as space and resources needed to maintain and manage inventory. Products that are delivered late can cause penalties such as lost sales and loss of customer good will. This paper addresses this trend by considering an objective that sums the penalties for earliness and tardiness for a set of jobs to be processed in a flow shop. Most research on flow shops has assumed that the sequence of jobs to process will be the same on each machine. These schedules are referred to as permutation schedules. This is done for two reasons. First it simplifies the computational effort and second it is often not practical to change the sequence of jobs from one machine to the next. In this research only permutation schedules are considered.

Formally, suppose there is a set of n jobs to be processed in a flow shop with M machines. Let d_j be the due date of job j ($j=1,\ldots,n$). Let p_{jm} and C_{jm} represent the processing time and completion time of job j ($j=1,\ldots,n$) on machine m ($m=1,\ldots,M$). The earliness of job j, E_j, is defined as $E_j=\max\{d_j-C_{jM},\,0\}$, for $j=1,\ldots,n$ and the tardiness of job j, T_j, is defined as $T_j=\max\{C_{jM}-d_j,\,0\}$, for $j=1,\ldots,n$. The objective function, Z, can be expressed as $Z=\sum_{j=1}^{n} E_j+T_j$.

Since the objective in the problem is non-regular, inserting idle time into a schedule for the jobs can help to reduce the earliness of some jobs and thus improve the objective. However, in certain production environments, the insertion of idle time may actually be undesirable, or even impossible. For instance, idle time should be avoided when the capacity of the shop is limited when compared with the demand. Also, idle time should not be inserted when the machine has a high operating cost, and/or when starting a new production run involves high setup costs or times. Some specific examples of production environments where the insertion of idle time is undesirable have been given by Landis (1993) and Korman (1994). In this research only schedules without unforced inserted idle time are considered. Therefore if the job to be sequenced in position j is denoted as $[j]$ and $C_{[0]1}=0$ then $C_{[j]1}=C_{[j-1]1}+p_{[j]1}$ and $C_{[j]m}=\max\{C_{[j]m-1},\,C_{[j-1]m}\}+p_{[j]m}$ for $m=2,\ldots,M$.

A large number of papers have been published on scheduling models with earliness and tardiness costs. Baker and Scudder (1990) provide an excellent survey of the initial work on early/tardy scheduling. A recent survey of multi-criteria scheduling which includes problems with earliness and tardiness penalties is given in Hoogeveen (2005). Most of the research with an objective based on early and tardy job completion costs deals with single-machine environments. Recent research for single-machine environments with an early/tardy objective and no idle

*Corresponding author. Email: schallerj@ecsu.ctstateu.edu

ISSN 0020–7543 print/ISSN 1366–588X online

http://dx.doi.org/10.1080/00207543.2012.663945
http://www.tandfonline.com

time is summarised in Valente (2009). Scheduling models with inserted idle time, on the other hand, are reviewed in Kanet and Sridharan (2000).

Only five papers address objectives based on earliness and tardiness costs in flow shops. Moslehi *et al*. (2009) present an optimal procedure for minimising the sum of maximum earliness and tardiness in a two-machine flow shop. Chandra *et al*. (2009) present approaches for permutation flow shop scheduling with earliness and tardiness penalties when all the jobs have a common due date. Madhushini *et al*. (2009) present branch-and-bound algorithms for scheduling permutation flow shops for a variety of objectives including minimising earliness and tardiness without inserted idle time. Zegordi *et al*. (1995) present a simulated annealing algorithm with specialised knowledge for scheduling permutation flow shops to minimise the sum of weighted earliness and tardiness without inserted idle time. Rajendran (1999) presents heuristics for scheduling a kanban flow shop to minimise the sum of weighted flow time, weighted tardiness and weighted earliness of containers.

Objectives that are based on meeting distinctive due dates are similar to the objective considered in this paper. One such objective is minimising total tardiness. The reason the objectives are related is that our objective includes total tardiness. Several heuristics have been found to be effective for scheduling flow shops to minimise total tardiness, and other heuristics have been successfully implemented for single-machine scheduling for both the total earliness and tardiness objective and the total tardiness objective. Vallada *et al*. (2008) found that neighbourhood searches developed by Kim *et al*. (1996) and simulated annealing algorithms developed by Parthasarathy and Rajendran (1997) and Hasiji and Rajendran (2004) were effective for minimising total tardiness in flow shops. Framinan and Leisten (2008) developed a variable greedy algorithm for the problem and found it to be more effective in minimising total tardiness than the simulated annealing algorithms in flow shops. Genetic algorithms have been effective for minimising total tardiness in flow shops (Vallada and Ruiz 2010) and minimising total earliness and tardiness on a single machine (Singh 2010). Holthaus and Rajendran (2005) developed a fast ant colony algorithm that was found to be effective for scheduling a single machine to minimise total tardiness.

In this paper we propose a genetic algorithm for the problem. This algorithm is described in the next section. Section 3 describes the computational tests and results and Section 4 concludes the paper.

2. Proposed genetic algorithm

A genetic algorithm is a metaheuristic search procedure that uses a multiple-solution search technique. This approach has been found to quickly generate good solutions for a wide variety of scheduling problems. In a genetic algorithm an initial population of chromosomes is first created and then successive populations (or generations) of chromosomes are created using some methodology until a stopping condition is met. A chromosome corresponds to a solution for the problem. For this problem solutions can be represented by a permutation sequence of the jobs. The jth gene in a chromosome corresponds to the job in the jth position of a sequence.

Elements of the genetic algorithm proposed for the problem considered in this research include the initialisation of the population, a selection mechanism, a mating operator, a mutation operator, local searches, a generational scheme and stopping criteria. The genetic algorithm used in this research is referred to as GA.

2.1 *Initial population*

An initial population of 40 (population size) unique chromosomes (permutation sequences) is created. The procedure creates 38 chromosomes randomly, one chromosome using the earliest due date (*EDD*) dispatching rule and one chromosome using the NEH_{edd} heuristic (Kim 1993). Each of the randomly generated chromosomes (sequences) is created by first generating a random number between 0 and 1 for each job and then sorting the numbers corresponding to each job (lowest to highest) to create a sequence of jobs (chromosome). If the sequence is not already in the initial population it is added.

2.2 *Selection operator*

The genetic algorithm in this research uses a selection operator called n-tournament. With this approach a percentage of individuals, a parameter called ''pressure'', is selected and the individual with the lowest total earliness and tardiness among these individuals is selected for the mating process. The pressure parameter was set at $(10 + \lfloor n/10 \rfloor * 5)\%$.

2.3 *Mating operator*

Uniform order-based (UOB) crossover is used as the mating operator. The goal of this operator is to generate two good individuals, called offspring, from two progenitors (chromosomes) that were selected using the selection operator described in the previous subsection. The UOB crossover operator tries to copy jobs at each position from the first parent with a probability p_{u1} to a child. The unfilled positions are then filled from the unused jobs in the order they appear in the second parent. The operator then creates a second child by attempting to copy jobs at each position from the second parent with a probability p_{u2} to a child. The unfilled positions are then filled from the unused jobs in the order they appear in the first parent. The value of p_{u1} is set at $ET(p_2) / (ET(p_1) + ET(p_2))$ and p_{u2}, at $ET(p_1) / (ET(p_1) + ET(p_2))$, where $ET(p_1)$ and $ET(p_2)$ are the total earliness and tardiness associated with the sequences of parents 1 and 2, respectively. With this approach, better parents tend to have more jobs copied to children.

For example, consider the sequences associated with the following two parents:

$$\text{Parent 1: } 1-2-3-6-5-4$$

$$\text{Parent 2: } 4-5-2-3-1-6$$

If after the first step child 1 inherits positions 2, 5 and 6 from the first parent so that the partially formed child is $(\_, 2, \_, \_, 5, 4)$ then the resulting sequence for the first child is:

$$3-2-1-6-5-4.$$

The genetic algorithm in this research performs the mating operator with the two selected parents with a probability of 35%.

2.4 *Mutation operator*

The genetic algorithm in this research uses a swap mutation operator. With this operator each job in the permutation sequence is selected with a 3% probability and swaps positions with a job in a randomly chosen different position in the sequence.

2.5 *Local searches*

Local searches are used in the genetic algorithm proposed in this research. The first local search is a job insertion search. In this search a job is removed from the sequence and is inserted into all n positions. The final position of the job is the position that results in the lowest total tardiness. This search is carried out in all n jobs of each generated offspring after the mating and mutation operators have been applied with a probability of 12%. This local search is also applied to the best individual found in the initial generation of the population. If the insertion search is not applied, then one pass of an adjacent pairwise exchange procedure is carried out with a probability of 50%.

2.6 *Generational scheme*

The genetic algorithm proposed in this research has two criteria for accepting generated offspring into the population: (1) an offspring must be better (have a lower total earliness and tardiness) than the worst individual in the population; and (2) the offspring has a unique sequence (there are no other individuals in the population with the same sequence).

2.7 *Stopping criteria*

The proposed genetic algorithm in this research has two stopping criteria: (1) if a solution is found with zero total earliness and tardiness the procedure stops; and (2) if a time limit is exceeded when a generation is completed the procedure stops.

3. Computational test

The proposed genetic algorithm is tested on randomly generated problems of various sizes in terms of the number of jobs and number of machines and under various conditions of due date range and tightness. Also, as a basis of comparison, several other procedures are included in the computational test. An optimal branch-and-bound procedure (O) was used to generate optimal solutions for small-sized problems ($n = 8$, 10 and 12).

The following heuristics are included: the simulated annealing algorithms of Zegordi *et al.* (1995) and Parthasarathy and Rajendran (1997), a neighbourhood search based on the neighbourhood searches of Kim *et al.* (1996), Framinan and Leisten (2008)'s variable greedy algorithm and the fast ant colony algorithm developed by Holthaus and Rajendran (2005). The simulated annealing algorithm developed by Zegordi *et al.* (1995) considers weighted earliness and tardiness for flow shops and is modified in this research by setting the weights for earliness and tardiness for each job to 1. The procedure is referred to as ZSA in this research. The procedures by Kim *et al.* (1996), Parthasarathy and Rajendran (1997) and Framinan and Leisten (2008) minimise total tardiness in flow shops and are modified to consider the total earliness and tardiness objective in this research. Kim *et al.* (1996)'s neighbourhood search (which includes both the insertion and swap operators) is referred to as NS, Parthasarathy and Rajendran (1997)'s simulated annealing procedure is referred to as PRSA and Framinan and Leisten (2008)'s variable greedy algorithm is referred to as FLVG. The fast ant colony algorithm by Holthaus and Rajendran (2005) was modified for the flow shop environment and to consider the earliness and tardiness objective. This procedure is referred to as FACO.

3.1 *Data and performance measures*

The heuristic procedures described in the previous subsection and the genetic algorithm described in Section 2 were tested on problems of various sizes in terms of the number of jobs and the number of machines for 15 sets of distributions of due date range and tightness. Each problem set consists of 10 problems. The problems within a problem set have the same number of jobs and machines, and the due dates for the jobs are generated using the same distribution. Eleven levels of number of jobs (n) to be scheduled were tested: $n = 8$, 10, 12, 15, 20, 25, 30, 40, 50, 75 and 100. Three levels of number of machines (M) were tested: $M = 5$, 10 and 20. The processing times of the jobs for each machine were generated using a uniform distribution over the integers 1 and 100. The due dates for the jobs were also randomly generated using a uniform distribution over the integers $MS\,(1 - r - R/2)$ and $MS\,(1 - r + R/2)$, where MS is an estimated makespan found for the problem using the makespan lower bound proposed in Taillard (1993), and R and r are two parameters called due date range and tardiness factors. Three levels of due date range (R) were tested: $R = 0.2$, 0.6 and 1.0 and five levels of due date tightness (r) were tested: $r = 0.0$, 0.2, 0.4, 0.6 and 0.8. These levels of R and r result in 15 sets of due date parameters for each n and M combination. After some experimentation, a time limit was set to stop each heuristic procedure. The time limit was set to $(n * 0.2 + 0.5) * (n/20)^2 * (M/5)$ seconds.

The procedures were coded in Turbo Pascal and were tested on a Dell Inspiron 1525 GHz laptop computer. Each procedure was performed once for each problem. The measures of performance used to evaluate the procedures for the test for the small-sized problems ($n = 8$, 10 and 12) are the percentage deviation (% *Dev*) of the total earliness and tardiness of the solution generated by each procedure from the optimal total earliness and tardiness and the number of times each procedure generated an optimal solution. More specifically, the percentage deviation % *Dev* is calculated as $\%Dev = [(Z_h - Z_O)/\ Z_O] * 100$, where $Z_O =$ the total earliness and tardiness generated by the optimal procedure, and $Z_h =$ the total earliness and tardiness of the solutions generated by the heuristic procedures (NS, PRSA, ZSA, GA, FLVG, FACO). For the problems with more than 12 jobs, the measures of performance are the percentage deviation (% *Dev*) of the total earliness and tardiness of the solution generated by each procedure from the lowest total earliness and tardiness generated by the procedures and the number of times each procedure generated the best solution. $\%\ Dev = [(Z_h - Z_B)/\ Z_h] * 100$, where $Z_B =$ the lowest total earliness and tardiness of the solutions generated by the three procedures, and $Z_h =$ the total earliness and tardiness of the solutions generated by the heuristic procedures (NS, PRSA, ZSA, GA, FLVG, FACO).

3.2 *Results of the test*

Table 1 shows the % *Dev* for each procedure and the number of times each procedure generated an optimal solution for each level of number of jobs to be sequenced (n) and number of machines (M) for the small-sized problems.

Table 1. Percent deviation from optimal solution and number of times optimal (in parentheses).

Problem size		Procedure					
n	*M*	NS	PRSA	ZSA	GA	FLVG	FACO
8	5	1.45 (108)	0.93 (117)	1.06 (106)	0.01 (149)	0.10 (141)	0.02 (145)
8	10	1.10 (110)	0.65 (121)	0.55 (115)	0.00 (150)	0.03 (147)	0.00 (147)
8	20	0.82 (105)	0.60 (126)	0.67 (101)	0.00 (149)	0.02 (148)	0.00 (150)
10	5	1.32 (88)	1.25 (89)	2.07 (58)	0.09 (138)	0.21 (128)	0.16 (128)
10	10	2.30 (68)	0.97 (102)	1.70 (63)	0.08 (144)	0.33 (126)	0.11 (131)
10	20	0.58 (81)	0.57 (107)	1.37 (57)	0.06 (141)	0.08 (135)	0.06 (137)
12	5	2.31 (55)	2.11 (72)	4.45 (27)	0.27 (115)	0.57 (99)	0.70 (80)
12	10	2.72 (47)	1.64 (73)	3.74 (24)	0.40 (108)	0.55 (93)	0.53 (86)
12	20	1.83 (44)	1.21 (86)	2.86 (27)	0.10 (133)	0.35 (100)	0.28 (96)

Table 2. Percent deviation from best solution and number of times best (in parentheses) for $M = 5$.

	Procedure					
n	NS	PRSA	ZSA	GA	FLVG	FACO
15	2.89 (45)	3.31 (43)	6.81 (8)	0.55 (108)	1.64 (55)	1.47 (39)
20	3.95 (28)	3.25 (37)	9.17 (1)	0.65 (86)	3.25 (19)	3.02 (10)
25	2.65 (31)	3.29 (29)	9.38 (1)	0.56 (88)	4.27 (15)	3.57 (9)
30	3.11 (34)	2.97 (27)	10.40 (0)	0.83 (78)	4.39 (18)	4.21 (4)
40	2.51 (29)	2.63 (36)	11.18 (0)	0.91 (79)	7.00 (5)	4.56 (1)
50	3.68 (30)	2.83 (40)	11.52 (0)	0.86 (73)	6.66 (7)	5.21 (1)
75	5.07 (9)	2.25 (41)	11.12 (0)	0.79 (90)	5.93 (6)	4.63 (4)
100	6.66 (3)	1.63 (61)	11.63 (0)	1.05 (70)	4.74 (12)	4.76 (4)

The results show that the GA, FLVG and FACO procedures consistently generate optimal or close to optimal solutions. Each of these procedures generated solutions that on average were within 1% of the total earliness and tardiness of an optimal solution for each problem size and generated an optimal solution for over 50% of the problems for each problem size. The GA procedure had the lowest % *Dev* for each problem size and generated the most optimal solutions for eight of the nine problem sizes. The NS and PRSA procedures' results were not as good with average % *Devs* over 1% for several problem sizes and also generated an optimal solution for less than 50% of the problems on several problem sizes. The ZSA procedure's results were poor with average % *Devs* over 10% for six of the nine problem sets and generating optimal solutions for less than 10% of the problems for eight of the nine problem sets.

Tables 2, 3 and 4 show the % *Dev* for each procedure and the number of times each procedure generated the best solution for each level of number of jobs to be sequenced (n). Table 2 shows the results for $M = 5$, Table 3, for $M = 10$ and Table 4, for $M = 20$.

The results show that the GA procedure performs the best on each problem size. This procedure has the lowest average % *Dev* for each problem size and generated the best solution more times than any of the other procedures for each problem size. Also, the average % *Dev* for the GA procedure is less than 1% for 20 of the 24 problem sizes. Among the other procedures, only the FLVG procedure has an average % *Dev* less than 1% on any of the problem sizes. Generally, the PRSA procedure was second best and had average % *Devs* less than 3.5% for all problem sizes and less than 2% for four problem sizes. The results for the FLVG and FACO procedures tended to deteriorate as the number of jobs increased. The results for the ZSA procedure were very poor with average % *Devs* over 20% for all problem sizes and did not generate a solution that was best for any problem.

These results indicate that the genetic algorithm is the best procedure for the problem. The results also show that incorporating specialised knowledge for the problem into a simulated annealing algorithm will not necessarily result in better solutions.

Table 3. Percent deviation from best solution and number of times best (in parentheses) for $M = 10$.

	Procedure					
n	NS	PRSA	ZSA	GA	FLVG	FACO
15	4.09 (28)	1.82 (66)	6.03 (9)	0.39 (103)	1.40 (62)	1.68 (31)
20	4.01 (17)	2.77 (40)	8.15 (2)	0.55 (95)	2.98 (24)	2.76 (14)
25	3.54 (20)	2.92 (33)	9.27 (2)	0.52 (94)	3.71 (22)	4.44 (5)
30	3.90 (10)	3.08 (33)	10.33 (0)	0.52 (101)	4.77 (9)	4.83 (6)
40	4.02 (20)	2.82 (32)	12.26 (0)	0.87 (88)	5.74 (9)	6.53 (1)
50	4.07 (18)	2.69 (34)	12.07 (0)	0.75 (89)	6.59 (8)	5.70 (2)
75	7.54 (5)	2.35 (45)	13.08 (0)	0.82 (91)	5.63 (6)	6.05 (3)
100	7.94 (4)	1.99 (48)	13.88 (0)	1.19 (85)	5.28 (9)	5.80 (4)

Table 4. Percent deviation from best solution and number of times best (in parentheses) for $M = 20$.

	Procedure					
n	NS	PRSA	ZSA	GA	FLVG	FACO
15	2.40 (30)	1.65 (46)	4.34 (9)	0.22 (112)	0.84 (60)	1.02 (30)
20	2.98 (20)	2.16 (35)	6.45 (0)	0.40 (98)	1.97 (25)	2.50 (10)
25	3.00 (13)	2.76 (26)	7.61 (0)	0.39 (106)	3.09 (17)	3.70 (1)
30	3.83 (14)	3.09 (28)	9.71 (1)	0.53 (100)	3.98 (9)	4.60 (0)
40	4.07 (13)	3.27 (34)	9.99 (1)	0.71 (95)	4.30 (8)	5.70 (0)
50	3.92 (14)	3.21 (22)	11.82 (0)	0.59 (103)	6.12 (9)	6.66 (2)
75	4.97 (9)	2.24 (40)	11.82 (0)	1.04 (89)	5.59 (11)	6.21 (1)
100	7.02 (8)	2.26 (43)	12.41 (0)	1.03 (87)	4.67 (11)	5.63 (1)

Table 5. Percent deviation from best solution by r for $n = 50$ and $M = 10$.

	Procedure					
R	NS	PRSA	ZSA	GA	FLVG	FACO
0.0	4.34	1.88	14.29	1.72	5.05	3.99
0.2	6.12	3.87	16.33	1.15	11.66	8.37
0.4	6.84	3.14	19.42	0.33	8.31	10.04
0.6	1.74	2.30	6.93	0.39	5.41	3.77
0.8	1.33	2.29	3.41	0.69	2.53	2.35

Table 6. Percent deviation from best solution by R for $n = 50$ and $M = 10$.

	Procedure					
R	NS	PRSA	ZSA	GA	FLVG	FACO
0.2	2.78	2.00	9.06	0.68	5.15	4.34
0.6	2.84	2.89	8.45	0.41	6.51	4.56
1.0	6.60	3.19	18.71	1.16	8.12	8.22

In order to show the effect of the due date range (R) and tardiness factor (r) on the results, Tables 5 and 6 are presented. Table 5 shows the % *Dev* by due date tardiness factor (r) for $n = 50$ and $M = 10$.

The results are consistent with previous results. The GA procedure's % *Dev* averaged less than 2% for each of the due date tightness parameters and was the lowest among the procedures for each of the values of r. Each of the other procedures' % *Dev* averaged greater than 3% for at least two values of r and all the other procedures, with the exception of the PRSA procedure, had a % *Dev* that averaged greater than or equal to 5% for at least one of the due date tightness parameters.

Table 6 shows the % *Dev* by due date range factor (R) for $n = 50$ and $M = 10$.

The results are again consistent with previous results. The GA procedure had the lowest average % *Dev* for each value of R and averaged less than 1.25% for each of the values. None of the other procedures had a % *Dev* that was

less than 2% for any value of R. It also appears that there is greater variability in the quality of the solutions generated by the procedures when the range is high. % *Devs* generally increase for the procedures as R increases and was greatest when $R = 1.0$.

4. Conclusion

In this paper a genetic algorithm was proposed for minimising total earliness and tardiness in permutation flow shops without allowing unforced inserted idle time. This procedure and five other heuristic procedures were tested on problems of various sizes in terms of the numbers of jobs and machines and 15 sets of distributions that determine the tightness and range of due dates.

The results showed that the proposed genetic algorithm consistently generates solutions with a lower total earliness and tardiness than the other procedures tested.

There are two possible avenues for additional research. One area of additional research would be lifting the restriction of not allowing idle time to be inserted in the schedule. Since the objective is non-regular, inserting idle time could reduce the objective so ways to insert idle time into a schedule to minimise the objective are needed. The second area of additional research would be incorporating setup time considerations into the problem. The case where setup times are sequence-dependent could be considered.

References

Baker, K.R. and Scudder, G.D., 1990. Sequencing with earliness and tardiness penalties: a review. *Operations Research*, 38 (1), 22–36.

Chandra, C., Mehta, P., and Tirupati, D., 2009. Permutation flow shop scheduling with earliness and tardiness penalties. *International Journal of Production Research*, 47 (20), 5591–5610.

Framinan, J.M. and Leisten, R., 2008. Total tardiness minimization in permutation flow shops: a simple approach based on a variable greedy algorithm. *International Journal of Production Research*, 46 (22), 6479–6498.

Hasija, S. and Rajendran, C., 2004. Scheduling in flowshops to minimize total tardiness of jobs. *International Journal of Production Research*, 42 (11), 2289–2301.

Holthaus, O. and Rajendran, C., 2005. A fast ant-colony algorithm for single-machine scheduling to minimize the sum of weighted tardiness of jobs. *Journal of the Operational Research Society*, 56 (8), 947–953.

Hoogeveen, H., 2005. Multicriteria scheduling. *European Journal of Operational Research*, 167 (3), 592–623.

Kanet, J.J. and Sridharan, V., 2000. Scheduling with inserted idle time: problem taxonomy and literature review. *Operations Research*, 48 (1), 99–110.

Kim, Y., 1993. Heuristics for flowshop scheduling problems minimizing mean tardiness. *Journal of Operational Research Society*, 44 (1), 19–28.

Kim, Y., Lim, H.G., and Park, M.W., 1996. Search heuristics for a flowshop scheduling problem in a printed circuit board assembly process. *European Journal of Operational Research*, 91 (1), 124–143.

Korman, K., 1994. A pressing matter. *Video*, 17 (2), February, 46–50.

Landis, K., 1993. Group technology and cellular manufacturing in the Westvaco Los Angeles VH Department, Project report in IOM 581, School of Business, University of Southern California.

Madhushini, N., Rajendran, C., and Deepa, Y., 2009. Branch-and-bound algorithms for scheduling in permutation flowshops to minimize the sum of weighted flowtime/sum of weighted tardiness/sum of weighted flowtime and weighted tardiness/sum of weighted flowtime, weighted tardiness and weighted earliness of jobs. *Journal of the Operational Research Society*, 60 (7), 991–1004.

Moslehi, G., *et al*., 2009. Two-machine flow shop scheduling to minimize the sum of maximum earliness and tardiness. *International Journal of Production Economics*, 122 (2), 763–773.

Parthasarathy, S. and Rajendran, C., 1997. A simulated annealing heuristic for scheduling to minimize mean weighted tardiness in a flowshop with sequence-dependent setup times of jobs – A case study. *Production Planning and Control*, 8 (5), 475–483.

Rajendran, C., 1999. Formulations and heuristics for scheduling in a kanban flowshop to minimize the sum of weighted flowtime, weighted tardiness and weighted earliness of containers. *International Journal of Production Research*, 37 (5), 1137–1158.

Singh, A., 2010. A hybrid permutation-coded evolutionary algorithm for the early/tardy scheduling problem. *Asia-Pacific Journal of Operational Research*, 27 (6), 713–725.

Taillard, E., 1993. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64 (2), 278–285.

Valente, J.M.S., 2009. Beam search heuristics for the single machine scheduling problem with linear earliness and quadratic tardiness costs. *Asia-Pacific Journal of Operational Research*, 26 (3), 319–339.

Vallada, E. and Ruiz, R., 2010. Genetic algorithms with path relinking for the minimium tardiness permutation flowshop problem. *Omega*, 38 (1–2), 57–67.

Vallada, E., Ruiz, R., and Minella, G., 2008. Minimising total tardiness in the m-machine flowshop problem: A review and evaluation of heuristics and metaheuristics. *Computers & Operations Research*, 4 (4), 1350–1373.

Zegordi, S.H., Itoh, K., and Enkawa, T., 1995. A knowledgeable simulated annealing scheme for the early/tardy flow shop scheduling problem. *International Journal of Production Research*, 33 (5), 1449–1466.