A new mixed integer linear programming model for flexible job shop scheduling problem

Mohsen Ziaee*

Department of Industrial Engineering, University of Bojnord, 94531-55111 Bojnord, Iran

Abstract. In this paper, a mixed integer linear programming (MILP) model is presented to solve the flexible job shop scheduling problem (FJSP). This problem is one of the hardest combinatorial problems. The objective considered is the minimization of the makespan. To the best of our knowledge, the MILP model presented by Ozguven et al. [Appl. Math. Model. 34 (2010) 1539] is the best model in the literature and obtains best results for the problem. The computational results show that the mathematical model proposed in this paper is superior to the one presented by Ozguven.

Keywords. Scheduling, Flexible Job Shop, Makespan, Mixed integer linear programming.

1. Introduction

The job shop scheduling problem (JSP) is an important scheduling problem in the literature (Jain and Meeran, 1998) and has attracted many scheduling researchers due to its applicability and difficulty (Blazewicz et al., 1996; Jain and Meeran, 1999). The $n \times m$ classical JSP is defined as follows: set of n jobs must be performed on m machines, and each job i consists of J_i operations processed on the machines. Each job has a specified processing order on the machines that is fixed and known in advance, i.e., each operation has to be performed on a given machine without interruption. The machines are continuously available throughout the planning horizon and cannot process more than one operation at a time. The processing times for all the operations are fixed and known (Baker, 1974; Pinedo, 2002). A common objective function for this problem is the minimization of the makespan that is the time needed to complete all the jobs.

In the flexible job shop scheduling problem (FJSP) that is an extension of the classical JSP, each operation can be executed by any machine among a set of candidate machines; therefore, this problem has two subproblems: the assignment of the operations to machines and the determination of the sequence of operations on each machine. The FJSP is strongly NP-hard even if there are only two machines and each job has at most three operations (Garey et al., 1976).

In the literature, many methods have been developed to solve more complex problems in the field of scheduling such as FJSP and JSP (Chiang and Lin, 2013; Yuan and Xu, 2013; Li et al., 2013; Li and Pan, 2012; Yuan et al., 2013), but most of them are metaheuristic algorithms. Development of MILP models for these problems has become increasingly important in recent years, because they can be easily solved using a robust software system and used for benchmarking the heuristic approaches

* Telefax: +98 583 2284626.
E-mail address: ziaee@iust.ac.ir, ziaee2@gmail.com.

and understanding the structure of the problems (Unlu and Mason, 2010). A review of MILP formulations for the flow shop and the job shop scheduling problems presented in the literature is provided by Pan (1997). For the FJSP, a review of the mathematical models developed for this problem can be found in Özgüven et al. (2010). They also presented a MILP model for the FJSP and compared it with a model of Fattahi et al. (2007) concluding that their model is superior to that of Fattahi et al. (2007) in terms of model size, CPU time and objective function value. A more recent literature review and comparative evaluation of mathematical models for the FJSP is provided by Demir and Isleyen (2013). They investigated all the mathematical models for the FJSP existing in the literature in terms of binary variables used for sequencing the operations on the machines, and computationally compared the models assuming that the objective function is makespan. They conclude that the model presented by Özgüven et al. (2010) is the best model in the literature in terms of the computational time.

In this study, a new mixed integer linear programming (MILP) model is presented to solve the FJSP with the objective of minimizing makespan (section 2). The performance of the proposed MILP model is evaluated by using several benchmark problems and the results of computational studies are presented (section 3). Concluding remarks are given in section 4.

Other assumptions considered in the problem studied in this paper are as follows:

(1) Jobs are independent of each other, all of them have equal priorities and are available at time zero.

(2) Machines are independent of each other.

(3) Setup times and transportation times are assumed to be negligible and zero.

(4) An operation cannot be processed by more than one processor (machine) at the same time.

2. MILP formulation

The problem notations are as follows:
n: number of jobs,
m: number of machines,
i,i': index of jobs; $i,i'=1,\ldots,n$,
J_i: number of operations of job i,
j,j': index of operations; $j=1,\ldots, J_i$, $j'=1,\ldots, J_{i'}$,
k,k': index of machines; $k,k'=1,\ldots,m$,
t_{ijk}: the processing time required on machine k for operation j of job i,
c_{ijk}: completion time of operation j of job i (denoted by (i, j)) on machine k,
A_{ij}: set of machines that are capable to process operation j of job i, and the operation must be performed on only one of the alternative machines in this set,
$y_{iji'j'}$: binary variable which defines the processing order of the operations (i, j) and (i', j') belonging to two different jobs ($i<i'$) on the same machine; it takes the value 1 if operation (i', j') precedes operation (i, j), and 0 otherwise,
x_{ijk}: binary variable that takes the value 1 if operation (i, j) is executed on machine k, and 0 otherwise,
$Cmax$: makespan,
M: a large number.

Here, a MILP formulation is presented for the problem.

Minimize $Cmax$ (1)

subject to:

$$\left(\sum_{k'\in A_{ij}} c_{ijk'}\right)-\left(c_{i(j+1)k}-t_{i(j+1)k}\right)\le M.(1-x_{i(j+1)k}) \quad \forall i, \forall j<J_i,\ k\in A_{i(j+1)}, \tag{2}$$

$$c_{ijk}-t_{ijk}-c_{i'j'k}\ge -M.(1-y_{iji'j'})-M.(2-x_{ijk}-x_{i'j'k}) \quad \forall i,\ \forall j\le J_i,\ k\in(A_{ij}\cap A_{i'j'}),\ \forall i'>i,\ \forall j'\le J_{i'}, \tag{3}$$

$$c_{i'j'k}-t_{i'j'k}-c_{ijk}\ge -M.y_{iji'j'}-M.(2-x_{ijk}-x_{i'j'k}) \quad \forall i,\ \forall j\le J_i,\ k\in(A_{ij}\cap A_{i'j'}),\ \forall i'>i,\ \forall j'\le J_{i'}, \tag{4}$$

$$\left(\sum_{k\in A_{ij}} x_{ijk}\right)=1 \quad \forall i,\ \forall j\le J_i, \tag{5}$$

$$M.(1-x_{i1k})+c_{i1k}-t_{i1k}\ge 0 \quad \forall i, k\in A_{i1}, \tag{6}$$

$$c_{ijk}-M.x_{ijk}\le 0 \quad \forall i,\forall j\le J_i,\ k\in A_{ij}, \tag{7}$$

$$Cmax\ge \sum_{k\in A_{iJ_i}} c_{iJ_ik} \quad \forall i, \tag{8}$$

$$c_{ijk}\ge 0 \quad \forall i,\forall j\le J_i,\ k\in A_{ij}, \tag{9}$$

$$x_{ijk}\in\{0,1\} \quad \forall i,\forall j\le J_i,\ k\in A_{ij}, \tag{10}$$

$$y_{iji'j'}\in\{0,1\} \quad \forall i,\forall j\le J_i,\ \forall i'>i,\ \forall j'\le J_{i'}, \tag{11}$$

The objective (1) is the minimization of the makespan. Constraint set (2) ensures precedence restrictions between consecutive operations of each job, it means that the completion time of jth operation of job i should not be greater than the start time of its ($j+1$)th operation. Constraint sets (3) and (4) define the order of any two operations performed on the same machine and guarantee that they will not clash. For each two operations (i, j) and (i', j'), ($i\neq i'$), constraint (3) is active only if the two operations can be executed on the same machine (k) (i.e. $x_{ijk}=x_{i'j'k'}=1$) and operation (i, j) is performed after operation (i', j') (i.e. $y_{iji'j'}=1$); otherwise, i.e. if $x_{ijk}=0$ or $x_{i'j'k'}=0$ or $y_{iji'j'}=0$, this constraint is inactive. Note that, operation (i, j) is not necessarily positioned immediately after operation (i', j') when $y_{iji'j'}=1$. Clearly, for any two operations processed on the same machine, either constraint (3) or constraint (4) is active. Constraint set (5) means that each operation is assigned to exactly one machine. If operation (i, j) is not assigned to machine k, constraint (7) sets the completion time of it on machine k equal to zero. Constraint set (6) ensures that the start time of each job (i.e. its first operation) should be greater than or equal to the time zero. Constraint set (8) determines the makespan, i.e. the completion time of last operation of each job should not be greater than the makespan. Constraint set (9) is a non-negativity constraint for variables c_{ijk}. Constraint sets (10) and (11) define the binary variables x_{ijk} and $y_{iji'j'}$, respectively.

3. Computational experiments

In this section, the results of computational studies are presented. As mentioned in section 1, the MILP model presented by Özgüven et al. (2010) is the best model for the FJSP in the literature (Demir and Isleyen, 2013). Therefore, we compared the computational results of proposed MILP model (henceforth MILP-Zia) with those of Özgüven et al. (2010) (henceforth MILP-Literature). The benchmark instances used were the set of 20 problems taken from Fattahi et al. (2007). They are divided into two categories: small size FJSPs (denoted by SFJS1-SFJS10) and medium-large size FJSPs (denoted by MFJS1-MFJS10). Both MILP-Zia and MILP-Literature were solved for each of the benchmark instances using the software LINGO Release 8.0 (LINDO Systems Inc., 1999) that uses the algorithm of Branch and Bound to optimally solve the problem. We used the default settings of the solver. The problems were run on a Pentium IV, 2.2 GHz and 2.0 GB RAM PC. The running time for each benchmark problem was limited to 3600 CPU seconds. The computational results are presented in Table 1. The first column (*name*) indicates the name of each test problem. The second column (*size*) refers to the size of each instance denoted by *i, j, k* indices, that means number of jobs, operations and machines, respectively. Note that, in any of these test problems, all the jobs have the same number of operations. The first set of columns contains the results of MILP-Literature and the second set contains those from MILP-Zia. *Integers, Non-integers, Constraints, CPU time (s),* and *Cmax* represent the number of integer variables, the number of non-integer variables, the number of constraints, the running time in seconds, and the makespan value, for each instance, respectively. RPD is the relative percentage deviation and is computed by the following relation:

$$RPD = \frac{C\,max_1 - C\,max^*}{C\,max^*} \times 100\,,$$

where $Cmax_1$ is the makespan value obtained by the corresponding MILP model and $Cmax^*$ is the best value of makespan achieved by the two MILP models (i.e. MILP-Literature and MILP-Zia). As shown in the table, average RPD value for MILP-Zia is the best possible value, i.e. 0, compared to 6.73 for MILP-Literature. The results also show that the difference between the solutions obtained increases as the instances size increases; the instance MFJS10 has RPD value of 45.8, for example. Fig. 1-3 graphically represent the difference between RPD values, Cmax values, and total number of variables of the two MILP models, respectively. Differences between the number of constraints of the two models are shown in Fig. 4. Herein, MILP-Zia is statistically compared with MILP-Literature. A one-way analysis of variance (ANOVA) (Montgomery, 2000) is done to test the null hypothesis that the means of RPD values of the two MILP models are equal. The results for this ANOVA are given in Table 2. As it can be seen in the table, the difference between the means of two models is meaningful at a significance level of 5%. We also compare the models via Tukey's pair-wise comparisons test (Montgomery, 2000). The results reported in Fig. 5 show that there is a significant difference between the two models.

The approach presented in this paper is based on the method of Manne (1960) to handle binary variables for formulating FJSP, that uses precedence variables, instead of sequence-position variables (introduced by Wagner (1959)) or time indexed variables (proposed by Bowman (1959)) (See (Demir and Isleyen, 2013)). In comparison with MILP-Literature, new precedence variable, $y_{iji'j'}$, replaces $y_{iji'j'k}$ in MILP-Literature, where $y_{iji'j'}$ is equal to 1 if operation (*i*, *j*) precedes operation (*i'*, *j'*), and 0 otherwise. The definition of $y_{iji'j'}$ is without reference to the operation's machine. This substituting leads to constraint sets (3) and (4) and significantly reduces the number of binary variables and the number of lines of code and thus, greatly enhances the efficiency of the model; because, the running time increases and computational feasibility decreases as the number of variables or the number of constraints increase (French, 1982; Liao and You, 1992; Wilson, 1989); the number of variables is however more important, since a formulation with a fewer number of constraints more often requires a longer computational time for finding proven optimal solution (Nemhauser and Wolsey, 1988). The difference between the number of variables of the two models is much more than that of constraints of them as seen in the computational results, implying that the new model is more efficient than the previous one. Moreover, the new model does not use the variables S_{ijk} and C_i in MILP-Literature. More improvement in the model performance can be obtained by using dominance relationships (Zhu and Heady, 2000) to force certain of the $y_{iji'j'}$ values to 0 or 1. For instance, if J_i is sufficiently large for all the jobs, then it can be shown that $y_{iji'j'}$ must be set to 0 if $j \leq \alpha$ and $j' \geq \beta$ for some values of α and β (for example, $\alpha=0.1J_i$ and $\beta=0.9J_{i'}$), that leads to decreasing the number of binary variables.

Table 1
The computational results of MILP-Literature and MILP-Zia

Name	Size	MILP-Literature						MILP-Zia					
	(i.j.k)	Integers	Non-integers	Constraints	CPU time (s)	Cmax	RPD	Integers	Non-integers	Constraints	CPU time (s)	Cmax	RPD
SFJS1	2.2.2	16	19	42	0	66	0	12	9	38	0	66	0
SFJS2	2.2.2	10	15	30	0	107	0	9	7	26	0	107	0
SFJS3	3.2.2	26	24	67	0	221	0	21	11	61	0	221	0
SFJS4	3.2.2	26	24	67	0	355	0	22	11	61	0	355	0
SFJS5	3.2.2	36	28	87	1	119	0	24	13	81	1	119	0
SFJS6	3.3.2	39	34	99	0	320	0	34	16	90	0	320	0
SFJS7	3.3.5	36	40	93	0	397	0	34	19	84	0	397	0
SFJS8	3.3.4	45	40	111	7	253	0	40	19	102	1	253	0
SFJS9	3.3.3	55	40	131	1	210	0	45	19	122	1	210	0
SFJS10	4.3.5	48	45	124	0	516	0	46	21	112	1	516	0
MFJS1	5.3.6	103	72	241	3600	468	0	84	34	226	60	468	0
MFJS2	5.3.7	128	84	291	3600	461	2.90	100	40	276	3600	448	0
MFJS3	6.3.7	190	103	422	3600	505	7.91	143	49	404	3600	468	0
MFJS4	7.3.7	250	120	549	3600	611	2.00	184	57	528	3600	599	0
MFJS5	7.3.7	243	118	535	3600	680	21.86	188	56	514	3600	558	0
MFJS6	8.3.7	307	133	670	3600	715	3.32	234	63	646	3600	692	0
MFJS7	8.4.7	475	165	1022	3600	1406	16.78	364	79	990	3600	1204	0
MFJS8	9.4.8	519	182	1119	3600	1303	16.97	415	87	1083	3600	1114	0
MFJS9	11.4.8	751	218	1601	3600	1891	17.02	574	104	1557	3600	1616	0
MFJS10	12.4.8	899	237	1906	3600	2580	45.85	651	113	1858	3600	1769	0
Average							6.73						0

Table 2. Results of one-way ANOVA for the two models (MILP-Literature and MILP-Zia)

Source	DF	SS	MS	F	P
Factor	1	453	453	6.5	0.014
Error	38	2618.8	68.9		
Total	39	3071.7			

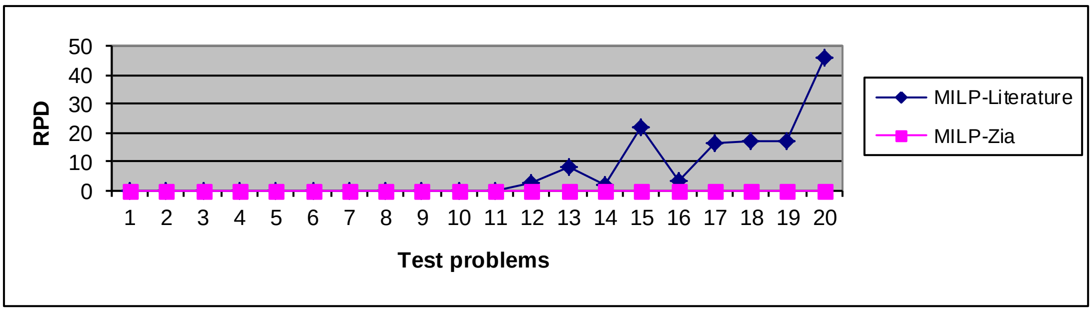

Fig. 1
Comparison of the RPD values of the benchmark instances

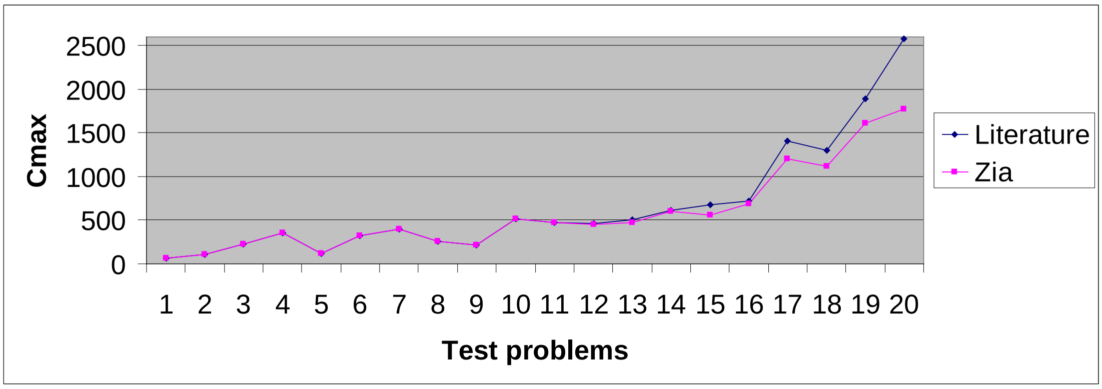

Fig. 2
Comparison of the C_{max} values of the benchmark instances

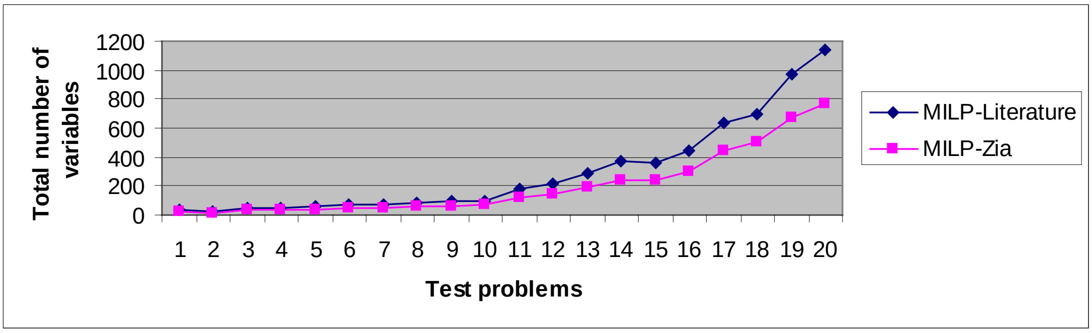

Fig. 3
Comparison of the total number of variables of the benchmark instances

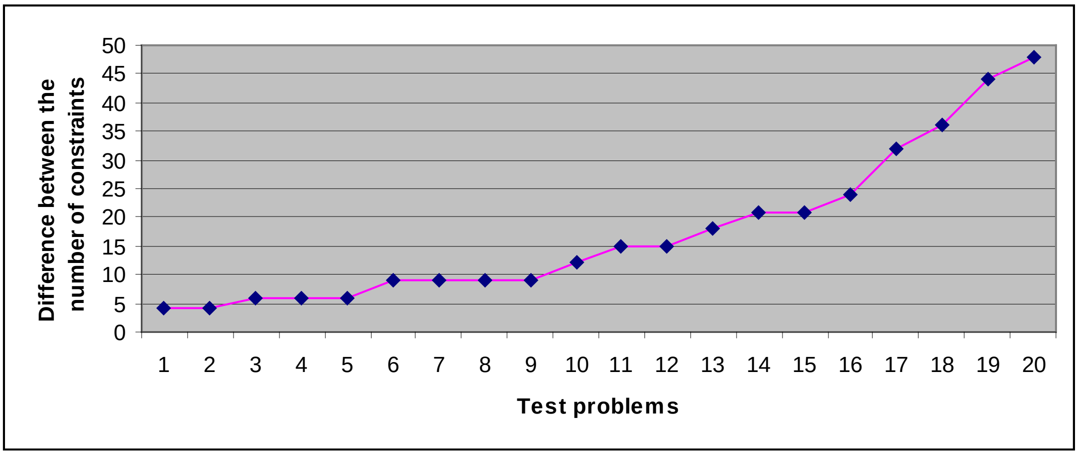

Fig. 4
Differences between the number of constraints of the two models

```
Tukey 95% Simultaneous Confidence Intervals
All Pairwise Comparisons

Individual confidence level = 95.00%


C1 subtracted from:

      Lower  Center   Upper  ----+---------+---------+---------+-----
C2  -12.045  -6.730  -1.416  (----------*---------)
                             ----+---------+---------+---------+-----
                               -10.0      -5.0       0.0       5.0
```

Fig. 5
Results of Tukey's pair-wise comparisons test for the two models

4. Conclusion

This paper studies the FJSP which is one of the hardest combinatorial problems. The objective is the minimization of the makespan. A MILP model was presented for solving the problem. The computational results of the proposed MILP model were compared with those of the MILP model of Özgüven et al. (2010) (called here MILP-Literature) that is the best model in the literature in terms of the computational time (Demir and Isleyen, 2013). The results showed that our model is superior to MILP-Literature with respect to all the considered performance measures including RPD value, number of constraints, and total number of variables. This improvement in the quality of the results of MILP-Literature, can be useful for optimally solving larger FJS problems in reasonable time, and thus, the proposed MILP model is more beneficial for the performance evaluation of the heuristics developed for the problem. In future work, we will try to adjust the formulation to solve some more complex scheduling problems.

References

[1] A.S. Jain, S. Meeran, Deterministic job-shop scheduling: Past, present and future, European Journal of Operational Research 113 (2) (1998) 390–434.

[2] A.S. Jain, S. Meeran, A state-of-the-art review of job-shop scheduling techniques, European Journal of Operations Research 113 (1999) 390–434.

[3] J. Blazewicz, W. Domschke, E. Pesch, The job shop scheduling problem: Conventional and new solution techniques, European Journal of Operational Research 93 (1996) 1–33.

[4] K. Baker, Introduction to sequencing and scheduling, NewYork: Wiley (1974).

[5] M. Pinedo, Scheduling: theory, algorithms and systems, Englewood cliffs, NJ: Prentice-Hall (2002).

[6] M.R. Garey, D.S. Johnson, R. Sethi, The complexity of flow shop and job-shop scheduling, Mathematics of Operations Research 1 (2) (1976) 117–129.

[7] T.-C. Chiang, H.-J. Lin, A simple and effective evolutionary algorithm for multiobjective flexible job shop scheduling, International Journal of Production Economics 141 (2013) 87–98.

[8] Y. Yuan, H. Xu, Flexible job shop scheduling using hybrid differential evolution algorithms, Computers & Industrial Engineering 65 (2013) 246–260.

[9] J.-Q. Li, Q.-K. Pan, M. F. Tasgetiren, A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance

activities, Applied Mathematical Modelling 38 (2013) 1111–1132.

[10] J.-q. Li, Q.-k. Pan, Chemical-reaction optimization for flexible job-shop scheduling problems with maintenance activity, Applied Soft Computing 12 (2012) 2896–2912.

[11] Y. Yuan, H. Xu, J. Yang, A hybrid harmony search algorithm for the flexible job shop scheduling problem, Applied Soft Computing 13 (2013) 3259–3272.

[12] Y. Unlu, S.J. Mason, Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems, Computers and Industrial Engineering 58 (2010) 785–800.

[13] C.H. Pan, A study of integer programming formulations for scheduling problems, International Journal of Systems Science 28 (1) (1997) 33–41.

[14] C. Özgüven, L. Ozbakır, Y. Yavuz, Mathematical models for job-shop scheduling problems with routing and process plan flexibility, Applied Mathematical Modelling 34 (2010) 1539–1548.

[15] P. Fattahi, M.S. Mehrabad, F. Jolai, Mathematical modeling and heuristic approaches to flexible job shop scheduling problems, Journal of Intelligent Manufacturing 18 (2007) 331–342.

[16] Y. Demir, K. Isleyen, Evaluation of mathematical models for flexible job-shop scheduling problems, Applied mathematical modeling 37 (2013) 977–988.

[17] LINDO Systems Inc., LINGO User’s Guide, LINDO Systems Inc.: Chicago (1999).

[18] D.C. Montgomery, Design and analysis of experiments, Fifth ed., NewYork: John Wiley & Sons (2000).

[19] A.S. Manne, On the job-shop scheduling problem, Operations Research 8 (1960) 219–223.

[20] H.M. Wagner, An integer linear programming model for machine scheduling, Naval Research Logistics Quarterly 6 (1959) 131–140.

[21] E.H. Bowman, The scheduling sequence problem, Operations Research 7 (1959) 621–624.

[22] S. French, Sequencing and scheduling: an introduction to the mathematics of the job-shop, Chichester, UK: Ellis Horwood (1982).

[23] J.M. Wilson, Alternative formulations of a flow-shop scheduling problem, OR Journal (Journal of the Operational Research Society) 40(4) (1989) 395–9.

[24] C.-J. Liao, C.-T. You, Improved formulation for the job-shop scheduling problem, Journal of the Operational Research Society 43(11) (1992) 1047–54.

[25] G.L. Nemhauser, L.A. Wolsey, Integer and Combinatorial Optimization, John Wiley, New York (1988).

[26] Z. Zhu, R.B. Heady, Minimizing the sum of earliness/tardiness in multi-machine scheduling: a mixed integer programming approach, Computers & Industrial Engineering 38 (2000) 297–305.