7e Conférence Francophone de MOdélisation et SIMulation - MOSIM'08 - du 31 mars au 2 avril 2008 - Paris - France
«Modélisation, Optimisation et Simulation des Systèmes : Communication, Coopération et Coordination»

BRANCH AND BOUND ALGORITHM FOR TOTAL WEIGHTED TARDINESS MINIMZATION ON PARALLEL MACHINES

N. SOUAYAH, I. KACEM

Ecole Polytechnique de Tunisie
B.P. 743, 2078 La Marsa Tunisie
s_nizou@yahoo.fr, kacem@utt.fr

M. HAOUARI, C. CHU

Université de Technologie de Troyes
12 rue Marie Curie, Troyes, France
mohamed.haouari@ept.rnu.tn, chu@utt.fr

ABSTRACT : *This paper considers the problem of scheduling n independent jobs on m identical parallel machines to minimize the total weighted tardiness. We develop several lower bounds and an upper bound obtained from a classical genetic algorithm. These bounding procedures are used in a conventional branch and bound (BAB) algorithm. Computational experiments are performed on randomly generated test problems and results demonstrate the effectiveness of our approach.*

KEYWORDS : *Parallel machines, weighted tardiness, lower bound, Lagrangian relaxation.*

1. INTRODUCTION

This paper deals with the identical parallel machine scheduling problem to minimize total weighted tardiness. The weighted tardiness minimization problem on parallel machines has many important applications in industry. The objective of this paper is to develop an exact method to solve this problem.
The weighted tardiness of a job i is defined as $w_iT_i = w_imax(0, C_i - d_i)$, where C_i, d_i and w_i are respectively the completion time, due date and weight of job i. It is assumed that all jobs are available at time zero, each machine cannot process more than one job at any time and no job can be preempted. The problem treated here is at least NP-hard, since the similar problem with $m = 2$ and unweighted was proved NP-hard in the strong sense by Bruno and *al.*, 1974.
Elmaghraby and Park, 1974 proposed a branch-and-bound algorithm to minimize the total penalty of tardiness of jobs where the due date of each job is equal to its processing time. This result was improved by Barnes and Brennan, 1977. Potts and Van Wassenhoves, 1985 presented a branch-and-bound algorithm for the single machine total weighted tardiness problem. They obtained lower bounds using a Lagrangian relaxation approach and they proposed a fast procedure to solve the linear problem: the multiplier adjustment method.
Azizoglu and Kirka, 1998 developed a branch-and-bound algorithm, for the problem of tardiness minimization on parallel machines, that incorporates the dominance properties with an efficient lower bound which reduces the original problem to the problem of single machine total tardiness.
Yalaoui and Chu, 2002 proposed a branch-and-bound algorithm for the parallel machine total tardiness problem. Shim and Kim, 2006 proposed also a branch-and-bound algorithm for the same problem adressed, they obtained better results. They solved problems with up 30 jobs and 2 machines.
Since it is not easy to obtain optimal solutions for parallel machines weighted and unweighted tardiness problems of practical size, heuristic algorithms have been developed. Various dispatching rules have been proposed to compute priorities of the jobs, or modified rules for single machine tardiness problem.
In this paper, we present several lower bounds for the weighted tardiness problem. These bounding procedures are used in a conventional branch-and-bound approach, in order to minimize total weighted tardiness. Computational results are given for problems with up to 30 jobs and 3 machines.

The paper is organized as follows. Section 2 gives a formulation of the problem. Section 3 presents the lower bounds. Section 4 presents the branch and bound algorithm and Section 5 reports the computational results. Finally, some concluding remarks are given in section 6.

2. PROBLEM FORMULATION

We have to schedule a set J of n jobs on m parallel machines. Each job i has a processing time p_i, a due date d_i and a weight w_i. Each machine can process only one job at a given time and preemption is

not allowed. For a given sequence, a job i is called tardy if its completion time C_i is strictly larger than its due date and its weighted tardiness is measured by $w_iT_i = w_imax(0, C_i - d_i)$.
The objective is to find a processing order that minimizes $\sum_{i=1}^{n} w_iT_i$, so we can formulate our problem as follows,

$$\begin{array}{lll} \text{(P)} \quad \text{minimize} & \sum_{i=1}^{n} w_iT_i & \\ \text{subject to} & T_i \geq (C_i - d_i) & (1) \\ & T_i \geq 0. & \end{array}$$

where $(C_i)_{i\leq n}$ is a feasible completion time vector.

3. LOWER BOUNDS

In this section we present six lower bound procedures for the problem (P) noted $P||\sum w_iT_i$. Note that these procedures can be applied to compute lower-bounds on the weighted tardiness of a partial schedule.

3.1. Lower bound LB1

The principle of this bound consists in solving the lagrangian relaxation of the constraint (1) in problem (P) using *multiplier adjustment* method.
From Potts and Van Wassenhove, (1985) the lagrangian relaxation of the problem (P), yields the problem *(PLR)*

$$\begin{array}{lll} (PLR): L(\lambda) = \min & \sum_{i=1}^{n}(w_iT_i + \lambda_i(C_i - d_i - T_i)) & \\ & 0 \leq \lambda_i \leq w_i \quad \forall \quad i = 1, ..., n & \\ & T_i \geq 0 \quad \forall \quad i = 1, ..., n & \end{array}$$

where $\lambda = (\lambda_1, ..., \lambda_n)$ is the vector of corresponding nonnegative multipliers and $(C_i)_{i\leq n}$ is a feasible completion time vector. Then, we transform the weighted tardiness minimization problem to a weighted flow-time minimization problem with new weights equal to λ_i, $i = 1, ..., n$. Hence, we use a lower bound for the new problem given by Eastman (1964). Finally, we maximize this lower bound by choosing special values of λ_i. We use the *multiplier adjustment* method introduced by Potts and Van Wassenhove, (1985).

3.2. Lower bound LB2

This lower bound is based on the splitting relaxation. This relaxation reduces the problem to find an optimal solution for a single-machine weighted tardiness minimization problem, where the processing time of job i is $\frac{p_i}{m}$. As this problem is NP-Hard, we use a lower bound instead of searching an optimal solution. We denote by $p'_i = \frac{p_i}{m}$ the new processing time. By applying the same Lagrangian relaxation used in $LB1$, we obtain a new lower bound $LB2$.

3.3. Lower bound LB3

This lower bound is based on an assignment problem involving some lower bounds on the completion times of jobs. The lower bound is computed in two steps. In the first step, we compute an earliest completion time $C_{[i]}$ of job scheduled in position i (i.e., for every i, $C_{[i]}$ is a lower bound on the i^{th} smallest completion time in any feasible schedule). In the second step, we define an assignment problem between jobs and the above completion times $C_{[i]}$ that we solve by using the algorithm proposed by Baptiste and Le Pape, (2005) instead of the Hungarian method. The cost of assigning job j to the date $C_{[i]}$ is $w_j \max(0, C_{[i]} - d_j)$. In order to compute the completion times $C_{[i]}$, we use four lower bounds and we chose the best.

3.4. Lower bound LB4

The principle of this lower bound is a generalization of the lower bound introduced by Kacem, (2007) for the tardiness minimization on a single machine. Let $(t_i)_{1\leq i\leq n}$ denote the set of starting time of jobs. The lower bound *LB4* is given by the solution of the following linear problem

$$\begin{array}{lll} \min & \sum_{i=1}^{n} w_i \max(t_i + p_i - d_i, 0) & \\ \text{subject to} & \sum_{i=1}^{n} \alpha_i^k(t_i + p_i) \geq LB_{Eastman}(p, \alpha^k) & \\ & \forall \quad 1 \leq k \leq q, & \\ & \sum_{i=1}^{n}(t_i + p_i) \geq \sum_{i=1}^{n} C_i^{SPT} & \end{array}$$

where $LB_{Eastman}(p, \alpha^k)$ is Eastman's lower bound for the weighted flowtime problem when the processing times are given in vector p and the weights are given in vector α^k, C_i^{SPT} is the completion time of job i obtained by scheduling jobs indexed in SPT (Shortest Processing Time) order on the earliest available machine, q is the number of constraints and $\alpha^k = (\alpha_i^k)_{i\leq n}$ represents the fictitious weights associated to the jobs. We choose $(\alpha_i^k)_{1\leq k\leq q}$ in order to obtain the different permutations of jobs in the sequence. In other words, each constraint correspond to a position of jobs in the sequence SWPT. The lower bound is computed using the linear programming solver CPLEX.

3.5. Lower bound LB5

This lower bound has the same principle of the first lower bound *LB1*. We apply the same Lagrangian relaxation and in order to set the completion time C_i, we use a schedule σ yielded by a perturbation of the SWPT order. In order to find the best lower bound, we vary σ and we determine σ' such that $LB5(\sigma') \geq LB5(\sigma)$.

3.6. Lower bound LB6

We derive a lower bounding scheme based on a Lagrangian relaxation which we solve with the subgradient method. We determine for each job i a lower bound a_i and an upper bound b_i for the tardiness T_i. Therefore, our problem can be written

$$(P_1) \quad \min \quad \sum_{i=1}^{n} w_i T_i \tag{2}$$

$$C_i - d_i \leq T_i \quad \forall \quad i \tag{3}$$

$$a_i \leq T_i \leq b_i \quad \forall \quad i. \tag{4}$$

We obtain a lower bound by performing a Lagrangian relaxation on the constraints (6), which transform problem to a minimization of a weighted flowtime problem which we solve using the Eastman's lower bound. Then, to find the best possible lower bound, we apply the subgradient method. We use the variant proposed by Sherali and Ulular, 1990 known as the *Average Direction Search (ADS)* strategy.

4. BRANCH AND BOUND ALGORITHM

The branch and bound algorithm developed exploit the lower bounds proposed previously and an upper bound for the weighted tardiness problem given by a classical genetic algorithm. In order to minimize the computational effort, we use the best of the lower bounds. The branch-and-bound starts by computing an initial solution which provides the first upper bound given by the genetic algorithm. Every node represents a partial schedule. The branching consists in scheduling a new job after a partial schedule on the earliest machine available. Before the creation of a new node a lower bound is calculated. If the value of the lower bound obtained is larger than or equal to the value of the upper bound, then this node is removed.

5. RESULTS AND ANALYSIS

In this section, we describe the computational results. The branch and bound algorithm was coded in the *C* language and we use Cplex software to solve the linear programs of some lower bounds. Computational experiments were done on a personal computer with an Intel Pentium IV 1.8 GHz and 3G RAM, running under Windows XP.

The test examples were randomly generated. For the experiments, 10 problems for each of all combinations of two levels for the number of machines (2,3), five levels for the number of jobs (10,15,20,25,30), five levels for T and five level for R the two parameter for controlling tightness of due dates as presented in Table 1. We generate, for each job, an integer processing time p_i using a uniform distribution $[1, 100]$, an integer weight w_i using a uniform distribution $[1, 10]$ and an integer due date in the interval $[P \times (1 - T - (R/2)), P \times (1 - T + (R/2))]$, where $P = \sum_{i=1}^{n} \frac{p_i}{m}$.

R \ T	0.2	0.4	0.6	0.8	1
0.2	1	2	3	4	5
0.4	6	7	8	9	10
0.6	11	12	13	14	15
0.8	16	17	18	19	20
1	21	22	23	24	25

Table 1: Problem groups

To evaluate the performance of the lower bounds presented in this paper, we have computed the mean value obtained from 10 instances for each group with the different levels for the number of machines and jobs. From them, we can make the following remarks:

- The lower bounds LB1, LB5 and LB6 have relatively the same performance and behavior but in computational time, LB1 dominates largely the others (see Table 3). Indeed, the use of the multipliers adjustment method in LB1 is faster than the resolution using CPLEX in LB5 and the subgradient algorithm in LB6.
- Lower bound LB3: this procedure require a low computational time ($10^{-4}s$) but does not gives a good lower bound for our problem compared to the other bounds.
- Lower bound LB4: this lower bound is less effective in terms of performance and computational time. This is due to the set of the constraints given by the choice of the fictitious weights that must be judiciously chosen and also to CPLEX software that needs a large computation time.
- Lower bound LB2: this lower bound can be computed easily but it is dominated in all the instances. This observation proves that the relaxation used, that consists in transforming the parallel machines prob-

lem to a single-machine problem, is not efficient for the considered problem.
The suitable and/or potential lower bounds for each group are summarized in Table 2.

Groups	Not dominated lower bounds
1	LB1,LB3, LB5
2	LB1, LB3,LB5, LB6
3	LB1, LB5,LB6
4	LB1, LB5,LB6
5	LB1, , LB5,LB6
6	LB1,LB5
7	LB1, LB5
8	LB1, LB5,LB6
9	LB1,LB5,LB6
10	LB1, LB5, LB6
11	LB1, LB5
12	LB1, LB5, LB6
13	LB1, LB5, LB6
14	LB1, LB5, LB6
15	LB1, LB5, LB6
16	LB1, LB5
17	LB1, LB5, LB6
18	LB5, LB6
19	LB1, LB5, LB6
20	LB1, LB5, LB6
21	LB1, LB5
22	LB1, LB5, LB6
23	LB1, LB5, LB6
24	LB1, LB5, LB6
25	LB1, LB5, LB6

Table 2: Suitable lower bounds

Bounds	LB1	LB2	LB3	LB4	LB5	LB6
Time	0	0.0000033	0.000043	0.050016	0.007448	0.0014001
Performance	6565.55	5343.14	801.67	2039.64	6600.27	6798.53

Table 3: Mean values for the lower bounds and their computation times

The computational results of the branch and bound algorithm are given in Tables 4-7 where "Node", "Time" and " NS" represent respectively, the average of nodes generated, the computation time in CPU seconds and the number of problems not solved over 10 problems of the corresponding class in a period of 20 minutes.
We note that the branch and bound algorithm use for each group the maximum of the suitable lower bounds of Table 2.
As shown in Tables 4-7, the performance of the proposed algorithm BAB depends on the tightness of due dates T and its relative range R.
Numerical tests show that, for all the problems when both T and R are large, optimal solutions can be obtained easily. But, when T and R are small, the BAB algorithm does not give a good solution, possibly because most jobs are not tardy and the lower bound cannot eliminate the nodes.
We note that BAB algorithm is efficient for instances when T is small and R is large. Indeed, the problems were solved in relatively short time possibly because the most jobs are tardy and the majority of nodes are eliminated by the lower bound.
With our branch and bound algorithm 74.80% of the problems with 20 jobs and 3 machines are solved. This percentage decreases when the number of jobs increases and we have only 32.80% of the problems with 30 jobs and 3 machines that are solved, which proves that the performance of the BAB algorithm is badly affected by the number of jobs. This is due to the computational effort needed and the limitation time to 20 minutes.

Jobs	m=2		m=3	
	Node	NS	Node	NS
5	34.00	0	136.33	0
10	20125.99	0	10563.25	0
15	69114056.15	11.60	32439763.81	4.00
20	127650343.48	22.80	127704389.53	25.20
25	151703346.82	34.40	179757404.74	33.20
30	155477253.54	44.80	236961094.45	67.20

Table 4: Mean values for the number of generated nodes and percentage of unsolved problem

Groups	m=2			m=3		
	Node	Time	NS	Node	Time	NS
1	291146.00	0.81	0	39815.20	0.07	0
2	4122.60	0.01	0	6767.70	0.01	0
3	1476.20	0.01	0	10844.10	0.02	0
4	613.20	0.00	0 0	9282.60	0.02	0
5	1128.40	0.01	0	11180.90	0.02	0
6	139848.50	0.21	0	15115.50	0.03	0
7	4977.30	0.01	0	6802.90	4.22	0
8	821.30	0.00	0	9199.80	0.02	0
9	754.40	0.01	0	85101.10	0.02	0
10	987.50	0.00	0	12573.10	0.03	0
11	12389.50	0.02	0	25054.90	0.04	0
12	13560.20	0.03	0	7089.40	1.14	0
13	811.60	0.01	0	7182.60	0.01	0
14	450.60	0.00	0	9421.40	0.02	0
15	718.90	0.00	0	9122.90	0.02	0
16	13122.20	0.02	0	3745.90	0.01	0
17	4505.00	0.01	0	4752.80	0.01	0
18	1505.10	5.25	0	7131.00	25.23	0
19	988.50	0.01	0	9661.70	0.02	0
20	1061.10	0.01	0	10762.20	0.02	0
21	3973.80	0.01	0	6566.20	0.02	0
22	513.80	0.00	0	4119.30	0.01	0
23	1905.40	0.03	0	6696.70	0.01	0
24	745.60	0.00	0	10178.20	0.02	0
25	1023.00	0.01	0	12504.20	0.03	0

Table 5: Performance of the branch-and-bound algorithm for $n = 10$ and $m = 2$, $m = 3$

Groups	m=2			m=3		
	Node	Time	NS	Node	Time	NS
1	663913450.00	1200.00	10	580526298.80	1200.00	10
2	525260439.50	1200.00	10	458772834.60	1200	10
3	12228613.70	153.55	1	237664605.20	560.63	3
4	59833.90	0.15	0	5082701.40	12.39	0
5	44395.30	0.09	0	4360029.90	9.57	0
6	662973142.30	1200.00	10	607236278.10	1200.00	10
7	492579567.90	1117.79	9	346003182.60	874.61	5
8	2394387.30	6.48	0	37219367.20	88.02	0
9	40742.80	12.285	0	3979911.80	9.70	0
10	29993.80	0.07	0	3553537.00	7.92	0
11	70330366.50	120.00	1	53423505.50	120.00	1
12	189142478.00	388.87	3	337447535.00	778.27	6
13	227047.40	0.62	0	6433085.40	15.92	0
14	66686.70	0.18	0	1074483.60	2.90	0
15	33505.60	39.71	0	2223988.60	5.10	0
16	62694487.20	120.00	1	1.00	0.00	0
17	380610050.70	834.74	6	265424812.30	690.76	5
18	160522.00	600.95	3	323435.80	1200.00	10
19	45704.10	0.13	0	1288037.60	3.45	0
20	30381.10	0.07	0	3146784.90	7.73	0
21	1.00	0.00	0	1.00	0.00	0
22	127874786.40	375.56	3	208351636.80	442.06	3
23	79224.00	0.20	0	22869574.60	46.44	0
24	382330.20	1.06	0	4910019.20	11.44	0
25	56457.90	0.16	0	1294138.90	3.29	0

Table 6: Performance of the branch-and-bound algorithm for $n = 20$ and $m = 2$, $m = 3$

Groups	m=2			m=3		
	Node	Time	NS	Node	Time	NS
1	549427658.20	1200.00	10	463034861.00	1200.00	10
2	339564437.30	1200.00	10	293200797.00	1200.00	10
3	289157154.50	1200.00	10	291467473.30	1200.00	10
4	112308722.90	425.19	3	252135869.10	979.21	7
5	2088963.70	4.83	0	396060690.00	1200.00	10
6	562160476.20	1200.00	10	483887281.70	1200.00	10
7	298789751.90	1200.00	10	292392009.30	1200.00	10
8	139849306.20	759.25	6	270121780.70	999.91	8
9	40805724.80	152.40	1	126729289.80	527.33	3
10	28753604.60	122.19	1	356498594.60	1038.47	1
11	1.00	0.00	0	43384521.90	120.00	1
12	331091720.20	1200.00	10	321586367.50	1200.00	10
13	116438045.30	493.78	4	210275609.00	824.52	6
14	72654154.80	304.60	2	157123784.60	618.71	4
15	61879033.30	258.91	2	272269885.00	929.22	6
16	1.00	0.00	0	1.00	0.00	0
17	395825072.70	1200.00	10	370376176.50	1200.00	10
18	290108.20	1200.00	10	293199.20	1200.00	10
19	36635345.60	143.14	0	222081760.60	869.28	6
20	66862059.70	258.16	2	190901312.50	698.28	4
21	37337081.30	120.00	1	1.00	0.00	0
22	165388510.30	610.59	5	144366276.40	746.11	6
23	221449366.60	788.00	5	290092109.20	1056.02	8
24	13096752.80	53.72	0	259854979.80	1031.79	8
25	5078307.50	20.26	0	215892689.20	848.18	5

Table 7: Performance of the branch-and-bound algorithm for $n = 30$ and $m = 2$, $m = 3$

4. CONCLUSION

This paper studies scheduling problems on parallel machines to minimize total weighted tardiness. We presented an optimal solution algorithm by developing several lower bounds on the tardiness of jobs for a partial schedule and a branch and bound algorithm using them. Computational experiments shows that we can find optimal solutions in some cases with 30 jobs and 3 machines within a reasonable amount of CPU time. In future research, we need to develop more effective rules and to devise heuristic algorithms that give good solution. Also, we can consider various constraints like dependent and independent setup times.

ACKNOWLEDGEMENT

This work is supported in part by the General Consul Champagne-Ardenne, France (Project OCIDI, grant UB902-CR20122-289E).

REFERENCES

Azizoglu, M. and Kirka, O., 1998. *Tardiness minimization on parallel machines.* International Journal of Production Economics 55 163-168.

Barnes, J. W. and Brennan, J. J., 1977. *An improved algorithm for scheduling jobs on identical machines.* AIIE Transactions, vol. 9, No. 1.

Baptiste, P. and Le Pape, C., 2005. *Scheduling a single machine to minimize a regular objective function under setup constraints.* Discrete Optimization, 2 83-99.

Bruno, J. W., Coffman, E. G. and Sethi, R., 1974. *Scheduling independent tasks to reduce mean finishing time.* AIIE Transactions, vol. 17, pp 382-387.

Eastman, W. L., Even, S. and Issacs, I. M., 1964. *Bounds for the optimal scheduling of n jobs on m processors.* Management Science, vol. 11, pp 268-279

Elmaghraby, S. E. and Park, S. H., 1974. *Scheduling jobs on a number of identical machines.* AIIE Transactions, vol. 6, No. 1.

Kacem. I., 2007. *Lower bounds for Tardiness Minimization on a Single Machine with Family Setup Times.* International Journal of Operations Research. Vol 4, No 1, 1-14 .

Potts, C. N. and Van Wassenhove, L. N., 1985. *A branch and bound algorithm for the total weighted tardiness problem.* Operations Research, vol. 33, No. 2 pp 363-377.

Sherali, H.D. and Ulular O., (1990) *Conjugate gradient methods using quasi-Newton updates with inexact line searches.* Journal of Mathematical Analysis and Applications, 150, 359-377, .

Shim, S-O. and Kim, Y-D., 2006. *Scheduling on parallel identical machines to minimize total tardiness.* European Journal of Operation Research.

Yalaoui, F. and Chu, C., 2002. *Parallel machines scheduling to minimize total tardiness.* International Journal of Production Economics 76, 265-279.