Volume 22 (1), pp. 77–87
http://www.orssa.org.za

ORiON
ISSN 0529-191-X
©2006

A rescheduling heuristic for the single machine total tardiness problem

JC Nyirenda[*]

Received: 19 September 2005; Revised: 18 November 2005; Accepted: 18 February 2006

Abstract
In this paper, we propose a rescheduling heuristic for scheduling N jobs on a single machine in order to minimise total tardiness. The heuristic is of the interchange type and constructs a schedule from the modified due date (MDD) schedule. Unlike most interchange heuristics that consider interchanges involving only two jobs at a time, the newly proposed heuristic uses interchanges that may involve more than two jobs at any one time. Experimental results show that the heuristic is effective at reducing total tardiness producing schedules that either similar or better than those produced by the MDD alone. Furthermore, when applied to some test problems the heuristic found optimal schedules to all of them.

Key words: Scheduling, heuristic, tardiness.

1 Introduction

The problem considered in this paper is as follows. There is a set $Z = \{1, 2, \ldots, N\}$ of jobs simultaneously available to be processed on a single machine under the common assumptions listed in Baker [2]. Job i requires t_i units of processing time, for all $i \in Z$, on the machine which can only process one job at a time without pre-emption and inserted idle time. In addition, each job i is given a due date d_i, for all $i \in Z$. If job i is finished late, a penalty equal to its tardiness $T_i = \max(C_i - d_i, 0)$ is incurred, where C_i is the completion time of job i. The objective is to find the optimal job sequence σ^* that minimises the total tardiness, that is

$$f(\sigma^*) = \min_{\sigma \in \pi} f(\sigma) \tag{1}$$

where $f(\sigma) = \sum_{i=1}^{N} \max(C_{[i]} - d_{[i]}, 0)$, where σ is an arbitrary job sequence, where the subscript $[i]$ denotes the job in position i of σ and where π is the set of all possible sequences of which there are $N!$.

According to Du and Leung [7], the total tardiness problem is NP-hard. As a result, all optimization methods that have been developed to solve the problem in (1) utilize

[*]Department of Statistical Sciences, University of Cape Town, Private Bag, Rondebosch 7701, South Africa, e-mail: nyirenda@stats.uct.ac.za

combinatorial techniques, such as branching and bounding, and dynamic programming (see Fisher [9], Emmons [8], Srinivasan [19], Picard and Queyranne [14], Potts and Van Wassenhove [15], Chu [5], Szwarc and Mukhopadhyay [20], Della Croce *et al.* [6] and Hirakawa [10], to mention but a few). The solution methods owe their efficacy to either the dominance rules of Emmons [8], or/and the decomposition theorem of Lawler [12], which are employed to help restrict the search for an optimal solution. Despite this, still, the methods require considerable computer memory and time to implement.

Consequently, a number of heuristics have been proposed for solving the problem in (1). These are methods for solving problems by an intuitive approach in which the structure of the problem can be interpreted and exploited intelligently so as to obtain a reasonable solution. Heuristic techniques are the only way of tackling many large optimisation problems. Noteworthy heuristics for solving the total tardiness problem include those by Wilkerson and Irwin [21], Baker and Bertrand [3], Potts and Van Wassenhove [16], Holsenback and Russell [11] and Panwalkar *et al.* [13].

In this paper, we shall focus our attention on the heuristic by Baker and Bertrand [3], known as the modified due date (MDD) rule. The MDD rule has been investigated by many researchers and has been shown to be very effective in reducing total tardiness and in some cases it has been found to produce optimal tardiness schedules (see Baker and Bertrand [3], Baker and Kanet [4]). Furthermore, recent research has shown that the heuristics of Wilkerson and Irwin [21], and Panwalkar *et al.* [13] are equivalent to the MDD rule (see Alidaee and Gopalan [1]). The remainder of this paper is organized as follows. In the following section we present and prove a further local optimality condition for a subsequence of consecutive jobs for the tardiness problem. In §3 we utilise the condition and one of Emmons' dominance rules to develop a heuristic which may be used to improve tardiness schedules generated by the MDD rule. The condition may be seen as a generalized form of the idea of reducible tardiness proposed by Holsenback and Russel [11]. In §4 we demonstrate the mechanics of the heuristic by solving a simple numerical example. This is followed by a large-scale experiment in §5 where we compare the performance of the heuristic to the MDD rule. We conclude the paper with some final remarks in §6.

2 A local optimality condition for jobs in a MDD schedule

Previously it was thought that the presence of overlapping tardiness intervals in tardiness schedules almost automatically led to sub-optimality. A tardiness interval of a job is defined as the difference between its completion time and its original due date. Thus, a job that is early has no tardiness interval. Contrary to these earlier expectations, some schedules that have exhibited this characteristic have been found to be optimal (see Baker [2]). Here, we shall discuss a further optimality condition for the mean tardiness problem that may be used together with the MDD rule to improve schedule performance. The condition is stated as follows.

Proposition 1 *For any given sequence σ, of N jobs, let there be a subsequence of r consecutive jobs, beginning in position k of σ, then if*

$$\max\left(\sum_{i=1}^{r-1} t_{[k+i]} - \max\left(d_{[k]} - C_{[k]}, 0\right), 0\right) < \sum_{i=1}^{r-1} \min\left(\max\left(C_{[k+i]} - d_{[k+i]}, 0\right), t_{[k]}\right)$$

an alternative, better sequence may be constructed by interchanging the job in the k^{th} position, say $J_{[k]}$, with the subsequence of jobs $(J_{[k+1]}, J_{[k+2]}, \ldots, J_{[k+r-1]})$ in σ.

Proof: In the optimal schedule under consideration jobs are either early or tardy. It is easy to show through pair-wise interchange argument that either our condition is satisfied or an alternate optimal schedule can be constructed. Interchanging job $J_{[k]}$ with a subsequence of jobs $(J_{[k+1]}, J_{[k+2]}, \ldots, J_{[k+r-1]})$ has the effect of delaying the completion time of $J_{[k]}$ by $\sum_{i=1}^{r-1} t_{[k+i]}$ thus increasing its tardiness by at most that amount, *i.e.* by

$$\max\left(\sum_{i=1}^{r-1} t_{[k+i]} - \max\left(d_{[k]} - C_{[k]}, 0\right), 0\right).$$

On the other hand, the subsequence of jobs will be completed t_k time units earlier, resulting in total decrease in tardiness of

$$\sum_{i=1}^{r-1} \min\left(\max\left(C_{[k+i]} - d_{[k+i]}, 0\right), t_{[k]}\right).$$

It follows that, if the condition fails, the jobs should be left in their original positions. Alternatively, if the condition holds, interchanging the jobs in the manner described will yield an improved solution. ■

3 A rescheduling heuristic for improving MDD schedules

The heuristic that we propose is of the interchange type and constructs a schedule from the MDD schedule. The heuristic functions by promoting further job interchanges that may lead to a reduction in tardiness in these schedules. Unlike most pair-wise interchange heuristics, which consider interchanges involving only two jobs at a time, our heuristic uses interchanges, which may involve more than two jobs at any one time. The dominance rule of Emmons [8], which we incorporate into the heuristic, is used to act as a guiding instrument for identifying which jobs in the MDD schedule the heuristic must consider when searching for sub-optimal condition. The rule is expressed in the theorem below. However, the following lemma is required for the proof of the theorem.

Lemma 1 *If $x \geq y$ and $u \geq v$ then* $\max(u, y) + \max(v, x) \geq \max(u, v) + \max(v, y)$.

Proof: If $u \geq x$, then

$$\max(u, y) + \max(v, x) \geq u + \max(v, y) = \max(u, x) + \max(v, y).$$

Otherwise if $u < x$, then

$$\max(u,y) + \max(v,x) \geq \max(v,y) + x = \max(v,y) + \max(u,x) \qquad \blacksquare$$

We now state and prove one of Emmons theorems. The proof is new and hence different from Emmons' original proof.

Theorem 1 (Emmons [8]) *If $t_i \leq t_j$ and $d_i \leq \max(d_j, T+t_j)$ for some T, then amongst the schedules which start both jobs i and j after time T, there is an optimal schedule with job i before job j (jobs i and j need not be adjacent).*

Proof: Suppose that the theorem is not true. Then there is an optimal schedule in which job j occurs before job i. Let this schedule be as shown in Figure 1 and let $s_o > T$. Consider swapping job i and job j. Since $t_i \leq t_j$ jobs between jobs i and j are moved forward and the tardiness contribution from job i and job j before the swap is $B = \max(0, s_o + t_j - d_j) + \max(0, x - d_i)$. There are two cases to consider.

Case 1: If $d_i \leq d_j$ then

$$\begin{aligned} B &= \max(d_j, s_o + t_j) + \max(d_i, x) - d_i - d_j \\ &\geq \max(d_j, x) + \max(d_i, s_o + t_j) - d_i - d_j \\ &= \max(0, x - d_j) + \max(0, s_o + t_j - d_i) \\ &\geq \max(0, x - d_j) + \max(0, s_o + t_i - d_i), \end{aligned}$$

using Lemma 1. The last expression on the right hand side above represents the tardiness of the swap.

Case 2: If $d_i \leq s_o + t_j$ and $d_i > d_j$ then $s_o + t_j > d_j$ and x (which is greater than $s_o + t_j$) is greater than d_i. Hence,

$$\begin{aligned} B &= s_o + t_j - d_j + x - d_i \\ &\geq s_o + t_j - d_i + x - d_j \\ &= \max(0, x - d_j) + \max(0, s_o + t_j - d_i) \\ &\geq \max(0, x - d_j) + \max(0, s_o + t_i - d_i). \end{aligned}$$

The last expression on the right hand side above again represents the tardiness of the swap. $\blacksquare$

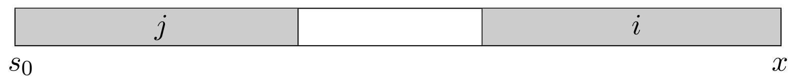

Figure 1: *An example of job i and j in the schedule.*

The MDD rule is consistent with Emmons' theorem above for jobs i and j when $t_i \leq t_j$ and $\max(d_i, T + t_i) \leq \max(d_j, T + t_j)$ since, in this case, the MDD rule also sequences job i before job j. Thus, in a sequence generated by the MDD rule, we may assume that if job i is sequenced before job j, where $t_i \leq t_j$, then job i is optimally sequenced with respect to job j so that interchanging these jobs would not yield a reduction in tardiness.

In the case where $t_i > t_j$, and job i is sequenced immediately before job j in the MDD schedule and job j is followed consecutively by at least one other short job (shorter than job j) and if at least one of the short jobs is tardy then the MDD schedule may not be optimal.

Let

- $J_{[i]}$ denote the job in position i of the schedule,
- $C_{[i]}$ denote the completion time of the job in position i of the schedule,
- $t_{[i]}$ denote the processing time of the job in position i of the schedule,
- $d_{[i]}$ denote the due date of the job in position i of the schedule,
- T denote to mean time now, that is $\sum_{k=1}^{i-1} t_{[k]}$
- r denote the number of short jobs consecutively following a long job in the schedule,
- N denote the total number of jobs requiring scheduling.

The procedure that we propose attempts to discover whether a long job which has been sequenced before a subsequence of short jobs in the MDD schedule should actually be occupying its present position or whether it should be preceded by at least two of the short jobs. The heuristic does this by sequentially searching through the MDD schedule beginning with the job in position $(N-2)$ right up to the job in first position.

1	2	......	i	$i+1$	......	$i+r-1$	$i+r$	......	$N-1$	N

Figure 2: *An example of a MDD job sequence for the purpose of our heuristic.*

Suppose that Figure 2 depicts part of the MDD sequence of N jobs, and suppose that the heuristic is currently at the job occupying the i^{th} position in the schedule. Call this job a candidate job and let T be the total processing time of the first $(i-1)$ jobs in the schedule. We assume that the jobs $J_{[i+k]}$, $k = 1, 2, 3, \ldots, N-i$, have already been checked and rescheduled where necessary by the heuristic. Suppose the heuristic is currently considering the job in position i, that is, job $J_{[i]}$. First, the heuristic checks to establish whether or not the processing time of job $J_{[i]}$ is greater than each of the processing times of at least two jobs immediately and consecutively following it in the schedule, *i.e.* whether $t_{[i]} > t_{[i+k]}$, where $k = 1, 2, \ldots, r$ and $r \geq 2$. If the condition is false, the heuristic leaves job $J_{[i]}$ in its current position and resumes the search with the next job up the schedule, that is, $J_{[i-1]}$ and T is decreased by $t_{[i-1]}$. However, suppose $t_{[i]} > t_{[i+k]}$, where $k = 1, 2, \ldots, r$ and $r \geq 2$ as depicted in Figure 2. Furthermore, suppose the heuristic is currently considering swapping job $J_{[i]}$ with a subsequence of p short jobs. Here, we assume the heuristic has already considered swapping in turn job $J_{[i]}$ with subsequences of $(1, 2, \ldots, p-1)$ short jobs and found that swapping would not yield a reduction in tardiness. Next the heuristic computes the tardiness contribution in the current schedule caused by a subsequence made up of the candidate job and the p short jobs following it, that is $J_{[i]}, J_{[i+1]}, \ldots, J_{[i+p]}$. We shall call this subsequence the current subsequence. The total tardiness figure for the *current subsequence* is compared with the total tardiness that would result if an *alternative subsequence* of the same jobs were considered in which the long job is now preceded by the p short jobs.

Notice that in the result of Proposition 1, it is implicitly assumed that if a swap such as we have described above were made, then the short jobs in the revised schedule would retain their relative positions with respect to each other. However, we note that, because the scheduled time for the short jobs to start processing is brought forward, the short jobs may be required to change positions in the alternative subsequence depending on their processing time and due dates. The heuristic thus uses the MDD rule to assign positions to the short jobs in the revised schedule, starting at time $T = \sum_{k=1}^{i-1} t_{[k]}$.

If the tardiness contribution of the alternative subsequence is less than that of the current subsequence then the current subsequence is replaced with the alternative subsequence in the schedule, and if $p < r$ then the heuristic moves to job $J_{[i+p]}$ where it continues the search and T is increased by $\sum_{k=0}^{p-1} t_{[i+k]}$; otherwise, if $p = r$ the heuristic continues its search with $J_{[i+p-1]}$ and T is increased by $\sum_{k=0}^{p-2} t_{[i+k]}$. However, if the tardiness contribution of the alternative subsequence is greater than or equal to that of the current subsequence then the current subsequence is retained in the schedule and the heuristic continues its search by moving to job $J_{[i-1]}$ and T is decreased by $t_{[i-1]}$. The process is repeated and terminates when the heuristic attempts to reschedule a job in position 0.

The heuristic described above may be implemented as an algorithm. Let $p \in \{0, 1, 2, \ldots, r\}$, let * denote a long job in a subsequence, let T_C be the tardiness of the current subsequence defined as $T_C = \max(C^*_{[i]} - d^*_{[i]}, 0) + \sum_{k=1}^{p} \max(C_{[i+k]} - d_{[i+k]}, 0)$, and let T_A be tardiness of the alternative subsequence defined as $T_A = \sum_{k=0}^{p-1} \max(C_{[i+k]} - d_{[i+k]}, 0) + \max(C^*_{[i+p]} - d^*_{[i+p]}, 0)$, where in each case the p short jobs are assigned positions in the subsequences according to the MDD rule. The steps of the heuristic may now be described as follows.

Step 0

(Initialisation) Schedule the N jobs according to the MDD rule. Make job $J_{[N-2]}$ a candidate job. Set $p \leftarrow 0$; $i \leftarrow N - 2$ and $T \leftarrow \sum_{k=1}^{i-1} t_{[k]}$, *i.e.* the total processing time of the jobs preceding job $J_{[N-2]}$ in the schedule.

Step 1

If $i = 0$ then STOP. Set $p \leftarrow p + 1$. If $i + p > N$ or $t_{[i]} \leq t_{[i+p]}$ then go to step 2, else go to step 3.

Step 2

Leave $J_{[i]}$ in its current position and decrease T by $t_{[i]}$. Set $p \leftarrow 0$, $i \leftarrow i - 1$. Go to step 1.

Step 3

If $T_A < T_C$ go to step 4, else go to step 1.

Step 4

Replace the current subsequence with the alternative subsequence so that job $J_{[i]}$ in the original schedule becomes job $J_{[i+p]}$ in the revised schedule and assign positions to the p short jobs according to the MDD rule starting at time T. Set $p \leftarrow 0$, $i \leftarrow i + p$ and increase T by $\sum_{k=1}^{p} t_{[i+k-1]}$. Go to step 1.

Notice that during the actual implementation of the heuristic long jobs, which are immediately succeeded, by a single short job are also considered as candidates (*i.e.* when p=1 above). In this case, subsequences are made up of only two jobs, $J_{[i]}$ and $J_{[i+1]}$. Observe that a swap involving two such jobs in the original MDD schedule would not yield any improvement with regard to total tardiness. However, due to rescheduling, which occurs when implementing our heuristic, there is a possibility of a long job being placed just before a short job in the revised schedule such that if a swap were made between the long job and the short job a reduction in tardiness would result. The heuristic is very effective in improving MDD schedules having found optimal solutions to all test problems in Baker [2].

4 A numerical example

We now illustrate the working of the heuristic by means of a worked numerical example. Consider a set of 8 jobs in MDD order with their processing times t_i, due dates d_i and tardiness T_i, as shown in Table 1.

Job	1	2	3	4	5	6	7	8
t_i	121	79	83	102	130	147	88	96
d_i	260	266	336	400	337	269	719	683
T_i	0	0	0	0	178	393	31	163

Table 1: *MDD schedule processing times, due dates and tardiness values.*

The total tardiness for this schedule is 765. The heuristic starts by considering job $J_{[6]}$ as a candidate job for rescheduling. Job $J_{[6]}$ is followed by two short jobs: $J_{[7]}$ and $J_{[8]}$. Initially the heuristic attempts to interchange job $J_{[6]}$ with a subsequence made up of just one short job, namely job $J_{[7]}$. But the total tardiness of the alternate subsequence of 481 is greater than the total tardiness of the current subsequence of 424 and therefore no swapping is carried out. Next the heuristic considers swapping job $J_{[6]}$ with a subsequence of short jobs made up of jobs $J_{[7]}$ and $J_{[8]}$. Here the total tardiness of the alternate subsequence of 577 is less than the total tardiness of the current subsequence of 587. Thus the current subsequence is replaced by the alternate subsequence resulting in schedule shown in Table 2. Note that the two short jobs have been assigned positions in the revised schedule of Table 2 according to the MDD rule, starting at time $\sum_{i=1}^{5} t_{[i]} = 515$.

Next the heuristic again selects the recently rescheduled long job, now $J_{[8]}$, in the revised schedule as a candidate for rescheduling. But because job $J_{[8]}$ is in the last position of the schedule, it is fixed in that position. The heuristic moves one step up to job $J_{[7]}$ and makes this job its candidate for rescheduling. However, because $t_{[7]} < t_{[8]}$, the heuristic

Job	1	2	3	4	5	6	7	8
t_i	121	79	83	102	130	96	88	147
d_i	260	266	336	400	337	683	719	269
T_i	0	0	0	0	178	0	0	577

Table 2: *Revised schedule following a swap.*

leaves job $J_{[7]}$ in its current position and again moves one step up, now to $J_{[6]}$, in the revised schedule of Table 3. Job $J_{[6]}$ is followed by only one short job. Swapping these jobs would not reduce tardiness and therefore the heuristic leaves job $J_{[6]}$ in its current position and moves up to the next job in the schedule. The next job in the schedule is job $J_{[5]}$ and it is followed by two short jobs, namely $J_{[6]}$ and $J_{[7]}$. However, because both the short jobs are early in the revised schedule, interchanging job $J_{[5]}$ with either job $J_{[6]}$ or a subsequence made up of jobs $J_{[6]}$ and $J_{[7]}$ will not reduce tardiness; therefore the heuristic leaves job $J_{[5]}$ in its current position and moves up to the next job in the schedule, *i.e.* job $J_{[4]}$. Repeating the process in this way we find that there is no other job in the revised schedule that requires rescheduling — either because the processing time of the candidate job is less than its immediate successor or because the overall reduction in tardiness if a swap were made is not greater than 0. The total tardiness for the improved schedule is therefore 755 and it turns out to be the optimal tardiness value for the problem (see Baker [2]).

5 Experimental results

An experiment involving 20 problems was conducted in order to compare the performance of the proposed heuristic and the MDD rule on the total tardiness problem. In the experiment, each of the 20 problems consist of scheduling 100 jobs. The 20 problems were generated in the manner suggested by Potts and Van Wassenhove [15], as follows. First, for each job i, $(i = 1, 2, \ldots, 100)$, an integer processing time t_i was generated from a uniform distribution on the interval $[1, N]$, and then the total processing time, $P = \sum t_i$, $(i = 1, 2, \ldots, 100)$ was computed. The relative range of due dates (RDD) and the average tardiness factor (TF) were selected from the sets $\{0.2, 0.4, 0.6, 0.8, 1.0\}$ and $\{0.2, 0.4, 0.6, 0.8\}$ respectively, and for each job i a due date d_i was generated from a uniform distribution on the interval $[P(1-TF-RDD/2),\ P(1-TF+RDD/2)]$. Fisher [9], Schrage and Baker [18], and Potts and Van Wassenhove [17] have observed that problems with $TF = 0.6$ and $TF = 0.8$ appear to be the hardest to solve, particularly when RDD is small.

The results in Table 3 compare the relative performance of the MDD sequence produced with and without the aid of the rescheduling heuristic. Note that the application of the heuristic leads to an improvement in the performance of the MDD sequence. The greatest improvement occurred in problems with small values of both RDD and TF.

		Tardiness		% improvement
RDD	TF	MDD	MDD+ heuristic	over MDD by MDD+heuristic
0.2	0.2	2921	2218*	24
0.2	0.4	22195	20072*	10
0.2	0.6	57772	53626*	7
0.2	0.8	108443	102763*	5
0.4	0.2	28	28	0
0.4	0.4	14195	12513*	12
0.4	0.6	48202	45310*	6
0.4	0.8	102808	102253*	1
0.6	0.2	0	0	0
0.6	0.4	7748	7373*	5
0.6	0.6	40828	39146*	4
0.6	0.8	105367	105286	0
0.8	0.2	0	0	0
0.8	0.4	1788	1778*	1
0.8	0.6	36806	36442*	1
0.8	0.8	109626	109581	0
1.0	0.2	0	0	0
1.0	0.4	0	0	0
1.0	0.6	39447	39243*	1
1.0	0.8	114495	114494	0

Table 3: *Tardiness values obtained for the MDD rule with and without the aid of the heuristic. The results marked with asterisk represent the best performance for the heuristic.*

6 Conclusion

In this paper we have presented and proved a further local optimality condition for a subsequence of consecutive jobs for the total tardiness problem. We have developed a heuristic by incorporating the local optimality condition, one of Emmons' dominance rules and the MDD rule. The heuristic utilizes the MDD sequence as seed to generate new improved tardiness sequences. The results of a computational experiment show that the rescheduling heuristic is effective in minimizing total tardiness. The heuristic generates sequences that are either better or similar to those produced by the MDD rule. Furthermore, when tried on some test problems, the heuristic was able to find optimal solutions for all the problems. In future it will be useful to compare empirically the performance of the proposed heuristic to the available heuristics for the single machine tardiness problem.

Acknowledgements

The author would like to thank the two anonymous referees whose comments helped to improve this paper.

References

[1] Alidaee B & Gopalan S, 1997, *A note on the equivalence of two heuristics to minimize total tardiness*, European Journal of Operational Research, **96**, pp. 514–

517.

[2] Baker KR, 1974, *Introduction to sequencing and scheduling*, Wiley, New York (NY).

[3] Baker KR & Bertrand JWM, 1982, *A dynamic priority rule for scheduling against due dates*, Journal of Operations Management, **3**, pp. 37–42.

[4] Baker KR & Kanet JJ, 1983, *Job shop scheduling with modified due dates*, Journal of Operations Management, **4**, pp. 11–22.

[5] Chu C, 1992, *A branch and bound algorithm to minimize total tardiness with different due dates*, Naval Research Logistics, **39**, pp. 265–283.

[6] Croce FD, Tadei R, Baracco P & GRSSO A, 1998, *A new decomposition approach for the single machine total tardiness scheduling problem*, Journal of the Operational Research Society, **49**, pp. 1101–1106.

[7] Du J & Leung JYT, 1990, *Minimising total tardiness on one machine is NP-hard*, Mathematics of Operations Research, **15**, pp. 483–495.

[8] Emmons H, 1969, *One-machine sequencing to minimise certain functions of job tardiness*, Operations Research, **17**, pp. 701–715.

[9] Fisher ML, 1976, *A dual algorithm for the one machine scheduling problem*, Mathematical Programming, **11**, pp. 229–251.

[10] Hirakawa Y, 1999, *A quick algorithm for the sequencing on one machine to minimize total tardiness*, International Journal of Production Economics, **60-61**, pp. 549–555.

[11] Holsenback JE & Russell RM, 2000, *A heuristic algorithm for sequencing on one machine to minimize total tardiness*, Journal of the Operational Research Society, **53**, pp. 800–806.

[12] Lawler EL, 1977, *A 'pseudopolynomial' algorithm for sequencing jobs to minimize total tardiness*, Annals of Discrete Mathematics, **1**, pp. 331–342.

[13] Panwalkar SS, Smith ML & Koulamas CP, 1993, *A heuristic for the single machine tardiness problem*, European Journal of Operational Research, **70**, pp. 304–310.

[14] Picard J & Queyranne M, 1978, *The time-dependent traveling salesman problem and its application to the tardiness problem in one machine scheduling*, Operations Research, **26**, pp. 86–100.

[15] Potts CN & Van Wassenhove LN, 1997, *Dynamic programming and decomposition approaches for the single machine total tardiness problem*, European Journal of Operations Research, **32**, pp. 405–414.

[16] Potts CN & Van Wassenhove LN, 1991, *Single machine tardiness sequencing heuristics*, Institute of Industrial Engineers Transactions, **23**, pp. 346–354.

[17] Potts CN & Van Wassenhove LN, 1982, *A decomposition algorithm for the single machine total tardiness problem*, Operations Research Letters, **1**, pp. 177–181.

[18] Schrage L & Baker KR, 1978, *Dynamic programming solution of sequencing problems with precedence constraints*, Operations Research, **26**, pp. 444–449.

[19] Srinivasan V, 1971, *A hybrid algorithm for the one machine sequencing problem to minimize total tardiness*, Naval Research Logistics Quarterly, **18**, 317–327.

[20] Szwarc W & Mukhopadhyay SK, 1996, *Decomposition of the single machine total tardiness problem*, Operations Research Letters, **19**, pp. 243–250.

[21] Wilkerson LJ & Irwin JD, 1971, *An improved algorithm for scheduling independent tasks*, American Institute of Industrial Engineers Transactions, **3**, pp. 239–245.