Some Properties of the Optimal Decomposition Conditions for the Single Machine Tardiness Problem

Jaideep T. Naidu
Thomas Jefferson University

We consider the well-known optimal decomposition algorithm for the tardiness problem. The algorithm presents conditions which determine the positions a job could occupy in an optimal sequence. This resulted in optimal solutions for up to 100 jobs. We analyze these conditions and present simplified conditions. We then study a more recent Rule which when combined with these conditions resulted in optimal solutions for up to 500 jobs. We present several properties under which this recent rule is satisfied. We provide mathematical proofs for our properties. We believe that our study will enable more theoretical research in this field and will eventually enable optimal solutions for very large job sets.

Keywords: tardiness, decomposition, optimal algorithm, machine scheduling

INTRODUCTION

In various production settings, there is typically a single bottleneck machine. Avoiding late shipments is the most important goal of the shop floor manager. Cost of late shipments, loss of goodwill and loss of potential customers can be detrimental to any organization in the long term. Hence, the classical single machine tardiness problem has been extensively studied by researchers (Naidu, 2011).

The single machine total tardiness problem (1//TT problem) may be stated as follows. There is a set of n jobs at time zero ready to be scheduled on a single machine which is continuously available. Each of the n jobs (1, …, n) is to be processed without interruption on a single machine which can handle only one job at a time. For a job j, there is a processing time p_j, and a due date d_j. Given a processing order of the jobs, its completion time and its tardiness will be $C_j = \sum_{i=1}^{j} p_i$ and $t_j = \max\{0, C_j - d_j\}$ respectively.

The objective is to find a processing order which minimizes the total tardiness $\sum_{j=1}^{n} t_j$. The single machine tardiness problem has always been considered to be an intractable problem. This problem has been shown to be NP-hard (Du and Leung, 1990). This means the computational time increases exponentially with the increase in the number of jobs that need to be processed. A dynamic programming algorithm (Potts and Van Wassenhove, 1982) was developed and was based on a decomposition theorem (Lawler, 1977). This dynamic programming algorithm presented three conditions that determine the positions that a job could occupy in an optimal sequence. This significantly reduced the core storage requirements and enabled optimal solutions for up to 100 jobs. In a more recent study, an Elimination Rule (Szwarc, 1993) was presented which further reduced the number of positions a job could occupy. Based on this elimination rule, optimal solutions for problems up to 150 jobs have been reported (Szwarc and Mukhopadhyay, 1996). In a later study (Naidu et al. 2002), it was reported that the structure of this optimal algorithm is strikingly similar to that of a well-known heuristic. The most recent comprehensive study (Koulamas, 2010) of the single machine tardiness problem emphasizes on the importance of further theoretical study on the design of exact algorithms for the 1//TT problem so that optimal solutions for job sets of more than 500 jobs are possible. A new heuristic algorithm which focused on a special structure of this problem was later proposed (Naidu, 2011). More recently, a deep learning-driven algorithm based on decomposition and symmetric decomposition was presented (Bouska et al. 2023).

THE DECOMPOSITION CONDITIONS – A SIMPLER APPROACH

In this section, we first present the decomposition algorithm of Lawler and the decomposition conditions of Potts and Van Wassenhove. We then interpret these conditions by presenting properties of certain jobs that hold when these conditions are satisfied. Using these properties, we present simplified conditions of these properties and illustrate our simplified conditions and the entire process of decomposition with the aid of a numerical example.

The Decomposition Conditions of Potts and Van Wassenhove

Label the jobs in EDD order ($d_1 \leq \ldots \leq d_n$), and ties broken by the SPT rule ($p_j \leq p_{j+1}$). Let $p_j = \max_{i=1,\ldots,n}\{p_i\}$ with ties broken by choosing the rightmost job. A problem decomposes with job j in position k ($j \leq k$) if the search for an optimal solution is restricted to schedules in which jobs 1, …, j-1, j+1, …, k are scheduled in the first k-1 positions, jobs k+1,…, n are scheduled in the last n-k positions, and job j in position k. The results of Lawler [2] and Potts and Van Wassenhove [3] show that the problem decomposes with job j in position k, for some k satisfying one of the following three conditions:

$$k = j \text{ and } C_j < d_{j+1} \tag{1}$$

$$k = j + l, \ldots, n-1 \text{ and } d_k \leq C_{k-1} < d_{k+1} - p_k \tag{2}$$

$$k = n \text{ and } C_{n-1} \geq d_n \tag{3}$$

There are often several candidate values of k satisfying these conditions. Once k is selected the problem decomposes into two sub-problems. The first of which is a (k-1)-job problem in which jobs 1, …, j-1, j+1, …, k are to be scheduled, starting at time zero. In the second (n-k)-job problem, jobs k+1, …, n are to be scheduled, starting at time C_k. Recursive application of the conditions for each of these sub-problems could generate more sub-problems which can then be solved by a dynamic programming algorithm. In the case of small problem sizes where core storage requirements is not an issue, the decomposition into sub-problems will be basically continued until each of the sub-problems contain only one job and then the sequence of all the jobs will be fixed.

We use the following additional notation in this paper

t_k represents the tardiness of a job in position k i.e., $t_k = C_k - d_k$;
s_k represents the slack of a job in position k i.e., $s_k = d_k - C_k$;

We now present properties (in the form of Remarks) of certain jobs under which one of the three conditions of Lawler and Potts and Van Wassenhove hold. We first present a property based on Condition 1.

Remark 1. The tardiness (if tardy) of job j+1 is less than its processing time (i.e., $t_{j+1} < p_{j+1}$) when their Condition 1 is satisfied.
Proof: $C_j < d_{j+1} \Rightarrow C_j + p_{j+1} - d_{j+1} < p_{j+1} \Rightarrow t_{j+1} < p_{j+1}$.

We next present three properties based on Condition 2: $d_k \leq C_{k-1} < d_{k+1} - p_k$. The following property is based on the inequality $d_k \leq C_{k-1}$.

Remark 2. Job k is tardy and its tardiness t_k is at least equal to its processing time (i.e., $t_k \geq p_k$) when the inequality: $d_k \leq C_{k-1}$ of Condition 2 is satisfied.
Proof: $d_k \leq C_{k-1} \Rightarrow C_{k-1} \geq d_k \Rightarrow C_{k-1} + p_k - d_k \geq p_k \Rightarrow t_k \geq p_k$.

Remark 3. Job k-1 cannot be early when the inequality: $d_k \leq C_{k-1}$ of Condition 2 is satisfied.
Proof: $d_k \leq C_{k-1} \Rightarrow C_{k-1} \geq d_k \Rightarrow C_{k-1} \geq d_{k-1}$ (since $d_k \geq d_{k-1}$). Thus, Job k-1 cannot be early. It will be on time or tardy.

Next, consider the inequality $C_{k-1} < d_{k+1} - p_k$ of Condition 2. This is same as $C_k < d_{k+1}$.

Remark 4. The tardiness (if tardy) of job k+1 is less than its processing time (i.e., $t_{k+1} < p_{k+1}$) when the inequality: $C_{k-1} < d_{k+1} - p_k$ of Condition 2 is satisfied.
Proof: $C_{k-1} < d_{k+1} - p_k \Rightarrow C_{k-1} + p_k < d_{k+1} \Rightarrow C_k < d_{k+1} \Rightarrow C_k + p_{k+1} - d_{k+1} < p_{k+1} \Rightarrow t_{k+1} < p_{k+1}$.

We now present a property based on Condition 3.

Remark 5. Job n is tardy and its tardiness is at least equal to its processing time (i.e., $t_n \geq p_n$) when their Condition 3: $C_{n-1} \geq d_n$ is satisfied.
Proof: $C_{n-1} \geq d_n \Rightarrow C_{n-1} + p_n - d_n \geq p_n \Rightarrow t_n \geq p_n$.

Based on Remarks 1, 2, 4, and 5, we present simplified conditions of Lawler [2] and Potts and Van Wassenhove [3] as follows. We refer to them as Conditions 1^*; 2^*; and 3^*.

$k = j$ and $t_{j+1} < p_{j+1}$ (**1^***)
$k = j+1, \ldots, n-1$ and $t_k \geq p_k$ & $t_{k+1} < p_{k+1}$(**2^***)
$k = n$ and $t_n \geq p_n$ (**3^***)

Note that in the case of Condition 2^*, both $t_k \geq p_k$ & $t_{k+1} < p_{k+1}$ must be satisfied for job j to decompose in any of the positions $k = j+1, \ldots, n-1$.

A Numerical Example to Illustrate the Decomposition Procedure Using Our Simplified Conditions

We present a numerical example to explain and illustrate the entire decomposition procedure based on our simplified conditions presented above. Consider a set of 6 jobs in EDD order with their processing times p_i, due dates d_i, and tardiness t_i as given below.

TABLE 1

Job	1	**2**	3	4	5	6	
p_i	7	20	15	8	10	12	
d_i	10	25	40	41	52	55	
t_i	0	2	2	9	8	17	$\sum TT = 38$

where TT is total tardiness of all the jobs

Note that $p_2 = \max_{i=1,\ldots,6}\{p_i\}$. Since $t_3 < p_3$, Condition 1^*: $t_{j+1} < p_{j+1}$ is satisfied. Hence $k = 2$ is a candidate decomposition position for job 2. For job 2 to decompose in positions 3, 4, and 5, Condition 2^*: $t_k \geq p_k$ *and* $t_{k+1} < p_{k+1}$ needs to be satisfied at each of these positions. Since $t_3 < p_3$, $t_k \geq p_k$ is not satisfied. Hence job 2 does not decompose in position k = 3. However, position k = 4 satisfies both $t_k \geq p_k$ *and* $t_{k+1} <$

p_{k+1} of Condition 2[*] (since $t_4 > p_4$ and $t_5 < p_5$). Thus job 2 decomposes in position k = 4. And position k = 5 is not a candidate for decomposition since $t_5 < p_5$ and $t_6 > p_6$. Finally, Condition 3[*]: $t_n \geq p_n$ is satisfied (since $t_6 > p_6$) and hence position $k = 6$ is a candidate for decomposition. Thus job 2 has three candidate positions k = 2; k = 4; and k = 6 and the resulting sequences that will be considered for further decomposition are: Sequence 1. [1,**2**,3,4,5,6]; Sequence 2. [1,3,4,**2**,5,6]; and Sequence 3. [1,3,4,5,6,**2**].

Decomposition of Sequence 1. Let us first consider [1,**2**,3,4,5,6] where job 2 is fixed in position [2] resulting in two subsequences {1}, [2], {3, 4, 5, 6}.

TABLE 2

Job	1	**2**	**3**	4	5	6	
p_i	7	**20**	15	8	10	12	
d_i	10	**25**	40	41	52	55	
t_i	0	**2**	2	9	8	17	$\sum TT = 38$

Note that $p_3 = \max_{i=3, 4, 5, 6}\{p_i\}$. Since $t_4 > p_4$ and $t_5 < p_5$, Condition 2[*]: $t_k \geq p_k$ *and* $t_{k+1} < p_{k+1}$ is satisfied and thus job 3 decomposes in position k = 4. Finally, Condition 3[*]: $t_n \geq p_n$ is satisfied (since $t_6 > p_6$) and hence position $k = 6$ is a candidate for decomposition. Thus, job 3 has two candidate positions k = 4 and k = 6 and the resulting sequences that will be considered for further decomposition are:
Sequence 1a. [1,2,4,**3**,5,6]; and Sequence 1b. [1,2,4,5,6,**3**].

TABLE 3

Job	1	**2**	3	**4**	5	6	
p_i	7	**20**	8	**15**	10	12	
d_i	10	**25**	41	**40**	52	55	
t_i	0	**2**	0	**10**	8	17	$\sum TT = 37$

Note that the above Sequence 1a cannot be decomposed any further. As for Sequence 1b i.e., [1,2,4,5,6,**3**], see Table 4 below.

TABLE 4

Job	1	**2**	4	5	6	**3**	
p_i	7	**20**	8	10	12	**15**	
d_i	10	**25**	41	52	55	**40**	
t_i	0	**2**	0	0	2	**32**	$\sum TT = 36$

As is obvious from TABLE 4, Sequence 1b also cannot be decomposed further.

Decomposition of Sequence 2. Let us consider [1,3,4,**2**,5,6] where job 2 is fixed in position [4] resulting in two subsequences {1, 3, 4}, [2], {5, 6}.

TABLE 5

Job	1	3	4	**2**	5	6	
p_i	7	15	8	**20**	10	12	
d_i	10	40	41	**25**	52	55	
t_i	0	0	0	25	8	17	$\sum TT = 50$

From Table 5 above, it is clear that Sequence 2 cannot be decomposed further.

Decomposition of Sequence 3. Let us consider [1,3,4,5,6,**2**] where job 2 is fixed in position [6] resulting in two subsequences {1, 3, 4, 5, 6}, [2].

TABLE 6

Job	1	3	4	5	6	**2**	
p_i	7	15	8	10	12	**20**	
d_i	10	40	41	52	55	**25**	
t_i	0	0	0	0	0	47	$\sum TT = 47$

From Table 6 above, it is clear that Sequence 3 also cannot be decomposed any further.

Due to these decomposition conditions, only *six* sequences had to be considered here to find the optimal solution. The optimal solution happens to be the sequence [1, 2, 4, 5, 6, 3] in Table 4 above and the total tardiness = 36 units. Without such decomposition conditions, one would end up considering 6! (6 factorial) i.e., 720 sequences to find the optimal solution. It is interesting to note that for large problem sets, the core storage requirements are high even with the aid of decomposition conditions. That is the primary reason that optimal solutions for up to 100 jobs could be found with these conditions. If one could eliminate more sequences, then optimal solutions for larger job sets can be found.

PROPERTIES BASED ON SZWARC'S ELIMINATION RULE

Here we first present Szwarc's Rule. Consider a set of jobs 1, …, **j**, …, i, …, r, …, n in EDD order ($j < i < r$) and where $p_j = \max_{i=1,\dots,n}\{p_i\}$.

Szwarc's Rule [4]. *Eliminate position r for job j if there is a job i ($j < i < r$ and $d_i > d_j$) such that $C_r \leq d_i + p_i$.*

To illustrate this, consider the earlier numerical example where positions 2, 4, and 6 were candidates for further decomposition. According to Szwarc [4], job 2 does not decompose in position k = 4 since $C_4 = 50$ and $d_3 + p_3 = 55$ thus satisfying Szwarc's Rule.

We present the following properties which are satisfied when Szwarc's Rule holds.

Property 1. *The tardiness (if tardy) of job i has to be less than its processing time (i.e., $t_i < p_i$) for Szwarc's Elimination Rule to hold.*

Proof: Consider the set of jobs 1, …, j, …, i, …, r, …, n. For Szwarc's Rule to hold, $C_r \leq d_i + p_i$. And $C_r \leq d_i + p_i \Rightarrow C_i < d_i + p_i$ (since $C_i < C_r$). And $C_i < d_i + p_i \Rightarrow C_i - d_i < p_i$. Hence $t_i < p_i$. $\square$

Next, we present another property which holds whenever Szwarc's Rule is satisfied. Consider a set of jobs 1, …, **j**, …, i, …, r, …, n. We know that for job r to be on the list obtained by the conditions of Lawler

[2] and Potts and Van Wassenhove [3], the condition $t_r \geq p_r$ has to hold. And for Szwarc's rule to be satisfied, the condition $t_i < p_i$ has to hold (Property 1). We now present Property 2.

Property 2. *No job from the set of jobs i+1, ..., r can have tardiness greater than the processing time of job i for Szwarc's Rule to hold.*
Proof: Consider a set of jobs 1, …, j, …, i, …, k, …, r, …, n in EDD order. We prove Property 2 by contradiction. Let job k have tardiness greater than the processing time of job i. That is, let $t_k > p_i$. Now $t_k > p_i \Rightarrow C_k - d_k > p_i \Rightarrow C_k > p_i + d_k \Rightarrow C_k > p_i + d_i$ (since $d_k \geq d_i$). But for Szwarc's rule to hold, $C_r \leq p_i + d_i$. And since $C_k < C_r$, the condition $C_k > p_i + d_i$ clearly contradicts Szwarc's Condition. Thus, no job between i and r can have tardiness greater than the processing time of job i.

Let job r have tardiness greater than the processing time of job i. That is, let $t_r > p_i$. Now $t_r > p_i \Rightarrow C_r - d_r > p_i \Rightarrow C_r > p_i + d_r \Rightarrow C_r > p_i + d_i$ (since $d_r \geq d_i$). But for Szwarc's rule to be satisfied, the condition $C_r \leq p_i + d_i$ has to hold. And the above condition $C_r > p_i + d_i$ clearly contradicts Szwarc's Condition. Thus, none of the jobs i+1, …, r can have tardiness greater than the processing time of job i for Szwarc's rule to hold. $\square$

ADDITIONAL PROPERTIES BASED ON TWO CLASSES OF SEQUENCES

Here, we present several more properties related to Szwarc's Rule. We consider two classes of sequences. Class I - where positions i, r are adjacent, and Class II - where i, r are not adjacent.

Class I. Consider a set of jobs 1,…, **j**,…, i, r,…, n in EDD order where $j < i < r$, jobs i, r are adjacent jobs and $p_j = \max_{i=1,\ldots,n}\{p_i\}$.
We present two properties for this class based on the following two sub cases.
(a) i is on time i.e., $C_i = d_i$; and (b) i is tardy i.e, $C_i > d_i$.

Property 3. *Given $C_i = d_i$, where i, r are adjacent positions ($j < i < r$), Szwarc's Rule is satisfied as long as: (i) $p_i \geq p_r$; and (ii)* $d_i = d_r$.
Proof: For Szwarc's Rule to hold, $C_r \leq d_i + p_i$. And $C_r \leq d_i + p_i \Rightarrow C_i + p_r \leq d_i + p_i \Rightarrow p_r \leq p_i \Rightarrow p_i \geq p_r$ (since $C_i = d_i$). Note however, that we assume that $d_i = d_r$. for Property 3 to hold. Otherwise it is not possible for job r to have tardiness equal to or greater than its processing time. $\square$
Next, we present Property 4 based on sub case (b).

Property 4. *Given $C_i > d_i$, where i, r are adjacent positions ($j < i < r$), Szwarc's Rule is satisfied as long as: $p_i \geq p_r + t_i$.*
Proof: $C_i > d_i$ (since i is tardy). For Szwarc's Rule to hold, $C_r \leq d_i + p_i$. Now $C_r \leq d_i + p_i \Rightarrow C_i + p_r \leq d_i + p_i \Rightarrow (C_i - d_i) + p_r \leq p_i \Rightarrow p_i \geq p_r + (C_i - d_i) \Rightarrow p_i \geq p_r + t_i$. $\square$

We now present a Numerical Example to illustrate Property 4 (see Table 7 below)

TABLE 7

Job	1	**2**	3	4	
p_i	7	20	15	8	
d_i	10	25	40	41	
t_i	0	2	2	14	$\sum TT = 18$

Note that the processing time of job 4 cannot be greater than 13 for Szwarc's Rule to hold. That is, Szwarc's Rule holds as long as $p_i \geq p_r + t_i$ where $t_i = 2$ in this case.

We consider only two sub cases in Class I because job i cannot be early. For position r to be on the list obtained by Lawler [2] and Potts and Van Wassenhove [3], the condition
$t_r \geq p_r$ has to hold. If job i is early i.e, $C_i < d_i$, it is not possible for job r to have tardiness greater than or equal to its processing time (since positions i and r are adjacent).

Class II. Consider a set of jobs 1,…, **j**,…, i, …, r,…, n in EDD order where $j < i < r$, jobs i, r are <u>not</u> adjacent jobs and $p_j = \max_{i=1,\ldots,n}\{p_i\}$.
We present three properties for this class since we have three sub cases here.
(a) i is on time i.e, $C_i = d_i$; (b) i is tardy i.e., $C_i > d_i$; and (c) i is early i.e., $C_i < d_i$.
Based on sub case (a), we present Property 5.

Property 5. *Given $C_i = d_i$, Szwarc's Rule is satisfied as long as: $p_i \geq (p_{i+1} + \ldots + p_r)$.*
Proof: For Szwarc's Rule to hold, the condition $C_r \leq d_i + p_i$ must be satisfied. And
$C_r \leq d_i + p_i \Rightarrow C_i + p_{i+1} + \ldots + p_r \leq d_i + p_i$. And since $C_i = d_i$, $p_i \geq (p_{i+1} + \ldots + p_r)$. $\square$

Based on sub case (b), we present Property 6 as follows.

Property 6. *Given $C_i > d_i$, Szwarc's Rule is satisfied as long as: $p_i \geq (p_{i+1} + \ldots + p_r) + t_i$.*
Proof: Since $C_i > d_i$, job i is tardy and its tardiness $t_i = C_i - d_i$. For Szwarc's Rule to hold, condition $C_r \leq d_i + p_i$ must be satisfied. And $C_r \leq d_i + p_i \Rightarrow C_i + p_{i+1} + \ldots + p_r \leq d_i + p_i \Rightarrow C_i - d_i + p_{i+1} + \ldots + p_r \leq p_i$. Thus, Szwarc's Rule holds only if $p_i \geq (p_{i+1} + \ldots + p_r) + t_i$. $\square$

Finally, we present Property 7 based on sub case (c).

Property 7. *Given $C_i < d_i$, Szwarc's Rule is satisfied as long as: $p_i \geq (p_{i+1} + \ldots + p_r) - s_i$.*
Proof: Since $C_i < d_i$, job i is early and its slack $s_i = d_i - C_i$. For Szwarc's Rule to hold, condition $C_r \leq d_i + p_i$ must be satisfied. And $C_r \leq d_i + p_i \Rightarrow C_i + p_{i+1} + \ldots + p_r \leq d_i + p_i \Rightarrow C_i - d_i + p_{i+1} + \ldots + p_r \leq p_i$. Thus, Szwarc's Rule holds only if $p_i \geq (p_{i+1} + \ldots + p_r) - s_i$. $\square$
It is interesting to note that Property 7 can be said to have a longer reach when compared to Property 6 and Property 5. This means, when job i is early, then it has a greater potential of stretching itself more toward its right in the EDD sequence thereby resulting in more positions being eliminated while still satisfying Szwarc's rule.

CONCLUSIONS

We studied the well-known optimal decomposition algorithm for the tardiness problem. We presented simplified conditions of this algorithm which enables one to visually determine the various decomposition positions of a given job. We then did a thorough review of Szwarc's Elimination Rule and presented several properties under which the Szwarc's rule is satisfied. We also provide mathematical proofs of our properties. We believe that our study will enable more theoretical research for this problem and will eventually enable optimal solutions for very large job sets. Future research should focus on incorporating these theoretical findings to make the optimal algorithm more powerful and faster since the core storage requirements will be less when more positions are eliminated.

REFERENCES

Bouska, M., Sucha, P., Novak, A., & Hanzalek, Z. (2023). Deep learning-driven scheduling algorithm for a single machine problem minimizing the total tardiness. *European Journal of Operational Research, 308*, 990–1006.

Du, J., & Leung, J.Y.-T. (1990). Minimizing total tardiness on one machine is NP-hard. *Mathematics of Operations Research*, *15*, 483–495.

Koulamas, C. (2010). The single-machine total tardiness problem: Review and extensions. *European Journal of Operational Research*, *202*, 1–7.

Lawler, E.L. (1977). A ‘pseudo-polynomial’ algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics*, *1*, 331–342.

Naidu, J. (2003). A note on a well-known dispatching rule to minimize total tardiness. *Omega - The International Journal of Management Science*, *31*, 137–140.

Naidu, J. (2011). A new algorithm for a special structure of the single machine tardiness problem. *AIMS International Journal of Management*, *5*(1), 21–34.

Naidu, J.T., Gupta, J.N.D., & Alidaee, B. (2002). Insights into two solution procedures for the single machine tardiness problem. *Journal of the Operational Research Society*, *53*, 800–806.

Potts, C.N., & Van Wassenhove, L.N. (1982). A decomposition algorithm for the single machine total tardiness problem. *Operations Research Letters*, *1*, 177–181.

Szwarc, W. (1993). Single machine total tardiness problem revisited. In Y. Ijiri (Ed.), *Creative and Innovative Approaches to the Science of Management* (pp. 407–419). Quorum Books.

Szwarc, W., & Mukhopadhyay, S.K. (1996). Decomposition of the single machine total tardiness problem. *Operations Research Letters*, *19*, 243–250.