A NEW HEURISTIC FOR THE SINGLE MACHINE TOTAL WEIGHTED TARDINESS PROBLEM

Pedro Palominos
Industrial Engineering Department, University of Santiago of Chile
Av. Ecuador 3769, Santiago, Chile
pedro.palominos@usach.cl

Patricio Araya
Afiliação
Industrial Engineering Department, University of Santiago of Chile
patricio.araya@usach.cl

Patricio Silva
Afiliação
Industrial Engineering Department, University of Santiago of Chile
Patricio.silva@usach.cl

RESUMO

Este trabalho apresenta uma nova heurística para o problema de minimização da demora total ponderada numa máquina. A proposta heurística é do tipo greedy baseado em intercâmbio de tarefa que diminuí a função objetivo. A implementação da heurística foi desenvolvida em MS Excel. A heurística é comparada com os dados da OR-Library e os resultados obtidos são muito bem, porque na maioria dos casos analisaram as soluções eram iguais as soluções ótimas, com um desvio de percentagem média de 1.1% já que os casos estudados.

PALAVARAS CHAVE: Heurística, Programação, Demora Total Ponderada

Área principal: Combinatorial Optimization

ABSTRACT

This paper presents a new heuristic for the Single Machine Total Weighted Tardiness (SMTWT) problem. The proposed heuristic is of the greedy type based on task exchange that decreases the objective function. The implementation of the heuristic was developed in MS-Excel. The heuristic is compared with the data set of the OR-Library and the results obtained are very good, because in most of the cases analyzed the solutions were equal to optimal solutions, with an average percent deviation of 1.1% for the cases studied.

KEYWORDS: Heuristics, Scheduling, Total Weighted Tardiness

Main area: Combinatorial Optimization

1. Introduction

Scheduling is a form of decision–making that plays an important role in manufacturing as well in the service industry. In a competitive environment, the due date is one of the most important factors that firms have to consider, because the companies may incur contractual penalties for not delivering the product or service on time. Moreover, since companies have more than one customer, they must prioritize according to the importance given by the company, which must be reflected in the priorities to be allocated to the production orders in the workshop.

In considering shop floor scheduling problems, one machine scheduling issue provides the basis from which we can extend to another complex problem. The single-machine total weighted tardiness problem (SMTWT) is strongly NP-hard (Lesntra, 1977) and can be stated as follows: Each of *n* jobs (numbered 1,...,n) is to be processed without interruption on a single machine that can handle no more than one task at a time. Job *j (j = 1,...,n)* becomes available for processing at time zero, requires an uninterrupted positive processing time *t(j)* on the machine, has a positive weight *w(j)*, and has a due date *d(j)* by which it should ideally be finished. For a given processing task order, the earliest completion time *C(j)* and tardiness *T(j) = max{C(j)-d(j),0} of task j (j = 1,...,n)* can be readily computed. The problem is to find a processing order of the jobs with minimum total weighted tardiness, for which the following objective function is used:

$$Min \sum_{j=1}^{n} T_j \cdot w_j \qquad (1)$$

In the literature, solving techniques can be classified into three main categories: a) exact solutions methods, b) dispatching rules, and c) heuristic approaches. Exact solutions methods consist mainly in branch and bound algorithms (Shwimer 1972, Potts 1985, Adbul-Razaq 1990) and dynamic programming (Barker 1978, Schrage 1978), however the solutions are computationally inefficient, especially when the number of tasks to be programmed is greater than 50 (Bilge 2007).

In relation to dispatching, a wide variety of rules have been proposed to solve problems quickly (Morton 1984, Potts 1991, Panwalkar 1993, Alidaee 1996). The most popular rules for the SMTWT problem are EDD (Earliest Due Date), WSPT (Weighted Shortest Processing Time), and MCOVERT (Modified Cost Over Time), but they have poor worst-case performance (Tasgetiren 2006).

With respect to heuristics as well as to the use of metaheuristics such as genetic algorithms (GA), tabu search (TS), particle swarm optimization (PSO), ant colony optimization (ACO), and other more refined metaheuristics [Bilge 2007, Tasgetiren 2006, Sen 2003, Ferrolho 2007, Geiger 2010), they offer a compromise between computational expense and solution quality, but their computational implementation is not very easy.

In particular, the present work is focused on the use of greedy algorithms, which yield good solutions in a reduced time and are of easy use and computer implementation. Specifically, the new heuristic proposed was developed in Visual Basic language with support of MS-Excel lists, which causes the program to be easily transportable.

2. A New Greedy Heuristic

The new proposed heuristic consists of two phases: a) Getting an initial solution, and b) application of a greedy algorithm.

2.1 Getting an initial solution

The idea of using an initial solution before applying the greedy algorithm is based on the need to reduce the computational effort and to find a better solution from an existing solution. In order to

obtain an initial solution, some simple priority rule, a combination or a variant of the rules can be used. In this particular case rule EDD (Earliest Due Date) (Jackson 1995) is first applied to all tasks to have an initial sequence, then a variant of the WSPT (Weighted Shortest Processing Time) (Smith 1956) rule is applied to improve the previous solution. The variant of the WSPT rule is based on ordering the tasks according to the tj/wj ratio only for the groups of tasks which present tardiness, leaving untouched all jobs that do not present tardiness. Also, those tasks which are delayed but are programmed between two tasks that are not delayed retain their position. This application is illustrated in Figure 1.

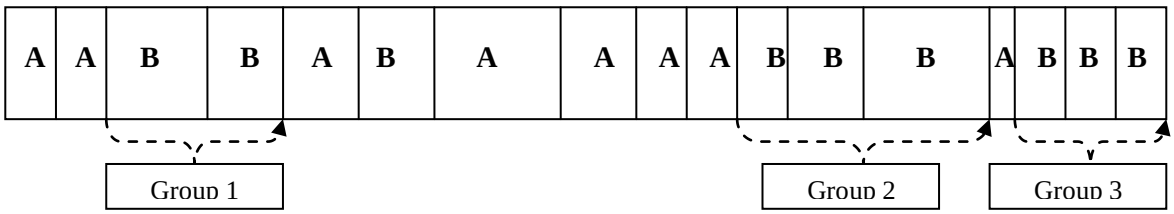

Figure 1 Application of the WSPT variant.

When the EDD rule is applied, those tasks which are not delayed, called A, are generated; tasks with delay are called B. For example, in Figure 1 we apply the WSPT variant rule to the groups of the delayed tasks, in this case to groups 1, 2 and 3. This application aims at considering the importance of the tasks to be scheduled, since clearly the objective function having two tasks with the same tardiness and different importance is not affected in the same way. In our analysis, three types of tasks were defined, that are described in what follows.

a) *Type 1 Tasks.* They include all those tasks whose processing time is low and have a large weight. These tasks have to be programmed as soon as possible since they are of high priority. These kinds of tasks will have a major impact on minimization of the flow. They will be programmed at the beginning of the group.

b) *Type 2 Tasks.* They are those tasks that have long processing times and low weights. These types of tasks will be programmed at the end of the group, because they have little importance and their high processing time will not help to reduce the flow.

c) *Type 3 Tasks.* They are those tasks in which the *tj/wj* ratio produces an equivalent effect between having tasks with low processing time and low importance, versus tasks with high processing time and high importance. In this case, these type 3 tasks were programmed after those of type 1, but before those of type 2, within the group.

2.2 Greedy algorithm

After obtaining the initial solution, the proposed greedy heuristic is applied. That heuristic is based on exchanging those jobs that contribute to the minimization of total weighted tardiness (TWT). The evaluation of the convenience of the possible changes comes from two evaluations called direct effect and intermediate effect, which are detailed in points 2.2.1 and 2.2.2.

The operation of the heuristic is characterized by following a route along the sequence found in Phase I (initial solution), stopping at the time of finding in its way some task that presents tardiness. Soon an evaluation of possible exchanges takes place by means of both effects indicated above, between the task found and some other task that is programmed before it. If there is any exchange that provides some benefit to the objective function, then the algorithm exchanges the tasks and goes on searching for delayed tasks starting from the one which follows the previous modified task. In general terms, the scheme of the greedy algorithm is shown in Figure 2.

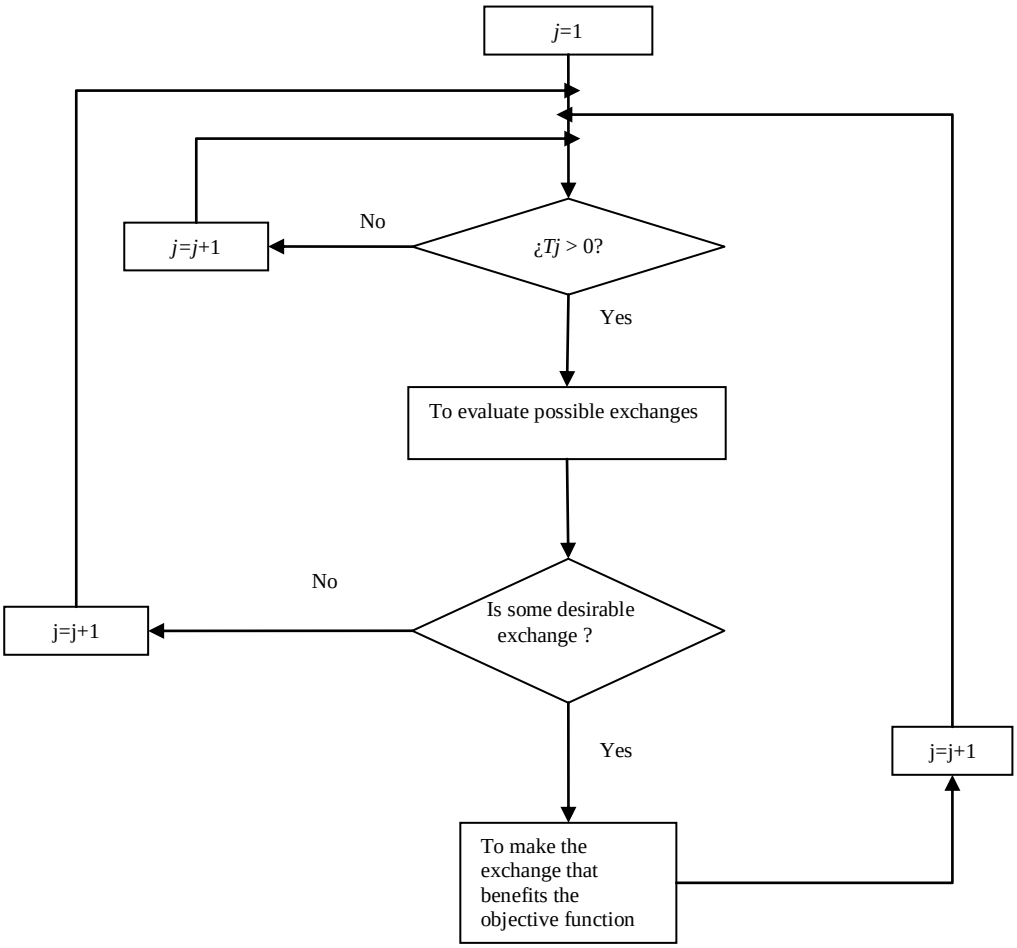

Figure 2 General scheme of the proposed greedy algorithm.

2.2.1 Direct effect

In this case the convenience of a direct exchange between two jobs is evaluated by quantifying the possible direct exchange between the task that was found with tardiness *(Tj' <0)* and some task i that is programmed prior to it.

a) Case 1. The candidate task *i* to be delayed does not present tardiness. In this case three situations occur:
i. – The slack *(Hi)* of the candidate task to be delayed absorbs the tardiness generated by the exchange; when making the task swapping, the slack of the task candidate to be delayed *i* is enough to absorb the tardiness generated by the exchange with the task that is subject to analysis *i'* that does present tardiness. The previous situation is fulfilled if

$$H_i > \sum_{j=i}^{i'} t_j \qquad (2)$$

where *Hi: Max [di-Fi, 0]*.
Because the slack absorbs all possible tardiness generated by the exchange, the effect on the task to be delayed has no effect on the objective function. However, with the task that goes on the following situations can occur:

i.1. - Task *i'* that will be advanced reduces all its tardiness, since:

$$T_i' < \sum_{j:i}^{i'-1} tj \tag{3}$$

contributing to reduce the total weighted tardiness of the objective function in – $Ti' \times Wi'$, where *wi'* is the weight of the analyzed task *i'*.

i.2. - Task *i'* that will be advanced does not achieve the reduction of all its tardiness, because

$$T_i' > \sum_{j:i}^{i'-1} tj \tag{4}$$

Nevertheless, it also generates a contribution to the reduction of the objective function that quantitatively will be

$$-W_i' * \sum_{j:i}^{i'-1} tj \tag{5}$$

where *Wi'* is the weight of analyzed task *i'* and the summation shown will be the sum of the processing times of the intermediate tasks between the candidates to be exchanged.

ii. - The slack *(Hi)* of the task candidate *i* to be delayed is not enough to absorb the tardiness generated by the exchange. When exchanging the task that does not present tardiness with that which presents delay, the slack generated by task *i* is not enough to absorb the tardiness generated by the exchange, and for that reason it will be tardy. The previous situation is fulfilled if

$$H_i < \sum_{j=i}^{i'} tj \tag{6}$$

This exchange worsens the objective function by

$$W_i * (\sum_{j=i}^{i'} tj - H_i) \tag{7}$$

while with task *i'* that went ahead the following can happen:

ii.1.- All its tardiness is taken away, since: $T_i' < \sum_{j=i}^{i'-1} tj$ (3)

Contributing to the objective function with $-W_i' * \sum_{j:i}^{i'-1} tj$ (5)

ii.2.- Task *i'* that goes ahead does not deduct all its tardiness because

$$T_i' > \sum_{j=i}^{i'-1} tj \tag{4}$$

contributing to the objective function with the following amount:

$$-W_i' * \sum_{j:i}^{i'-1} tj \tag{5}$$

b) Case 2: The task candidate to be delayed *i* is with tardiness; Evidently, tardiness of the task that is late increased side, being its new tardiness the following one:

$$T_{inew} = Ti + \sum_{j=i+1}^{i'} tj \quad (8)$$

The objective function gets worse by $Wi * \sum_{j=i+1}^{i'} tj$ (7)

Meanwhile, with the task that goes ahead *i'* the following two situations can occur:

i.- Tardiness *i'* is completely suppressed, because $T_i^{'} < \sum_{j=i}^{i'-1} tj$ (3)

generating a contribution to the objective function in $-W_i^{'} * T_i^{'}$ (9)

ii.- It does not succeed to deduct all the tardiness of *i'* because $T_i^{'} > \sum_{j=i}^{i'-1} tj$ (4)

contributing to the objective function with $-W_i^{'} * \sum_{j:i}^{i'-1} tj$ (5)

After analyzing this first effect, called the direct effect, those exchanges that generate negative values were labeled as candidates, that is to say, they allow minimizing the objective function (total weighted tardiness).

2.2.2 Intermediate effect

The second effect is based on analyzing what happens to the jobs that are in the middle of two exchange candidates. This question originates because these are also affected by the change. In this situation two cases appear, which are described below:

a) Case 1: The processing time of task candidate *i* to be delayed is greater than the processing time of task *i'* that attempts to go ahead (*ti* > *ti'*), and the consequence is that the intermediate task sees beneficiaries at a time Δt = abs [*ti-ti'*].
From the above, a generic analysis of any task *j* that is in the middle between the exchange candidates *i* and *i'*. This analysis is the following:

i. - If the analyzed task is not tardy (*Tj* = 0), its incidence on the objective function will be nil.

ii. - If the analyzed task presents tardiness (*Tj* > 0), two situations can occur:

ii.1. - The difference in time Δt generated is enough to absorb the tardiness presented by the task, and therefore this task does not present tardiness. This will happen as long as $\Delta t > Tj$, and the contribution to the objective function is $-W_j * T_j$

ii.2. - The difference in time Δt generated is not enough to absorb the tardiness presented by the task, because $\Delta t < Tj$, then the new tardiness of task *j* will be $T_{jnew} = T_i - \Delta t$, and its contribution to the objective function will be $-W_j * \Delta t$

b) Case 2: The processing time of the task that is a candidate to be delayed is less than the processing time of the task that tries to go ahead (*Ti* > *Ti'*), and the consequence is that the intermediate tasks are affected by a time Δt = abs [*Ti-Ti'*]. From the above an analysis is made for whatever task *j* that is in the middle between those exchanged. The analysis is the following:

i. – The analyzed task j presents tardiness and therefore it will be increased in a time Δt, presenting a new tardiness value that will be equivalent to $T_{jnew} = T_j + \Delta t$, affecting the objective function by $\Delta t \times w_j$.

ii. - The analyzed task does not present tardiness, so the two following situations can occur:

ii.1. - The generated time difference Δt gets absorbed by the slack of task *j*, therefore tardiness will continue being zero, and this will occur if $H_j > \Delta t$, so there will be no incidence on the objective function.

ii.2. - The generated time difference Δt does not get absorbed by the slack of task *j*, therefore this task will be tardy. This will occur if $H_j < \Delta t$ is detrimental to the objective function by an amount $(\Delta t - H_j) \times w_j$.

It should be noted that the above analysis corresponds to that of an intermediate task, and this has to be made for all the intermediate task s between task s *i* and *i'* that will be exchanged.

Finally, the mixture of both effects analyzed above, the direct and the intermediate effect, constitute the global effect which is satisfied by the sum of these effects for an exchange of tasks *i* and *i'*. Besides, the pair of candidates to be exchanged that cause a greater deduction from the objective function must be indicated, choosing one to carry out the exchange between tasks, that is placing *i'* in the position in which *i* was, and vice versa

2.2.3. Final considerations of the algorithm

In order to avoid stagnation of the heuristic when exchanging tasks *i* and *i'*, it starts to go from the following task to the task *i* that was delayed, with the purpose of preventing the heuristic from going again over already explored spaces in the same iteration.

When the heuristic is iterating, it is always reducing voraciously the objective function. In view of this the stopping criterion will be when after a complete route the value of the objective function is no longer modified, and it is then considered that there is no way for the heuristic to further decrease the objective function. The stopping criterion is given by Figure 3, in which *p* corresponds to the current iteration number.

If $\sum_{i:1}^{n}(TiWi)_p - \sum_{i:1}^{n}(TiWi)_{p-1} = 0$ Then
    Stop
Else
    Next iteration
End If

Figure 3 Stopping criteria in pseudo-code

The general diagram of the heuristic proposal appears in pseudo-code below, where the direct and intermediate effects can be seen, as well as the specified stopping criterion.

2.3. Pseudo-Code of Greedy Heuristic

Definition of variables

A: Vector of tasks with initial ordering. [*a1, a1,....., an*]
B: Vector of tasks programmed at some time of the heuristic [*b1, b1,....., bn*]
n: Number of tasks to be scheduled
Dai: Due dates in the A vector to from task *i*.

Dbi: Due dates in the B vector to from task *i*.
k, i, j, h, l, t: Counters.
Fbi: Flow corresponding to task *i* of vector B.
pos, pointer: Position indicators within vectors A and B
total: impact on objective function
Min: Auxiliary variable that keeps the minimum due dates of a vector.
M: Very great value.

Initial solution

Scheduling by EDD

```
Min = M
k=1
While A ≠ {Ø} Do
  For i=1 to n Step 1
    If Dai < Min Then
      Min = Di
      pos = i
    End If
  Next i
Save task pos in position k on vector B
To eliminate task pos of vector A
k=k+1
n=n-1
Loop
n=0
k=0
i=0
```

Scheduling by WSPT

```
l=1
For i=l to n Step l
  If (Fbi-Dbi) > 0 Then
    For j=i+1 to n
      If (Fbj-Dbj) > 0 Then
       k=k+1
      Else
        iF k≥2
        To order tasks of Vector B from i to i+k according to wspt order
        End if
       End if
      Next j
    End If
   l=j-i
  Next i
```

Direct influence

```
For i=1 to n
  If (Fbi-Dbi) > 0 Then
    For j=i +1 to n
      To exchange task i with task j
        If  (Exchange reduces tardiness) Then
          To mark like candidate
        End if
       Next j
     End If
```

Intermediate effect

For h=i+1 to n
If (task h is candidate) Then
To measure impact in F.O. of the change due to the movement of the intermediate tasks between tasks h and i.
Total = repercussion in F.O. of direct influence and intermediate effect.
End If
Next h
For t=i+1 to n
To look for task that has the greater reduction in the Objective Function.
pointer= task position that reduces more tardiness of the F.O.
Next t
To exchange task i with pointer task
Next i

3. Experimental Results

The computational development of the proposed heuristic was made in Visual BASIC 5.0 using as MS-Excel sheets as calculation sheets. The application was on the data base available in the OR-Library (Beasley 1990). The problems considered were 375, grouped in problems of 40, 50 and 100 tasks, each of them with 125 different problems.
The results were measured by the number of successes with the well-known solution and by the percent deviation between the best well-known solution in the data base and the solution found by the heuristic proposal. The percent deviation is calculated by the following formula:

$$\Delta(\%) = \left(\frac{Z_{cal} - Z_{best}}{Z_{best}}\right) * 100 \qquad (10)$$

Where Z_{cal} = weighted tardiness value calculated by algorithms, Z_{best} = best known weighted tardiness value available in OR-Library. Furthermore, the results obtained by the heuristic are compared with the heuristic General SB Routine by means of Lekin software (Lekin ®, 2006), for the set of 40 and 50 tasks, since for 100 jobs it were not possible to do it because of a limitation of the available version, as well as Backward Forward heuristics (BFH) (Maheswaran 2003).
The results obtained for the evaluated sets appear in Table 1, where the first column indicates the number of tasks of each instance (# of jobs); the second column gives the number of instances of the set (n), the third column shows the number of successes of the heuristic (n_{op}); the fourth column indicates the percentage of deviation average ($\Delta(\%)_{avg}$), the fifth column is the standard deviation of the average error expressed as percentage (σ (%)), and finally the sixth column is the average CPU time.

Table 1 Results of the heuristic

# of Jobs	n	n_{op}	$\Delta(\%)_{avg}$	σ (%)	t_{avg} (seconds)
40	125	58	0.77	1.67	2.77
50	125	44	1.55	3.54	4.06
100	125	33	0.98	2.28	23.02

When comparing the heuristic with the General SB Routine heuristic by means of the Lekin software (Lekin ®, 2006), the results are shown in Table 2. Furthermore, a statistical analysis to prove the significance of the difference of means between the proposed heuristic and the GSBR heuristic allowed us to conclude that the mean values are significantly better with the proposed heuristic, for 40 as well as for 50 jobs, with a confidence level of 95%.

Table 2. Comparative results between the proposed heuristic (HG) and the GSBR heuristic, for sets of 40 and 50 jobs.

Efficiency measurement	40 task		50 task	
	HG	GSBR	HG	GSBR
# of optimum reached	58	47	44	36
% of optimum solutions	46.4	37.6	35.2	28.8
Average (%)	0.7	4.1	1.5	2.2
Largest error (%)	9.4	88.1	23.4	24.6
Smallest error (%)	0.0	0.0	0.0	0,0
Standard deviation (%)	1.6	11.9	3.5	4.1
Deviation of the average (%)	1.0	5.7	2.0	2.4

Finally in Table 3, we compare the percentage deviation of the proposed and BFH heuristic in relation a best known weighted tardiness value available in OR-Library.

Table 3 Results of the heuristic proposal versus BFH heuristic

# of Jobs	$\Delta(\%)_{avg}$ proposal heuristic	$\Delta(\%)_{avg}$ BFH heuristic
40	0.77	0.99
50	1.55	1.51
100	0.98	4.11

4. CONCLUSIONS

The main contribution of the present work was to propose a greedy heuristic based on rules that are generated from an analysis of the candidate tasks to be exchanged in the scheduling. We think that the efficiency of the proposed heuristic is quite good for the set of 40, 50 and 100 tasks of the OR-Library, in percentage deviation, and in CPU time results. Moreover, comparing our heuristic with the GSBR heuristic, it got a greater number of optimum values, better average error, lower standard deviation, smaller average deviation, and a larger amount of better results for the test problems corresponding to sets of 40 and 50 tasks. The same is true when comparing the results with the BFH heuristics.

It should also be pointed out that the implementation of the heuristic was made with Microsoft Visual BASIC© and Microsoft Excel©, which ensures its execution with most computers with Windows© operating systems, which is very convenient for small manufacturing companies. Finally, future work will be oriented in two way directions: first, studying the impact on the final solution of the use of different scheduling rules in the first stage of the heuristic to get the first solution, and second, to generalize the task exchange analysis to other scheduling problems such as parallel machines problems and flow-shops, among others.

5 Acknowledgments

The authors acknowledge with thanks the support of the Universidad de Santiago de Chile under DICYT-USACH Project 060517PB.

6 References

Adbul-Razaq T.S., Potts N.C., and Van Wassenhove L.N, (1990), A survey of algorithms for the single machine total weighted tardiness scheduling problem. Discrete applied Mathematics, vol. 26, pp 235-253.

Alidaee B. and Ramakrishnan, (1996), A computational experiment of COVERT-AU class of rules for single machine tardiness scheduling problem, Computer and Industrial Engineering vol 30, Nº2, pp. 201-209.

Barker K.R. and Schrage L.E, (1978), Finding an optimal sequence by dynamic programming: an extension to precedence-related tasks, Operations Research, vol 26, Nº1, pp. 111-120.

Beasley J.E., (1990), OR-Library: distributing test problems by electronic mail", Journal of the Operational Research Society Vol. 41 (11) pp1069-1072 (http://people.brunel.ac.uk/~mastjjb/jeb/orlib/wtinfo.html, 4, 2012).

Bilge Ü, Kurtulan M., Kırac F., (2007), A tabu search algorithm for the single machine total weighted tardiness problem, European Journal of Operational Research, 176, pp 1423-1435.

Ferrolho A. and Crisóstomo M., (2007), Single machine total weighted tardiness problem with genetic algorithms, Computer Systems and Applications, 2007. AICCSA '07. IEEE/ACS International Conference, pp. 1-8.

Geiger Martin, (2010), On heuristics search for the single machine total weighted tardiness problem some theoretical insights and their empirical verification, European Journal of Operations Research, vol 207, pp. 1235-1243.

Jackson J.R., (1955) "Scheduling a Production Line to Minimize Maximum Tardiness", Research Report 43, Management Science Research Project, University of California, Los Angeles.

Lesntra J. K., Kinnooy Kan A.H. and Brucker P., (1977), Complexity of machine scheduling problems, *Annals of Discrete Mathematics*, vol. 1, pp. 343-362.

Maheswaran R. and Ponnambalam S., (2003), An investigation on single machine total weighted tardiness problem, International J. Adv. Manufacturing Technology, vol. 22 , pp. 243-248.

Morton T.E., Rachamadugu R.V. and Vepsalainen A., (1984), Accurate Myopic heuristics for tardiness scheduling, working paper 36-83-84, GSIA, Carnegie Mellon University, Pittsburgh, P.A.

Panwalkar S.S., Smith M.L., and Koulamas C.P., (1993), A heuristics for the single machine tardiness problems. European Journal of Operations Research, Vol. 70, pp. 304-310.

Potts C.N. and Van Wassenhove L.N., (1985), A branch and bound algorithm for the total weighted tardiness problems, Operations research, vol 33, Nº2, pp 363-377.

Potts C.N. and Van Wassenhove L.N., (1991), Single Machine tardiness sequencing heuristics, s, IIE Transactions , vol 23, Nº4, pp. 346-354.

Schrage L.E. and Barker K.R., (1978), Dynamic programming solution of sequencing problem with precedence constraints, Opertions Research , Vol. 26, Nº3, pp 444-449.

Sen T.,Sulek J.M., Dileepan P., (2003), Static scheduling research to minimize weighted and unweighted tardiness: A state-of-the-art survey. International Journal of Production Economics (83), pp.1-12.

Shwimer J., (1972), On the n-task, one-machine, sequence independent scheduling problem with tardiness penalties: a branch-bound solutions. Management Science, vol. 18, Nº6, ppb301-b313.

Smith W.E., (1956) "Various Optimizers for Single Stage Production", Naval Research Logistics Quarterly, Vol. 3, pp. 59–66.

Tasgetiren M., Liang Y., Sevkli M. and Gencyilmaz G., (2006), Particle swarm optimization and differential evolution for the single machine total weighted tardiness problem, International Journal of Production research, vol. 44, Nº22, pp. 4737-4754.