Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 47 (2015) 247 – 254
Procedia Computer Science

Analysis for Maximal Optimized penalty for the Scheduling of Jobs with Specific Due Date on a Single Machine with Idle Time

D.Jagan[a], A.N. Senthilvel[b], R.Prabhakar[c], S. Uma Maheswari[d]

[a] *PG Scholar, Department of Computer Science and Engineering, Coimbatore Institute of Technology, Coimbatore - 641014, India.*
[b] *Assistant Professor(SG), Department of Computer Science and Engineering, Coimbatore Institute of Technology, Coimbatore - 641014, India.*
[c] *Emeritus Professor, Department of Computer Science and Engineering, Coimbatore Institute of Technology, Coimbatore - 641014, India.*
[d] *Associate Professor, Department of Electronics and Communication Engineering, Coimbatore Institute of Technology, Coimbatore,641014, India.*

Abstract

In the real world the Scheduling of Jobs in industries is provided without any idle time which is very tedious. Practically it becomes difficult when any of the spare part has started to malfunction and has to be changed in the machine then some idle time is needed in order to undergo the change. In this proposed work some amount of idle time is allotted to schedule the jobs in a single machine which includes three stages namely scheduling strategy, inserting idle time and optimizing the net penalty value of all the jobs.

Peer-review under responsibility of organizing committee of the Graph Algorithms, High Performance Implementations and Applications (ICGHIA2014)

Keywords: Single machine scheduling, Early/Tardy scheduling, optimization techniques;

*D .Jagan Email:jssjagan@gmail.com

Peer-review under responsibility of organizing committee of the Graph Algorithms, High Performance Implementations and Applications (ICGHIA2014)
doi:10.1016/j.procs.2015.03.204

1. Introduction

Scheduling problem is a common happening. It persists based on which the choice of order is numbered and should be performed accordingly. Generally, a Scheduling Problem involves: Jobs in manufacturing plants, aircraft waiting for landing lane clearance or a bank customers at in a queue of teller window. The basic unit of Job Shop Process is the operation where one can say operation as an elemental task to be performed, but as far as the Theory of Scheduling is concerned the operation need not be defined and the theory is concerned only with what the operations really are.
Primary attributes of each operation are:

- A symbol identifying the operation with a particular job.
- A symbol identifying the operation with a particular machine.
- A real number representing the processing time of the operation.

Each job has a partial ordering of operations which is comprised of job. The partial ordering between operations is given by a binary relationship known as precedence. If x and y are two operations of the same job, if x wants to get processed first before y, then it is said that x precede y. Then it is denoted as $x>y$. The precedence relationship is transitive if $x>y$ and $y>z$ and implies that $x>z$.

Scheduling in industries involves generally a single machine scheduling problem. In this work a single machine is considered in which n independent jobs have to be scheduled. Each job has attributes such as Job id, Processing time, early penalty, late penalty and due date. Each job Ji has a processing time pi. That job has to be completed before the due date di. If the job Ji completes execution before the due date means then the early penalty αi of the job will be used for calculation and if the job Ji has completed after the due date means then the late penalty βi of the job is used.
The earliness Ei of the job Ji can be calculated as Ei=max(0,di-Ci) and lateness Li can be calculated as Li=max(Ci-di,0). The objective function of this scheduling is to minimize the net cost penalty of the jobs.

$$Net\_penalty = \sum_{i=1}^{n} ((\text{di} - \text{Ci}) * \alpha \text{ i}) + \sum_{\text{i}=1}^{\text{n}} ((\text{Ci} - \text{di}) * \beta \text{i}) \qquad \text{---------} (1)$$

In order optimize the net penalty first the jobs are scheduled using the scheduling strategies and passed into the machine and the penalty value is calculated. Then to further optimize the net penalty the optimization techniques such as Genetic Algorithm, Bee Colony Optimization, Ant Colony Optimization, Branch and Bound and other evolutionary techniques may be used.

So far the work made consideration of the early/tardy scheduling. For ETSP there are many procedures proposed in the recent years. Among the different procedures Shyam Sundar and Alok Singh[2] proposed the procedure based on an Artificial bee colony algorithm which considers the new swarm intelligence approach for scheduling a jobs in a single machine. J.M.S Valente et al[1] proposed a solution for ETSP based on a hybrid genetic algorithm, they compared the results of scheduling jobs on various versions of genetic algorithm. Pei Chan Chang[4]proposed the solution for ETSP based on Branch and bound approach for a single machine, in this approach the Just-In-Time schedule were eliminated and overlapped. J.M.S Valente et al[6] proposed a method for ETSP with no idle time using the lower bounds such as Lagrangean relaxation and multiplier adjustment method. All the above works consider scheduling of jobs in the single machine with no idle time

However, when the scheduling in a industry is taken into consideration then there is some amount of idle time has to be inserted .The idle time may be needed for changing the spare parts of the machine or if the employee is new to operate the machine then he need some time to get acknowledged with the new machine.

2. Proposed work

Many existing works in scheduling of jobs in single machine were considered with no idle time in between each jobs. The reason behind the scenario was authors focused on ETSP. According to the ETSP the machine should be idle only when no job is in ready state, ie., the jobs which are ready to process has to be scheduled without any idle time. The job should to be made continuously available for the machine to process. However when we consider the scheduling of jobs on a single machine in industry, then there may exist some idle time in between the jobs because some amount of time may be needed to change the spars of the machine or the employee may be new to the machine and may lag to work efficiently with the machine.

To meet the objective function, the scheduling process is classified into three major stages. To minimize the penalty, the first stage is scheduled the jobs so that they get executed as per the criteria. Second stage is carried out to insert idle time either in between the jobs or before starting the jobs. Third stage follows any optimization techniques to optimize the net penalty.

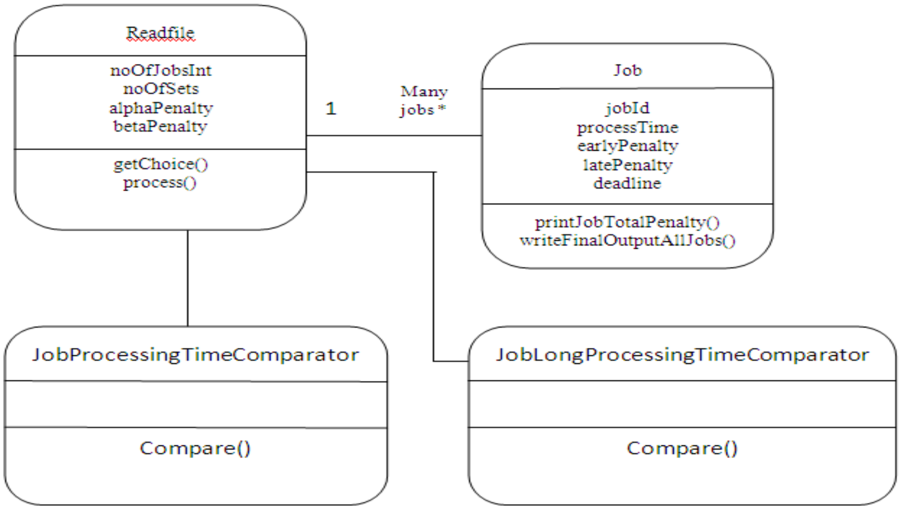


Fig 1: Class Diagram

Fig 1 Illustrates classes such as Readfile, Job, JobProcessingTimeComparator and JobLongProcessingTimeComparator. In the Readfile class the input file is read with the job details and an individual objects for each and every jobs is created based on the job class. The Parameters used in Job class are considered as the major attributes of the job for scheduling.

First Stage:

The First stage is to schedule the jobs for the machine. This can be done either by Considering only the processing time or Considering both the processing time and the penalty values (αi and βi).

Method 1:

The Scheduling Strategy used are First Come First Serve, Longest Processing Time, and Shortest Processing Time. By using these strategies the jobs are scheduled based on their processing time.

Method 2:

The method 2 considers both the processing time and penalty values (αi and βi) for scheduling. The major difference between method 1 and method 2 is the consideration of processing time. For each job to present in the early list the first criteria is $\alpha i > \beta i$ and the second criteria is SPT.

```
D= αi – βi;  //D – difference between early and late penalty.
If(D>0)
{
Move the job into early list;
}
Else
{
Move the job into late list;
}
```

After the jobs get split into two lists such as early list and late list, then on both jobs the list scheduling strategy FCFS, SPT (or) LPT is followed for scheduling.

Penalty Calculation:

For calculating the early/tardy penalty consider the completion time Ci of the job Ji , due date di, early penalty αi and late penalty βi

```
If(Ci<di)
{
Penalty=(di-Ci)* αi;
//this shows the early penalty, here the completion time of the job is less than the due date;
}
Else
{
Penalty=(Ci-di)*βi
//this shows the late penalty,the completion time of the job is greater than the due date;
}
```

Second Stage:

For initializing the idle time we have three methods, that is by inserting the idle time in between the jobs or by inserting some amount of idle time after the jobs in early list that had been executed already or by inserting some amount of idle time before starting the first job in the early list.

Third Stage:

After scheduling the jobs and calculating the net penalty of all the jobs in a set the penalty value needs to be further optimized , for optimizing the penalty either Genetic Algorithm, Bee Colony Optimization, Particle Swarm Optimization, can be used. From the above techniques we need to identify which techniques gives less penalty value for the same set of problem.

3. Performance Analysis

In this section, some of the computational results and comparisons on the performance of several scheduling strategies are discussed. The strategies are tested for set of values ranging 15, 50, 75 and 100 jobs. With which only the first process of the proposed system has been implemented and the remaining process are in the development stage.

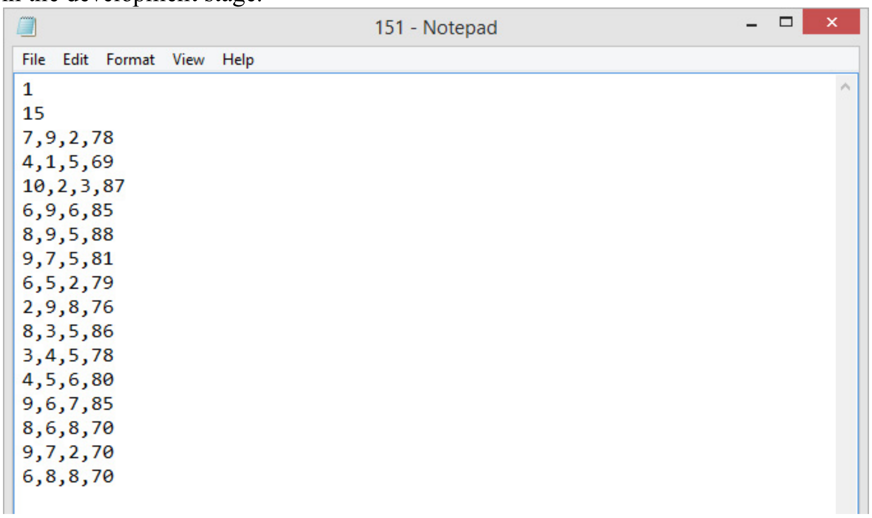
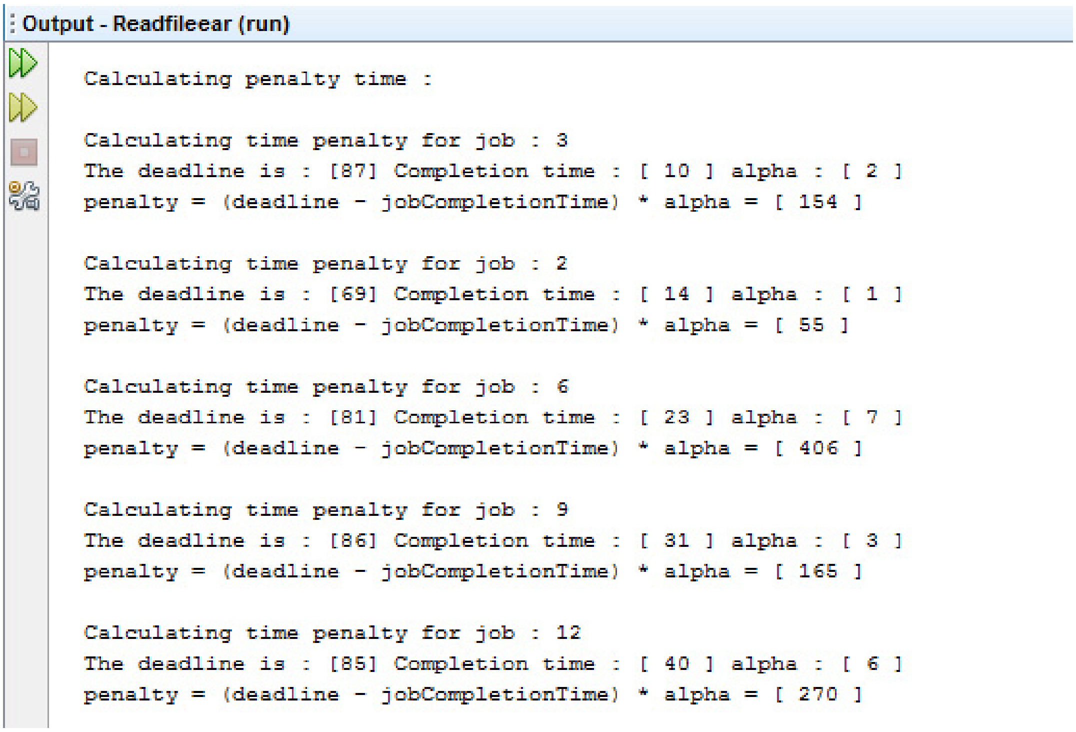


Fig 2: Sample Input File

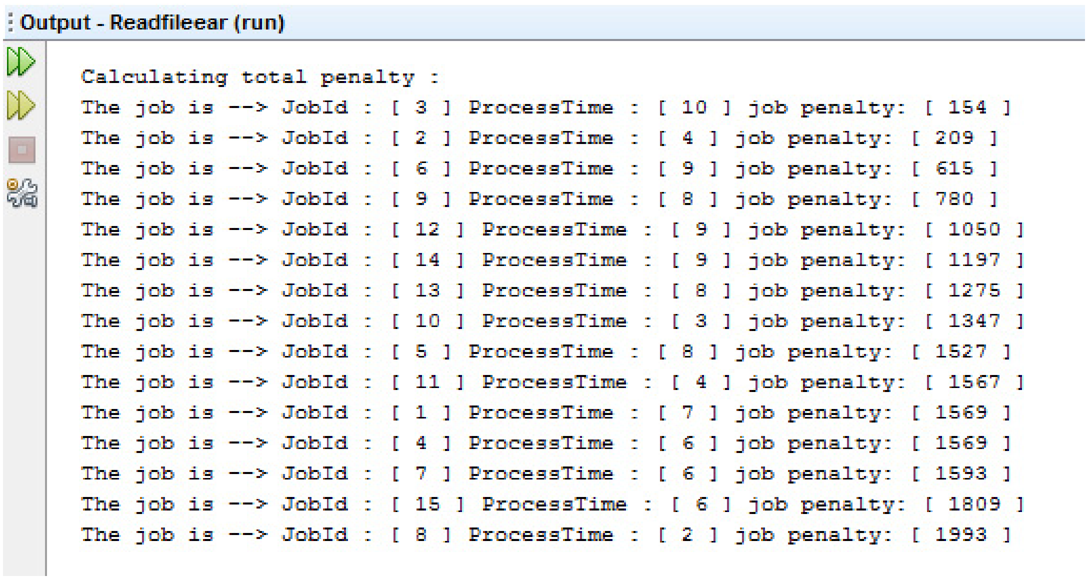
Fig 2 Illustrates a sample file of a set with 15 jobs. In the above file the first line shows the number of sets in the file, second line shows the number of jobs in the first set and the third line shows the jobs attributes such as processing time, early penalty, late penalty and due date.Test instances are borrowed from Jorge M.S. Valente et al



```
Output - Readfileear (run)

Calculating penalty time :

Calculating time penalty for job : 3
The deadline is : [87] Completion time : [ 10 ] alpha : [ 2 ]
penalty = (deadline - jobCompletionTime) * alpha = [ 154 ]

Calculating time penalty for job : 2
The deadline is : [69] Completion time : [ 14 ] alpha : [ 1 ]
penalty = (deadline - jobCompletionTime) * alpha = [ 55 ]

Calculating time penalty for job : 6
The deadline is : [81] Completion time : [ 23 ] alpha : [ 7 ]
penalty = (deadline - jobCompletionTime) * alpha = [ 406 ]

Calculating time penalty for job : 9
The deadline is : [86] Completion time : [ 31 ] alpha : [ 3 ]
penalty = (deadline - jobCompletionTime) * alpha = [ 165 ]

Calculating time penalty for job : 12
The deadline is : [85] Completion time : [ 40 ] alpha : [ 6 ]
penalty = (deadline - jobCompletionTime) * alpha = [ 270 ]
```

Fig 3: Penalty calculation

Fig 3 Illustrates the output for calculating penalty of the jobs based on the scheduling strategies.



```
Output - Readfileear (run)

Calculating total penalty :
The job is --> JobId : [ 3 ] ProcessTime : [ 10 ] job penalty: [ 154 ]
The job is --> JobId : [ 2 ] ProcessTime : [ 4 ] job penalty: [ 209 ]
The job is --> JobId : [ 6 ] ProcessTime : [ 9 ] job penalty: [ 615 ]
The job is --> JobId : [ 9 ] ProcessTime : [ 8 ] job penalty: [ 780 ]
The job is --> JobId : [ 12 ] ProcessTime : [ 9 ] job penalty: [ 1050 ]
The job is --> JobId : [ 14 ] ProcessTime : [ 9 ] job penalty: [ 1197 ]
The job is --> JobId : [ 13 ] ProcessTime : [ 8 ] job penalty: [ 1275 ]
The job is --> JobId : [ 10 ] ProcessTime : [ 3 ] job penalty: [ 1347 ]
The job is --> JobId : [ 5 ] ProcessTime : [ 8 ] job penalty: [ 1527 ]
The job is --> JobId : [ 11 ] ProcessTime : [ 4 ] job penalty: [ 1567 ]
The job is --> JobId : [ 1 ] ProcessTime : [ 7 ] job penalty: [ 1569 ]
The job is --> JobId : [ 4 ] ProcessTime : [ 6 ] job penalty: [ 1569 ]
The job is --> JobId : [ 7 ] ProcessTime : [ 6 ] job penalty: [ 1593 ]
The job is --> JobId : [ 15 ] ProcessTime : [ 6 ] job penalty: [ 1809 ]
The job is --> JobId : [ 8 ] ProcessTime : [ 2 ] job penalty: [ 1993 ]
```

Fig 4: Total Penalty

Fig 4 Illustrates the calculation part for total penalty of jobs along with their job id and processing time of that job.

Table 1: Shortest Processing Time

50 JOBS K	50-02-02	50-02-04	50-04-02	50-04-04	50-06-02	50-06-04	50-08-02	50-08-04	50-10-02	50-10-04
K=1	29143	25388	31168	26288	42324	21460	29546	25156	35705	32122
K=2	44380	25489	31171	29431	40121	23821	36719	30611	38233	22697
K=3	30199	22646	32045	28186	40213	30041	38280	30673	40122	21202
K=4	36786	21967	30379	29364	36158	25864	28156	25089	44715	30314
K=5	29923	25518	37265	21522	37152	23867	32270	35608	45301	29872
K=6	37688	27934	34148	27753	37606	24620	34774	29281	33988	29025
K=7	34759	29236	29052	20033	40367	29433	31159	25920	34664	28115
K=8	37030	26098	34083	27753	41988	28236	43252	31984	38385	27891
K=9	34800	20116	31179	26532	37032	31591	29855	28079	36392	29320
K=10	36034	31401	41797	25097	33419	31430	37815	33042	40704	35869

In Table 1, the value K represents the set number in the file, and the first row represents the file name. The table above shows the penalty for the set of 50 jobs in ten different files. Here the job with shortest processing time gets executed first, and the remaining other jobs are scheduled in the increasing order of processing time.

Table 2: Longest Processing Time First

K	50-02-02	50-02-04	50-04-02	50-04-04	50-06-02	50-06-04	50-08-02	50-08-04	50-10-02	50-10-04
K=1	13980	19883	17926	22327	21727	25375	17305	20738	22650	16874
K=2	27941	18560	16611	21963	23347	22046	24673	25468	28927	23732
K=3	17839	18902	17593	26955	21909	20282	30815	23586	24607	20027
K=4	24399	18368	18157	21623	24539	23196	25364	19645	24196	19590
K=5	18356	18951	19512	20798	22027	22885	22744	27293	34453	26858
K=6	19417	19078	19642	22449	24745	24260	20595	25077	23460	26189
K=7	20765	23773	22889	18988	22065	23219	30362	26214	22991	15465
K=8	24642	21919	19557	17715	23380	17143	32067	24998	23767	29486
K=9	18375	17929	15320	23347	21348	21802	19810	21759	28055	22965
K=10	20399	22335	17247	16064	21670	20862	24070	20467	28686	33351

In Table 2, the value K represents the set number in the file, and the first row represents the file name. This table shows the penalty for the set of 50jobs in ten different files. Here the job with Longest processing time gets executed first and the remaining other jobs are scheduled in the increasing order of processing time.

Table 3: First Come First Serve

K	50-02-02	50-02-04	50-04-02	50-04-04	50-06-02	50-06-04	50-08-02	50-08-04	50-10-02	50-10-04
K=1	21124	20565	23558	24859	30890	21070	21104	17889	34643	19707
K=2	32183	18484	21468	23920	30251	24822	32063	30108	28321	26585
K=3	21067	17763	24453	27825	31163	27025	31174	24426	33476	19481
K=4	30007	17110	23045	23311	25902	21253	24451	21761	33562	23274
K=5	26554	19983	28407	21131	26628	25190	25002	30372	41901	29919
K=6	26782	22691	25369	24894	27601	20815	25033	22704	24131	26237
K=7	27123	22720	23280	17260	30996	23567	28634	23184	22546	18886
K=8	26963	23192	24211	23019	27452	20015	33165	25341	29756	27610
K=9	29367	15907	22713	20379	30879	23286	24199	24257	29152	22865
K=10	28355	27510	31684	16672	26444	27908	29253	25223	30658	31754

In Table 3, the value K represents the set number in the file, and the first row represents the file name. The table above table shows the penalty for the set of 50jobs in ten different files. Here the jobs get executed in which order it arrives.

From the tables above it is clearly shown that the Longest Processing time strategy gives less net penalty value.

4. Discussion

```
if(jobCompletionTime < deadline)
{
    int alpha = job.earlyPenalty;
    int penalty = (deadline - jobCompletionTime) * alpha;
    System.out.println("The deadline is : [" + deadline + "] " +
            "Completion time : [ " + jobCompletionTime + " ] " +
            "alpha : [ " + alpha + " ]");
     System.out.println("penalty = (deadline - jobCompletionTime) * alpha "
             + "= [ " +penalty+ " ]");
     job.penalty = penalty;
     job.isEarlyPenalty=true;
}
else
{
    int beta = job.latePenalty;
    int penalty = (jobCompletionTime - deadline) * beta;
    System.out.println("The deadline is : [" + deadline + "] " +
            "Completion time : [ " + jobCompletionTime + " ] " +
            "beta : [ " + beta + " ]");
     System.out.println("penalty = (jobCompletionTime - deadline) * beta "
             + "= [ " +penalty+ " ]");
     job.penalty = penalty;
     job.isEarlyPenalty=false;
}
```

Fig 5: Algorithm for calculating penalty

Fig 5: Illustrates the algorithm used for calculating penalty of the jobs based on the completion time and deadline. The complexity for the proposed algorithm when one job is given is o(1). When the number of jobs is *n* then the time complexity will be o(n).

5. Conclusion and Future work

As a conclusion that the different scheduling strategies were implemented and from the result was obtained that the Longest Processing time first gives lesser penalty value. The Results were analyzed with the set of 15, 50, 75 and 100 jobs. So far the first process of the proposed work had been implemented. The time complexity for the proposed algorithm is o(n). The penalty value obtained from the work needs to be further optimized using optimization techniques such as Genetic Algorithm, Bee Colony Optimization and Particle Swarm Optimization. From these three techniques the technique which gives a maximized optimized penalty is taken into consideration.

References

[1]. Jorge M.S. Valente, Jose Fernando Goncalves and Rui A.F.S. Alves," A Hybrid Genetic Algorithm for the Early/Tardy Scheduling Problem", Asia Pacific Journal of Operational Research, vol.23, No.3(2006), 393-405.

[2]. Shyam Sundar and A. Singh, "A Swarm Intelligence Approach to the Early/Tardy Scheduling Problem", Swarm and Evolutionary Computation 4(2012),25-32.

[3]. Richard W.Conway, "Theory of Scheduling", 1967.

[4]. Pei Chann Chang, "A Branch and Bound Approach for Single Machine Scheduling with Earliness and Tardiness Penalties", Computers and Mathematics with Applications 27(1999), 133-144.

[5]. James C.Bean, " Genetic Algorithms and Random Keys for Sequencing and Optimization", ORSA Journal on Computing, vol.6, No.2,Spring 1994,154- 160.

[6]. Jorge M.S. Valente and Rui A.F.S. Alves, "Improved Lower Bounds for the Early/Tardy Scheduling Problem with No Idle Time", April 2003.