International Journal of Mathematical Archive-5(7), 2014, 79-88
Available online through www.ijma.info ISSN 2229 – 5046

EXACT ALGORITHM FOR MINIMIZING THE SUM OF TOTAL TARDINESS AND TOTAL LATE WORK PROBLEM

Tariq S. Abdul-Razaq
Mathematics Department, College of Science, Al-Mustansiriyah University, Iraq.

Ali A. Al-Maliky*
Mathematics Department College of Science, Al-Mustansiriyah University, Iraq.

(Received On: 05-07-14; Revised & Accepted On: 23-07-14)

ABSTRACT

This paper presents a branch and bound (BAB) algorithm for minimizing the sum of total tardiness and total late work within the single machine problem. Late work for job i is the amount of processing performed on i after its due date d_i. Branch and bound (BAB) is proposed. This BAB proposes two lower bounds one is based on the decomposition property of the bi-criteria problem the other one based on relaxation of objective .and two dominance rules with special cases. Based on results of computational experiments, conclusions are formulated on the efficiency of the BAB algorithm.

Keywords: *Total tardiness, total Late work, branch and bound algorithm, bi-criteria scheduling.*

1. INTRODUCTION

The late work criterion estimates the quality of a solution on the basis of the duration of late parts of particular jobs. Late work combines the features of two parameters: tardiness and the number of tardy jobs. Formally speaking, in the non-preemptive case the late work parameter (V_j) for job j in a given schedule is defined as

$Vj=\min\{\max\{0,Cj-dj\},pj\}=\min\{Tj, pj\}$ or, in a more extensive way, as

$$V_j = \begin{cases} 0 & Cj \le dj \quad j=1,2,\ldots,n \\ Tj & dj < Cj < dj+pj \quad j=1,2,\ldots,n \\ pj & Cj \ge dj+pj \quad j=1,2,\ldots,n \end{cases}$$

The parameter Vj was first introduced by Blazewicz [5], who called it "information loss", referring to a possible application of the performance measures based on it. The phrase "late work" was proposed by Potts and Van Wassenhove [9].

Applications of the late work minimization problems arise in control systems ([5], [9]), where the accuracy of control procedures depends on the amount of information provided as their input. Leung [7] pointed out another application of late work scheduling in computerized control systems, where data are collected and processed periodically, for late worksee{[1],[2],[3]}.

The tardiness T_i and late work V_i appears to be very important in production planning for both customers and managers. Suppose the customers' orders as job to be executed, then minimizing total cost is equivalent to minimize total tardiness and total parts of orders which are not executed on time:

Interesting applications of the late work criteria arise in agriculture, where performance measures based on due-dates are especially useful [4]. Late work criteria can be applied in any situation where a perishable commodity is involved [9].

{In this paper the bicriteria on single machine scheduling that deal with the sum of total tardiness ($\sum T_i$) and total late work ($\sum V_i$)}.

*Corresponding author: Ali A. Al-Maliky**

The organization of this paper is as follows. Section 2 presents the problem formulation. Section 3 provides special cases section 4 incorporating solution techniques to calculate upper and lower bounds of the multicriteria value. Section 5 presented dominance rules section 6 summarizes results of computational experiments and it is followed by a conclusions is given in section 7.

2. FORMULATION OF THE 1// ($\sum T_I + \sum V_I$) PROBLEM

Our scheduling problems can be described as follows: We are given a set of jobs N={1,...,n} which are to be processed on single – machine, and available for processing at a time zero ,no precedence relationship exists between jobs and preemption is not allowed. Each job requires an integer processing time p_j on the machine, and ideally should be completed at its due date d_j if a schedule is given σ= (1,…,n) then a completion time $C_i = \sum_{j=1}^{i} P_j$ for each job i and consequently tardiness T_i=mac{C_i-d_i,0} and late work V_i= min{T_i , P_i} is calculated .The object is to find a processing order of the jobs on single machine to minimize the multi criteria ($\sum T_i + \sum V_i$). This problem can be stated more precisely as follows:

Given a schedule (1,..., n) then we can compute(total cost) total tardiness $\sum T_i$ and total late work $\sum V_i$.

This problem denote by (P_1) can be written as follows:

$$\left.\begin{aligned} &M= \min_{\sigma \in S}\{\textstyle\sum T_i(\sigma) + \sum V_i(\sigma)\} \\ &\text{s.t.} \\ &T_{\sigma(i)} \geq 0 && i=1\ldots n \\ &T_{\sigma(i)} \geq C_{\sigma(i)} - d_{\sigma(i)} && i=1\ldots n \\ &V_{\sigma(i)} = Min\{T_{\sigma(i)}, P_{\sigma(i)}\} && i=1\ldots n \end{aligned}\right\} \quad (p_1)$$

where σ(i) denoted the position of job i in the ordering σ and (S) denotes the set of all enumerated schedules if such schedule exists.

3. SPECIAL CASES FOR THE PROBLEM (P_1)

For the problem (p_1) we will state some special cases as follows:

Case- (1): For 1// ($\sum T_i + \sum V_i$) problem (p_1) if d_i=d $\forall$ i=1,…,n then (SPT) rule is optimal.

Proof : First if d_i=d $\forall$ i, and order the jobs in SPT rule, then the jobs become in EDD and SPT order, hence we have minimum $\sum T_i$ [8].

Second if d_i=d $\forall$ i, then any order of the jobs gives minimum $\sum V_i$ [9].

Hence the SPT rule is optimal for problem (p_1).

Case - (2): If the EDD schedule gives $T_{max}(EDD)= \sum V_i(EDD)$ and $T_{max}(EDD) =\sum T_i(EDD)$ then this schedule is an optimal for the 1// $\sum T_i + \sum V_i$) problem(p_1).

Proof: It is well known that $T_{max}(EDD)$ is a lower bound for $\sum V_i$, i.e., $T_{max}(EDD) \leq \sum V_i(EDD)$. Hence if $T_{max}(EDD)= \sum V_i(EDD)$ **,**this means that $\sum V_i(EDD)$ is minmum for $\sum V_i$. Also it is well known that

$T_{max}(EDD) \leq \sum T_i(EDD)$ .Hence if $T_{max}(EDD) =\sum T_i(EDD)$**,** this means that $\sum T_i(EDD)$ is minimum for $\sum T_i$.

Consequently the EDD schedule is an optimal for the
1// $\sum T_i + \sum V_i$ problem.

Case - (3): If $T_{max}(EDD) = 0$ then there exists optimal solution for problem (P_1)

Proof: *It is clear*.

4. THE SOLUTION TECHNIQUES FOR THE PROBLEM (P_1)

It not easy to find optimal solution for the problem (P_1) since both problems $1//\sum_{i=1}^{n}T_i$ and $1//\sum_{i=1}^{n}V_i$ are NP_hard [9]. Hence we solved this problem by using BAB method to get an optimal solution.

4.1. Heuristics to Calculate Upper Bound (UB) for (P_1)

For the problem (P_1) we proposed heuristic method The heuristic H_1 with value UB_1 is simply obtained as follows:

Step-(0): Let $N=\{1,...,n\}$, $K=1$, $t=C_0=0$ and set $\sigma = (\varphi)$.

Step-(1): Calculate $X_i, \forall i \in N$ as follows:
$X_i = Max\ \{p_i, d_i - c_{k-1}\}$ ($c_{k-1}$is completion time at position k-1)

Step-(2): Find a job $j^* \in N$, such that
$X_{j^*} = Min\{X_j\}$ then assign job j^* in position K of $\sigma = (\sigma, \sigma(K))$.

Step-(3): Set $t= t+p_{j^*}$, $N =N-\{j^*\}$, $K=K+1$, if $K< n$ go to step (2), otherwise go to step (4).

Step-(4): Compute $UB_1= (\sum T_i + \sum V_i)(\sigma)$ for the resulting sequence jobs $\sigma = (\sigma(1), \ldots, \sigma(n))$

Step-(5): Stop

Example: We illustrate our first heuristic H_1 in five jobs for the problem (P_1) Data for the processing times and due dates are.

$p_i=(2,3,4,5,2)$ $d_i=(3,4,7,6,8)$, $i=(1,\ldots,5)$

Hence we get the sequence (1, 2, 5, 3, 4) and for this sequence we have

$UB_1=\sum_{i=1}^{n}T_i + \sum_{i=1}^{n}V_i=25$

4.2. Derivation of Lower Bound (LB) for the problem(p_1)

Deriving a lower bound for a problem that has a multicriteria function is very difficult since it is not easy to find a sequence that gives the minimum for the two objectives. Since our problem (P_1) is NP-hard we may find a sequence that does well on both criteria and find lower bound (LB). Now we will derive lower bounds for the problem (P_1).

4.2.1, Decomposition of Problem (P_1) to Derive first Lower Bound(LB_1)

In this subsection we decompose the problem (P_1) into two subproblems with a simpler structure.

As shown in the previous section the problem (P_1) has an objective function:

$M= \underset{\sigma \in S}{Min}\{\sum_{i=1}^{n}T_i(\sigma)+ \sum_{i=1}^{n}V_i(\sigma)\}$.

The problem (p_1)can be decomposed into two subproblem (SP_1) and (SP_2)

$M_1=\min \sum_{i=1}^{n} T_{\sigma(i)}$

s.t.

$$\left.\begin{array}{ll} T_{\sigma(i)} \geq C_{\sigma(i)} - d_{\sigma(i)} & i = 1, \ldots, n \\ T_{\sigma(i)} \geq 0 & i = 1, \ldots, n \end{array}\right\} \ldots\ldots (SP1)$$

$$M_1=\min \sum_{i=1}^{n} V_{\sigma(i)}$$

s.t.

$$\left.\begin{aligned} V_{\sigma(i)} &\le C_{\sigma(i)} - d_{\sigma(i)} && i=1,\dots,n \\ V_{\sigma(i)} &\ge 0 && i=1,\dots,n \\ V_{\sigma(i)} &\le P_{\sigma(i)} && i=1,\dots,n \end{aligned}\right\} \dots\dots.(SP2)$$

Theorem 4.1: [3] $M_1+M_2 \le M$ where M_1&M_2 and M are the minimum objective function values of (SP1),(SP2), and (P1) respectively.

The first lower bound(LB_1) is based on decomposing (P_1) into two subproblem (SP_1) and (SP_2) as above, then calculate a lower bound for (SP_1) and calculate a lower bound for (SP_2) then applying theorem (4.1) to get a lower bound LB_1 for our problem (P_1) as follows:

For the subproblem (SP_1) we calculate a lower bound by sequencing the n jobs in EDD order (sequencing the n jobs in non decreasing order of d_j) to find the maximum tardiness T_{max} (EDD) $\le \sum_{i}^{n} T_i$ (opt)[8]. For the subproblem (SP_2) we calculate a lower bound by the same technique to find maximum tardiness T_{max} (EDD) $\le \sum_{i}^{n} V_i$ [9], then we apply the lower bound theorem(4.1) to get initial lower LB = T_{max} (EDD)+T_{max} (EDD) $\le$ M_1+M_2

Hence ILB = $2T_{max}$ (EDD)

Let σ be a partial sequence for K jobs have been assigned to the first K positions the lower bound LB_1 is given by

$LB_1(\sigma)$=Exact cost of (σ)+cost of (*S*),
where *S* is the set of unsequence jobs (n-k). for each job j in σ its actual tardiness and late work is determined as:

T_j= max{c_j-$d_{j,0}$}and V_j=min{T_j, P_j},j=1,..,k . for unscheduled jobs (j $\in S$, j=k+1,…,n) sequencing in EDD, order to calculate T_{max} (EDD), and using theorem (4.1) to get

$LB_1(\sigma)$=Exact cost of (σ)+ $2T_{max}$ (EDD)

4.2.2. Deriving a second lower bound (LB_2)

To construct the second lower bound (LB_2). We use here the fowllowing results:

1. An optimal schedule for problem1/d_i=d/ $\sum_{i}^{n} T_i$ with equal due date can be obtained by SPT rule. [8]

2. An optimal schedule for problem1/d_i=d/ $\sum_{i}^{n} V_i$ with equal due date can be obtained by any schedule with value of $\sum_{i}^{n} V_i = C_{max}$-d.[9]

Let σ be a partial sequence for K jobs have been assigned to the first K positions.

Jobs:	j_1	j_2	…….	j_k	?	……..	j_t	?

C_j: 0 c_{j1} c_{j2} … c_{jk} complation time

LB_2 (σ)=Exact cost of (σ)+cost of (*s*), where σ the partial sequence for k jobs and *s* is the set of unsequence jobs .

For the jobs of σ, their actual tardiness and late work is determined as
T_{j_h}=max{ C_{j_h} - d_{j_h},0} and V_{j_h} =min{ T_{j_h} , P_{j_h} }, h=1,……,k

For any unsecheduled job j_g its tardiness T_{j_g} cannot be less than max{C_{j_g} -d,0}, where d is the maximum due date among the unscheduled jobs:

$T_{j_g} \geq \max\{C_{j_g} - d_{j_h}, 0\}$, g=k+1,……..,n

Also its late work V_{j_g} cannot less than max{C_{max}-d,0}.

$V_{j_g} \geq \max\{C_{max} - d\}$, g=k+1,……,n

Now consider the relaxed problem where the unscheduled jobs have common due date d,

$$d=\max\{dj_{k+1}, dj_{k+2}, \ldots, dj_n\}$$

The minimum total tardiness and total late work with respect to common due date d can be found by sequencing the jobs in SPT order .In this case the SPT order is optimal for both $\sum_1^n T_i$ and $\sum_1^n V_i$. Thus completion C_j of the unscheduled jobs with their tardiness T_{j_L} and late work V_{j_L} can be applying the SPT rule.

$LB_2(\sigma)=[T_{j_1}+V_{j_1}+\ldots+T_{j_k}+V_{j_k}]+[T_{j_{k+1}}+V_{j_{k+1}}+\ldots+T_{j_n}+V_{j_n}]$

$LB_2(\sigma)$=Exact cost of (σ)+cost of (S)

Hence sequencing the unscheduled jobs in the SPT order and replacing their original due date d_{j_l} by a large artificial due date d is done temporarily in order to determine (LB_2) the lower bound for the current partial schedule σ.

5. DOMINANCE RULES (DR)

If it can be shown that an optimal solution can always be generated without branching from a particular node of the search tree, then that node is dominated and can be eliminated. The main goal of (DR) usually specify whether a node can be eliminated before its (LB) is calculated. It is clear that (DR) are particularly useful when a node can be eliminated which has a (LB) that is less than the optimal solution [6]. The first result for (DR) is given by next.

Lemma 5.1: If $d_j \geq t$, where $t=\sum_1^n P_i$ then there exists an optimal sequence in which job j is sequenced last.

The second result is a consequence of dynamic programing (DP) .If the final two jobs of a partial sequence can be interchange without increasing the time at which the machine become available to process the next unsequence job, then this partial sequence is dominated.

Let σ be an initial partial sequence of jobs, let s be the set of jobs not sequenced in σ and let C(σ) denoted the completion time of the last job of σ. Also assumed that σ= σ_1i, whenever σ is not empty. The next of our dominance rules is based on (DP)

Lemma 5.2: For job $j \in s$, if we have two initial partial sequence of jobs $\sigma_1$ji and $\sigma_1$ij such that $C(\sigma_1 ji) \leq C(\sigma_1 ij)$ then $\sigma_1$ij is dominated.

Notes:
1. If in lemma (5.2) $C(\sigma_1 ji)=C(\sigma_1 ij)$ then either $\sigma_1$ji or $\sigma_1$ij (but not both) is discarded.
2. For all nodes that remain after we apply the (DR), we can use the procedure described in section (2.4) to comput (LB)**.**

6. COMPUTATIONAL EXPERIENCE WITH BRANCH AND BOUND (BAB) ALGORITHM

An intensive work of numerical experimentations has been performed. We first present below how instances (test problems) can be randomly generated.

6.1. Test Problems

Test problems were generated as follows: for each job j, an integer processing time p_j generated from the uniform distribution [1, 10]. Also, for each job j, an integer due date d_j is generated from the uniform distribution

[p(1-TF-RDD/2) , p(1-TF+RDD/2)], where p= Σp_i, i=1,…,n, depending on the relative range of due date(RDD) and on the average tardiness factor (TF). For both parameters, the values 0.2, 0.4, 0.6, 0.8, 1.0 are considered. For each selected value of n, two problems were generated for each of the five values of parameters producing 10 problems for each value of n.

6.2 Computational Experience with the Lower and Upper Bounds of BAB Algorithm

The BAB algorithm was tested by coding it in Matlab 7.9.0 (R2008b) and implemented on Intel I Core I i3 CPU M380 @ 2.53 GHZ, with RAM 4.00 GB personal computer.

In the branch and bound algorithm, we proposed two lower bounds at root node of search tree we calculated LB_1 and LB_2 and set

ILB=Max{ LB_1, LB_2 } as an initial lower bound.Obviously it is possible to apply the two lower bounds (LB_1,LB_2) at each node of the search tree of the BAB algorithm.Since in either case, the computational requirement for both lower bounds are comparable , the LB_2 method computationally much faster and therefore we adopt this approach at each node.

In table (3.1), we give the comparative of computational results of BAB algorithm for the problem (P_1). We list 5 problems for each value of n$\in${4,5,6,7,8,9},and also 10 problems of n $\in${10,14}. The optimal value was computed, upper bound (UB), initial lower bound (ILB), the number of generated nodes (Nodes), the computational time (Time), and the number of unsolved problems (Status).

We determined a condition for stopping the BAB algorithm and consider that the problem is unsolved (state is 1), that the BAB algorithm is stopped after a fixed period of time, here after 1800 second (i.e. after 30 minutes).This means a problem remained unsolved within the time limit of 30 minutes , computation was abandoned for that problem .

It is well known that the number of jobs (size of the problem) are likely to affect the efficiency of BAB algorithm.

Table (1): Shows the performance of, initial lower bound, upper bound, number of nodes and computational time in seconds of BAB algorithm without special cases and dominance rules. n$\in${4,5,…,9,10,14}.

n	EX	CEM	Optimal	UB	ILB	Nodes	Time	Status
4	1	10	10	10	10	16	0.1	0
	2	21	21	21	12	14	0.01	0
	3	49	49	49	39	12	0.007	0
	4	42	42	42	29	11	0.007	0
	5	39	39	39	35	15	0.009	0
5	1	34	34	34	25	34	0.1	0
	2	74	74	74	71	23	0.02	0
	3	51	51	51	18	25	0.02	0
	4	23	23	26	6	43	0.03	0
	5	49	49	51	35	38	0.02	0
6	1	43	43	43	34	39	0.1	0
	2	50	50	50	39	42	0.05	0
	3	60	60	61	48	75	0.07	0
	4	99	99	102	44	101	0.1	0
	5	122	122	122	107	34	0.04	0
7	1	67	67	69	60	74	0.24	0
	2	82	82	84	66	156	0.25	0
	3	114	114	123	96	422	0.66	0
	4	121	121	124	109	222	0.23	0
	5	101	101	101	81	102	0.14	0
8	1	121	121	129	108	298	0.58	0
	2	129	129	138	103	680	1.4	0
	3	168	168	175	144	1177	3.12	0
	4	72	72	73	57	280	0.5	0
	5	114	114	124	103	806	1.5	0
9	1	162	162	172	147	1927	6.67	0
	2	134	134	144	109	5311	42.37	0
	3	190	190	190	174	297	0.7	0
	4	232	232	232	215	235	0.502	0
	5	180	180	180	145	907	2.67	0

n	EX	Optimal	UB	ILB	Nodes	Time	Status
10	1	225	233	172	96766	9.1	0
	2	200	210	163	58779	5.4	0
	3	240	244	193	27551	2.6	0
	4	186	187	168	6792	0.6	0
	5	277	277	252	2742	0.2	0
	6	291	291	265	8455	0.7	0
	7	254	260	225	14667	1.3	0
	8	261	261	180	187267	18.05	0
	9	239	239	217	7025	0.6	0
	10	200	211	153	181579	17.6	0
14	1	403	404	379	196138	19.8	0
	2	324	339	286	17372435	1800.001	1
	3	459	459	414	4321823	488.1	0
	4	549	556	493	5086123	587.4	0
	5	361	361	291	15770121	1800.0009	1
	6	482	486	441	402602	41.3	0
	7	478	478	403	3503851	362.2	0
	8	499	508	464	491464	50.7	0
	9	463	477	432	5433607	604.9	0
	10	509	539	440	1976177	219.1	0

Table (2): Shows The performance of initial lower bound, upper bound, number of nodes and computational time in seconds of BAB algorithm with special cases and dominance rules(DR) $n \in \{4,5,\ldots,9,10,14,20\}$.

n	EX	CEM	Optimal	UB	ILB	Nodes	Time	Status
4	1	10	10	10	10	9	0.1	0
	2	21	21	21	12	10	0.008	0
	3	49	49	49	39	7	0.003	0
	4	42	42	42	29	7	0.003	0
	5	39	39	39	35	6	0.003	0
5	1	34	34	34	25	24	0.1	0
	2	74	74	74	71	15	0.01	0
	3	51	51	51	18	16	0.01	0
	4	23	23	26	6	15	0.01	0
	5	49	49	51	35	10	0.01	0
6	1	43	43	43	34	21	0.1	0
	2	50	50	50	39	22	0.02	0
	3	60	60	61	48	28	0.02	0
	4	99	99	102	44	35	0.02	0
	5	122	122	122	107	14	0.007	0
7	1	67	67	69	60	26	0.1	0
	2	82	82	84	66	48	0.03	0
	3	114	114	123	96	73	0.05	0
	4	121	121	124	109	39	0.02	0
	5	101	101	101	81	22	0.01	0
8	1	121	121	129	108	43	0.13	0
	2	129	129	138	103	161	0.1	0
	3	168	168	175	144	250	0.2	0
	4	72	72	73	57	59	0.04	0
	5	114	114	124	103	157	0.124	0
9	1	162	162	172	147	110	0.4	0
	2	134	134	144	109	352	0.3	0
	3	190	190	190	174	61	0.05	0
	4	232	232	232	215	79	0.1	0
	5	180	180	180	145	91	0.07	0

n	EX	Optimal	UB	ILB	Nodes	Time	Status
10	1	225	233	172	74	0.22	0
	2	200	210	163	187	0.15	0
	3	240	244	193	354	0.46	0
	4	186	187	168	177	0.16	0
	5	277	277	252	82	0.068	0
	6	291	291	265	109	0.09	0
	7	254	260	225	591	0.76	0
	8	261	261	180	1443	4.67	0
	9	239	239	217	43	0.032	0
	10	200	211	153	627	0.735	0
14	1	403	404	379	663	0.95	0
	2	324	339	286	6549	39.4	0
	3	459	459	414	1303	2.3	0
	4	549	556	493	3222	8.7	0
	5	361	361	291	917	1.47	0
	6	482	486	441	1199	2.154	0
	7	478	478	403	444	0.49	0
	8	499	508	464	1180	1.69	0
	9	463	477	432	4173	12.8	0
	10	509	539	440	7981	44.8	0
20	1	780	786	722	19568	382.7	0
	2	1240	1252	1143	12698	123.05	0
	3	940	984	868	36989	864.4112	0
	4	999	1007	915	11883	869.663	0
	5	539	540	517	12552	865.37	0
	6	893	893	768	3370	113.79	0
	7	940	944	868	6076	124.43	0
	8	1248	1261	1143	5670	123.75	0
	9	940	948	868	36989	393.56	0
	10	860	876	774	12654	879.7	0

In table (1) and (2) we have:

EX=The number of the test problem.

CEM=Complete enumeration method

Optimal=The optimal value obtained by BAB method.

UB=Upper bound

ILB=Initial lower bound

Nodes = The number of generated nodes

Time=Computational time in seconds

States = $\begin{cases} 0 & \text{if the example is solved} \\ 1 & \text{o.w.} \end{cases}$

From table (1) we observe that the lower bound (LB), even though it is quickly computed, such a weak lower bound is clearly unable to effectively restrict the search in a branch and bound algorithm. And we see the effect of dominance rules in table (2) on the results,especially for the unsolved problems and computational time. .
The following table summarize table (1)

Table - 3: Summary of the table (1) of BAB algorithm.

n	Av. nodes	Av. time	Unsolved problem
4	13.6	0,0234	0
5	23.6	0.028	0
6	58.2	0.033	0
7	195.2	0.042	0
8	648.2	0.184	0
9	1735.4	0.19	0
10	59612.3	5.165	0
14	5275974	577.2	2

Table - 4: Summary of the table (2) of BAB algorithm

n	Av. nodes	Av. time	Unsolved problem
4	7.8	0.02 3 4	0
5	16	0.028	0
6	24	0,033	0
7	41.6	0.042	0
8	134	0.118	0
9	148.6	0.184	0
10	368.7	O.735	0
14	2763.1	11.47	0
20	15844.9	474.04	0

Table(3)and (4) shows the average number of nodes, computational time in seconds and the unsolved problems for the 5 problem of each n= 4,5,6,7,8,9 , and 10 problems of each n=10,14.It is clear from table (1)and (2) that whenever n increases, the number of nodes and the computational time increase. Hence, the BAB algorithm can solve the problem $1//\sum_{i=1}^{n}T_i+\sum_{i=1}^{n}V_i$ of size less than or equal to 14 jobs and with special cases and dominance rules can solve 20 jobs with reasonable time.

7. CONCLUSIONS

In this paper a branch and bound (BAB) algorithm is proposed to find an optimal solution for the problem of minimizing a bi-criteria. A computational experiment for the branch and bound (BAB) algorithm on a large set of test problems are given.

The main conclusion to be drawn for our computational results are:

1. That the upper bound (UB1) is more effective.
2. The second lower bound (LB_2) is mor effective than first lower bound (LB_1)
3. The special cases and dominance rules are p help in solving roblem up to 20 job.

An interesting future research topic would involve experimentation with the approximation algorithms for the following bi-criteria problems:

1. $1//Lex(\sum_{i=1}^{n}Ti , \sum_{i=1}^{n}Vi)$.
2. $1//F(\sum_{i=1}^{n}Ti , \sum_{i=1}^{n}Vi)$

REFRENCES

[1].Abbas,D. A., " Multi Objective Local search Algorithms To Solve Scheduling Problems", M.Sc. thesis, Univ. of Al-Mustansiriya h, College of Science, Dept. of Mathematics(2011).

[2] Al ayoubi.M.A., "A Single Machine Scheduling Problem to Minimize the Sum of Total Completion Times and Total Late Works" Mathematic Dept College of science Al- Mustansiriyah University Vol. 23, No 7, 2012.

[3] AL Nuaimy,A.A" Exact algorithm for minimizing the sum of total late work and maximum late work problem" Diyala journal for pure science.

[4] Alaminana M, Escudero LF, Landete M,Monge JF, Rabasa A, Sanchez-Soriano J. WISCHE; A DSS for water irrigation.

[5] Blazewicz J. Scheduling preemptible tasks on parallel processorswith information loss. Technique et Science Informatiques 1984; 3 (6):415-20.

[6]. Hariri ,A.M.A.,Gupta, .JND. "Single-machine scheduling to minimize maximum tardiness with minimum number of tardy jobs" Annals of Operations Research 92,107-123(1999)**.**

[7] Leung JY-T. Minimizing total weighted error for imprecise computation tasks and related problems. In: JY-T Leung. editor. Handbook of scheduling : algorithms.

[8] Michael, L. Pinedo., "Scheduling:Theory, Algorithms, and Systems", handbook. (2008).

[9] Potts CN. Van Wassenhove LN. Single machine scheduling to minimize total late work. Operations Research 1991; 40(3): 586-95.

Source of support: Nil, Conflict of interest: None Declared