SCITECH
RESEARCH ORGANISATION
Volume 13, Issue 1
Published online: January 16, 2018|

Journal of Progressive Research in Mathematics
www.scitecresearch.com/journals

Branch and Bound Method to Solve The Sum of Two Objective Functions

Al- Zuwaini[1] Mohammed Kadhim, Hussein Kamil Taher[2]
[1]Mkzz50@ yahoo.com [2]ssaa.kamal@ yahoo.com
Department of Mathematics, College of Computer Sciences and Mathematics
Thi-Qar University, Thi-Qar, Iraq

Abstract

In this paper, the problem of sequencing a set of n jobs on single machine was considered to minimize multiple objectives function (MOF). The objective is to find the optimal solution (scheduling) for n independent jobs to minimize the objective function consists of a sum of weighted number of early jobs and total weighted of completion time. This problem is strongly NP-hard and to resolve it we derived two lower bounds (LB1, LB2) and heuristic method to get an upper bound which are used in root node of branch and bound tree. Some special cases and dominance rule were proposed and proved. Results of extensive computational tests show that the proposed (BAB) algorithm effective in solving problems with up to (30) jobs in time less than or equal to (30) minutes.

Keywords: Single machine; Scheduling; number of early jobs; completion time.

1. Introduction

We consider the scheduling n jobs on a single machine to minimize the bi-criteria problems. Our objective is to find a schedule that minimize sum of weighted number of early jobs and total weighted of completion time, with condition all jobs are available at time zero. we denoted to this problem by $\sum_{j=1}^{n} h_j G_j + \sum_{j=1}^{n} w_j C_j$ where h_j the penalty of early job j, $h_j \geq 1$ and w_j the weighted number of completion time for job j,$w_j \geq 1$.

Clearly that our problem consist of two sub problems. The first one $1/ / \sum_{j=1}^{n} h_j G_j$ it is NP-hard problem. When h_j=1, $\forall j$ that is $1/ / \sum_{j=1}^{n} G_j$ the number of early jobs is p-Type and it's studied firstly by Lann and Mosheiov [1]. They developed an polynomial-time algorithm to find the minimum number of early and tardy jobs on a single-machine where machine idle time is permitted. Furthermore Lann and Mosheiov[2] solves the problem of the maximum number of on-time jobs on parallel identical machines. Huang RH and Yang CL (2007)[3], solved the $1/ / \sum_{j=1}^{n} G_j$ problem in polynomial time, and successfully developing such an algorithm. Computational performance of this algorithm on problems with various sizes is provided. Baruch Mor and GurMosheiov(2014) they studied the number of early jobs on aproportionate

flowshop[4]. The problem $1//\sum_{j=1}^{n} W_j C_j$ is solved by The shortest weighted processing time (SWPT) rule of Smith[5]. In the case of $W_j = 1$ ($j \in N$) the base(shortest processing time rule)(STP) Give solution to the problem $1//\sum_{j=1}^{n} C_j$. But if conditions add precedence the problem becomes NP-hard, even in the case where it is $p_j = 1$ or W_j=1∀j [6,7]. Branch and bound algorithms have been used to the problem $1/prec/\sum_{j=1}^{n} W_j C_j$ By many researchers remind them Ronnie et al (1975)[8]. Potts (1981) and Potts (1985)[9] gives the best algorithms to resolve the matter, where he was able problem solving up to 100 job.

2. Formulation of the problem

Single machine scheduling models seem to be very important for understanding and modeling multiple machines models. A set N={1,2,…,n} of n independent jobs has to be scheduled on a single machine in order to optimize a given criterion.

This study concerns the one machine scheduling problem with multiple objectives function which is denoted by $(1//\sum_{j=1}^{n}(h_j G_j + w_j C_j))$.

In this problem, preemption is not allowed, no precedence relation among jobs is assumed and all jobs are available at the time zero, (that is r_j=0 $\forall j$). Each job j has positive integer processing time p_j, due date d_j with weighted (importance weight for completion time of job j) w_j and positive number (penalty for early of job j) h_j.

The start time of job j is denoted by t_j, ($t_j \geq 0$) and its completion time by C_j, ($C_j = t_j + p_j$), if job j completed befor it's due date ($C_j < d_j$ then G_j=1) and job j is said to be early otherwise ($C_i \geq d_i$, then G_j=0), and job j is said to be late job or just in time, for each job j can be calculate the slack time $S_j = d_j - p_j$. Our objective is to find a schedule that minimize the sum of penalty number for early jobs, which is given by:

$$h_j G_j = \begin{cases} h_j & if \quad d_j > C_j \\ 0 & if \quad d_j \leq C_j \end{cases}$$

and total weighted of completion time, the objective is to find the schedule $\pi = (\pi(1), \pi(2), \dots, \pi(n))$ of the jobs that minimize the total cost R which is formulated in mathematic form as:

$$\left.\begin{aligned} &\min R = min_{\pi \in \delta}\{\sum_{j=1}^{n} h_j G_j + \sum_{j=1}^{n} w_j C_j\} \\ &\textbf{\textit{subject to}} \\ &C_{\pi(j)} \geq p_{\pi(j)} \qquad j = 1,2,\dots,n \\ &C_{\pi(j)} = C_{\pi(j-1)} + P_{\pi(j)} \; j = 2,3,\dots,n \\ &G_{\pi(j)} \in \{0,1\} \qquad j = 1,2,\dots,n \\ &p_{\pi(j)} > 0, \;\; d_{\pi(j)} > 0, \;\; w_{\pi(j)} > 0, \;\; h_{\pi(j)} > 0 \end{aligned}\right\} \dots\dots\dots\dots\textbf{\textit{(H)}}$$

Where δ the set of all feasible solutions, $\pi(j)$denoted the position of job j in the ordering π.

Theorem (1)

For weights $h_1, h_2, \dots, h_n$ the following measures are equivalent

(i)$\sum_{j=1}^{n} h_j G_j$ (ii)$\sum_{j=1}^{n} h_j U_j$

Proof:

Let C=$\sum_{j=1}^{n} p_j$, consider an instance of the weighted number of tardy jobs $\sum_{j=1}^{n} h_j U_j$ problem. Let $p_j^{'} = p_j, h_j^{'} = h_j$ and $d_j^{'} = C - d_j + p_j$ for j=1, ..., n

Suppose S is an optimal schedule for this instance. Define a schedule $S^{'}$ as follows:

If a job j is the k_{th} job scheduled in S, then $j^{'}$ is the $(n-k+1)_{th}$ job scheduled in $S^{'}$. Clearly we have $C'_j = C - C_j + p_j$, and

$$h_j U_j^{'} = \begin{cases} h_j & C_j^{'} > d_j^{'} \\ 0 & C_j^{'} \le d_j^{'} \end{cases}$$

$$= \begin{cases} h_j & C - C_j + p_j > C - d_j + p_j \\ 0 & C - C_j + p_j \le C - d_j + p_j \end{cases}$$

$$= \begin{cases} h_j & -C_j > -d_j \\ 0 & -C_j \le -d_j \end{cases}$$

$$= \begin{cases} h_j & C_j < d_j \\ 0 & C_j \ge d_j \end{cases}$$

$$= h_j G_j$$

Therefore, the minimum weighted number of early jobs is an equivalent with the minimum weighted number of tardy jobs. Hence as we know that the weighted number of tardy jobs problem on one machine is NP-hard [10], then the weighted number of early jobs must also be NP-hard.■

Algorithm(HY)[3]

The feasible schedule is composed of *Set E* and *Set Q*. *Set E* precedes *Set Q*, so jobs in the *Set E* must be given priority over the jobs in *Set Q*.

Step 1. Let Set E be empty. Place all jobs in Set Q, and Sequence them according to the MST rule.

Step 2. Find the last early job in Set Q, say $J_{[k]}$, in the Current schedule including Set E and Set Q. If all the jobs in Set Q are non-early, go to Step 4.

Step 3. Find the job in the partial sequence $(J_{[k]}, J_{[k+1]}, \dots, J_{[n]})$ with the longest processing time, and move it to Set E, which may be sequenced in any order. Return to Step 2.

Step 4. Generate an optimal schedule by taking Set E preceding Set Q.

3. Problem decomposition

In this section, the problem (H) is decomposed into five sub problems (HP_1), (HP_2), (HP_3), (HP_4) and (HP_5) which are simple structure of the original problem as follow:

i. The $(1/\ /\sum_{j=1}^{n} h_j G_j)$ problem, weighted number of early jobs, it is NP-hard.

$$\left.\begin{array}{l} R_1 = min_{\pi\in\delta}\{\sum_{j=1}^{n} h_{\pi(j)}\, G_{\pi(j)}\} \\ \textbf{\textit{subject to}} \\ C_{\pi(j)} \geq p_{\pi(j)} \qquad j = 1,2,\dots,n \\ C_{\pi(j)} = C_{\pi(j-1)} + p_{\pi(j)} \; j = 2,3,\dots,n \\ G_{\pi(j)} \in \{0,1\} \qquad j = 1,2,\dots,n \\ p_{\pi(j)} > 0, \;\; d_{\pi(j)} > 0, \;\; h_{\pi(j)} > 0 \end{array}\right\} \dots\dots\dots (HP_1)$$

ii. The $(1/\ /\sum_{i=1}^{n} w_i C_i\ )$ problem, total weighted of completion time which it solved by (WSPT)[5].

$$\left.\begin{array}{l} R_2 = min_{\pi\in\delta}\{\sum_{j=1}^{n} w_{\pi(j)} C_{\pi(j)}\} \\ \textbf{\textit{subject to}} \\ C_{\pi(j)} \geq p_{\pi(j)} \qquad j = 1,2,\dots,n \\ C_{\pi(j)} = C_{\pi(j-1)} + p_{\pi(j)} \qquad j = 2,3,\dots,n \\ p_{\pi(j)} > 0, \;\; w_{\pi(j)} > 0 \qquad j = 1,2,\dots,n \end{array}\right\} \dots\dots\dots (HP_2)$$

iii. The $(1/\ /\sum_{j=1}^{n} G_j + \sum_{j=1}^{n} C_j)$problem, sum number of early jobs and completion time, when $h_j = w_j = 1$

$$\left.\begin{array}{l} R_3 = min_{\pi\in\delta}\{\sum_{j=1}^{n} G_j + \sum_{j=1}^{n} C_j\} \\ \textbf{\textit{subject to}} \\ C_{\pi(j)} \geq p_{\pi(j)} \qquad j = 1,2,\dots,n \\ C_{\pi(j)} = C_{\pi(j-1)} + p_{\pi(j)} \qquad j = 2,3,\dots,n \\ G_{\pi(j)} \in \{0,1\} \qquad j = 1,2,\dots,n \\ p_{\pi(j)} > 0, \;\; d_{\pi(j)} > 0 \qquad j = 1,2,\dots,n \end{array}\right\} \dots (HP_3)$$

iv. The $(1/\ /\sum_{j=1}^{n} G_j + \sum_{j=1}^{n} w_j C_j)$ problem, sum number of early jobs and total weighted of completion time, when $h_j, \forall j$

$$\left.\begin{array}{l} R_4 = min_{\pi\in\delta}\{\sum_{j=1}^{n} G_j + \sum_{j=1}^{n} w_j C_j\} \\ \textbf{\textit{subject to}} \\ C_{\pi(j)} \geq p_{\pi(j)} \qquad j = 1,2,\dots,n \\ C_{\pi(j)} = C_{\pi(j-1)} + p_{\pi(j)} \qquad j = 2,3,\dots,n \\ G_{\pi(j)} \in \{0,1\} \qquad j = 1,2,\dots,n \\ p_{\pi(j)} > 0, \;\; w_{\pi(j)} > 0 \qquad j = 1,2,\dots,n \end{array}\right\} \dots (HP_4)$$

v. The $(1/\ /\sum_{j=1}^{n} h_j G_j + \sum_{j=1}^{n} C_j)$ problem sum weighted number of early jobs and total completion time, when $w_j = 1, \forall j$

$$\left.\begin{array}{ll} R_5 = min_{\pi\in\delta}\{\sum_{j=1}^{n} h_j G_j + \sum_{j=1}^{n} C_j\} & \\ \textbf{subject to} & \\ C_{\pi(j)} \geq p_{\pi(j)} & j = 1,2,\dots,n \\ C_{\pi(j)} = C_{\pi(j-1)} + p_{\pi(j)} & j = 2,3,\dots,n \\ G_{\pi(j)} \in \{0,1\} & j = 1,2,\dots,n \\ p_{\pi(j)} > 0,\ \ h_{\pi(j)} > 0 & j = 1,2,\dots,n \end{array}\right\} \dots (HP_5)$$

Theorem (2) [11]

$R_1 + R_2 \leq R$ where R_1, R_2 and R are the minimum objective function values of HP_1, HP_2 and H respectively.

4. Special Cases

A machine programming downsides of sort NP-hard isn't simply resolvable and it's tougher once the target operate is multi objective. victimization some Mathematical programming strategies to seek out best answer for this kind of downside as: dynamic programming and branch and bound methodology. generally special cases for this downside are often resolved. A special case for programming downside suggests that finding best schedule directly while not victimization mathematical programming techniques. A special case, if it exists, depends on satisfying some conditions so as to create the matter simply resolvable. These conditions rely on the target operate additionally because the jobs [12]. during this section, some special cases of downside (H) are given.

Case (1): WSPT rule is optimal solution for $1//\sum_{j=1}^{n} h_j G_j + \sum_{j=1}^{n} w_j C_j$ problem if $C_j \geq d_j$ for each job j in WSPT.

Proof:

Let π be a schedule orderly according to SWPT rule such that $C_j \geq d_j$ for each j $\in$ π, then $\sum_{j=1}^{n} G_j = 0$ and the problem $1//\sum_{j=1}^{n} w_j C_j$ which is optimal by WSPT rule [5], then the schedule π gives an optimal for $1//\sum_{j=1}^{n} h_j G_j + \sum_{j=1}^{n} w_j C_j$ problem.■

Case (2): If πbe a scheduling orderly such that $\frac{P_j}{w_j} \leq \frac{P_{j+1}}{w_{j+1}} \forall j,\ j = 1,2,\dots,n\text{-}1$, j in π and $p_1 = d_1$, $p_j = d_j - d_{j-1}$, j=2,...,n then scheduleπ is optimal solution for $1//\sum_{j=1}^{n} h_j G_j + \sum_{j=1}^{n} w_j C_j$

Proof :

For the conditions $p_1 = d_1$ and $p_j = d_j - d_{j-1}$, j$\in \pi$ we get $C_j = d_j \forall$j$\in \pi$ and $E_j = 0$ then the problem (H) reduced to $1//\sum_{j=1}^{n} w_j C_j$ which is solved by smith rule [5], then the schedule π is optimal for $1//\sum_{j=1}^{n} h_j G_j + \sum_{j=1}^{n} w_j C_j$. ■

Case (3): If $\sum_{j=1}^{n} G_j$ (WSPT)=$\sum_{j=1}^{n} G_j$ (HY) and h_j=1 $\forall j$ then the WSPT rule gives an optimal solution for problem (H).

Proof:

As we know that WSPT gives an optimal solution for $1/\ /\sum_{j=1}^{n} w_j C_j$ [5] and HY algorithm gives optimal solution for $1/\ /\sum_{j=1}^{n} G_j$ [3] then WSPT is optimal solution for $1/\ /\sum_{j=1}^{n} h_j G_j + \sum_{j=1}^{n} w_j C_j$.■

Case (4): If $w_j = p_j$, h_j=1, then HY algorithm gives an optimal solution for problem (H).

Proof:

From $w_j = p_j \forall j$ we sad that any order gives a solution for problem $\sum_{j=1}^{n} w_j C_j$, and the solution of problem H depend upon solution of problem $1//\sum_{j=1}^{n} h_j G_j$ but h_j=1 then HY algorithm is optimal solution for problem (H).■

Case (5): The HY algorithm is optimal solution for $1/p_j = p, w_j = w, h_j = h\ /\ \sum_{j=1}^{n} h_j G_j + \sum_{j=1}^{n} w_j C_j$ problem.

Proof:

Since $p_j = p$ and $w_j = w \forall j$ hence any order gives optimal for $1//\sum_{j=1}^{n} w_j C_j$, but the problem $\sum_{j=1}^{n} h_j G_j$ is solved by HY algorithm (when h_j=h), then HY algorithm is optimal solution for problem H.■

Case (6): If $w_j = h_j = 1\ \forall j$ and $\sum_{j=1}^{n} G_j$ (SPT)=$\sum_{j=1}^{n} G_j$ (HY) then SPT rule is optimal solution for problem (H).

Proof:

Since $\sum_{j=1}^{n} G_j$ (SPT)=$\sum_{j=1}^{n} G_j$ (HY) and SPT gives an optimal solution for $1/\ /\ \sum_{j=1}^{n} C_j$ [5], then SPT gives an optimal solution for the problem $1/\ /\sum_{j=1}^{n} h_j G_j + \sum_{j=1}^{n} w_j C_j$ with given condition.■

Case (7): If h_j=1, d_j=d, p_j/w_j=K $\forall j$ then the LPT(largest processing time) is optimal solution for the problem H.

proof:

since p_j/w_j=K $\forall j$ then any order is optimal solution for the problem $1/\ /\ \sum_{j=1}^{n} w_j C_j$ and the problem (H) is depends on $1/\ /\sum_{j=1}^{n} h_j G_j$. Since h_j=1 and d_j=d then the LPT rule is optimal for$\sum_{j=1}^{n} G_j$ and its optimal solution for the problem (H).■

Case (8): if $\sum_{j=1}^{n} p_j < d$, d=min$\{d_j\}$, $j = 1,2,\dots,n$ then (WSPT) is the optimal solution for the problem $1/\ /\sum_{j=1}^{n} h_j G_j + \sum_{j=1}^{n} w_j C_j$.

proof:

from $\sum_{j=1}^{n} p_j < d$ and d=min$\{d_j\}$ $\forall j$, clearly $C_j < d_j$ for any order, then all the jobs are early for any order and $\sum_{j=1}^{n} h_j G_j = \sum_{j=1}^{n} h_j$ it's constant. So(WSPT) is optimal solution for the problem 1/ /$\sum_{j=1}^{n} h_j G_j + \sum_{j=1}^{n} w_j C_j$.■

5. Branch and Bound (BAB) Method

Our branch and bound algorithm uses forward sequencing branching rule for which nodes at level (L) of the search tree correspond to initial partial sequenced in the first (L) positions.

5.1 Upper bound (UB)

We suggest a heuristic method which is applied at the root node of the branch tree to find an upper bound UB on the minimize value of problem (H).

Algorithm of (UB):-

Step (1):-

Ordered the jobs by WSPT rule to get the sequence $\pi = (\pi(1), \pi(2), \dots, \pi(n))$ let L=0, T=0 and i=1.

Step (2) :-

If $\frac{p_{\pi(i)}}{w_{\pi(i)}} = \frac{p_{\pi(j)}}{w_{\pi(j)}}$, then ordered the jobs i, j according to the MST rule.

Step (3) :-

Compute the completion time for each job j in π, $C_{\pi(j)} = \sum_{i=1}^{j} p_{\pi(i)}$.

Step (4) :-

If $d_{\pi(j)} > C_{\pi(j)}$, set L=L+$h_{\pi(j)}$and $T = T + w_{\pi(j)} C_{\pi(j)}$. Go to step (5); otherwise, set $T = T + w_{\pi(j)} C_{\pi(j)}$.

Step (5):-

Set i=i+1, if i ≤ n go to step 2, otherwise

UB=L+T.

Step (6):- Stop.

6. Derivation of Lower Bound

The lower bound for the problem H is based on decomposed problem H to two sub problems HP_1 and HP_2. Moreover, calculated R_1 and R_2 to be the lower bounds for HP_1 and HP_2respectively as in section (2.3), and applied theorem (2.1) to get a lower bound LB for problem H.

for sub problem HP_1we relax the penalty of early jobs by putting h=min$\{h_j\}$for j=1,...,n then applying (HY) algorithm $R_1 = \sum_{j=1}^{n} G_j h$. in addition, for problem HP_2, $R_2 = \sum_{j=1}^{n} w_j C_j$ (WSPT), hence LB $= R_1 + R_2$, is initial lower bound which is used in root node of search tree.

Below we gives an example to illustrates the procedures of the lower and upper bounds

1. Example: consider the six jobs in table below.

J	1	2	3	4	5	6
p_j	3	5	6	9	11	15
d_j	41	24	50	30	23	42
w_j	9	5	8	6	14	11
h_j	8	10	9	6	13	5
S_j	38	19	44	21	12	27

Solution:

Let A=∅, B=MST, π=∅, set A precedes B Job 3 is the last early job in B then A=(3), B=(5,2,4,6,1)

Job 2 is the last early job in B and job 6 is larger processing time from any job after job 2, then A=(6,3), B=(5,2,4,1)

Clearly is not early job in B, then π=(6,3,5,2,4,1)

$\sum_{j=1}^{n} G_j = 2$, and h = min(h_j)=5, j=1,2,…,6

$LB_1 = \sum_{j=1}^{n} G_j h = 2(5) = 10$

$LB_2 = 1238$, $LB = LB_1 + LB_2 = 10+1238 = 1248$

UB=1273

optimal schedule is(1,3,5,2,6,4)with optimal cost is 1273.

7. Dominance Rule

Because of branching theme, the scale of the search tree is directly joined to the length of this sequence (which represents the amount of nodes). Hence, a preprocessing step is performed so as to get rid of as several positions as attainable. Reducing this sequence is completed by victimization many dominance rules. Dominance rules typically specify whether or not a node may be eliminated before its edge is calculated. Clearly, dominance rules are notably helpful once a node may be eliminated that features an edge that's not up to the optimum solution[12]. a number of dominance rules are valid for decrease of the sum of weighted number of early jobs and total weighted completion time. As within the a preprocessing step, similar dominance rules also are used at intervals the branch and bound procedure to chop nodes that's dominated by others. These enhancements result in terribly massive decrease within the variety of nodes to get the best answer.

Below state dominance rule to decrease the number of nodes in search tree as well as decreasing the time.

Theorem(7.1): Let δ_k be a partial sequence $k \subset N$ for $i, j \in \bar{k} = N - k$ and let C be a completion time of last job in δ_k, if $p_i \leq p_j$, $w_i \geq w_j$ and $C \geq \max\{d_i, d_j\}$. Then $i \prec j$ in optimal solution for the problem H.

Proof:

Let (δ_k, j, i) be the schedule which is obtained by interchanging jobs i and $j$in (δ_k, i, j).

Since $C \geq d_i$ and $C \geq d_j$, then the jobs i *and* j are late i.e.$h_i G_i = h_j G_j = 0$. So the effect on the cost depends only on weighted completion time. Since $p_i \leq p_j$, $w_i \geq w_j$ then $\frac{p_i}{w_i} \leq \frac{p_j}{w_j}$ and $i \prec j$ in optimal solution for the problem (H).■

8. Computational Experience

An intensive work of numerical experimentations has been performed. We first present in subsection (8.1) how instances (test problems) can be randomly generated.

8.1 Test Problems

The data were generated in this paper in the same way as in [13] that generates as follows:

- The processing time P_j is uniformly distributed in the interval [1,10].
- The due date d_i is uniformly distributed in the interval

[P(1-TF-RDD/2),P(1-TF+RDD/2)]; where $P = \sum_{j=1}^{n} P_j$ depending on the relative range of due date (RDD) and on the average tardiness factor (TF).

- The an integer weights w_j were generated from uniform

distribution [1,10].

- The an integer penalty h_j were generated from uniform

distribution[1,10].

For both parameters, the values 0.2, 0.4, 0.6, 0.8 and 1.0 are considered. For each selected value of n (where n is the number of jobs), ten problems were generated.

8.2. Computational Experience with the Lower and Upper Bound of (BAB) Algorithm

The BAB algorithm was tested by coding it in MATLAB 8.3.0 (R2014a) and implemented on Intel (R) Core(TM) i3-6100U CPU @ 2.30 GHZ, with RAM 4.00 GB personal computer.

Table (1.1) and table(1.2) illustrates the results for problem (H) obtained by(BAB) algorithm, we list 10 problems for each value of n, where n ∈{5,10,15,20,25,30}, and the optimal value, upper bound (UB), initial lower bound (ILB), the number of generated nodes (Nodes), the computational time in second (Time), and the number of unsolved problems (Status) and table (1.3) shows the results for problem (H) number of example which can solved in number of nodes of BAB algorithm.

The stopping condition for the BAB algorithm was determined and we consider that the problem is unsolved (state is 1), that the BAB algorithm is stopped after a fixed period of time, here after 30 minutes.

We observed from table (1.1), the heuristics of upper bound is good algorithm. It gives the value for objective function equal or near optimal value.

From the observation of the table (2.1) we find that the method of branch and bound (BAB) gives the optimal solution comparable to the method of complete enumeration (CE) Method.

Table (2.1) Compare between BAB and CE

n	BAB	CE
5	260	260
6	437	437
7	471	471
8	666	666
9	761	761
10	631	631

Table (1.1): The performance of initial number of nodes and computational time in second of BAB algorithm.

N	AV. of opt.	AV. Of time	AV. Of nodes	no. of un sol.
5	274.6	0.0097	12.1	0
10	1166.8	0.0418	193.3	0
15	2402.8	2.4798	13924	0
20	3531	8.6435	37100.3	0
25	5948.5	856.9530	1369890.3	1
30	7888.6	671.14604	2134214.5	2

Table (1.2): The performance of initial lower bound, upper bound, number of nodesand computational time in second of BAB algorithm (n=25)

n	EX	Optimal	UB	ILB	Nodes	Time	Status
25	1	5499	5510	5453	22854	4.1230	0
	2	6073	6073*	6041	2060	0.3679	0
	3	6064	6064*	5978	562765	120.9006	0
	4	5547	5547*	5494	37387	7.1316	0
	5	6717	6726	6668	20063	3.6163	0
	6	6696	6701	6618	1414914	265.6157	0
	7	5271	5283	5219	1795998	393.1902	0
	8	5975	5975*	5910	171254	30.7993	0
	9	5660	5669	5646	1219	0.2178	0
	10	5983	5983*	5897	9670389	1800	1
no of opt.			5	0	–	–	–

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.

* = The upper bound gives the optimal value.

** = The initial lower bound gives the optimal value.

$$\text{Status}=\begin{cases}0 & if\ the\ problem\ is\ solved\\ 1 & if\ the\ problem\ isn't\ solved\end{cases}$$

Table (1.3): The performance of number of example which solved in number of nodes of BAB algorithm For a table (1.1).

Table(1.3): number of example which solved in number of nodes in branch tree

n / no.of nodes	5	10	15	20	25	30
0-100	10	3		1		
101-500		6	2	2		
501-1000		1	4			1
1001-10000			3	6	2	1
10001-100000					3	1
100001-500000			1	1	1	
500001-1000000					1	1
1000001-20000000					3	6

REFERENCES

[1] A. Lann, and G. Mosheiov, "Single machine scheduling to minimize the number of early and tardy jobs." *Computers & Operations Research*, vol. 8, pp. 769-781, 1996.

[2] A. Lann, and G. Mosheiov, "A note on the maximum number of on-time jobs on parallel identical machines**".** Computers & Operations Research, vol. 30, pp.1745-1749, 2003.

[3] Huang RH and Yang CL (2007). "Single-machine scheduling to minimize the number of early jobs". Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management, 2–4 December, pp. 955, 957, doi: 10.1109/IEEM.2007.4419333.

[4] Baruch Mor and GurMosheiov ''Minimizing the number of early jobs on a proportionate flowshop'' Journal of the Operational Research Society (2014), 1–4.

[5] Smith, W.E., "Various optimizers for single stage production**"**, Naval Research Logistics Quarterly 3/1, 59-66 (1956).

[6] E. L. Lawler, Sequencing jobs to minimize total weighted completion time subject to precedence constraints. Ann. Discrete. Math. 2, (1978) 75-90.

[7] J. K. Lenstra and A. H. G. RinnooyKan, Complexity of scheduling under precedence constraints. Ops. Res,. 26 (1978) 22-35.

[8] A. H. G. RinnooyKan; B. J. Lageweg and J. K. Lenstra, Minimizing total costs in one-machine scheduling, Oper, Res. 23 (1975) 908-927.

[9] C. N. Potts, A Lagrangean based branch and bound algorithm for single machine sequencing with precedence constraints to minimize total weighted completion time, Management Science, Vol. 31, No. 10, October (1985) 1300-1311.

[10] R. M. Karp, Reducibility among combinatorial problems. In complexity of computer computations, Miller, R. E. and Thatcher, J. W. Eds. Plenum Press, New York (1972) 95-103.

[11] Araibi S.M., "Machine Scheduling Problem to Minimize Two and Three Objectives Function" M.Sc thesis, Dept. of mathematics, college of Education for Pure Sciences, Thi-Qar University (2012).

[12] Husein, N.A.,'' Machine Scheduling Problem to Minimize Multiple Objective Function'', M.Sc thesis, Dept. of Mathematics College of Education (Ibn AL-Haitham), Baghdad University (2012).

[13] Abdul-Razaq, T.S., Potts C.N. and Van Wassenhove, ''A survey of algorithms for the single machine total weighted tardiness scheduling problem'', Discrete App. Math. 26235-253(1990).