Branch and Bound Method to Minimize Three Objective Functions with Unequal Release Dates

Araibi Sami M [(1)] Al-Zuwaini Mohammed K [(2)]

[(1)] Dept. of Mathematics, College of Education for Pure Sciences, Thi-Qar University

s_m8339@yahoo.com

[(2)] Department of Mathematics College of Computer Science and Mathematics Thi-Qar University

mkzz50@yahoo.com

Abstract

In this paper, a single machine scheduling problem is considered to minimize Multiple Objectives Function (MOF). The aim in this study is to find the optimal solution for the sum of the number of tardy jobs ($\sum_{j=1}^{n} U_j$), maximum tardiness (T_{max}) and makespan (C_{max}) with unequal release dates. Ten special cases are derived and proved that yield optimal solutions. A Branch and Bound algorithm is proposed with two lower bounds (LB_1, LB_2) and two upper bounds (UB_1, UB_2) that introduced in this paper, in order to find the exact (optimal) solution for it with two dominance rules which helped in reducing the number of branches in the search tree. Results of extensive computational tests show the proposed (BAB) algorithm is effective in solving problem up to (40) jobs at a time less than or equal to (30) minutes. In general, this problem is strongly NP-hard, and to the best of our knowledge this problem was not studied before.

Keywords: Single machine, Scheduling, Multi-criteria, Number of tardy jobs, Maximum tardiness, Makespan, Release date.

1. Introduction

In this paper, the problem of scheduling *n* independent jobs on a single machine with unequal release date was considered to minimize the sum of the number of tardy jobs, maximum tardiness and makespan by using the (BAB) method. This problem is denoted by ($1/ r_j / \sum_{j=1}^{n} U_j + T_{max} + C_{max}$).

2. Approaches for Multi-Criteria Problems

In the literature there are two approaches for the multi-criteria problems [6]:

The first one is *hierarchical* or *lexicographical* minimization. The performance criteria $f^1, \ldots, f^k$ are indexed in order of decreasing importance. First, f^1 is minimized. Next, f^2 is minimized subject to the constraint that the schedule has minimal f^1 value. If necessary, f^3 is minimized subject to the constraint that the values for f^1 and f^2 are equal to the values determined in the previous step. The obvious disadvantage of this approach is that the resulting schedule may perform very poorly on all criteria except the first one; this is a serious

setback if the criteria are not subject to a hierarchy but are somewhat equally important. The first result on multi-criteria scheduling, obtained by Smith, is based on this approach [5]. The studies by Chang P. and Su L.(2001) [3] and Chen W., et al.(1997) [4] are examples of hierarchical minimization problems with earliness and tardiness costs. The computational complexity results in hierarchical minimization are reviewed in Lee and Vairaktarakis (1993) [9].

The second method is *simultaneous* minimization. In the simultaneous approach there are two types, the first one typically generates all efficient schedules and selects the one that yields the best composite objective function value of these criteria. The second is to find the sum of these objectives as we used in this paper. Several scheduling problems considering the simultaneous minimization of various forms of earliness and tardiness costs have been studied in the literature (see, e.g. Hoogeveen, (1995) [7]; Moslehi, et al. (2005) [10]).

3. Problem Formulation

Single machine scheduling models seem to be very simple but are 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.

The (MOF) problem (1/ r_j / $\sum_{j=1}^{n} U_j$+T_{max}+ C_{max}) can be stated as follows:

A set of *n* independent jobs N={1,2,…,*n*} are available for processing at time r_j, job $j(j = 1,2,\dots,n)$ is to be processed without interruption on a single machine that can be handled only one job at a time, requires processing time p_j, and ideally should be completed at its due date d_j. For a given sequence σ of the jobs, completion time $C_{\sigma(j)}$, number of tardy jobs $\sum U_{\sigma(j)}$ and the tardiness $T_{\sigma(j)}$ are given by:

$$\left.\begin{aligned} C_{\sigma(1)} &= r_{\sigma(1)} + p_{\sigma(1)} \\ C_{\sigma(j)} &= max\{C_{\sigma(j-1)}, r_{\sigma(j)}\} + p_{\sigma(j)} \quad , \; j = 2,\dots,n \end{aligned}\right\} \cdots (1)$$

$$U_{\sigma(j)} = \begin{cases} 1 & \text{if } \; C_{\sigma(j)} > d_{\sigma(j)} \quad , \; j = 1,2,\dots,n \\ 0 & otherwise \end{cases} \quad \cdots (2)$$

$$T_{\sigma(j)} = max\{C_{\sigma(j)} - d_{\sigma(j)}, 0\} \quad , \; j = 1,2,\dots,n \qquad \cdots (3)$$

The problem is strongly NP-hard because the problem $1/r_j / \sum_{j=1}^{n} U_j$ is NP-hard [3] and the problem $1/r_j/T_{max}$ is strongly NP-hard [13].

The problem is to find a sequence σ that minimizes the cost M. This problem is denoted by (Z) and can be stated as follows:

$$\left.\begin{array}{l} \mathrm{M} = min_{\sigma\in S}\{\sum_{\mathrm{j}=1}^{n} \mathrm{U}_{\sigma(\mathrm{j})} + \mathrm{T}_{max}(\sigma) + \mathrm{C}_{max}(\sigma)\} \\ s.t. \\ \mathrm{C}_{\sigma(\mathrm{j})} \geq r_{\sigma(\mathrm{j})} + p_{\sigma(\mathrm{j})} \quad , \quad \mathrm{j} = 1,2,\dots,n \\ \mathrm{C}_{\sigma(\mathrm{j})} \geq \mathrm{C}_{\sigma(\mathrm{j}-1)} + p_{\sigma(\mathrm{j})} \quad , \quad \mathrm{j} = 2,\dots,n \\ \mathrm{T}_{\sigma(\mathrm{j})} \geq \mathrm{C}_{\sigma(\mathrm{j})} - d_{\sigma(\mathrm{j})} \quad , \quad \mathrm{j} = 1,2,\dots,n \\ \mathrm{T}_{\sigma(\mathrm{j})} \geq 0\,,\ r_{\sigma(\mathrm{j})} \geq 0,\ p_{\sigma(\mathrm{j})} > 0 \quad , \quad \mathrm{j} = 1,2,\dots,n \end{array}\right\} \dots (\mathrm{Z})$$

Where $\sigma(\mathrm{j})$ denotes the position of job j in the ordering σ and S denotes the set of all enumerated schedules.

Note that if r_{j}=0 for each job j ∈ N, then the problem (Z) is reduced to the problem $1//\sum_{\mathrm{j}=1}^{n} \mathrm{U}_{\sigma(\mathrm{j})}+\mathrm{T}_{max}(\sigma)+ \mathrm{C}_{max}(\sigma)$.

4. Decomposition of Problem (Z) and Some Basic Results

In this section the problem (Z) is decomposed into three subproblems with a simple structure. Some results are stated which help in solving the problem (Z).

$\mathrm{M}=min_{\sigma\in S}\{\sum_{\mathrm{j}=1}^{n} \mathrm{U}_{\sigma(\mathrm{j})} + \mathrm{T}_{max}(\sigma) + C_{max}(\sigma)\}$

This problem can be decomposed into three subproblems (SZ_1), (SZ_2) and (SZ_3).

$$\left.\begin{array}{l} \mathrm{H}_1 = min_{\sigma\in S}\{\mathrm{T}_{max}(\sigma)\} \\ s.t. \\ \mathrm{C}_{\sigma(\mathrm{j})} \geq r_{\sigma(\mathrm{j})} + p_{\sigma(\mathrm{j})} \quad , \quad \mathrm{j} = 1,2,\dots,n \\ \mathrm{C}_{\sigma(\mathrm{j})} \geq \mathrm{C}_{\sigma(\mathrm{j}-1)} + p_{\sigma(\mathrm{j})} \quad , \quad \mathrm{j} = 2,\dots,n \\ \mathrm{T}_{\sigma(\mathrm{j})} \geq \mathrm{C}_{\sigma(\mathrm{j})} - d_{\sigma(\mathrm{j})} \quad , \quad \mathrm{j} = 1,2,\dots,n \\ \mathrm{C}_{\sigma(\mathrm{j})} > 0, \mathrm{T}_{\sigma(\mathrm{j})} \geq 0, r_{\sigma(\mathrm{j})} \geq 0, p_{\sigma(\mathrm{j})} > 0\,, \quad \mathrm{j} = 1,2,\dots,n \end{array}\right\} \dots.(\mathrm{SZ}_1)$$

$$\left.\begin{array}{l} \mathrm{H}_2 = min_{\sigma\in S}\{C_{max}(\sigma)\} \\ s.t. \\ \mathrm{C}_{\sigma(\mathrm{j})} \geq r_{\sigma(\mathrm{j})} + p_{\sigma(\mathrm{j})} \quad , \quad \mathrm{j} = 1,2,\dots,n \\ \mathrm{C}_{\sigma(\mathrm{j})} \geq \mathrm{C}_{\sigma(\mathrm{j}-1)} + p_{\sigma(\mathrm{j})} \quad , \quad \mathrm{j} = 2,\dots,n \\ \mathrm{C}_{\sigma(\mathrm{j})} > 0, r_{\sigma(\mathrm{j})} \geq 0, p_{\sigma(\mathrm{j})} > 0 \quad , \quad \mathrm{j} = 1,2,\dots,n \end{array}\right\} \dots.(\mathrm{SZ}_2)$$

$$\left.\begin{array}{l} \mathrm{H}_3 = min_{\sigma\in S}\{\sum_{\mathrm{j}=1}^{n} \mathrm{U}_{\sigma(\mathrm{j})}\} \\ s.t. \\ \mathrm{C}_{\sigma(\mathrm{j})} \geq r_{\sigma(\mathrm{j})} + p_{\sigma(\mathrm{j})} \quad , \quad \mathrm{j} = 1,2,\dots,n \\ \mathrm{C}_{\sigma(\mathrm{j})} \geq \mathrm{C}_{\sigma(\mathrm{j}-1)} + p_{\sigma(\mathrm{j})} \quad , \quad \mathrm{j} = 2,\dots,n \\ \mathrm{U}_{\sigma(\mathrm{j})} = 1 \quad \text{if} \quad \mathrm{C}_{\sigma(\mathrm{j})} > d_{\sigma(\mathrm{j})} \quad , \quad \mathrm{j} = 1,2,\dots,n \\ \mathrm{U}_{\sigma(\mathrm{j})} = 0 \qquad otherwise \end{array}\right\} \dots.(\mathrm{SZ}_3)$$

Theorem (1): $\mathrm{H}_1 + \mathrm{H}_2 + \mathrm{H}_3 \leq \mathrm{M}$ where $\mathrm{H}_1, \mathrm{H}_2, \mathrm{H}_3$ and M are the minimum objective function values of (SZ_1) , (SZ_2), (SZ_3) and (Z), respectively.

Proof:

Let H_1 be the maximum tardiness obtained by using preemption scheduling (i.e. (GEDD) order (sequencing the *n* jobs in non-decreasing order of smallest remaining due dates) [5]) then $H_1 \leq min\{T_{max}\}$ … (4)

Let H_2 be the makespan obtained by using (SRD) rule [1] then $H_2 \leq min\{C_{max}\}$… (5)

And let H_3 be the number of tardy jobs obtained by using relax release dates $r^* = min_{1\leq j\leq n}\{r_j\}$ and $r^* = r_j$, $j = 1,2,..,n$, to get the problem $1//\sum_{j=1}^{n} U_j$ having due date $d_j = d_j - r^*$ for each job j, this relaxed problem is solved by Moor algorithm [11] then $H_3 \leq min\{\sum_{j=1}^{n} U_j\}$ … (6)

Then from (4), (5) and (6) we get

$H_1 + H_2 + H_3 \leq min\{\sum_{j=1}^{n} U_j\} + min\{T_{max}\} + min\{C_{max}\}$

$\Longrightarrow H_1 + H_2 + H_3 \leq min\{\sum_{j=1}^{n} U_j + T_{max} + C_{max}\}$

Hence $H_1 + H_2 + H_3 \leq$M. ■

5. Special cases yield optimal solution

A machine scheduling problem of type NP-hard is not easily solvable and it is more difficult when the objective function is multi objective. Using some mathematical programming techniques to find the optimal solution for this kind of problem as: dynamic programming and branch and bound method. Some time special cases for this problem can be solved. A special case for scheduling problem means finding an optimal schedule directly without using mathematical programming techniques. A special case if it exists depends on satisfying some conditions in order to make the problem easily solvable. These conditions depend on the objective function as well as the jobs [8]. In this section some special cases of our problem (Z) are given.

Case (1): If $r_j = r$ and $d_j = d$, ∀j (j=1,2,…,*n*), then (MA) gives an optimal solution, for $1/r_j = r, d_j = d/\sum_{j=1}^{n} U_j + T_{max} + C_{max}$.

Proof:

Since $C_{max} = r + \sum_{j=1}^{n} p_j$ and $T_{max} = max\{r + \sum_{j=1}^{n} p_j - d\ , 0\}$ in any order. Then problem (Z) reduces to the problem $1/r_j = r/\sum_{j=1}^{n} U_j$, but this problem was solved by (MA) [12]. Then (MA) gives an optimal solution for the problem $1/r_j = r, d_j = d/\sum_{j=1}^{n} U_j + T_{max} + C_{max}$. ■

Case (2): If all jobs have a common due date (i.e. $d_j = d$, $(j = 1,2, \dots, n)$) and if (SRD) gives $d \geq C_{max}$, then (SRD) is an optimal solution for $1/r_j, d_j = d, d \geq C_{max}$ / $\sum_{j=1}^{n} U_j + T_{max} + C_{max}$ problem.

Proof:

Sine $d \geq C_{max}$, then no job with be tardy, i.e. T_{max}=0 and $\sum_{j=1}^{n} U_j$=0. Then the problem $1/r_j$, $d_j = d, d \geq C_{max}$ / $\sum_{j=1}^{n} U_j + T_{max} + C_{max}$ is reduced to $1/r_j / C_{max}$, but this problem was solved by (SRD) [1]. Then (SRD) gives the optimal solution. ■

Case (3): If $r_j = r$, $p_j = p$ and $d_j = d$, (j=1,2,…,*n*), then (MA) gives an optimal solution.

Proof:

Since $C_{max} = r + np$, $T_j = max\{ r + jp - d, 0\}$ and

$T_{max} = \begin{cases} r + np - d & if \quad r + np - d > 0 \\ 0 & otherwise \end{cases}$ in any order.

Then the problem $1/r_j = r, p_j = p, d_j = d/\sum_{j=1}^{n} U_j + T_{max} + C_{max}$ is reduced to $1/r_j = r /\sum_{j=1}^{n} U_j$, but this problem was solved by (MA) [12]. So (MA) gives an optimal solution for the problem $1/r_j = r, d_j = d, p_j = p/ \sum_{j=1}^{n} U_j + T_{max} + C_{max}$. ■

Case (4): If the (SRD) rule satisfies $C_j \leq d_j$ for each job j in (SRD), then (SRD) gives an optimal solution for $1/r_j / \sum_{j=1}^{n} U_j + T_{max} + C_{max}$ problem.

Proof:

Since $C_j \leq d_j$, ∀ j in SRD. Then no job with be tardy, i.e. $\sum_{j=1}^{n} U_j$ =0 and T_{max}=0. Then the problem $1/r_j$, $C_j \leq d_j$ / $\sum_{j=1}^{n} U_j + T_{max} + C_{max}$ is reduced to $1/ r_j/C_{max}$, but this problem was solved by (SRD) [1]. So (SRD) gives an optimal solution. ■

Case (5): If (SRD) rule satisfy $C_j - d_j = k$, ∀ job j in (SRD) (where *k* is a positive integer), then (SRD) gives an optimal solution for $1/r_j / \sum_{j=1}^{n} U_j + T_{max} + C_{max}$ problem.

Proof:

Since $C_j - d_j = k$, ∀ j in (SRD). Then $\sum_{j=1}^{n} U_j = n$ and $T_{max} = k$. Then the problem $1/r_j$, $C_j - d_j = k/ \sum_{j=1}^{n} U_j + T_{max} + C_{max}$ is reduced to $1/ r_j/C_{max}$, but this problem was solved by (SRD) [1]. Then (SRD) gives an optimal solution. ■

Case (6): The (SRD) rule is optimal for the problem $1/r_j / \sum_{j=1}^{n} U_j + T_{max} + C_{max}$ if $d_j = d$ and $min_{j \in SRD}\{C_j\} > d$, (j=1,2,…,*n*).

Proof:

Since $min_j\{ C_j\} > d$, (j=1,2,…,*n*) , i.e. $\sum_{j=1}^{n} U_j$= *n*. And T_{max}=$C_{max} - d$. Then the problem $1/r_j$,$d_j = d, min_{j \in SRD}\{C_j\} > d/ \sum_{j=1}^{n} U_j + T_{max} + C_{max}$ is reduced to $1/ r_j/C_{max}$, but this problem was solved by (SRD) [1]. Then (SRD) gives an optimal solution. ■

Case (7): If $r_j = r$ and T_{max}(MA) = T_{max}(EDD), then (MA) gives optimal solution for $1/r_j = r / \sum_{j=1}^{n} U_j + T_{max} + C_{max}$ problem.

Proof:

Since $C_{max} = r + \sum_{j=1}^{n} p_j$ in any order. Since $1/ r_j = r/T_{max}$ minimized by (EDD) rule [2]. And T_{max}(MA) = T_{max}(EDD). Then (MA) schedule is optimal for both criterion ($\sum_{j=1}^{n} U_j$ and T_{max}). Hence (MA) is optimal for $1/r_j/ \sum_{j=1}^{n} U_j + T_{max} + C_{max}$ problem. ■

Case (8): If $p_j = p$, $r_j = r$ and if there is a schedule π satisfy $d_j = jp + r$, $\forall$ job $j \in \pi$, then the schedule π gives an optimal solution for $1/p_j = p$, $r_j = r$, $d_j = jp + r / \sum_{j=1}^{n} U_j + T_{max} + C_{max}$ problem.

Proof:

For the condition of processing time and release dates $C_{max} = r + np$ in any order, $C_j = jp + r$ and $d_j = jp + r$, $\forall j, j \in \pi$, then $C_j = d_j$, this means all the jobs are just-in-time (JIT), i.e. $T_{max}=0$ and $\sum_{j=1}^{n} U_j = 0$ and the cost of objective function depends on C_{max} only. Hence the schedule π gives the optimal solution for $1/p_j = p$, $r_j = r$, $d_j = jp + r / \sum_{j=1}^{n} U_j + T_{max} + C_{max}$ problem. ■

Case (9): If $r_j = r$ and $\sum_{j=1}^{n} U_j$(MA)=$\sum_{j=1}^{n} U_j$(EDD), then (EDD) rule is optimal for the problem $1/r_j / \sum_{j=1}^{n} U_j + T_{max} + C_{max}$.

Proof:

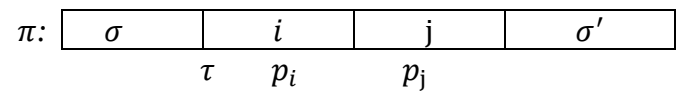
The condition of release date gives $C_{max} = r + \sum_{j=1}^{n} p_j$ in any order. Since $1/ r_j = r/\sum_{j=1}^{n} U_j$ is minimized by (MA)[12]. And $\sum_{j=1}^{n} U_j$(EDD) = $\sum_{j=1}^{n} U_j$(MA). Then (EDD) rule is optimal for both criterion ($\sum_{j=1}^{n} U_j$ and T_{max}). Hence (EDD) rule is optimal for $1/r_j$ / $\sum_{j=1}^{n} U_j + T_{max} + C_{max}$ problem. ■

Case (10): If the schedule π satisfies (SRD) and (EDD) at the same time, then π gives an optimal solution for $1/ r_j / \sum_{j=1}^{n} U_j + T_{max} + C_{max}$ problem.

Proof:

Consider the sequence $\pi = \sigma i j \sigma'$ where σ and σ' Partial sequences and i and j are two jobs with $r_i \le r_j$ and $d_i \le d_j$.

Let τ be a completion time of last job in σ and μ be the number of tardy jobs in σ ($\mu \ge 0$). Let $\pi' = \sigma j i \sigma'$ a new sequence (by interchange jobs i and j in original sequence) (see Fig (1)).



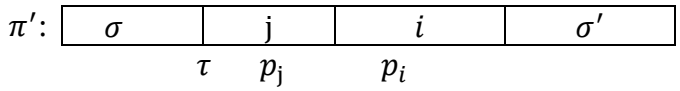


Fig (1): The schedules π and π'

In π: The number of tardy jobs in subsequence σij in π.

Let $R_i = max\{\tau, r_i\}$.

$C_i = R_i + p_i$ and $C_j = max\{C_i, r_j\} + p_j$.

If $C_i > d_i$.

Then $\sum_{j\in\sigma ij\subset\pi} U_j = \begin{cases}\mu+2 & \text{if job j is late}\\ \mu+1 & \text{if job j is early}\end{cases}$

If $C_i \le d_i$.

Then $\sum_{j\in\sigma ij\subset\pi} U_j = \begin{cases}\mu+1 & \text{if job j is late}\\ \mu & \text{if job j is early}\end{cases}$

In π': The number of tardy jobs in subsequence σji in π'.

Let $R_j = max\{\tau, r_j\}$

$C'_j = R_j + p_j$ and $C'_i = C'_j + p_i$.

If $C'_j > d_j$, then, job i is late and $\sum_{j\in\sigma ji\subset\pi'} U'_j = \mu + 2$.

If $C'_j \le d_j$.

Then $\sum_{j\in\sigma ji\subset\pi'} U'_j = \begin{cases}\mu+1 & \text{if job } i \text{ is late}\\ \mu & \text{if job } i \text{ is early}\end{cases}$

Then $\sum_{j\in\sigma ij\subset\pi} U_j \le \sum_{j\in\sigma ji\subset\pi'} U'_j$ … (7)

For schedule : The maximum tardiness in subsequence σij in π.

$T_{max} = max\{T, T_j, T_i\}$, where $T = max_{b\in\sigma}\{T_b\}$,

$$T_j = max\{C_j - d_j, 0\} \text{ and}$$

$$T_i = max\{C_i - d_i, 0\}$$

For schedule π': The maximum tardiness in σji in π'.

T'_{max}=$max\{T, T'_j, T'_i\}$,

$$T'_j = max\{C'_j - d_j, 0\} \text{ and}$$

$$T'_i = max\{C'_i - d_i, 0\}$$

Since $C'_i \geq C'_j$ and $d_i \leq d_j$, then $T'_i \geq T'_j$

So T'_{max}=$max\{T, T'_i\}$,

Since $r_j \geq r_i \Rightarrow max\{\tau, r_j\} \geq max\{\tau, r_i\} \Rightarrow R_j \geq R_i$

$\Rightarrow C'_i > C_i$ (Since $R_j \geq R_i$ and $p_j > 0$).

$\Rightarrow max\{C'_i - d_i, 0\} > max\{C_i - d_i, 0\} \Rightarrow T'_i > T_i$.

Since $C'_i = C'_j + p_i$ and $C_j = max\{C_i, r_j\} + p_j$.

- Either $C_i > r_j \Rightarrow C'_i = R_j + p_j + p_i$ and $C_j = R_i + p_j + p_i$.

$\Rightarrow C'_i \geq C_j$ (Since $R_j \geq R_i$).

- Or $r_j > C_i \Rightarrow C'_i = R_j + p_j + p_i$ and $C_j = r_j + p_j$.

$\Rightarrow C'_i > C_j$ (since $R_j \geq r_j$ and $p_i > 0$).

$\Rightarrow max\{C'_i - d_i, 0\} > max\{C_j - d_j, 0\} \Rightarrow T'_i > T_j$.

Therefore T'_{max}=$max\{T, T'_i\} \geq max\{T, T_i, T_j\} = T_{max}$

Then $T_{max} \leq T'_{max}$ … (8)

The makespan in $\sigma ij \subset \pi$ is $C_{max} = C_j$

The makespan in $\sigma ji \subset \pi'$ is $C'_{max} = C'_i$

Since $C'_i \geq C_j$,then $C_{max} \leq C'_{max}$ …(9)

From (7), (8) and (9), we get

$\Rightarrow \sum_{j \in \sigma ij \subset \pi} U_j + T_{max} + C_{max} \leq \sum_{j \in \sigma ji \subset \pi'} U'_j + T'_{max} + C'_{max}$ … (10)

Then from (10) π gives an optimal solution for $1/r_j/\sum_{j=1}^{n} U_j + T_{max} + C_{max}$. ■

6. Dominance Rules

Because of branching scheme, the size of the search tree is directly linked to the length of the current sequence (which represents the number of nodes). Hence, a preprocessing step

is performed in order to remove as many positions as possible. Reducing the current sequence is done by using several dominance rules. Dominance rules usually specify whether a node can be eliminated before its lower bound is calculated. Clearly, dominance rules are particularly useful when a node can be eliminated which has a lower bound that is less than the optimum solution. Some of dominance rules are valid for minimization of the sum of the number of tardy jobs, maximum tardiness and makespan.

As in the preprocessing step, similar dominance rules are also used within the branch and bound procedure to cut nodes that is dominated by others. These improvements lead to very large decrease in the number of nodes to obtain the optimal solution.

Below two of dominance rules are stated in order to decrease the number of nodes in search tree as well as decreasing the solution time [8].

Dominance Rule (1): If $r_i \leq r_j$ and $d_i \leq d_j$ then job i proceed job j in the optimal solution for the problem (Z).

Proof: It is clear from case (10).

■

Dominance Rule (2): If δ_k be a partial sequence which it's jobs are scheduled, K⊂N. For $i, j \in \overline{K}$=N−K, if $r_i < r_j$, $p_i < p_j$ and $(r_i + p_i) - d_i \geq (r_j + p_j) - d_j > 0$. Then job i proceed job j in the optimal solution for the problem (Z).

Proof:

Let (δ_k, j, i) be the schedule which is obtained by interchanging jobs i and j in (δ_k, i, j). All jobs other than i and j have the same completion time in (δ_k, i, j) as in (δ_k, j, i). So the difference in completion time between (δ_k,i,j) and (δ_k, j, i) depends only on the completion time of jobs i and j. Let τ be a completion time of last job in δ_k.

In (δ_k, j, i), $C_{max} = C_i$

In (δ_k,i,j), $C'_{max} = C'_j$

But ($C'_j \leq C_i$) since ($max\{\tau, r_j\} \geq max\{\tau, r_i\}$) and ($p_i > 0$)

Then $C'_{max} \leq C_{max}$ … (11)

The maximum tardiness in (δ_k,j,i) is:

$T_{max} = max\{T_m, T_j, T_i\}$, where $T_m = max_{b \in \delta_k}\{T_b\}$

$$T_j = max\{(max\{\tau, r_j\} + p_j) - d_j, 0\} \text{ and}$$

$$T_i = max\{(C_j + p_i) - d_i, 0\}$$

Since $(r_i + p_i) - d_i \geq (r_j + p_j) - d_j$ and $C_j > max\{\tau, r_j\}$, then $T_i \geq T_j$

So T_{max}=$max\{T_m, T_i\}$

The maximum tardiness of (δ_k, i, j) is:

T'_{max}=$max\{T_m, T'_j, T'_i\}$,

$$T'_i = max\{(max\{\tau, r_i\} + p_i) - d_i, 0\} \text{ and}$$

$$T'_j = max\{(max\{C'_i, r_j\} + p_j) - d_j, 0\}$$

But ($T'_j \leq T_i$) since

$(r_i + p_i) - d_i \geq (r_j + p_j) - d_j$, $max\{\tau, r_j\} + p_j \geq max\{\tau, r_i\} + p_i$ and $p_j > 0$.

And ($T'_i \leq T_i$) since $C_i \geq C'_i$ and it has the same due date.

Then $T'_{max} \leq T_{max}$ … (12)

From the condition, $(r_i + p_i) - d_i \geq (r_j + p_j) - d_j > 0$ the jobs i and j are late in both partial schedules (δ_k,j,i) and (δ_k,i,j).

Then from (11), (12) and lateness of jobs i and j we get, $i \prec$ j in the optimal solution for the problem (Z).

7. The Upper bound (UB)

In this section, heuristic methods are proposed and applied once at the root node of search tree (BAB) algorithm to find an upper bound (UB) on (Z). We suggested two heuristic methods, the best one of the heuristic is used to provide an upper bound (UB).

7.1. Heuristic (1)

The following algorithm (heuristic) is proposed to obtain the first upper bound (UB_1) for problem (Z).

Algorithm UB_1

Step (1): Order the jobs by (SRD) to obtain a sequence $\sigma = (\sigma_{(1)}, \sigma_{(2)}, \dots, \sigma_{(n)})$.

Step (2): If there exists a tie (jobs with the same release dates) applying (MA) on these jobs, otherwise go to Step (5).

Step (3): Let Q be a sequence of tardy jobs which is obtained from σ.

Step (4): Let π=$(\pi_{(1)}, \pi_{(2)}, \dots, \pi_{(n)})$ be a new sequence which is obtained from σ after ordering the jobs of Q according to (EDD), go to Step (6).

Step (5): Compute UB_1= $\sum_{j=1}^{n} U_{\sigma(j)} + T_{max}(\sigma) + C_{max}(\sigma)$, go to Step (7).

Step (6): Compute $UB_1 = \sum_{j=1}^{n} U_{\pi(j)} + T_{max}(\pi) + C_{max}(\pi)$.

Step (7): Stop.

Example (1): The first heuristic is illustrated in four jobs scheduling problem.

j	1	2	3	4
r_j	3	4	3	3
p_j	4	5	8	7
d_j	10	15	14	12

Solution:

The (SRD) schedule is (1,3,4,2). Applying (MA) on the jobs (1,3,4) the schedule (1,4,3,2) is obtained. Order the tardy jobs (3,4) by (EDD) rule to get π=(1,4,3,2).

Compute $UB_1 = \sum_{j=1}^{4} U_{\pi(j)} + T_{max}(\pi) + C_{max}(\pi)$=42.

It should be noted that an optimal sequence is (1, 4, 3, 2) for this example, and the optimal value is 42 which is obtained by using complete enumeration.

7.2. Heuristic (2)

The following algorithm (heuristic) is proposed to obtain the second upper bound (UB_2) for problem (Z).

Algorithm UB_2

Step (1): Order the jobs by (SRD) to get a sequence $\sigma = (\sigma_{(1)}, \sigma_{(2)}, \dots, \sigma_{(n)})$.

Step (2): If there exists a job i $(i = 1, \dots, n)$ such that $C_{\sigma(i)} \geq r_{\sigma(k)}$, $k = i+1,\ i+2, \dots, n$, then a new sequence $\pi = (\pi_{(1)},\ \pi_{(2)}, \dots, \pi_{(n)})$ which is obtained from σ after ordering the jobs $i+1,\ i+2, \dots, n$ according to (EDD) rule, otherwise go to Step (4).

Step (3): Compute $UB_2 = \sum_{j=1}^{n} U_{\pi(j)} + T_{max}(\pi) + C_{max}(\pi)$, go to Step (5).

Step (4): Compute $UB_2 = \sum_{j=1}^{n} U_{\sigma(j)} + T_{max}(\sigma) + C_{max}(\sigma)$.

Step (5): Stop.

Hence our upper bound is UB=min{ UB_1 , UB_2 }.

Example (2): The second heuristic is illustrated in four jobs scheduling problem.

j	1	2	3	4
r_j	3	6	5	2

p_j	4	3	6	7
d_j	10	15	13	11

Solution:

The (SRD) schedule is σ =(4,1,3,2). At the first completion time ($C_{\sigma(1)}$= 9) the set of jobs (1,2,3) are found which ready jobs ($r_j < C_{\sigma(1)}$), j ∈{1,2,3}. The jobs (1,2,3) are ordered by (EDD) rule, to get π=(4,1,3,2).

Compute $UB_2= \sum_{j=1}^{4} U_{\pi(j)} + T_{max}(\pi) + C_{max}(\pi)$ = 32.

It should be noted that an optimal sequence is (4, 1, 3, 2) for this example, and the optimal value is 32 which is obtained by using complete enumeration.

8. The Lower Bound (LB)

In this section, two lower bounds LB_1 and LB_2 are derived for problem (Z) and LB=$max\{LB_1, LB_2\}$.

8.1. The First Lower Bound (LB_1)

The first lower bound is based on decomposing (Z) into three subproblems $(SZ_1), (SZ_2)$ and (SZ_3) as shown in Section (4), then H_1 was calculated to be the lower bound for (SZ_1), H_2 to be the minimum value for (SZ_2), H_3 to be the lower bound for (SZ_3) and applying Theorem (1) to get a lower bound LB_1 for problem (Z).

8.2. The Second Lower Bound (LB_2)

To obtain the second lower bound for problem (Z), the relaxation of constraints of objective function will be as follows:

If there exists a job *i* such that $r_i \le r_j$ and $d_i \le d_j$, $j = 1, \dots n-1$, then job *i* schedule in the first position. For the remaining jobs we assume that $r^*=min_{j\in N/\{i\}}\{r_j\}$ and $d^*=max_{j\in N/\{i\}}\{d_j\}$, and applying Case (1). The optimal solution for the new problem is a lower bound for problem (Z).

Proposition (1):

The (LB_2) is a lower bound for problem (Z).

Proof:

Firstly, if there is a job j satisfies the condition $r_j \le r_i$ and $d_j \le d_i$, $i = 1, \dots, n-1$, then job j is scheduled in the first position by Dominance rule (1). For remaining jobs $r^*=min_{j\in N/\{i\}}\{r_j\}$ is assumed and $d^*=max_{j\in N/\{i\}}\{d_j\}$. This problem can be solved by using Case (1). The first assumption for the release date to get a maximum reduction of the completion time and makespan, and the second assumption for the due date to get a maximum

reduction of the tardy jobs, maximum tardiness and the number of tardy jobs. This means that by this way the maximum reduction to the cost of the objective function is obtained.

Secondly, if there is no job satisfies above condition, then we assume $r^*=min_{1\le j\le n}\{r_j\}$ and $d^*=max_{1\le j\le n}\{d_j\}$ and repeat above argument. ■

Example (3): The second lower bound is illustrated in five jobs scheduling problem.

j	1	2	3	4	5
r_j	0	1	7	6	9
p_j	3	4	6	10	2
d_j	3	10	17	18	15

Solution:

Since $d_1 \le d_j$ and $r_1 \le r_j$, j ∈{2,3,4,5}, then job 1 scheduled in the first position. Since $d_2 \le d_j$ and $r_2 \le r_j$, j ∈{3,4,5}, then job 2 scheduled in the second position. $r^*=min\{7,6,9\}=6$ is assumed and $d^*=max\{17,18,15\}=18$. Applying (MA) for the remaining jobs {3,4,5}, we get

j	1	2	3	5	4
C_j	3	7	13	15	25
T_j	0	0	0	0	7

Then LB_2= 1+7+25=33

It should be noted that an optimal sequence is (1,2,3,5,4) for this example, and the optimal value is 33 which is obtained by using complete enumeration.

9. Branch and Bound (BAB) algorithm

In this section, a description of our branch and bound (BAB) algorithm is given and its implementation. The two heuristic methods are applied at the top of search tree (root node) the better of the two heuristic sequences is used to provide an upper bound (UB) on cost of an optimal schedule is obtained by choosing the better of two upper bounds from Section (7). Also at the top of the search tree an initial lower bound (ILB) on the cost of an optimal schedule is obtained by choosing the better of two lower bounds from Section (8). Our algorithm uses a forward sequencing branching rule for which nodes at level k of the search tree correspond to initial sequences in which jobs are sequenced in the first k positions.

The branching procedure describes the method to partition a subset of possible solution. These subsets can be treated as a set of solutions of corresponding subproblems of the original problem. The bounding procedure indicates how to calculate a lower bound (LB) on the optimal solution value for each subproblem generated in the branching process. The search strategy describes the method of choosing a node of the search tree to branch from it; we

usually branch from a node with smallest lower bound (LB) among the recently created nodes.

Example (4): The (BAB) algorithm is illustrated in five jobs scheduling problem.

j	1	2	3	4	5
r_j	0	5	8	12	14
p_j	8	6	5	6	10
d_j	16	26	24	22	32

The (BAB) tree algorithm to give the optimal solutions for the problem $1/r_j/\sum_{j=1}^{n} U_j + T_{max} + C_{max}$, is shown in Fig (2).

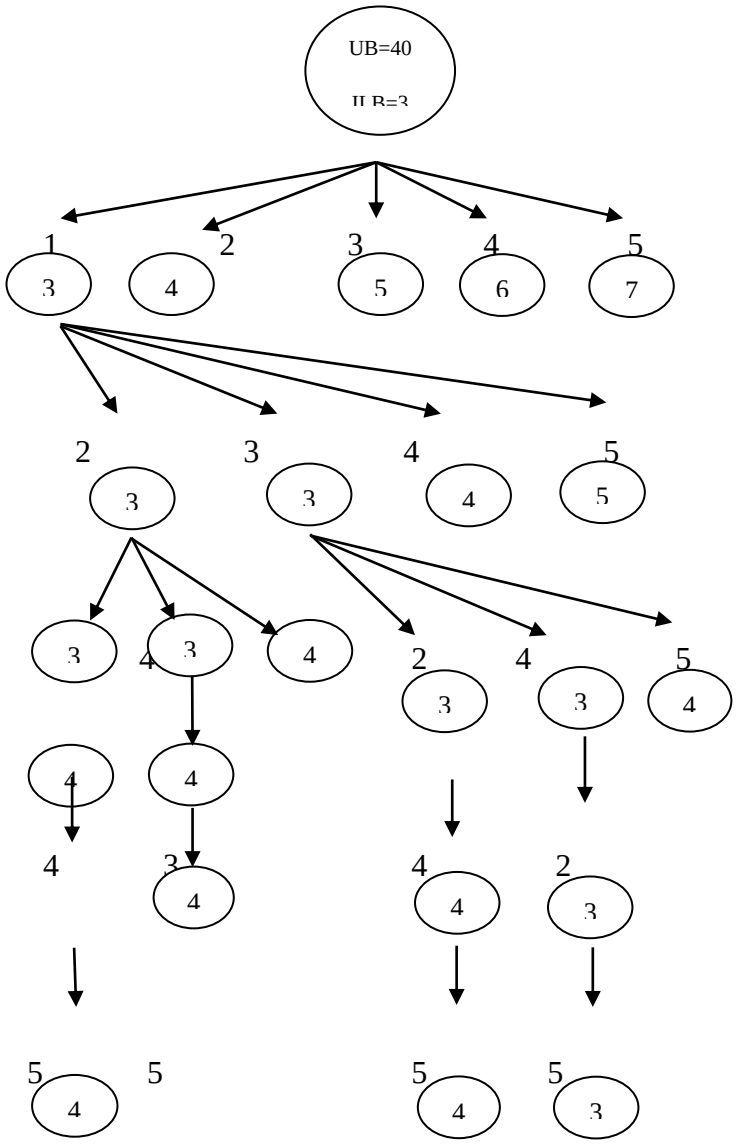


Fig (2): The (BAB) tree for problem (Z)

The (BAB) method gives the schedule (1,3,4,2,5) that gives ($\sum_{j=1}^{n} \mathrm{U_j} + \mathrm{T}_{max} + \mathrm{C}_{max}$)=39.

10. Computational Experience

An intensive work of numerical experimentations has been performed. Subsection (10.1) shows how instances (test problems) can be randomly generated.

10.1. Test Problems

There exists in the literature a classical way to randomly generate test problems of scheduling problems.

- The processing time p_j is uniformly distributed in the interval [1, 10].
- The release date r_j is uniformly distributed in the interval [0,αP], where [α=0.125, 0.25, 0.50, 0.75, 1.00] and P=$\sum_{i=1}^{n} p_i$.
- The due date d_j is uniformly distributed in the interval [P(1-TF-RDD/2),P(1-TF+RDD/2)];where P= $\sum_{i=1}^{n} p_i$.

  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 and 1.0 are considered. For each selected value of n where n is the number of jobs, five problems were generated.

10.2. Computational Experience with the Lower and Upper Bounds of (BAB) Algorithm

The (BAB) algorithm was tested by coding it in MATLAB 7.10.0 (R2010a) and implemented on Intel(R) Core(TM)2 Duo CPU T6670 @ 2.20 GHZ, with RAM 2.00 GB personal computer.

Table (1), shows the results for problem (Z) obtained by (BAB) algorithm. The first column "n" refers to the number of jobs, the second column "EX" refers to the number of examples for each instance n, where $n \in \{5, 10, 15, 20, 25, 30, 35, 40\}$, the third column "Optimal" refers to the optimal values obtained by (BAB) algorithm for problem (Z), the fourth column "UB" refers to the upper bound, the fifth column "ILB" refers to the initial lower bound, the sixth column "Nodes" refers to the number of nodes, the seventh column "Time" refers to the time cost 'by second' to solve the problem, the last column "Status" refers to the problem solved '0' or not '1'. The symbols "*" refers to UB gives an optimal solution and "**" refers to ILB gives an optimal solution. The (BAB) algorithm was stopped when the sum of "status column ≥3".

A condition for stopping the (BAB) algorithm was determined and considering 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).

If the value of UB=ILB then the optimal is UB and there is no need to branch in the search tree of (BAB) algorithm.

From Table (1), it is noticed that the heuristic of upper bound is good algorithm. It gives the value for objective function equal to optimal or near optimal values.

Table (1): The performance of initial lower bound, upper bound, number of nodes and computational time in second of (BAB) algorithm for (Z).

n	EX	Optimal	UB	ILB	Nodes	Time	Status
5	1	65	65*	64	14	0.0028	0
	2	32	33	31	14	0.0021	0
	3	43	43*	41	14	0.0018	0
	4	44	44*	44**	0	0.0008	0
	5	33	34	32	14	0.0022	0
10	1	117	117*	113	93	0.0107	0
	2	70	71	67	96	0.0105	0
	3	103	104	100	119	0.0131	0
	4	125	125*	121	162	0.0160	0
	5	59	61	56	54	0.0065	0
15	1	167	168	163	366	0.0386	0
	2	139	140	135	119	0.0143	0
	3	163	166	160	309	0.0324	0
	4	100	105	97	630	0.0641	0
	5	140	143	136	210	0.0227	0
20	1	203	206	198	1469	0.1628	0
	2	210	214	205	12664	1.2978	0
	3	166	171	162	1263	0.1351	0
	4	182	187	176	2985	0.3243	0
	5	165	171	160	8836	0.9188	0

Table (1): continued

n	EX	Optimal	UB	ILB	Nodes	Time	Status
25	1	261	267	257	2602	0.3099	0
	2	256	262	250	4921	0.5762	0
	3	274	281	270	35120	4.0163	0
	4	223	229	217	11453	1.3186	0
	5	258	264	253	33863	3.8647	0
30	1	379	384	374	5047	0.6587	0
	2	300	305	294	150111	18.9608	0
	3	277	284	269	767672	98.9491	0
	4	261	269	252	4308611	551.2338	0
	5	284	294	277	849200	106.0457	0
	1	407	413	400	30517	4.8198	0

35	2	396	404	390	108684	15.9127	0
	3	343	350	335	1727223	256.8693	0
	4	394	401	387	11750681	1664.3691	0
	5	339	349	330	7261310	969.0710	0
40	1	393	403	387	838338	138.2405	0
	2	436	446	428	3447719	606.1721	0
	3	449	459	440	3735047	665.1139	0
	4	428	433	414	11173344	1800.0122	1
	5	380	390	368	12063100	1800.0019	1

Table (2) summarizes Table (1)

Table (2) is the summary of Table (1), and shows the average of nodes and computational times for the solved problems, also, shows the unsolved problems among the 5 problems of each n, where $n \in \{5, 10, 15, 20, 25, 30, 35, 40\}$.

Table (2): Summary of Table (1) of (BAB) algorithm

n	Av. Nodes	Av. Time	Unsolved problem
5	11.2	0.0019	0
10	104.8	0.0114	0
15	326.8	0.0344	0
20	5443.4	0.5678	0
25	17591.8	2.0171	0
30	1216128.2	155.1696	0
35	4175683	582.2084	0
40	2673701.3	469.8422	2

10. Conclusions

In this paper, the problems of scheduling jobs on one machine for a variety of three-criteria are considered.

A branch and bound algorithm is proposed to find the optimal solution for the problem $1/r_j/\sum_{j=1}^{n} U_j + T_{max} + C_{max}$ with two lower bounds (LB_1, LB_2), two upper bounds (UB_1, UB_2) and two dominance rules. Ten special cases for the problem (Z) are derived and proved.

References

[1] Baker, K.R., "Introduction to Sequencing and Scheduling", John Wily, New York (1974).

[2] Brucker P.,"Scheduling Algorithms", Springer-Verlag Berlin Heidelberg (2006).

[3] Chang P., Su L., "Scheduling *n* Jobs on One Machine to Minimize the Maximum Lateness with a Minimum Number of Tardy Jobs", Computer and Industrial Engineering, 40, 349-360 (2001).

[4] Chen T.,Qi X., and Tu F., "Single Machine Scheduling to Minimize Weighed Earliness Subject to Maximum Tardiness", Computer of Operation Research, 24 , 147-152 (1997).

[5] Horn, W.A., "Some Simple Scheduling Algorithm", Naval Research Logistics Quarterly 21, 177-185 (1974).

[6] Hoogeveen J.A,"Single Machine Scheduling to Minimize a Function of Two or Three Maximum Cost Criteria", Journal of Algorithms, 21,415-433 (1996).
[7] Hoogeveen, J.A., Van de Velde, S.L.," Minimizing Total Completion Time and Maximum Cost Simultaneously is Solvable in Polynomial Time", Operations Research Letters 17, 205-208 (1995).
[8] 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).
[9] Lee, C.V., and Vairaktarakis, G.L.," Complexity of Single Machine Hierarchical Scheduling: A Survey Complexity in Numerical Optimization", World Scientific 19, 269-298 (1993).
[10] Moslehi G., Moghaddam R., Vasei M., and Azaron A.,"Optimal Scheduling for a Single Machine to Minimize the Sum of Maximum Earliness and Tardiness Considering Idle Insert", Applied Mathematics and Computation 167, 1430-1450 (2005).
[11] Moore,J.M.,"An *n* Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs", Mgnt. Sci. 15(1), 102-109 (1968).
[12] P. Baptiste.,"An $O(n^4)$ Algorithm for Preemptive Scheduling of a Single Machine to Minimize the Number of Late Jobs", Operations Research Letters, 24:175-180, 1999.
[13] Pinedo, M.L.,"Scheduling Theory, Algorithms, and Systems", Springer Science+Business Media, Llc., New York (2008).

الخلاصة

في هذا البحث, درسنا مسألة جدولة الماكنة الواحدة لتصغير دالة متعددة الاهداف (MOF). هدفنا في هذه الدراسة هو ايجاد الحل الامثل لمجموع عدد النتاجات المتأخرة ($\sum_{j=1}^{n} U_j$) وأكبر زمن تأخير (T_{max}) وأكبر زمن إتمام (C_{max}) مع وقت تحضير للنتاجات غير متساوي. تم اشتقاق وبرهان عشرة حالات خاصة تعطي الحلول المثلى. كذلك اقترحنا خوارزمية التفرع والتقيد مع اثنين من القيود الدنيا (LB_1, LB_2) واثنين من القيود العليا (UB_1, UB_2) قُدمت في هذا البحث من اجل ايجاد الحل الامثل للمسألة قيد الدراسة مع قاعدتين للهيمنة تساعدان في تقليص عدد التفرعات في شجرة البحث. نتائج الاختبارات الحسابية أثبتت بأن خوارزمية التفرع والتقيد المقترحة فعالة في حل المسائل لغاية (40) نتاج في وقت أقل او يساوي (30) دقيقة. بشكل عام هذه المسألة من نوع strongly NP-hard, وحسب معرفتنا ان هذه المسألة لم تُدرس من قبل.