ISSN 1991- 8690 الترقيم الدولي 8690 - 1991

Branch and Bound Method to Solve Multiple Objective Function

Mohammad Kadhim Al-Zuwaini [1] Najah Ali Husein[2]
mkzz50@yahoo.com najah_math2006@yahoo.com

[1] Math. Dep. - College of Computer Science and Mathematics - Thi-qar University
[2]Math. Dep. - College of Education - Al-Qadisiyh University

Abstract:

This paper presents a branch and bound algorithm for sequencing a set of jobs on a single machine scheduling with the objective of minimizing total cost of flow time and maximum earliness, when the jobs may have unequal ready time.

For solving this problem we proposed two lower bounds (LB_1, LB_2) by decomposing the problem into two subproblems. The lower bounds of the problem is the sum of the lower bounds of two subproblems. The proposed heuristic algorithm, which is used as an upper bound in the branch and bound (BAB) algorithm, is effective in finding an optimal or near optimal schedule. Also, we prove some special cases of the problem which lead to optimal solution. We stated and proved three dominance rules. Results of extensive computational tests show the proposed (BAB) algorithm is effective in solving problems up to about (50) jobs at a time less than or equal to (30) minutes.

Keywords: Flow time; Maximum earliness; Scheduling; Ready time.

طريقة التفرع والتقيد لحل دالة متعددة الأهداف

محمد كاظم الزويني[1] نجاح علي حسين[2]

[1] قسم الرياضيات – كلية علوم الحاسبات والرياضيات – جامعة ذي قار

[2] قسم الرياضيات – كلية التربية – جامعة القادسية

المستخلص:

يقدم هذا البحث خوارزمية التفرع والتقيد لترتيب مجموعة من النتاجات على الماكنه الواحدة, الهدف تصغير الكلفة الكلية لزمن انسياب النتاجات وكلفة أكبر تبكير عندما يكون للنتاجات أزمنة تحضير غير متساوية.لحل هذه المسألة تم اشتقاق قيدين أدنين (LB_1 ,LB_2) بتجزئة المسألة الأصلية إلى مسألتين جزئيتين. القيدان الأدنيان للمسألة قيد البحث هما مجموع القيود الدنيا للمسألتين الجزئيتين. الخوارزمية التقريبية المقترحة والتي استخدمت في طريقة التفرع والتقيد كقيد أعلى كانت تعطي حالا امثلا أو قريبا من الحل الأمثل. كذلك برهنا بعض الحالات الخاصة للمسأله والتي تقودنا إلى الحل الأمثل وتم برهان ثلاث قواعد هيمنة. تم تقييم كفاءة خوارزمية التفرع والتقيد المقترحة على مجموعة كبيرة من المسائل الأختباريه يبين بأنها فعالة في حل المسائل إلى ما يقارب من (50) نتاجاً وبزمن اقل أو يساوي (30) دقيقة.

1. Introduction:

The problem of sequencing n jobs on one machine under different assumptions and multiple criteria is considered extensively. In this study the objective function to be minimized of two criteria with unequal ready times: sum of flow time denoted by $\sum F_i$ plus maximum earliness denoted by E_{max} . We assume that the two criteria have the same importance. Denote this problem by $1/r_i/\sum F_i + E_{\max}$.

This problem is of a remarkable importance in addition to processing and minimizing the time of the flow of works on the machine. This is achieved from the arrival time in the work site (when it is ready for working on the machine) to the time of the work achievement. Furthermore it is possible to reduce the strong time for the works which require from the achievement till delivery to the beneficiaries. The process of strong time is sometimes expensive and complex. The processing of such type of problems has considerable importance especially in the field of agriculture and industry. This is especially true when handling the problems of factories which produce items with short periods of validity for use such as food, chemical substance, serums, crops and fruits.

The following are some of literature review:

For the problem $1/r_i/\sum F_i$ when all jobs are available at time zero that is well known and can polynomially solved (Smith 1956) [16]. The preemptive version, $1/r_i$, pmtn $/\sum F_i$, of the problem can be solved optimally in polynomial time by the Shortest Remaining Processing Time (SRPT) rule, as shown by Baker (1974) [6]. Lenstra et al. (1977) [14] have shown that the $1/r_i/\sum F_i$ problem is NP – complete.

The solution of the preemptive problem provides a lower bound for the objective value of the non – preemptive problem, Ahmed and Bagchi (1990) [1] have shown that this lower bound dominates all other known lower bounds for the $1/r_i/\sum F_i$ problem.

Al-Zuwaini (2000) [4] used efficient branch and bound technique with a suitable lower bound and proved some dominance rules for $1/r_i/\sum F_i$ problem. Several researchers studied this criteria ($\sum F_i$) with other criteria to consider multi–criteria scheduling for example ($\sum F_i + \sum U_i$) see [5].

The problem $1/r_i = 0/E_{\max}$ can be solved optimally in polynomial time when inserted idle time is not allowed, maximum earliness is minimized by the minimum Slack Time (MST) rule [10]. When inserted idle time is allowed, minimum E_{max} value is zero and it is trivial to convert any sequence to such a schedule [2]. Mahnam and Moslehi (2009) [15] used the optimal solution of $1/r_i$, pmtn $/E_{\max}$ by using MRST rule to obtain a lower bound for $1/r_i/E_{\max}$ problem with no – uniform – idle – time assumption. Also several researchers studied this criteria (E_{max}) with other criteria to study multi – criteria scheduling for example (E_{max} +T_{max}) see [15].

Koksalan et al (1998) [12] proposed a heuristic to give "generate all approximately efficient sequences " for the problem to minimize the flow time and maximum earliness on a single machine . Ahmet and koksalan (2003) [2], used Genetic algorithm scheduling problem: The total completion times and the maximum earliness. Kurz and conterbury (2005) [13] used genetic algorithm to find the set of efficient points for $1//(\sum C_i, E_{\max})$ problem. Al-Assaf (2007) [3] used the BAB algorithm to find the optimal solution for the problem $1//\sum C_i + E_{\max}$ and proposed a polynomial algorithm with in special range for the problem $1//(\sum C_i, E_{\max})$.

Huang and Yang (2009) [11] presented an algorithm for efficient scheduling in terms of total flow time and maximum earliness.

Delphi, A. M (2011) [9] proposed efficient algorithm that can be used to enumerate the set of strict pareto optimal for the bi–criteria scheduling problem without release dates on a single machine which is studied like $1//(\sum C_i, E_{\max})$.

2. Problem Formulation:

The general problem of scheduling jobs on a single machine to minimize the total cost that can be stated as follows: A set of n independed jobs N={1,2,….,n} which has to be scheduled without preemption on a single machine that can handle at most one job at a time. The machine is assumed to be continuously available from time zero onwards and no precedence relationship exists between jobs. Each job j, $j \in N$ has an integer processing time P_j, a release date r_j and ideally should be completed at its due date d_j. For any given schedule (1,2,…,n), the flow time of job j, F_j and the maximum earliness E_{max} can be respectively defined as: $F_j=C_j-r_j$, where C_j be a completion time for job j, given by the relationship:

$C_1 = r_1 + p_1$, $C_j = \max\{r_j, C_{j-1}\} + p_j$ for j=2,3,…,n and $E_{\max} = \max_{1\le j\le n}\{E_j\}$, $E_j = \max\{d_j - C_j, 0\}$, j=1,2,…….,n.

The objective is to find the schedule that minimize the sum of the total flow time and maximum earliness costs of all jobs with release dates on a single machine (i.e. minimize the Multiple Objective Function (MOF) denoted by $\left(\sum_{j=1}^{n} F_j + E_{\max}\right)$. It is clear that our model differs from the other models (See for example). Koksalan et al. (1998) [12], Ahmet and Koksalan (2003) [2], Kurz and Canterbury (2005) [13], AL-Assaf (2007) [3], Huang and Yang (2009) [11]. In that we consider a more general and realistic problem dealing with arbitrary release dates. The problem is strongly NP-hard because the $1//\sum C_i + E_{\max}$ problem with zero release date is NP-hard [12][2][3].

Our scheduling problem can be stated mathematically more precisely as follows:

Given a schedule δ = (1,2,….,n), then for each job $j \in \delta$ can be calculated the flow time F_j and the maximum earliness E_{max}. The objective is to find a schedule, σ = (σ(1), σ(2), …, σ(n)) that belongs to a neighborhood of δ that minimizes the total cost Z(σ), where $Z(\sigma) = \sum_{j=1}^{n} F_{\sigma(j)} + E_{\max}(\sigma)$.

Let S be a set of all schedules, $|S| = n!$, then we can formulate our problem in mathematical form as:

3. Decomposition of the Problem (P) and Some Basic Results:

In this section we decompose the problem (P) into two subproblems with a simpler structure, and state some results which help us in solving the problem (P).

As shown in section (2) the problem (P) has an objective function:

$$M = \min_{\sigma\in S}\{Z(\sigma)\} = \min_{\sigma\in S}\left\{\sum_{j=1}^{n} F_{\sigma(j)} + E_{\max}(\sigma)\right\}$$

This problem can be decomposed into two subproblems (P_1) and (P_2).

$$
\left.
\begin{aligned}
& Z_1 = \min_{\sigma \in S} \left\{ \sum_{j=1}^{n} F_{\sigma(j)} \right\} \\
& S.\,to: \\
& C_{\sigma(j)} \geq r_{\sigma(j)} + p_{\sigma(j)} \qquad j = 1,2,\ldots,n \\
& C_{\sigma(j)} \geq C_{\sigma(j-1)} + p_{\sigma(j)} \qquad j = 1,2,\ldots,n \\
& F_{\sigma(j)} = C_{\sigma(j)} - r_{\sigma(j)} \qquad j = 1,2,\ldots,n \\
& F_{\sigma(j)} \geq P_{\sigma(j)} \qquad j = 1,2,\ldots,n \\
& p_{\sigma(j)} > 0, \quad r_{\sigma(j)} \geq 0 \qquad j = 1,2,\ldots n
\end{aligned}
\right\} \quad (P_1)
$$

$$
\left.
\begin{aligned}
& Z_2 = \min_{\sigma \in S} \{\max\{ E_{\sigma(j)} \}\} \qquad j = 1,2,\ldots,n \\
& S.to: \\
& C_{\sigma(j)} \geq r_{\sigma(j)} + p_{\sigma(j)} \qquad j = 1,2,\ldots,n \\
& C_{\sigma(j)} \geq C_{\sigma(j-1)} + p_{\sigma(j)} \qquad j = 1,2,\ldots,n \\
& E_{\sigma(j)} \geq d_{\sigma(j-1)} - C_{\sigma(j)} \qquad j = 1,2,\ldots,n \\
& p_{\sigma(j)} > 0, \; r_{\sigma(j)} \geq 0 \qquad j = 1,2,\ldots,n \\
& E_{\sigma(j)} \geq 0 \qquad j = 1,2,\ldots,n
\end{aligned}
\right\} \quad (P_2)
$$

Theorem (1) [3]

If Z_1, Z_2 and M are the minimum objective function values of (P_1), (P_2) and (P) respectively then $Z_1 + Z_2 \leq M$.

Theorem (2)

If $r_i = r$ for all $i \in N$, then MST schedule gives the optimal solution for the problem $1/r_i / E_{max}$.

Proof:

Consider the schedule $S_1 = \sigma_1 ij \sigma_2$, Where σ_1 and σ_2 are two partial sequences of jobs, i and j are two jobs with $d_i - p_i \geq d_j - p_j$ and let T be the completion time for the last job of σ_1.

Let S_2 be the schedule obtained from S_1 by interchanging jobs i and j and leaving the rest of the sequence unaltered.

Let E be the maximum earliness for the jobs σ_1 and σ_2 under S_1, and E' be the maximum earliness for the jobs of σ_1 and σ_2 under S_2. Clearly $E = E'$.

Let E_i, E_j be the earliness of i and j respectively under S_1 and E'_i, E'_j be the earliness of i and j respectively under S_2.

We, therefore, have the maximum earliness under S_1 $E_{\max} = \max\{E, E_i, E_j\}$

and the maximum earliness under S_2

$$E'_{\max} = \max\{E', E'_i, E'_j\} = \max\{E, E'_i, E'_j\}$$

Now under S_1

$$E_i = \max\{d_i - (T + p_i), 0\}, \quad E_j = \max\{d_j - (T + p_i + p_j), 0\},$$

and under S_2

$$E'_j = \max\{d_j - (T + p_j), 0\}, \quad E'_i = \max\{d_i - (T + p_j + P_i), 0\}$$

It is clear that $E_i \geq E'_i$ and $E_i \geq E'_j$

Since $E_{\max} = \max\{E, E_i, E_j\} \geq \max\{E, E_i\}$

$$\geq \max\{E, E'_i, E'_j\}$$

$$= \max\{E', E'_i, E'_j\} = E'_{\max}$$

So $E_{\max} \geq E'_{\max}$

Thus MST schedule gives the optimal solution for the $1 / r_i = r / E_{max}$ problem.

Theorem (3)

If $r_i = r$ for all $i \in N$, then SPT schedule gives the optimal solution for the problem $1/r_i/\sum_{i=1}^{n} F_i$.

Proof:

Consider the schedule $S_1 = (\sigma_1 ij \sigma_2)$, Where σ_1 and σ_2 are partial sequence of jobs and i, j are two jobs with $p_i \geq p_j$.

Let T be the completion time for the last job of σ_1.

for the schedule S_1 we have

$$F_i + F_j = 2(T + p_i - r) + p_j \qquad \text{.....(1)}$$

Now consider the new schedule S_2 which is obtained from schedule S_1 by interchanging the position of jobs i and j, $S_2 = (\sigma_1 ji \sigma_2)$

For the schedule S_2 we have

$$F'_j + F'_i = 2(T + p_j - r) + p_i \qquad \text{.....(2)}$$

From (1) and (2) we have

$$\sum_{\substack{i=1\\ i\in S_1}}^{n} F_i - \sum_{\substack{i=1\\ i\in S_2}}^{n} F_i' = 2(T + p_i - r) + p_j - 2(T + p_j - r) - p_i$$

Then $\sum_{\substack{i=1\\ i\in S_1}}^{n} F_i - \sum_{\substack{i=1\\ i\in S_2}}^{n} F_i' \geq 0$ (since $P_i \geq P_j$)

Hence the schedule S_2 is preferred rather than S_1, by repeating the process, we get SPT schedule gives the optimal solution for the $1/r_i = r/\sum_{i=1}^{n} F_i$ problem.

4. Special Cases Yield Optimal Solution:

A machine scheduling problem type NP-hard is not easily solvable and it is more difficult when the objective function is multi objective. Using some mathematical programming to find optimal solution for this kind of problem as: dynamic programming and branch and bound method. Some time studied a special cases for this problem. A special case for scheduling problem means finding an optimal schedule directly with out 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. In this section we give some special cases of our problem (P).

Case (1):

If for any $i \in N$, $p_i=p$, $r_i=r$ and $d_i=d$, then any sequence gives an optimal solution for the problem (P).

Proof*:*

Since all jobs have a constant data w.r.t. r_i, p_i and d_i, then we have

$$\sum_{i=1}^{n} F_i + E_{\max} = \sum_{i=1}^{n} C_i - \sum_{i=1}^{n} r_i + \max_{1\leq i\leq n}\{E_i\}$$

$$= nr + \frac{1}{2}n(n+1)p - nr + \max\{d - p, d - 2p, ..., d - np, 0\}$$

$$= \frac{1}{2}n(n+1)p + \max\{d - p, 0\}$$

$$= \begin{cases} (\frac{1}{2}n^2 + \frac{1}{2}\mathrm{n} - 1)\mathrm{p} + \mathrm{d} & \text{if } \mathrm{d} > \mathrm{p} \\ (\frac{1}{2}n^2 + \frac{1}{2}n)p & \text{if } \mathrm{d} \leq \mathrm{p} \end{cases}$$

Then $\sum_{i=1}^{n} F_i + E_{\max} =$ constant

Therefore any sequence gives an optimal solution for the problem (P).

Case (2):

If $r_i = r$, $\forall i \in N$ and $d_j = kp_j$, then SPT schedule is optimal for the problem (P) for each positive integer k greater than one (k > 1).

Proof:

Since $r_i = r$, $\forall i \in N$, then SPT schedule gives an optimal solution for $1/r_i = r/\sum_{i=1}^{n} F_j$ (by theorem 3).

Let $S_j = d_j - p_j$ be the slack time of job j, $\forall j \in N$

Since $d_j = kp_j$, then $S_j = kp_j - p_j = (k-1)p_j$ for all job j in SPT rule Since SPT schedule gives a non decreasing order of the processing time of jobs.

i.e. $p_1 \le p_2 \le \ldots\ldots\ldots \le p_n$

That is $(k-1)\, p_1 \le (k-1)p_2 \le \ldots\ldots\ldots \le (k-1)p_n$

So $S_1 \le S_2 \le \ldots\ldots\ldots \le S_n$ (that is MST ordered)

By using theorem (2) we obtained that MST schedule gives an optimal solution for the problem (P).

Case (3):

If for all $i \in N$, $r_i = r$, $p_1 \le p_2 \le \ldots. \le p_n$ and $d_1 \le d_2 \le \ldots. \le d_n$, then SPT and EDD schedules give the optimal solution for the problem (P).

Proof:

Consider a schedule $\sigma = (\sigma_1 ij \sigma_2)$, where σ_1 and σ_2 are partial sequences of jobs and i, j are two jobs with $r_i = r_j$, $p_i \le p_j$ and $d_i \le d_j$. Let $\sigma' = (\sigma_1 ji \sigma_2)$ be the schedule which is obtained by the interchanging the jobs i and j in σ (see Fig. 1). For these scheduling we study two cases, in each case we will make a comparison between them.

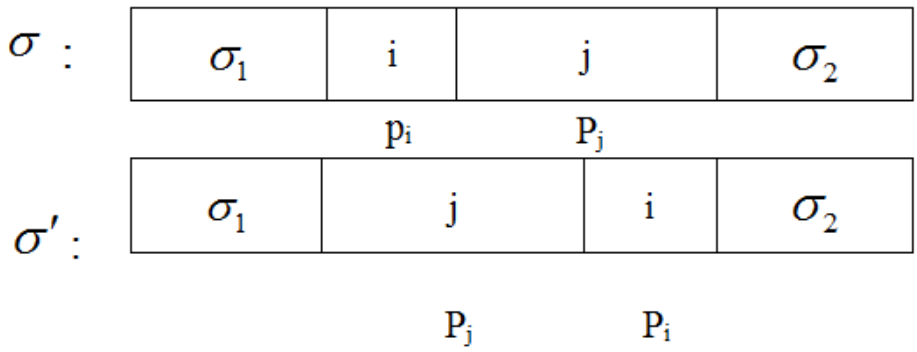

Fig -1- The schedule σ and σ'

1) If $r_i = r_j$, $p_i \le p_j$ and $d_i \le d_j$ yields that $S_i \le S_j$.

in this case we have :

The condition on the release dates and processing times ensure that (by theorem 3).

$$\sum F_k(\sigma) \le \sum F_k(\sigma') \qquad \ldots. (3)$$

And the condition on the slack times ensure that (by theorem 2).

$E_{\max}(\sigma) \le E_{\max}(\sigma')$

Hence $\sum F_k(\sigma) + E_{\max}(\sigma) \le \sum F_k(\sigma') + E_{\max}(\sigma')$.

2) If $r_i = r_j$, $p_i \le p_j$ and $d_i \le d_j$ yields that $S_i \ge S_j$.

In this case we have :

The condition on the release dates and processing times ensure that (3) (by theorem 3) is satisfied and the addition in cost which is obtained from (3) is equal to $p_j - p_i$.

i.e. $\sum F_k(\sigma') = \sum F_k(\sigma) + p_j - p_i$ ....(4)

Since $C_i(\sigma) = C_j(\sigma') - p_j + p_i$, then $d_i - C_i(\sigma) = d_i - C_j(\sigma') + p_j - p_i$

Since $S_i = d_i - p_i \ge S_j = d_j - p_j$, then $d_i - p_i - C_j(\sigma') \ge d_j - p_j - C_j(\sigma')$

$\Rightarrow d_i - p_i + p_j - C_j(\sigma') \ge d_j - C_j(\sigma')$ from which we deduce that

$E_{\max}(\sigma) \ge E_{\max}(\sigma')$ the addition in cost which is obtained from this inequality is equal to $S_i - S_j$.

i.e. $E_{\max}(\sigma) = E_{\max}(\sigma') + (S_i - S_j)$ ....(5)

Since $p_i \le p_j \Rightarrow p_j - p_i \ge 0 \quad \forall i, j$ ….(6)

Since $d_i \le d_j \Rightarrow d_j - d_i \ge 0 \quad \forall i, j$ ….(7)

From (6) and (7) we have $S_i - S_j \le p_j - p_i$

$\Rightarrow E_{\max}(\sigma') + (S_i - S_j) \le E_{\max}(\sigma') + p_j - p_i$ (by adding $E_{\max}(\sigma')$ for both sides).

By (5) we get $E_{\max}(\sigma) \le E_{\max}(\sigma') + p_j - p_i$

$\Rightarrow \sum F_k(\sigma) + E_{\max}(\sigma) \le \sum F_k(\sigma) + E_{\max}(\sigma') + p_j - p_i$

(by adding $\sum F_k(\sigma)$ to both sides).

From (4) we get $\sum F_k(\sigma) + E_{\max}(\sigma) \le \sum F_k(\sigma') + E_{\max}(\sigma')$.

Case (4):

If for all $i \in N$, $r_i = r$ and SPT schedule gives $E_{\max}(SPT) = E_{\max}(MST)$. Then SPT is optimal for the problem (P).

Proof:

Since $r_i = r$, $\forall i \in N$, then SPT schedule gives the optimal solution for the $1/r_i = r/\sum F_i$ problem (theorem 3).

But $E_{\max}$ minimized by MST rule if $r_i = r$, $\forall i \in N$ (theorem 2), then SPT schedule is optimal for both criterion

(Since SPT gives $E_{\max}(SPT) = E_{\max}(MST)$).

So SPT is optimal for the problem (P).

Case (5):

If $r_1 \le r_2 \le ..... \le r_n$, $p_1 \le p_2 \le ..... \le p_n$ and SPT (SRD) schedule satisfying $C_i \ge d_i$ for all $i \in N$, then SPT (SRD) schedule gives the optimal solution for the problem (P).

Proof:

The condition $C_i \geq d_i$ means that each job j is a late job and $E_j = 0 \ \forall j$, then the problem reduced to $1/r_i/\sum_{i=1}^{n} F_i$, But Al-Zuwaini (2000) [4] proved that SPT is optimal solution for $1/r_i/\sum_{i=1}^{n} F_i$ problem provided that $r_i \leq r_{i+1}$ for each job i, i=1,2,….,n-1.

Case (6):

If for all $i \in N$, $p_i = p$ and $r_i = r$, then any schedule satisfying $C_i \geq d_i$, $\forall i \in N$, gives the optimal solution for the problem (P).

Proof:

Let S be a schedule satisfying the condition ($C_i \geq d_i$, $\forall i \in N$)

Since $C_i \geq d_i$ for all $i \in N$, then for each job i, job i is either late or JIT (Just In Time) and $E_i = 0$ for all $i \in N$, so $E_{\max} = 0$.

Then the problem (P) reduces to $1/r_i = r, p_i = p/\sum_{i=1}^{n} F_j$ problem.

But $\sum_{i=1}^{n} F_i = \sum_{i=1}^{n} C_j - \sum_{i=1}^{n} r_i = \frac{1}{2} n(n+1)p$ = constant.

So any schedule gives the optimal solution for the $1/r_i = r$, $p_i = p/\sum_{i=1}^{n} F_j$ problem.

Therefore S schedule gives the optimal solution for the problem (p).

Case (7):

If for all $i \in N$, $r_i = r$ and SPT schedule satisfying $d_i + p_j \leq d_j$ for each i and j, then SPT is optimal for the problem (P).

Proof :

Since $r_i = r$, $\forall i \in N$, then SPT schedule gives an optimal solution for $1/r_i = r/\sum_{i=1}^{n} F_i$ (by theorem 3).

Since $d_i + p_j \leq d_j , \forall i, j$ in SPT schedule

$\Rightarrow \quad d_i \leq d_j - p_j , \ \forall i, j$ in SPT schedule

$\Rightarrow \quad d_i - p_i \leq d_j - p_j , \ \forall i, j$ in SPT schedule (Since $p_i \geq 0$).

Thus by using Theorem (2) we get that SPT rule gives the optimal solution for the problem (P).

5. Branch and Bound (BAB) Method to Find Optimal Solution:

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

5.1 Upper bound procedure:

Let t be a time which is a machine available after it ;

$R_i(t) = \max(t, r_i)$ the earliest beginning time of job i at time t.

$C_i(t) = R_i(t) + P_i$ the earliest completion time of job i at time t.

Then as in {chn, C. (1990[7], 1992[8])} we will use component of two rules first in first out (FIFO) and shortest processing time (SPT) to define a non-decreasing piecewise continuous function G(i,t) of job i at time t as : $G(i,t) = R_i(t) + C_i(t)$.

This function only depends on the parameters defining job i.

In this subsection, we build a heuristic to evaluate their performance to obtain an upper bound which is using in root node of search tree in Branch and method.

Heuristics to Calculate Upper Bound (UB)

Given a set of jobs N={1,2,…,n}.

Step (1) : Initialized t = 0 , A = {1,2,….,n} and $\sigma = \phi$

Step (2) : Select job i with $\min_{i \in A}$ G(i,t). Breat ties by choosing i with min { $R_i(t)$ }, and further ties by choosing i with min d_i.

Step (3) : Update t , A and σ , such that $t = C_i(t)$, A=A-{i}, $\sigma = \sigma \cup \{i\}$

Step (4) : If $A \neq \phi$, return to step 2.

Step (5) : Compute $UB = \sum_{i=1}^{n} F_i(\sigma) + E_{\max}(\sigma)$.

5.2 Derivation of Lower Bound

A lower bound for the problem (P) is based on decomposition (P) into two subproblems (P_1) and (P_2) as shown in section (3), then calculation of Z_1 to be the lower bound for P_1 and Z_2 to be the lower bound for P_2, then applying theorem (1) to get a lower bound (LB) for our problem (P).

For subproblem P_1 we used two lower bounds, the first one depends on relaxation the capacity of the machine (the machine can handle more than one job), that is we use SRPT rule to solve the $1/r_i, pmtn/\sum F_i$ problem which is suggested by Baker (1974) [6] to obtain a lower bound for P_1 which is denoted by $LB_1(P_1)$. The second lower by using the lower bound which is proposed by Al-Zuwaini (2000)[4] depends on relaxation the constraint on the jobs, that is relaxation the release date, we denoted this lower bound by $LB_2(P_1)$.

$LB_2(P_1)$ = nr*+α - R, where $r^* = \min_{i \in N}\{r_i\}$, $R = \sum_{i=1}^{n} r_i$ and α be an optimal solution for $1/r_i = 0/\sum F_i$ problem.

For the subproblem P_2, we propose the following algorithm to obtain a lower bound.

Algorithm [LB(P_2)] :

Given a set of jobs N={1,2,3,…,n}.

Step (1): Compute $S_i = d_i - p_i$ for i=1,2,…,n, where S_i be the slack time of job i.

Step (2): Beginning from $S^* = \max_{1 \le i \le n}\{r_i\}$, where S^* be a start time which is all jobs are available and order the jobs according to MST rule.

Step (3): Evaluate $E_i = \max\{d_i - C_i, 0\}$, where $C_i = S^* + \sum_{j=1}^{i} P_j$

for i=1,2,…,n.

Step (4): Find $E'_{\max} = LB(E_{\max}) = \max_{1\le i\le n}\{E_i\}$

We establish next that $LB(P_2)$ given by the above algorithm is a valid lower bound for the subproblem P_2.

Proposition (4):

Algorithm[LB(P2)] gives a lower bound for the $1/r_i/E_{\max}$ problem.

Proof :

It is well known when we are allowed to insert machine idle time, the **maximum** earliness value can be easily forced to its minimum value of zero. When the idle time decreasing the maximum earliness be increasing.

Then made the value of idle time $\max_{1\le i\le n}\{r_i\}$ contributes to minimize the maximum earliness. (By theorem 2) MST rule gives min. maximum earliness for $1/r_i = r/E_{\max}$ problem. So the algorithm[LB(P2)] gives a lower bound for the $1/r_i/E_{\max}$ problem.

Therefore we used two lower bounds for our problem (P)

i. $LB_1 = LB_1(P_1) + LB(P_2)$

ii. $LB_2 = LB_2(P_1) + LB(P_2)$

5.3 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 total flow time and maximum earliness.

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

Below we state some of dominance rules in try to decrease number of nodes in search tree as well as decreasing the time.

Dominance Rule (1)

If δ_k be a partial sequence which it's jobs are schedule, $K \subset N$. For $i, j \in \overline{K} = N - K$, if $I_i < I_j$, $2(r_j - r_i) \ge p_i - p_j$ and $d_i - (r_i + p_i) \le d_j - (r_j + p_j)$. Then $i \prec j$ in optimal solution for the problem (P).

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 flow time in (δ_k, i, j) as in (δ_k, j, i). So the difference in total flow time between

(δ_k , i, j) and (δ_k , j, i) depends only on the flow time of jobs i and j. The total flow time of jobs i and j in (δ_k , j, i) is:

$F_j + F_i = r_j + 2p_j + p_i - r_i$ .... (8)

The total flow time of jobs i and j in (δ_k , i, j) is:

a- If $r_i + p_i \le r_j$

$$F_i^* + F_j^* = p_i + p_j \quad \text{.... (9)}$$

b- If $r_i + p_i > r_j$

$$F_i^* + F_j^* = r_i + 2p_i + p_j - r_j \quad \text{.... (10)}$$

From (8) and (9) we get $(F_j + F_i) > (F_i^* + F_j^*)$

From (8) and (10) we get

$$(F_j + F_i) - (F_i^* + F_j^*) = 2(r_j - r_j) + (p_j - p_i) \ge 0$$

Then in both cases $(F_j + F_i) \ge (F_i^* + F_j^*)$

So $i \prec$ j for $1/r_i/\sum_{i=1}^{n} F_i$ problem.

On the other hand, the maximum earliness of (δ_k , j, i) is:

$E_{\max} = \max\{E_m, E_j, E_i\}$, where $E_m = \max_{1 \le m \le k}\{E_m\}$

$$E_j = \max\{d_j - (r_j + p_j), o\} \text{ and}$$

$$E_i = \max\{d_i - (r_i + p_j + p_i), o\}$$

Since $d_i - (r_i + p_i) \le d_j - (r_j + p_j)$, then $E_i \le E_j$

So $E_{\max} = \max\{E_m, E_j\}$

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

$E_{\max}^* = \max\{E_m, E_i^*, E_j^*\}$, where $E_m = \max_{1 \le m \le k}\{E_m\}$

$$E_i^* = \max\{d_i - (r_i + p_i), 0\} \text{ and}$$

$$E_j^* = \max\{d_j - (r_i + p_i + I_j + p_j, 0\}$$

But $(E_i^* \le E_j)$ since $d_i - (r_i + p_i) \le d_j - (r_j + p_j)$

and $(E_j^* \le E_j)$ since C_j, $j \in (\delta_k, i, j)$ greater than or equal

C_j, $j \in (\delta_k, j, i)$ and it has the same due date.

Therefore $E_{\max}^* = \max\{E_m, E_i^*, E_j^*\} \le \max\{E_m, E_j) = E_{\max}$

Then $E_{\max}^* \le E_{\max}$

So $i \prec$ j for $1/r_i/E_{\max}$ problem.

Hence $i \prec$ j in optimal solution for the problem (P).

Dominance Rule (2)

Let δ_k be a partial sequence which it's jobs are scheduled, $K \subset N$. For $i, j \in \bar{k} = N - K$ and let C be the completion time of the last job of δ_k. If $P_i \le P_j$ and $C \ge \max\{d_i, d_j\}$. Then $i \prec$ j in optimal solution for the problem (P).

Proof :

Let (δ_k , j, i) be the schedule which is obtained by interchanging jobs i and j in (δ_k , i, j). Since $C \ge d_i$ and $C \ge d_j$, then the earliness of jobs i and j is equal to zero. So the effect on the cost depend only on flow time.

The difference in total flow time between (δ_k , j, i) and (δ_k , i, j) depends only on flow time for jobs i and j.

The total flow time of jobs i and j in (δ_k , j, i) is:

$F_1 = 2C + 2P_j + p_i - (r_i + r_j)$.

The total flow time of jobs i and j in (δ_k , i, j) is:

$F_2 = 2C + 2P_i + p_j - (r_i + r_j)$

$F_2 \le F_1$ (since $p_i \le p_j$)

Then $i \prec$ j in optimal solution for the problem (P).

Dominance Rule (3)

Let δ_k be a partial sequence which it's jobs are scheduled, $K \subset N$. For $i, j \in \bar{k} = N - K$, if $C(k,i) \ge C(k,j)$, $C(k,i) - C(k,j) \ge (P_i - P_j)(|k^c| - 1)$ and $d_i - d_j \ge C(k,i) - C(k,j)$. Then (δ_k , i) is dominated by (δ_k , j).

Proof *:*

Yanai and fujie (2004)[17] show that (δ_k , i) is dominated by (δ_k , j) provided that $C(k,i) \ge C(k,j)$ and $C(k,i) - C(k,j) \ge (P_i - P_j)(|k^c| - 1)$ for $1/r_i / \sum_{i=1}^{n} F_i$ problem.

Since $d_i - d_j \ge C(k,i) - C(k,j)$, then $d_i - C(k,i) \ge d_j - C(k,j)$

So the maximum earliness of (δ_k , i) is grater than or equal the maximum earliness of (δ_k , j).

Hence (δ_k , i) is dominated by (δ_k , j).

6. Computational Experience:

An intensive work of numerical experimentations has been performed. We first presented in subsection (6.1) how instances (test problems) can be randomly generated.

6.1 Test Problems

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

- The processing time P_i is uniformly distributed in the interval [1,10].
- The release date r_i is uniformly distributed in the interval $[0, \alpha P]$, where [α = 0.125, 0.25, 0.50, 0.75, 1.00] and $P = \sum_{i=1}^{n} P_i$.
- The due date d_i 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, ten problems were generated.

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

The BAB algorithm was tested by coding it in Matlab 7.9.0 (R2009b) and implemented on Intel (R) Core (TM) i3 CPU M380 @ 2.53 GHZ, with RAM 4.00 GB personal computer.

In BAB algorithm, we proposed two lower bounds (see 5.2), in root node of search tree we calculated LB_1 as an initial lower bound. Whereas at bounding procedure we calculated LB_2 for each subproblem (node) generated in the branching process.

In Table (1), we give the comparative of computational results of BAB algorithm for the problem (P). We list 10 problems for each value of n, where $n \in \{5,10,15,20,25,30,35,40,45,50\}$, and 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 stopped the BAB algorithm and consider that the problem is unsolved (state is 0), that the BAB algorithm is stopped after a fixed period of time, here after 1800 second (i.e. after 30 minutes).

We observed from Table (1), the heuristic of upper bound is good algorithm, it gives the value for objective function equal to optimal or near optimal value.

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

n	EX	Optimal	UB	ILB	Nods	Time	Status
5	1	71	71*	71**	0	0.0007	0
	2	64	64*	58	5	0.0016	0
	3	90	97	84	14	0.0012	0
	4	31	31*	27	5	0.0011	0
	5	35	35*	32	5	0.0011	0
	6	46	46*	37	14	0.0011	0
	7	62	64	54	14	0.0021	0
	8	78	78*	60	14	0.0011	0
	9	71	71*	67	5	0.0011	0
	10	77	77*	62	21	0.0015	0
10	1	299	299*	273	54	0.0038	0
	2	185	193	165	150	0.0076	0
	3	262	266	231	155	0.0079	0
	4	177	180	148	281	0.0133	0
	5	208	214	169	564	0.0260	0
	6	189	199	165	54	0.0032	0
	7	167	170	153	54	0.0032	0
	8	219	219*	188	127	0.0065	0
	9	218	218*	186	71	0.0040	0
	10	267	269	228	93	0.0050	0
15	1	677	686	614	2499	0.1180	0
	2	392	392*	383	15	0.0068	0
	3	616	628	544	314	0.0160	0
	4	416	419	393	146	0.0080	0
	5	474	477	438	230	0.0119	0
	6	545	567	516	1014	0.0474	0
	7	419	419*	396	133	0.0076	0
	8	542	544	466	5895	0.2685	0
	9	495	495*	450	119	0.0077	0
	10	465	475	428	15353	0.6939	0
20	1	807	807*	723	20	0.0139	0
	2	697	700	616	57502	2.7259	0
	3	906	918	838	48972	2.3034	0
	4	814	815	730	574	0.0291	0
	5	829	845	747	540	0.0272	0
	6	1043	1077	974	33070	1.5449	0
	7	708	721	643	398	0.0206	0
	8	551	554	525	1063	0.0519	0
	9	764	770	730	6522	0.3094	0
	10	681	682	611	228	0.0125	0
25	1	1294	1295	1175	34313	0.6019	0
	2	1225	1243	1146	686	0.0357	0
	3	1422	1428	1344	6070	0.2883	0
	4	989	1007	883	28822	1.0405	0
	5	1306	1326	1240	8819	0.4151	0
	6	1468	1476	1389	1703	0.0820	0
	7	1363	1367	1212	2318341	107.7491	0
	8	1060	1063	972	155660	7.1875	0
	9	933	933*	829	5081	0.2382	0
	10	1053	1063	937	149166	6.8734	0

Table (1) : continued

n	EX	Optimal	UB	ILB	Nods	Time	Status
30	1	1496	1526	1375	10903843	516.8847	0
	2	1850	1868	1805	943	0.0498	0
	3	1881	1882	1833	545	0.0303	0
	4	1584	1622	1509	702471	33.7975	0
	5	1319	1320	1258	4542	0.2307	0
	6	1871	1889	1807	569854	28.3455	0
	7	1566	1567	1503	7032	0.3489	0
	8	1890	1893	1791	454724	21.5672	0
	9	1740	1742	1628	5867	0.2873	0
	10	1469	1470	1356	508721	25.2964	0
35	1	1873	1911	1813	291196	14.3918	0
	2	2230	2231	2091	73178	3.5949	0
	3	1931	1984	1858	31325	1.5498	0
	4	1761	1793	1705	2203	0.1131	0
	5	2028	2041	1928	3430203	167.4190	0
	6	1835	1835*	1715	11276	0.5625	0
	7	2363	2391	2241	41368	2.0225	0
	8	2115	2129	1953	1113016	54.8254	0
	9	2541	2578	2421	9717782	474.1790	0
	10	2079	2093	2014	73817	3.6010	0
40	1	2724	2747	2652	336813	16.9737	0
	2	2980	3011	2878	1055892	52.5247	0
	3	2823	2825	2666	264540	13.1510	0
	4	2868	2878	2799	133522	6.7137	0
	5	2469	2469*	2412	23461	1.1767	0
	6	2649	2672	2530	18523549	921.1932	0
	7	2649	2686	2550	289685	14.3133	0
	8	2018	2023	1916	36478695	1800.0005	1
	9	2692	2695	2634	89966	4.5667	0
	10	2323	2334	2259	2237399	111.4485	0
45	1	4555	4631	4370	800766	41.5030	0
	2	3881	3953	3844	46906	2.4708	0
	3	4103	4130	3898	354124	18.4565	0
	4	3980	3991	3876	95951	5.0590	0
	5	3616	3627	3468	9338034	493.9656	0
	6	3411	3411*	3275	34997857	1800.0006	1
	7	3576	3621	3392	15714078	815.9384	0
	8	3917	3923	3650	674696	35.5959	0
	9	3594	3594*	3426	42171	2.2217	0
	10	4302	4315	4026	34966217	1800.0001	1
50	1	3973	3985	3778	35461181	1800.0034	1
	2	5029	5105	4873	35114631	1800.0007	1
	3	3837	3839	3780	46812	2.4330	0
	4	3979	4024	3854	27206119	1357.3581	0
	5	4590	4590*	4517	5853	0.3102	0
	6	4175	4218	3999	2765145	141.3378	0
	7	4886	4943	4751	287911	14.9796	0
	8	4710	4713	4547	14378944	737.7821	0
	9	3605	3605*	3472	479867	24.5355	0
	10	3839	3883	3671	1491761	76.0985	0

Optimal = the optimal value obtained by BAB method.
UB = upper bound .
ILB = initial lower bound.

Nodes = the number of general nodes.
Time = Computational time in seconds.
* = The upper bound gives the optimal value.
** = The initial lower bound gives the optimal value.

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

The following Table summarizes Table (1):

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

N	Av. nodes	Av. time	Unsolved problem
5	9.7	0.0013	0
10	160.3	0.0081	0
15	2571.8	0.1186	0
20	14888.9	0.7039	0
25	270866.1	12.4512	0
30	1315854.2	62.6838	0
35	1478536.4	72.2259	0
40	2550536.3	126.8957	1
45	3383340.75	176.9014	2
50	5832801.5	294.3544	2

Table (2) is the summary of Table (1), that shows the average of nodes and computational time for the solved problems, also, shows the unsolved problems among the 10 problems of each n, where $n \in \{5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$.

We observed that whenever n increases the number of nodes and the computational time increasse also.

7. Conclusion

In this paper, we have developed exact and approximate methods for the problem of scheduling n jobs on a single machine to minimize the sum of total flow time and maximum earliness with unequal ready times. The proposed heuristic algorithm, which are used as an upper bound in the branch and bound algorithm, is effective in finding an optimal or near optimal schedule. The lower bounds which is used in branch and bound algorithm based on decomposition the problem into two subproblems. The proposed branch and bound algorithm can be used to find optimal solution for the problem up to (50) jobs at a time less than or equal to (30) minutes. Several results concerning optimality of seven solvable special cases are presented. We stated and proved three dominance rules which help in reducing the number of branches in the search tree.

References:

[1] Ahmadi R. H. and Bagchi U., "Lower bound for single machine scheduling problem ", Naval Research Logistics, 37,967-79 (1990).

[2] Ahmet Burak, Koksalan M., "Using Genetic Algorithm for Single-Machine Bicriteria Scheduling Problems, European Journal of Operational Res,145,543-556 (2003).

[3] Al-Assaf S.S.,"Solving multiple objectives scheduling problems", M.Sc. thesis Univ. of Al- Mustansiriyah, College of Science, Dep. of Mathematics (2007).

[4] Al-Zuwaini M. K.," Single machine scheduling to minimize total cost of flow time with unequal ready times", Journal of Basrah Researches, V.25,Par.3,94-108 (2000).

[5] Al-Zuwaini M.K.," One machine scheduling problem with release dates and two criteria" ,Journal of Thi-Qar University, No.2, Vol.6,1-14, (2011).

[6] Baker, K.R "Introduction to Sequencing and Scheduling " Wile New York, (1974).

[7] Chu C., " One machine scheduling for minimizing total flow time with release dates ", Proc. of Rensselaer's Sec. Conf. on C./M. Troy. Ny. May 21-23,pp570-576, (1990).

[8] Chu C., " A branch-and-bound algorithm to minimize total flow time with unequal release dates ", Naval Research Logistics, 39, 859–875 (1992).

[9] Delphi A. M., "Algorithms to solve multicriteria scheduling problems on single machine", M.Sc. thesis Univ. of Al-Mustansiriyah, College of Science, Dep. of Mathematics (2011).

[10] Franch, S. "Sequencing and Scheduling an Introduction to Mathematics of Job Shop", John Wiley & Sons, New York (1982).

[11] Huang R-H and Yang C-L," An algorithm for minimizing flow time and maximum earliness on a single machine", Journal of the Operational Research Society 60,873-877, (2009).

[12] Koksalan M., Azizoglu M., Kondakci S., "Minimizing flow time and maximum earliness on a single machine ", IIE Transaction 30, 192-200 (1998).

[13] Kurz, M.E., and Canterbury, S., "Minimizing total flow time and maximum earliness on a single machine using multiple measures of fitness", Genetic and Evolutionary Computation Conference, 803-809 (2005).

[14] Lenstra J.K., Rinnooy Kan A.H.G., Bruck B., "Complexity of machine scheduling problems", Annals of discrete math. 1, 343-362, (1977).

[15] Mahnam M. and Moslehi G.,"A branch and bound algorithm for minimizing the sum of maximum earliness and tardiness with unequal release times",Eng Optimize 41, 521-536, (2009).

[16] Smith W.E., "Version Optimizer for Single Stages Production", Naval Res. Logistics Quarter 3, 59-66, (1956).

[17] Yanai S. and Fujie T., " On a dominance test for the single machine scheduling problem with release dates to minimize total flow time ", Journal of the Operation Research Society of Japan 47, No.2,96-111, (2004).