Minimizing total Completion Time and Maximum late Work Simultaneously

T. S. Abdul-Razaq and F. Sh. Fandi

Department of Mathematics, College of Science , Al_Mustansiriya University

Department of Mathematics, College of Education, for Pure Sciences, Al-Anbar university

Received in: 1 April 2012 Accepted in: 22 April 2012

Abstract

In this paper, the problem of scheduling jobs on one machine for a variety multicriteria are considered to minimize total completion time and maximum late work. A set of *n* independent jobs has to be scheduled on a single machine that is continuously available from time zero onwards and that can handle no more than one job at a time. Job i,(i=1,…,n) requires processing during a given positive uninterrupted time p_i, and its due date d_i.

For the bicriteria problems, some algorithms are proposed to find efficient (Pareto) solutions for simultaneous case. Also for the multicriteria problem we proposed general algorithms which gives efficient solutions within the efficient range.

Keyword:- schedule, multicriteria, single machine, bicriteria.

Introduction

The problem of scheduling a set N={1,…..,n} of n jobs on a single machine to minimize a variety of multicriteria may be stated as follows. Each job i, (i=1,….,n) is to be processed on a single machine which can handle only one job at a time. Associ- -ated with job i its processing time p_i and its due date d_i. All jobs are available for processing at time zero. A schedule σ=(1,…,n) specifies for each job when it is executed while observing the machine availability constraints and the schedule produce a completion time $c_i=\Sigma p_i$ for each job i,

In this paper, for the first time. We use the late work object in multicriteria scheduling problem. Given a schedule σ, the late work $v_i(\sigma)$ for job i,(i=1,…,n) which is amount of processing performed on job i after its due date d_i, is easy compute. We abbreviate $v_i(\sigma)$ to v_i for each job i, hence we have:

(1) If $v_i=0$, then job i is early with $c_i \le d_i$.

(2) If $0 < v_i < p_i$, then job i is partially early.

(3) If $v_i = p_i$, then job i is late with $c_i \ge d_i + p_i$.

This means that the late work for job i is given by

$$v_i=\begin{cases} 0 & \text{if} \quad c_i \le d_i, \quad i=1,\dots,n \\ c_i - d_i & \text{if} \quad d_i < c_i < d_i + p_i, \quad i=1,\dots,n \\ p_i & \text{if} \quad c_i \ge d_i + p_i, \quad i=1,\dots,n \end{cases}$$

Notation and Basic Concepts:

The following notation will be used in this paper:

n = number of jobs.

P_i = processing time of job i.

d_i = due date of job i.

c_i = completion time of job i.

Σc_i= the total completion times.

v_i= the late work penalty for job i.
V_{max}= Max $\{v_i\}$, the maximum late work.
L_i = lateness of job i, $L_i = c_i - d_i$.
T_i = tardiness of job i, T_i = Max $\{c_i - d_i ,0\}$.
L_{max} = Max$\{L_i\}$, maximum lateness of all job in a schedule.
T_{max} = Max$\{T_i\}$, maximum tardiness of all job in a schedule.
The cost f_i for each job i, (i=1,…,n) usually takes on the variable c_i. Common optimality criteria are usually in the form Σf_i and f_{max}= Max$\{f_i\}$.
Theorem(1)(Jackson 1955)[1]. The 1// Tmax problem is minimized by sequencing the jobs according to the earliest-due- date (EDD) rule, that is, in order of non-decreasing di. ■
Theorem (2)(Smith 1956)[2]. The 1//$\sum c_i$ problem is minimized by sequencing the jobs according to the shortest –processing-time (SPT) rule ,that is, in order of non-decreasing pi. ■
Theorem(3)(Lawler 1973)[3].The 1//fmax problem, fmax is minimized as follows: while there are unassigned jobs, assign the job that has minimum cost when scheduled in the last unassigned position in that position.■
Definition(1)[4]: A feasible schedule for a problem 1//Lex(f,g) is a schedule in which the primary criterion f is satisfied.
Definition(2)[4]: An optimal schedule for 1//Lex(f,g) is a feasible schedule that minimizes the secondary criterion g.
Definition(3):A feasible schedule σ^* is Pareto optimal, or non-dominated(efficient), with respect to the performance criteria f and g if there is no feasible schedule π such that both $f(\pi) \le f(\sigma^*)$ and $g(\pi)\le g(\sigma^*)$, where at least one of the inequalities is strict.
Definition(4)[5]: The function F(f,g) is said to be non-decreasing in both argument, if for any pair of outcome value (x,y) of the functions f and g, we have $F(x,y) \le F(x+A , y+B)$ for each pair of non-negative value A and B.
Theorem (4)[6]: If the composite objective function F(f,g) is non-decreasing in both argument, then there exists a Pareto optimal schedule that minimize F.
Definition(5)[7]: The term "optimize" in a multi-objective decision making problem refers to a solution around which there is no way of improving any objective without worsening at least one other objective.
Definition (6)[8]: A point $x = (x_1, . . . , x_k)$ is Pareto optimal within a given set S, if S does not contain any other point $y=(y_1, . . . , y_k)$ with $y_i \le x_i$ for i =1, . . . , k. Correspondingly, a schedule σ corresponds to a Pareto optimal point if there is no feasible schedule π with $f^k_{max}(\pi) \le f^k_{man}(\sigma)$ for k=1, . . . , K, where at least one of the inequalities is strict; in this case, we say that σ is not dominated.
Definition(7)[8]:hierarchical minimization: The performance criteria $f_1, . . . ,f_k$ 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.
Theorem(5)[9]: Consider the composite objective function F with $F(\pi)= F(f_1(\pi),\dots,f_k(\pi))$, where F is non- decreasing in all performance criteria f_k. There is a Pareto optimal schedule with respect to $f_1,\dots,f_k$ that minimizes the function F.

f_{max} is Regular Measure :

Now, we will consider the bicriteria that concerns the hierarchical and simultaneous minimization of bicriteria regular cost function for jobs i, which means that $f_i(c_i)$ does not decrease when c_i increased. Hoogeveen and Van de Valde [10] proved that $1//F(\Sigma c_i,f_{max})$

problem is solved in polynomial time, Van Wassenhove and Gelders [11] solved $1//F(\Sigma c_i, T_{max})$ problem, Emmons [12] presented the hierarchical problem $1//Lex(f_{max}, \Sigma c_i)$, where $f^*=Min\{f_{max}\}$ denotes the optimal solution value of the $1// f_{max}$ problem, which is solved in $O(n^2)$ time by Lawler algorithm[3]. Let $f_{max}= V_{max}$ in our study, since criterion V_{max} is a particular case of the function f_{max} .

Now, consider the following two problems:
$1//Lex(\Sigma c_i, V_{max})$ problem, and $1//Lex(V_{max}, \Sigma c_i)$ problem.

The $1//Lex(\Sigma c_i, V_{max})$ problem (P1):

This problem can be written as:

Min V_{max}

s.t

$c_{\sigma(i)} \geq p_{\sigma(i)}$ $\quad$ i=1,…,n.

$c_{\sigma(i)} = c_{\sigma(i-1)} + p_{\sigma(i)}$ $\quad$ i=2,…,n.

$$v_{\sigma(i)}=\begin{cases} 0 & \text{if } c_{\sigma(i)} \leq d_{\sigma(i)}, \quad i=1,\dots,n \\ c_{\sigma(i)}-d_{\sigma(i)} & \text{if } d_{\sigma(i)} < c_{\sigma(i)} < d_i + p_{\sigma(i)}, \quad i=1,\dots,n \\ p_{\sigma(i)} & \text{if } d_{\sigma(i)} + p_{\sigma(i)} \leq c_{\sigma(i)}, \quad i=1,\dots,n \end{cases}$$

$p_{\sigma(i)} \geq v_{\sigma(i)}$ $\quad$ i=1,…,n.

$v_{\sigma(i)} \geq 0$ $\quad$ i=1,…,n.

$\Sigma c_i = \Delta$, where $\Delta = \Sigma c_i(SPT)$.

It is clear that the SPT rule is feasible for problem(P1). Also the SPT rule is optimal for problem(P1) if there exist a tie (jobs with equal processing times) then order these jobs so that V_{max} is minimum.

The $1//Lex(V_{max}, \Sigma c_i)$ problem(P2):

This problem can be written as:

Min Σc_i

s.t

$c_{\sigma(i)} \geq p_{\sigma(i)}$ $\quad$ i=1,…,n.

$c_{\sigma(i)} = c_{\sigma(i-1)} + p_{\sigma(i)}$ $\quad$ i=2,…,n.

$$v_{\sigma(i)}=\begin{cases} 0 & \text{if } c_{\sigma(i)} \leq d_{\sigma(i)}, \quad i=1,\dots,n \\ c_{\sigma(i)}-d_{\sigma(i)} & \text{if } d_{\sigma(i)} < c_{\sigma(i)} < d_i + p_{\sigma(i)}, \quad i=1,\dots,n \\ p_{\sigma(i)} & \text{if } d_{\sigma(i)} + p_{\sigma(i)} \leq c_{\sigma(i)}, \quad i=1,\dots,n \end{cases}$$

$p_{\sigma(i)} \geq v_{\sigma(i)}$ $\quad$ i=1,…,n.

$v_{\sigma(i)} \geq 0$ $\quad$ i=1,…,n.

$V_{max} = \Delta$, where $\Delta = V_{max}(Lawler)$.

Algorithm(1) for finding an optimal value for problem(P2):

Step(0): Compute V_{max} by using (Lawler algorithm) and set $\sigma = (\varphi)$.
Step(1): Put $\Delta = V_{max}$(Lawler), $N=\{1,\dots,n\}$, $K=n$, $t=C_{max}=\Sigma p_i$.
Step(2): Calculate v_i, $\forall i \in N$ as follows:

$$v_i = \left\{\begin{array}{llll} 0 & \text{if} & c_i \le d_i\,, & i=1,\dots,n \\ c_i - d_i & \text{if} & d_i < c_i < d_i + p_i\,, & i=1,\dots,n \\ p_i & \text{if} & d_i + p_i \le c_i\,, & i=1,\dots,n \end{array}\right\}$$

Step(3): Find a job $j^* \in N$, such that
1.$v_{j^*} \le \Delta$.
2. $p_{j^*} \ge p_i$, $\forall j^*, i \in N$ and $v_i \le \Delta$.
Then assign job j^* in position K of $\sigma = (\sigma(K), \sigma)$.
Step(4): Set $t=t-p_{j^*}$, $N=N-\{j^*\}$, $K=K-1$, if $K>1$ go to step(2), otherwise go to step(5).
Step(5): Calculate $\Sigma c_{\sigma(i)}$ for the sequence jobs $\sigma = (\sigma(1),\dots,\sigma(n))$.
Step(6): Stop.

Proposition(1):

Algorithm(1) gives an optimal solution of $1//Lex(V_{max}, \Sigma c_i)$ problem.

Proof:

Since the problem $1//Lex(V_{max}, \Sigma c_i)$ can be written as:

Min Σc_i

S.t

$v_i \le V_{max}$ (Lawler) $\quad\quad i=1,\dots,n$

First notice that at each decision point the set of schedulable jobs is inspected (i.e. these jobs that can be scheduled without violating the maximum late work (V_{max}(Lawler))(which is obtained by Lawler algorithm)), any job j^* that is chosen in step (3) of algorithm(1) to be scheduled last. Second, if there exists a tie (more than one schedulable job i) then we choose the job j^* with largest p_{j^*} to be schedule last which minimizes Σc_i also. Hence, any schedule constructed by algorithm(1) is optimal.■

Algorithm(1) is a general algorithm can be used for solving problems of the following forms:

(1) $\quad 1//Lex(f_{max}, \Sigma c_i)$, where $f_{max} \in \{L_{max}, T_{max}, V_{max}, V^w_{max}\}$.

(2) $\quad 1//Lex(f_{max}, \Sigma c_i^2)$, where $f_{max} \in \{L_{max}, T_{max}, V_{max}, V^w_{max}\}$.

The $1//Lex(V^w_{max}, T_{max})$ problem(P3):

This problem can be written as:

Min T_{max}

s.t

$V^w_{max} = \Delta$, where $\Delta = V^w_{max}$(Lawler).

Algorithm(2) for finding an optimal value for problem(P3):

Step(0): Compute V^{w}_{max}=min{max{w_iv_i}}, by using (Lawler algorithm).

Step(1): Put $\Delta=V^{w}_{max}$(Lawler), N={1,…,n}, K=n, $t=C_{max}=\Sigma p_i$ and set $\sigma=(\varphi)$.

Step(2): Calculate w_iv_i, $\forall i\in N$ as follows:

$$w_iv_i=\begin{cases} 0 & \text{if} \quad c_i\le d_i, \quad i=1,\dots,n \\ w_i(c_i-d_i) & \text{if} \quad d_i<c_i<d_i+p_i, \quad i=1,\dots,n \\ w_ip_i & \text{if} \quad d_i+p_i\le c_i, \quad i=1,\dots,n \end{cases}$$

Step(3): Find a job $j^*\in N$, such that

1. $w_{j^*}v_{j^*}\le\Delta$.
2. $d_{j^*}\ge d_i$, $\forall j^*,i\in N$ and $w_iv_i\le\Delta$.

Then assign job j^* in position K of $\sigma=(\sigma(K),\sigma)$.

Step(4): Set $t=t-p_{j^*}$, N =N-{j*}, K=K-1, if K >1 go to step(2), otherwise go to step(5).

Step(5): Calculate $T_{max}(\sigma)$ for the sequence jobs $\sigma=(\sigma(1),\dots,\sigma(n))$.

Step(6): Stop.

Example(1): Consider the problem 1//Lex(V^{w}_{max}, T_{max}) with the following data:

i	1	2	3	4
p_i	3	1	2	4
d_i	4	6	1	2
w_i	2	3	2	5

By applying Lawler's algorithm (Step(0)) we get a schedule σ=(4,1,3,2) with V^{w}_{max}=10 and T_{max}=8.

Put Δ=10.

i	1	2	3	4	Job j^*
$t=C_{max}$	w_iv_i				
10	6	$\boxed{3}$	4	20	2
9	$\boxed{6}$	*	4	20	1
6	*	*	$\boxed{4}$	20	3
4	*	*	*	$\boxed{10}$	4

Hence, we get optimal schedule σ = (4,3,1,2) with V^{w}_{max} =10 and T_{max}=5.

7. Minimizing Total Completion Time and Maximum late work.

In this section we will try to find all efficient (Pareto optimal) solutions for 1//F(Σc_i, V_{max}) problem which is denoted by problem(P4).

7.1 The 1//F(V_{max}, Σc_i) problem(P4):

The 1//F(Σc_i, V_{max}) problem can be written as:

Min Σc_i

s.t.

$V_{max}\le\Delta$, where $\Delta\in[V_{max}(\text{Lawler}), V_{max}(\text{SPT})]$.

It is clear that the 1//F(Σc_i, V_{max}) problem originates from 1// Σc_i problem and 1//V_{max} problem, both problems are solvable in O(n log n) time. In order to find the set of Pareto optimal points, we solve the problem of minimizing Σc_i subject to $V_{max}\le\Delta$, where Δ corresponds to the V_{max} value of a Pareto optimal point.

Algorithm(3) for finding a set of all efficient solutions for problem(P4):

Step(0): Put $\Delta = \Sigma p_i$, $N=\{1,...,n\}$, $K=n$, $t=C_{max}=\Sigma p_i$ and set $\sigma = (\varphi)$.
Step(1): Calculate v_i, $\forall i \in N$ as follows:

$$v_i = \begin{cases} 0 & \text{if} \quad c_i \le d_i, \quad i=1,...,n \\ c_i - d_i & \text{if} \quad d_i < c_i < d_i + p_i, \quad i=1,...,n \\ p_i & \text{if} \quad d_i + p_i \le c_i, \quad i=1,...,n \end{cases}$$

Step(2): Find a job $j^* \in N$, such that
1.$v_{j^*} \le \Delta$.
2. $p_{j^*} > p_i$, $\forall j^*, i \in N$ and $v_i \le \Delta$, if $p_{j^*} = p_i$ choose the job with smallest v_{j^*}.
Then assign job j^* in position K of $\sigma = (\sigma(K), \sigma)$.
If no job with $v_{j^*} \le \Delta$ go to step(5).
Step(3): Set $t=t-p_{j^*}$, $N = N-\{j^*\}$, $K=K-1$, if $K>1$ go to step(1), otherwise go to step(4).
Step(4): Compute $V_{max}(\sigma)$ for the sequence jobs $\sigma = (\sigma(1), \ldots, \sigma(n))$ and $\Delta = V_{max} - 1$, set $t=C_{max}$, $N=\{1,...,n\}$, $K=n$, $\sigma = (\varphi)$, go to step(1).
Step(5): Stop.

Computational results for algorithm(3) for problem(P4):

8.1Test Problems:

Algorithm(3) was tested by coding it in matlab 7.9.0(R2009b) and implemented on Intel Core (TM) i3 CPU M380 @ 2.53 GHZ, with RAM 4.00 GB personal computer. Test problems were generated as follows: for each job j, an integer processing time p_j was generated from the uniform distribution [1,10]. Also, for each job j, an integer due date is generated from the distribution$[p_i, \Sigma p_i]$. Results for 10,15, 30 and 50 job problems are given in Table (1) Because of page limitations, only five examples are given for each problem size. Table(1) should be read as follows:

NEP = the actual number of efficient points for the problem at hand. Remember that the procedure finds all these points.

MNEP = maximum possible number of efficient points for the problem at hand. Since an efficient point could exist for each value of the maximum late work between V(SPT*) and V(Lawler), this maximum possible number equals $V_{max}(SPT^*) - V_{max}(Lawler) + 1$.

From our computational results we conclude that:
(a) The number of efficient points is usually small.
Therefore, the decision maker should be able to make his choice.
(b) We can find all efficient points. This allows for a justified choice of a sequence that takes both objective functions into account and that is not dominated by any other sequence. Very attractive efficient points are found for all the test problems. In most cases we discovered efficient point that were surprisingly close to the optimum value for both objectives.

Remark 1:

If we replace the criterion V_{max} in problem(P4) by the criterion $f_{max} \in \{L_{max}, T_{max}, V^{w}_{max}, T^{w}_{max}\}$ then the efficient Algorithm(3) above can be used for their solutions. Also for the $1//F(\Sigma c_i, T_{max})$ problem, which is solved by Vanwassenhove and Gelder [11]. This means that algorithm(3) is a general one and simple that can solve problems of form $1//F(\Sigma c_i, f_{max})$.

Analysis of the Three Criteria.

First, we present the main multicriteria scheduling results that have appeared in the literature. Hoogeveen[8] solves the problem $1//F(f_{max}, g_{max})$ and the problem $1//F(f_{max}, g_{max}, h_{max})$. He presented a polynomial algorithm for both problems and he showed that these can be used if precedence constraints exist between the jobs or if all penalty functions are non-decreasing in the job completion times. Hoogeveen[13] solves the general problem $1//F(f^1_{max},\dots, f^k_{max})$, k finite integer number and each one of these penalty functions is assumed to be non-decreasing in the job completion time.

There are two methods for dealing with conflicting criteria, the first one is hierarchical minimization of the performance criteria $f_1, f_2,\dots, f_k$ which are indexed in order of decreasing importance. The second method is simultaneous minimization. The criteria are aggregated into a single composite objective function $F(f_1, f_2,\dots, f_k)$, which minimized.

In our study we are looking for algorithms, that can be used for solving multicriteria scheduling problem, which is to find the efficient solutions or at least approximation to it. Hence we search for feasible solutions yielding the best compromise among objectives that constitutes a so called efficient solutions set.

The $1//F(\sum c_i, T_{max}, V_{max})$ problem(P5):

In this section, we will try to find efficient (Pareto optimal) solutions for $1//F(\Sigma c_i ,T_{max}, V_{max})$.

This problem can be defined as:

$$\left.\begin{array}{ll}
\text{Min} \begin{cases}\sum c_i \\ T_{max} \\ V_{max}\end{cases} & \\
\text{s.t.} & \\
c_i \geq p_i & i=1,\dots,n. \\
c_i=c_{(i-1)}+ p_i & i=2,\dots,n. \\
v_i\leq p_i & i=1,\dots,n. \\
v_i\leq T_i & i=1,\dots,n. \\
T_i\geq c_i - d_i & i=1,\dots,n. \\
T_i \geq 0 & i=1,\dots,n.
\end{array}\right\} \quad \dots\dots \text{P(5)}$$

It is clear that this problem is difficult to solve, we will present later local search algorithm to find near optimal solution. This problem has the following special cases.

(1) The $1//Lex(\sum c_i, T_{max}, V_{max})$ problem.
(2) The $1//Lex(\sum c_i, V_{max}, T_{max})$ problem.
(3) The $1//Lex(T_{max}, \sum c_i ,V_{max})$ problem.
(4) The $1//Lex(T_{max}, V_{max}, \sum c_i)$ problem.
(5) The $1//Lex(V_{max}, \sum c_i ,T_{max})$ problem.
(6) The $1//Lex(V_{max}, T_{max}, \sum c_i)$ problem.

Algorithm(4) for finding efficient solutions for problem (P5):

Step(0): Put $\Delta = \sum p_i$, $N=\{1,\ldots..,n\}$, $K=n$, $t=C_{max}=\sum p_i$, and $\sigma = (\varphi)$.

Step(1): Calculate T_i, v_i, $\forall i \in N$ as follows:

$T_i = \text{Max}\{c_i - d_i, 0\}$and $v_i=\min\{T_i, p_i\}$.

Step(2): Find a job $j^* \in N$,such that

$T_{j^*} \le \Delta$ and $p_{j^*} > p_i$, $\forall j^*, i \in N$ and $T_i \le \Delta$, if $p_{j^*}=p_i$ choose the job with smallest T_{j^*} and if $T_{j^*}=T_i$ choose the job with smallest v_{j^*}. Then assign job j^* in position K of $\sigma = (\sigma(K), \sigma)$, If no job with $T_{j^*} \le \Delta$ go to step(5).

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

Step(4): Compute $\Sigma c_{\sigma(i)}$,$T_{max}(\sigma)$ and $V_{max}(\sigma)$ for the sequence jobs $\sigma=(\sigma(1),\ldots,\sigma(n))$, and set $\Delta = T_{max}(\sigma)-1$, set $t=C_{max}$, $N=\{1,\ldots..,n\}$, $K=n$, $\sigma = (\varphi)$, go to step(1).

Step(5): Put $\Delta = V_{max}(\sigma)-1$, $N=\{1,\ldots,n\}$, $K=n$, $t=C_{max}=\sum p_i$, and $\sigma = (\varphi)$.

Step(6): Calculate v_i, $\forall i \in N$ as follows:

$$v_i = \begin{cases} 0 & \text{if} \quad c_i \le d_i, & i=1,\ldots,n \\ c_i - d_i & \text{if} \quad d_i < c_i < d_i + p_i, & i=1,\ldots,n \\ p_i & \text{if} \quad d_i + p_i \le c_i, & i=1,\ldots,n \end{cases}$$

Step(7): Find a job $j^* \in N$, such that

$v_{j^*} \le \Delta$ and $p_{j^*} > p_i$, $\forall j^*, i \in N$ and $v_i \le \Delta$, if $p_{j^*}=p_i$ choose the job with smallest v_{j^*}. Then assign job j^* in position K of $\sigma = (\sigma(K), \sigma)$. If no job with $v_{j^*} \le \Delta$ go to step(10).

Step(8): Set $t=t-p_{j^*}$, $N=N-\{j^*\}$, $K=K-1$, if $K>1$ go to step(6), otherwise go to step(9)

Step(9): Calculate $\Sigma c_{\sigma(i)}$, $T_{max}(\sigma)$ and $V_{max}(\sigma)$ for the sequence jobs $\sigma=(\sigma(1),\ldots,\sigma(n))$, go to step(5).

Step(10): Stop.

The 1//F($\Sigma w_i c_i$, T_{max}, V_{max}) problem(P6):

In this section, we will try to find efficient (Pareto optimal) solutions for 1//F($\Sigma w_i c_i$, T_{max}, V_{max}) problem.

This problem can be defined as:

$$\left.\begin{array}{ll} \text{Min} \begin{cases} \sum w_i c_i \\ T_{max} \\ V_{max} \end{cases} & \\ \text{s.t.} & \\ c_i \ge p_i & i=1,\ldots,n. \\ c_i = c_{(i-1)} + p_i & i=2,\ldots,n. \\ v_i \le p_i & i=1,\ldots,n. \\ v_i \le T_i & i=1,\ldots,n. \\ T_i \ge c_i - d_i & i=1,\ldots,n. \\ T_i \ge 0 & i=1,\ldots,n. \end{array}\right\} \quad \ldots.. \text{P(6)}$$

Algorithm(5) for finding efficient solutions for problem(P6):

Step(0): Put $\Delta=\sum p_i$, $N=\{1,\ldots,n\}$, $K=n$, $t=C_{max}=\sum p_i$, and $\sigma=(\varphi)$.

Step(1): Calculate T_i, v_i, $\forall i\in N$ as follows:

$T_i=\text{Max}\{c_i-d_i, 0\}$ and $v_i=\min\{T_i, p_i\}$.

Step(2): Find a job $j^*\in N$, such that

$T_{j^*}\le\Delta$ and $p_{j^*}/w_{j^*}>p_i/w_i$, $\forall j^*, i\in N$ and $T_i\le\Delta$, if $p_{j^*}/w_{j^*}=p_i/w_i$ choose the job with smallest T_{j^*} and if $T_{j^*}=T_i$ choose the job with smallest v_{j^*}. Then assign job j^* in position K of $\sigma=(\sigma(K),\sigma)$. If no job with $T_{j^*}\le\Delta$ go to step(5).

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

Step(4): Compute $\Sigma w_{\sigma(i)}c_{\sigma(i)}$, $T_{max}(\sigma)$ and $V_{max}(\sigma)$ for the sequence jobs $\sigma=(\sigma(1),\ldots,\sigma(n))$, and set $\Delta=T_{max}(\sigma)-1$, set $t=C_{max}=\sum p_i$, $N=\{1,\ldots,n\}$, $K=n$, $\sigma=(\varphi)$ and go to step(1).

Step(5): Put $\Delta=V_{max}(\sigma)-1$, $N=\{1,\ldots,n\}$, $K=n$, $t=C_{max}$, and $\sigma=(\varphi)$.

Step(6): Calculate v_i, $\forall i\in N$ as follows:

$$v_i=\begin{cases}0 & \text{if} \quad c_i\le d_i, \quad i=1,\ldots,n\\ c_i-d_i & \text{if} \quad d_i<c_i<d_i+p_i, \quad i=1,\ldots,n\\ p_i & \text{if} \quad d_i+p_i\le c_i, \quad i=1,\ldots,n\end{cases}$$

Step(7): Find a job $j^*\in N$, such that

$v_{j^*}\le\Delta$ and $p_{j^*}/w_{j^*}>p_i/w_i$, $\forall j^*, i\in N$ and $v_i\le\Delta$, if $p_{j^*}/w_{j^*}=p_i/w_i$ choose the job with smallest v_{j^*}. Then assign job j^* in position K of $\sigma=(\sigma(K),\sigma)$. If no job with $v_{j^*}\le\Delta$ go to step(10).

Step(8): Set $t=t-p_{j^*}$, $N=N-\{j^*\}$, $K=K-1$, if $K>1$ go to step(6), otherwise go to step(9).

Step(9): Calculate $\Sigma w_{\sigma(i)}c_{\sigma(i)}$, $T_{max}(\sigma)$ and $V_{max}(\sigma)$ for the sequence jobs $\sigma=(\sigma(1),\ldots,\sigma(n))$, go to step(5).

Step(10): Stop.

Example(2): Consider the problem $1//F(\Sigma w_ic_i, T_{max}, V_{max})$ with the following data:

i	**1**	**2**	**3**	**4**	**5**
p_i	10	3	9	1	4
d_i	20	14	25	29	16
w_i	4	1	8	5	2
p_i/w_i	5/2	3	9/8	1/5	2

Hence, efficient sequences for the three-criterion problem(P6) of minimizing Σw_ic_i, T_{max} and V_{max}.

SEQUENCES	Σw_ic_i	T_{max}	V_{max}
SWPT=(4,3,5,1,2)	236	13	4
(4,3,5,2,1)	238	7	7
(4,5,1,2,3)	309	4	3
(4,5,2,1,3)	311	2	2
(5,2,1,3,4)	426	1	1

Conclusions

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

We proposed algorithms, which give all efficient solutions within the efficient range for the problems $1//F(\Sigma f_i, f_{max})$, $1//F(\Sigma f_i, f_{max}, g_{max})$, where $\Sigma f_i=\Sigma c_i$ and $f_{max}, g_{max}\in\{L_{max}, T_{max}, V_{max}\}$.

References

1. Jackson J.R.(1955)."Scheduling a production line to minimize maximum tardiness" Res. Report 43 management science, Res. Project, University of California, Loss Angles, CA,
2. Smith W.E.(1956)."Various optimizers for single stage production", Naval Research Logistics Quarterly 3/1, 59-66.
3. Lawler E.L.(1973)."Optimal sequencing of a single machine subject to precedence constraints", Management Science 19/5, 544-546.
4. French S. (1982). "Sequencing and Scheduling : An introduction to the mathematical of the job shop ", John wiley and sons, New York.
5. Hoogeveen J.A. (2005)." Invited Review Multi-criteria scheduling ", European Journal of Operational Research 167, 592–623.
6. Fry TD, Armstrong RD. Lewis, H. (1989). "A framework for single machines Multiple objective sequencing research", OMEGA; 17 (6): 595-607.
7. Landa silva E. K. and Burke S. Petrovic ,"An Introduction to Multiobjective Metaheuristics for Scheduling and Time tabling "
8. Hoogeveen, J.A. (1996).” Single machine scheduling to minimize a function of two or three maximum cost criteria “ , Journal of Algorithms, 21,415-433.
9. Bagchi T.P., (1999). "Multiobjective Scheduling by Genetic Algorithms", Kluwer Acadimic publisher.
10. Hoogeveen, J.A. and van de Velde, S.L. (1995). “ Minimizing total completion time and maximum cost simultaneously is solvable in polynomial time”, Operations Research Letters 17, 205–208.
11. Van Wassenhove, L.N. and Gelders. F. (1980). "Solving a bicriterion scheduling problem", European Journal of Operational Research 4/1, 42-48.
12. Emmons, H. (1975). "A note on a scheduling problem with dual criteria", Naval Res. Logis. Quarterly, 22, 615-516 .
13. Hoogeveen, J.A. (1991). ” Single machine scheduling to minimize a function of K maximum cost criteria “ ,CWI,BS-R9113.

Table(1): Computational results

Problem Size	**Problem Number**	**NEP**	**MNEP**	**Efficient Points**
10	1	2	2	(273, 8), (262, 9)
	2	2	2	(208, 8), (189, 10)
	3	4	4	(290, 7), (227, 8), (211, 9), (202, 10)
	4	1	1	(194, 10)
	5	3	4	(353, 7), (317, 8), (293, 10)
15	1	3	3	(388, 8), (354, 9), (337, 10)
	2	2	2	(657, 9), (571, 10)
	3	3	4	(752, 7), (613, 9), (511, 10)
	4	2	2	(599, 9), (501, 10)
	5	2	2	(618, 8), (474, 9)
30	1	4	4	(2737, 7), (2418, 8), (2283, 9), (2206, 10)
	2	2	2	(1833, 9), (1755, 10)
	3	4	4	(2275, 7), (2165, 8), (1861, 9), (1749, 10)
	4	6	6	(2599, 5), (2499, 6), (2384, 7), (2283, 8), (2226, 9), (2214, 10)
	5	1	1	(1982,10)
50	1	5	5	(6192, 6), (5869, 7), (5477, 8), (5110, 9), (4863, 10)
	2	2	2	(6223, 9), (5518)
	3	4	4	(5945, 7), (5794, 8), (5437, 9), (5228, 10)
	4	6	6	(5715, 4), (5276, 5), (5105, 6), (4810, 7), (4652, 8), (4641, 9)
	5	2	2	(5432, 10), (6607, 9)

مجموع أوقات الإتمام مع أعظم عمل تأخير

طارق صالح عبد الرزاق و فراس شاكر فندي
قسم الرياضيات، كلية العلوم، الجامعة المستنصرية.
قسم الرياضيات، كلية التربية للعلوم الصرفة، جامعة الأنبار

استلم البحث في: 1 نيسان 2012 قبل البحث: 22 نيسان 2012

الخلاصة

في هذا البحث المسألة لجدولة الوظائف على الماكنة الواحدة للمعايير المتنوعة درست المسألة لتصغير مجموع أوقات الإتمام مع أعظم عمل تأخير. المجموعة n من الوظائف المستقلة التي يمكن جدولتها على ماكنة واحدة والتي تتطلب معالجة باستمرار من الوقت صفر 0 وصاعدا ولذلك لا يمكن ان تعالج اكثر من وظيفة واحد في وقت واحد. الوظيفة i تتطلب وقت معالجة موجب معطى مستمر p_i ، الموعد المثالي لإكمال الوظيفة d_i .

لمسائل جدولة المعايير الثنائية ، بعض الخوارزميات اقترحت لإيجاد الحلول الكفوءة في حالة المسائل التي معايرها متساوية الأهمية. و لمسائل المعايير المتعددة ايضا، فقد اقترحنا خوارزميات عامة لإعطاء الحلول الكفوءة ضمن المدى الكفوء.

الكلمات المفتاحية:- الجدولة، المعايير المتعددة، الماكنة الواحدة، المعايير الثنائية.