MULTIPLE OBJECTIVE FUNCTION ON A SINGLE MACHINE SCHEDULING PROBLEM

Abdul Razaq T. S.* Al Saidy S. K.* Al Zuwaini M. K.**

* Dept. of Math\ College of Science\ University of Al Mustansiriyah

** Dept. of Math\ College of Computer Science and Mathematic\ Thi qar University

ABSTRACT

We consider a single machine scheduling problem to minimize a multiple objective function; sum of earliness, tardiness and completion time. As this problem is complete NP-hard we propose a branch and bound algorithm to obtain an optimal solution. The implementation of optimizing algorithms dose seen to be promising but it need longer time. Thus we tackle the problem with local search methods: descent method, simulated annealing and threshold acceptance. The performance of these heuristic methods is evaluated on a large set of test problems, and the results are also compared with these obtained by genetic algorithm and hybrid method which is combining the simulated annealing with the genetic algorithm. The best results are obtained with the hybrid method. We solved the problem optimality with up to 35 jobs and approximately with up to 150000 jobs.

Keywords: Scheduling, single machine, local search, decent, simulated annealing, threshold accepting, genetic, hybrid

1. Introduction

In this paper, we consider the problem of scheduling jobs on a single machine to minimize sum of earliness, tardiness and completion time with constraint that the penalty rates are equals for each job, which is in contrast with most works on *E/T/C* models.

The *E/T* problem with no idle time, however, has been considered by several authors, and both exact and heuristic approaches have been proposed. Among the exact approaches, branch and bound algorithms were presented by Abdul-Razaq and Potts (1988) [1], the lower bounding procedure of Abdul-Razaq and Potts was based on the sub gradient optimization approach and the dynamic programming state-space relaxation technique.

For problem $1/d_j = d/\sum(E_j + T_j)$ with a restricted common due date, Hall, et al. (1991) [10] and Hoogeveen & Van de Velde (1991) [11] establish NP-hardness. Kanet (1981) [13] derives properties of an optimal solution for the unrestricted common due date version of this problem. Abdul-Razaq & Mahmood (2001) [2] found optimal and near optimal solution where jobs divided into F families each family f, ($f=1,\ldots, F$) contains n_f jobs.

Among the heuristics, Enumerative Algorithms and local search, Ow and Morten (1989) [18], developed several dispatch rules and a filtered beam search procedure a neighborhood search algorithm was also presented by George Li (1996) [9]. Celso, et al. (2005) [6], proposed a tabu search-based heuristic and a genetic algorithm which exploit specific properties of the optimal solution for problem $1/d_j = d/\sum(E_j + T_j)$. Chichang (2005) [8], proposed a genetic algorithm with sub-indexed partitioning genes (*GASP*) to allow more flexible job assignments to machines for a problem $P//E/T$. Jan & Frank (2000) [13], derived some structural properties useful in connection with the search for an approximate solution for a problem $Pm/d_j = d, r_j/E/T$, Martin & Dirk (2003) [14], considered a problem $1/d_j = d/E/T$; they applied meta-heuristics, namely evolutionary strategies, simulated annealing and threshold accepting. Hall, et al. (1991) [10], show that a problem $1/d_j = d/E/T$ is NP-hard in the ordinary sense, and they proposed an O ($n\sum_j P_j$) pseudo polynomial dynamic programming algorithm. In this paper we present a branch and bound algorithm based on the lower bounds obtained from dynamic programming state space relaxation and relaxation of the objective function for the general problem, we find an optimal solution for special cases. Further, we use a local search, genetic algorithm and we propose hybrid method to find near optimal solutions.

2. Formulation of the Problem:

The general problem of scheduling jobs on a single machine to minimize the total cost that can be state as follows. A set of *n* independent jobs *N*= {*1, 2,…, n*} are available for processing at time zero, has to be scheduled without preemptions 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 unforced machine idle time is not allowed. Each job *j*, $j\in N$ requires a processing time p_j and should ideally be completed on it is due date d_j. For any given schedule (*1, 2,.., n*), the completion time, the earliness and the tardiness of job *j* can be respectively defined as:

$$C_j = \sum_{i=1}^{j} p_i \ , \quad E_j = \max\{d_j - C_j, 0\} \quad \text{and} \quad T_j = \max\{C_j - d_j, 0\}$$

The objectives is then to find the schedule that minimizes the multiple objective function (*MOF*) defined by $1//\sum(E_j + T_j + C_j)$. It is clear that our model differs from the other models in that we consider a more general and realistic problem dealing with arbitrary due dates. The inclusion of both earliness and tardiness cost in the objective function is compatible with the philosophy of just in time production, which emphasizes producing goods only when they are needed. As a generalization of weighted tardiness scheduling, the problem is strongly NP-hard. To the best of our knowledge, we know of no published work on penalties $E/T/C$ problem. Our scheduling problem can be state more precisely as follows:

Given a schedule (*1, 2,…, n*), then for each job *j* we can calculate C_j, E_j and T_j. The objective is to find a schedule *δ* = (*δ* (*1*),…, *δ*(*n*)) of the jobs that minimize the total cost $Z(\delta)$ where

$$Z(\delta) = (\mathrm{E}_{\delta(\mathrm{i})} + T_{\delta(\mathrm{i})} + C_{\delta(\mathrm{i})})$$

Let *S* be a set

$$Min\{\mathrm{Z}\ (\delta)\};\ \mathrm{Min}\sum(\mathrm{E}_{\delta(\mathrm{i})} + T_{\delta(\mathrm{i})} + C_{\delta(\mathrm{i})})$$

Subject to:

$$\left.\begin{array}{ll} P_{\delta(j)} > 0, & \mathrm{j}=1,..., \\ E_{\delta(j)} \geq 0, & \mathrm{j}=1,...,\mathrm{n} \\ T_{\delta(j)} \geq 0, & \mathrm{j}=1,...,\mathrm{n} \\ \mathrm{T}_{\delta(\mathrm{j})} - \mathrm{E}_{\delta(\mathrm{j})} \geq \mathrm{C}_{\delta(\mathrm{j})} - \mathrm{d}_{\delta(\mathrm{j})}, & \mathrm{j}=1,...,\mathrm{n} \\ \mathrm{T}_{\delta(\mathrm{j})} + \mathrm{E}_{\delta(\mathrm{j})} = \left|\mathrm{C}_{\delta(\mathrm{j})} - \mathrm{d}_{\delta(\mathrm{j})}\right|, & \mathrm{j}=1,...,\mathrm{n} \end{array}\right\} \quad (P)$$

of all

schedules, $|S| = n!$, then our problem can formally be stated as:

where $\delta(j)$ denotes the position of job *j* in the ordering δ.

3. Special cases

It is clear that the problem (P) is NP-hard since the problem E/T is NP- hard [14]. The problem (p) is considered by Mohammed (2005) [16], he used a local search to find the near optimal solution for problem (p) and he solved the problem with up to 150 jobs. A special case of problem (p), if all jobs have the common due date (i.e. $d_j = d,\ j = 1,2,...,n$), then, the resulting problem denoted by $1/d_j = d/\sum(E_j + T_j + C_j)$ has an optimal solution given by the following result:

Theorem (1)

The *SPT* rule gives an optimal solution for problem $1/d_j = d/\sum(E_j + T_j + C_j)$.

Proof:

Let $\pi = \delta ij\delta'$ be a sequence, where δ and δ' be a subsequences and *i*, *j* two jobs with $p_i \geq p_j$ and let C be denoted to completion time of jobs of subsequence δ, and *d* the common due date for all jobs, then:

First: If $C + p_i > d$ (i .e. job *i* is late), then,

$$E_i + T_i + C_i = C + p_i - d + C + p_i = 2C + 2p_i - d .$$

Since *i* is late then *j* is late

$$E_j + T_j + C_j = C + p_i + p_j - d + C + p_i + p_j = 2C + 2p_j + 2p_i - d$$

For the sequence π

$$\sum_{i,j\in\pi}(E_i + T_i + C_i) = 4C + 4p_i + 2p_j - 2d$$

Now let $\pi' = \delta ji\delta'$ be a new sequence obtained from π by interchange *i* and *j*, then,

$$E_j + T_j + C_j = \begin{cases} 2C + 2p_j - d & , \ if \quad C + p_j > d \\ d & , \ if \quad C + p_j \leq d \end{cases}$$

$E_i + T_i + C_i = 2C + 2p_j + 2p_i - d$ (Since job *i* is late).

For the sequence π'

$$\sum_{i,j\in\pi'}(E_i + T_i + C_i) = \begin{cases} 4C + 4p_j + 2p_i - 2d & , \ if \quad C + p_j > d \\ 2C + 2p_j + 2p_i & , \ if \quad C + p_j \leq d \end{cases}$$

$$\sum_{i\in\pi}(E_i + T_i + C_i) - \sum_{i\in\pi'}(E_i + T_i + C_i) = \begin{cases} 2p_i - 2p_j \geq 0 & , \ if \ C + p_j > d \\ 2C + 2p_i - 2d \geq 0 & , \ if \ C + p_j \leq d \end{cases} \qquad (1)$$

Second. If $C + p_i \leq d$ (i.e. job *i* is early).

For sequence π , we get $E_i + T_i + C_i = d$

$$E_j + T_j + C_j = \begin{cases} 2C + 2p_i + 2p_j - d & , \quad if \quad C + p_i + p_j > d \\ d & , \quad if \quad C + p_i + p_j \le d \end{cases}$$

$$\sum_{i,j\in\pi}(E_i + T_i + C_i) = \begin{cases} 2C + 2p_i + 2p_j & , \quad if \quad C + p_i + p_j > d \\ 2d & , \quad if \quad C + p_i + p_j \le d \end{cases}$$

for the sequence π', if i is early then j is early

$$E_j + T_j + C_j = d$$

$$E_i + T_i + C_i = \begin{cases} 2C + 2P_i + 2P_j - d, & if \ C + p_i + p_j > d \\ d, & if \ C + p_i + p_j \le d \end{cases}$$

$$\sum_{i,j\in\pi'}(E_i + T_i + C_i) = \begin{cases} 2C + 2P_i + 2P_j, & if \ C + p_i + p_j > d \\ 2d, & if \ C + p_i + p_j \le d \end{cases}$$

$$\sum_{i\in\pi}(E_i + T_i + C_i) - \sum_{i\in\pi'}(E_i + T_i + C_i) = 0 \qquad (2)$$

then from (1) and (2) we obtain that *SPT* rule gives an optimal solution for $1/d_i = d/\sum(E_i + T_i + C_i)$ problem.

Proposition (1)

If the jobs ordered as *SPT* rule such that $C_i > d_i$, $\forall i$ in *SPT* , then *SPT* rule gives optimal solution for $1//\sum(E_i + T_i + C_i)$ problem.

Proof:

Let π be a schedule orderly according to *SPT* rule such that $C_i > d_i$, $i = 1,2,...,n$. Then for each job i ($i \in \pi$) , i is late job and,

$$Min\ \{\sum (E_i + T_i + C_i)\} = Min\ \{2\sum C_i - \sum d_i\}$$

but $\sum_{i\in\pi} d_i$ is a constant, then $Min\{2\sum C_i - \sum d_i\}$ depend on $\sum C_i$ only. Smith [20] shows that *SPT* rule gives an optimal solution for $1//\sum C_i$ problem. So π is optimal schedule for $1//\sum(E_i + T_i + C_i)$ problem if $C_i > d_i$, $\forall i \in \pi$.

Proposition (2)

If there is a schedule π , such that $C_i < d_i$, $\forall i \in \pi$, then schedule π is optimal solution for $1//\sum(E_i + T_i + C_i)$ problem with value $\sum_{i\in\pi} d_i$.

Proof:

Since for each job i, $C_i < d_i$, $i \in \pi$ then each job i is an early job and $\sum(E_i + T_i + C_i) = \sum d_i$. Since $\sum_{i\in\pi} d_i$ is constant. Then π is optimal solution for $1//\sum(E_i + T_i + C_i)$ problem with value $\sum_{i=1}^{n} d_i$.

Proposition (3)

If there is a schedule π orderly as SPT rule and $C_i = d_i$, $\forall i \in \pi$ then schedule π gives an optimal solution for $1//\sum(E_i + T_i + C_i)$ problem.

Proof:

Since $C_i = d_i$, $\forall i \in \pi$ then $\sum_{i\in\pi} T_i = \sum_{i\in\pi} E_i = 0$ and the problem $1//\sum(E_i + T_i + C_i)$ reduce to $1//\sum C_i$ problem. Smith [20] shows that *SPT* rule is an optimal solution for $1//\sum C_i$ problem. Then, π is optimal solution for $1//\sum(E_i + T_i + C_i)$ problem if $C_i = d_i$, $\forall i \in \pi$.

Proposition (4)

For the $1//\sum(E_i + T_i + C_i)$ problem, the *SPT* schedule (1, 2,…,*n*) is optimal if $p_1 = d_1$ and $p_j = d_j - d_{j-1}$, $j = 2,3,\dots,n$.

Proof:

Suppose $\delta = (1,2,\dots,n)$ be a schedule orderly as *SPT* rule and satisfies the condition of the proposition. Since $p_1 = d_1$, hence $C_1 = p_1 = d_1$, $p_2 = d_2 - d_1 = d_2 - p_1$ and $C_2 = p_1 + p_2 = p_1 + d_2 - p_1 = d_2$.

Thus, we concluded that $C_j = d_j$ for $j = 1,2,\dots n$, by using proposition (3) we get that δ is an optimal for $1//\sum(E_i + T_i + C_i)$ problem.

Theorem (2)

A schedule π obtained by ordering its jobs in non-decreasing order of due date (EDD rule) is optimal for the $1//\sum(E_i + T_i + C_i)$ problem if $T_i \le P_i$ for all job *i* sequenced in π.

Proof:

For each job i, $T_i \le T_{\max} = \sum_{j=1}^{k} p_j - d_k \le p_k$, for some job *k*. Let $\pi = (1, 2,\dots, n)$ be a schedule obtained by *EDD* rule.

Suppose $ER = \{ j \in \pi : C_j \le d_j \}$ and $LT = \{ j \in \pi : C_j > d_j \}$. Now consider

$$Min\{\sum_{i\in\pi}(E_i+T_i+C_i)\}=Min\{\sum_{i\in ER}(E_i+C_i)\}+Min\{\sum_{i\in LT}(T_i+C_i)\}$$
$$=Min\sum_{i\in ER}d_i+Min\sum_{i\in LT}(T_i+C_i-d_i+d_i)$$
$$=Min\{\sum_{i\in\pi}d_i+2\sum_{i\in LT}T_i\}$$

The first term in the R.H.S is constant and the second term ($\sum_{i\in LT}T_i$) is minimized by *EDD* rule since $T_i \le p_i$ for each job i, $i\in\pi$ [19]. Hence the *EDD* rule is optimal for $1//\sum(E_i+T_i+C_i)$ problem if $T_i \le p_i$.

Theorem 3.

If the jobs sequencing according to *SPT* rule such that $d_i + p_i \le C_{i+1}$, then *SPT* rule gives an optimal solution for $1//\sum_{i=1}^{n}(E_i + T_i + C_i)$ problem.

Proof:

Let π be a *SPT* schedule with $d_i + p_i \le C_{i+1}$ for each job i in π . Thus,

$$d_i + p_i \le C_i + p_{i+1}\ ,\ i=1, ..., n\text{-}1$$
$$p_i - p_{i+1} \le C_i - d_i$$

Since $i\in\pi$ (π is *SPT* schedule), then $p_i \le p_{i+1}$ and there are three cases for $C_i - d_i$ for each i in π .

Case (i). If $C_i - d_i < 0$, then $T_i = 0,\ \forall i\in\pi$ and $\sum_{i=1}^{n}(E_i + T_i + C_i) = \sum d_i$ (Constant)

Case (ii). If $C_i - d_i = 0$, then $E_i = T_i = 0,\ \forall i\in\pi$ and

$\sum_{i=1}^{n}(E_i + T_i + C_i) = \sum C_i$,

Smith [20] shows that *SPT* rule is optimal for $1//\sum C_i$ problem.

Case (iii). If $C_i - d_i > 0$, then $E_i = 0,\ \forall i\in\pi$ and

$\sum_{i=1}^{n}(E_i + T_i + C_i) = 2\sum C_i - \sum d_i$,

Since $\sum d_i$ is constant, then *SPT* gives an optimal solution.

Hence for the three cases, *SPT* rule gives an optimal solution for $1//\sum(E_i + T_i + C_i)$ Problem if $d_i + P_i \le C_{i+1}$..

4. Lower Bounds

We propose more than one lower bound for problems (*P*).

4.1 Dynamic Programming State-Space Relaxation (DPSSR).

Christofides et al. (1981) [17] developed the dynamic programming state space relaxation (*DPSSR*) method for various routing problem. Abdul- Razaq and Potts (1988) [1] for the first time in scheduling using *DPSSR* method to obtained a lower bound for problem

E/T. In this method, a relaxed problem is obtained from a dynamic programming formulation by mapping the original state-space onto a smaller state-space and performing the recursion on this smaller state-space. The procedure used in this section to compute a lower bound is based on Abdul-Razaq and Potts [1].

Let $N = \{1, 2, \ldots, n\}$ be set of n jobs, let $S \subseteq N$ be an arbitrary subset of jobs. Let $f(S)$ is the minimum cost of scheduling jobs of S in the first initial $|S|$ positions. The object is to find $f(N)$ from the following recursion equations

$$f(S) = \underset{i \in S}{Min} \left\{ f(S-\{i\}) + g_i \left(\sum_{j \in S} p_j \right) \right\} \qquad (3)$$

where $f(\varphi) = 0$, $g_i\left(\sum_{j \in S} p_j\right)$ is the cost of completed job i at time $t = \sum_{j \in S} p_j$. A state S is mapped onto a state $\sum_{i \in S} p_i$. (We assume that $T = \sum_{i \in N} p_i < 2^n$ to ensure that there are fewer states in the relaxed problem than in the original problem:

If $T \geq 2^n$, it is more efficient to solve the original problem). The relaxed problem is solved by computing $f_0(T)$ from the recursion equations

$$f_0(t) = \min_{i \in N} \{ f_0(t - p_i) + g_i(t) \} \qquad (4)$$

That are initialized by setting $f_o(t) = \infty$ for $t<0$ and $f_o(0) = 0$ where $g_i(t) = E_i + T_i + C_i$, the cost of scheduling job i to be completed at time t is $g_i(t)$.

Theorem 4 : (Abdul Razaq 1987)[3]

If $f(N)$ is obtained from (3) and if $f_o(T)$ is obtained from (4), then, $f_o(T) \leq f(N)$.

Hence $LB1 = f_o(T)$ is a lower bound for our problem P.

Theorem (5)

For the $1//\sum(E_i + T_i + C_i)$ problem if $\sum C_i$ is obtained by *SPT* rule, then a lower bound *LB4* is given by $LB4 = Max\ \{2\sum_{i \in N} C_i - \sum_{i \in N} d_i, \sum_{i \in N} d_i\}$.

Proof:

To show that *LB2* is a valid *LB* for $1//\sum(E_i + T_i + C_i)$ problem, there are two cases, either $LB2 = 2\sum_{i \in N} C_i - \sum_{i \in N} d_i$ or $LB2 = \sum_{i \in N} d_i$

Case1. If $(2\sum_{i \in N} C_i - \sum_{i \in N} d_i) = \text{Max}\ \{2\sum_{i \in N} C_i - \sum_{i \in N} d_i, \sum_{i \in N} d_i\}$

To show

$$2\sum_{i \in N} C_i - \sum_{i \in N} d_i \leq \sum_{i \in N} (E_i + T_i + C_i).$$

Since for job i we have $C_i - d_i + C_i \leq E_i + T_i + C_i$

$$2\sum_{i \in N} C_i - \sum_{i \in N} d_i \leq \sum_{i \in N} (E_i + T_i + C_i).$$

Hence $LB2 = 2\sum_{i\in N} C_i - \sum_{i\in N} d_i$ is a lower bound for $\sum_{i\in N}(E_i + T_i + C_i)$ problem.

Case 2. If $\sum_{i\in N} d_i = Max\ \{2\sum_{i\in N} C_i - \sum_{i\in N} d_i, \sum_{i\in N} d_i\}$ this means that

$$\sum_{i\in N} d_i - \sum_{i\in N} C_i \geq 0 \quad \text{and}$$

$$\sum_{i=1}^{n} E_i = \sum_{i=1}^{n}(d_i - C_i)\,,\ \sum_{i=1}^{n}(E_i + T_i + C_i) = \sum_{i=1}^{n} d_i\ .$$

Hence $LB2 = \sum_{i\in N} d_i$ is a lower bound for $\sum_{i\in N}(E_i + T_i + C_i)$ problem.

4.2 An Upper Bound (UB)

In this section, we propose a heuristic method which is applied at the root node of the branch and bound tree to find an upper bound *UB* on the minimum value of problem (*P*). We now give precise details of our heuristic:

Heuristic UB

- Step 1. Ordered the jobs according to *EDD* rule, assume the resulting sequence is σ .
- Step 2. put $SE = \{job\ i \in \sigma : C_i < d_i\}$
  $SC = \{job\ i \in \sigma : C_i = d_i\}$
  $ST = \{job\ i \in \sigma : C_i > d_i\}$.
- Step 3. Let $\sigma_1$be the order of the jobs in subset *SE* according to *LPT* (i.e. $p_i \geq p_{i+1}$),
  σ_2 be the order of the jobs in subset *SC* according *SPT* (i.e. $p_i \leq p_{i+1}$) and σ_3 be the order of the jobs in subset *ST* according to *SPT*.
- Step 4. Let π be a schedule consist of $\sigma_{1,}\ \sigma_2$ and σ_3 i.e. $\pi = (\sigma_1, \sigma_2, \sigma_3)$.
- Step 5. Upper bound $UB = \sum_{i=1}^{n}(E_i + T_i + C_i)$ is obtained for π.

To find a lower bound for our problem (*P*), we applied the methods, which are described in section (4).The method which gives a big lower bound will be used in *BAB* method.

5. Optimal Solution by Using BAB Algorithms.

In this section, we consider the problems which are described in the previous sections. We wish to find a schedule which minimizes the sum of of earliness, tardiness and completion time. We shall use a branch and bound (*BAB)* method forward branching to find exact solution for problem (*P*). The *BAB* method starts by applying the special cases given in section (3). If the data for our problems satisfy the conditions of the special cases for the problem (*P*) then that problem is solved with out branching. If the date does not satisfy the conditions, then at this stage the *BAB* is started.

In our *BAB* method we applied the upper bound given in subsection (4.2) at the root node of the search tree to provide an (*UB*) on the cost of optimal schedule. Also at the root node of the search tree an initial lower bound on the cost of an optimal schedule is obtained from *LB* given in subsection (4.1). A dynamic programming dominance (*DP* dominance) rule is used from level two of the search tree in an attempt to eliminate some of the nodes which are not leads to optimal solution. For all nodes which are not eliminated by *DP* dominance rule, we can use the bounding procedure described in the previous section to compute a lower bound *LB*. If the *LB* for any node is greater than or equal to the current upper bound (*UB*) already computed, then this node is discarded, otherwise it may be selected for next branching.

The *BAB* method continues in a similar way by using forward branching procedure. Whenever a complete sequence is obtained, this sequence is an evaluated and the (*UB*) is altered if the new value is less than the old one. The procedure is repeated until all nodes have been considered (by using backtracking procedure). Backtracking procedure is the movement from the lowest level to the upper level in the search tree. The (*UB*) at the end of this procedure is the optimal of our scheduling problem.

6. Local search

To solve the scheduling problems already maintained in the previous sections, one tends to use optimization algorithms, which for sure always find optimal solution. However, not for all optimization problems, polynomial time optimization algorithms can be constructed. This is being because some of the problems are NP–hard. In such cases one often uses heuristic (local search) algorithms which tend toward but do not guarantee the finding of optimal solutions for any instance of an optimization problem. Hence in the recent years, much attention has been devoted to a number of local search heuristics for solving scheduling problems. Essentially, local search consists of moving from one solution to another, in the neighborhood, according to some defined rules. The sequence of solutions can be called a trajectory in the solution space. This trajectory depends heartily on the initial solution and on the neighborhood generation adopted. The main weakness of basic algorithms is their inability to escape from local optimal [15].

In this section we investigate the performance of several local search heuristics.

6.1 Descent Method (DM)

We suggested the following heuristic as a descent method to find an optimal or near optimal solution for our problem. In this heuristic method we shall relate the weighted with each schedule to prevent repetition of the schedule.

Algorithm (DM) for problem (P)

- Step1. Arrange the jobs according to *SPT* rule, to obtain an initial schedule $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$; $k=1$.
- Step 2. Evaluate the weighted of schedule π
$$H(k)=\sum_{i\in\pi}\pi(i) \ * \ 2^{i-1}, \ i=1,2,\ldots,n.$$
- Step 3. Evaluate the cost $Z(\pi)=\sum_{i=1}^{n}(E_i + T_i + C_i)$ as current solution.
- Step 4. Select randomly $i, 1 \le i \le n$.
- Step 5. Select randomly $j, 1 \le j \le n-1$.
- Step 6. if $(i=j+1)$ go to step 5.
- Step 7. Insert the job in position i at position $j+1$ (.i.e. $\pi(j+1)=\pi(i)$),

to obtain a new schedule π'; k=k+1.

- Step 8. Evaluate $H(k+1)=\sum_{i\in\pi'}\pi'(i)*2^{i-1}$, the weighted of schedule π' .
- Step 9. Let L =1.
- Step 10. If ($H(k+1)$) = $H(L)$ go to step 4.
- Step 11. Let L=L+1.
- Step 12. If ($L\leq K$) go to step *Step*10.
- Step 13. Evaluate $Z(\pi')$.
- Step 14. If ($Z(\pi')<Z(\pi)$); $\pi=\pi'$; $Z(\pi)=Z(\pi')$, Go to step 4.
- Step 15. The process is repeated from step 4 until no improvement can be found and stop.

6.2 The Simulated Annealing (SA) Method

In this subsection. We will refine the basic versions of the *SA*, we may vary or specify three elements in algorithm of simulated annealing: the distribution on the set of neighbors, the stopping criteria and the treatment of the control parameter *T*. The *SA* is a local search algorithm where, given the current solution π, another solution π' is drawn uniformly from the neighborhood of π ($N(\pi)$). If π' is better than π, then the next solution, is set equal to π', if not we accept the deterioration with a certain probability *P* (where $P=EXP(-\Delta/T)$, Δ a difference of costs) and do not accept with probability *1- P*. The algorithm terminates if there is no change after *L* repetitions. Otherwise, the iteration continues with a new temperature (*T*).

Simulated annealing (SA) algorithm.

1. Common, We use a simulated annealing as a local search to find a best schedule gets a minimum cost of objective function.

2. Input, π : an initial solution (Current solution). *T*: an initial temperature (controls the possibility of the acceptance of a deteriorating solution).

L: iteration number (decides the number of repetitions until a solution reaches a stable state under the temperature).

$Z(\pi)$: Cost of objective function which is associated with schedule π

$N(\pi)$: the neighborhood of schedule π

3. Output, the best schedule belong to $N(\pi)$ which is minimized the objective function.

4. Do $k=1, L$

5. Select at random $\pi'\in N(\pi)$ (a new solution)

6. $\Delta = Z(\pi') - Z(\pi)$

if ($\Delta < 0$) *or* ($P>\alpha$, α random from interval (0,1))

then $\pi=\pi'$; $Z(\pi)=Z(\pi')$

End if

$T=\alpha T$

End do

7. If stop criterion is not true, pick a new control parameter T; go to step 4.

6. 3 Threshold Accepting (T A) Method

In this subsection we shall use threshold accepting method to solve our problems. To obtain an initial solution we used the descent method presented in (6.1) as a current solution and the initial threshold value in our algorithm is equal to 0.5% of the multi objective functions value of the starting solution. Further, we used a geometrically decreasing threshold scheme and execute move iterations in later search phases.

Threshold Accepting (TA) Algorithm

- Step .1 Let π be initial solution
  Evaluate $Z(\pi)$
  Set $UB := Z(\pi)$, $\mu = \pi$, $\epsilon = 0.000001$.
  Set $t = 0.5\%\ UB$, be an initial threshold value, and $\beta = 0.5$.
- Step .2 Select randomly $\pi 1 \in N(\pi)$ as a new solution
  Evaluate $Z(\pi 1)$.
- Step .3 If ($UB > Z(\pi 1)$), $UB := Z(\pi 1)$, $\mu = \pi 1$.
- Step .4 If ($Z(\pi 1) < Z(\pi) + t$) then
  $\pi = \pi 1$, $Z(\pi) = Z(\pi 1)$
  End if
- Step .5 If ($t > \epsilon$) then
  $t = \beta t$
  Go to step *2*
  End if
- Step .6 If ($UB < Z(\pi)$) then
  $Z(\pi) = UB$, $\pi = \mu$
  End if
- Step .7 Stop.

7. Genetic Algorithm (G A)

Genetic algorithm has seen limited usage as a scheduling tool. It is based loosely an evolution and the concept of "survival of the fittest". A vector is used to represent the parent chromosome and an allele is defined as the value represented by a single element within the vector. A set of parents is generated and operations are performed to represent the pooling of genes that result in an improvement to the species. Operations that may be performed are crossover, a random exchange of genes between parents, and reproduction, which allows the best solution to the next generation [5].

Initial GAs was programmed using a series of zeros and ones. Other variants included the use of integers in the vector. A common problem within scheduling applications was the creation of solutions that were not feasible (Bean, 1994) [4]. Now, we give an example state by Bryan and Bahram (2002) [5], if we are given six jobs placed in the following two sequences,

1-4-6-3-2-5
5-1-4-2-3-6

and they perform crossover at element three removing job 6 from the sequence one and replacing it in the position of job 4 in sequence two and removing job 4 from sequence two and replacing it with job 6 resulting job sequences will be

1-4-4-3-2-5
5-1-6-2-3-6

The two new sequences are obviously not feasible solutions. To avoid this possibility Bean (1994) [4] and Bryan & Bahram (2002) [5] proposed using random keys within the vectors to act as surrogates for the final job sequences. Vectors of random numbers selected from 0 to 1 are generated to represent the parents. To transform the offspring (sequences of random numbers) into a feasible job sequences they start with the smallest number and place its position number in the first position of the job sequence, the next smallest number is identified and its position number is placed in the next position of the job sequence; and so on until all elements are assigned. To avoid this not feasible solutions we proposed the following method: in the final job sequence (not feasible solutions) we fixed the elements which are performed crossover on them in the same positions, and we change the job which is listed twice from sequence one with the job which is listed twice in sequence two. Using the process previously we get the following two sequences each of which represent a feasible solution.

1-6-4-3-2-5
5-1-6-2-3-4

Hybrid method apply when the parents generate identical offspring (i.e. generate two identical chromosomes (sequences) have the same properties of his parents) and make cycle. To avoid this cycle we hybrid parent one by using descent method to perform parent are different from parent two and continue by crossover and reproduction operations.

7.2 Genetic simulated Annealing (GSA) Algorithm

In this section we present a hybrid *GSA* by combining the simulated annealing with the genetic algorithm. This method aims to creating an alternative search technique incorporating the best characteristics presented by each method while solving the problems (P). The ultimate goal is to improve the efficiency by reducing the computational processing time to improve the effective the solution. The results of *GSA* are comparing with results of the *GA* and local search methods.

Celia and Potts (1996) [7] used hybrid method in which descent is incorporated into the genetic algorithm to evaluate on the problem of scheduling jobs in permutation flow-shop to minimize the weighted completion time. Celse et al. (2005) [6] propose a hybrid strategies that combine genetic algorithms and tabu search concepts for the single-machine scheduling problem with a common due date performance is measured by the minimization of the sum of earliness and tardiness penalties of the jobs.

8. Computational Experience

All the algorithms of this study are coded in Fortran Power Stations (Fortran 90) and run on a Pentium IV hp-Compaq computer with a 2.8 GHz processed and 256 Mb of RAM memories. The branch and bound algorithms are tested on problems 10, 15, 20, 25, 30 and 35 jobs for problems (P). We using the method of data generation that given by Abdul-Razaq and Potts (1993). That were generated as follows: five integers were generated for every job i, namely processing time p_i, and due date d_i. p_i were generated randomly from the uniform distributions [1,10]. Due date d_i for every problem were generated from the uniform distribution $[h_1T, h_2T]$ where

$$h_1 \in \{0.2, 0.4, 0.6, 0.8\}$$
$$h_2 \in \{0.4, 0.6, 0.8, 1.0\}$$
$$T = \sum_{i=1}^{n} p_i \text{ , } h_1 < h_2$$

For each selected value of n, two problems were generated for each of the ten pairs of h_1, h_2 producing 20 problems for each value of n. We solved the problem (P) with up to 150000 jobs; the problems are tackled by reams of all of the 5 approaches, namely *DM*, *SA*, *TS GA* and *GSA*.

Table (1) illustrates the comparative of computational results and CPU in second for 5 approaches with the optimal value (obtain from BAB method), n=20 for problem (P). Table (2) gives difference of the average of computational results of the 5 approaches with n=100, 200, 500, 1000, 1500, 5000, 10000, 15000, 50000, 100000 and 150000 jobs for problem (P). Figure (1) indicates that the average CPU time (running time in second) required by each method (for problem p).

Table (1) Comparative results and CPU in second, n=20 for the problem:

$$1//\sum(E_i+T_i+C_i)$$

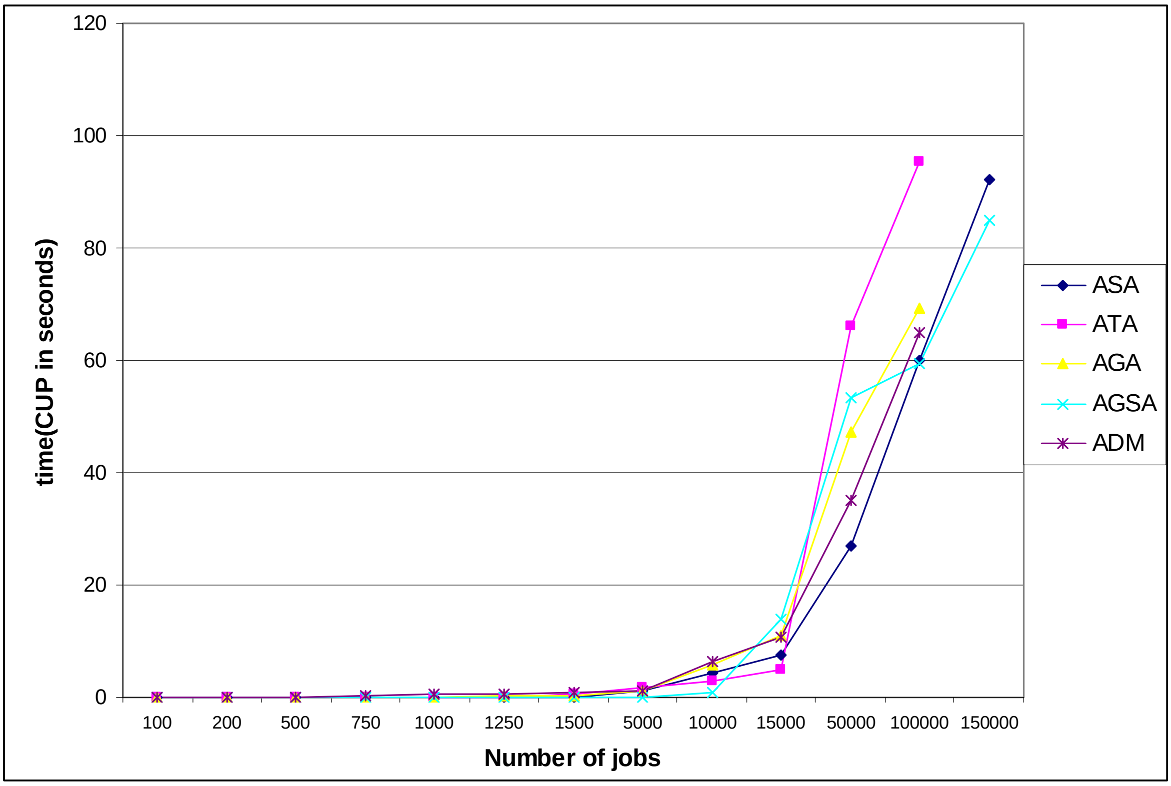
No.	BAB	time	SA	time	TA	time	GA	time	GSA	time	DM	time
1	1561	9.4	1955	0	1945	0	1705	0	1573	0	1817	.02
2	1465	30	1481	0	1473	0	1470	0	1465*	0	1469	.016
3	1538	15.6	1542	0	1542	0	1574	0	1542	0	1542	.028
4	1710	30.4	1724	0	1722	0	1722	0	1710*	0	1720	.02
5	1122	11.48	1174	0	1128	.06	1126	.01	1126	.06	1126	.16
6	1605	14.8	1617	0	1627	.06	1617	0	1613	0	1651	.17
7	1326	33.4	1400	0	1336	.06	1336	0	1326*	0	1342	.02
8	1473	31.5	1540	0	1476	0	1478	0	1473*	0	1492	.16
9	1509	41.6	1525	0	1523	0	1519	.05	1509*	0	1517	.17
10	1099	30.04	1117	0	1163	.05	1105	.06	1105	0	1129	.16
11	1125	30	1171	0	1167	0	1125*	0	1125*	.05	1163	.17
12	1370	44.1	1388	0	1384	0	1384	.05	1384	0	1387	.11
13	1334	50	1371	0	1353	.06	1339	.05	1335	.06	1363	.16
14	1946	52.6	1956	0	1986	0	1956	0	1946*	.06	1956	.11
15	1288	44.1	1289	.05	1291	0	1289	.05	1288*	.05	1307	.16
16	1054	33	1080	0	1062	.05	1064	.06	1054*	.05	1070	.17
17	1458	28	1460	0	1464	0	1476	0	1460	0	1464	.11
18	1429	25.9	1475	0	1437	0	1439	.05	1437	.05	1457	.16
19	1246	44.1	1282	0	1250	0	1256	.06	1456	.06	1256	.16
20	1494	50.2	1512	0	1518	.05	1494*	0	1494*	0	1510	.16

**: Indicates that the problem has an optimal solution equals to the heuristic value.*

Table (2) Averages deviations about the best average value for the problem $1//\sum(E_i+T_i+C_i)$

n.	deviations of averages				
	SA	TA	GA	GSA	DM
100	0	0.9	9.9	0.2	2
200	58.4	0	78.8	4.13	56.8
500	236.9	0	49.56	45.78	1014.87
1000	0	896	868	0	6751
1500	0	684	148	0	738.4
5000	126.1	362	0	0	461.3
10000	1157.8	527.2	84.17	0	2103.9
15000	1850	0	1951	696.2	11532
50000	0	2408.1	3358	0	48511
100000	18690	11481	5611	0	19234
150000	101581	21953	29915	0	11403

Fig. (1): Time averages for the problem. $1//\sum(E_i+T_i+C_i)$



Where:
ATA: Time averages of the results of the threshold accepting.
ASA: Time averages of the results of the simulated annealing.
AGA: Time averages of the results of the genetic algorithm.
AGSA: Time averages of the results of the hybrid (genetic simulated annealing).
ADM: Time averages of the results of the descent method

9. Conclusions.

In the present chapter we consider a single machine scheduling problems to minimize the sum of earliness, tardiness and completion time penalties of the jobs and its special cases. As these problems are NP-hard, we propose a branch and bound algorithms to obtain an optimal solution with up to 35 jobs, the implementation of optimizing algorithms does seem to be promising. Thus we decided to tackle the problems with local search and meta-heuristics methods. We solved the problem with up to 150000 jobs. From tables (1) and (2) we see that *BAB* algorithm gives an optimal value but it need longer time comparative with the local search methods. Furthermore, by comparing the performance of the local search methods with optimal solution or best solutions, hybrid strategies GSA method to get the better of other methods.

References:

[1] Abdul-Razaq, T. S. and Potts, C. N. "Dynamic Programming State-Space Relaxation for single- Machine Scheduling", *J. Opt .Res.* Vol.39. No. 2.141- 152 (1988).

[2] Abdul-Razaq, T. S. and Mahmood, A. A. "Single Machine Earliness and Tardiness Schedule with Set-up Time". *The first National Conference In Mathematical Science 25-26/Sep/2001. College of Science, Al-Mustansiriyah University Baghdad-Iraq 3/7*

[3] Abdul-Razaq, T. S. "Machine Scheduling Problem: A Branch and Bound Approach", *Ph. D. Thesis Department of Mathematics university of keele, July*, 1987.

[4] Bean, J. C. "Genetic algorithms and Random Keys for Sequencing and Optimization" *ORSA Journal on Computing* 6(2), (1994), 154-160.

[5] Bryan, R. k. and Bahran Alidaee, "Single Machine Scheduling to Minimize Total Weighted Late Work: a Comparison of Scheduling Rules and Search Algorithm", *Computers and Industrial Engineering* 43 (2002)509-528.

[6] Celso M.Hino, Debora P. Ronconi and Andre B. mends, " Minimizing Earliness and Tardiness Penalties in a Single-Machine Problem with a Common Due Date" *European Journal of Operational Research* 160 (2005) 190-201.

[7] Celia A.Glass and Chris N .Potts, "A Comparison of local search methods for flow shop scheduling", Annals operations Research 63 (1996) 489-509.

[8] Chichang Jou, "A genetic algorithm with sub-indexed partitioning genes and its application to production scheduling of parallel machines", *Computers and Industrial Engineering* 48 (2005) 39-54.

[9] George Li, "Theory and Methodology Single Machine Earliness and Tardiness Scheduling", *European Journal of Operational Research*, 1996.

[10] Hall, N. G. , Kubink W. and S. P. Sethi, "Earliness-Tardiness Scheduling Problem. II: Deviation of Completion Times about a Restrictive Common Due Date", *Operations Research,* 39(1991), 847-856.

[11] Hoogeveen, J. A. and S. L. Van de Velde "Scheduling a Round a Small Common Due Date" *European Journal of Operational Research*. 55(1991), 237-242.

[12] Jan Bank and Frank werner, "Heuristic Algorithms for Unrelated parallel Machine scheduling with a Common Due Date, Release Dates and Linear Earliness and Tardiness Penalties", *OTTO-Van-Guericke-Universitat Magdeburg, Postfach 4120, 39016 Magdeburg Germany(2000).*

[13] Kanet, J. J. "Minimizing the Average Deviation of jobs Completion Times about a Common Due Date", *Naval Research Logistics Quarterly* 28 (1981), 643-651.

[14] Martin Feldmann and Dirk Biskup, “Single-Machine Scheduling for Minimizing Earliness and Tardiness Penalties by Meta-heuristic Approaches”, *Computers and Industrial Engineering* 44 (2003) 307-323.

[15] Modurerira A., Ramoscand Silva S.C. “A genetic approach to dynamic scheduling for total weighted tardiness problem”, (*Internet*).

[16] Mohammed A. Hussam “Genetic and Local Search Algorithm as Robust and Simple Optimization Tools.” *M.SC. Mathematics Department, College of Science, University of AL-Mustansiriyah*, (2005).

[17] Nicos Christofides, A. Mingozzi and P. Toth, “State-Space Relaxation Procedures for the computation of Bounds to Routing problems”, *NETWORKS*. Vol. 11(1981) 145-164.

[18] Ow, P. and T. Morten, “The Single Machine early-Tardy Problem”, *Management Science*, 35 (1989) 177-191.

[19] Rinnooy Kan, A. H. G. “Machine scheduling problems: classification, complexity and computations, matins Mijhoff”, / *The HaGue Holland* 1976.

[20] Smith, W. E. “Various Optimizers for Single-Stage Production”, *Naval Research Logistics Quarterly* 3 (1956).

دالة هدف مركبة في مسالة جدولة الماكنة الواحدة

طارق صالح عبد الرزاق* صاحب كحيط الساعدي* محمد كاظم زغير**

*قسم الرياضيات/ كلية علوم/ الجامعة المستنصرية

**قسم الرياضيات/ كلية الحاسبات والرياضيات/ جامعة ذي قار

الخلاصـــة

في هذا البحث درسنا مسالة جدولة الماكنة الواحدة لتصغير دالة هدف مركبة "مجموع التبكير والتـأخيرالاسالب وزمن إتمام النتاجات على الماكنه الواحده " . أن هذه المسألة من نوع NP-hard لذا اقترحنا طريقةالتفرع والتقيد لايجاد الحل الامثل علماً ان الحل الأ مثل يتطلب وقت أطول. وقد استخدمنا طرق البحث المحلي لايجاد الحلول التقريبية حسبت النتائج لهذه الطرق وقورنت النتائج مع الحل الامثل وكذالك مع الخوارزمية الجينية وطريقة التهجين المقترحه التي هجنا فيها simulated annealing مع الخوارزمية الجينية.افضل النتائج اعطتها طريقة التهجين. وجدنا الحل امثليا للمسألة لغاية 35 نتاج. وتقريبياً لغاية ١٥٠٠٠٠ نتاج.