Journal of Kerbala University , Vol. 7 No.1 Scientific. 2009

Single machine scheduling to minimizing sum penalty number of late jobs subject to minimize the sum weight of completion time

Hussam A. Mohammed €
Hanan A. Cheachan £
Qhassan A. Khtan ¥

Abstract:-

This paper considers of scheduling (n) jobs on single machine to minimize the sum penalty number of late jobs with minimum sum of weighted of completion time $1||Lex(\sum_{i=1}^{n}\alpha_i C_i, \sum_{i=1}^{n}\beta_i U_i)$. To solve this lower bound and some dominance rule are derived and it is incorporated in a branch and bound (B&B) algorithm. We propose heuristic method to find near optimal solution. We also report on computational experience with the branch and bound. Also we develop compare and test different local search method (Threshold accepted (TA), Tabu search (TS) and Ant colony optimization (ACO) algorithm) for the problem. Computational experience is found that these local search algorithms solve problem to 5000 jobs with reasonable time.

Keyword:- Scheduling Single machine, Bicriteria, Lexicographical optimization, Branch and bound, Local search methods

المستخلص:-

(في هذا البحث دُرست مسألة جدولة الماكنة, أخذ بنظر الاعتبار مسألة جدولة (n) من الأعمال على ماكنة واحدة. الهدف هو أيجاد الجدولة المثلى لتلك الأعمال لتصغير مسألة $1||Lex(\sum_{i=1}^{n}\alpha_i C_i, \sum_{i=1}^{n}\beta_i U_i)$ وقدمنا الصيغة الرياضية ووصفنا عدد من طرائق الحل لهذه المسألة. لحل هذه المسألة فقد اقترحنا قيد أدنى لاستخدامه في طريقة التفرع والتقيد (B&B) بالإضافة إلى عدد من قواعد الهيمنة برهنت لتحذف التفرعات في هذه الطريقة. أيضاً لحل المسألة ذات حجم كبير من الأعمال استخدمت طرائق البحث المحلي (TA, TS and ACO). طرائق البحث المحلي استخدمت لتصغير الزمن المستخدم لإيجاد الحل يصل إلى 5000 عمل في زمن معقول. عملنا هذا قدم مساهمة لمسائل جدولة الماكنة الواحدة.

1. Introduction :

The majority of the scheduling research has concentrated on single criterion problem. However in practice the scheduling decisions require consideration of multiple criteria to present more realistic solutions to decision maker.

The general multi objective combinatorial optimization problems can be formulated as follows:

Minimize or maximize $F(s) = (f_1(s), f_2(s), \dots, f_k(s))$ such that $s \in S$ where s is a solution. S is the set of feasible solution, k is number of objective in the problem, $F(s)$ is the image of s in the k-objective space and each $f_i(s), i = 1, 2, \dots, k$ represents one (minimization or maximization) objective.

Suppose that we have selected the two performance criteria say f and g, that we want to take into

€Department of Mathematics, College of Education, University of Kerbala, Kerbala, Iraq, (hussammath@uokerbala.edu.iq).
£Department of Mathematics, College of Sciences, University of Mustansiriyah, Baghdad, Iraq, (hanan_altaai@yahoo.com).
¥ Department of Mathematics, College of Education, University of Kerbala, Kerbala, Iraq, (ghassanmath@uokerbala.edu.iq).

account, without loss of generality, we assume that these criteria are to be minimized, unless we are extremely lucky there will be no schedule that achieves the minimum value for both performance criteria simultaneously, which implies that we have to give in the quality of at least one of the two criteria. If one performance criteria, say f is far more important than the other one, then an obvious approach is to find the optimum value with respect to criteria f which we denote by f^* and choose from among the set of optimum schedules for f the one that performs best on g. Such an approach is called hierarchical optimization or lexicographical optimization (denoted by Lex). Recently, much research has been directed scheduling problems with multiple criteria, please refer to Nagar et. al. (1995) as well as T'kindt, Billaut (2002) and Pinedo (2002) for a comprehensive overview. In the past, several researchers have explored sequencing problems where the two criteria are weighted flow time and maximum tardiness (see Smith (1956), Heck and Roberts (1972), Burns (1976), Bansal (1980), Miya Zaki (1981), Sen and Gupta (1983)). Framinan and Leisten (2006) considered the problem of Makespan minimization in a permutation flow shop where the maximum tardiness is limited by given upper bound. They obtained a good heuristic to solve this problem. Pathumnakul and Egbelu (2006) considered the problem of minimizing the weighted earliness penalty in assembly job shops, where no late jobs are allowed. They developed a heuristic to solve this problem. Azizoglue et al. (2003), Guahua and Benjamin (2008) and Gwo (2007) in our researches, we will consider the single machine scheduling problem with minimizing the number of late jobs which is measured by minimizing of maximum earliness for the jobs. The studies by Feldmann and Biskup (2003), Hino et al. (2005) Valente and Alves (2006) are examples of lexicographical minimization of the total of earliness and tardiness penalties for the jobs.The rest of the paper is organized as follows. The formal description for the problem is given in section (2). Upper and lower bounds are given in sections (3). Dominance rules and special case are given in section (4). In section (5) local search heuristic methods. Branch and bound algorithm is given in section (6). Computational experience is given in section (7). Finally a conclusion and future work are given in section (8).

2. Formulation of the $\mathbf{1}\,||\,\boldsymbol{Lex}\left(\sum_{i=1}^{n}\boldsymbol{\alpha_i C_i}, \sum_{i=1}^{n}\boldsymbol{\beta_i U_i}\right)$ problem

Our scheduling problem can be state more precisely as follows: A set of (n) jobs $N = \{1, 2, \dots, n\}$ are variables for processing at time zero. Job $i\ (i = 1, \dots, n)$ is to be processed without interruption on single machine that can be handling only one job at a time requires processing time p_i and be completed before its due dare d_i. With weighted (importance weight for completion time of job i) α_i and positive number (penalty for lateness of job i) β_i. For a given a sequence σ, $\sigma = (\sigma(1), \sigma(2), \dots, \sigma(n))$ of jobs completion time $C_{\sigma(i)} = \sum_{k=1}^{i} p_{\sigma(k)}$ and $U_{\sigma(i)}$ are given by:

$C_{\sigma(1)} = p_1$ and $C_{\sigma(i)} = C_{\sigma(i-1)} + p_{\sigma(i)}\ ,\quad i = 2, \dots, n$

$$U_{\sigma(i)} = \begin{cases} 1 & if \quad C_{\sigma(i)} > d_{\sigma(i)} \\ 0 & if \quad C_{\sigma(i)} \leq d_{\sigma(i)} \end{cases}$$

A feasible schedule is sequence of all jobs for which minimizes the sum weighed completion time, it is desired to find a feasible solution which minimize sum penalty number of late jobs.

Let S denote the set of all feasible schedule where $\sigma \in S$, the objective is to find the processing order of the jobs that minimize the sum penalty number of late jobs. In the following problem:

$$\left.\begin{aligned}
&\underset{\sigma \in S}{\text{Min}} \sum_{i=1}^{n} \beta_i U_i \\
&s.t. \\
&C_{\sigma(i)} \geq p_{\sigma(i)}, \qquad i = 1, \ldots, n \\
&C_{\sigma(i)} = C_{\sigma(i-1)} + p_{\sigma(i)}, \quad i = 2, \ldots, n \\
&U_{\sigma(i)} = \begin{cases} 1 & if \quad C_{\sigma(i)} > d_{\sigma(i)} \\ 0 & if \quad C_{\sigma(i)} \leq d_{\sigma(i)} \end{cases} \\
&\sum_{\sigma \in S} U_{\sigma(i)} = n_{\mathcal{T}}, \quad \text{where } n_{\mathcal{T}} \text{ is number of late jobs} \\
&\alpha_{\sigma(i)} \geq 0, \quad \beta_{\sigma(i)} \geq 0
\end{aligned}\right\} (P_{Lex})$$

Using the field classification suggest by Graham et al. [10] the problem P_{Lex} denoted by $1 \mid\mid Lex \left(\sum_{i=1}^{n} \alpha_i C_i, \sum_{i=1}^{n} \beta_i U_i \right)$

3. Upper bound procedure

In this section, we propose heuristic method which is applied once at the root node of branch and bound (B&B) search tree to find an upper bound (UB) on the minimum value of $1 \mid\mid Lex \left(\sum_{i=1}^{n} \alpha_i C_i, \sum_{i=1}^{n} \beta_i U_i\right)$ problem.

3.1 Derivation of an upper bound

In this section, we propose an algorithm for finding UB. The algorithm initially finds a schedule of jobs by WSPT algorithm (weighted shortest processing time order of the jobs). The general step of our algorithm has $s_1 = \sigma\sigma_1 ij\sigma_2$ as its current feasible sequence where σ is a partial sequence of jobs that is already fixed and $\sigma_1 ij\sigma_2$ is subsequence of the remaining jobs which is obtain by WSPT rule. The partial sequence σ_1 is a WSPT sequence is chosen to contain as many jobs as possible i and j are individual jobs and σ_2 partial sequence containing the other jobs, job i and jobs of σ_1 are not in WSPT rule and no constraint on job j. now consider the new feasible schedule $s_2 = \sigma i\sigma_1 j\sigma_2$ and compare the schedule s_1 and s_2, if the current feasible sequence s_1 has smaller sum of penalty number of late jobs $(\sum_{i=1}^{n} \beta_i U_i)$ than s_2 then s_1 it is retained as the current feasible sequence and $\sigma\sigma_1$ becomes the fixed jobs of the schedule, otherwise s_2 becomes the current feasible sequence where σi is the fixed jobs the schedule repeat this technique until we get the final form the feasible schedule with $\sum_{i=1}^{n} \beta_i U_i$ as small as possible.

3.2 Lower bound (LB)

Using earliest due date (EDD) rule to find the minimum penalty number of late jobs say $n_{\mathcal{T}}$, and denote this schedule by $s = (\sigma_1, \sigma_{EDD})$ where σ_1 and σ_{EDD} two partial sequences of jobs. At the beginning $\sigma_1 = \emptyset$ and σ_{EDD} is obtained by EDD rule. The main step in the LB is to transform some jobs from σ_{EDD} to σ_1 with reduction in processing time if necessary, such that the new schedule s is feasible and calculate $\beta' \sum U_i$ where $\beta' = \min_{1 \leq i \leq n} \{\beta_i\}$ for this schedule this LB can be stated as follows:

Let σ_1 be the partial sequence of jobs that is already fixed and σ_{EDD} partial sequence (belong to the unsequence jobs) sequenced by EDD such that s is feasible. This means listing the set of all feasible schedules $R = \{s_1, s_2, \ldots, s_r\}$ where each feasible schedule $s_k = (\sigma_1, \sigma_{EDD}), k = 1, \ldots, r$ with minimum

number of penalty late job $n_{\mathcal{T}}$ and with sum weighted completion time $\sum_{j=1}^{n} \alpha_j C_j$. Here our LB chooses the schedule s_k with minimum $\sum_{j \in s_k} \alpha_j C_j$.

property (1): For $1 \mid \mid Lex\ (\sum_{i=1}^{n} \alpha_i C_i, \sum_{i=1}^{n} \beta_i U_i)$ problem there exist optimal schedules in which on time jobs (early jobs) are sequenced in Smith back ward and late jobs are sequenced WSPT order.

Proof: Suppose that there is a feasible partial sequence σ in which job j is sequenced before job i where i and j are (on time) early jobs and $d_i,\ d_j \geq R = \sum_{i=1}^{n} p_i - \sum_{k=1}^{n_{\mathcal{T}}} p_k, p_j > p_i$ where $n_{\mathcal{T}}$ minimum number of late jobs given by WSPT.

Consider the partial σ' which obtain from σ by removing job j from its original position and inserting it immediately after job i, only job j has a later completion time in sequence σ' than σ, $\sum_{k=1}^{j} C_k(\sigma') < \sum_{k=1}^{i} C_k(\sigma)$ job j is on time σ'. Thus σ' is also a feasible partial sequence. Repeating this procedure a finite number of times. We obtained an efficient partial sequence in which (on time) early jobs appear in (Smith back ward). It is clear for the late jobs, if we consider i and j such that $p_j > p_i$ for the remaining partial sequence σ' in which j precedes i and $C_j(\sigma) > d_j, C_i(\sigma) > d_i$, when we interchange position of job j and job i in the new partial sequence σ_1' such that both jobs are late, hence the number of late jobs doesn't increasing but decreasing in completion times i.e. $\sum_{k=1}^{j} C_k(\sigma_1') < \sum_{k=1}^{i} C_k(\sigma_1)$. Hence the late jobs are sequenced in WSPT order.

4 Dominance rules and special cases :

For $1 \mid \mid Lex\ (\sum_{i=1}^{n} \alpha_i C_i, \sum_{i=1}^{n} \beta_i U_i)$ problem, we shall prove the following dominance rules:

1. Let S be in which job k be on time and job j be late then schedule s' be dominate of S in which job j be late than schedule s' be dominate of S in which job j be on time and job k is late if the following condition hold:

$$\alpha_k p_j \leq \alpha_j p_k,\ \ d_j \leq d_k$$

Proof: In a schedule S where job j is late and job k is on time (and k is sequenced before j), if we interchange job k and j on S such that job j is (on time) early time, after interchange if number of late jobs doesn't increasing. The condition $\alpha_k p_j \leq \alpha_j p_k$ ensure that the increasing only the completion of job k.

2. Let S be a schedule in which jobs j is late and job k is on time then a schedule s' dominate of S in which job j be on time and job k is late if the following conditions hold:

$$\alpha_k p_j \leq \alpha_j p_k,\ d_j \leq d_k,\ \sum_{i=1}^{h} p_i \leq d_h,\ \ h = 1, \dots, k$$

Proof: Consider a sequence S in which job k before job j, and job k is on time, job j is late. Let s' be new sequence by interchange jobs j and k, then property (1) shows that only jobs from $(1, \dots, k-1)$ are sequence before job j, hence job j and its predecessors can be sequenced so that each one of them is (on time) early job. Thus, the interchange doesn't increase the number of late jobs. Moreover the condition $\alpha_k p_j \leq \alpha_j p_k$ shows that the interchange only causes an increase of completion time of job k.

4.1 Special cases :

We consider special cases for $1 \mid \mid Lex\ (\sum_{i=1}^{n} \alpha_i C_i, \sum_{i=1}^{n} \beta_i U_i)$ problem as follows:

Case1: If $d_i \geq T = \sum_{i \in N} p_i, \forall\ i = 1, \dots, n$ then no late jobs and optimal sequence can be obtain by WSPT rule for the resulting $1 \mid \mid \sum_{i=1}^{n} \alpha_i C_i$ problem.

Case2: If $\alpha_j p_i \leq \alpha_i p_j, d_i \leq d_j, \forall\, i, j$ (agreeable) and $\beta_i = 1 (\text{or } \beta_i = \beta), \forall\, i$ then the sequence can be obtain by WSPT rule give optimal sequence.

5. Local search for $1 \mid \mid Lex\,(\sum_{i=1}^{n} \alpha_i C_i, \sum_{i=1}^{n} \beta_i U_i)$ problem :

It is clear to solve scheduling problems one tends to use branch and bound (B&B) or Dynamic programming (DP) to find optimal solutions, however, these approaches has two main disadv-antages:
It is mathematically complex and thus a lot of to be invested.
When it concerns NP-hard problem, the computational time requirements are enormous for large sized problem. To avoid these draw backs we can appeal to heuristics methods. In recent year, the improvement in heuristic methods has becomes under the name 'local search methods' as well as there are Genetic algorithm and Ant colony algorithm. In this section several local search heuristic are implemented on the problem of scheduling (n) of jobs on single machine to minimize the $Lex\,(\sum_{i=1}^{n} \alpha_i C_i, \sum_{i=1}^{n} \beta_i U_i)$ (minimize the sum penalty number of late jobs with minimum sum of weighted of completion). For the representation of solution the natural representation will be used. For each local search method a set of parameter setting is necessary for arriving at high performing algorithm. Conclusions concerning implementation of different setting are discussed.
First we introduce some neighbourhoods for a permutation problem, where the step of feasible solutions is given by the set of permutations of (n) jobs [14]. Jump (Ju) In a permutation $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$, select an arbitrary job $\sigma(i)$ and jump it to a smaller position $j, i > j$, or to a large position $k, k > i$. Thus, we have $|N(\sigma)| = (n-1)^2$. Pairwise Interchange (PI) In a permutation σ select two arbitrary jobs $\sigma(i)$ and $\sigma(j), i \neq j$ and interchange them, and $|N(\sigma)| = n(n-1)/2$.
Adjacent Pairwise Interchange (API) This is a special case of both the jump and the pairwise interchange neighbourhood. In a permutation σ, two adjacent jobs $\sigma(i)$ and $\sigma(i+1), (1 \leq i \leq n-1)$ are interchanged to generate a neighbour σ, where $|N(\sigma)| = (n-1)$.
Dyna search dynamic programming (Dyna) In this move we composed of set of independent interchange moves; each such move exchange the jobs at positions i and $j,\ i \neq j$. Two interchange move are independent if they don't over lap, that is if for two moves involving position i, j and k, l we have that $min\{i, j\} \geq max\{k, l\}$ or vice versa.
Now, we propose algorithm AH which is applied at local search methods to provide a best solution.
Algorithm AH

Step(1) Select an initial solution $\sigma_{Ini} = (\sigma(1), \sigma(2), \ldots, \sigma(n))$ obtain from the WSPT rule and calculate objective value of σ_{Ini} say $f(\sigma_{Ini}) = f_{Ini}$.

Step(2) In this step will be change the initial sequence σ_{Ini} by the others neighbourhoods and calculate values function for every one i.e.

For the neighbour Ju, have σ_{Ju} and f_{Ju}.

For the neighbour PI, have σ_{PI} and f_{PI}.

For the neighbour API, have σ_{API} and f_{API}.

For the neighbour Dyna, have σ_{Dyna} and f_{Dyna}.

Step(3) Now choose $min\, f^* = \{f_{Ju}, f_{PI}, f_{API}, f_{Dyna}, f_{Ini}\}$ and $min\, \sigma^* = \{\sigma_{Ju}, \sigma_{PI}, \sigma_{API}, \sigma_{Dyna}, \sigma_{Ini}\}$, then set $f_{Ini} = min\, f^*$ and $\sigma_{Ini} = min\, \sigma^*$

Now, we give details about local search methods (Meta-heuristic methods) which are used to solve $1 \mid \mid Lex\,(\sum_{i=1}^{n} \alpha_i C_i, \sum_{i=1}^{n} \beta_i U_i)$ problem.

5.1 Threshold Accepting (TA) method :

Threshold accepting (TA) method is similar to Simulated Annealing (SA) that uses deterministic acceptance rule for solution that cause a deterioration in the objective value. Here a move is accepted provide that if doesn't increase the objective function by more than V where V is threshold value. Now, describe the TA method as follows:

Step(1) (Initialization) In this step a feasible solution $\sigma = (\sigma(1), \sigma(2), \dots, \sigma(n))$ obtained from the best shorted weighted processing time (WSPT). Is chosen to be the initial current solution σ^* for TA method.

Step(2) Using algorithm AH for generated new solution σ^*.

Step(3) In this step we are updating the threshold values V_k for $1 \le k \le l$ we use $V_k = \xi V_{k-1}$ with $\xi = (V_l/V_1)^{\frac{1}{l-1}}$ where V_1 and V_l are initial and final threshold values respectively and set

$$\left.\begin{aligned} V_1 &= r_1 f(s_1) \\ V_l &= r_l f(s_1) \end{aligned}\right\}$$

where r_1, r_l are parameters that determine $r_1 = 0.02$, $r_l = 0.0001$ and s_1 is feasible solution.

Step(4) (Termination) The algorithm is terminated after (500) iteration at a feasible solution.

5.2 Tabu search (TS) method

Glover [9] combines the deterministic iterative improvement algorithm with a possibility to accept cost increasing solution. In this way the search is directed a way from local minima, such that other parts of the solution space can be inspected. This is done by maintaining a finite list of that are not acceptable in next few iterations. This list is called the Tabu list. However, a solution on the tabu list may be accepted if its quality is in some sense good enough, in which case it is said to attain a certain aspiration level.

Determination (TS) algorithm parameters:

To determine the parameters of TS for our problem $1 \mid\mid Lex\ (\sum_{i=1}^{n} \alpha_i C_i, \sum_{i=1}^{n} \beta_i U_i)$, we discuss each of the following issues:

Step(1) (Initialization) Determine the initial solution as above mentioned method.

Step(2) (Neighbourhood generated) Using the same procedure in step (2) in TA.

Step(3) Add the new neighbourhood in the tabu list if not exist otherwise ignore it.

Step(4) (Termination) This termination condition used here is the same one described in section (5.1).

5.3 Ant colony optimization (ACO) algorithm

Basic definition of ACO The main idea of the ACO is to keep a population or colony of (n) artificial ants that interactively builds solution by continually applying a probabilistic decision policy (n) times until a solution is found. Ants that found a good solution mark their path through the decision space by putting some a mount of pheromone on the edges of the path. Ants of the next iteration are attracted to the pheromone resulting in a higher probability to follow the already traversed good paths. In addition to the pheromone values, the ants will usually be guided by some problem specific heuristic for evaluating possible decisions regarding which direction to take along the way. In ACO algorithm ants have a memory that stores visited components of their current path. A part from the construction of solutions and depositing of pheromone the ACO incorporates other methods, pheromone evaporation it causes the amount of pheromone on each edge to decrease over time. The important property of evaporation is that it prevents premature convergence to a suboptimal solution. In this manner the ACO has the capability of "forgetting" bad solution of the search space [6].

The Ant colony optimization (ACO)

Ant colony optimization (ACO) is a meta-heuristic uses artificial ants to good solution to difficult combinatorial optimization problem. It can be described by the following steps:

Step(1) (Initial pheromonetical) The ant colony optimization used here is slightly different from the

traditional ant colony optimization. At the beginning an initial solution is generated by the same technique described in step (1) of section (5.1). The initial pheromone triad $\tau_{\circ ij}(t)$ will be the invert of initial solution objective.

Step(2) (Population) Each artificial ant k iteratively and independently generates a complete solution by selecting a job j to be on the ith position of the sequence. This selection depends on the pheromone trial $\tau_{ij}(t)$ and heuristic information ε_{ij} for the problem $1\,||Lex\,(\sum_{i=1}^{n}\alpha_i C_i, \sum_{i=1}^{n}\beta_i U_i)$ is obtained by using the heuristic WSPT. The transition probability p_{ij}^{k} that job j is selected by ant k to be processed a position i in the sequence informally given by:

$$p_{ij}^{k} = \begin{cases} \dfrac{(\tau_{ij})^{\lambda}(\varepsilon_{ij})^{\varphi}}{\sum_{j\in S_j^*}(\tau_{ij}^*)^{\lambda}(\varepsilon_{ij}^*)^{\varphi}} & \text{if} \quad j^* \in S_j^* \\ 0 & o.w \end{cases}$$

where S_j^* is the set of unscheduled jobs at position i and λ, φ are two parameters that weight the relative importance of the pheromone trial the heuristic information.

Step(3) All solution are evaluated and the best solution is improved by using AH.

Step(4) (Pheromone update) After an ant has selected the next job j a local pheromone update is performed at element (i,j) of the pheromone matrix according to

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\tau_{\circ} \qquad \ldots (1)$$

For some constant $\rho, 0 < \rho \leq 1$ and where $\tau_{\circ} = \frac{1}{m\, f(\sigma_{SWPT})}$ where $f(\sigma_{SWPT})$ is the value of $1\,||Lex\,(\sum_{i=1}^{n}\alpha_i C_i, \sum_{i=1}^{n}\beta_i U_i)$ problem when the jobs are ordered according WSPT. At the end of an iteration of the algorithm once all the ants have built a solution, pheromone is added to the arcs used by the ant that found the best tour from the beginning of the trial. This updating rule is called the global updating rule of pheromone

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij} \qquad \ldots (2)$$

Where $0 < \rho < 1$ is a pheromone decay parameter and $\Delta\tau_{ij}$ equal to

$$\Delta\tau_{ij} = \begin{cases} \dfrac{1}{\text{best cost function}} & \text{if} \quad (i,j) \in \text{best sequence} \\ 0 & \text{o.w} \end{cases} \qquad \ldots (3)$$

Step(5) The termination condition used here is the same one described in section (5.1).

6. Using B&B algorithm to solve $1\,||Lex\,(\sum_{i=1}^{n}\alpha_i C_i, \sum_{i=1}^{n}\beta_i U_i)$ problem :

Now give the main feature of our B&B algorithms, which are used for solving the $1\,||Lex\,(\sum_{i=1}^{n}\alpha_i C_i, \sum_{i=1}^{n}\beta_i U_i)$ problem. The algorithms starts by reorder (renumbered) the jobs according to WSPT and calculated sum weighted completion time and sum penalty number of late jobs. Prior to their application at the root node of search tree to obtain an UB by using the heuristic method, introduced in section (3.1) and put it as UB. Also at the root node of search tree an initial lower bound INLB on the cost of an optimal schedule is given in section (3.2). The active new search procedure for single machine with $Lex\,(\sum_{i=1}^{n}\alpha_i C_i, \sum_{i=1}^{n}\beta_i U_i)$ problem is used to select a node from which to branch.

7. Computational results :

The B&B algorithm was test by coding it in Microsoft FORTRAN Power Station and the local search methods were tested by coding them in Matlab R2008a and runs on a Pentium IV at 3.33 GHz, 512 MB computer.

The tested problem instances are generated as follows:

For $n = 10, 20, 30, 40, 50, 100, 200, 500, 1000, 2000 \& 5000$ and integer p_i for $i \in N = \{1, 2, \dots, n\}$ is generated by randomly selecting integers from interval $[1, 10]$ an integer weights α_i and lateness penalty β_i are drawn from distribution in the range $[1, 10]$. An integer due date d_i was generated for each job j in the same way as those of Abdul Razaq and Potts [1].

7.1 Comparative computation results for B&B method and local search :

In this section we are going to compare the results which we have reached in B&B method.

In table (1) we compare the number of nodes for the solved problem in B&B for $n = 10, 20, 30, 40$ for problem $1 \mid\mid Lex\ (\sum_{i=1}^{n} \alpha_i C_i, \sum_{i=1}^{n} \beta_i U_i)$ where NS1 is the number of problems solved that required not more than (200) nodes, NS2 is the number of problems solved that required over (200) nodes and not more than (1000) nodes, NS3 is the number of problems solved that required over (1000) nodes, finally NU explains the number of unsolved problems that require more than $(10{,}000{,}000)$ nodes.

From our computational experience with B&B algorithm and by the results of table (1) we observe that algorithm appears capable of solving problem with up to (40) jobs satisfactory for problem $1 \mid\mid Lex\ (\sum_{i=1}^{n} \alpha_i C_i, \sum_{i=1}^{n} \beta_i U_i)$.

Table (1) Average number of nodes (Ave. nod) for $1 \mid\mid Lex\ (\sum_{i=1}^{n} \alpha_i C_i, \sum_{i=1}^{n} \beta_i U_i)$ problem

N	ACT	Ave. nod	NS1	NS2	NS3	NU
10	1.11 E-05	38.000	15	/	/	/
20	1.22 E-04	4757.533	6	8	1	/
30	4.244 E-3	16145.33	/	/	15	/
40	6.3984	2012135.9	/	/	13	2

ACT: Average computation time (in second).

NS1: Number of solved problems that require not more than 200 nodes.

NS2: Number of solved problems that require more than 200 nodes and not more than 1000 nodes.

NS3: Number of solved problems that require more than 1000 nodes and not more than $10{,}000{,}000$ nodes.

NU: Number of unsolved problem.

It's clear from table (1) the average computation time and number of nodes increase as n increase.

In the following table (2) show the efficiency local search heuristic methods (Threshold Accepting (TA), Tabu search (TS) and Ant colony optimization (ACO)) have been approached in terms of comparable rate of value and time in case of using the algorithm AH.

Table (2) Compare between local search methods for $1 \mid \mid Lex\ (\sum_{i=1}^{n} \alpha_i C_i, \sum_{i=1}^{n} \beta_i U_i)$ problem

jobs		TS		TA		ACO		$\sum_{j \in SWPT} w_j U_j$
		API	AH	API	AH	API	AH	
10	AVf	19V	19V	19V	19V	19V	19V	20.6
	ATf	0.1282	0.1584	0.0367T	0.1390	0.1111	0.3413	
20	AVf	32.7	32.6V	33.2	32.7	33.6	32.7	34.9
	ATf	0.1409	0.1666	0.0366T	0.1348	0.1572	0.39324	
30	AVf	53.4	49.9V	49.9V	49.9V	50.5	49.9V	56.5
	ATf	0.1614	0.2036	0.0373T	0.1366	0.2089	0.39838	
40	AVf	44.9	42.4V	42.8	42.9	43.1	42.4V	47.8
	ATf	0.1769	0.2279	0.0361T	0.1372	0.2454	0.4302	
50	AVf	80.4	75.7V	76.6	76.5	76.5	75.7V	82.1
	ATf	0.1941	0.2366	0.0390T	0.1409	0.3054	0.5930	
100	AVf	188	182.6	185	184.3	185	182.2V	189.3
	ATf	0.2780	0.4966	0.0403T	0.1639	0.6596	1.0022	
200	AVf	381.1	373V	375.6	376	376.1	375.6	382.3
	ATf	0.4602	1.0473	0.0591V	0.2257	1.3547	2.0209	
500	AVf	593.3	592.1	593.8	590.1V	592	591.8	594.7
	ATf	1.0055	3.1581	0.074T	0.3595	5.1288	5.7562	

AVf : Average value objective function.
ATf : Average time objective function (in second).
V: best value.
T: best time.
TA: Threshold accepting.
TS: Tabu search.
ACO: Ant colony optimization.
Table (2) show that Tabu gives the best values with the AH algorithm neighbourhoods and Threshold accepting gives the best times with the API neighbourhood.
In Table (3) showed that the ACO took along time when jobs greater than and equal to 1000 jobs and the same thing for the TS in 5000 jobs, then put the condition when the operation calculate more than 30 sec stopped and stated number of loops which its took.

Table (3) Compare between local search methods for $1 \mid \mid Lex\ (\sum_{i=1}^{n} \alpha_i C_i, \sum_{i=1}^{n} \beta_i U_i)$ problem for high jobs

jobs		TS		TH		AC		$\sum_{j \in SWPT} w_j U_j$
		API	AH	API	AH	API	AH	
1000	AVf	1200.5	1198.4	1199.5	1197.9V	1199.6	1199.2	1202
	ATf	2.0336	6.5211	0.1644T	0.5845	24.76	21.5928	
	ALf	500	500	500	500	450.8	490.1	
2000	AVf	3884.4	3884.7	3887	3882.4V	3885.9	3884.6	3887.3
	ATf	3.9614	17.2749	0.2559T	1.0418	26.1959	30	
	ALf	500	500	500	500	182.8	114.1	
5000	AVf	9731.3	9732.5	9734.9	9731.3V	9733.9	9733.6	9734.9
	ATf	10.548	30	0.9796T	3.1311	31.478	31.8337	
	ALf	500	285.3	500	500	59.6	38.2	

ALf : Average number of loops.

Table (3) show that TA gives the best values with the AH algorithm neighbourhoods and gives the best times with the API neighbourhood.

7.2 Efficiency of local search algorithms :

The computational times of all algorithms for the $1||Lex\left(\sum_{i=1}^{n}\alpha_i C_i, \sum_{i=1}^{n}\beta_i U_i\right)$ problem are approximately (except Ant colony algorithm), since the computational time of ACO is very large as compared with computational time of neighbourhood search methods. Indeed this difference of time comes from the way that uses to generate the new sequence in each method. Where the neighbourhood search algorithms use API, AH neighbour which needs small time to perform its procedure, but the computational time of Ant colony algorithm is gradually increasing with the increase of problem size, since the procedure of ACO dependent on the accumulated pheromone on each node.

8. Conclusion :

This paper has developed a number of solution procedures for machine scheduling minimizing $Lex\left(\sum_{i=1}^{n}\alpha_i C_i, \sum_{i=1}^{n}\beta_i U_i\right)$. There procedure can be subdivided into implicit enumerative B&B algorithm and local search heuristic.

The B&B method firstly is used to solve this problem with algorithm to find upper bound and lower bound which is proposed. Several dominance rules are used to prune the search tree in B&B are proved.

The local search methods used to solve all the large problems the result show the robustness and flexibility of local search heuristics.

Future work Some suggestions for future research are described as follows:

First, the extension the propose of the exact for $1||Lex\left(\sum_{i=1}^{n}\alpha_i C_i, \sum_{i=1}^{n}\beta_i U_i\right)$ problem by driving a good lower bound or using the dominance rule in branch and bound algorithm.

Second, using the local search heuristic should be explored finding an improvement potential of various polynomially bounded scheduling heuristic.

References :

[1] T.S. Abdul Razaq and C. Potts. Dynamic programming state-space relaxation for single machine scheduling. European Journal of Operational Research Society. 39:141–152, 1999.

[2] M.Azizoglu, S. Kondakci and M. Köksalan. Single machine scheduling with maximum earliness and number tardy. Computers & Industrial Engineering. 45:257–268, 1986.

[3] S. P. Bansal. single machine scheduling to minimizing weighted sum of completion times with secondary criteria a branch and bounded approach. European Journal of Operational Research Society. 5:177–181, 1980.

[4] D. Biskup and M. Feldmann. Bench marks for scheduling on a single machine against restrictive and unrestrictive common due dates. Computer and OR. 28:787–801, 2005.

[5] M. Burns. scheduling to minimize the weighted sum of completion time with secondary criteria. Naval Research Logistic. 23:125–129, 1976.

[6] M. Dorigo and L.M. Gambardla. Ant algorithm for discrete optimization. Massachusetts Institute of Technology Artificial Life. 5:137–172, 1999.

[7] M. Feldmann and D. Biskup. Single machine scheduling for minimizing earliness and tardiness penalties by Meta-heuristic approaches. Computers and Industrial Engineering. 44:307–323, 2003.

[8] J. M. Framinan and R. Leisten. A heuristic for scheduling a permutation flow shop with Makespan objective subject to maximum tardiness. International Journal of Production Economics, 99: 28–40, 2006.

[9] F. Glover. Tabu search part I. ORSA Journal on Computing. 1:190–206, 1989.

[10] R. L. Graham, E. L. Lawler, J. K. Lenstra and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling. A surrey; Annals of Discrete Mathematics. 5:287–326, 1979.
[11] Guohua Wan, P. Benjamin and C. Yen. Single machine scheduling to minimize total weighted earliness subject to minimal number of tardy jobs. European Journal of Operational Research in Press. Available on line 1, 2008.
[12] H. Heck, S. Roberts. A note on the extension of a result on scheduling with secondary criteria. Naval Research Logistics, 19: 403–405,1972.
[13] C. M. Hino, D. P. Ronconi and A. B. Mendes. Minimizing earliness and tardiness penalties in a single machine problem with a common due date. European Journal of Operational Research. 160:190–201, 2005.
[14] S. M. Johnson. Optimal two and three stage production schedules with setup time included. Naval Research Logistics Quart. 1:61–68, 1954.
[15] S. Miya Zaki. One machine scheduling problem with dual criteria. Journal of Operational Research of Japan. 24:37-50, 1981.
[16] A. Nagar, S. S. Heragu and J. Haddock. Multiple and bicriteria scheduling. A literature Survey; European Journal of Operational Research. 81:88–104, 1995.
[17] S. Pathumrakul and P. J. Egbelu. An algorithm for minimizing weighted earliness penalty in assembly job shops. International of Production Economics. 103:230–245, 2006.
[18] M. L. Pinedo. Scheduling theory. Algorithm and system. Prentice Hull. 2002.
[19] T. Sen and S. K. Gupta. A branch and bound procedure to solve bicriteria scheduling problem. IIE Transactions. 15:84–88, 1983.
[20] W. E. Smith. Various optimizers for single–stage production. Naval Research Logistics. 3:59–66, 1956.
[21] V. T'kindt and J. –C. Billaut. Multicriteria scheduling theory models and algorithms. Springer, Berlin. 2002.
[22] Wei–Yan Chan and Gwo–Ji Sheen. Single machine scheduling with multiple perform measure: Minimizing job dependent earliness and tardiness subject to the number of tardy jobs. International Journal of Production Economics. 109:214–229, 2007.
[23] J. M. S. Valente R. A. F. Alves. Heuristic for the early/tardy scheduling problem with release dates. International Journal of Production Economics. 106:261–274, 2007.