ISSN 1991-

الترقيم الدولي ٨٦٩٠ - ١٩٩١
8690

Website: http://jsci.utq.edu.iq Email: utjsci@utq.edu.iq

# Using Ant Colony Algorithm to solve NP-Complete problems

Waffa Abdul-Abbas

Mathematical Department - College of Education - University of Thi-Qar

## Abstract

We consider the problem of scheduling jobs on a single machine about a common due date. Our objective is to determine the common due date and processing sequence of new jobs together with the re-sequencing of old jobs which minimize the sum of jobs' earliness/tardiness, completion time penalties and due date related penalty. We drive properties that can be used to find the optimal common due date and processing sequence. Since our problem is NP-hard, we propose Ant colony algorithm (ACO) to solve the problem efficiently. Results from computational study reveal that Ant colony algorithm (ACO) can

## المستخلص

تناولنا في هذا البحث نوعا من مسائل جدولة الماكنه الواحدة و بموعد تسليم ثابت. الهدف من المسالة هو حساب موعد التسليم وإيجاد متتابعة النتاجات القديمة وإعادة ترتيب النتاجات الجديدة لتصغير المجموع ألوزني النتاجات التبكير والتأخير وزمن الإتمام وعلاقة النتاجات بموعد تسليمها. وكذلك قمنا بإعداد بعض الخواص التي يمكن استخدامها لايجادافضل موعد تسليم ثابت ومتتابعة الإعمال. هذه المسألة من الأنواع المعقدة (NP-hard) لذلك فان وجود خوارزمية متعددة الحدود مقيدة (Polynomial bounded algorithm) لإيجاد الحل الأمثل غير ممكن لذلك افترضنا خوارزمية مستعمرة النمل لإيجاد حلول كفوءة للمسالة. النتائج الحسابية أظهرت إن خوارزمية مستعمرة النمل أعطت نتائج قريبه من الحل الأمثل.

## 1.Introduction

Many researchers have studied the scheduling models that deal with earliness and tardiness penalties since early 1980s.

In some formulations of E/T problems, the due dates are given as problem parameters; while in others the due dates are decision variables whose values are determined as part of the scheduling problem. Such problems arise when a firm is required to offer a due date to its customer during sales negotiations or after order receiving. Cheng and Gupta [2] present a comprehensive literature survey involving due date determination decisions. A number of

authors have studied the E/T models where all jobs share a common due date, but the due date is allowed to be a decision variable. Common due date models are relevant for certain situations where several jobs constitute a single customer order, multiple fabrication operations come together for assembly operations, or material handling economies dictate infrequent shipments such as monthly overseas shipments. E/T models considering common due date determination can be classified according to the types of penalties. Some models prescribe common penalties [4]; others permit differences between the earliness and tardiness penalties [9]; and still others permit differences among jobs [3]. The reader is also referred to Hall and Posner [6], Kahlbacher [7], and Weng and Ventura [12] for more recent publications.

On the other hand, most due date determination models assume that the production system carries no workload at each scheduling epoch. However, this assumption might be questionable in practice. In some situations, the scheduler would need to update the existing schedule (reschedule the sequence of old jobs) when some new jobs have arrived into the system. Hence it might be more realistic to consider the models of due date determination (for new jobs) with resequencing (for old jobs) at each scheduling epoch. This line of research can be found in Unal *et al*. [11] and Li and Cheng [8].

One way to extend typical due date determination E/T models is to include some additional penalties such as due date penalty and completion time penalty. These additional penalties are introduced by Panwalker *et al*. [9]. The due date penalty provides a disincentive for setting due dates too late. The completion time penalty provides an incentive to turn around orders rapidly. Note that the completion time penalty tends to induce shortest-first sequencing, whereas the earliness cost induces the reverse sequencing, at the start of scheduling.

In this paper, we consider the problem of scheduling jobs on a single machine about a common due date to minimize total penalty cost. Earliness and tardiness penalty rates are allowed to be arbitrary for each job, which is in contrast with most works on E/T models which assume somewhat simpler types of penalty rates such as a single penalty rate, a single early rate and a single tardy rate, or a single arbitrary rate for a job. Also two additional penalties (due date penalty and completion time penalty) are included in the model. In addition, rescheduling the sequence of old jobs is included in the model and the resequencing problem is considered at each scheduling epoch.

Hall and Posner [6] prove that a special case of E/T model with a single arbitrary rate for a job is NP-hard. Note that our model is a much more complex one and the problem is obviously NP-hard. We derive some scheduling properties to find the optimal common due date and job sequence. Then we propose and evaluate Ant colony algorithm (ACO) that takes advantage of some structural properties of our model, and find promising results from a computational study.

## 2. The model

To describe our model, let $n$ be the total number of jobs to be scheduled ( $r$ old jobs and $m$ new jobs). The processing time of job $i$ is known and is denoted as $P_i$, and $d_i$ denotes the due date of job $i$. As a result of scheduling decisions, job $i$ will be assigned a completion time, denoted $C_i$. Let $E_i$ and $T_i$ represent the earliness and tardiness, respectively, of job $i$. These quantities are defined as $Ei= \max\{0, d_i- C_i \}$ and $Ti=\max\{0, , C_i- d_i \}$ , respectively, where $d_i= d^0$ for old jobs $i=1, \ldots, r$ and $d_i= d$ for new jobs $i=r+1, \ldots, n$.

Note that the common due date of new jobs, *d*, is a decision variable and that of old jobs $d^0$, is given as a fixed value. Let $\alpha_i$ and $\beta_i$ denote the earliness and tardiness penalty rates for job *i*, respectively. In addition, let $\lambda$ and $\theta$ denote a single due date and a single completion time penalty rates for new jobs, respectively. Then, the objective function for a schedule *S* and a common due date *d* can be written as

$$\text{Min } f(d,S) = \sum_{i=1}^{n}(\alpha_i E_i + \beta_i T_i) + \sum_{i=r+1}^{n}(\lambda d + \theta C_i) \quad \text{.....(1)}$$

Our objective is to find the optimal common due date of new jobs and the optimal sequence of all jobs simultaneously to minimize the total weighted penalty cost. That is, the problem is to find *d** and *S** minimizing the objective function *f* (*d*,*S*) . We assume that the machine can process at most one job at a time and can not be kept idle as long as there are jobs to be processed. We also assume that no preemption of the jobs is allowed.

## 3. Properties of optimal common due date and job sequence

In this problem, we distinguish new jobs from old jobs, and try to find the optimal common due date of the new jobs and the optimal sequence of all the jobs in order to minimize total penalty cost. Note that we have only to determine the optimal job sequence (actually resequencing) for the old jobs since their common due date is given as a fixed value. Also note that the new jobs are assumed to be processed after the completion of all old jobs. The jobs, either old or new, are divided into an early job set *E* and a tardy job set *T*, whose jobs are completed exactly or ahead of the common due date ($Ci \le d$) and after the common due date ($Ci > d$), respectively. And if we let $S_1$ (or $S_2$) be a schedule of the old (or new) jobs such that *S* consists of $S_1$ and $S_2$, the objective function (1) can be rewritten as

$$f(d, S) = f_1(S_1) + f_2(d, S_2) \quad \text{..... (2)}$$

Where

$$f_1(S_1) = \sum_{i=1}^{n}(\alpha_i E_i + \beta_i T_i)$$

And

$$f_2(d, S_2) = \sum_{i=r+1}^{n}(\alpha_i E_i + \beta_i T_i + \lambda d + \theta C_i)$$

According to our assumption on scheduling principle, minimizing $f(d, S)$ is equivalent to minimizing $f_1(S_1)$ and $f_2(d, S_2)$ separately. Hence we try to obtain $S_1$* which minimizes $f_1(S_1)$ and ($d$*, $S_2$*) which minimizes $f_2(d, S_2)$, in order to obtain the optimal common due date ($d$*) and the optimal sequence ($S$*).

First consider the problem of sequencing old jobs to minimize $f_1(S_1)$.

**Property 1**: $S_1$* is determined by
(1) early jobs are sequenced in non-decreasing order of $\alpha_i / P_i$ ,
(2) tardy jobs are sequenced in non-increasing order of $\beta_i / P_i$ .

The property should be obvious only if the meanings of $\alpha_i/P_i$ and $\beta_i/P_i$ are appropriately interpreted. And hence, the proof is omitted.

Now let us consider the problem of determining the common due date and the job sequence of the new jobs to minimize $f_2(d, S_2)$. Following two properties are associated with this problem.

**Property 2**: For any given sequence $S_2$, the optimal common due date $d^*$ is the same time point as one of the new job completion times.

The property is straightforward to prove by contradiction. The proof is similar to the proof found in Hall and Posner [6] and is omitted.

**Property 3**: Given an early job set $E$ and a tardy job set $T$ under a common due date ($d$), $S_2^*$ is determined by

(1) for $J_i \in E$, jobs are sequenced in non-decreasing order of the value $(\alpha_i - \theta)/P_i$.

(2) for $J_i \in T$, jobs are sequenced in non-increasing order of the value $(\beta_i + \theta)/P_i$.

**Proof**

Consider a sequence $S_E$ of $e$ early jobs in which there exists a pair of adjacent jobs, $i$ and $j$, with $j$ following $i$, such that $(\alpha_i - \theta)/P_i > (\alpha_j - \theta)/P_j$. We can assume that the common due date, $d$, is equal to the completion time of the last job in $S_E$. Now construct a new sequence, $S_E'$, in which jobs $i$ and $j$ are interchanged in sequence and all other jobs finish at the same time as in $S_E$. Let $A$ denote the set of jobs preceding jobs $i$ and $j$ and $B$ denote the set of early jobs following $i$ and $j$, in both schedules. In addition, let $P_A$ and $P_B$ denote the total processing times of the jobs in set $A$ and $B$, respectively. Also, let $EC_k(S)$ represent the cost incurred by the $k$-th job in schedule $S$. We first show that $\sum_{k=1}^{e} EC_k$ is smaller under $S_E'$ than under $S_E$ which can be written as follows.

$$\sum_{k=1}^{e} EC_k(S_E) = \sum_{k\in A} EC_k(S_E) + EC_i(S_E) + EC_j(S_E) + \sum_{k\in B} EC_k(S_E)$$

$$= \sum_{k\in A} EC_k(S_E) + \alpha_i(P_j + P_B) + \theta(P_A + P_i)$$
$$+ \alpha_j P_B + \theta(P_A + P_i + P_j) + \sum_{k\in B} EC_k(S_E)$$

And

$$\sum_{k=1}^{e} EC_k(S_E') = \sum_{k\in A} EC_k(S_E') + EC_i(S_E') + EC_j(SE') + \sum_{k\in B} EC_k(S_E')$$

$$= \sum_{k\in A} EC_k(S_E') + \alpha_j(P_i + P_B) + \theta(P_A + P_j)$$
$$+ \alpha_i P_B + \theta(P_A + P_j + P_i) + \sum_{k \in B} EC_k(S_E')$$

Therefore,

$$\sum_{k=1}^{e} EC_k(S_E) - \sum_{k=1}^{e} EC_k(S_E') = P_j(\alpha_i - \theta) - P_i(\alpha_j - \theta) > 0$$

In words, interchanging jobs i and j reduces the relevant cost. Hence we can conclude that any sequence of early jobs not satisfying (1) can be improved with respect to $f_2$.

2) Consider a sequence $S_T$ of $t$ tardy jobs in which there exists a pair of adjacent jobs, $i$ and $j$, with $j$ following $i$, such that $(\beta_i + \theta)/P_i < (\beta_j + \theta)/P_j$. And assume that the common due date, d, is equal to the start time of the first job in $S_T$. From $S_T$, construct a new sequence, $S_T'$, in which jobs i and j are interchanged in sequence and all other jobs finish at the same time as in $S_T$. Let E, A and B denote the set of early jobs, the set of tardy jobs preceding jobs i and j and the set of jobs following i and j in both schedules, respectively. And let $P_E$ and $P_A$ denote the total processing time for the jobs in set E and A, respectively. Also, let $TC_k(S)$ represent the relevant cost of the k-th job in schedule S. Then,

$$\sum_{k=1}^{t} TC_k(S_T) = \sum_{k\in A} TC_k(S_T) + TC_i(S_T) + TC_j(S_T) + \sum_{k \in B} TC_k(S_T)$$
$$= \sum_{k\in A} TC_k(S_T) + \beta_i(P_A + P_i) + \theta(P_A + P_i + P_E)$$
$$+ \beta_j(P_A + P_i + p_j) + \theta(P_A + P_i + P_j + P_E) + \sum_{k \in B} TC_k(S_T)$$

And

$$\sum_{k=1}^{t} TC_k(S_T') = \sum_{k\in A} TC_k(S_T') + TC_i(S_T') + TC_j(S_T') + \sum_{k \in B} TC_k(S_T')$$
$$= \sum_{k\in A} TC_k(S_T') + \beta_j(P_A + P_j) + \theta(P_E + P_A + P_j)$$
$$+ \beta_i(P_A + P_j + P_i) + \theta(P_E + P_A + P_j + P_i) + \sum_{k \in B} TC_k(S_T')$$

Therefore,

$$\sum_{k=1}^{t} TC_k(S_T) - \sum_{k=1}^{t} TC_k(S_T') = P_i(\beta_j + \theta) - P_j(\beta_i + \theta) > 0$$

From the above equation, we can conclude that any sequence of tardy jobs that does not satisfy (2) can be improved with respect to $f_2$.

From properties 2 and 3, the optimal common due date and processing sequence for new jobs can be identified easily once the jobs are divided into an early job set *E* and a tardy job set *T*. In this case, the optimal common due date should be equal to the completion time of the last job in *E*. However, notice that there are $2^m$ possible cases of partitioning *m* new jobs into *E* and *T* sets. Hence it is critical to search 'good' *E* and *T* sets (those that will lead to a relatively better objective function value) in an effective way in order to solve the problem efficiently. For this, we propose and investigate Ant colony algorithm (ACO).

## 4.Local search for the problem

Let *N* be an arbitrary permutation of *n* jobs. For simplicity assume *N*= {1, 2, …, *n*}.

The objective is to find a schedule $\sigma = (\sigma(1),\sigma(2),...,\sigma(n))$ of the jobs that minimize the total cost $f(\sigma)$.

First we introduce some neighborhoods for a permutation problem, where the step of feasible solutions is given by the set of permutations of (*n*) jobs [10]

- **Pairwise Interchange (PI)** In a permutation $\sigma$ 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 the pairwise interchange neighbourhood. In a permutation $\sigma$, two adjacent jobs $\sigma(i)$ and $\sigma(i+1), (1 \le i \le n-1)$ are interchanged to generate a neighbour $\sigma$, where $|N(\sigma)| = (n-1)$.

### 4.1 Algorithm AH [5]:

**Step (1)** (Initialization) in this step a feasible solution $\sigma = (\sigma(1),\sigma(2),...,\sigma(n))$ is generated randomly.

**Step (2):** In this step the initial sequence $\sigma_{\text{int}}$ will be changed by the others neighborhoods and function values are calculated for every one i. e.

**a.** For the neighbor PI, have $\sigma_{PI}$ and $f_{PI}$.

**b.** For the neighbor API, have $\sigma_{API}$ and $f_{API}$.

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

**Step (4):** (Termination) the algorithm is terminated after (500) iterations at a feasible solution.

### 4.2 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 [1].

### 4.3 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 (4-1). The initial pheromone triad $\tau_{oij}(t)$ 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 *i*th position of the sequence. This selection depends on the pheromone trial $\tau_{ij}(t)$ and heuristic information $\varepsilon_{ij}$ for the our problem is obtained by using properties 2 and 3 if an early job set *E* and a tardy job set *T* are given.. 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:

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

where $S_j^{*}$ is the set of unscheduled jobs at position *i* and $\lambda,\varphi$ 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

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\rho\tau_o$$

For some constant $\rho, 0<\rho\le 1$ and where $\tau_o=\dfrac{1}{m\,f(\sigma_{rand})}$ where $f(\sigma_{rand})$ is the value of our problem when the jobs are ordered randomly. 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

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho\Delta\tau_o$$

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

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

**Step (5)** The termination condition used here is the same one described in section (4-1).

## 5. Computational results

In this section, we evaluate the quality of the ACO solutions for minimizing $f_2(d,S_2)$ by means of a computational study. The ACO developed in this study is coded it in Matlab R2008a and runs on a Pentium IV at 3.33 GHz, 512 MB computer. We tested the ACO on random problems, and compared the ACO solutions with the optimal solutions obtained using an enumeration method presented in section (3).We generated random problem instances for varying problem sizes of $m$=15, 20 and 25 jobs. The job processing times were selected from a discrete uniform distribution between 1 and 50. The earliness penalty rates ($\alpha_i$) were drawn from a discrete uniform distribution between 1 and 20. Also, a single due date penalty rate (λ) of 2 and a single completion time penalty rate (θ) of 1 were given for all $m$ new jobs. We used four types of tardiness penalty rate ($\beta_i$) distributions in our experiments. For type A, the tardiness penalty rates were drawn from a discrete uniform distribution between 1 and 20 (same as the earliness penalty rates); for type B, the tardiness penalty rates were generated from a discrete uniform distribution between 11 and 30 (twice as the earliness penalty rates); for type C, the tardiness penalty rates were generated from a discrete uniform distribution between 41 and 60 (five times as the earliness penalty rates); for type D, the tardiness penalty rates were generated from a discrete uniform distribution between 1 and 10 (half of the earliness penalty rates). Note that the case of type D is somewhat unusual but still possible. Ten problem instances were generated and solved for each combination of problem size $m$ and tardiness penalty rate types. Results of the computational study are summarized in table 1. The computational study shows that the proposed ACO is excellent in both solution quality and computation time.

Table 1(a) Results for type ○A problems

| M=15 | | | M=20 | | | M=25 | | |
|---|---|---|---|---|---|---|---|---|
| Optimal | ACO | AH | Optimal | ACO | AH | Optimal | ACO | AH |
| 23.367 | 23.367 | 23.367 | 43.635 | 43.635 | 43.635 | 65.437 | 65.513 | 65.547 |
| 21.706 | 21.783 | 21.831 | 37.539 | 37.539 | 37.539 | 68.764 | 68.764 | 68.764 |
| 32.542 | 32.542 | 32.542 | 49.553 | 49.553 | 49.628 | 74.376 | 74.376 | 74.376 |
| 29.752 | 29.752 | 29.752 | 35.539 | 35.539 | 35.539 | 75.552 | 75.638 | 75.689 |
| 20.132 | 20.255 | 20.181 | 40.819 | 40.885 | 40.899 | 78.964 | 78.964 | 78.964 |
| 35.162 | 35.162 | 35.162 | 35.912 | 35.912 | 35.912 | 66.845 | 66.845 | 66.845 |
| 23.069 | 23.069 | 23.207 | 40.278 | 40.278 | 40.278 | 60.099 | 60.217 | 60.236 |
| 28.329 | 28.329 | 28.399 | 32.331 | 32.331 | 32.331 | 52.129 | 52.129 | 52.129 |
| 16.346 | 16.346 | 16.346 | 30.342 | 30.531 | 30.436 | 80.135 | 80.135 | 80.204 |
| 30.912 | 30.912 | 30.912 | 39.604 | 39.684 | 39.711 | 85.651 | 85.651 | 85.651 |

Table 1(b) Results for type B problems

| M=15 | | | M=20 | | | M=25 | | |
|---|---|---|---|---|---|---|---|---|
| Optimal | ACO | AH | Optimal | ACO | AH | Optimal | ACO | AH |
| 17.073 | 17.073 | 17.073 | 27.864 | 27.864 | 27.864 | 38.892 | 38.892 | 38.892 |
| 19.512 | 19.512 | 19.512 | 22.867 | 22.867 | 22.867 | 40.486 | 40.486 | 40.486 |
| 13.065 | 13.065 | 13.065 | 23.421 | 23.421 | 23.421 | 43.179 | 43.179 | 43.179 |
| 13.721 | 13.721 | 13.901 | 28.952 | 28.974 | 28.982 | 38.650 | 38.650 | 38.650 |
| 22.933 | 22.933 | 22.933 | 14.637 | 14.637 | 14.637 | 31.315 | 31.376 | 31.382 |
| 18.654 | 18.654 | 18.654 | 31.727 | 31.727 | 31.727 | 33.737 | 33.737 | 33.737 |
| 16.438 | 16.572 | 16.742 | 23.544 | 23.813 | 23.879 | 41.615 | 41.615 | 41.615 |
| 10.655 | 10.655 | 10.655 | 24.038 | 24.038 | 24.038 | 40.874 | 40.886 | 40.905 |
| 12.912 | 12.912 | 12.986 | 23.925 | 23.925 | 23.925 | 38.811 | 38.811 | 38.874 |
| 19.765 | 19.765 | 19.765 | 21.303 | 21.303 | 21.462 | 34.439 | 34.439 | 34.439 |

Table 1(c) Results for type C problems

| M=15 | | | M=20 | | | M=25 | | |
|---|---|---|---|---|---|---|---|---|
| Optimal | ACO | AH | Optimal | ACO | AH | Optimal | ACO | AH |
| 36.126 | 36.126 | 36.126 | 43.659 | 43.684 | 43.851 | 68.467 | 68.467 | 68.467 |
| 28.709 | 28.709 | 28.709 | 36.288 | 36.288 | 36.288 | 53.865 | 53.865 | 53.865 |
| 24.225 | 24.225 | 24.225 | 49.362 | 49.457 | 49.872 | 61.749 | 61.749 | 61.749 |
| 38.608 | 38.662 | 38.687 | 38.026 | 38.026 | 38.026 | 53.507 | 53.643 | 53.922 |
| 37.462 | 37.462 | 37.462 | 46.661 | 46.661 | 46.688 | 58.397 | 58.397 | 58.416 |
| 30.110 | 30.110 | 30.110 | 42.719 | 42.719 | 42.719 | 65.840 | 65.840 | 65.840 |
| 27.726 | 27.777 | 27.862 | 39.154 | 39.242 | 39.231 | 67.068 | 67.068 | 67.068 |
| 38.465 | 38.465 | 38.551 | 44.209 | 44.209 | 44.209 | 69.859 | 69.896 | 69.905 |
| 27.370 | 27.370 | 27.407 | 32.938 | 32.938 | 32.938 | 61.133 | 61.133 | 61.133 |
| 31.601 | 31.601 | 31.601 | 40.988 | 40.988 | 40.988 | 57.769 | 57.769 | 57.804 |

Table 1(d) Results for type D problems

| M=15 | | | M=20 | | | M=25 | | |
|---|---|---|---|---|---|---|---|---|
| Optimal | ACO | AH | Optimal | ACO | AH | Optimal | ACO | AH |
| 34.202 | 34.202 | 34.202 | 64.841 | 64.902 | 64.977 | 89.240 | 89.240 | 89.240 |
| 31.193 | 31.193 | 31.193 | 69.252 | 69.252 | 69.252 | 84.631 | 84.631 | 84.631 |
| 41.636 | 41.636 | 41.636 | 78.521 | 78.644 | 78.961 | 75.038 | 75.174 | 75.302 |
| 44.458 | 44.483 | 44.517 | 69.228 | 69.228 | 69.228 | 71.439 | 71.439 | 71.439 |
| 49.875 | 49.952 | 49.981 | 64.332 | 64.332 | 64.518 | 83.153 | 83.153 | 83.153 |
| 34.436 | 34.436 | 34.436 | 79.592 | 79.592 | 79.592 | 73.741 | 73.741 | 73.788 |
| 36.268 | 36.268 | 36.268 | 74.852 | 74.852 | 74.852 | 84.944 | 84.944 | 84.944 |
| 42.417 | 42.417 | 42.417 | 74.924 | 74.983 | 74.961 | 76.137 | 76.137 | 76.639 |
| 47.094 | 47.243 | 47.548 | 64.305 | 64.305 | 64.433 | 87.341 | 87.341 | 87.431 |
| 49.189 | 49.189 | 49.189 | 79.328 | 79.328 | 79.328 | 81.938 | 81.938 | 81.938 |

## References

[1] Baker K and Scudder G (1989) On the assignment of optimal due dates. Journal of the Operational Research Society, 40, 93-95.

[2] Cheng TCE and Gupta M (1989) Survey of scheduling research involving due date determination decisions. European Journal of Operational Research, 38, 156-165.

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

[4] Emmons H (1987) Scheduling to a common due date on parallel common processors. Naval Research Logistics Quarterly, 34, 803-810.

[5] 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(1–2): 28–40, 2006.

[6] Hall N and Posner M (1991) Earliness-tardiness scheduling problems I:Weighted deviation of completion times about a common due date. Operations Research, 39, 836-846.

[7] Kahlbacher H (1993) Scheduling with monotonous earliness and tardiness penalties. European Journal of Operational Research, 64, 258-277.

[8] Li C and Cheng TCE (1999) Due-date determination with resequencing. IIE Transactions, 31, 183-188.

[9] Panwalker S, Smith M and Seidmann A (1982) Common due-date assignment to minimize total penalty for the one machine scheduling problem. Operations Research, 30, 391-399.

[10] R. Ruiz and T stützle. Simple and effective iterated greedy algorithm for permutation flowshop scheduling problem. European Journal of Operational Research. 177(3):2033–2049, 2007.

[11] Unal A, Uzsoy R and Kiran A (1997) Rescheduling on a single machine with part-type dependent setup times and deadlines. Annals of Operations Research, 70, 93-113.

[12] Weng M and Ventura J (1994) A note on ‘single machine scheduling for minimizing total cost with identical, asymmetrical earliness and tardiness penalties’. International Journal of Production Research, 32, 2727-2729.