Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling (CI-Sched 2007)

A Discrete Differential Evolution Algorithm for the Total Earliness and Tardiness Penalties with a Common Due Date on a Single-Machine

M. Fatih Tasgetiren, Quan-Ke Pan, Yun-Chia Liang, and P. N. Suganthan

***Abstract*— In this paper, a discrete differential evolution (DDE) algorithm is presented to solve the single machine total earliness and tardiness penalties with a common due date. A new binary swap mutation operator called *Bswap* is presented. In addition, the DDE algorithm is hybridized with a local search algorithm to further improve the performance of the DDE algorithm. The performance of the proposed DDE algorithm is tested on 280 benchmark instances ranging from 10 to 1000 jobs from the OR Library. The computational experiments showed that the proposed DDE algorithm has generated better results than those in the literature in terms of both solution quality and computational time.**

I. Introduction

Among all types of scheduling objectives, earliness and tardiness play important roles in the Just-in-Time (JIT) environment. In a JIT production system, a job completing earlier than its due date incurs an earliness penalty (inventory cost) whereas a job completing later incurs a tardiness penalty (imposed by customers). If the optimal sequence cannot be constructed without considering the value of the due date, the common due date is called restrictive. In a single machine scheduling problem with common due date, *n* number of jobs are available to be processed at time zero. Each job *j* has a processing time p_j and a common due date d. Preemption is not allowed and the objective is to sequence jobs with a restrictive common due date such that the sum of weighted earliness and tardiness penalties is minimized. That is,

$$f(S)=\sum_{j=1}^{n}\left(\alpha_j E_j+\beta_j T_j\right) \quad (1)$$

When the job *j* completes its operation before its due date, its earliness is given by $E_j=\max(0,d-C_j)$, where C_j is the completion time of the job *j*. On the other hand, if the job finishes its operation after its due date, its tardiness is given by $T_j=\max(0,C_j-d)$. Earliness and tardiness penalties are given by α_j and β_j, respectively.

It is well-known that for the case of restrictive common due date with general penalties, there exists an optimal schedule with the following properties:

1. No idle times are inserted between consecutive jobs [3]
2. The schedule is V-Shaped. In other words, jobs that are completed at or before the due date are sequenced in non-increasing order of the ratio p_j/α_j. On the other hand, jobs whose processing starts at or after the due date are sequenced in non-decreasing order of the ratio p_j/β_j. Note that there might be a straddling job, that is, the job that its processing is started before its due date and completed after its due date [2].
3. There is an optimal schedule in which either the processing of the first job starts at time zero or one job is completed at the due date [2].

The complexity of the restrictive common due-date problem is proved to be NP-complete in the ordinary sense [4]. Therefore, only small-sized instances of the single machine scheduling problem with a common due date can be solved to optimality with reasonable computational time using exact algorithms. When the problem size increases, the computational time of exact methods grows explosively. On the other hand, heuristic algorithms require generally acceptable time and memory requirements to reach a near-optimal or optimal solution. In past decades, most research focused on developing metaheuristic algorithms based on variants of local search methods. For example, James & Buchanan [5] and Wan and Yen [6] both employed a tabu search (TS) algorithm. Lee & Choi [7] proposed a genetic algorithm (GA), and Lee & Kim [8] developed a parallel genetic algorithm. Feldmann & Biskup [9] applied different metaheuristics such as evolutionary search (ES), simulated annealing (SA) and threshold accepting (TA) whereas M'Hallah [10] proposed a hybrid algorithm that combines GA, hill climbing (HC) and SA. Hino et al. [1] compared the performance of TS, GA, and their hybridization. Hendel & Sourd [11] employed neighborhood search based on the adjacent pairwise interchange (API) method. Very recently, A. C Nearchaou [12] proposed a differential evolution approach while a sequential exchange approach is presented by S-W Lin et al. [13].

M. Fatih Tasgetiren is with the Department of Operations Management and Busıness Statistics, Sultan Qaboos University, P.O.Box 20, Al Khod 123, Muscat, Sultanate of Oman: mfatih@squ.edu.om

Quan-Ke Pan is with the College of Computer Science, Liaocheng, University, Liaocheng, Shandong Province, 252059, P. R. China; qkpan@lctu.edu.cn

Yun Chia-Liang is with the Department of Industrial Engineering and Management, Yuan Ze University, 135 Yuan-Tung Road, Chungli, Taoyuan, 320 Taiwan, R.O.C. ycliang@saturn.yzu.edu.tw

P. N Suganthan is with the School of Electrical and Electronic Engineering Nanyang Technological University, Singapore 639798; *epnsugan@ntu.edu.sg* .

Differential evolution (DE) is one of the latest evolutionary optimization methods proposed by Storn & Price [14]. Like other evolutionary-type algorithms, DE is a population-based, stochastic global optimizer. In a DE algorithm, candidate solutions are represented as chromosomes based on floating-point numbers. In the mutation process of a DE algorithm, the weighted difference between two randomly selected population members is added to a third member to generate a mutated solution followed by a crossover operator to combine the mutated solution with the target solution so as to generate a trial solution. Then a selection operator is applied to compare the fitness function value of both competing solutions, namely, target and trial solutions to determine who can survive for the next generation.

Since DE was first introduced to solve the Chebychev polynomial fitting problem by Storn & Price [14, 15], it has been successfully applied in a variety of applications that can be found in Corne et al. [16], Lampinen [17], Babu & Onwubolu [18], and Price et al. [19].

The applications of DE on combinatorial optimization problems are still considered limited, but the advantages of DE include a simple structure, immediately accessible for practical applications, ease of implementation, speed to acquire solutions, and robustness that are sustained in the literature. However, the major obstacle of successfully applying a DE algorithm to combinatorial problems in the literature is due to its continuous nature. To remedy this drawback, this research proposes a novel discrete differential evolution (DDE) algorithm to solve the single machine total earliness and tardiness penalties with a common due date.

The paper is organized as follows. Section II introduces the modified MHRM heuristic. Section III gives the details of the proposed DDE algorithm and the local search employed. The computational results over benchmark problems are discussed in Section IV. Finally, Section V summarizes the concluding remarks.

II. THE MODIFIED MHRM HEURISTIC

A. MHRM Heuristic

Here we follow Pan et al. [20, 21]. In a single-machine with *n* number of jobs, at most one job can be completed on the due date. For this reason, there will be two sets of jobs: an early job set denoted by S^E where the jobs are completed before the due date and a tardy job set denoted by S^T where the jobs are completed after the due date. Consistent with the HRM heuristic [1], the MHRM heuristic consists of: (i) determining these two sets, (ii) constructing a sequence for each set, and (iii) setting the final schedule S as the concatenation of both sequences. In order to ensure that S will satisfy properties (1) and (2), there will be no idle time between consecutive jobs, and the sequences of S^E and S^T will be "\-shaped" and "/-shaped", respectively.

At each generation, the non-scheduled jobs with the maximum ratios p_j/α_j and p_j/β_j are considered for inclusion in one of the two sets. According to the distance between each job's possible completion time and the due date, just one of the jobs is included. Adjustments in the inserted idle time at the beginning of the sequence are also considered. Finally, when all jobs are scheduled, an attempt to satisfy the property (3) is made. Following notation is used:

P : set of jobs to be allocated

g : idle time inserted at the beginning of the schedule

S^E : set of jobs completed on the due date or earlier

S^T : set of jobs completed after the due date

S : schedule representation $S = \left(g, S^E, S^T\right)$

e : candidate job for S^E

t : candidate job for S^T

E^e : distance between the possible completion time of the job e and the due date

T^t : distance between the possible completion time of the job t and the due date

d^T : time window available for inserting a job in set S^T

d^E : time window available for inserting a job in set S^E

p_j : the processing time of job j

H : total processing time, $H = \sum_{j=1}^{n} p_j$

B. MHRM Heuristic Procedure

Step 1: Let $P = \{1,2,..,n\}$; $S^E = S^T = \Phi$,

$g = \max\{0, d - H \times \frac{1}{n} \sum_{j=1}^{n} \left(\frac{\beta_j}{\alpha_j + \beta_j} \right)\}$; $d^E = d - g$ and $d^T = g + H - d$.

Step 2: Set $e = \arg\max_{j \in p}\{p_j/\alpha_j\}$ and $t = \arg\max_{j \in p}\{p_j/\beta_j\}$ (in case of tie, select the job with the longest p_j).

Step 3: Set $E^e = d^E - p_e$ and $T^t = d^T$. If $E^e \le 0$ then go to step 5. If $T^t - p_t \le 0$ then go to step 6.

Step 4: Choose the job to be inserted:

If $E^e > T^t$ then $S^E = S^E + \{e\}$, $d^E = d^E - p_e$ and $P = P - \{e\}$.

If $E^e < T^t$ then $S^T = S^T + \{t\}$, $d^T = d^T - p_t$ and $P = P - \{t\}$.

If $E^e = T^t$ then if $\alpha_e > \beta_t$ then $S^T = S^T + \{t\}$, $d^T = d^T - p_t$ and $P = P - \{t\}$;

else $S^E = S^E + \{e\}$, $d^E = d^E - p_e$ and $P = P - \{e\}$.

Go to step 7.

Step 5: Adjustment of the idle time (end of the space before the due date):

If $g + E_e < 0$ then $S^T = S^T + \{t\}$, $d^T = d^T - p_t$
and $P = P - \{t\}$
Else: $S^{E'} = S^E$, $S^{T'} = S^T \cup P$, $g' = d\sum_{j \in S^{E'}} p_j$,
$S' = (g', S^{E'}, S^{T'})$;
$S^{E''} = S^E + \{e\}$, $S^{T''} = S^T \cup P - \{e\}$,
$g'' = d - \sum_{j \in S^{E''}} p_j$, $S'' = (g'', S^{E''}, S^{T''})$.
If $f(S^{E'}) \le f(S^{E''})$
then $S^T = S^T + \{t\}$, $d^E = 0$, $d^T = d^T - p_t + g' - g$,
$g = g'$ and $P = P - \{t\}$.
Else $S^E = S^E + \{e\}$, $d^E = 0$, $d^T = d^T + g'' - g$, $g = g''$
and $P = P - \{e\}$.
Go to step 7.

Step 6: Adjustment of the idle time (end of the space after the due date):

If $g < T^t$ then $S^E = S^E + \{e\}$, $d^E = d^E - p_e$
and $P = P - \{e\}$
Else $S^{T'} = S^T$, $S^{E'} = S^E \cup P$, $g' = d - \sum_{j \in S^{E'}} p_j$,
$S' = (g', S^{E'}, S^{T'})$;
$S^{T''} = S^T + \{t\}$, $S^{E''} = S^E \cup P - \{t\}$,
$g'' = d - \sum_{j \in S^{E''}} p_j$, $S'' = (g'', S^{E''}, S^{T''})$.
If $f(S^{E'}) \le f(S^{E''})$
then $S^E = S^E + \{e\}$, $d^T = 0$, $d^E = d^E - p_e + g - g'$,
$g = g'$, $P = P - \{e\}$;
Else $S^T = S^T + \{t\}$, $d^T = 0$, $d^E = d^E + g - g''$, $g = g''$
, $P = P - \{t\}$.

Step 7: If $P \neq \Phi$ then go to step 2.

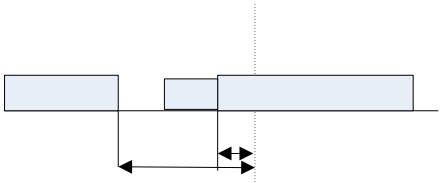
Step 8: If there is a straddling job (it must be the last job in S^T), then $S^{E'} = S^E$, $S^{T'} = S^T$,
$g' = d - \sum_{j \in S^{E'}} p_j$, $S' = (g', S^{E'}, S^{T'})$.
If $f(S') < f(S)$ then $g = g'$.

End of the algorithm.

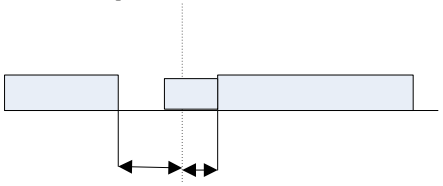
The MHRM heuristic is a modified version of HRM heuristic presented in Hino et al. [1]. The main difference between HRM and MHRM heuristics is due to the calculation of the inserted idle time g in Step 1 such that

$$g = \max\{0, d - H \times \sum_{j=1}^{n} \frac{\beta_j}{\alpha_j + \beta_j}) \quad (2)$$

instead of $g = \max\{0, d - 0.5 * H)$ in Hino et al. [1]. By doing so, the inserted idle time completely depends on the particular instance. It implies that if the total tardiness penalty of a particular instance is greater than the total earliness penalty of that instance ($\sum \beta_j > \sum \alpha_j$), the inserted idle time would be larger for that particular instance. Hence more jobs would be completed before the due date. In other words, more jobs would be early. Since $\sum \beta_j > \sum \alpha_j$, the total penalty imposed on the fitness function would be less than the one used in the HRM heuristic. In addition, the following modification is made in Step 3. As shown in Fig. 1, if the distance between the possible completion time of candidate job t and the due date is less than or equal to zero, both the start time and the completion time of the job t are before or at the due date, i.e., the job t is not a straddling job. In our algorithm, $T^t - p_t \le 0$ is employed instead of $T^t \le 0$ because $T^t - p_t \le 0$ implies that the job t is a straddling job. In this case, the adjustment of the idle time for the end of the space after the due date through Step 6 should be made. Accordingly, necessary modifications are also made in Step 5, 6, and 8.



a. End of the space after the due date in the HRM heuristic



b. End of the space after the due date in the MHRM heuristic

Fig.1. Difference Between HRM and MHRM Heuristics.

III. DISCRETE DIFFERENTIAL EVOLUTION ALGORITHM

Currently, there exist several mutation variations in DE. The *DE/rand/1/bin* schemes of Storn & Price [14] is presented below. The DE algorithm starts with initializing the initial population with the size of *NP*. Each individual has an *n*-dimentional vector with parameter values determined randomly and uniformly between predefined search range.

To generate a mutated individual, the DE mutates vectors from the target population by adding the weighted difference between two randomly selected target population members to a third member as follows:

$$v_{ij}^t = x_{aj}^{t-1} + F\left(x_{bj}^{t-1} - x_{cj}^{t-1}\right) \quad (3)$$

where a, b, and c are three randomly chosen individuals from the population such that $(a \neq b \neq c \in (1,.., NP))$. $F > 0$ is a mutation scale factor which affects the differential variation between two individuals.

Following the mutation phase, the crossover operator is applied to obtain the trial individual such that:

$$u_{ij}^{t} = \begin{cases} v_{ij}^{t}, if \quad r_{ij}^{t} \le CR \quad or \quad j = D_j \\ x_{ij}^{t-1}, \qquad Otherwise \end{cases} \tag{4}$$

where D_j refers to a randomly chosen dimension ($j=1,..,n$), which is used to ensure that at least one parameter of each trial individual u_{ij}^{t} differs from its counterpart in the previous generation u_{ij}^{t-1}. CR is a user-defined crossover constant in the range [0, 1], and r_{ij}^{t} is a uniform random number between 0 and 1. In other words, the trial individual is made up with some parameters of mutant individual, or at least one of the parameters randomly selected, and some other parameters of the target individual.

To decide whether or not the trial individual u_{ij}^{t} should be a member of the target population for the next generation, it is compared to its counterpart target individual x_{ij}^{t-1} at the previous generation. The selection is based on the survival of the fitness among the trial population and target population such that:

$$x_{ij}^{t} = \begin{cases} u_{ij}^{t}, if \quad f\left(u_{ij}^{t}\right) \le f\left(x_{ij}^{t-1}\right) \\ x_{ij}^{t-1}, otherwise \end{cases} \tag{5}$$

The pseudo code of the DE algorithm is given in Fig. 2.

Initialize parameters
Initialize target population
Evaluate target population
Do {
  Obtain mutant population
  Obtain trial population
  Evaluate trial population
  Make selection
  Apply local search (optional)
While (Not Termination)

Fig. 2. Standard DE Algorithm.

It is obvious that standard DE equations cannot be used to generate discrete/binary values since positions are real-valued. Instead we propose a DDE algorithm whose solutions are based on binary 0-1 values. In the DDE algorithm, the target population is constructed based on the binary 0-1 values as represented by $X_i = [X_1, X_2, ., X_{NP}]$. For the mutant population the following equations can be used:

$$V_i^{t} = m_1 \oplus F_k\left(X_i^{t-1}\right) \tag{6}$$

$$V_i^{t} = m_1 \oplus F_k\left(X_a^{t-1}\right) \tag{7}$$

$$V_i^{t} = m_1 \oplus F_k\left(G^{t-1}\right) \tag{8}$$

where X_a^{t-1} is randomly chosen individual from the target population; G^{t-1} is the global best solution; m_1 is the mutation probability; and F_k is the mutation operator with the mutation strength of *k*. In other words, a uniform random number *r* is generated between [0, 1]. If *r* is less than m_1 then the mutation operator is applied to generate the mutant individual. In this paper, the $V_i^{t} = m_1 \oplus F_k\left(G^{t-1}\right)$ version of mutation operators is employed in order to provide information exchange between the target population member and the global best. In the mutation equation, *k* represents the mutation strength, which is the key to the success of the algorithm. The higher the value of *k* is, the higher the possibility that the algorithm would have excessive randomness. On the other hand, the lower the value of *k* is, the lower the possibility that the algorithm would escape from the local minima. So care must be taken in the choice of the value of the mutation strength.

Following the mutation phase, the trial individual is obtained such that:

$$U_i^{t} = c_1 \oplus CR\left(X_i^{t-1}, V_i^{t}\right) \tag{9}$$

where *CR* is the crossover operator, and c_1 is the crossover probability. In other words, the *ith* individual is recombined with its corresponding mutant individual to generate the trial individual.

Finally, the selection is based on the survival of the fitness among the trial population and target population such that:

$$X_i^{t} = \begin{cases} U_i^{t} \quad if\ f\left(U_i^{t}\right) \le f\left(X_i^{t-1}\right) \\ X_i^{t-1} \qquad otherwise \end{cases} \quad 1 \le i \le NP \tag{10}$$

It is important to note that a binary solution representation is employed for the problem on hand. The x_{ij}^{t}, the *jth* dimension of the particle X_i^{t}, denotes a job; if $x_{ij}^{t} = 0$, the job *j* is completed before or at the due date, which belongs to the set S^E; if $x_{ij}^{t} = 1$, the job *j* is finished after the due date, which belongs to the set S^T. The binary representation is unique in terms of determining the early and tardy job sets. The individual representation is shown in Table I. From Table I, it is trivial to see that the jobs J_1, J_4 and J_6 belong to the early job set; and the jobs J_2, J_3 and J_5 belong to the tardy job set.

TABLE I
SOLUTION REPRESENTATION

Jobs, j	1	2	3	4	5	6
x_{ij}	0	1	1	0	1	0

To be employed in the DDE algorithm, the PTL crossover proposed in Pan et al. [21] is employed. An illustration of the two-cut PTL crossover is shown in Table I.

TABLE II
PTL CROSSOVER OPERATOR

Two-Cut PTL Crossover						Two-Cut PTL Crossover					
P1	0	**0**	**1**	0	1	P1	1	**0**	**1**	0	1
P2	1	1	1	0	1	P2	1	0	1	0	1
O1	**0**	**1**	1	0	1	O1	**0**	**1**	1	0	1
O2	1	1	1	**0**	**1**	O2	1	0	1	**0**	**1**

In the PTL crossover, a block of strings from the first parent is determined by two cut points randomly. This block is either moved to the right or left corner of the offspring. Then the offspring is filled out with the remaining strings from the second parent. In this paper, one of these two unique offspring is chosen randomly with an equal probability.

In addition, a binary swap (***Bswap***) mutation operator was presented in both mutation operator and the local search. The ***Bswap*** operator consists of two steps:

- Generate two random integers, *u* and *v*, in the range [*1,n*];
- if $x_{iu}^t = x_{iv}^t$, then $x_{iu}^t = (x_{iu}^t + 1) \bmod 2$;
  else $x_{iu}^t = (x_{iu}^t + 1) \bmod 2$ and $x_{iv}^t = (x_{iv}^t + 1) \bmod 2$.

To figure out how the individuals are updated in the DDE algorithm, an example is illustrated in Table III. Assume that the mutation and crossover probabilities are 1.0, two-cut PTL crossover and ***Bswap*** mutation operators are employed. Given the individual and the global best solution, the global best is first mutated. For example, in Table III, the dimension *u*=1 and *v*=3 are chosen randomly. Since both dimensions have the value of zero, the value of the dimension *u* is flipped from 0 to 1, thus generating the mutant individual V_i. Then the individual V_i is recombined with its corresponding individual in the target population to generate the trial individual U_i. Finally, the target individual X_i is compared to the trial individual U_i to determine which one would survive for the next generation based on the survival of fitness.

TABLE III
INDIVIDUAL UPDATE

X_i	1	0	1	0	1	
G	0	1	0	1	0	
Binary Swap Mutation						
G	**0**	1	**0**	1	0	
V_i	**1**	1	**0**	1	0	
Two-Cut PTL Crossover						f
X_i	1	**0**	**1**	0	1	45
V_i	1	1	0	1	0	
U_i	1	1	0	**0**	**1**	40
Selection						
$f(U_i) = 40 < f(X_i) = 45 \quad X_i = U_i, So$						
X_i	1	1	0	0	1	

After applying the DDE operators, the sets S^E and S^T are determined from the binary representation. Then every fitness calculation follows property (2). Note that the set S^T might contain a straddling job. If there is a straddling job, the first job in the early job set is started in time zero. After completing the last job of the early job set, the straddling job and the jobs in the tardy job set are sequenced. On the other hand, if there is no straddling job, the completion time of the last job in the early job set is matched with the due date and the processing in the tardy job set is followed immediately.

The neighborhood search in this study was based on the simple *Bswap* neighborhood. It should be noted that the following local search was applied to the global best solution, G^t, at each iteration *t*. The pseudo code of the local search is given in Fig.3.

```
LocalSearch(){
 s=perturbation(G^t)
 for (loop=1;loop≤size;loop++){
   flag=true
   while (true){
      s1 ←swap(s)
      if f(s1) ≤ f(s) then s ← s1;
     else flag=false}}
 if f(s) ≤ f(G^t)then G^t ← s}
```

Fig.3. Local Search Employed

In the neighborhood search algorithm above, *s* refers to the perturbed global best solution G^t at each generation *t*. That is, the global best solution is perturbed by swapping two jobs randomly; one from the tardy set, and the other from the early set. Then the ***Bswap*** operator was applied to the perturbed solution *s*. The size of the local search was set to size=min(30n,6000).

IV. EXPERIMENTAL RESULTS

The DDE algorithm was coded in Visual C++ and run on an Intel P IV 3.0 GHz PC with 512MB memory. Regarding the parameters of the DDE algorithms, m_1 and c_1 are taken as 0.8. Insert mutation operator (*F*) with the mutation strength of *k*=3 is used. As a crossover operator, the two-cut PTL is employed. The population size was 20. One of the solutions in the population is constructed with the MHRM heuristic, the rest is constructed randomly. The proposed DDE algorithm was applied to the benchmark problems that Biskup & Feldmann [2] developed a total of 280 instances ranging from 10 to 1000 jobs and restricting the common due date from 0.2 to 0.8 of the sum of all processing times. These instances can be downloaded at the OR-Library web site http://www.ms.ic.ac.uk/jeb/orlib/ schinfo.html.

10 runs were carried out for each problem instance to report the statistics based on the percentage relative deviations (Δ) from the upper bounds in Biskup & Feldmann [2]. To be more specific, Δ_{avg} was computed as follows:

$$\Delta_{avg} = \sum_{i=1}^{R} \left(\frac{(F_i - F_{ref}) * 100}{F_{ref}} \right) / R \qquad (11)$$

where F_i, F_{ref}, and R were the fitness function value generated by the DDE algorithm in each run, the reference fitness function value generated by Biskup & Feldmann [2], and the total number of runs, respectively. For convenience, $\Delta_{\min}$, $\Delta_{\max}$, and Δ_{std} denote the minimum, maximum, and standard deviation of percentage relative deviation in fitness

function value over R runs, respectively. For the computational effort consideration, t_{min}, t_{max}, t_{avg} ,and t_{std} denote the minimum, maximum, average time and the standard deviation until termination. The maximum generation number is fixed to 50 and the DDE algorithm is terminated when the global best solution is not improved in 10 consecutive generations.

The computational results of the MHRM heuristic and the DDE algorithm are given in Table VII and Table VIII, respectively. Table VII shows that the MHRM heuristic is superior to its counterpart HRM heuristic in terms of relative percent improvement.

TABLE IV
COMPARISON OF RESULTS (Δ_{avg})

		F&V		DPSO		DDE	
n	*h*	Δ_{avg}	Δ_{std}	Δ_{avg}	Δ_{std}	Δ_{avg}	Δ_{std}
20	0.2	-3.84	0.03	-3.83	0.02	-3.84	0.00
	0.4	-1.63	0.03	-1.62	0.02	-1.63	0.00
50	0.2	-5.65	0.03	-5.68	0.03	-5.68	0.01
	0.4	-4.64	0.02	-4.63	0.05	-4.65	0.01
100	0.2	-6.18	0.02	-6.18	0.02	-6.19	0.01
	0.4	-4.94	0.03	-4.90	0.04	-4.93	0.01
200	0.2	-5.73	0.02	-5.77	0.01	-5.76	0.01
	0.4	-3.79	0.02	-3.72	0.02	-3.72	0.02
500	0.2	-6.40	0.00	-6.41	0.01	-6.41	0.01
	0.4	-3.52	0.01	-3.54	0.01	-3.54	0.02
Avg		**-4.63**	**0.02**	**-4.63**	**0.02**	**-4.64**	**0.01**

Most recently, Hino et al. [1] developed a TS, GA and hybridization of both of them denoted as HTG and HGT. In addition, Pan et al. [21] developed a discrete particle swarm optimization (DPSO) algorithm to solve the same benchmark suite. Since Hino et al. [1] and Pan et al. [21] employed the same benchmark suite of Biskup & Feldmann [2], we compare our results to Feldmann & Biskup [9], Pan et al. [21] and Hino et al. [1]. Since the DDE algorithm is stochastic, its minimum, maximum, average, and standard deviation of the 10 runs for each instance should be given to evaluate its performance. However, Hino et al. [1] conducted 10 runs and picked the best out of 10 runs even though their tabu search contains a random component when updating the idle time. It implies that no information is at present about the average, and worst case behavior as well as the robustness of their algorithm. For this reason, we compare the minimum percentage relative deviation ($\Delta_{\min}$) of the DDE algorithm to Hino et al. [1] since only the minimum deviation is reported, and the average percentage relative deviation (Δ_{avg}) of the DDE algorithm to Feldmann & Biskup [9] where the average percentage relative deviations are given for only 20 to 500 jobs with h=0.2 and 0.4. Note that in Feldmann & Biskup [9], the average percentage improvements and their standard deviations are given using the best solution from all the heuristics, namely, ES, SA, TA and TAR.

TABLE V.
COMPARISON OF RESULTS (Δ_{min})

n	*h*	**DPSO**	**TS**	**GA**	**HTG**	**HGT**	**DDE**
	0.2	0.00	0.25	0.12	0.12	0.12	0.00
	0.4	0.00	0.24	0.19	0.19	0.19	0.00
	0.6	0.00	0.10	0.03	0.03	0.01	0.00
10	0.8	0.00	0.00	0.00	0.00	0.00	0.00
	0.2	-3.84	-3.84	-3.84	-3.84	-3.84	-3.84
	0.4	-1.63	-1.62	-1.62	-1.62	-1.62	-1.63
	0.6	-0.72	-0.71	-0.68	-0.71	-0.71	-0.72
20	0.8	-0.41	-0.41	-0.28	-0.41	-0.41	-0.41
	0.2	-5.70	-5.70	-5.68	-5.70	-5.70	-5.69
	0.4	-4.66	-4.66	-4.60	-4.66	-4.66	-4.66
	0.6	-0.34	-0.32	-0.31	-0.27	-0.31	-0.34
50	0.8	-0.24	-0.24	-0.19	-0.23	-0.23	-0.24
	0.2	-6.19	-6.19	-6.17	-6.19	-6.19	-6.19
	0.4	-4.94	-4.93	-4.91	-4.93	-4.93	-4.94
	0.6	-0.15	-0.01	-0.12	0.08	0.04	-0.15
100	0.8	-0.18	-0.15	-0.12	-0.08	-0.11	-0.18
	0.2	-5.78	-5.76	-5.74	-5.76	-5.76	-5.77
	0.4	-3.75	-3.74	-3.75	-3.75	-3.75	-3.75
	0.6	-0.15	-0.01	-0.13	0.37	0.07	-0.15
200	0.8	-0.15	-0.04	-0.14	0.26	0.07	-0.15
	0.2	-6.42	-6.41	-6.41	-6.41	-6.41	-6.43
	0.4	-3.56	-3.57	-3.58	-3.58	-3.58	-3.56
	0.6	-0.11	0.25	-0.11	0.73	0.15	-0.11
500	0.8	-0.11	0.21	-0.11	0.73	0.13	-0.11
	0.2	-6.76	-6.73	-6.75	-6.74	-6.74	-6.76
	0.4	-4.37	-4.39	-4.40	-4.39	-4.39	-4.38
	0.6	-0.06	1.01	-0.05	1.28	0.42	-0.06
1000	0.8	-0.06	1.13	-0.05	1.28	0.40	-0.06
Avg		**-2.15**	-2.01	-2.12	-1.94	-2.06	**-2.15**

As seen in Table IV, the DDE algorithm performed slightly better than Feldmann & Biskup [9] and Pan et al. [21] in terms of the average percentage relative improvement and the standard deviation. For this reason, one can conclude that the DDE algorithm was at least as good as all the metaheuristics tested in Feldmann and Biskup [9] and the DPSO algorithm in Pan et al. [21].

Table V summarizes the computational results to be compared to those in both Hino et al. [1] and Pan et al. [21]. As seen in Table V, the DDE algorithm outperforms almost all the metaheuristics of Hino et al. [1] in terms of the minimum percentage relative deviation. Besides the minimum and average performance of the DDE algorithm was better than all the metaheuristics in Hino et al. [1], it is also interesting to note that as seen in Table VIII, even the maximum percentage relative deviation of the DDE algorithm was better than TS, HGT and HTG algorithms of Hino et al. [1]. In other words, the worst case performance of the DDE algorithm was better than Hino et al. [1]. In comparison of DDE to DPSO, both algorithms generated the same average relative percent deviations. Regarding the CPU time requirement of the DDE algorithm, t_{max} was not more than 1.09 seconds on overall mean whereas Hino et. al. [1]

reported that their average CPU time requirement was 21.5 and 7.8 seconds for TS and hybrid strategies, respectively. In addition, the DDE algorithm was so robust such that the mean Δ_{std} was 0.00. To sum up, all the statistics show and prove that the DDE algorithm was superior to all the metaheuristics presented in Hino et al. [1].

The final comparison is due to HGT algorithm of Hino et al. [1] and TAR algorithm of Feldmann & Biskup [9]. In Feldmann & Biskup [9], the TAR algorithm showed superior performance among the five metaheuristics tested. When compared to the TAR and HGT algorithms, the performance of the DDE algorithm along with the DPSO algorithm was superior to both of them as shown in Table VI.

TABLE VI.
COMPARISON OF RESULTS (Δ_{min})

n	*h*	**DPSO**	**HGT**	**TAR**	**DDE**
10	0.2	0.00	0.12	0.00	0.00
	0.4	0.00	0.19	0.00	0.00
20	0.2	-3.84	-3.84	-3.84	-3.84
	0.4	-1.63	-1.62	-1.63	-1.63
50	0.2	-5.70	-5.70	-5.64	-5.69
	0.4	-4.66	-4.66	-4.62	-4.66
100	0.2	-6.19	-6.19	-6.16	-6.19
	0.4	-4.94	-4.93	-4.86	-4.94
200	0.2	-5.78	-5.76	-5.72	-5.77
	0.4	-3.75	-3.75	-3.63	-3.75
500	0.2	-6.42	-6.41	-6.39	-6.43
	0.4	-3.56	-3.58	-3.49	-3.56
1000	0.2	-6.76	-6.74	-6.72	-6.76
	0.4	-4.37	-4.39	-4.29	-4.38
Avg		**-4.11**	-4.09	-4.07	**-4.11**

V. CONCLUSIONS

DE is one of the recent evolutionary optimization methods. It has been widely used in a wide range of applications. To the best of our knowledge, this is the first reported application of DDE algorithm to the single-machine total earliness and tardiness penalties with a common due date problem in the literature. Unlike the standard DE, the DDE algorithm employs a binary solution representation and works on a discrete domain. In addition, the DDE algorithm is hybridized with the neighborhood search to solve well-known benchmark suites in the literature.

The proposed DDE algorithm was applied to the benchmark problems that Biskup and Feldmann [2] developed a total of 280 instances ranging from 10 to 1000 jobs and restricting the common due date from 0.2 to 0.8 of the sum of all processing times. The computational results show that the proposed DDE algorithm generated better results than the existing approaches in the literature.

As the future work, the authors have already solved the same problem with the DPSO algorithm. In addition, authors have also employed the binary version of continuous PSO algorithm and already solved the same problem. A detailed analysis of DPSO, DDE and Binary PSO algorithms with a variety of local search algorithms will be presented in the literature with comparisons to the very recent approaches such as A. C Nearchou [12] and S-W Lin et al. [13] in the near future.

Acknowledgement: Dr P. N. Suganthan acknowledges the financial support offered by the A*Star (Agency for Science, Technology and Research) under the grant # 052 101 0020.

REFERENCES

[1] 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*, vol. 160, pp. 190-201, 2005.

[2] D. Biskup and M. Feldmann, "Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates," *Computers & Operations Research*, vol. 28, pp. 787-801, 2001.

[3] T. C. E. Cheng, and H. G. Kahlbacher, "A proof for the longest/job/first policy in one/machine scheduling," Naval Research Logistics, vol. 38, pp. 715-720, 1990.

[4] N. G. Hall, W. Kubiak, and S. P. Sethi, "Earliness-tardiness scheduling problems II: weighted deviation of completion times about a restrictive common due date," *Operations Research*, vol. 39, no. 5, pp. 847-856, 1991.

[5] R. J. W. James and J. T. Buchanan, "Using tabu search to solve the common due date early/tardy machine scheduling problem," *Computers & Operations Research*, vol. 24, pp. 199-208, 1997.

[6] G. Wan and B. P. C. Yen, "Tabu search for single machine with distinct due windows and weighted earliness/tardiness penalties," *European Journal of Operational Research*, vol. 142, pp. 271-281, 2002.

[7] C. Y. Lee and J. Y. Choi, "A genetic algorithm for jobs sequencing with distinct due dates and general early-tardy penalty weights," *Computers & Operations Research*, vol. 22, pp. 857-869, 1995.

[8] C. Y. Lee and S. J. Kim, "Parallel genetic algorithms for the earliness/tardiness job scheduling problem with general penalty weights," *Computers & Industrial Engineering*, vol. 28, pp. 231-243, 1995.

[9] M. Feldmann and D. Biskup, "Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches," *Computers & Industrial Engineering*, vol. 44, pp. 307-323, 2003.

[10] R. M'Hallah, "Minimizing total earliness and tardiness on a single machine using a hybrid heuristic," *Computers & Operations Research*, to appear.

[11] Y. Hendel and F. Sourd, "Efficient neighborhood search for the one-machine earliness-tardiness scheduling problem," *European Journal of Operational Research*, to appear.

[12] A. C Nearchou, "A differential evolution approach for the common due date early/tardy job scheduling problem" *Computers & Operations Research*, to appear.

[13] S-W Lin, S-Y Chou, and K-C Ying, "A sequential exchange approach for minimizing earliness-tardiness penalties of single-machine scheduling with a common due date," *European Journal of Operational Research*, to appear.

[14] Storn, R. and Price, K. (1995) "Differential Evolution – a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces," *Technical Report TR-95-012*, ICSI, 1995.

[15] Storn, R. and Price, K. (1997) "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Space," *Journal of Global Optimization*, vol. 11, pp. 341-359.

[16] Corne, D., Dorigo, M., and Glover, F. (eds.) (1999) "*Part Two: Differential Evolution," New Ideas in Optimization*, McGraw-Hill, pp. 77-158.

[17] Lampinen, J. (2001) "A Bibliography of Differential Evolution Algorithm," *Technical Report*, Lappeenranta University of Technology, Department of Information Technology, Laboratory of Information Processing.

[18] Babu, B. V. and Onwubolu, G. C. (eds.) (2004) *New Optimization Techniques in Engineering*, Springer Verlag.

[19] Price, K., Storn, R., and Lampinen, J. (2006) *Differential Evolution – A Practical Approach to Global Optimization*, Springer-Verlag.

[20] Q. K. Pan, M. F. Tasgetiren, and Y. C. Liang, "A discrete particle swarm optimization algorithm for single machine total earliness and tadiness problem with a common due date", in *Proceedings of the World Congress on Evolutionary Computation, CEC2006*, Vancouver, Canada, pp. 11050-11057.

[21] Q. K. Pan, M. F. Tasgetiren, and Y. C. Liang, "Minimizing total earliness and tardiness penalties with a common due date on a single machine using a discrete particle swarm optimization algorithm", *Ant Colony Optimization and Swarm Intelligence. ANTS2006. LNCS 4150,* Springer-Verlag, 2006, pp. 460-467

TABLE VII.
STATISTICS FOR THE MHRM HEURISTIC (Δ)

	h	10	20	50	100	200	500	1000	Mean
HRM	0.2	1.53	-3.97	-5.33	-6.02	-5.63	-6.32	-6.68	**-4.50**
	0.4	8.68	0.46	-3.87	-4.42	-3.51	-3.46	-4.26	**-1.48**
	0.6	19.27	9.78	7.59	4.69	3.71	2.53	3.23	**7.26**
	0.8	22.97	13.52	8.10	4.70	3.71	2.53	3.23	**8.39**
	Mean	**13.11**	**5.17**	**1.62**	**-0.26**	**-0.43**	**-1.18**	**-1.12**	**2.42**
	h	10	20	50	100	200	500	1000	Mean
MHRM	0.2	1.00	-3.57	-5.45	-6.02	-5.62	-6.32	-6.69	**-4.67**
	0.4	5.91	-0.49	-4.03	-4.27	-3.52	-3.45	-4.27	**-2.02**
	0.6	2.77	2.02	1.51	1.50	1.71	1.41	1.55	**1.78**
	0.8	3.95	4.07	2.13	1.43	1.71	1.41	1.55	**2.32**
	Mean	**3.41**	**0.51**	**-1.46**	**-1.84**	**-1.43**	**-1.74**	**-1.97**	**-0.65**

TABLE VIII.
STATISTICS FOR THE DDE ALGORITHM

		Δ				Time Until Termination			
n	h	Δ_{min}	Δ_{max}	Δ_{avg}	Δ_{std}	t_{min}	t_{max}	t_{avg}	t_{std}
10	0.2	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00
	0.4	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.01
	0.6	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00
	0.8	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.01
20	0.2	-3.84	-3.84	-3.84	0.00	0.00	0.02	0.01	0.01
	0.4	-1.63	-1.63	-1.63	0.00	0.00	0.02	0.01	0.01
	0.6	-0.72	-0.72	-0.72	0.00	0.00	0.02	0.01	0.01
	0.8	-0.41	-0.41	-0.41	0.00	0.00	0.02	0.00	0.01
50	0.2	-5.69	-5.67	-5.68	0.01	0.02	0.04	0.03	0.01
	0.4	-4.66	-4.62	-4.65	0.01	0.03	0.05	0.04	0.01
	0.6	-0.34	-0.34	-0.34	0.00	0.02	0.04	0.03	0.01
	0.8	-0.24	-0.24	-0.24	0.00	0.02	0.04	0.03	0.00
100	0.2	-6.19	-6.17	-6.19	0.01	0.10	0.23	0.15	0.04
	0.4	-4.94	-4.91	-4.93	0.01	0.11	0.29	0.18	0.06
	0.6	-0.15	-0.15	-0.15	0.00	0.11	0.16	0.13	0.02
	0.8	-0.18	-0.18	-0.18	0.00	0.11	0.15	0.12	0.02
200	0.2	-5.77	-5.74	-5.76	0.01	0.21	0.60	0.34	0.13
	0.4	-3.75	-3.68	-3.72	0.02	0.24	0.71	0.42	0.15
	0.6	-0.15	-0.15	-0.15	0.00	0.24	0.44	0.32	0.07
	0.8	-0.15	-0.15	-0.15	0.00	0.24	0.44	0.32	0.06
500	0.2	-6.43	-6.40	-6.41	0.01	0.55	1.96	1.09	0.45
	0.4	-3.56	-3.52	-3.54	0.02	0.71	2.01	1.23	0.45
	0.6	-0.11	-0.11	-0.11	0.00	0.78	2.14	1.30	0.44
	0.8	-0.11	-0.11	-0.11	0.00	0.78	2.13	1.30	0.44
1000	0.2	-6.76	-6.74	-6.75	0.01	1.42	4.40	2.54	1.02
	0.4	-4.38	-4.34	-4.36	0.01	1.71	4.90	3.34	1.01
	0.6	-0.06	-0.06	-0.06	0.00	1.88	4.89	3.17	1.00
	0.8	-0.06	-0.06	-0.06	0.00	1.83	4.89	3.16	1.02
Mean		**-2.15**	**-2.14**	**-2.15**	**0.00**	**0.40**	**1.09**	**0.69**	**0.23**