Minimizing Total Earliness and Tardiness Penalties with a Common Due Date on a Single-Machine Using a Discrete Particle Swarm Optimization Algorithm

Quan-Ke Pan[1], M. Fatih Tasgetiren[2], and Yun-Chia Liang[3]

[1] College of Computer Science, Liaocheng University, Liaocheng, China
qkpan@lctu.edu.cn

[2] Department of Management, Fatih University, Istanbul, Turkey
ftasgetiren@fatih.edu.tr

[3] Department of Industrial Engineering and Management
Yuan Ze University, Taiwan
ycliang@saturn.yzu.edu.tw

Abstract. In this paper, a discrete particle swarm optimization (DPSO) algorithm is presented to solve the single machine total earliness and tardiness penalties with a common due date. A modified version of HRM heuristic presented by Hino et al. in [1], here we call it MHRM, is also presented to solve the problem. In addition, the DPSO algorithm is hybridized with the iterated local search (ILS) algorithm to further improve the solution quality. The performance of the proposed DPSO algorithm is tested on 280 benchmark instances ranging from 10 to 1000 jobs from the OR Library. The computational experiments showed that the proposed DPSO algorithm has generated better results, in terms of both percentage relative deviations from the upper bounds in Biskup and Feldmann [2] and computational time, than Hino et al. [1].

1 Introduction

In a single machine scheduling problem with a common due date, n jobs are available to be processed at time zero. Each job has a processing time and a common due date. Preemption is not allowed and the objective is to sequence jobs such that the sum of weighted earliness and tardiness penalties is minimized. That is,

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

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 of job j are given by parameters α_j and β_j, respectively. It is well-known that for the case of restrictive common due date with general penalties, there is an optimal schedule given 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, a 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]. Feldmann and Biskup [5] applied different meta-heuristics such as evolutionary search (ES), simulated annealing (SA) and threshold accepting (TA). In addition, Hino et al. [1] most recently compared the performance of TS, GA, and their hybridization. PSO was first introduced to optimize continuous nonlinear functions by Eberhart and Kennedy [6]. Authors have successfully proposed a DPSO algorithm to solve the no-wait flowshop scheduling in [7]. Based on the experience above, this study aims at solving the single-machine total earliness and tardiness penalties with a common due date problem by the DPSO algorithm.

Section 2 introduces the modified MHRM heuristic. Section 3 provides the details of the proposed DPSO algorithm. The computational results on benchmark instances are discussed in Section 4. Finally, Section 5 summarizes the concluding remarks.

2 Modified MHRM Heuristic

Consistent with the HRM heuristic in [1], the MHRM heuristic consists of: (i) determining the early and tardy job 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 V-shaped. The following notation consistent with Hino et al. [1] 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 $S = (g, S^E, S^T)$
e : candidate job for S^E
t : candidate job for S^T
E^e : distance between the possible completion time of job e and the due date

T^t : distance between the possible completion time of 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, $H = \sum_{j=1}^{n} p_j$

The procedure of the modified $MHRM$ heuristic is summarized as follows:

Step 1: Let $P = 1, 2, , n; S^E = S^T = \phi$,$g = max\{0, d - H \times \sum_{j=1}^{n} \frac{\beta_j}{\alpha_j+\beta_j}\}$; $d^E = d - g$ and $d^T = g + H - d$.
Step 2: Set $e = arg\ max_{j\epsilon p}\{p_j/\alpha_j\}$and $t = arg\ max_{j\epsilon p}\{p_j/\beta_j\}$ (in case of a tie, select the job with the longest p_j).
Step3: Set $E^e = d^E - p_e$ and $T^t = d^T$.
If $E^e \leq 0$, then go to step 5. If $T^t - p_t \leq 0$, 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 $S^T = S^E T + \{t\}$,$d^T = d^T - p_t$ 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 $S^{E'} = S^E$,$S^{T'} = S^T \cup P$,$g' = d - \sum_{j\epsilon S^{E'}} p_j$, $S' = (g', S^{E'}, S^{T'})$;
$S^{E''} = S^E + \{e\}$, $S^{T''} = S^T \cup P - \{e\}$, $g'' = d - \sum_{j\epsilon S^{E''}} p_j$,
$S'' = (g'', S^{E''}, S^{T''})$.
If $f(S^{E'}) \leq f(S^{E''})$, 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 $S^{T'} = S^T$,$S^{E'} = S^E \cup P$,$g' = d - \sum_{j\epsilon S^{E'}} p_j$, $S' = (g', S^{E'}, S^{T'})$;
$S^{T''} = S^T + \{t\}$, $S^{E''} = S^E \cup P - \{t\}$, $g'' = d - \sum_{j\epsilon S^{E''}} p_j$,
$S'' = (g'', S^{E''}, S^{T''})$.
If $f(S^{E'}) \leq f(S^{E''})$, then $S^E = S^E + \{e\}$,$d^T = 0$, $d^E = d^E - p_e + g' - g$, $g = g'$ and $P = P - \{e\}$;
else $S^T = S^T + \{t\}$, $d^T = 0$, $d^E = d^E + g - g''$, $g = g''$ and $P = P - \{t\}$.
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), then $S^{E'} = S^E$, $S^{T'} = S^T$, $g' = d - \sum_{j\epsilon S^{E'}} p_j$, $S' = (g', S^{E'}, S^{T'})$. If $f(S') \leq f(S)$ then $g = g'$.

The main difference between HRM and MHRM heuristics is due to the fact that the inserted idle time g is calculated in Step 1 such that :

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

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, that is, $\sum \beta_j > \sum \alpha_j$, 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. If the distance between the possible completion time of candidate job t and the due date is smaller or equal to zero, both the start time and the completion time of the job t will be before or at the due date, i.e., the job t is not a straddling job. In our algorithm, $T^t - p_t \leq 0$ is employed instead of $T^t \leq 0$ because $T^t - p_t \leq 0$ 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.

3 Discrete Particle Swarm Optimization Algorithm

It is obvious that standard PSO equations cannot be used to generate a discrete job permutation since position and velocity of particles are real-valued. Instead, Pan et al. [7] proposed a new method to update the position of particles as follows:

$$X_i^t = c_2 \oplus F_3(c_1 \oplus F_2(w \oplus F_1(X_i^{t-1}), P_i^{t-1}), G^{t-1}). \tag{3}$$

Given that λ_i^t and δ_i^t are temporary individuals, the update equation consists of three components: The first component is $\lambda_i^t = w \oplus F_1(X_i^{t-1})$, which represents the velocity of the particle. F_1 indicates the binary swap operator with the probability of w. In other words, a uniform random number r is generated between 0 and 1. If is less than w, then the swap operator is applied to generate a perturbed permutation of the particle by $\lambda_i^t = F_1(X_i^{t-1})$, otherwise current permutation is kept as $\lambda_i^t = X_i^{t-1}$. The second component is $\delta_i^t = c_1 \oplus F_2(\lambda_i^t, P_i^{t-1})$ where F_2 indicates the one-cut crossover operator with the probability of c_1. Note that λ_i^t and P_i^{t-1} will be the first and second parents for the crossover operator, respectively. It is resulted either in $\delta_i^t = F_2(\lambda_i^t, P_i^{t-1})$ or in $\delta_i^t = \lambda_i^t$ depending on the choice of a uniform random number. The third component is $X_i^t = c_2 \oplus F_3(\delta_i^t, G^{t-1})$ where F_3 indicates the two-cut crossover operator with the probability of c_2. It is resulted either in $X_i^t = F_3(\delta_i^t, G^{t-1})$ or in $X_i^t = \delta_i^t$ depending on the choice of a uniform random number. The pseudo code of the DPSO algorithm is given in Fig.1.

A binary solution representation is employed for the problem. The x_{ij}^t, the j^{th} 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.

After applying the DPSO 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.

```
Initialize parameters
Initialize population
Evaluate
Do{
  Find the personal best
  Find the global best
  Update position
  Evaluate
  Apply local search(optional)
}While (Not Termination)
```

Fig. 1. DPSO algorithm with a local search

```
$s_0 = G^t$
$s$=LocalSearch($s_0$)
Do{
  $s_1$=perturbation($s$)
  $s_2$=LocalSearch($s_1$)
  $s$=AcceptanceCriterion($s, s2$)
}While (Not Termination)
 if $f(s) < f(G^t)$ then $G^t = s$)
```

Fig. 2. Iterated local search algorithm

At the end of each iteration, the ILS algorithm is applied to the global best solution G^t to further improve the solution quality. The ILS algorithm in Fig.2 was based on the simple binary swap neighborhood. The perturbation strength was 5 binary swaps to avoid getting trapped at the local minima. In the $LocalSearch$ procedure, the binary swap operator was used with the size of $min(6n, 600)$ and the size of the $do - while$ loop was 10. The binary swap operator consists of two steps: (i) generate two random integers, a and b, in the range $[1, n]$; (ii) if $x_{ia}^t = x_{ib}^t$, then $x_{ia}^t = (x_{ia}^t + 1)mod2$; else $x_{ia}^t = (x_{ia}^t + 1)mod2$ and $x_{ib}^t = (x_{ib}^t + 1)mod2$.

4 Computational Results

The DPSO algorithm is coded in Visual C++ and run on an Intel P IV 2.4 GHz PC with 256MB memory. Regarding the parameters of the DPSO algorithm, the crossover probabilities were taken as $c_1 = c_2 = 0.8$, respectively. The swap probability was set to $w = 0.95$. One of the solutions in the population is constructed with the MHRM heuristic, the rest is constructed randomly. The proposed DPSO algorithm was applied to the benchmark problems developed in Biskup and Feldmann [2]. 10 runs were carried out for each problem instance and the average percentage relative deviation was computed as follows:

$$\Delta_{avg} = \sum_{i=1}^{R} \left(\frac{(F_i - F_{ref}) \times 100}{F_{ref}} \right) / R \tag{4}$$

where F_i,F_{ref}, and R were the fitness function value generated by the DPSO algorithm in each run, the reference upper bounds generated by Biskup and Feldmann [2], and the total number of runs, respectively. The maximum number of iterations was fixed to 50 and the algorithm was terminated when the global best solution was not improved in 10 consecutive iterations. The computational results of the MHRM heuristic are given in Table 1 where the MHRM heuristic is superior to its counterpart HRM heuristic in terms of relative percent improvement.

Most recently, Hino et. al. [1] developed a TS, GA and hybridization of both of them denoted as HTG and HGT. They employed the same benchmark suite of Biskup and Feldmann [2]. Table 2 summarizes the computational results of the DPSO and those in Hino et al. [1]. As seen in Table 2, the DPSO algorithm outperforms all the metaheuristics of Hino et al. [1] in terms of the minimum percentage relative devia-tion since the largest improvement of -2.15 on overall mean is achieved. Besides the average performance of the DPSO algorithm, it is also interesting to note that even the worst performance of the DPSO algorithm, i.e., the maximum percentage relative deviation, was better than TS, GA, HGT and HTG algorithms of Hino et al. [1]. Regarding the CPU time requirement of the DPSO algorithm, the maximum CPU time until termination was not more than 1.33 seconds on overall average whereas Hino et al. [1] reported that their average CPU time requirement was 21.5 and 7.8 seconds for TS and hybrid

Table 1. Statistics for the MHRM Heuristic

	h	10	20	50	100	200	500	1000	**Mean**
	0.2	1.53	-3.97	-5.33	-6.02	-5.63	-6.32	-6.68	-4.50
HRM	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**
	0.2	1.00	-3.57	-5.45	-6.02	-5.62	-6.32	-6.69	**-4.67**
MHRM	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**

strategies, respectively. To sum up, all the statistics show and prove that the DPSO algorithm was superior to all the metaheuristics presented in Hino et al. [1]. Note that the best results so far in the literature are reported in bold in Table 2.

Table 2. Statistics for the DPSO Algorithm

		DPSO				*TS*	*GA*	*HTG*	*HGT*
	h	Δ_{min}	Δ_{max}	Δ_{avg}	Δ_{std}	Δ_{min}	Δ_{min}	Δ_{min}	Δ_{min}
10	0.2	**0.00**	0.00	0.00	0.00	0.25	0.12	0.12	0.12
	0.4	**0.00**	0.00	0.00	0.00	0.24	0.19	0.19	0.19
	0.6	**0.00**	0.00	0.00	0.00	0.10	0.03	0.03	0.01
	0.8	**0.00**	0.00	0.00	0.00	**0.00**	**0.00**	**0.00**	**0.00**
20	0.2	**-3.84**	**-3.84**	**-3.84**	0.00	**-3.84**	**-3.84**	**-3.84**	**-3.84**
	0.4	**-1.63**	**-1.63**	**-1.63**	0.00	-1.62	-1.62	-1.62	-1.62
	0.6	**-0.72**	**-0.72**	**-0.72**	0.00	-0.71	-0.68	-0.71	-0.71
	0.8	**-0.41**	**-0.41**	**-0.41**	0.00	**-0.41**	-0.28	**-0.41**	**-0.41**
50	0.2	-5.68	-5.67	-5.68	0.01	**-5.70**	-5.68	**-5.70**	**-5.70**
	0.4	**-4.66**	-4.58	-4.64	0.03	**-4.66**	**-4.60**	**-4.66**	**-4.66**
	0.6	**-0.34**	**-0.34**	**-0.34**	0.00	-0.32	-0.31	-0.27	-0.31
	0.8	**-0.24**	**-0.24**	**-0.24**	0.00	**-0.24**	-0.19	-0.23	-0.23
100	0.2	**-6.19**	-6.16	-6.18	0.01	**-6.19**	-6.17	**-6.19**	**-6.19**
	0.4	**-4.94**	-4.88	-4.92	0.02	-4.93	-4.91	-4.93	-4.93
	0.6	**-0.15**	**-0.15**	**-0.15**	0.00	-0.01	-0.12	0.08	0.04
	0.8	-0.18	**-0.18**	**-0.18**	0.00	-0.15	-0.12	-0.08	-0.11
200	0.2	**-5.78**	-5.73	-5.76	0.02	-5.76	-5.74	-5.76	-5.76
	0.4	-3.74	-3.67	-3.72	0.03	-3.74	**-3.75**	**-3.75**	**-3.75**
	0.6	**-0.15**	**-0.15**	**-0.15**	0.00	-0.01	-0.13	0.37	0.07
	0.8	**-0.15**	**-0.15**	**-0.15**	0.00	-0.04	-0.14	0.26	0.07
500	0.2	**-6.42**	-6.39	-6.41	0.01	-6.41	-6.41	-6.41	-6.41
	0.4	-3.56	-3.49	-3.53	0.02	-3.57	**-3.58**	**-3.58**	**-3.58**
	0.6	**-0.11**	**-0.11**	**-0.11**	0.00	0.25	**-0.11**	0.73	0.15
	0.8	**-0.11**	**-0.11**	**-0.11**	0.00	0.21	**-0.11**	0.73	0.13
1000	0.2	**-6.76**	-6.73	-6.75	0.01	-6.73	-6.75	-6.74	-6.74
	0.4	-4.38	-4.32	-4.36	0.02	-4.39	**-4.40**	-4.39	-4.39
	0.6	**-0.06**	**-0.06**	**-0.06**	0.00	1.01	-0.05	1.28	0.42
	0.8	**-0.06**	**-0.06**	**-0.06**	0.00	1.13	-0.05	1.28	0.40
	Mean	**-2.15**	**-2.13**	**-2.15**	0.01	-2.01	-2.12	-1.94	-2.06

5 Conclusions

A modified version of the HRM heuristic with much better results is developed along with the discrete version of the PSO algorithm. The DPSO algorithm is hybridized with the ILS algorithm to further improve the computational results. The proposed DPSO algorithm was applied to 280 benchmark instances of Biskup and Feldmann [2]. The solution quality was evaluated according to the reference upper bounds generated by Biskup and Feldmann [2]. The computational results show that the proposed DPSO algorithm generated better results than those in Hino et. al. [1].

References

1. Hino C.M., Ronconi D.P., Mendes A.B.: Minimizing earliness and tardiness penalties in a single-machine problem with a common due date. European Journal of Operational Research, **160** (2005) 190–201
2. Biskup D., Feldmann M.: Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates. Computers & Operations Research, **28** (2001) 787–801
3. Cheng T.C.E., Kahlbacher H.G.: A proof for the longest-job-first policy in one-machine scheduling. Naval Research Logistics, **38** (1990) 715–720.
4. Hall N.G., Kubiak W., Sethi S.P.: Earliness-tardiness scheduling problems II: weighted- deviation of completion times about a restrictive common due date. Operations Research, **39**(5) (1991) 847–856
5. Feldmann M., Biskup D.: Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches. Computers & Industrial Engineering, **44** (2003) 307–323
6. Eberhart R.C., Kennedy J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan (1995) 39–43
7. Pan Q.K., Tasgetiren M.F., Liang Y.C.: A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem with makespan criterion. In: Proceedings of the International Workshop on UK Planning and Scheduling Special Interest Group (PLANSIG2005), City University, London, UK (2005) 34–43