Un-restricted Common Due-Date Problem with Controllable Processing Times

Linear Algorithm for a Given Job Sequence

Abhishek Awasthi[1], Jörg Lässig[1] and Oliver Kramer[2]

[1]*Department of Computer Science, University of Applied Sciences Zittau/Görlitz, Görlitz, Germany*

[2]*Department of Computing Science, Carl von Ossietzky University of Oldenburg, Oldenburg, Germany*

Keywords: Scheduling, Algorithms, Simulated Annealing, NP-Hard.

Abstract: This paper considers the un-restricted case of the Common Due-Date (CDD) problem with controllable processing times. The problem consists of scheduling jobs with controllable processing times on a single machine against a common due-date to minimize the overall earliness/tardiness and the compression penalties of the jobs. The objective of the problem is to find the processing sequence of jobs, the optimal reduction in the processing times of the jobs and their completion times. In this work, we first present and prove an essential property for the controllable processing time CDD problem for the un-restricted case along with an exact linear algorithm for optimizing a given job sequence for a single machine with a run-time complexity of $O(n)$, where n is the number of jobs. Henceforth, we implement our polynomial algorithm in conjunction with a modified Simulated Annealing (SA) algorithm and Threshold Accepting (TA) to obtain the optimal/best processing sequence while comparing the two heuristic approaches, as well. The implementation is carried out on appended CDD benchmark instances provided in the OR-library.

1 INTRODUCTION

The Common Due-Date scheduling problem involves sequencing and scheduling of jobs over machine(s) against a common due-date. Each job possesses a processing time and different penalties per unit time in case the job is completed before or after the due-date. For the controllable processing time case, in addition to the CDD, the processing times of some or all the jobs can be reduced to a certain minimum value at a cost of some penalty per unit of reduction. This controlling of the processing times can help the jobs to reduce their earliness/tardiness penalties if the penalties incurred due to the compressions are relatively smaller than the earliness/tardiness penalties. The objective of solving the problem is to obtain the optimal job sequence, final processing times of the jobs and the completion times of all the jobs to minimize the total weighted penalty. Generally speaking, there are two classes of common due-date problem, which have proven to be NP-hard, namely the restrictive and the un-restrictive CDD problem. In this work, we consider the un-restrictive case of the problem, where the common due-date is greater than or equal to the sum of the processing times of all the jobs and each job possesses different penalties. The CDD has already been proven to be NP-hard, and clearly the controllable case is NP-hard as well (Yunqiang et al., 2013; Biskup and Feldmann, 2001). For the controllable processing times common due-date problem, (Biskup and Cheng, 1999) studied it with constant penalties for earliness/tardiness and distinct penalties for compression. They also considered the penalty for the completion time of the jobs and proved the similarity of the problem to the assignment problem. (Shabtay and Steiner, 2007) made an extensive survey for scheduling with controllable processing times, covering research in this field from the last 25 years. (Wan, 2007) studied the common due window problem with controllable processing times with constant earliness/tardiness penalties and distinct compression costs and discussed some properties for the optimal solution along with a polynomial algorithm for solving the problem. (Tseng et al., 2009) studied the general CDD problem with compressible processing times with different due-dates and presented a heuristic algorithm to minimize the total tardiness and the compression penalties. (Nearchou, 2010) studied a slightly different version of the problem where the objective was to minimize the total weighted comple-

Awasthi A., Lässig J. and Kramer O..
Un-restricted Common Due-Date Problem with Controllable Processing Times - Linear Algorithm for a Given Job Sequence.
DOI: 10.5220/0005398205260534
In *Proceedings of the 17th International Conference on Enterprise Information Systems* (ICEIS-2015), pages 526-534
ISBN: 978-989-758-096-3

tion times and the compression costs and presented a population based metaheuristic algorithm, considering four different heuristic approaches namely, differential evolution, particle swarm optimization, genetic algorithms and evolution strategies. We would emphasize here that the CDD problem with controllable processing times for a fixed due-date has not been studied before for the asymmetric penalties. In this paper, we consider the single machine case for the un-restricted CDD problem with asymmetric penalties and controllable processing times with distinct linear costs. We make a theoretical study of the problem and first present an important property for this problem. We then propose an $O(n)$ exact polynomial algorithm to optimize a given job sequence on a single machine.

2 PROBLEM FORMULATION

In this Section, we present the mathematical notation of the common due-date problem with the controllable processing times. Let n be the number of jobs and d be the common due-date. Besides, p_i and m_i be the actual and minimum processing times, respectively with x_i being the actual reduction in the processing time for job i. Let, E_i, T_i and C_i be the earliness, tardiness and completion time of job i. And α_i, β_i and γ_i be the earliness, tardiness and compression penalties per time unit for any job i. Clearly, E_i and T_i can be expressed as $E_i = \max\{0, d - C_i\}$ and $T_i = \max\{0, C_i - d\}$ for $i = 1, 2, \ldots, n$. Then, the objective function of the problem can be expressed as,

$$\min \sum_{i=1}^{n} (E_i \cdot \alpha_i + T_i \cdot \beta_i + x_i \cdot \gamma_i) \,. \tag{1}$$

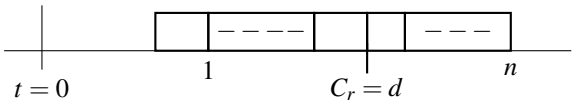

Figure 1: Assume that the rth job finishes at the due-date d in the optimal solution.

We now present some properties for both the CDD and CDD with controllable processing times. Let the solution value for the case when there is no compression of the processing times and the due-date lies at the completion time of job r, as shown in Figure 1, be Sol_r, then we have

$$Sol_r = \sum_{i=1}^{r-1} \sum_{j=i+1}^{r} p_j \alpha_i + \sum_{i=r+1}^{n} \sum_{j=r+1}^{i} p_j \beta_i \,, \tag{2}$$

where

$\sum_{i=1}^{r-1} \sum_{j=i+1}^{r} p_j$ = the total earliness for any job i and

$\sum_{i=r+1}^{n} \sum_{j=r+1}^{i} p_j$ = the total tardiness for any job i.

Let us assume that the reductions in the processing times in the optimal schedule be x_i for all $i = 1, 2, \ldots, n$. Then the objective function value (Sol'_r) when the due-date position is at C_r will be

$$Sol'_r = \sum_{i=1}^{r-1} \sum_{j=i+1}^{r} (p_j - x_j)\alpha_i + \sum_{i=r+1}^{n} \sum_{j=r+1}^{i} (p_j - x_j)\beta_i + \sum_{j=1}^{n} x_j \gamma_j \,. \tag{3}$$

We first present and prove an important property regarding the amount of compression of the processing times of the jobs.

Property 1. *If controlling the processing times fetches a better solution, then the compression of the processing times should be to their minimum value.*

Proof. If the compression of the processing times fetches a better solution, then we have $Sol'_r \leq Sol_r$. Using Equation (2) and (3), we obtain

$$\sum_{i=1}^{r-1} \sum_{j=i+1}^{r} x_j \alpha_i + \sum_{i=r+1}^{n} \sum_{j=r+1}^{i} x_j \beta_i - \sum_{j=1}^{n} x_j \gamma_j \geq 0 \,. \tag{4}$$

Let us assume that instead of reducing the processing times by x_j, we reduce them by y_j, where $y_j < x_j$ $\forall j = 1, 2, \ldots,$. Let the solution value for this case be $Sol'_{r'}$ and $x_j - y_j = \delta_j$. If $Sol'_{r'} < Sol'_r$, then with some manipulation of the terms we get

$$\sum_{i=1}^{r-1} \sum_{j=i+1}^{r} \delta_j \alpha_i + \sum_{i=r+1}^{n} \sum_{j=r+1}^{i} \delta_j \beta_i - \sum_{j=1}^{n} \delta_j \gamma_j \leq 0 \,. \tag{5}$$

Since $x_j \geq 0 \; \forall j = 1, 2, \ldots, n$, Equation (4) should also hold for any $\delta_j > 0$. However, Equation (5) is a contradiction. Hence, our assumption that $Sol'_{r'} < Sol'_r$ is wrong. This proves that the solution value only improves if we reduce the processing times furthermore, which in turn shows that the best solution value will be obtained for maximum possible compression of the processing times. □

Let the solution value for the CDD when the $(r+1)$th job ends at the due-date be Sol_{r+1} as shown in Figure 2, then

$$Sol_{r+1} = \sum_{i=1}^{r} \left(\sum_{j=i+1}^{r+1} p_j \right) \alpha_i + \sum_{i=r+2}^{n} \left(\sum_{j=r+2}^{i} p_j \right) \beta_i \,. \tag{6}$$

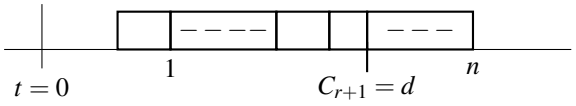

Figure 2: Schedule with the completion time of job $r+1$ lying at the due-date, $C_{r+1} = d$.

For the optimal reductions x_i in the processing times, the solution value Sol'_{r+1} will be

$$Sol'_{r+1} = \sum_{i=1}^{r}\left(\sum_{j=i+1}^{r+1}(p_j - x_j)\right)\alpha_i + \sum_{j=1}^{n} x_j\gamma_j + \sum_{i=r+2}^{n}\left(\sum_{j=r+2}^{i}(p_j - x_j)\right)\beta_i \,. \tag{7}$$

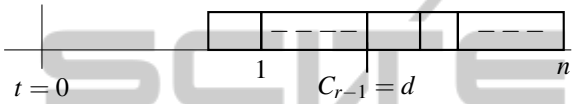

Figure 3: Schedule with the completion time of job $r-1$ lying at the due-date, $C_{r-1} = d$.

Let the solution value for CDD when the $(r-1)$th job ends at the due-date be Sol_{r-1} (Figure 3), then

$$Sol_{r-1} = \sum_{i=1}^{r-2}\left(\sum_{j=i+1}^{r-1} p_j\right)\alpha_i + \sum_{i=r}^{n}\left(\sum_{j=r}^{i} p_j\right)\beta_i \,. \tag{8}$$

As earlier, for a reduction of x_i in the processing times, the solution value Sol'_{r-1} will be

$$Sol'_{r-1} = \sum_{i=1}^{r-2}\left(\sum_{j=i+1}^{r-1}(p_j - x_j)\right)\alpha_i + \sum_{j=1}^{n} x_j\gamma_j + \sum_{i=r}^{n}\left(\sum_{j=r}^{i}(p_j - x_j)\right)\beta_i \,. \tag{9}$$

Since we know Sol_r is optimal we have

$$Sol_r \leq Sol_{r+1} \text{ and} \tag{10}$$

$$Sol_r \leq Sol_{r-1} \,. \tag{11}$$

Rearranging the terms in Sol_r, Sol_{r+1} and Sol_{r-1} from Equations (2), (6) and (8) respectively, we get

$$\begin{aligned} Sol_r &\leq Sol_{r+1} \,, \\ \sum_{i=r+1}^{n}\beta_i &\leq \sum_{i=1}^{r}\alpha_i \end{aligned} \tag{12}$$

and

$$\begin{aligned} Sol_r &\leq Sol_{r-1} \,, \\ \sum_{i=1}^{r-1}\alpha_i &\leq \sum_{i=r}^{n}\beta_i \,. \end{aligned} \tag{13}$$

Equation (12) also implies that

$$\sum_{i=k+1}^{n}\beta_i \leq \sum_{i=1}^{k}\alpha_i, k = r, r+1, \ldots, n-1, \tag{14}$$

that is, if the sum the tardiness penalties for the jobs $r+1$ to n is less than the sum of the earliness penalties for the jobs from 1 to r, then the inequality also holds for any $k \geq r$, since $\beta_i > 0$ and $\alpha_i > 0$ for $i = 1, 2, \ldots, n$.
Likewise, Equation (13) also implies that

$$\sum_{i=1}^{k-1}\alpha_i \leq \sum_{i=k}^{n}\beta_i, k = 1, 2, \ldots, r, \tag{15}$$

that is, if the sum the earliness penalties for the jobs 1 to $r-1$ is less than the sum of the tardiness penalties for the jobs from r to n, then the inequality also holds for any $k \leq r$, since $\beta_i > 0$ and $\alpha_i > 0$ for $i = 1, 2, \ldots, n$. This proves that the difference of the sum of the earliness penalties and the sum of the tardiness penalties changes sign before and after the optimal position of the due-date. The idea for Equation (14) and (15) has been adopted from (Cheng, 1989), where these properties were derived for a special case of the CDD problem.
We now use this property of the CDD prove an essential property for the un-restricted case of the CDD with controllable processing times.

Theorem 1. *If the due-date position in the optimal schedule of un-restricted case of the CDD lies at the completion time of some job r, then its position remains unchanged for the controllable case of the un-restricted CDD problem.*

Proof. Refer to the Appendix 1. □

3 THE EXACT ALGORITHM

In the previous section we proved that if the due-date position for the general common due-date problem lies at the completion time of a job then its position remains unchanged for the controllable processing time case as well. We now present how to utilize this property to formulate an exact algorithm to optimize a given job sequence for the un-restricted of the CDD with controllable processing times.

Algorithm 1: Linear Algorithm to optimize any given sequence of the un-restricted CDD with controllable processing times.

$C_i \leftarrow \sum_{k=1}^{i} p_k \; \forall \; i = 1,2,\ldots,n$
$\tau \leftarrow \underset{i=1,2,\ldots,n}{\arg\max}(C_i \leq d)$
if $(\tau \neq 0)$ **then**
  $pes \leftarrow \sum_{i=1}^{\tau} \alpha_i$
  $pls \leftarrow \sum_{i=\tau+1}^{n} \beta_i$
  if $(C_\tau \leq d) \wedge (pls < pes)$ **then**
    $C_i \leftarrow C_i - C_\tau + d \; \forall \; i$
  while $(\tau > 0) \wedge (pls < pes)$ **do**
    $pes \leftarrow pes - \beta_\tau$
    $pls \leftarrow pls + \alpha_\tau$
    $t \leftarrow 1$
    $\tau \leftarrow \tau - 1$
  if $(t = 1)$ **then**
    $C_i \leftarrow C_i - C_{\tau+1} + d \; \forall \; i$
$\sigma \leftarrow \underset{i=1,2,\ldots,n}{\arg\min}(C_i > d)$
$pls \leftarrow \sum_{i=\sigma}^{n} \beta_i$
$i \leftarrow \sigma$
$lShift \leftarrow 0$
while $(i \leq n)$ **do**
  if $(\gamma_i \leq pls) \wedge (p_i > m_i)$ **then**
    $dec \leftarrow p_i - m_i$
    $lShift \leftarrow lShift + dec$
  $pls \leftarrow pls - \beta_i$
  $C_i \leftarrow C_i - lShift$
  $i \leftarrow i + 1$
$\sigma \leftarrow \sigma - 1$
$ple \leftarrow \sum_{i=1}^{\sigma-1} \alpha_i$
$i \leftarrow \sigma$
$rShift \leftarrow 0$
while $(i > 1)$ **do**
  if $(\gamma_i \leq ple) \wedge (p_i > m_i)$ **then**
    $inc \leftarrow p_i - m_i$
    $rShift \leftarrow rShift + inc$
  $ple \leftarrow ple - \alpha_{i-1}$
  $C_{i-1} \leftarrow C_{i-1} + rShift$
  $i \leftarrow i - 1$
$PT_1 \leftarrow p_i$
$PT_i \leftarrow C_i - C_{i-1}, \; i = 2,3,\ldots,n$
Compute $E_i, T_i, \; i = 1,2,3,\ldots,n$
return $\sum_{i=1}^{n}(E_i \cdot \alpha_i + T_i \cdot \beta_i + \gamma_i(p_i - PT_i))$

To optimize a given sequence for the un-restricted case, we first find the optimal position of the due-date for the uncompressed then reduce the processing times of the jobs closest to the due-date moving outward. Consider Figure 1, the optimal position of the due-date is at C_r. In the next step, we first reduce the processing times of tardy jobs starting with job $r+1$. Reducing its processing time such that C_{r+1} moves closer to d will not only reduce the tardiness of job $r+1$ but of all the jobs which follow, provided the penalty incurred by compressing the processing time of job $r+1$ is less than the reduction in the weighted tardiness penalties of the jobs $r+1, r+2, \ldots, n$. Thereafter, we compress job $r+2$ and reduce its tardiness along with all the jobs following it. If a compression does not offer any reduction in the overall penalty then we move on to the next job without compressing the current job.

We perform the same operations in the sequential manner for the remaining jobs, starting with job r to job 2. However, in this case the reduction in the processing times leads the jobs to move towards right, *i.e.*, closer to the due-date. Notice that the reduction in the first job is never going to improve the penalty since the earliness of the first job will not be altered by its compression but will only offer more penalty due to compression. Algorithm 1 presents the pseudo code for optimizing a given sequence.

4 PROOF OF OPTIMALITY

We now provide the proof of the optimality of Algorithm 1 with respect to the solution value. Recall that we consider the un-restricted case of the CDD.

Theorem 2. *Algorithm 1 returns the optimal solution value to the un-restricted case of the CDD with controllable processing times for any given job sequence with a run-time complexity of $O(n)$.*

Proof. Since there is only one way that the due-date position may be between the completion times of two consecutive jobs, we need to first calculate the sum of penalties before and after the due-date such that the first job starts at time zero and all the jobs follow without any idle time. The schedule with $t^* = 0$ will be optimal if the sum of the tardiness penalties is already greater than the sum of earliness penalties. If that is not the case, we shift all the jobs towards right, as long as the sum of the tardiness penalties of jobs finishing after the due-date is less than or equal to the some of the earliness penalties of all the jobs which complete before the due-date, according to Equation (14) and (15).

Hence, we first optimize any given sequence for the general CDD problem and obtain the due-date position. We have already proved in Theorem 1 that the due-date position for the general CDD and the controllable processing times cases will be unaltered

Table 1: Results obtained for single machine common due-date problem with controllable processing times. For any given number of jobs, there are 10 different instances provided and each instance is designated a number k.

Jobs	**10**		**20**		**50**		**100**	
Approach	SA	TA	SA	TA	SA	TA	SA	TA
k = 1	763	763	2576	2642	14698	14605	60156	60124
k = 2	598	598	2555	2555	11864	11890	50534	50357
k = 3	672	672	3137	3111	13840	13774	57524	57483
k = 4	757	757	2761	2761	11925	11867	60729	60837
k = 5	473	473	1949	1936	12446	12373	46217	45999
k = 6	669	669	2767	2767	12252	12194	52097	52012
k = 7	913	913	3124	3124	14848	14848	53729	53742
k = 8	497	497	1492	1492	17598	17597	68199	68044
k = 9	510	510	1774	1760	11871	11864	48789	48747
k =10	601	601	1824	1824	11856	11841	51056	51003

for the un-restricted case of the CDD. It is clear that the optimal solution will occur only if the jobs are brought closer to the due-date since the due-date position should not change and the best case would be the one when all the jobs finish at the due-date, which is impossible. Hence, we reduce the processing times of jobs starting from the most adjacent one to the due-date moving further away. The processing time of a job is reduced only if the penalty incurred due to compression is less than the penalty reduced by the reduction in the earliness (tardiness) of the jobs before (after) it.

As for the run-time complexity, the first part of Algorithm 1 is to optimize a given sequence for the un-restricted CDD problem to find the optimal position of the due-date. It can be easily observed that the complexity for this part is of linear run-time. The next expensive operations in terms of the run-time occur at the next two *while* loops and they are both of $O(n)$ in the worst case. The remaining steps are all linear. Hence the overall complexity of Algorithm 1 is $O(n)$. □

5 RESULTS

Due to the unavailability of benchmark instances for this problem as per our knowledge, we first present our methodology to append the benchmark instances of the general CDD provided in the OR-library with the extra parameters for the controllable processing time case (Beasley, 1990). The instances provided in (Beasley, 1990) provide the processing times, earliness/tardiness penalties and the due-date. Hence, we append the information about the minimum processing times and the cost of controlling the processing times per unit time. We take the minimum processing time of any job as $m_i \sim DU(0.6p_i, p_i)$ and $\gamma_i \sim DU(1,5)$, where $\sim DU(a,b)$ is a discrete uniform random number between a and b. The rest of the parameters remain the same as in (Beasley, 1990). Our updated benchmark instances can be found at www.enterprise-application-development.org/research/benchmark-data.html.

5.1 Modified Simulated Annealing

We use a modified Simulated Annealing algorithm to generate job sequences and Algorithm 1 to optimize each sequence to its minimum penalty. Our experiments over all the instances suggest that an ensemble size (M) of $\approx n/10$ and the maximum number of iterations of $500n$, where n is the number of jobs, work best for the provided instances in general. The runtime for all the results is the time after which the solutions mentioned in Table 1 are obtained on average after 10 different replications. The initial temperature (T_0) is kept as twice the standard deviation of the energy at infinite temperature: $\sigma_{E_{T=\infty}} = \sqrt{\langle E^2\rangle_{T=\infty} - \langle E\rangle^2_{T=\infty}}$. We estimate this quantity by randomly sampling the configuration space (Salamon et al., 2002). An exponential schedule for cooling is adopted with a cooling rate of $1-10^{-4}$. One of the modifications from the standard SA is in the acceptance criterion. We implement two acceptance criteria: the Metropolis acceptance probability, $\min\{1, \exp((-\triangle E)/T)\}$ and a constant acceptance probability of $c \cdot 10^{-2}$, c being a constant. A solution is accepted with this constant probability if it is rejected by the Metropolis criterion. This concept of a constant probability is useful when the SA is run for many iterations and the metropolis acceptance probability is almost zero, since the temperature would become infinitesimally small.

Another modification is the increase in the temperature after the annealing temperature becomes less than one unit. In such a case we increase the temperature to $1/10$th of the initial temperature. Apart from this, we also incorporate elitism in our modified SA. Elitism has been successfully adopted in evolutionary

Table 2: Table 1 continued...

Jobs	200		500		1000	
Approach	SA	TA	SA	TA	SA	TA
k = 1	205307	205088	1324829	1325310	5354091	5460812
k = 2	224213	224136	1425334	1426581	5138818	5259803
k = 3	215669	215528	1371225	1370031	5051903	5165823
k = 4	243170	242875	1372239	1371316	5128463	5256762
k = 5	214962	214936	1211943	1212666	5278280	5393347
k = 6	199305	199157	1189069	1189804	5073112	5185676
k = 7	210827	210815	1357730	1357399	5514919	5633111
k = 8	190068	189791	1287052	1286917	5099539	5204219
k = 9	215707	215556	1396230	1397012	5215425	5375118
k = 10	228356	228071	1264289	1264434	5187998	5292440

algorithms for solving complex optimization problems (Gen et al., 1994; Kim, 2013). Lässig and Sudholt made theoretical studies analysing speed-ups in parallel evolutionary algorithms with elitism applied to combinatorial optimization problems (Lässig and Sudholt, 2011; Lässig and Sudholt, 2013). In (Lässig and Hoffmann, 2009) it is shown that for a large class of quality measures, the best possible probability distribution is a rectangular distribution over certain individuals sorted by their objective values, which can be seen as a mild form of elitism. As for the perturbation rule, we first randomly select a certain number of jobs in any job sequence and permute them randomly to create a new sequence. The number of jobs selected for this permutation is taken as $c + \lfloor \sqrt{n/10} \rfloor$, where n is the number of jobs and c is a constant. This modified SA works well for the CDD and the Aircraft Landing Problem as well (Awasthi et al., 2013b; Awasthi et al., 2013a).

5.2 Threshold Accepting

Threshold Accepting is another heuristic algorithm based on Simulated Annealing, proposed by (Dueck and Scheuer, 1990). The basic difference from the SA is the different acceptance rules. Unlike the standard SA where a worse solution is accepted as per the metropolis acceptance criterion, TA instead accepts *'every new configuration which is not much worse than the old one'* (Dueck and Scheuer, 1990). We now present the exact details of the acceptance criterion adopted by us in this work.

The initial temperature (T_0) is kept the same as in Simulated Annealing. As opposed to the exponential cooling schedule of SA, we adopt probabilistic arithmetic cooling scheduling in TA. Let, mE_j and mE_{j-1} be the mean of the energy (in this optimization problem, energy is the objective function values) of all the states in the current (j) and the previous iteration ($j-1$), respectively. Then, the temperature T_j is reduced as

Table 3: Average run-time in seconds for all the 10 different instances for each job over 10 iterations of the algorithm.

Jobs	SA	TA
10	0.173	0.490
20	1.147	0.931
50	6.017	7.768
100	13.510	21.637
200	50.973	88.822
500	171.806	221.410
1000	361.553	603.711

$$T_j = \begin{cases} T_{j-1} - \delta, & \text{if } mE_j - mE_{j-1} \leq prob\ , \\ T_{j-1}, & \text{otherwise}\ . \end{cases} \quad (16)$$

In Equation (16), δ and $prob$ are defined as $\delta = \tau \cdot T_0$ and $prob = v \cdot T_0/\sqrt{M}$, where $\tau = c \cdot 10^{-1}$ and $v = c \cdot 10^{-4}$, c being a constant and M is the ensemble size. The acceptance criterion for Threshold Accepting as proposed by (Dueck and Scheuer, 1990) is the current temperature T_j. The remaining parameters such as the ensemble size, perturbation size and the number of iterations are kept the same for both SA and TA, so as to exactly compare the two approaches.

In Table 1 and 2, we present our results for the CDD with controllable processing times for the unrestricted case, for the benchmark instances. For the first 10 instances with 10 jobs, SA and TA both reach the optimal solution as it turns out by the comparison results with integer programming. For the small instances with 20, 50 and 100 jobs, we observed that the Threshold Acceptance offers predominantly better results in terms of the objective values as shown in Table 1. TA achieves better results for 17 and equal results for 8 instances out of 30 instances. However, for the large instances with 200, 500 and 1000 jobs, the difference in the two heuristic approach was marginal with 16 better results for SA and 14 for TA, as shown in Table 2. Hence, as far as the quality of the solution is concerned, Threshold Accepting and Simulated Annealing offer almost the same quality of results. This was not the case for the run-times of the two approaches. In Table 3 we can see the time

required by both the approaches. The run-times of TA and SA are almost the same for small instances till 50 jobs. However, for large instances with 100 and more jobs, Simulated Annealing performs much faster then the Threshold Accepting. Considering, the same quality of solution values for both, in our experience the modified Simulated Annealing algorithm proposed in this work performs better than Threshold Accepting for the optimization problem dealt with.

6 CONCLUSION AND FUTURE DIRECTION

In this paper we present a novel property for the problem of scheduling against a common due-date with controllable processing times for the un-restricted case. We show that the due-date position in the optimal schedule for the un-restricted case remains the same for both the CDD and for controllable processing time cases. We then present an $O(n)$ algorithm for a given sequence and prove the run-time complexity and its optimality with respect to the solution value. We implement our algorithm over the benchmark instances provided by (Biskup and Feldmann, 2001) and appended by us for all the instances till 10^3 jobs. Besides, we encourage other researchers interested in this problem to test other approaches with our benchmark instances.

ACKNOWLEDGEMENTS

The research project was promoted and funded by the European Union and the Free State of Saxony, Germany.

REFERENCES

Awasthi, A., Kramer, O., and Lässig, J. (2013a). Aircraft landing problem: An efficient algorithm for a given landing sequence. In *16th IEEE International Conferences on Computational Science and Engineering (CSE 2013)*, pages 20–27.

Awasthi, A., Lässig, J., and Kramer, O. (2013b). Common due-date problem: Exact polynomial algorithms for a given job sequence. In *15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2013*, pages 258–264.

Beasley, J. (1990). OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072.

Biskup, D. and Cheng, T. (1999). Single-machine scheduling with controllable processing times and earliness, tardiness and completion time penalties. *Engineering Optimization*, 31(3):329–336.

Biskup, D. and Feldmann, M. (2001). Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates. *Computers & Operations Research*, 28(8):787 – 801.

Cheng, T. (1989). Optimal due-date assignment and sequencing in a single machine shop. *Applied Mathematics Letters*, 2(1):21–24.

Dueck, G. and Scheuer, T. (1990). Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, 90(1):161 – 175.

Gen, M., Tsujimura, Y., and Kubota, E. (1994). Solving job-shop scheduling problems by genetic algorithm. In *IEEE International Conference on Systems, Man, and Cybernetics, 1994. Humans, Information and Technology.*, volume 2, pages 1577–1582.

Kim, J. (2013). Genetic algorithm stopping criteria for optimization of construction resource scheduling problems. *Construction Management and Economics*, 31(1):3–19.

Lässig, J. and Hoffmann, K. (2009). Threshold-selecting strategy for best possible ground state detection with genetic algorithms. *Phys. Rev. E*, 79:046702.

Lässig, J. and Sudholt, D. (2011). Analysis of speedups in parallel evolutionary algorithms for combinatorial optimization. In *Proceedings of the 22nd International Conference on Algorithms and Computation*, ISAAC'11, pages 405–414. Springer-Verlag.

Lässig, J. and Sudholt, D. (2013). General upper bounds on the runtime of parallel evolutionary algorithms. pages 1–33.

Nearchou, A. (2010). Scheduling with controllable processing times and compression costs using population-based heuristics. *International Journal of Production Research*, 48(23):7043–7062.

Salamon, P., Sibani, P., and Frost, R. (2002). *Facts, Conjectures, and Improvements for Simulated Annealing*. Society for Industrial and Applied Mathematics.

Shabtay, D. and Steiner, G. (2007). A survey of scheduling with controllable processing times. *Discrete Applied Mathematics*, 155(13):1643 – 1666.

Tseng, C., Liao, C., and Huang, K. (2009). Minimizing total tardiness on a single machine with controllable processing times. *Computers & Operations Research*, 36(6):1852 – 1858.

Wan, G. (2007). Single machine common due window scheduling with controllable job processing times. In *Combinatorial Optimization and Applications*, volume 4616 of *Lecture Notes in Computer Science*, pages 279–290. Springer Berlin Heidelberg.

Yunqiang, Y., Cheng, T., Cheng, S., and Wu, C. (2013). Single-machine batch delivery scheduling with an assignable common due date and controllable processing times. *Computers & Industrial Engineering*, 65(4):652 – 662.

APPENDIX 1

Proof of Theorem 1

Proof. Since there is only one way that the due-date position may be between the completion times of two consecutive jobs (*i.e.*, when the first job starts at time $t = 0$), we need to first calculate the sum of penalties before and after the due-date such that the first job starts at time zero and all the jobs follow without any idle time. Thereafter, we shift all the jobs towards right as long as the sum of the tardiness of jobs finishing after the due-date is less than or equal to the some of the earliness penalties of all the jobs which complete before the due-date. Rearranging the terms of Sol'_r from Equation (3), we have

$$\begin{aligned} Sol'_r &= \sum_{i=1}^{r-1}\sum_{j=i+1}^{r}(p_j - x_j)\alpha_i + \sum_{j=1}^{n} x_j\gamma_j \\ &\quad + \sum_{i=r+1}^{n}\sum_{j=r+1}^{i}(p_j - x_j)\beta_i \,, \\ Sol'_r &= \sum_{i=1}^{r-1}\sum_{j=i+1}^{r} p_j\alpha_i - \sum_{i=1}^{r-1}\sum_{j=i+1}^{r} x_j\alpha_i - \sum_{i=r+2}^{n}\sum_{j=r+2}^{i} x_j\beta_i \\ &\quad + \sum_{j=1}^{n} x_j\gamma_j + \sum_{i=r+1}^{n}(p_{r+1} - x_{r+1})\beta_i \\ &\quad + \sum_{i=r+2}^{n}\sum_{j=r+2}^{i} p_j\beta_i \,. \end{aligned} \tag{17}$$

With some manipulations of terms, Sol'_r can be also written as

$$\begin{aligned} Sol'_r &= \sum_{i=1}^{r-2}\sum_{j=i+1}^{r-1} p_j\alpha_i - \sum_{i=1}^{r-2}\sum_{j=i+1}^{r-1} x_j\alpha_i + \sum_{i=1}^{r-1}(p_r - x_r)\alpha_i \\ &\quad + \sum_{j=1}^{n} x_j\gamma_j + \sum_{i=r+1}^{n}\sum_{j=r+1}^{i} p_j\beta_i - \sum_{i=r+1}^{n}\sum_{j=r+1}^{i} x_j\beta_i \,. \end{aligned} \tag{18}$$

Similarly, the terms of Sol'_{r+1} in Equation (7) can rearranged such that

$$\begin{aligned} Sol'_{r+1} &= \sum_{i=1}^{r}\sum_{j=i+1}^{r+1}(p_j - x_j)\alpha_i + \sum_{i=r+2}^{n}\sum_{j=r+2}^{i}(p_j - x_j)\beta_i \\ &\quad + \sum_{j=1}^{n} x_j\gamma_j \,, \\ &= \sum_{i=1}^{r-1}\sum_{j=i+1}^{r} p_j\alpha_i - \sum_{i=1}^{r-1}\sum_{j=i+1}^{r} x_j\alpha_i + \sum_{i=r+2}^{n}\sum_{j=r+2}^{i} p_j\beta_i \\ &\quad + \sum_{i=1}^{r}(p_{r+1} - x_{r+1})\alpha_i - \sum_{i=r+2}^{n}\sum_{j=r+2}^{i} x_j\beta_i + \sum_{j=1}^{n} x_j\gamma_j \,. \end{aligned} \tag{19}$$

Likewise Sol'_{r-1} in Equation (9) can also be expressed as

$$\begin{aligned} Sol'_{r-1} &= \sum_{i=1}^{r-2}\sum_{j=i+1}^{r-1}(p_j - x_j)\alpha_i + \sum_{i=r}^{n}\sum_{j=r}^{i}(p_j - x_j)\beta_i \\ &\quad + \sum_{j=1}^{n} x_j\gamma_j \,, \\ &= \sum_{i=1}^{r-2}\sum_{j=i+1}^{r-1} p_j\alpha_i - \sum_{i=1}^{r-2}\sum_{j=i+1}^{r-1} x_j\alpha_i + \sum_{i=r+1}^{n}\sum_{j=r+1}^{i} p_j\beta_i \\ &\quad + \sum_{i=r}^{n}(p_r - x_r)\beta_i - \sum_{i=r+1}^{n}\sum_{j=r+1}^{i} x_j\beta_i + \sum_{j=1}^{n} x_j\gamma_j \,. \end{aligned} \tag{20}$$

Now we prove by contradiction that the position of the due-date will not change even after the optimal reduction in the processing times. Let us assume that Sol'_r is not optimal, then we have

$$Sol'_r > Sol'_{r+1} \text{ and} \tag{21}$$

$$Sol'_r > Sol'_{r-1} \,. \tag{22}$$

Substituting values of Sol'_r from Equation (17) and Sol'_{r+1} from Equation (19) in Equation (21), we have

$$\begin{aligned} Sol'_r &> Sol'_{r+1} \,, \\ \sum_{i=r+1}^{n}(p_{r+1} - x_{r+1})\beta_i &> \sum_{i=1}^{r}(p_{r+1} - x_{r+1})\alpha_i \text{ and} \end{aligned}$$

$$(p_{r+1} - x_{r+1})\left(\sum_{i=r+1}^{n}\beta_i - \sum_{i=1}^{r}\alpha_i\right) > 0 \,. \tag{23}$$

Now, using Equation (12), we obtain

$$p_{r+1} < x_{r+1} \,. \tag{24}$$

Likewise, substituting the values of Sol'_r from Equation (18) and Sol'_{r-1} from Equation (20) in Equation (22), we get

$$(p_r - x_r)\left(\sum_{i=1}^{r-1}\alpha_i - \sum_{i=r}^{n}\beta_i\right) > 0 \,. \tag{25}$$

Equation (13) then fetches

$$p_r < x_r \,. \tag{26}$$

Equation (24) and (26) show that if the optimal solution for the uncompressed case occurs such when the due-date position is at C_r for some r, then for the

compressed case, the position of the due-date will remain fixed at C_r as well, since a change in the position of the due-date will require a compression in the processing time which is more than the actual processing time itself. □