Common Due-Window Problem: Polynomial Algorithms for a Given Processing Sequence

Abhishek Awasthi
Department of Computer Science,
University of Applied Sciences,
Görlitz, Germany
Email: abhishek.awasthi@hszg.de

Jörg Lässig
Department of Computer Science,
University of Applied Sciences,
Görlitz, Germany
Email: joerg.laessig@hszg.de

Oliver Kramer
Department of Computing Science
University of Oldenburg,
Oldenburg, Germany
Email: oliver.kramer@uni-oldenburg.de

Thomas Weise
School of Computer Science and Technology,
University of Science and Technology of China,
Hefei, Anhui, China
Email: tweise@ustc.edu.cn

Abstract—The paper considers the Common Due-Window (CDW) problem where a single machine processes a certain number of jobs against a common due-window. Each job possesses different processing times but different and asymmetric earliness and tardiness penalties. The objective of the problem is to find the processing sequence of jobs, their completion times and the position of the given due-window to minimize the total penalty incurred due to tardiness and earliness of the jobs. This work presents exact polynomial algorithms for optimizing a given job sequence for a single machine with the run-time complexity of $O(n^2)$, where $n$ is the number of jobs. We also provide an $O(n)$ algorithm for optimizing the CDW with unit processing times. The algorithms take a sequence consisting of all the jobs $(J_i, i = 1, 2, \ldots, n)$ as input and return the optimal completion times, which offers the minimum possible total penalty for the sequence. Furthermore, we implement our polynomial algorithms in conjunction with Simulated Annealing (SA) to obtain the best processing sequence. We compare our results with that of Biskup and Feldmann [2] for different due-window lengths. This is a preview version of the paper [1] (see page 10 for the reference). Read the full piece in the proceedings.

I. INTRODUCTION

The Common Due-Window (CDW) scheduling problem involves sequencing and scheduling of jobs over machine(s) against a given common due-window. The objective is to find the position of the due-window of a given length and the job sequence to minimize the total tardiness and earliness penalties. Each job possesses a processing time and different penalties per unit time in case the job is completed before or later than the due-window. The jobs which are completed between or at the due-window are called straddle jobs and do not incur any penalty. Similar to the Common Due-Date (CDD) problem, the CDW also occurs in the supply chain management industry to reduce the earliness and tardiness of the goods produced.

Common due-date problems have been studied extensively during the last 30 years with several variants and special cases [3], [4], [5], [6], [7], [8]. CDW is an extension of the CDD with the presence of a common due-window instead of a common due-date. However, several important similar properties hold for both the problems. In 1994, Krämer and Lee studied the due-window scheduling for the parallel machine case and presented three properties for the CDW [9].

Property 1. There exists an optimal schedule without machine idle time between the first and the last job.

Proof: Refer to [9], [10].

Property 2. In any optimal schedule, jobs completed before $d_l$ (the left due-date) are sequenced in the reverse SPT order, and the jobs which start after $d_r$ (the right due-date) are sequenced in the SPT order.

Proof: Refer to [9], [10].

Property 3. Let $C_i$ be the completion time of job $i$, then there exists an optimal schedule where one of the jobs finishes at $d_l$ or at $d_r$, i.e.,

a) $C_i = d_l$ for some $i$, or
b) $C_i = d_r$ for some $i$.

Proof: Refer to [9], [10].

Krämer and Lee also showed that the CDW with unit weight case is also NP-complete and provided a dynamic programming algorithm for the two machine case [9]. Liman et al. considered the CDW with constant earliness/tardiness penalties and proposed an $O(n \log n)$ algorithm to minimize the weighted sum of earliness, tardiness and due-window location [11]. The same authors also studied the CDW on a single machine with controllable processing times with constant penalties for earliness, tardiness and window location, and different penalties for compression of job processing.
times. They showed that the problem can be formulated as an assignment problem and can be solved using the well-known algorithms [12].

In 2002, Chen and Lee studied the CDW on parallel machines and solved the problem using a Branch and Bound algorithm and showed that the problem can be solved up to 40 jobs on any number of machines [13] in a reasonable time. In 2005, Biskup and Feldmann dealt with the general case of the CDW problem and approached it with three different metaheuristic algorithms, namely, evolutionary strategy, simulated annealing and threshold accepting. They also validated their approaches on 250 benchmark instances up to 200 jobs [2]. Wan studied the common due-window problem with controllable processing times with constant earliness/tardiness penalties and distinct compression costs, and discussed some properties of the optimal solution along with a polynomial algorithm for the solving the problem in 2007 [14]. Zhao et al. studied the CDW with constant earliness/tardiness penalties and window location penalty, and proposed polynomial time approximation schemes [15].

In 2010, Yeung et al. formulated a supply chain scheduling control problem involving single supplier and manufacturer and multiple retailers. They formulated the problem as a two machine CDW and presented a pseudo-polynomial algorithm to solve the problem optimally [16]. Cheng et al. considered the common due-window assignment problem with time-dependent deteriorating jobs and a deteriorating maintenance activity. They proposed a polynomial algorithm for the problem with linear deterioration penalties and its special cases [17]. Gerstl and Mosheiov studied the due-window problem with linear deterioration penalties and its special cases [17].

In this paper, we consider the single machine case for the CDW problem with asymmetric penalties for both the general and the unit-time job cases. We make a theoretical study of the CDW problem and present an $O(n^2)$ and $O(n)$ polynomial exact algorithms to optimize a given job sequence on a single machine for the general and unit-time job cases, respectively.

\section{Problem Formulation}

In this section, we give the mathematical notation of the common due-window problem based on [2]. We also define some new parameters which are later used in the presented algorithms in the next section.

Let
\begin{align*}
n & = \text{the total number of jobs}, \\
d_l & = \text{the left common due-date}, \\
d_r & = \text{the right common due-date}, \\
p_i & = \text{the processing time of job } i, \quad i = 1, 2, \ldots, n, \\
E_i & = \text{the earliness of job } i, \\
T_i & = \text{the tardiness of job } i, \\
C_i & = \text{the completion time of job } i, \\
W_i & = \text{the straddle jobs, i.e. if } d_l \leq C_i \leq d_r, \forall i, \\
\alpha_i & = \text{the earliness penalty per unit time for job } i, \\
\beta_i & = \text{the tardiness penalty per unit time for job } i.
\end{align*}

\begin{align*}
E_i & = \max \{0, d_l - C_i\} \\
T_i & = \max \{0, C_i - d_r\} \\
& \quad i = 1, 2, \ldots, n.
\end{align*}

The objective of the problem is to schedule the jobs against the due-window to minimize the total penalty incurred by the earliness and tardiness of the jobs.

\begin{equation}
\min \sum_{i=1}^{n} \{\alpha_i \cdot E_i + \beta_i \cdot T_i\}.
\end{equation}

Before stating the algorithm we first introduce a new vector $DT_i = C_i - d_l$ and $SD_i = C_i - d_r$, $i = 1, 2, \ldots, n$. $DT_i$ and $SD_i$ are just the algebraic deviation of the completion time of any job $i$ from the left and the right due-date, respectively.

\begin{definition}
Let PL be a vector of length $n$ where element of PL ($PL_i$) is the effective penalty possessed by any job $i$ such that
\begin{equation}
PL_i = \begin{cases} 
-\alpha_i, & \text{if } DT_i \leq 0 \\
\beta_i, & \text{if } SD_i > 0 \\
0, & \text{otherwise .}
\end{cases}
\end{equation}

We also define two new vectors, $D$ and $S$, to express the objective function mentioned in Equation (1) in a compact form. $D_i$ and $S_i$ are defined for all $i$, $1, 2, \ldots, n$ such that
\begin{align}
D_i & = \begin{cases} 
DT_i, & \text{if } DT_i \leq 0 \\
0, & \text{if } DT_i > 0,
\end{cases} \\
S_i & = \begin{cases} 
0, & \text{if } SD_i < 0 \\
SD_i, & \text{if } SD_i \geq 0.
\end{cases}
\end{align}

With the above definitions we can now express the objective function stated by Equation (1) as $\min(Sol)$, where
\begin{equation}
Sol = \sum_{i=1}^{n} \{(D_i + S_i) \cdot PL_i\}.
\end{equation}

We now present and prove an important property for the Common Due Date problem. Later on, we extend this property for the Common Due Window problem.

\section{Novel Property for the Common Due-Window Date}

\begin{theorem}
If the optimal due-date position in any given job sequence of the CDD lies between $C_{r-1}$ and $C_r$, i.e., $C_{r-1} < d \leq C_r$, then the following relations hold for the two cases
\begin{enumerate}
\item[Case 1: If $C_{r-1} < d < C_r$]
\item[i) $\sum_{i=k+1}^{n} \beta_i \leq \sum_{i=1}^{k} \alpha_i, \quad k = r, r+1, \ldots, n$.
\end{enumerate}
\end{theorem}
Case 2: If \( C_r = d \)

i) \[
\sum_{i=k+1}^{n} \beta_i \leq \sum_{i=1}^{k} \alpha_i, \quad k = r, r+1, \ldots, n \text{ and}
\]

ii) \[
\sum_{i=1}^{k} \alpha_i \leq \sum_{i=k+1}^{n} \beta_i, \quad k = 1, 2, 3, \ldots, r - 1.
\]

Proof: We know from the property proved by [22] that the optimal schedule of the CDD for any job sequence either has \( t^* = 0 \) or one of the job finishes at the due-date. Hence, we consider these two cases separately.

Case 1: Optimal schedule with \( C_{r-1} < d < C_r \)

Let us first consider the case when the optimal schedule for any sequence lies strictly between \( C_{r-1} \) and \( C_r \), i.e. \( C_{r-1} < d < C_r \), as shown in Figure 1. We know from [22] that such a case can occur only when the first job starts at time \( t = 0 \) and all the following jobs are processed without any machine idle time. Let the difference between \( C_{r-1} \) and \( d \) in Figure 1 be \( y \) such that \( y = d - C_{r-1} \). Let \( E_i \) and \( T_i \) be the earliness and tardiness penalties of any job \( i \), for this particular case, respectively. Hence, the solution value \( \text{Sol}_d \) for the schedule in Figure 1 can be written as

\[
\text{Sol}_d = \sum_{i=1}^{r-1} E_i \cdot \alpha_i + \sum_{i=r}^{n} T_i \cdot \beta_i.
\]  

(6)

![Fig. 1. Assume that the first job starts at time \( t = 0 \) and the due-date lies between the completion times of two consecutive jobs, with \( y = d - C_{r-1} \).](image)

Now, the only possibility to get another schedule is to shift all the jobs to the right such that one of the job finishes at the due-date. Figure 2 shows the right shift of all the jobs by \( y \) units. It is clear that after this right shift of all the jobs, job \( r-1 \) offers no penalty. Hence, the earliness of the early jobs in Figure 2 will be \( E_i - y \) for \( i = 1, 2, \ldots, r - 2 \) and the tardiness of the tardy jobs will be \( T_i + y \) for \( i = r, r+1, \ldots, n \). We can now write the solution value for Figure 2 as \( \text{Sol}'_d \) where

\[
\text{Sol}'_d = \sum_{i=1}^{r-2} (E_i - y) \cdot \alpha_i + \sum_{i=r}^{n} (T_i + y) \cdot \beta_i.
\]  

(7)

![Fig. 2. Assume that the \((r-1)\)th job finishes at the due-date \( d \) in the optimal schedule.](image)

Since we already assumed that Figure 1 is the optimal schedule, we have

\[
\text{Sol}_d \leq \text{Sol}'_d.
\]  

(8)

Note that in Figure 1 the earliness of job \( r \) is \( y \). Hence \( \text{Sol}_d \) can be rewritten as

\[
\text{Sol}_d = \sum_{i=1}^{r-2} E_i \cdot \alpha_i + y \cdot \alpha_{r-1} + \sum_{i=r}^{n} T_i \cdot \beta_i.
\]  

(9)

Likewise, the terms in \( \text{Sol}'_d \) can also be expanded as

\[
\text{Sol}'_d = \sum_{i=1}^{r-2} (E_i - y) \cdot \alpha_i + \sum_{i=r}^{n} (T_i + y) \cdot \beta_i.
\]  

(10)

Substituting the value of \( \text{Sol}_d \) from Equation (9) and \( \text{Sol}'_d \) from Equation (10) in Equation (8), we get

\[
\sum_{i=1}^{r-2} (E_i - y) \cdot \alpha_i + \sum_{i=r}^{n} (T_i + y) \cdot \beta_i \leq \sum_{i=1}^{r-2} E_i \cdot \alpha_i + \sum_{i=r}^{n} T_i \cdot \beta_i + y \cdot \alpha_{r-1} + \sum_{i=r}^{n} T_i \cdot \beta_i - y \cdot \alpha_{r-1}.
\]

(11)

Since \( y > 0 \) due to case constraint, Equation (11) fetches us

\[
\sum_{i=1}^{r-2} \beta_i \leq \sum_{i=r}^{n} \beta_i.
\]  

(12)

Clearly, if Equation 12 holds for any \( k = r \), then it will also hold for any \( k < r \), since \( \alpha_k \) and \( \beta_k \) are positive for all \( i \), \( i = 1, 2, \ldots, n \). This proves the first case of Theorem 1.

Case 2: Optimal schedule at \( C_r = d \)

In this case we assume that the optimal solution lies at the completion time of some job \( r \). Consider Figure 3, where the optimal schedule occurs with the due-date position at the completion time of job \( r \), i.e. \( C_r = d \).

![Fig. 3. Assume that the \( r \)th job finishes at the due-date \( d \) in the optimal schedule.](image)

Let, \( E_i \) and \( T_i \) be the earliness and tardiness of any job \( i \), respectively, for this particular case (Figure 3) and the solution value for this case be \( \text{Sol}_r \), then using Equation (5) we have

\[
\text{Sol}_r = \sum_{i=1}^{r-1} E_i \cdot \alpha_i + \sum_{i=r+1}^{n} T_i \cdot \beta_i.
\]  

(13)

![Fig. 4. Schedule with the completion time of job \( r + 1 \) lying at the due-date, \( C_{r+1} = d \).](image)

Let the solution value for the case when all the jobs are shifted to the left by \( p_{r+1} \), i.e., the \((r+1)\)th job ends at the due-date be \( \text{Sol}_{r+1} \), see Figure 4. Then the earliness of jobs 1 to \( r - 1 \) will increase by the processing time of job \( r + 1 \), compared to Figure 3, since the due-date position shifts to right by the same amount and job \( r \) will be early by \( p_{r+1} \). Besides, job
\[ r + 1 \text{ offers no penalty and the tardiness of all the jobs from } r + 2 \text{ to } n \text{ reduces by } p_{r+1}. \] Hence, the objective function value when the due-date is situated at \( C_{r+1} \) becomes
\[
Sol_{r+1} = \sum_{i=1}^{r-1} (E_i + p_{r+1}) \alpha_i + p_{r+1} \alpha_r + \sum_{i=r+2}^{n} (T_i - p_{r+1}) \beta_i . \tag{14}
\]

Since we assume that \( Sol_r \) is the optimal value, we have,
\[
Sol_r \leq Sol_{r+1} , \tag{16}
\]
and
\[
Sol_r \leq Sol_{r-1} . \tag{17}
\]

Notice that in the first case, when \( C_r = d \), the tardiness of job \( r = 1 \) is \( p_{r+1} \) and the earliness of job \( r = 1 \) is \( p_r \). Hence, rearranging the terms in \( Sol_r \) we get,
\[
Sol_r = \sum_{i=1}^{r-1} E_i \cdot \alpha_i + p_{r+1} \beta_r + \sum_{i=r+2}^{n} T_i \beta_i . \tag{18}
\]

Splitting the earliness penalty of job \( r - 1 \), \( Sol_r \) can also be expressed as,
\[
Sol_r = \sum_{i=1}^{r-2} E_i \cdot \alpha_i + p_{r+1} \cdot \beta_r + \sum_{i=r+2}^{n} T_i \beta_i . \tag{19}
\]

Substituting the values of \( Sol_r \) from Equation (18) and \( Sol_{r+1} \) from Equation (14) in Equation (16) we get,
\[
\sum_{i=r+1}^{n} p_{r+1} \cdot \beta_i \leq \sum_{i=1}^{r} p_{r+1} \cdot \alpha_i \text{ and } \tag{20}
\sum_{i=r+1}^{n} \beta_i \leq \sum_{i=1}^{r} \alpha_i .
\]

Likewise, substituting the values of \( Sol_r \) from Equation (19) and \( Sol_{r-1} \) from Equation (15) in Equation (17),
\[
Sol_l \leq Sol_{r-1} , \tag{21}
\]

Since \( \alpha_i \) and \( \beta_i \) are positive for all \( i \), Equation (20) also implies,
\[
\sum_{i=k+1}^{n} \beta_i \leq \sum_{i=1}^{k} \alpha_i , \tag{22}
\]
i.e., if the sum of 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 same inequality also holds for any \( k \geq r \), since \( \beta_i > 0 \) and \( \alpha_i > 0 \) for \( i = 1, 2, \ldots, n \). Likewise, Equation (21) implies that
\[
\sum_{i=1}^{k} \alpha_i \leq \sum_{i=k+1}^{n} \beta_i , \tag{23}
\]
i.e., if the sum of 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 same inequality also holds for any \( k \leq r - 1 \), since \( \beta_i > 0 \) and \( \alpha_i > 0 \) for \( i = 1, 2, \ldots, n \). This proves the second case of Theorem 1.

Since there is only one way that the due-date position may be between the completion times of two consecutive jobs, we first need to 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 sum of the earliness penalties of all the jobs which complete before the due-date.

---

**Theorem 2.** If in the optimal schedule of a CDW instance, jobs \( 1, 2, \ldots, r - 1 \) are early and jobs \( k, k + 1, \ldots, n \), \( k > r \) are tardy then, we have, \( \sum_{i=k}^{n} \beta_i \leq \sum_{i=1}^{r-1} \alpha_i \) for minimum possible value of \( k \).

**Proof:** Using Property 3, let us assume without loss of any generality that in the optimal schedule the left due-date \( d_l \) lies at the completion time of a job and the right due-date \( d_r \) lies between the completion times of two adjacent jobs.
Figure 6 depicts the optimal position of the due-window for the given instance of \( n \) jobs. Jobs 1, 2, \ldots, \( r - 1 \) are early and jobs \( k, k + 1, \ldots, n \) are tardy, then we can discard the straddle jobs (which are within the due-window) and convert the problem to a CDD problem as shown in Figure 7. The problem now converts to the CDD with \( d_i \) being the due-date. Hence, the properties of Theorem 1 will also hold for the CDW on the same lines as for the CDD.

IV. THE EXACT ALGORITHM

Using Theorem 2, we now present our exact polynomial algorithms for the CDW with distinct and unit-time jobs cases. As mentioned above in Properties 1, 2 and 3, we know that the optimal schedule of the CDW has no idle time of the machine between \( C_1 \) and \( C_n \). The idea of our algorithm is based on the approaches mentioned in [23], [24]. We first optimize any given sequence using our polynomial algorithm and use a modified Simulated Annealing (SA) algorithm to find the optimal/best processing sequence.

The jobs are initialized with the first job starting at time \( t = 0 \) and are shifted to the right by \( \min \{-DT_{n1}, -SD_{n2}\} \), i.e., minimum deviation of the completion times from the right and the left due-dates. This way, every shift ensure that one of the jobs finishes at one of the due-dates and we do not skip over the optimal position of the due-dates. Once the property mentioned in Theorem 2 is satisfied, we have our optimal schedule and no more shifting is required. We now present Algorithm 1 and 2 to optimize the CDW for the distinct and unit-processing times of the jobs, for a given processing sequence.

V. PROOF OF OPTIMALITY

We now provide the optimality of Algorithm 1 and 2 with respect to the solution value.

**Theorem 3.** Algorithm 1 and 2 are optimal for the given sequence with respect to the objective function value.

**Proof:** We first schedule a given job sequence such that the processing of the jobs starts at time \( t = 0 \) and move the jobs towards the right, i.e., increasing the overall tardiness penalty and decreasing the overall earliness penalty, to find the minimum value of \( k \) as mentioned in Theorem 2. We increase the completion times by \( \min \{ DT_i, SD_j \} \) \( DT \) and \( SD \) are defined in Section II), where job \( i \) possesses the least earliness from \( d_i \) and job \( j \) possesses the least earliness from \( d_j \). The reason behind performing this operation is due to the Property 2, i.e., either of \( d_i \) or \( d_j \) can lie at the completion time of a job. After every right shift, the values of \( DT \) and \( SD \) are updated for the next iteration as long as the property mentioned in Theorem 2 holds. For the case when the jobs are all of unit processing time, then \( \min \{ DT_i, SD_j \} \) is always equal to one and we do not need to update the two vectors but only sum up the earliness and tardiness penalties to check for Theorem 2.

VI. ALGORITHM RUN-TIME COMPLEXITY

In this section we study and prove the run-time complexity of Algorithms 1 and 2.

---

**Algorithm 1:** Exact algorithm for the general CDW for a given job sequence with a run-time complexity of \( O(n^2) \).

1. \( C_i = \sum_{k=1}^{i} p_k \; \forall \; i = 1, 2, \ldots, n \)
2. **Calculate** \( DT_i, SD_i \; \forall \; i \)
3. \( A_i = DT_i \; \forall \; i \)
4. \( B_i = SD_i \; \forall \; i \)
5. loop \( \leftarrow 1 \)
6. **while** loop \( \leq n \) **do**
   7. \( n1 = \arg \max_{i=1,2,\ldots,n} \{DT_i < 0\} \)
   8. \( n2 = \arg \max_{i=1,2,\ldots,n} \{SD_i < 0\} \)
   9. \( \tau = \min \{-DT_{n1}, -SD_{n2}\} \)
   10. **if** \( \tau \neq 0 \) **then**
      11. \( DT_i = DT_i + \tau \; \forall \; i \)
      12. \( SD_i = SD_i + \tau \; \forall \; i \)
      13. \( i1 = \arg \max_{i=1,2,\ldots,n} \{SD_i > 0\} \)
      14. \( i2 = \arg \max_{i=1,2,\ldots,n} \{DT_i \leq 0\} \)
      15. \( pe = \sum_{i=i1}^{i2} \alpha_i \)
      16. \( pl = \sum_{i=i1}^{i2} \beta_i \)
      17. **if** \( pl < pe \) **then**
         18. \( A_i = DT_i \; \forall \; i \)
         19. \( B_i = SD_i \; \forall \; i \)
   20. **loop** \( \leftarrow \) **loop** + 1
   21. \( DT_i = A_i \; \forall \; i \)
   22. \( SD_i = B_i \; \forall \; i \)
23. **Calculate** \( D_i, S_i, PL_i \; \forall \; i \)
24. \( Sol = \sum_{i=1}^{n} \{(D_i + S_i) \cdot PL_i\} \)
25. **return** Sol

**Algorithm 2:** Linear algorithm for a given sequence for the CDW with unit-time job processing times.

1. \( C_i = \sum_{k=1}^{i} p_k \; \forall \; i = 1, 2, \ldots, n \)
2. \( DT_i = C_i - d_i \; \forall \; i \)
3. \( SD_i = C_i - d_r \; \forall \; i \)
4. \( n1 = \arg \max_{i=1,2,\ldots,n} \{DT_i < 0\} \)
5. \( n2 = \arg \max_{i=1,2,\ldots,n} \{SD_i < 0\} \)
6. \( pe = \sum_{i=1}^{n1} \alpha_i \)
7. \( pl = \sum_{i=n2}^{n} \beta_i \)
8. \( t = 0 \)
9. **loop** \( \leftarrow 1 \)
10. **while** loop \( \leq n \) **do**
   11. \( n2 = n2 - 1 \)
   12. \( pe = pe - \alpha_n \)
   13. \( pl = pl + \beta_n \)
   14. \( n1 = n1 - 1 \)
   15. **if** \( pl < pe \) **then**
      16. \( t \leftarrow \) **loop**
      17. **loop** \( \leftarrow \) **loop** + 1
   18. \( DT_i = DT_i + t \; \forall \; i \)
   19. \( SD_i = SD_i + t \; \forall \; i \)
20. **Calculate** \( D_i, S_i, PL_i \; \forall \; i \)
21. \( Sol = \sum_{i=1}^{n} \{(D_i + S_i) \cdot PL_i\} \)
22. **return** Sol
TABLE I. Results obtained for single machine common due-window problem till 50 jobs. For each job there are 10 different instances each with a value for $\kappa$ and for each $\kappa$ there are 5 different due-windows.

<table>
<thead>
<tr>
<th>$k$</th>
<th>$h_1 - h_2$</th>
<th>10*</th>
<th>20</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>BR</td>
<td>Algo 1+SA</td>
<td>BR</td>
</tr>
<tr>
<td>1</td>
<td>0.1 - 0.2</td>
<td>1896</td>
<td>1896</td>
<td>4089</td>
</tr>
<tr>
<td></td>
<td>0.1 - 0.3</td>
<td>1330</td>
<td>1330</td>
<td>2713</td>
</tr>
<tr>
<td></td>
<td>0.2 - 0.5</td>
<td>540</td>
<td>540</td>
<td>1162</td>
</tr>
<tr>
<td></td>
<td>0.3 - 0.4</td>
<td>919</td>
<td>919</td>
<td>2294</td>
</tr>
<tr>
<td></td>
<td>0.3 - 0.5</td>
<td>587</td>
<td>587</td>
<td>1559</td>
</tr>
<tr>
<td>2</td>
<td>0.1 - 0.2</td>
<td>947</td>
<td>947</td>
<td>8251</td>
</tr>
<tr>
<td></td>
<td>0.1 - 0.3</td>
<td>539</td>
<td>539</td>
<td>5950</td>
</tr>
<tr>
<td></td>
<td>0.2 - 0.5</td>
<td>191</td>
<td>191</td>
<td>2770</td>
</tr>
<tr>
<td></td>
<td>0.3 - 0.4</td>
<td>432</td>
<td>432</td>
<td>4482</td>
</tr>
<tr>
<td></td>
<td>0.3 - 0.5</td>
<td>265</td>
<td>265</td>
<td>2923</td>
</tr>
<tr>
<td>3</td>
<td>0.1 - 0.2</td>
<td>1488</td>
<td>1488</td>
<td>5881</td>
</tr>
<tr>
<td></td>
<td>0.1 - 0.3</td>
<td>1012</td>
<td>1012</td>
<td>4067</td>
</tr>
<tr>
<td></td>
<td>0.2 - 0.5</td>
<td>398</td>
<td>398</td>
<td>1675</td>
</tr>
<tr>
<td></td>
<td>0.3 - 0.4</td>
<td>760</td>
<td>760</td>
<td>3035</td>
</tr>
<tr>
<td></td>
<td>0.3 - 0.5</td>
<td>462</td>
<td>462</td>
<td>1270</td>
</tr>
<tr>
<td>4</td>
<td>0.1 - 0.2</td>
<td>2128</td>
<td>2128</td>
<td>8977</td>
</tr>
<tr>
<td></td>
<td>0.1 - 0.3</td>
<td>1576</td>
<td>1576</td>
<td>6609</td>
</tr>
<tr>
<td></td>
<td>0.2 - 0.5</td>
<td>712</td>
<td>712</td>
<td>3113</td>
</tr>
<tr>
<td></td>
<td>0.3 - 0.4</td>
<td>462</td>
<td>462</td>
<td>1270</td>
</tr>
<tr>
<td></td>
<td>0.3 - 0.5</td>
<td>740</td>
<td>740</td>
<td>2823</td>
</tr>
<tr>
<td>5</td>
<td>0.1 - 0.2</td>
<td>1479</td>
<td>1479</td>
<td>6306</td>
</tr>
<tr>
<td></td>
<td>0.1 - 0.3</td>
<td>1012</td>
<td>1012</td>
<td>4067</td>
</tr>
<tr>
<td></td>
<td>0.2 - 0.5</td>
<td>398</td>
<td>398</td>
<td>1675</td>
</tr>
<tr>
<td></td>
<td>0.3 - 0.4</td>
<td>760</td>
<td>760</td>
<td>3035</td>
</tr>
<tr>
<td></td>
<td>0.3 - 0.5</td>
<td>462</td>
<td>462</td>
<td>1270</td>
</tr>
<tr>
<td>6</td>
<td>0.1 - 0.2</td>
<td>2093</td>
<td>2093</td>
<td>10204</td>
</tr>
<tr>
<td></td>
<td>0.1 - 0.3</td>
<td>1576</td>
<td>1576</td>
<td>7492</td>
</tr>
<tr>
<td></td>
<td>0.2 - 0.5</td>
<td>712</td>
<td>712</td>
<td>3113</td>
</tr>
<tr>
<td></td>
<td>0.3 - 0.4</td>
<td>462</td>
<td>462</td>
<td>1270</td>
</tr>
<tr>
<td></td>
<td>0.3 - 0.5</td>
<td>740</td>
<td>740</td>
<td>2823</td>
</tr>
<tr>
<td>7</td>
<td>0.1 - 0.2</td>
<td>1644</td>
<td>1644</td>
<td>3749</td>
</tr>
<tr>
<td></td>
<td>0.1 - 0.3</td>
<td>1287</td>
<td>1287</td>
<td>2519</td>
</tr>
<tr>
<td></td>
<td>0.2 - 0.5</td>
<td>670</td>
<td>670</td>
<td>991</td>
</tr>
<tr>
<td></td>
<td>0.3 - 0.4</td>
<td>952</td>
<td>952</td>
<td>1801</td>
</tr>
<tr>
<td></td>
<td>0.3 - 0.5</td>
<td>680</td>
<td>680</td>
<td>1069</td>
</tr>
<tr>
<td>8</td>
<td>0.1 - 0.2</td>
<td>1466</td>
<td>1466</td>
<td>3317</td>
</tr>
<tr>
<td></td>
<td>0.1 - 0.3</td>
<td>1121</td>
<td>1121</td>
<td>2342</td>
</tr>
<tr>
<td></td>
<td>0.2 - 0.5</td>
<td>492</td>
<td>492</td>
<td>1056</td>
</tr>
<tr>
<td></td>
<td>0.3 - 0.4</td>
<td>772</td>
<td>772</td>
<td>1767</td>
</tr>
<tr>
<td></td>
<td>0.3 - 0.5</td>
<td>513</td>
<td>513</td>
<td>1187</td>
</tr>
<tr>
<td>9</td>
<td>0.1 - 0.2</td>
<td>1835</td>
<td>1835</td>
<td>4673</td>
</tr>
<tr>
<td></td>
<td>0.1 - 0.3</td>
<td>1384</td>
<td>1384</td>
<td>3266</td>
</tr>
<tr>
<td></td>
<td>0.2 - 0.5</td>
<td>691</td>
<td>691</td>
<td>1355</td>
</tr>
<tr>
<td></td>
<td>0.3 - 0.4</td>
<td>1047</td>
<td>1047</td>
<td>2419</td>
</tr>
<tr>
<td></td>
<td>0.3 - 0.5</td>
<td>717</td>
<td>717</td>
<td>1474</td>
</tr>
</tbody>
</table>
Lemma 1. The run-time complexities of Algorithm 1 and 2 are $O(n^2)$ and $O(n)$, respectively, where $n$ is the total number of jobs.

Proof: It can easily observed that the complexity of the Algorithm 1 is dependent on the while loop in line 6. The computations of $n_1$, $n_2$, $DT$, $SD$, $i_1$ and $i_2$ are all of $O(n)$ and are computed $n$ times, in the worst case. Hence, the complexity of Algorithm 1 is $O(n^2)$. However, in Algorithm 2 we do not need to compute these parameters after every iteration of the while loop. All the computations inside both the while loops are of $O(n)$ and so are the calculations of $n_1$, $n_2$, $SD$, $DT$, $PL$ and $Sol$. Hence, the run-time complexity of Algorithm 2 is $O(n)$.

VII. RESULTS

We now present our computational results for the two cases of CDW, discussed in the paper. We use the CDW benchmark instances provided by Biskup and Feldmann in [2] and compare our results with theirs.

As described in [23], [24], [25], we use a modified Simulated Annealing algorithm to generate job sequences and Algorithm 1 and 2 to optimize each sequence to its minimum penalty for the two cases. Our experiments over all the instances suggest that an ensemble size of $4 + n/10$ and the maximum number of iterations of $500 \cdot n$, 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 I are obtained on average after 10 different replication. The initial temperature 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 [26]. An exponential schedule for cooling is adopted with a cooling rate of 0.9999 with the Metropolis acceptance criterion, $\min\{1, \exp((-\Delta E)/T)\}$ [26].

The modification from the standard SA is the increase in the temperature after the annealing temperature becomes less than 1 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 SA. Elitism has been successfully adopted in evolutionary algorithms for several complex optimization problems [27], [28]. Theoretical studies have been made analysing speed-ups in parallel evolutionary algorithms combinatorial optimization problems in [29], [30]. We observed that this concept works well for the CDD and the Aircraft Landing Problem problem [23], [24]. 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 $3 + \lfloor \sqrt{n/5} \rfloor$, where $n$ is the number of jobs. For large instances the size of this permutation is quite small, but we have observed that it works well with our modified simulated annealing algorithm.

Table I presents our results for the CDW where the due-window size for any instance is calculated using the values of $h_1$ and $h_2$. A due-window has a left ($d_l$) and right ($d_r$) due-date, where $d_l = |h_1 \cdot \sum_{i=1}^{n} p_i|$ and $d_r = |h_2 \cdot \sum_{i=1}^{n} h_i|$, as described in [2]. For the first 50 instances with 10 jobs we obtain the optimal solution for all the instances. For the remaining 100 instances, we achieve better results for 43 instances than Biskup and Feldmann [2], equal results for 55 and for only two instances we do not reach the best known solution value but are within a percentage gap of 0.537 and 0.342.

Table II shows our results for the same instances, but for unit-time processing time of all the jobs. We use Algorithm 2 to optimize any job sequence and the same modified SA to find the best processing sequence. Since, these instances have not been studied for the unit-time processing times, we are unable to compare our results with the literature. Table III shows the run-time in seconds for all the instances for both the cases. The run-times shown are the mean time for all the 10 different instances for each job, averaged over 10 replications of our approach.

VIII. CONCLUSION AND FUTURE DIRECTION

In this paper we present a novel property for the problem of scheduling against a common due-window for the general and the unit processing time cases. We present a theoretical
study for the CDW and its similarity to the CDD. Thereafter, we present an $O(n^2)$ algorithm for a the distinct processing time case and an $O(n)$ algorithm for the unit processing time case, to optimize a given job sequence and prove the runtime complexity and its optimality with respect to the solution value. We applied our algorithms to the benchmark instances provided by Biskup and Feldmann [2]. In the future we intend to implement metaheuristic approaches using graphics processing units (GPUs) and provide speed-ups in runtime for all the instances.

ACKNOWLEDGEMENTS

The research project was promoted and funded by the European Union and the Free State of Saxony, Germany. The authors take the responsibility for the content of this publication.

REFERENCES


This is a preview version of the paper [1] (see below for the reference). Read the full piece in the proceedings.

@inproceedings{ALKW2014CDWPPAFAGPS,
  author = {Abhishek Awasthi and J{"{o}}rg L{"{a}}ssig and Oliver Kramer and Thomas Weise},
  title = {Common Due-Window Problem: Polynomial Algorithms for a Given Processing Sequence},
  booktitle = {Proceedings of the IEEE Symposium on Computational Intelligence in Production and Logistics Systems (CIPLS’14), Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI’14)},
  publisher = {Los Alamitos, CA, USA: IEEE Computer Society Press},
  address = {Orlando, FL, USA: Caribe Royale All-Suite Hotel and Convention Center},
  year = {2014},
  month = dec # {9--12, },
  pages = {32--39},
  doi = {10.1109/CIPLS.2014.7007158},
}