Single Machine Scheduling with Interfering Job Sets

Ketan Khowala, John Fowler, Ahmet Keha, Hari Balasubramanian

Abstract We consider a bicriteria scheduling problem for a single machine in which jobs belong to either of two different disjoint sets, each set having its own performance measure. The problem has been referred to as interfering job sets in the scheduling literature and also been called multi-agent scheduling where each agent’s objective function is to be minimized. The performance criteria specifically selected in this paper are minimizing total completion time and number of tardy jobs for the two set of jobs. We present a forward SPT-EDD heuristic to generate the set of all non-dominated solutions for this problem considering a single machine environment where all the jobs are available at time zero and the jobs are not allowed to be preempted. The complexity of this specific problem is still open; however some pseudo-polynomial algorithms have been suggested by earlier researchers which have been used to compare the results from the proposed heuristic.

1 Introduction

Motivated by multiple objectives and tradeoffs that a decision maker has to make between conflicting objectives, multicriteria scheduling problems have been widely dealt with in the literature. In this domain of scheduling problem one typically has to satisfy multiple criteria on the same set of jobs. However, in some cases jobs belong to different job sets and must be processed using the same resource (or competing for the same machine), hence causing interference. These job sets can have different criteria to be minimized. These different sets may represent different customers or different agents whose requirements may differ. The complexity of this domain of problems can be attributed to the number of different job sets considered, performance criteria and restrictions on each set of jobs and the machine environment.

One of the earliest references on this subject is Peha [1995] which dealt with the problem of interfering job sets with objectives of minimizing weighted number of tardy jobs in one set and total weighted completion time in another set of jobs with unit processing time under an identical parallel machine environment. The assumption of unit processing times makes the problem easier to solve. The paper from Baker and Smith [2003] was the first paper formalizing scheduling problems with interfering job sets. They considered a single machine
problem involving the minimization of criteria including makespan, maximum lateness, and total weighted completion time \( (C_{\text{max}}, L_{\text{max}}, \sum w_j C_j) \). They showed that any combination of these criteria on different job sets can be solved in polynomial time by defining the optimization function as a linear combination of the criteria on different job sets, except the combination of \( \sum w_j C_j \) and \( L_{\text{max}} \) on different job sets which turns out to be NP-hard.

Agnetis et al. [2004] presented the complexity of generating non-dominated solutions for single machine as well as shop floor scheduling problems with interfering job sets, involving objectives of minimizing total completion times, total number of tardy jobs and total weighted completion times \( (\sum C_j, \sum U_j, \sum w_j C_j) \). Ng et al. [2006] proved that the problem involving the jobs sets with total completion time and total number of tardy jobs on a single machine is NP-hard under high multiplicity encoding and have presented a pseudo polynomial time algorithm for this problem. Cheng et al. [2006] have shown NP completeness of the problem where jobs belong to one of the multiple sets and each set is having the objective of minimizing the total weighted number of tardy jobs \( (\sum w_j U_j) \). They have also presented a polynomial time approximation scheme for this problem.

In the subsequent sections, we define our problem, talk about the structure of the problem and some key properties of the problem, present the forward SPT-EDD heuristic, and compare the computational performance of the non-dominated solution sets obtained from our heuristic with the Pareto optimal solution sets obtained by the pseudo polynomial algorithm by Ng et al. [2006].

2 Problem Description

The specific problem that we are looking into relates to single machine scheduling where all jobs are available at time zero and no preemption is allowed. We look at the single machine problem with interfering jobs from two disjoint sets, one having the objective of minimizing total completion time and other minimizing total number of tardy jobs. As discussed earlier, the complexity of this problem is still open. A pseudo- polynomial algorithm is presented by Ng et al. [2006] for this problem under binary encoding. We combine some of the intuition from Moore’s algorithm to determine the initial set of jobs that can be on time and then use a forward SPT-EDD heuristic to determine all the non-dominated points for this problem.

There are two disjoint interfering sets of jobs \( \xi_1 \) and \( \xi_2 \) with \( n_1 \) and \( n_2 \) number of jobs in each respective set. The total number of jobs that need to be scheduled is \( n = (n_1 + n_2) \). We seek to minimize the total completion time of the jobs in first set \( \xi_1 \) and for the jobs in second set \( \xi_2 \) we want to minimize the total number of tardy jobs. The processing time of the jobs in set \( \xi_1 \) and sets \( \xi_2 \) are represented by \( p^1_j \) and \( p^2_j \), respectively. Similarly the due dates for the jobs in the first set and the second set can be denoted by \( d^1_j \) and \( d^2_j \) respectively. However, for the purpose of the objectives considered herein, due dates are only relevant for the jobs in second set.

This problem can be denoted as \( 1 \mid \text{inter} \mid ND(\sum C_j, \sum U_j) \) using the Graham et al. [1979] notation. Clearly the notation highlights the interference between the job sets and \( ND(\sum C_j, \sum U_j) \) indicates that we are looking to find the non-dominated (or Pareto
optimal) points for this problem. The non dominated point will help a decision maker to determine the tradeoffs between the interfering sets of jobs competing for the same resource.

A solution \( X^* \) is Pareto optimal or non-dominated if there exists no other solution \( X \in S \) for which \( z_i(X) \leq z_i(X^*) \) and \( z_2(X) \leq z_2(X^*) \) where at least one of the inequalities is strict. Jaszkiewicz [2003] describes methods for evaluating the performance of multi-objective heuristics.

3 Structure of the Non Dominated Solutions

If considered independently, the single machine problems are easy to solve. Sorting the jobs in non decreasing order of processing times solves the problem of \( 1|| \sum C_j \) while the polynomial time Moore’s algorithm solves the problem of \( 1|| \sum U_j \). However, the complexity of these performance measures with interference is yet not known. There are a few important observations / properties regarding the non dominated solution for interfering job sets and with these objectives that can be observed in the following lemmas and help further explore the structure of the non dominated solutions.

Lemma 1: There always exists a non-delay schedule for all the strongly non dominated points on the Pareto optimal front.

Lemma 2(a): For all the strongly non dominated points, there exists an optimal schedule in which jobs in the job set \( S_1 \) are scheduled in SPT order, Ng et al. [2006].

2(b): For all the strongly non dominated points, there exists an optimal schedule in which jobs in the job set \( S_2 \) that are on time are scheduled in EDD order, Ng et al. [2006].

Lemma 3: For any non dominated point, the performance criteria \( \sum C_j \), for the jobs in the job set \( S_1 \) with preemptive scheduling remains the same as with non preemptive scheduling, provided the jobs in job set \( S_2 \) which caused the preemption is scheduled before the job that got preempted from \( S_1 \).

We define three subset of jobs \( S_1, S_2, S_3 \). Based on the above observation, for any non dominated point, the subset of jobs \( S_1 \) will contain all the jobs from \( S_1 \) arranged in SPT order, another subset \( S_2 \) of on time jobs from set \( S_2 \) which will be in EDD order and third subset \( S_3 \) of jobs that are tardy from set \( S_2 \) as well. The set \( S_3 \) can be arranged in any order after sets \( S_1 \) and \( S_2 \) without affecting the criteria (the interference is only between sets \( S_1 \) and \( S_2 \)). This is represented in the picture below as well.

Figure 1: Structure of the non dominated point with total completion time and number tardy jobs as performance criteria
Consider the following graph which represents the structure of the efficient frontier for this problem (set of non dominated points). Let the x-axis represent the criteria of minimizing $\sum C_j$ for job set $\xi_1$ and y-axis represent the criteria of minimizing $\sum U_j$ for job set $\xi_2$.

![Efficient Frontier Graph](image)

Figure 2: Efficient frontier representing non dominated points for job in set $C_1$ and $C_2$.

The points $Q_0$, $Q_1$, $Q_2$ and $Q_3$ in the above graph in Figure 2 represents the strongly non dominated points on the efficient frontier. The point $Q_3$ gives the best value of total completion time for jobs in set $\xi_1$. Similarly $Q_0$ gives the best value of total number of jobs that are on time from set $\xi_2$. The point $Q'_0$ is the point which is weakly non-dominated by the jobs in set $\xi_1$ and point $Q'_3$ is weakly non-dominated by the jobs in set $\xi_2$. The non dominated point $Q_0$ can be represented by $1|\text{inter}|\sum C_j, \sum U_j = Y_{\text{min}}$, where $Y_{\text{min}}$ is the minimum number of tardy jobs obtained by solving $1||\sum U_j$ for second set without interference. Similarly non dominated point $Q_3$ can be represented by $1|\text{inter}|\sum C_j = K_{\text{min}}, \sum U_j$, where $K_{\text{min}}$ is the minimum number of total completion time obtained by solving $1||\sum C_j$ for first set without interference.

4 Forward SPT-EDD Heuristic

Based on the above discussion about the structure of the non dominated point and the efficient frontier of this particular problem, we can further draw an additional observation to help build a forward SPT-EDD heuristic for this problem. This observation is illustrated in the following lemma.

Lemma 4: Under the restriction that the jobs that were tardy at one non dominated point will also remain tardy at the next non dominated point as we move in the direction of improving $\sum C_j$ (jobs from set $S_2$ are not allowed to move back to set $S_2$); the one job that needs to be moved from set $S_2$ to set $S_3$ (new jobs to become tardy now) will be the one which when removed from the schedule provides the best preemptive schedule for all
the jobs in job set $S_1$ without moving the position of other jobs in set $S_2$ (hence the best improvement in the value of total completion time).

With these four distinct properties of this problem discussed so far, we present a forward SPT-EDD algorithm that can be used to generate the non dominated points for this problem. In the forward logic presented below, we start with the Moore’s algorithm to determine the initial sets $S_2$ and $S_3$. As this logic is under certain assumption and restriction, it may not yield optimal solution. In the subsequent section we do compare the computational efficiency of this algorithm with the optimal solutions from the pseudo polynomial algorithm.

The Forward SPT-EDD algorithm can be summarized in the following steps (here we start from point the initial point $1 | \text{inter} | \sum C_j, \sum U_j = Y_{\text{min}}$ and then walk through point $1 | \text{inter} | \sum C_j = K_{\text{min}}, \sum U_j$ by moving jobs from set $S_2$ to $S_3$):

**Step 1:** Use Moore’s algorithm to determine the minimum number of tardy jobs by considering jobs in set $\xi_2$ alone. The solution from Moore’s algorithm will help determine the sets $S_2$ and $S_3$. The tardy jobs are placed in set $S_3$ while the on time jobs will be placed in set $S_2$. Note: This step may result in a non dominated solution that is not optimal, as the division between jobs from $\xi_2$ in sets $S_2$ and $S_3$ obtained by Moore’s algorithm, may not result in best possible value for $\sum C_j$ for jobs in the first set.

**Step 2:** With this initial division for set $\xi_2$ into sets $S_2$ and $S_3$, a non dominated solution can be determined for interfering jobs sets $S_1$ and $S_2$ using Lemma 2a and Lemma 2b.

**Step 3:** First the jobs in set $S_2$ are arranged in the EDD order in such a way that there is no earliness for the jobs in set $S_2$, except when there is an overlap between jobs within set $S_2$. In case of overlap, jobs with earlier due dates are placed ahead of jobs with later due dates. Now the jobs in set $S_1$ are arranged in SPT order allowing preemption. We finally use the property described in Lemma 3 to get the non preemptive schedule for this non dominated point.

**Step 4:** Now we move jobs from set $S_2$ to set $S_3$, using the property described in Lemma 4 to find the subsequent non dominated point and move in the direction which brings improvement in $\sum C_j$. Note: Because of the restriction made in Lemma 4, as we proceed along the frontier to find the subsequent non dominated points, we are not considering the jobs which were tardy and in set $S_3$ at earlier points on the frontier to be on time in the subsequent points. We may miss some opportunity of improving the criteria for the job set $S_1$ because of this. The example below illustrates this gap.

We consider an **example** with 5 jobs in each set of jobs $\xi_1$ and $\xi_2$ being represented by $p_j^1$ and $p_j^2$. For simplicity before numbering the jobs, jobs in set $\xi_1$ are arranged in SPT order while the jobs in set $\xi_2$ are arranged in EDD order. The final sequence at any non-dominated point is divided into 3 sets: $S_1$ which includes all jobs from $\xi_1$ arranged in SPT order, $S_2$ which includes on time jobs from set $\xi_2$ and $S_3$ which includes tardy jobs from set $\xi_2$.

For set $\xi_1$: $p_1^1=1$, $p_2^1=2$, $p_3^1=3$, $p_4^1=5$, $p_5^1=5$
For set $\xi_2$: $p_1^2 = 3$, $p_2^2 = 6$, $p_3^2 = 4$, $p_4^2 = 5$, $p_5^2 = 3$

$d_1^2 = 5$, $d_2^2 = 11$, $d_3^2 = 18$, $d_4^2 = 25$, $d_5^2 = 30$

The value of $\sum C_j$ for problem $1 || \sum C_j$ is 37, while $\sum U_j$ for problem $1 || \sum U_j$ is zero. In iteration (1) since all the jobs in set $\xi_2$ are on time, $S_2 = \{1, 2, 3, 4, 5\}$ and $S_3 = \{\}$. Jobs in set $S_1$ are arranged in SPT order while jobs in set $S_2$ are arranged in EDD order. This results in point $Q_0 (101, 0)$.

**Figure 3(a):** In the first step jobs in $S_1$ are allowed to be preempted. In the second step jobs in set $S_2$ are moved ahead to avoid preemption of jobs in set $S_1$. Note that the completion time of the jobs in $S_1$ remains the same.

In iteration (2), it is found that moving job #2 from set $S_1$ to $S_3$ will provide maximum improvement in $\sum C_j$, hence $S_2 = \{1, 3, 4, 5\}$ and $S_3 = \{2\}$. This gives the non-dominated point $Q_1 (61, 1)$.

**Figure 3(b):** In the first step job #2 from set $S_2$ is moved to $S_3$ and jobs in set $S_1$ are allowed to be preempted. In the second step jobs in set $S_2$ are moved ahead to avoid preemption of jobs in set $S_1$. Note that the completion time of the jobs in $S_1$ remains the same.

In iteration (3) it is found that moving job #1 from set $S_2$ to $S_3$ will provide maximum improvement in $\sum C_j$, hence $S_2 = \{3, 4, 5\}$ and $S_3 = \{2, 1\}$. This gives the non-dominated point $Q_2 (41, 2)$.

**Figure 3(c):** In the first step job #1 from set $S_2$ is moved to $S_3$ and jobs in set $S_1$ are allowed to be preempted. In the second step jobs in set $S_2$ are moved ahead to avoid preemption of jobs in set $S_1$. Note that the completion time of the jobs in $S_1$ remains the same.

In iteration (4) it is found that moving job #3 from set $S_2$ to $S_3$ will provide maximum improvement in $\sum C_j$, hence $S_2 = \{4, 5\}$ and $S_3 = \{2, 1, 3\}$. This gives the non-dominated point $Q_3 (37, 3)$. 

---

362
Figure 3(d): Job #3 from set $S_2$ is moved to $S_3$ and jobs in set $S_1$ are already in a non-preemptive SPT order.

Now since the best possible value of $\sum C_j$ is 37, making more jobs tardy will not provide any further improvement in $\sum C_j$, hence will result in weakly non-dominated points: $Q_4(37, 4)$ and $Q_5(37, 5)$.

5 Computational Experiments

To test the computational efficiency of the forward SPT-EDD heuristic we generate some sample data set and compare it against the results from the pseudo-polynomial algorithm of Ng et al. [2006]. We consider cases with 20, 30, 40 and 50 jobs in each sets $\xi_1$ and $\xi_2$ and generate six problem instances for each case. We select integer processing time numbers for both the set of jobs from $\sim U[1, 20]$. The due date, $d'_j$, of job $j$ is an integer generated from the uniform distribution $[P(L-R/2), P(L+R/2)]$, where $P = 0.5 P_1 + P_2$ ($P_1$ is the sum of processing time in of jobs in set $\xi_1$ and $P_2$ is the sum of processing time of jobs in set $\xi_2$) and the two parameters $L$ and $R$ are relative measures of the location and range of the distribution, respectively. We choose different values for $L$ and $R$ to generate three different ranges of due dates: $[0.1 P, 0.9 P]$, $[0.5 P, 0.9 P]$, and $[0.3 P, 1.1 P]$ and generate two problem instances for each range, hence generating six problem instances in total for each number of jobs.

Table 1 summarizes the comparison of average solution quality of the algorithm over 24 instances comparing it with the optimal solution obtained from the pseudo-polynomial algorithm. For the forward SPT-EDD heuristic the average percentage gap between our solution and the Pareto optimal point is less than 1%. Although the average number of non-Pareto optimal points within a solution set increases with the number of jobs in $\xi_2$, it remains in the range of 20-25% of the total number of points in the solution set.

<table>
<thead>
<tr>
<th># of Jobs in Each Set</th>
<th>Total Number of Test Instances</th>
<th>Forward SPT-EDD</th>
<th>Avg. Number of Non Pareto Optimal Points</th>
<th>Avg. % Gap of Non Pareto Optimal Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>6</td>
<td>3.8</td>
<td>1.14%</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>6</td>
<td>7.8</td>
<td>0.62%</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>6</td>
<td>7.7</td>
<td>0.47%</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>6</td>
<td>13.8</td>
<td>0.30%</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Summary of solution quality over 24 problem instances for the Forward SPT-EDD Heuristic.

In Table2 below we present the average computational time for the algorithm over 6 different problem instances for each number of jobs. It can be observed that the computational time for the pseudo-polynomial DP algorithm grows faster with the increase in the number of jobs in both sets and the total processing time of jobs in set $\xi_2$, an average of about 37 seconds for the 50-job instances. The computational time for the forward SPT-EDD heuristic is under 1 second even for the 50-job instances. The effect of computational complexity can be seen in the computation times. The pseudo-polynomial algorithm has a computational complexity of
\(O(n^2P^2)\) while the computational complexity of the proposed forward SPT-EDD heuristic is \(O(n^2)\).

<table>
<thead>
<tr>
<th># of Jobs in Each Set</th>
<th>Total Number of Test Instances</th>
<th>Average Run Time (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Dynamic Program</td>
</tr>
<tr>
<td>20</td>
<td>6</td>
<td>0.898</td>
</tr>
<tr>
<td>30</td>
<td>6</td>
<td>4.495</td>
</tr>
<tr>
<td>40</td>
<td>6</td>
<td>14.984</td>
</tr>
<tr>
<td>50</td>
<td>6</td>
<td>36.703</td>
</tr>
</tbody>
</table>

Table 2: Summary of average computational time over 24 problem instances for the Forward SPT-EDD Heuristic.

6 Conclusion and Feature Research

The proposed polynomial heuristic does a good job of providing a (nearly) non-dominated solution set; the average gap is less than 1% compared to the optimal solution. The computational experiment can be continued to see the effect of the increased run time with a larger number of jobs with the pseudo-polynomial algorithm. It can be clearly seen that this SPT-EDD heuristic performs quite well and will be useful in solving job sets each with a larger number of jobs e.g. 200 or higher. Similar approach could be adopted to solve other interfering job set problems with different performance criteria and even problems which have been classified as NP-hard. We further intend to carry our more computational results as well as explore the structure of the single machine problems with two interfering job sets with criterion of total weighted completion time and number of tardy jobs as well as total weighted completion time and maximum lateness. Further it will be interesting to see if the EDD rule will still hold for on time jobs or jobs with the criteria of minimizing maximum lateness when interfering with another job set that has the criterion to minimize total weighted completion time. As we are researching and investigating these problems in several dimensions, we will discuss additional results and findings during our conference presentation.

References