Computers & Operations Research 40 (2013) 248–252

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

A theoretical development for the total tardiness problem and its application in branch and bound algorithms

Shuyu Zhou [a,b], Zhaohui Liu [a,*]

[a] Department of Mathematics, East China University of Science and Technology, Shanghai 200237, China
[b] Rotterdam School of Management, Erasmus University, Post Box 1738, 3000 DR Rotterdam, the Netherlands

ARTICLE INFO

Available online 23 June 2012

Keywords:
Scheduling
Single machine
Total tardiness
Branch and bound

ABSTRACT

This paper deals with the single machine total tardiness problem, and proves that if the job sequences produced by two heuristics, named as Time Forward and Time Backward algorithms, have the same starting and ending job subsequences, then there exists an optimal job sequence with the starting and ending job subsequences. The computation experiments show that there is a significant improvement of the running time of a branch and bound algorithm with the incorporation of the new property.

1. Introduction

The single machine total tardiness problem (TTP) can be stated as follows. There is a set $N=\{1,2,\ldots,n\}$ of n jobs to be processed non-preemptively on a continuously available single machine which can handle only one job at a time. Job i ($i\in N$) becomes available at time zero, requires a processing time p_i ($p_i>0$) and has a due date d_i ($-\infty<d_i<+\infty$). When a sequence $\sigma=(\sigma(1),\sigma(2),\ldots,\sigma(n))$ of the jobs in N is prescribed, then for each job $\sigma(i)$, its completion time $C_{\sigma(i)}=\sum_{k\le i}p_{\sigma(k)}$ and tardiness $T_{\sigma(i)}=\max\{0,C_{\sigma(i)}-d_{\sigma(i)}\}$ can be computed. The task is to find a job sequence σ such that the total tardiness

$$TT(\sigma)=\sum_{i=1}^{n}T_{\sigma(i)}=\sum_{i=1}^{n}\max\{0,C_{\sigma(i)}-d_{\sigma(i)}\}$$

is minimized.

TTP is NP-hard in the ordinary sense [5] and most optimization algorithms use a combination of dynamic programming (DP) and branch and bound (BB) methods. They proceed by breaking the problem into subproblems by the decomposition principle introduced by Lawler [11], and strengthened by Potts and Van Wassenhove [16], Szwarc [17] and Chang et al. [2]. At the same time, the dominance conditions derived by Emmons [6] are used extensively to curtail the solution space. The most effective algorithms are by Szwarc et al. [18] and by Tansel et al. [19] that can both solve instances with up to 500 jobs. Szwarc et al. considered several combinations of methods like adding lower bounds, using modified due date in the decomposition position rule, and controlling subproblem list by only storing subproblems that need branching. They reported that SDD_2/d_i which only utilized the last method was the most efficient algorithm among other combinations for 500 job instances. Tansel et al. defined a so-called β-sequence in which the jobs are ordered in non-decreasing order of their modified due dates obtained through the use of Emmons' conditions, and proved that the β-sequence is optimal under some conditions. Then, for each subproblem, the sequence can be fixed as the β-sequence if the corresponding conditions are fulfilled.

The most well-known heuristic for TTP is by Wilkerson and Irwin [21], and then simplified by Baker and Bertrand [1], Lin [13], Panwalkar et al. [15]. Since the heuristic arranges the jobs one by one from time zero onwards using job pairwise interchanges, Yu [24] named it as Time Forward algorithm (TF). Naidu et al. [14] gave a sufficient condition under which TF is optimal. Yu [22] presented a heuristic, named by Time Backward algorithm (TB), which arranges the jobs one by one in the reverse order of time using job pairwise interchange rule. Note that Fadlalla et al. [7] also described another version of TB.

Some meta-heuristics for TTP have also been proposed in these years. The recent one is by Cheng et al. [3] that incorporates some elimination rules into an ant colony optimization algorithm. However, as pointed out by Koulamas [9], meta-heuristics do not perform better than those algorithms based on the decomposition principle for TTP, and perhaps they are more suitable to scheduling problems with less structure. Especially, based on the decomposition principle, a fully polynomial time approximation scheme (FPTAS) for TTP has been presented by Lawler [12], and then its computational complexity was improved by Kovalyov [10]. See Koulamas [8,9] and Sen et al. [20] for other researches related to the single machine total tardiness problem.

In this paper, we prove that if the job sequences produced by heuristics TF and TB have the same starting and ending job

* Corresponding author. Tel.: +86 21 64252308; fax: +86 21 64252018.
E-mail address: zhliu@ecust.edu.cn (Z. Liu).

0305-0548/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2012.06.005

subsequences, then there exists an optimal job sequence with the starting and ending job subsequences. Also, we perform the computation experiments and show that there is a significant improvement of the running time of BB algorithms with the incorporation of the new property.

The rest of the paper is organized as follows. Section 2 introduces the related work that stimulates our research. Section 3 deals with the property we propose. Section 4 presents a BB algorithm with the incorporation of the new property. The performance of the BB algorithm is evaluated in Section 5. Section 6 includes some concluding remarks.

2. Preliminaries

In this section we introduce some previous work that stimulates our research.

2.1. Consistent partial order

A partial order on the job set N can be described by a subset Q of $N^2=\{(i,j)|i,j\in N\}$, where $(i,j)\in Q$ $(i\neq j)$ means that job i precedes job j. Note that $(i,i)\in Q$ always holds from the reflexivity of partial order, and $Q_0=\{(i,i)|i\in N\}$ is called a null partial order. Q is called a consistent partial order (CPO) of TTP if there is an optimal sequence σ such that $\sigma^{-1}(i)<\sigma^{-1}(j)$ holds for any $(i,j)\in Q$ $(i\neq j)$, where $\sigma^{-1}(i)$ is defined as the position of job i in σ, and $\sigma^{-1}(j)$ is the position of job j. Obviously, appending a strong CPO to N can reduce the computational burden when we solve TTP especially by branch and bound algorithms.

Let $CPO(N)$ stand for the collection of all consistent partial orders on N. Given $Q\in CPO(N)$, we define $C_i^-(Q)=\sum_{(s,i)\in Q}p_s$ and $C_i^+(Q)=\sum_{(i,s)\notin Q}p_s+p_i$ for any job $i\in N$. $C_i^-(Q)$ and $C_i^+(Q)$ are called the earliest and the latest completion time of job i based on Q. Note that $C_i^-(Q_0)=p_i$ and $C_i^+(Q_0)=\sum_{k\in N}p_k$. Also, we define three subsets of N^2 as follows:

$$IC(Q)=\{(i,j)|i\neq j,\ p_i\le p_j,\ d_i\le\max\{d_j,C_j^-(Q)\}\},$$

$$BS(Q)=\{(i,j)|i\neq j,\ d_j\ge\min\{C_i^+(Q),\max\{d_i,C_i^+(Q)-p_j\}\},$$

$$DC(Q)=IC(Q)\cup BS(Q).$$

Emmons [6] proved that if $(i,j)\in DC(Q)$, then there exists an optimal sequence σ for TTP with $\sigma^{-1}(i)<\sigma^{-1}(j)$.

Assume that $Q\in CPO(N)$, $R\subseteq DC(Q)$ and $\{(j,i)|(i,j)\in R\}\cap Q=\emptyset$. Let $Q\oplus R$ be the partial order obtained from Q by adding the pairs in R and the pairs implied by transitivity. $Q\oplus R$ is called an augmented partial order from Q. Yu [23] proved that any augmented partial order obtained from the null partial order Q_0 by a series of augmentations is a CPO, i.e., the following lemma holds.

Lemma 1. *Let $R_k\subseteq DC(Q_k)$ and $Q_{k+1}=Q_k\oplus R_k$ for $k=0,1,2,\ldots$. Then each Q_k is a CPO.*

2.2. Heuristics

The heuristics TF and TB are as follows [1,13,22].

Time Forward Algorithm (TF).

*Step*1 Let $t=0$, $k=1$, and $S=N=\{1,2,\ldots,n\}$.
*Step*2 Determine $\sigma(k)$ such that $\max\{p_{\sigma(k)},d_{\sigma(k)}-t\}=\min_{i\in S}\max\{p_i,d_i-t\}$.
*Step*3 If $k=n$, then output the sequence σ and stop, else let $t=t+p_{\sigma(k)}$, $S=S\backslash\{\sigma(k)\}$, $k=k+1$ and return to Step 2.

Yu and Liu [25] proved that TF produces a locally optimal sequence with respect to the forward shifting neighborhood, which implies that no job in the sequence can be moved forward while keeping the other jobs unchangeable to generate a sequence with better performance. They also proved the worst-case performance ratio of TF is $n/2$, where the bound is tight. Given an instance I of TTP, let $TT^H(I)$ represent the total tardiness generated by heuristic H and $TT^*(I)$ denote the minimum total tardiness. Then, the worst-case performance ratio associated with heuristic H, is defined as $\sup_I\{TT^H(I)/TT^*(I)\}$, for all instances I.

Time Backward Algorithm (TB).

*Step*1 Let $t=\sum_{i=1}^n p_i$, $k=n$, $S=N=\{1,2,\ldots,n\}$.
*Step*2 Let $H=\{l|\ d_l+p_l\ge t,\ l\in S\}$.
*Step*3 If $H\neq\emptyset$, set $\sigma(k)$ as the job with largest due date in H; if $H=\emptyset$, set $\sigma(k)$ as the job with longest processing time in S.
Step 4 If $k=1$, then output the sequence σ and stop, else let $t=t-p_{\sigma(k)}$, $S=S\backslash\{\sigma(k)\}$, $k=k-1$ and return to Step 2.

Yu [22] proved that TB produces a locally optimal sequence with respect to the backward shifting neighborhood, and its worst-case performance ratio is 2^{n-2} that is tight.

Generally, TF performs better than TB. Since TF arranges the jobs from time zero onwards, TB arranges the jobs in the reverse order of time, and they both work according to the job pairwise interchange rule, we guess they are optimal when the resulting sequences are the same. We confirm the guess in the next section. In fact, we prove a stronger property that can be used in a BB algorithm.

3. A theoretical development

Let $\sigma_1=\alpha\beta_1\gamma$ and $\sigma_2=\alpha\beta_2\gamma$ be the job sequences produced by TF and TB, respectively. That is to say, the first $|\alpha|$ jobs in σ_1 and σ_2 are the same, and so are the last $|\gamma|$ jobs. We will show that TTP has an optimal sequence of the form $\alpha\beta\gamma$, where β is a permutation of the jobs in β_1 (or equivalently in β_2).

Lemma 2. *Assume that job i is processed before job j by both TF and TB, and $Q\in CPO(N)$. If $\sum_{k:\ \sigma_1^{-1}(k)<\sigma_1^{-1}(i)}p_k\le C_j^-(Q)-p_j$ and $\sum_{k:\ \sigma_2^{-1}(k)<\sigma_2^{-1}(j)}p_k\ge C_i^+(Q)-p_j$, then $(i,j)\in DC(Q)$.*

Proof. Since i is processed before j by TF, we have

$$\max\left\{p_i,d_i-\sum_{k:\sigma_1^{-1}(k)<\sigma_1^{-1}(i)}p_k\right\}\le\max\left\{p_j,d_j-\sum_{k:,\sigma_1^{-1}(k)<\sigma_1^{-1}(i)}p_k\right\}.$$

If $p_i\le p_j$, then

$$d_i\le\max\left\{p_j+\sum_{k:\sigma_1^{-1}(k)<\sigma_1^{-1}(i)}p_k,d_j\right\}\le\max\{C_j^-(Q),d_j\}.$$

Thus, $(i,j)\in IC(Q)$.

If $p_i>p_j$, then $d_i\le d_j$. And by TB

$$H=\left\{l\ |\ \sigma_2^{-1}(l)\le\sigma_2^{-1}(j),\ d_l+p_l\ge\sum_{k:\sigma_2^{-1}(k)\le\sigma_2^{-1}(j)}p_k\right\}\neq\emptyset;$$

otherwise, job i should be processed on position $\sigma_2^{-1}(j)$ rather than job j since job i has a longer processing time. Then, $j\in H$, and

$$d_j\ge\sum_{k:\sigma_2^{-1}(k)<\sigma_2^{-1}(j)}p_k\ge C_i^+(Q)-p_j.$$

Thus, $(i,j)\in BS(Q)$. □