Int. J. Production Economics 147 (2014) 117–124

Contents lists available at ScienceDirect

Int. J. Production Economics

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

Supply chain scheduling at the manufacturer to minimize inventory holding and delivery costs

Esaignani Selvarajah *, Rui Zhang

Odette School of Business, University of Windsor, 401 Sunset Avenue, Windsor Ontario N9B 3P4, Canada

ARTICLE INFO

Article history:
Received 3 December 2010
Accepted 12 August 2013
Available online 26 August 2013

Keywords:
Batching
Scheduling
Release time
Delivery cost
Manufacturer
Supply chain

ABSTRACT

In a supply chain, a manufacturer receives jobs from suppliers at distinct time points, and produces and delivers final products to customers in batches. The manufacturer can be modeled as a single machine in the supply chain and the problem can thus be modelled as minimizing the sum of weighted flow time and the batch delivery costs. Since the problem with arbitrary processing times, release times and weights is strongly $\mathcal{NP}$−hard, we first analyze some polynomially solvable special problems. Then we develop a heuristic algorithm to solve the general problem. We also develop a lower bound to study the performance of our heuristic algorithm and the computational experiments show that the solutions of the heuristic algorithm are close to optimal solution.

1. Introduction

Supply chain management has been one of the most important topics in manufacturing research. Scheduling models which consider inbound productions and outbound deliveries simultaneously can improve the overall performance of supply chains. The recently emerging research area of supply chain scheduling tries to address this problem. Hall and Potts (2003) study several supply chain scheduling problems by including delivery costs in addition to the scheduling costs in the objective. (Readers may refer to Chen and Hall, 2007; Dawande et al., 2006; Agnetis et al., 2006; Chen and Lee, 2008; Shabtay, 2010; Wang and Cheng, 2000; Hall et al., 2008; Pundoor and Chen, 2009). All these papers, consider the objective of minimizing the sum of weighted flow times (or sum of flow times), where weighted flow time can be interpreted as inventory holding costs. If we consider an individual production system (or organization) as a single machine, and times at which orders (or raw materials) arrive the system as release times, then the problem will be equivalent to batch scheduling on a single machine with release times. A job can be delivered to the customer if its processing is completed. In most cases, there incur delivery costs when delivering completed jobs to customers. Therefore, jobs are delivered in batches to customers to save delivery costs. However, the waiting of jobs in the system for delivery increases the holding cost. Therefore, we need to find a trade-off between increase in inventory holding cost and decrease in delivery cost. Selvarajah and Steiner (2009) study this problem at the supplier with multiple customers, where all jobs are available for processing at time zero. They present a 1.5-approximation algorithm, and perform a parametric analysis on the data to prove that the algorithm yields solutions closer to the optimal solutions for problems where data fall into realistic ranges. Halim and Ohta (1994) study batch scheduling to minimize sum of flow time at manufacturers when job arrivals and final product deliveries are to follow *just-in-time* philosophy.

In this paper, we study batch scheduling at the manufacturer, where jobs are received at distinct time points from suppliers, and there incurs a delivery cost each time a delivery is made to a customer. Thus, our problem is defined as follows: We are given a job set $\mathbb{J}=\{\mathbb{S}_1,\mathbb{S}_2,\ldots,\mathbb{S}_m\}$ for m customers, where $\mathbb{S}_i=\{J_{i,1},J_{i,2},\ldots,J_{i,n_i}\}$ is the job set for customer I_i, $i=1,2,\ldots,m$, and the number of jobs is $n=\sum_{i=1}^{m} n_i$. In most real cases, m is relatively stable for a manufacturer. This justifies our assumption that the number of customers, m is *fixed*. $J_{i,k}$ represents the k-th job of customer I_i. For each job $J_{i,k}$, denoted by $p_{i,k}$, $r_{i,k}$, and $w_{i,k}$, respectively are the *processing time*, *release time* and *weight*. Only the jobs belonging to the same customer can be delivered in the same batch. Associated with each batch is a *delivery cost* q_i, for customer I_i, $i\le m$. Our goal is to find a schedule which minimizes the sum of the holding and the delivery costs. In scheduling literature, the unit holding cost of job $J_{i,k}$ is called the weight $w_{i,k}$ and the time period during which $J_{i,k}$ is in the system is called the flow time $F_{i,k}$. Therefore, the cost of holding $J_{i,k}$ in the system is $(w_{i,k}\times F_{i,k})$. Let $C_{i,k}$ be the completion time of $J_{i,k}$. Then we know that the delivery time $D_{i,k}$ of job $J_{i,k}$ is greater or equal to $C_{i,k}$ and $F_{i,k}=D_{i,k}-r_{i,k}$. Extending the three-field notation (Graham et al., 1979), the problem can be denoted by $1|m,r_{i,k}|\sum w_{i,k}F_{i,k}+\sum b_i q_i$, where b_i is the number of batches for customer I_i in a schedule.

* Corresponding author. Tel.: +1 519 253 3000x3105; fax: +1 519 973 7073.
E-mail addresses: selvare@uwindsor.ca (E. Selvarajah), zhangr6@mcmaster.ca (R. Zhang).

0925-5273/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ijpe.2013.08.015

We also make the following assumptions: (1) once a job gets started, no interruption is allowed during its processing; (2) all the data is nonnegative integers; (3) all data is well-defined before time zero; (4) delivery costs are independent of batch sizes.

Lenstra et al. (1977) prove that sequencing of jobs with arbitrary release times on a single machine to minimize sum of completion time is strongly $\mathcal{NP}-$hard. Further, Labetoulle et al. (1984) prove that sequencing of jobs with arbitrary release times on a single machine to minimize sum of weighted completion time is strongly $\mathcal{NP}-$hard even when preemption is allowed. Therefore, it is clear that our batch scheduling problem is strongly $\mathcal{NP}-$hard even when preemption is allowed. Hall and Potts (2003) study similar problem with the assumption that jobs of the same customer follow a fixed job sequence. They interpret release times as arrival times of orders delivered by suppliers and present a dynamic programming algorithm with time complexity $O(n^{3m})$ for the problem where jobs have identical weights. In fact, their dynamic programming algorithm for given job sequence can easily be modified for the problem when jobs have arbitrary weights. We first study few special cases of the problem that can be solved polynomially. Then for the general problem, we develop heuristic algorithm and develop a lower bound to test the performance of the algorithm.

This paper is organized as follows. Section 2 discusses on some preliminaries of the problem. In Section 3, we study optimal batching when the job sequence for all jobs is fixed. Section 4 study the problem when the jobs of each customer have to be processed on the single machine according to a fixed permutation under two special conditions (i) when all jobs have equal processing times, (ii) when all jobs have unit processing time. Section 5 studies the general problem where jobs have arbitrary processing times, weights, and release times. A heuristic algorithm and a lower bound algorithm are presented, and the computational experiments show that the heuristic algorithm finds close to optimal solution. Finally conclusion and a brief discussion on future research are provided in Section 6.

2. Preliminaries

In traditional batch scheduling (Potts and Kovalyov, 2000), jobs of related families are grouped together and are processed in batches to minimize machine setup cost/time required for those families. In some batching problems, jobs of the same batch may not need to be processed consecutively and we call these problems batching with *batch-preemption*. It is clear that in traditional batching problems, jobs of the same batch are processed consecutively to minimize machine setup time/cost (refer to Fig. 1). Thus, traditional batch scheduling problems do not allow *batch-preemption*.

In our problem, we assume that setup times are sequence independent and thus can be attached to the processing time of each job. Therefore, setup times are included in processing times. However, we batch jobs to save batch delivery costs. Therefore, unlike in the traditional batch scheduling problem where a batch is a group of jobs processed using a single machine setup, in our problem a batch is a group of jobs delivered in single delivery to a customer. Consider a batching solution given in Fig. 2, where jobs

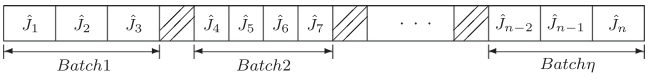

Fig. 1. Traditional batch scheduling with no batch-preemption.

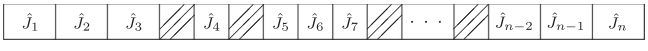

Fig. 2. Supply chain scheduling with $m > 1$ customers and batch-preemption.

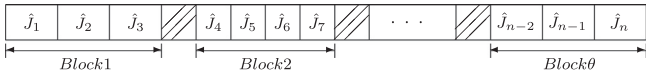

Fig. 3. Blocks and head-jobs in a schedule with θ blocks.

$\hat{J}_4$, $\hat{J}_{n-2}$, $\hat{J}_{n-1}$, and $\hat{J}_n$ belonging to customer I_i are delivered in a single batch, because increase in the holding cost of $\hat{J}_4$ due to its waiting time in the system until the completion time of $\hat{J}_n$ is off-set by the saving in the delivery cost q_i. After processing $\hat{J}_4$, jobs belonging to other batches are processed and delivered because $\hat{J}_n$ is released only at the end of the entire schedule. Thus in our problem, jobs of the same batch may not be processed consecutively, i.e., batches may be preempted.

Remark 1. There exists an optimal batching solution with no *batch-preemption* for the manufacturer's problem (i) when all release times are zero or (ii) when there is only one customer.

Note that when there are more than one customers and arbitrary job release times, then batching solution with no *batch-preemption* may not be optimal as explained before.

Consider a schedule $\sigma = \hat{J}_1, \hat{J}_2, \ldots, \hat{J}_n$ in which job $\hat{J}_k$ is the k-th job processed in σ, and $\hat{J}_k$ can be any job belonging to any customer I_i, $i = 1, 2, \ldots, m$. Then there may be an idle time before starting processing of job $\hat{J}_k$, if job $\hat{J}_{k-1}$ is completed before $\hat{J}_k$ is released. Let us call the set of jobs assigned between two consecutive idle times a *block*, and the first job in a block the *head-job* of that block. For example, in Fig. 3, there are θ blocks in the schedule. Job $\hat{J}_1, \hat{J}_4$ and $\hat{J}_{n-2}$ are the *head-jobs* of Block 1, Block 2 and Block θ, respectively.

In Propositions 1–3 we present simple properties of an optimal schedule.

Proposition 1. *There exists an optimal schedule in which if the machine is idle at time t, then* (1) *any job which is completed after t, must be started processing after t*; (2) *any job which is started processing before t, must be completed before t.*

Proposition 2. *There exists an optimal schedule in which if job $J_{i,k}$ is completed at time t, then it must be delivered in the first delivered batch for customer I_i at or after t.*

Proposition 3. *There exists an optimal batch schedule in which delivery of each batch* (*set of jobs*) *occurs at the completion time of its last processed job.*

In some production environment, jobs are processed in a pre-specified sequence due to some technical constraints. For example, jobs are processed in their arrival order in order to have a simplified material handling. Further, the knowledge of given fixed job sequence can be applied when developing heuristic algorithm for the general problem. Readers may refer to Section 5 where we use the optimal batching of given fixed job sequence when developing our heuristic algorithm. Therefore, we study this special case in the next two sections. Section 3 studies optimal batching when job sequence is fixed for all jobs which we call *fixed-all-job-sequence*. Section 4 studies optimal batching when job sequence of each customer is fixed and we call this *fixed-customer-job-sequence*. Here the fixed sequence refers to the arriving order, i.e., the jobs are sequenced as the non-decreasing order of their release time, either in one sequence (Section 3) or in multiple sequences due to multiple customers (Section 4).