Journal of the Operations Research
Society of Japan
2006, Vol. 49, No. 3, 174-187

BATCH SCHEDULING IN CUSTOMER-CENTRIC SUPPLY CHAINS

Esaignani Selvarajah George Steiner
McMaster University

(Received July 5, 2005; Revised November 17, 2005)

Abstract Supply chain scheduling is a new emerging area of research. We study batch arrival scheduling problems at the manufacturer in a multi-level customer-centric supply chain, where promised job due dates are considered constraints which must be satisfied. We analyze the tradeoff between inventory holding costs and batch delivery costs. We show that the problems are closely related to batch scheduling problems on a single machine with flow-time related objectives. We prove that minimizing the sum of total weighted flow time and delivery costs is strongly NP-hard. For the unweighted version of the problem, we present efficient solution algorithms both for single machine and assembly systems. We also develop a dynamic programming solution for minimizing the sum of maximum flow time and delivery costs.

Keywords: Scheduling, batches, supply chain, customer centric

1. Introduction

Supply chain management has been one of the most important research topics in manufacturing over the last fifteen years. Most of the supply chain literature focuses on inventory control issues on the strategic level, using stochastic models. However, Thomas and Griffin [21] point out in their review of the literature that for many products logistics expenditures can constitute as much as 30% of their cost. This underlines the need for research dealing with supply chain problems on the operational level, using deterministic rather than stochastic models. The recently emerging research area of supply chain scheduling tries to address this problem.

As it is a new area for research, there are relatively few papers dealing specifically with scheduling problems in supply chains. Hall and Potts [11] presented the first paper on this important topic. They model the supply chain using three levels for potential decision makers: the supplier, the manufacturer(s) and the customers. Figure 1 depicts the network for a problem with one supplier (S), one manufacturer (M), and m customers $(C_1, C_2, ..., C_m)$. The manufacturer receives the products (jobs) in batches and batch arrival times are specified by the supplier. The manufacturer processes the jobs and delivers the finished products to customers. Partitioning of jobs into batches and batch delivery times of finished products to customers are decided by the manufacturer to satisfy the demand during the planning horizon. Each stage of the supply chain is viewed by Hall and Potts [11] as a single machine for scheduling purposes and they study a variety of scheduling, batching and delivery problems with the objective of minimizing the overall scheduling and delivery cost. Chen and Hall [4] extended the model to supply chains with assembly-type manufacturing systems. Selvarajah and Steiner [19, 20] studied the supplier's scheduling problem in detail when the objective is to minimize the sum of inventory holding and delivery costs.

A common feature of the above papers is that they assume that whoever makes the

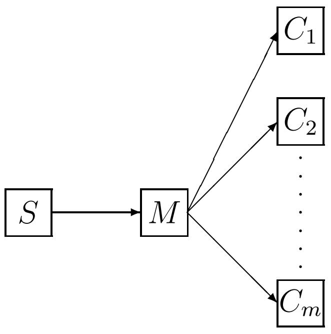

Figure 1: Network showing the supplier, manufacturer, and customer relationship

scheduling decisions, i.e., the supplier or the manufacturer, they incur the associated costs. This means that once the scheduling decisions have been made at one stage, the batches of products are *pushed* through the system to the next stage accordingly. Many modern manufacturing systems and supply chains use a *pull* approach, however, in which the scheduling decisions are made at the later stage and the products are *pulled* from the preceding stage to arrive in the right quantities and *just in time* (*JIT*) when they are needed. JIT manufacturing is a management philosophy that strives to increase value added and eliminate sources of manufacturing waste. The interested reader in JIT is referred to Monden [13] for more detail. Another key feature of JIT systems is that they strive to reduce lead times between the stages. Treville et. al [23] argue that supply chain coordination between partners without lead time reduction may not improve the chain's performance to its best possible. Another key point of successful JIT manufacturing is maintaining low inventory levels while meeting the customer demands on time. This puts increased emphasis on the synchronized movement of inputs and outputs in the production and delivery of goods and services to customers. Frequent deliveries are used to reduce inventories, which of course increases delivery costs. In this paper, we study this tradeoff in supply chains which operate as a pull system, i.e., the *recipient* of the goods or parts (the customer at stage 3 or the manufacturer at stage 2) makes the decisions how frequently and when the batches of products should be delivered to him. We *assume* that it is always the recipient who incurs the costs associated with any delivery schedule he or she decides. This implies that unreasonably high delivery frequencies with very high delivery costs will be avoided. We refer to such systems as *customer-centric* supply chains, where the generic word customer is used for the recipient.

Our study is motivated by the wide adoption of JIT systems in many successful production organizations. One example is BMW, the winner of the productivity award for manufacturing by Modern Materials Handling [2]. BMW's newly expanded manufacturing plant in Spartanburg, S.C., uses a pull system to build customer-specified vehicles within 10 days of order placement. Although the plant builds only two models, X-5 sports utility vehicles and the two-seater Z-4 roadster, there are many options available for each model in terms of shape, colour, and interior requirements. For example, for the X-5 model, there are 8 body variances, 12 colours, 19 engine choices, 16 interior choices, and 85 other options. The plant keeps its suppliers constantly informed of accurate and stable demand data. As a result, the plant is able to follow the JIT philosophy successfully.

In today's competitive environment, the most important objective for supply chains is to meet the customers' demand in a timely fashion. Classical push-type supply chain scheduling tries to meet the due dates by assigning tardiness or lateness penalties to the schedule and trying to minimize these penalties. In a customer-centric supply chain it is a *constraint* that all due dates (deadlines) must be met. In fact these due dates (final delivery schedules) are assumed to be given as input, and they drive (pull) the system.

Several papers in the literature have studied single-machine scheduling problems with deadline constraints. Smith [22] develops a polynomial-time algorithm for the sum of completion time problem with deadline constraints and Heck and Roberts [9] present an algorithm for minimizing the sum of completion times subject to not increasing the maximum tardiness. The weighted sum of completion time problem was shown to be strongly $\mathcal{NP}$-hard by Lenstra et al. [12]. A number of branch and bound algorithms have appeared in the literature for this problem. These include the algorithms of Potts and Van Wassenhove [16], Posner [15] and Werner [25]. The relatively most efficient one is a recent algorithm due to Pan [14].

In the model we study, there are N jobs, $J_1, J_2, \ldots, J_N$, to be processed at the manufacturer whose system may be modeled by either a single machine or an assembly-type operation with subtasks $J_{i,j}$ to be processed on l machines in a series for $i = 1, ..., N$ and $j = 1, ..., l$. Job J_i must be delivered to a customer at time D_i. The cost of these deliveries is borne by the customer. In the single-machine model, J_i requires processing for p_i time for $i = 1, 2, \ldots, N$. In the assembly operation, the processing time of subtask $J_{i,j}$ is denoted by $p_{i,j}$. (If a job skips a certain operation then $p_{i,j} = 0$ for the corresponding subtask.) Since no job is delivered before its deadline, the manufacturer wants to complete them as close to these deadlines as possible. Therefore, we assume that the jobs are processed in earliest due date (EDD) order and this leads to a feasible schedule, i.e., the manufacturer has sufficient capacity to make this schedule feasible for meeting the deadlines. The manufacturer receives parts and supplies for each job or subtask from its supplier(s) in batches and is charged a delivery cost of d for each batch. The manufacturer must receive the batches in time to enable him to meet the final deadlines, but does not want to receive the supplies too early because each job J_i incurs an inventory holding cost in the time interval $[a_i, D_i]$, where a_i is its *arrival time* at the manufacturer for $i = 1, 2, \ldots, N$. The inventory holding cost of a job J_i is closely related to its *flow time* defined as $D_i - a_i$. Since the delivery cost is measured in monetary terms, we multiply flow-time related performance measures by appropriate constants in order to maintain compatibility in measurement. Therefore, we multiply the sum of flow times by a constant h, which is the cost of holding a job in inventory over a unit time; multiply the maximum flow time by a constant K, which is the penalty cost associated with the maximum flow time; and multiply the flow time of job J_i by w_i, which is the holding cost of job J_i over a time unit when the objective is to minimize the sum of the weighted flow times and delivery costs. Without loss of generality we may assume that the job sequence is $J_1, J_2, \ldots, J_N$. Then the manufacturer wants to find the optimal *arrival time* a_j of each job J_j, the *number of batches* n and the partitioning of the jobs into arrival batches which defines the batch sizes so that the total cost is minimized. We assume there always exists a feasible schedule at the manufacturer, i.e.,

$$\max_{k=1,2,\ldots,j}\{a_k + \sum_{i=k}^{j} p_i\} \leq D_j \tag{1.1}$$

We consider the following objectives:

1. For the sum of flow times with batching, total cost $TC_1 = \sum_{j=1}^{N} h(D_j - a_j) + nd$;
2. For the maximum flow time with batching, total cost $TC_2 = K \max_{j=1,\ldots,N}(D_j - a_j) + nd$;

3. For the sum of weighted flow times with batching, total cost $TC_3 = \sum\limits_{j=1}^{N} w_j(D_j - a_j) + nd$.

Batch scheduling problems have been extensively studied before. Potts and van Wassenhove [18], Albers and Brucker [1], Webster and Baker [24], and Potts and Kovalyov [17] gave comprehensive surveys. Even though there have been many papers, only a few deal with batch scheduling problems with delivery costs. Cheng et al. [5] study batch scheduling on a single machine to minimize the sum of delivery costs and earliness penalties. Yang [26] analyzes a similar model with given batch delivery dates and Hall et al. [10] study related problems in different machine environments. Chen [3] presents a dynamic programming algorithm for single-machine scheduling and common due date assignment with earliness-tardiness penalties and batch delivery costs.

The paper proceeds as follows. In the next section, we study problems of batch arrival scheduling to minimize the total weighted flow time and delivery costs, i.e., cost function TC_3. First we prove that the problem is strongly NP-hard on a single machine even with a common due date for all the jobs. Following this, we present a linear-time dynamic programming algorithm for the problem on a *fixed* job arrival sequence. This algorithm is used repeatedly in Section 3 for minimizing TC_1 both for single-machine and assembly-shop environments. In Section 4, we present an efficient dynamic programming algorithm for batch arrival scheduling with objective TC_2. The last section contains our concluding remarks.

2. Batch Arrival Scheduling to Minimize the Total Weighted Flow Time and Delivery Costs

2.1. Complexity

Let us consider the batch arrival scheduling problem at the manufacturer when its system is modeled by a single machine and the objective is to minimize TC_3.

Theorem 2.1 *Minimizing* $TC_3 = \sum\limits_{j=1}^{N} w_j(D_j - a_j) + nd$ *is strongly NP-hard.*

Proof: Hall and Potts [11] have proved that minimizing the sum of total weighted flow times and delivery costs for a supplier in a push-type system is strongly NP-hard. They used the well-known strongly NP-hard 3-PARTITION problem to reduce it to their scheduling problem. We show how this reduction can be adapted to prove the strong NP-hardness of our problem.

3-PARTITION [8]:

Given $3r$ integers $u_1, ..., u_{3r}$, where $\sum_{i=1}^{3r} u_i = rz$ and $z/4 < u_i < z/2$, for $i = 1, ..., 3r$, does there exist a partition $A_1, ..., A_r$ of the index set $\{1, ..., 3r\}$, such that $|A_j| = 3$ and $\sum_{i \in A_j} u_i = z$, for $j = 1, ..., r$?

Consider the following instance of our scheduling problem: $N = 3r$, job J_i has $p_i = w_i = u_i$ and $D_i = rz$, for $i = 1, ..., 3r$, $d = z^2/2$ and let $C = r(r+2)z^2/2$ be a threshold value. We prove that there exists a batch arrival schedule for this instance with $TC_3 = \sum\limits_{j=1}^{N} w_j(D_j - a_j) + nd \le C$ if and only if there exists a solution for 3-PARTITION.

Suppose 3-PARTITION has a solution and assume, without loss of generality, that the integers are numbered so that $u_{3i-2} + u_{3i-1} + u_{3i} = z$, for $i = 1, ..., r$. Consider the schedule in which the jobs are scheduled in this sequence $J_1, ..., J_N$ and supply batch B_i for jobs $\{J_{3i-2}, J_{3i-1}, J_{3i}\}$ arrives at the time J_{3i-2} starts its processing, i.e., $a_{3i-2} = a_{3i-1} = a_{3i} =$

$(i-1)z$, for $i = 1, ..., r$. It is easy to see that each of the jobs $\{J_{3i-2}, J_{3i-1}, J_{3i}\}$ has flow time equal to $rz-(i-1)z = (r-i+1)z$. Therefore, $TC_3 = \sum_{i=1}^{r}(u_{3i-2}+u_{3i-1}+u_{3i})(r-i+1)z+rd = \sum_{i=1}^{r}(r-i+1)z^2 + rz^2/2 = C$ for this schedule.

Next we prove the theorem in the other direction. Suppose we have a schedule with n arrival batches and let x_i be the total processing time of the jobs corresponding to the ith batch B_i for $i = 1, ..., n$. It is clear that supplies for B_i must arrive at the time the last job in B_{i-1} completes its processing, i.e., at $\sum_{j=1}^{i-1} x_j$. Therefore, the flow time of the jobs in B_i will be $rz - \sum_{j=1}^{i-1} x_j = \sum_{j=i}^{n} x_j$ for $i = 1, ..., n$. Thus we have $TC_3 = \sum_{i=1}^{n} x_i \sum_{j=i}^{n} x_j + nd = \sum_{i=1}^{n} x_i \sum_{j=i}^{n} x_j + nz^2/2 = \left(\sum_{j=1}^{n} x_j\right)^2/2 + \sum_{j=1}^{n} x_j^2/2 + nz^2/2$. Thus minimizing TC_3 on n batches can be written as

$$\text{minimize} \quad \left(\sum_{j=1}^{n} x_j\right)^2/2 + \sum_{j=1}^{n} x_j^2/2 + nz^2/2 \tag{2.1}$$

$$\text{subject to} \quad \sum_{j=1}^{n} x_j = rz. \tag{2.2}$$

Since the first term of this objective is $(rz)^2$, it is easy to see that the whole function will be minimized when $x_1 = x_2 = ... = x_n = rz/n$. Thus for any n-batch solution we must have $TC_3 \geq (rz)^2/2 + n(rz/n)^2/2 + nz^2/2$. Simple arguments from calculus show that this expression reaches its minimum at $n = r$ and the minimum value is C. Thus if there exists a batching schedule with $TC_3 = C$, then we must have $n = r$ and each batch must have a size $x_j = z$. This implies that each batch has 3 jobs in it and 3-PARTITION has a solution.
$\square$

2.2. Batching a given job sequence on a single machine

In this section, we study the optimal batching problem at the manufacturer to minimize $TC_3 = \sum_{j=1}^{N} w_j(D_j - a_j) + nd$ *when* the order of job processing and arrival at the manufacturer is given and this is also the order of job arrivals. Without loss of generality, let this sequence be $J_1, ...J_N$. Note that the given job processing sequence is assumed to be feasible for meeting the promised delivery times. Let S_i denote the *latest start time* for job J_i such that the schedule is feasible. Consider an n-batch arrival schedule, and let i_j be the index of the first job of arrival batch B_j, i.e., the batch schedule is $\{i_1, i_1 + 1, \ldots, i_2 - 1\}, \{i_2, i_2 + 1, \ldots, i_3 - 1\}, \ldots, \{i_n, i_n + 1, \ldots, N\}$. Then it is easy to see that batch B_j should arrive at time S_{i_j} and not earlier. Thus

$$TC_3 = \sum_{k=1}^{n} \sum_{j=i_k}^{i_{k+1}-1} w_j(D_j - S_{i_k}) + nd. \tag{2.3}$$

Note that we assume that a feasible schedule exists for this problem thus the Equation 1.1 is satisfied.

If we define $S_{N+1} = D_N$, then we can write $S_{i_k} = D_N - \sum_{j=i_k}^{N} (S_{j+1} - S_j)$. Therefore,

$$\begin{aligned} TC_3 &= \sum_{j=1}^{N} w_j D_j - \sum_{k=1}^{n} \sum_{j=i_k}^{i_{k+1}-1} w_j \Big(D_N - \sum_{j=i_k}^{N} (S_{j+1} - S_j)\Big) + nd \\ &= \sum_{j=1}^{N} w_j(D_j - D_N) + \sum_{k=1}^{n} \sum_{j=i_k}^{i_{k+1}-1} w_j \sum_{j=i_k}^{N} (S_{j+1} - S_j) + nd. \end{aligned} \tag{2.4}$$

Let $S_{j+1} - S_j = p'_j$ for $j = 1, 2, \dots, N$. Note that $p'_j \geq p_j$ and p'_j can be interpreted as the length of time on the machine 'allocated' to job j. (We have $p'_j > p_j$ if there is an idle time in the schedule.) Then

$$TC_3 = \sum_{j=1}^{N} w_j(D_j - D_N) + \sum_{k=1}^{n} \sum_{j=i_k}^{i_{k+1}-1} w_j \sum_{j=i_k}^{N} p'_j + nd. \tag{2.5}$$

By extending the results of Coffman et al. [7], Albers and Brucker [1] presented an $O(N)$ time shortest path algorithm for finding the optimal batching to minimize the sum of weighted completion times of a fixed job sequence on a single machine. We show that minimizing TC_3 can also be formulated as a special shortest path problem by exchanging processing times for weights and weights for allocated times p'_j in the Albers-Brucker formulation:

$$\begin{aligned} TC_3 &= \sum_{j=1}^{N} w_j(D_j - D_N) + \sum_{k=1}^{n} \Big(\sum_{j=i_k}^{i_{k+1}-1} w_j \sum_{j=i_k}^{N} p'_j \Big) + nd \\ &= \sum_{j=1}^{N} w_j(D_j - D_N) + \sum_{k=1}^{n} \Big(\sum_{j=i_k}^{N} p'_j \sum_{j=i_k}^{i_{k+1}-1} w_j + d \Big) \end{aligned} \tag{2.6}$$

Since the first sum is a constant, it is sufficent to minimize the second sum only. If we set

$$c_{ij} = \Big(\Big(\sum_{v=i}^{N} p'_v \Big) \Big(\sum_{v=i}^{j-1} w_v \Big) + d \Big) \text{ for } 1 \leq i < j \leq N, \tag{2.7}$$

then $TC_3 = \sum_{j=1}^{N} w_j(D_j - D_N) + \sum_{j=1}^{n} c_{i_j, i_{j+1}-1}$. Thus c_{ij} represents the contribution to the cost by a batch containing jobs $J_i, J_{i+1}, ..., J_{j-1}$. Furthermore, for any $k > j \geq i$,

$$c_{ik} - c_{ij} = \Big(\sum_{v=i}^{N} p'_v \Big) \Big(\sum_{v=j}^{k-1} w_v \Big) = f(i)h(j,k), \text{ where } f(i) = \sum_{v=i}^{N} p'_v \text{ and } h(j,k) = \sum_{v=j}^{k-1} w_v.$$

Since $f(i)$ is monotone nonincreasing and $h(j,k) > 0$ for all $j < k$, our problem is also equivalent to the special shortest path problem discussed by Albers and Brucker. The following is the corresponding network:

a. Each job J_i $(i = 1, 2, ..., N)$ is represented by vertex i in the network.
b. The network contains directed edges (i, j) for all pairs $i < j$.
c. Edge (i, j) is assigned the edge length c_{ij}.
d. A dummy job J_{N+1} with $p_{N+1} = w_{N+1} = 0$ is added.

It is easy to see then that minimizing TC_3 is equivalent to finding the shortest path from vertex 1 to vertex $N + 1$ in the network.

Let F_j be the length of the shortest path from vertex j to vertex $N + 1$, and $F_j(k)$ the length of a shortest path from j to $N + 1$ which contains (j, k) as first edge. Then,

$F_j(k) = c_{jk} + F_k + d$, and $F_j = min\{F_j(k) | j < k \leq N + 1\}$.

Furthermore, $F_j(k) \leq F_j(l)$ for vertices $j < k < l$ is equivalent to $F_j(k) - F_j(l) = c_{jk} + F_k - c_{jl} - F_l \leq 0$. Since $c_{jl} - c_{jk} = f(j)h(k,l)$, we obtain that

$$F_j(k) \leq F_j(l) \text{ is equivalent to } f(j) \geq \frac{F_k - F_l}{h(k,l)}.$$

tail	$\cdots$	*next*(i)	i	*previous*(i)	$\cdots$	*head*

Figure 2: Structure of the queue q.

Thus for any two vertices $k < l$, if the threshold $\delta(k,l) = \frac{F_k - F_l}{h(k,l)} \leq f(j)$, then $F_j(k) \leq F_j(l)$ and l is called *not better than* k with respect to j. On the other hand, if $f(j) < \delta(k,l)$, then $F_j(k) > F_j(l)$ and l is called *better than* k with respect to j. Based on these monotonicity properties, we present Algorithm 1, which is a slightly modified version of the algorithm by Albers and Brucker adapted to our problem. It uses the queue data structure shown in Figure 2.

Algorithm 1 *Modified Algorithm of Albers and Brucker*

```
          begin
Step 1    $q = N + 1$; $F_{N+1} = 0$
Step 2:   for $j = N$ to 1 do begin
Step 3:       while $head(q) \neq tail(q)$ and $f(j) \geq \delta(next(head(q)), head(q))$ do
                  Delete $head(q)$ from $q$
Step 4:       $N(j) = head(q)$; $F_j = c_{jN(j)} + F_{N(j)}$
Step 5:       while $head(q) \neq tail(q)$ and $\delta(j, tail(q)) \leq \delta(tail(q), previous(tail(q)))$ do
                  Delete $tail(q)$ from $q$
Step 6:       Add $j$ to the tail of $q$
          end
          end
```

The correctness of the algorithm and its $O(N)$ time complexity can be proved the same way as in [1].

Theorem 2.2 *Algorithm 1 computes in $O(N)$ time the optimal batching which minimizes $\sum_{j=1}^{N} w_j(D_j - a_j) + nd$ on a* given *job sequence.*

Note that Algorithm 1 is also applicable to minimize the objective function TC_1.

3. Minimizing the Total Flow Time and Delivery Costs

In this section, we study simultaneous sequencing and batching of jobs for arrival at the manufacturer to minimize $TC_1 = (\sum_{j=1}^{N} h(D_j - a_j) + nd)$.

3.1. Scheduling batch arrivals on a single machine

Lemma 3.1 *There is an optimal schedule for all three objectives, TC_1, TC_2, TC_3, in which the order of job arrivals is the same as the order of job processing.*

Proof: A job cannot start processing until the job immediately preceding it in the EDD processing sequence is not completed, and the arrival of any job before the arrival of a job preceding it in the processing sequence can only make the total flow time larger. Therefore, no job should arrive in the optimal schedule before any of its predecessors in the EDD order. $\square$

Without loss of generality, we index jobs in the order they are in this common sequence of arrival and processing. Then the latest possible start time of job J_i, S_i, can be recursively calculated by $S_N = D_N - p_N$, and $S_i = \min\{D_i, S_{i+1}\} - p_i$ for $i = N-1, N-2, ..., 2, 1$.

Although the job arrival and job processing sequence is the same, the optimal schedule may contain jobs which arrive early and wait in the shop. Consider the following example for minimizing TC_1 with $h = 1$:

j	1	2	3	4	5	6
p_j	9	7	5	12	6	5
D_j	23	23	23	48	48	48

Let us say that we are given the job processing sequence {1,2,3,4,5,6} and we have to find the optimal 2-batching schedule. For this problem, the optimal batching is $B_1 = \{1, 2, 3, 4\}$ and $B_2 = \{5, 6\}$ with total flow time of 131. In this batching solution, the second batch arrives at $t = S_5 = 37$ and the processing of job set $\{5, 6\}$ is completed at $t = 48$; the first batch arrives at $t = S_1 = 2$ and the processing of job set $\{1, 2, 3\}$ is completed at $t = 23$, and the processing of job set $\{4\}$ is completed at $t = 35$. If we move job J_4 to the second batch, then B_2 must arrive at $t = 25$ so that the job set $\{4, 5, 6\}$ will be completed by $t = 48$; and the job set $\{1, 2, 3\}$ must arrive at $t = 2$ and will be completed at time $t = 23$. The total flow time of this new schedule will increase to 132. This shows that in the optimal schedule some jobs may arrive with early batches and wait in the shop.

Lemma 3.2 *There exists an optimal schedule, for all three objectives TC_1, TC_2, and TC_3, in which a batch arrives only when all the previously available jobs at the manufacturer have completed processing, i.e., a batch arrives only when the machine is available to start its processing.*

Proof: Since the job processing sequence is given, a job cannot be started before its immediately preceding job is completed. From Lemma 3.1, the order of job arrivals follows the processing sequence. Therefore, jobs assigned to any batch will be processed after the last job of the previous batch has completed processing. Thus arrival of a batch before the completion of the processing of the last job of the previous arrival batch can only increase the flow time of the schedule. □

Lemma 3.3 *Let job J_j be the first job to be processed in arrival batch B_k, then batch B_k should arrive at time S_j, for all three objectives TC_1, TC_2 and TC_3.*

Proof: Let us assume that there is an optimal schedule which does not satisfy the lemma, but it is consistent with Lemmas 3.1 and 3.2. Select the last batch which arrives before the latest start time of the first job of the batch in this schedule. If we delay the batch arrival time to the latest start time of the first job of this batch, the schedule will remain feasible. Furthermore, the flow times of all the jobs belonging to this batch will be reduced, flow times of all the jobs belonging to other batches will remain the same. This contradicts the optimality assumption for the schedule. □

Lemma 3.4 *There exists an optimal schedule, for all three objectives, TC_1, TC_2, and TC_3, in which the jobs to be delivered to customer(s) at the same due date are scheduled in LPT (Largest Processing Time first) order at the manufacturer.*

Proof: We know that jobs to be delivered at the same delivery time to customer(s) are processed consecutively at the manufacturer because of the EDD processing order. Without loss of generality, let this sequence be $J_1, ..., J_N$. Let us assume that the lemma is not true for an optimal schedule. Then there will be at least two jobs J_i and J_{i+1} at the manufacturer such that $p_i < p_{i+1}$ and $D_i = D_{i+1}$. If jobs J_i and J_{i+1} belong to the same arrival batch, then interchanging them will not affect the flow time of any job. If jobs J_i and J_{i+1} belong to different arrival batches, say, to B_k and B_{k+1}, respectively, then let the arrival time of batch B_k be t_k and the arrival time of B_{k+1} be t_{k+1}. From Lemma 3.3, we know $t_{k+1} = min\{D_{i+1}, S_{i+2}\} - p_{i+1}$. Interchange jobs J_i and J_{i+1} in these batches, which does not change the arrival batch sizes. Call the new batches $B_k^{'}$ and $B_{k+1}^{'}$. Thus in the new schedule, $B_k^{'}$ arrives at t_k and $B_{k+1}^{'}$ arrives at $t_{k+1}^{'} = min\{D_i, S_{i+2}\} - p_i = min\{D_{i+1}, S_{i+2}\} - p_i = t_{k+1} + p_{i+1} - p_i$. Note that the interchange will not affect the feasibility of the schedule. The interchange will not affect the flow times of the jobs in $B_k \backslash J_i$ in the original schedule. The flow time of every job in $B_{k+1} \backslash J_{i+1}$ is decreased by $p_{i+1} - p_i > 0$ compared to the original schedule. The flow time of J_{i+1} is increased by $t_{k+1} - t_k$ and the flow time of J_i is decreased by $t_{k+1}^{'} - t_k > t_{k+1} - t_k$. Thus the net change in the total flow time is a decrease by at least $p_{i+1} - p_i$, which contradicts the optimality of the original schedule. Therefore, any jobs J_i and J_{i+1} not in LPT order must belong to the same batch. Repeatedly resequencing the jobs with the same due date into LPT order within the batches does not change the cost or the feasibility of the schedule and leads to an optimal schedule satisfying the conditions of the lemma. □

The combination of EDD ordering with LPT ordering of jobs with the same due date within batches fixes the optimal sequence for the jobs. Since we know the job sequence, Algorithm 1 can be used to find the optimal batch sizes. Algorithm 2 summarizes the steps needed to find the optimal batch arrival schedule.

Algorithm 2: *Algorithm to minimize the sum of flow times and delivery costs*

```
Step 1:  Order the jobs in EDD order and schedule the jobs with the same due date in
         LPT order.
Step 2:  Call Algorithm 1 to find the optimal arrival batch sizes of the found sequence.
```

Theorem 3.1 *Algorithm 2 finds in $O(N \log N)$ time an optimal batch arrival schedule that minimizes TC_1, the sum of flow times and delivery costs.*

Proof: Step 1 finds the optimal job sequence by sorting, which requires $O(N \log N)$ time. Algorithm 1 finds the optimal batching of this job processing sequence in $O(N)$ time. □

3.2. Batch arrival scheduling for an assembly shop

In this section, we study the optimal batch arrival policy in an assembly shop where jobs are processed and assembled on a series of l machines. At each machine, a job may require parts which are delivered from one of q suppliers. The schematic of the supply chain for this problem is shown in Figure 3. The manufacturer has to deliver the right products in the right quantities at the promised times to customers and delivery costs are charged to the customers. In order to meet the promised delivery times, the manufacturer has to order the parts from the suppliers, and process and assemble them on the l machines. For any product J_i $(i = 1, 2, \dots, N)$ which does not need processing on the jth machine $(j = 1, 2, \dots, l)$, we set $p_{i,j}$ to zero. Suppliers have to deliver parts to the manufacturer at the manufacturer's required times and the costs for deliveries from part suppliers to the manufacturer are charged to the manufacturer. Thus the manufacturer wants to find the optimal batch arrival schedules for the parts from each supplier so that the total of sum of flow times and delivery costs is minimized while meeting promised delivery times to customers.

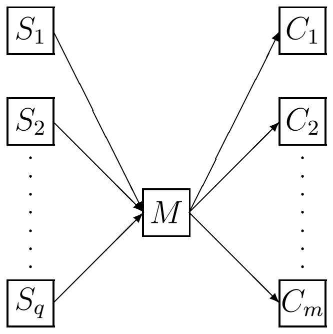

Figure 3: Network showing the manufacturer's relationship with q suppliers, and m customers

Let $S_{i,j}$ denote the *latest possible start time* of task $J_{i,j}$ in a feasible schedule. Then the fact that the jobs are 'pulling' their subtasks through the system can be captured by the following backward recursive calculations:

$$\left.\begin{array}{r}S_{N,l} = D_N - p_{N,l} \\ S_{i,l} = min\{D_i, S_{i+1,l}\} - p_{i,l},\ \text{for } i = 1, 2, ..., N-1 \\ S_{i,j} = min\{S_{i,j+1}, S_{i+1,j}\} - p_{i,j},\ \text{for } i = 1, 2, ..., N\text{: } j = 1, 2, ..., l-1\end{array}\right\} \quad (3.1)$$

Lemma 3.5 *There exists an optimal batch arrival schedule and associated production schedule in which task $J_{i,j}$ (for $i = 1, 2, ..., N$; $j = 1, 2, ..., l$) starts its processing at time $S_{i,j}$.*

Proof: By using $S_{i+1,l}$ as an upper bound on the completion time of task $J_{i,l}$, the first two rows of (3.1) ensure that sufficient time will be available at the last machine to finish the processing of $J_{i,l}$ by D_i. The calculations in the last row of (3.1) make sure that there is sufficient time also for task $J_{i,j}$ at machine j for $i = 1, 2, ..., N$; $j = 1, 2, ..., l$. (The schedule is feasible if $S_{i,1} \geq 0$ for $i = 1, 2, ..., N$.) It is also clear that the *arrival time of the parts* for $J_{i,j}, a_{i,j}$, must satisfy $a_{i,j} \leq S_{i,j}$ for the schedule to be feasible, but some parts may arrive early. Now suppose we have an optimal schedule in which there are some tasks starting before their latest start time. Consider the last such task, say $J_{r,k}$, and shift its processing to start at $S_{r,k}$. The shift will neither affect the feasibility of the schedule nor the flow time of any task. Repeatedly applying the above argument to all remaining early tasks will lead to a schedule which satisfies the lemma. □

Lemma 3.6 *The batch arrival scheduling problems from each supplier are separable (i.e., arrival schedule from a supplier is independent of arrival schedules from other suppliers) and can be solved independent of each other.*

Proof: We know from Lemma 3.5 that there exists an optimal production schedule in which each task $J_{i,j}$ starts at its latest possible start time $S_{i,j}$. Then $S_{i,j}$ can be viewed as the deadline for the arrival of the parts needed from their supplier. Since each task $J_{i,j}$ receives its part(s) from at most one supplier by assumption, each $S_{i,j}$ can become a delivery deadline only for one supplier. Thus whatever batch arrival times are scheduled from a supplier, this does not affect the flow time of other parts (tasks) from other suppliers. So by considering the delivery requirements from one supplier, we get a separable batch arrival scheduling problem for this supplier. Therefore, the problems can be solved saparate from each other for each supplier. □

Theorem 3.2 *The batch arrival scheduling problem at an assembly manufacturer can be optimally solved in $O(qlN\log(lN))$ time.*

Proof: By Lemma 3.6, parts arriving from each supplier can be scheduled for arrival in a separate batch scheduling problem. We have (at most) q of these problems. Each of them can be solved by Algorithm 2 in $O(lN\log(lN))$ time, thus the overall time required is at most $O(qlN\log(lN))$ indeed. $\square$

4. Minimizing Maximum Flow Time and Delivery Costs

In this section, we consider the batch arrival scheduling problem with objective $TC_2 = K\max_{j=1,2,\ldots,N}\{D_j - a_j\} + nd$ at the manufacturer. It is easy to see that Lemmas 3.1-3.3 apply to this problem too and they can be proved the same way.

Lemma 4.1 *There exists an optimal schedule in which the jobs to be delivered to customer(s) at the same due date are scheduled in LPT order at the manufacturer.*

Proof: Let there be an optimal schedule in which jobs with the same due date do not follow the LPT order. Without loss of generality, let the job sequence be $J_1, J_2, \ldots, J_N$. Then there will be at least two jobs J_i and J_{i+1} with $p_i < p_{i+1}$ and $D_i = D_{i+1}$. If J_i and J_{i+1} belong to the same batch, then interchanging these two jobs will not affect the maximum flow time. If jobs J_i and J_{i+1} belong to different arrival batches, say, to B_k and B_{k+1}, respectively, then let the arrival time of batch B_k be t_k and the arrival time of B_{k+1} be t_{k+1}. From Lemma 3.3, we know $t_{k+1} = min\{D_{i+1}, S_{i+2}\} - p_{i+1}$. Interchange jobs J_i and J_{i+1} in these batches without changing the arrival batch sizes. Call the new batches $B_k^{'}$ and $B_{k+1}^{'}$. Thus in the new schedule, $B_k^{'}$ arrives at t_k and $B_{k+1}^{'}$ arrives at $t_{k+1}^{'} = min\{D_i, S_{i+2}\} - p_i = min\{D_{i+1}, S_{i+2}\} - p_i = t_{k+1} + p_{i+1} - p_i$. Note that the interchange will not affect the feasibility of the schedule. Furthermore, the interchange will not affect the flow times of the jobs in $B_k \backslash J_i$ in the original schedule. The flow time of every job in $B_{k+1} \backslash\ J_{i+1}$ is decreased by $p_{i+1} - p_i > 0$ compared to the original schedule and the flow time of J_i clearly decreases. The only flow time that is increased is that of J_{i+1}, which goes up by $t_{k+1} - t_k < D_{i+1} - t_k$. We have, however, $D_{i+1} - t_k = D_i - t_k$, and the latter is the flow time of J_i in the original schedule. Therefore, the maximum flow time of the new schedule will not be greater than that of the original one. Repeating this interchange for every violation of the job order in the lemma will yield an optimal schedule satisfying its conditions. $\square$

Note that the lemma implies that there is a job sequence which is optimal for both the maximum flow time plus delivery cost and sum of flow time plus delivery cost objectives. To find the optimal arrival batching for $TC_2 = K\max_{j=1,2,\ldots,N}\{D_j - a_j\} + nd$, however, we cannot use Algorithm 1, which was designed for the sum of flow times objective. Therefore, we present a new dynamic programming algorithm below.

Algorithm 3 *Algorithm to minimize $TC_2 = K\max_{j=1,2,\ldots,N}\{D_j - a_j\} + nd$ on job sequence $J_1, J_2, \ldots, J_N$*

Let $f(k,j)$ be the minimum value of TC_2 on the first j jobs in a schedule using k arrival batches for $1 \le k \le j \le N$. For easier notation, we also define $f(k,j) = \infty$ for $1 \le j < k \le N$. The optimal value of TC_2 can be obtained by $\min_{k=1,\ldots,N} f(k,N)$. The recursive computation of $f(k,j)$ for $1 \le k \le j \le N$ is defined as follows.

$$f(k,j) = \min \begin{cases} K \min\limits_{r=k-1,...,j-1} \left\{ \max\{[f(k-1,r)-(k-1)d]/K, \max\limits_{u=r+1,...,j} \{D_u - S_{r+1}\}\} \right\} + kd, \\ K \max\left\{[f(k,j-1)-kd]/K, D_j - S_{i(k,j-1)}\right\} + kd, \end{cases} \quad (4.1)$$

where $S_{i(k,j-1)}$ is the starting time of the first job, $i(k,j-1)$, of the last batch in the schedule realizing $f(k,j-1)$. The first row of the recursion corresponds to taking the optimal schedule realizing $f(k-1,r)$ and adding to it a new arrival batch containing jobs $\{r+1,...,j\}$ for $r = k-1,...,j-1$. Here $[f(k-1,r)-(k-1)d]/K$ expresses the maximum flow time of the schedule realizing $f(k-1,r)$. The second row of the recursion corresponds to the case when job J_j is simply added to the last batch of the schedule realizing $f(k,j-1)$ without starting a new batch. To facilitate the computations, we need to store the index of the first job of the last batch in the schedule realizing $f(k,j)$, denoted by $i(k,j)$.

Initial conditions: $f(0,0)=0$ and $f(k,j)=\infty$ for $j,k=1,...,N$.

Theorem 4.1 *Algorithm 3 finds an optimal batch arrival schedule at the manufacturer to minimize the total cost of maximum flow time and deliveries in $O(N^3)$ time.*

Proof. The algorithm needs to compute $O(N^2)$ $f(k,j)$ values. Each computation needs $O(N)$ time. By storing the indices $i(k,j)$, we can obtain the optimal batching at the end by backtracking. □

5. Summary and Concluding Remarks

We have studied batch arrival scheduling problems at the manufacturer in a customer-centric supply chain where promised job due dates are considered constraints which must be satisfied. We have shown that the problems are closely related to batch scheduling problems on a single machine with flow-time related objectives. We proved that minimizing the sum of total weighted flow time and delivery costs is strongly NP-hard. For the unweighted version of the problem, we presented efficient solution algorithms both for single machine and assembly systems. We also developed a dynamic programming solution for minimizing the sum of maximum flow time and delivery costs.

Future research in this area may look at alternative objective functions or look for efficient heuristic or approximating solutions for the computationally difficult weighted case.

Acknowledgement

This research was supported in part by the Natural Sciences and Engineering Research Council of Canada under Discovery Grant 1798-03.

References

[1] S. Albers and P. Brucker: The complexity of one-machine batching problems. *Discrete Applied Mathematics*, **47** (1993), 87-107.

[2] BMW: Custom cars on demand - The automaker uses a pull system to build customer-specified vehicles within 10 days of order placement. *Modern Materials Handling*, (2004).

[3] Z.L. Chen: Scheduling with batch setup times and earliness-tardiness penalties. *European Journal of Operational Research* , **96** (1997), 518-537.

[4] Z.L. Chen and N.G. Hall: Supply chain scheduling - Assembly systems. *Working Paper*, (2005).

[5] T.C.E. Cheng, V.S. Gordon and M.Y. Kovalyov: Single machine scheduling with batch deliveries. *European Journal of Operational Research*, **94** (1996), 277-283.

[6] S. Chopra, and P. Meindl: *Supply Chain Management: Strategy, Planning, and Operation* (Prentice Hall, Toronto, 2004).

[7] E.G. Coffman, M. Yannakakis, M.J. Magazine, and C. Santos: Batch sizing and job sequencing on a single machine. *Annals of Operations Research*, **26** (1990), 135-147.

[8] M.R. Garey and D.S. Johnson: *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W.H. Freeman, San Francisco, 1979).

[9] H. Heck and S. Roberts: A note on the extension of a result on scheduling with secondary criteria. *Naval Research Logistics Quarterly*, **19** (1972), 403-405.

[10] N.G.Hall, M.A. Lesaoana and C.N.Potts: Scheduling with fixed delivery dates. *Operations Research*, **49** (2001), 134-144.

[11] N.G. Hall and C.N. Potts: Supply chain scheduling: Batching and delivery. *Operations Research*, **51** (2003), 566-584.

[12] Lenstra, J. K. A. H.G. Rinnooy Kan and P. Brucker: Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, **1** (1977), 342-362.

[13] Y. Monden: *Toyota production system.* (Second Edition, 1993).

[14] Y. Pan: An efficient exact algorithm for single machine scheduling with due dates to minimize total weighted completion time. *Operations Research Letters*, **31** (2003), 492-496.

[15] M.E. Posner: Minimizing weighted completion times with deadlines. *Operations Research*, **33** (1985), 562-574.

[16] C.N. Potts, and L. N. Van Wassenhove: An algorithm for single machine sequencing with deadlines to minimize total weighted completion time. *European Journal of Operational Research*, **12** (1983), 379-387.

[17] C.N. Potts and M.Y. Kovalyov: Scheduling with batching - A review. *European Journal of Operational Research*, **120** (2000), 228-249.

[18] C.N Potts and L.N.Van Wassenhove: Integrating scheduling with batching and lot-sizing - a review of algorithms and complexity. *Journal of Operational Research Society*, **43** (1992), 395-406.

[19] E. Selvarajah and G. Steiner: Batch scheduling in a two-level supply chain - A focus on the supplier. *European Journal of Operational Research*, to appear (2004).

[20] E. Selvarajah and G. Steiner: Batch Scheduling at the Upstream Supplier to Minimize Delivery and Inventory Holding Costs, submitted, (2005).

[21] D.J. Thomas and P.M. Griffin: Coordinated supply chain management. *European Journal of Operational Research*, **94** (1996), 1-15.

[22] W.E. Smith: Various optimizers for single-stage production. *Naval Research Logistics*, (1956) **Quarterly 3**, 59-66.

[23] S. Treville, R. D. Shapiro and A. Hameri: From supply chain to demand chain - the role of lead time reduction in improving demand chain performance. *Journal of Operations Management*, **21** (2004), 613-627.

[24] S. Webster, and K.R. Baker: Scheduling groups of jobs on a single machine. *Operations Research*, **43** (1995), 692-703.

[25] F. Werner: A branch and bound algorithm for minimizing weighted completion times with deadlines. *Optimization*, **28** (1993), 187-199.

[26] X. Yang: Scheduling with generalized batch delivery dates and earliness penalties. *IIE Transactions*, **32** (2000), 735-741.

George Steiner
Management Science and Information Systems
Michael G. DeGroote School of Business
McMaster University
1280 Main Street West
Hamilton Ontario
L8S 4M4 Canada
E-mail: steiner@mcmaster.ca