Scheduling with families of jobs and delivery coordination under job availability

Shisheng Li, Jinjiang Yuan *

Department of Mathematics, Zhengzhou University, Zhengzhou, Henan 450052, People's Republic of China

A R T I C L E   I N F O

Article history:
Received 26 February 2009
Received in revised form 10 June 2009
Accepted 13 June 2009
Communicated by D.-Z. Du

Keywords:
Scheduling
Families of jobs
Batching
Delivery

A B S T R A C T

We consider in this paper the scheduling of families of jobs in which both processing and delivery are coordinated together. Only one vehicle is available to deliver the jobs to specified customers. The jobs can be processed together to form processing batches on the machine and setups of batches are required when the machine is changing from one family to another. Jobs from different families cannot be transported together by the vehicle. The objective is to minimize the time when the vehicle finishes delivering the last delivery batch to its customer and returns to the machine. We propose an \( O(n \log n) \)-time optimal algorithm for the scheduling problem under the group technology assumption. For the scheduling problem without the group technology assumption, we show that the problem is NP-hard and give an \( O(f^2 n^f) \)-time dynamic programming algorithm, where \( n \) is the number of jobs, and \( f \) is the number of families; we also provide a heuristic algorithm with a performance ratio of \( 3/2 \).

1. Introduction

We are given a set of \( n \) nonpreemptive jobs \( J_1, J_2, \ldots, J_n \) that are classified into \( f \) families \( F_1, F_2, \ldots, F_f \). All jobs are available at time 0. The jobs are processed on a single machine. Specially, a set of jobs in the same family can be processed together to form processing batches on the machine. Setups of batches are required when the machine is changing from a processing batch of one family to a processing batch of another family. There is a capacitated vehicle to transport the processing completed jobs to specified customers. We assume that jobs of the same family have identical amount of physical space in a transportation vehicle and belong to a specified customer. We further assume that, jobs from distinct families cannot be transported together by the vehicle in a delivery batch. The goal is to find a schedule to minimize the time when the vehicle finishes delivering the last delivery batch to its customer and returns to the machine.

There are two variants of scheduling the families of jobs depending on when the jobs become available for delivery. Under the batch availability, a job becomes available only when the batch to which it belongs has been processed. An alternative assumption is the job availability (usually known in the literature as item availability), in which a job becomes available immediately after its processing is completed. In this paper, we adopt the assumption of job availability.

We follow the three-field \( \alpha|\beta|\gamma \) notation of Chang and Lee [5] to denote the problem under study, with extensions to include families of jobs and batch deliveries. Then the problem is denoted by \( 1 \rightarrow D, f|v = 1, s_i, c_i, t_i|C_{\text{max}} \). In the \( \alpha \) field, \( 1 \rightarrow D, f \) means that the jobs are processed on a single machine and processing completed jobs are delivered to customers located in \( f \) areas. In the \( \beta \) field, \( v = 1 \) means that there is only one vehicle to deliver the jobs, and \( s_i, c_i, \) and \( t_i \) denote the
setup time, the capacity of the vehicle and the delivery time for jobs in family $F_i (1 \leq i \leq f)$, respectively. In the $\gamma$ field, $C_{\text{max}}$ denotes the time when the vehicle finishes delivering the last delivery batch to its customer and returns to the machine.

Hall and Potts [9] considered a variety of scheduling, batching, and delivery scenarios that arise in an arborescent supply chain. The objective is to minimize the overall processing and delivery cost. Transportation capacity was not considered in their models. They showed that if decision makers at different stages of a supply chain make poorly coordinated decisions at the operational level, substantial inefficiencies may result. Eren"ugc et al. [7] emphasized the operational aspects of supply chains and the need for decision making at the supplier, plant, and distribution stages of a supply chain. Lee and Chen [12] studied the problem combining machine scheduling and finished job delivery together. Jobs are delivered in batches by vehicles. Both transportation times and vehicle capacity were considered in their models in which all jobs take the same amount of physical space in a transportation vehicle. Chang and Lee [5] extended Lee and Chen's work to the situation where each finished job has a different size; hence, assigning jobs to delivery batches amounts to solving the bin-packing problem. They proposed a heuristic with a worst-case analysis for each special scenario associated with different processing settings and customers. Li et al. [13] developed a single machine scheduling model that incorporates routing decisions of a delivery vehicle that serves customers at different locations. The objective is to minimize the sum of job arrival times. They proposed dynamic programming algorithms for the special case with a single customer and for the general case with arbitrary customers. To learn more about research results on this aspect, the reader is referred to see Ahmadi et al. [1], Thomas and Griffin [19], Mazdeh et al. [15], Zhong et al. [21], and Chen and Lee [6].

Another line of research related to the problem under research focuses on scheduling with families of jobs. Bruno and Downey [4] studied single machine problems with deadlines and setup times. They showed that the problem is strongly NP-hard. Baker and Magaine [3] examined a single machine scheduling with sequence-independent setups between different families of jobs. For this NP-hard problem of minimizing maximum lateness with job families, they exploited special structure to compress the effective problem size by creating composite jobs and accelerated the enumeration with dominance properties and lower bounds. Jin et al. [11] extended Baker and Magaine’s work to the situation where sequence-dependent setups were required between different families of jobs. They proposed a simulated annealing algorithm with the new neighborhood to solve the problem. Liaee and Emmons [14] reviewed scheduling theory concerning the processing of several families of jobs on single or parallel facilities. For various performance measures, they classified the different problems as NP-hard, efficiently solvable or open. Schaller [18] considered a single machine scheduling problem to minimize total tardiness when family setups exist, and proposed optimal branch-and-bound procedures.

There exist many research results on scheduling and batching problems (see the survey papers by Potts and Wassenhove [17], Webster and Baker [20], Potts and Kovalyov [16], and Allahverdi et al. [2]). However, these researches did not take transportation times into consideration, i.e., they assumed that delivery of a job can be made whence its processing is completed.

The remainder of this paper is organized as follows. In Section 2, we introduce some notations and three useful lemmas. In Section 3, we study the scheduling problem under the group technology (GT) assumption and propose an $O(n \log n)$-time optimal algorithm for $1 \rightarrow D, f | v = 1, s_i, c_i, t_i, GT| C_{\text{max}}$. Section 4 discusses the scheduling problem without the GT assumption. In Section 4.1, we show that problem $1 \rightarrow D, f | v = 1, s_i, c_i, t_i | C_{\text{max}}$ is NP-hard. In Section 4.2, we give an $O(f^2n^2)$-time dynamic programming algorithm, where $n$ is the number of jobs, and $f$ is the number of families. In Section 4.3, we provide a heuristic algorithm with a performance ratio of $3/2$.

### 2. Preliminaries

In this section, we first give some notations that will be used in this paper and then give three lemmas for the characterization of the delivery batch structure.

For each $i = 1, 2, \ldots, f$, the following notations will be used:

- $n_i$ the number of jobs in family $F_i$
- $s_i$ the setup time for family $F_i$
- $t_i$ the delivery time of jobs in $F_i$, i.e., $t_i$ is the time needed for a round of transportation of the vehicle to deliver the jobs in $F_i$ to their specified customer and return to the machine
- $c_i$ the capacity of the vehicle for family $F_i$, i.e., at most $c_i$ jobs in $F_i$ can be delivered in a round of transportation
- $j_{ij}$ the $j$th job in family $F_i$, $j = 1, 2, \ldots, n_i$
- $p_{ij}$ the processing time of job $j_{ij}$, $j = 1, 2, \ldots, n_i$
- $P_i$ the total processing times of jobs in family $F_i$, i.e., $P_i = \sum_{1 \leq j \leq n_i} p_{ij}$

Note that the objective value $C_{\text{max}}$ is a regular measure of performance. So we can assume that all jobs are processed on the machine without idle time. The following lemma can be easily proved by the pairwise interchange argument.

**Lemma 2.1.** For $1 \rightarrow D, f | v = 1, s_i, c_i, t_i | C_{\text{max}}$, there exists an optimal schedule such that the jobs within each family are processed and delivered in nondecreasing order of their processing times (SPT). □

We called a schedule $\pi$ an SPT schedule if, in schedule $\pi$, the jobs within each family are processed and delivered in SPT order. From Lemma 2.1, we can see that an optimal SPT schedule must be an optimal schedule.
For simplicity, we re-index the jobs within each family according to the SPT rule, i.e., \( p_{11} \leq p_{21} \leq \cdots \leq p_{in} \) for \( 1 \leq i \leq f \). Furthermore, for each \( i \) with \( 1 \leq i \leq f \), we define \( q_i \) and \( u_i \) to be the integers with \( n_i = c_i q_i + u_i \) and \( 0 < u_i \leq c_i \).

**Lemma 2.2.** For \( 1 \rightarrow D, f|v = 1, s_j, c_j, t_j|C_{\text{max}} \), there exists an optimal SPT schedule satisfying the following properties:

(i) Jobs assigned to one delivery batch are processed consecutively in a processing batch on the machine.

(ii) Early processed jobs are delivered no later than those processed later.

(iii) The delivery batch of each family can be determined by the first-only-empty (FOE) \([10]\) batch rule. Specially, for each family \( F_i \), \( 1 \leq i \leq f \), there are \( q_i + 1 \) delivery batches; the first delivery batch contains \( u_i \) jobs, and each of the other delivery batches contains exactly \( c_i \) jobs.

**Proof.** (i) and (ii) can be proved by the pairwise interchange argument. To prove (iii), let \( \pi \) be an optimal SPT schedule satisfying (i) and (ii). If a delivery batch of some family (say \( F_j \)), with exception of the first delivery batch, contains less than \( c_j \) jobs, we can always fill the delivery batch with more jobs from the earlier delivery batches of the same family without increasing the objective value. Repeating this procedure family by family at most \( f \) times, we can obtain an optimal schedule satisfying (iii). \( \square \)

**Lemma 2.3.** Problem \( 1 \rightarrow D, f|v = 1, s_j, c_j, t_j|C_{\text{max}} \) can be reduced to problem \( 1 \rightarrow D, f|v = 1, s_j, c_j = 1, t_j|C_{\text{max}} \) in linear time.

**Proof.** A direct consequence of Lemma 2.2. \( \square \)

### 3. Scheduling under GT assumption

In this section, we consider problem \( 1 \rightarrow D, f|v = 1, s_j, c_j, t_j|C_{\text{max}} \) with the group technology (GT) assumption \([14]\). Under the GT assumption, the jobs in a family must be processed consecutively. Thus, the problem may be separated into finding an optimal processing order of jobs in each family, and finding the optimal sequence of families. For ease of exposition, we denote the problem under GT assumption as \( 1 \rightarrow D, f|v = 1, s_j, c_j, t_j, GT|C_{\text{max}} \). We refer a feasible schedule subject to the GT assumption as a GT schedule.

It can be observed that Lemmas 2.1–2.3 are still valid for problem \( 1 \rightarrow D, f|v = 1, s_j, c_j, t_j, GT|C_{\text{max}} \).

**Lemma 3.1.** For \( 1 \rightarrow D, f|v = 1, s_j, c_j, t_j, GT|C_{\text{max}} \), there is an optimal GT schedule (called SPT-NO-IDLE schedule) such that the jobs within each family are processed according to SPT rule on the machine, and the delivery batches of each family are consecutively transported without idle time.

**Proof.** The first statement follows from Lemma 2.1. To prove the second statement, let \( \pi \) be an optimal SPT schedule under GT assumption. From Lemma 2.2(ii), we can assume that, for each family \( F_i \), there are no delivery batches of other families between the transportation of the first delivery batch and the last delivery batch of \( F_i \). So we can always delay the departure times of some delivery batches of \( F_i \) such that the vehicle is always busy from the departure of the first delivery batch of family \( F_i \) until the delivery completion time of \( F_i \). The result holds. \( \square \)

For each family \( F_i \), \( 1 \leq i \leq f \), we use \( \pi_i \) to denote an optimal SPT-NO-IDLE schedule of the problem restricted on family \( F_i \). The corresponding optimal objective value of \( \pi_i \) is denoted by \( C_{\text{max}}(\pi_i) \). Recall that \( q_i + 1 \) is the number of delivery batches in some optimal SPT-NO-IDLE schedule. We define an associated 3-tuple \( (a_i, b_i, \lambda_i) \) for each family \( F_i \) by the following way:

- \( a_i = s_i + p_i \),
- \( b_i = (q_i + 1)t_i \),
- \( \lambda_i = b_i - (C_{\text{max}}(\pi_i) - a_i) = s_i + p_i + (q_i + 1)t_i - C_{\text{max}}(\pi_i) \).

Under the above definition, we can see that \( a_i \) is the sum of the processing times of jobs and setup time of \( F_i \), \( b_i \) is the sum of the delivery times of the delivery batches of \( F_i \), and \( \lambda_i \) is the maximal overlap value of the processing time and the delivery time for family \( F_i \). It can be observed that

\[
C_{\text{max}}(\pi_i) = a_i - \lambda_i + b_i \quad \text{for} \quad 1 \leq i \leq f.
\]

The above discussion also tells us that, to guarantee an optimal SPT-NO-IDLE schedule of problem \( 1 \rightarrow D, f|v = 1, s_j, c_j, t_j, GT|C_{\text{max}} \), if the setup of family \( F_i \) starts at time \( h_i \), then the departure time of the first delivery batch of \( F_i \) can be given by \( \max\{h_i + a_i - \lambda_i, \delta_i\} \), where \( \delta_i \) is the delivery completion time of the family directly before \( F_i \). If \( F_i \) is the first family in the schedule, \( \delta_i \) is defined to be 0.

**Algorithm GT**

**Step 1.** For each family \( F_i \), \( 1 \leq i \leq f \), re-index the jobs according to the SPT rule, and use the FOE batch rule to determine the delivery batches. Then evaluate \( C_{\text{max}}(\pi_i) \).

**Step 2.** Associate a 3-tuple \( (a_i, b_i, \lambda_i) \) for each family \( F_i \) as defined above.

**Step 3.** Schedule each family as a single processing batch in the following way:

**Step 3.1** Partition the families into two subsets \( A_1 \) and \( A_2 \) by setting \( A_1 = \{ F_i : a_i \leq b_i \} \) and \( A_2 = \{ F_i : a_i > b_i \} \).

**Step 3.2** Process first the families in \( A_1 \) in nondecreasing order of \( a_i - \lambda_i \), then the families in \( A_2 \) in nonincreasing order of \( b_i - \lambda_i \) on the machine. Suppose that the setup of a family \( F_j \) starts at time \( h_j \), \( 1 \leq j \leq f \). At the first time \( t \) with \( t \geq h_j + a_j - \lambda_j \) and the vehicle being available, start to transport the delivery batches of \( F_j \) consecutively without idle time.

It can be observed that the running time of algorithm GT is \( O(n \log n) \).
Theorem 3.2. Algorithm GT is optimal for $1 \rightarrow D, f \mid v = 1, s_i, c_i, \tau_i \mid C_{\text{max}}$.

Proof. Let $\pi$ be an optimal GT schedule satisfying Lemma 3.1 but does not follow algorithm GT, then there exists at least two families $F_1$ and $F_2$ such that family $F_1$ follows immediately after family $F_2$, and one of following three cases occurs: (a) $F_j$ belongs to $A_k$, and $F_k$ belongs to $A_k$; (b) Both $F_j$ and $F_k$ belong to $A_k$, and $a_j - \lambda_j = a_k - \lambda_k$; (c) Both $F_j$ and $F_k$ belong to $A_k$, and $b_j - \lambda_j < b_k - \lambda_k$.

In the following, we will prove that interchanging the order of family $F_j$ and $F_k$ in $\pi$ does not increase the objective value, so the result holds by repeatedly interchanging families that do not follow algorithm GT.

Suppose that family $F_j$ (if any) precedes immediately before $F_j$ and family $F_k$ follows immediately after $F_k$ in $\pi$. That is, $F_j, F_k, F_k$, and $F_j$ are four consecutively processed families. In the case that $F_j$ or $F_k$ does not exist, it is assumed to be a dummy family with setup time, processing time and delivery time being 0. We perform an adjacent pairwise interchange of family $F_j$ and $F_k$, leaving the remaining families in their original positions. The resultant new schedule is denoted by $\pi'$.

Under $\pi$, we use $K_{j1}(K_{j1}'), K_{j1}(K_{j1}'), K_{l1}(K_{l1}'),$ and $K_{l1}(K_{l1}')$ to denote the processing completion time of $F_j, F_j, F_k,$ and $F_k$, respectively, and use $K_{l2}(K_{l2}'), K_{l2}(K_{l2}')$, $K_{l2}(K_{l2}')$, and $K_{l2}(K_{l2}')$ to denote the delivery completion time of $F_j, F_j, F_k,$ and $F_k$, respectively.

Obviously, we have $K_{j1}' = K_{j1}'$ and $K_{l2} = K_{l2}'$. After exchanging the positions of families $F_j$ and $F_k$, the starting time of family $F_j$ on the machine is not affected, i.e., under $\pi$ and $\pi'$, the starting time of family $F_j$ on the machine is $K_{j1} + a_j + a_l$. In the following, we will prove that after exchanging the positions of families $F_j$ and $F_k$, the departure time of the first delivery batch of family $F_j$ will not delay.

Under $\pi$, the delivery time of the first delivery batch of family $F_j$ can be expressed as $\max\{K_{j1} + a_j - \lambda_a, K_{l2}\} = \max\{K_{j1} + a_j - \lambda_a, K_{l2}\}$. Under $\pi'$, the delivery time of the first delivery batch of family $F_j$ can be expressed as $\max\{K_{j1}' + a_j - \lambda_a, K_{l2}'\} = \max\{K_{j1} + a_j + a_l - \lambda_a, K_{l2}\}$. Note that $K_{j1} = K_{l1}'$, we only need to show that $K_{j2}' \leq K_{l2}$.

From the execution of algorithm GT, we have

$$K_{j2} = \max\{K_{j2}, K_{j1} + a_j - \lambda_k\} + b_k = \max\{K_{j2}, K_{j1} + a_j - \lambda_j\} + b_j, K_{j1} + a_j + a_k - \lambda_k\} + b_k$$

$$= \max\{K_{j2} + b_j, K_{j1} + a_j - \lambda_j\} + b_j, K_{j1} + a_j + a_k - \lambda_k\} + b_k$$

$$= \max\{K_{j2} + b_j, K_{j1} + a_j - \lambda_j\} + b_j, K_{j1} + a_j + a_k - \lambda_k\} + b_k\}$$

Similarly,

$$K_{j2}' = \max\{K_{j2}' + a_j - \lambda_j\} + b_j$$

$$= \max\{\max\{K_{j2}', K_{j1}' + a_j - \lambda_k\} + b_k, K_{j1}' + a_j + a_l - \lambda_j\} + b_j$$

$$= \max\{K_{j2}' + b_j, K_{j1}' + a_j - \lambda_k\} + b_k, K_{j1}' + a_j + a_l + a_j + a_l\} + b_j\}$$

$$\leq \min\{a_j - \lambda_j, b_j - \lambda_k\}$$

Since $K_{j2} = K_{j2}'$, the first parts of the last “max” in (2) and (3) are equal. So, we only need to show that

$$\max\{K_{j1}' + a_j - \lambda_k\} + b_k, K_{j1}' + a_j + a_l - \lambda_j\} + b_j \leq \max\{K_{j1} + a_j - \lambda_j\} + b_j, K_{j1} + a_j + a_k - \lambda_k\} + b_k\}$$

Recall that $K_{j1} = K_{j1}'$. Subtracting $K_{j1} + a_j + a_k - \lambda_k$ and $K_{j1} + a_j - \lambda_j$, we have an equivalent inequality

$$\max\{-a_j - \lambda_j, -b_j - \lambda_k\} \leq \min\{a_j - \lambda_k, b_j - \lambda_k\},$$

or equivalently,

$$\min\{a_j - \lambda_k, b_j - \lambda_k\} \leq \min\{a_j - \lambda_j, b_j - \lambda_j\}$$

In case (a), according to algorithm GT, we have $a_j > b_j$ and $a_k \leq b_k$. Then $a_j - \lambda_k \leq b_k - \lambda_k$ and $b_j - \lambda_j \leq a_j - \lambda_j$, and so, (5) holds.

In case (b), according to algorithm GT, we have $a_j \leq b_j$ and $a_k \leq b_k$. Recall that $a_j - \lambda_j \leq a_k - \lambda_k$. Then $\min\{a_j - \lambda_k, b_j - \lambda_j\} \leq a_k - \lambda_k \leq \min\{a_j - \lambda_j, b_j - \lambda_j\}$. Thus, (5) holds.

In case (c), according to algorithm GT, we have $a_j > b_j$ and $a_k > b_k$. Recall that $b_j - \lambda_j < b_k - \lambda_k$. Then $\min\{a_k - \lambda_k, b_j - \lambda_j\} \leq b_j - \lambda_j \leq \min\{a_j - \lambda_j, b_j - \lambda_j\}$ holds. The result follows. □

Remark 3.3. For problem $1 \rightarrow D, f \mid v = 1, s_i, c_i, \tau_i \mid C_{\text{max}}$ without GT assumption, whence the processing batches of an optimal schedule are given, then the optimal schedule can be found by using algorithm GT.

4. Scheduling without GT assumption

In this section, we study the scheduling problem without the GT assumption. By Lemma 2.3, we can reduce problem $1 \rightarrow D, f \mid v = 1, s_i, c_i, \tau_i \mid C_{\text{max}}$ to $1 \rightarrow D, f \mid v = 1, s_i, c_i = 1, \tau_i \mid C_{\text{max}}$ in linear time.

For problem $1 \rightarrow D, f \mid v = 1, s_i, c_i = 1, \tau_i \mid C_{\text{max}}$, we first show its NP-hardness, then we give an $O(f^2)$-time dynamic programming algorithm and provide a heuristic algorithm with a performance ratio of 3/2.

4.1. NP-hardness proof

We need the following NP-complete Equal-Size Partition problem (see Garey and Johnson [8]).
**Equal-Size Partition:** Given a set of $2n + 1$ positive integers $a_1, a_2, \ldots, a_{2n}$ and $B$ such that $\sum_{i=1}^{2n} a_i = 2B$, does there exist a partition $I_1$ and $I_2$ of the index set of $S = \{1, \ldots, 2n\}$ such that $|I_1| = n$ and $\sum_{i \in I} a_i = B$ for $j = 1, 2$?

**Theorem 4.1.** Problem 1 $\Rightarrow$ $D, f|v = 1, s_i, c_i = 1, t_i|C_{\text{max}}$ is NP-hard.

**Proof.** The decision version of the scheduling problem is clearly in NP. To prove the NP-hardness, we use the NP-complete Equal-Size Partition for the reduction.

Given an arbitrary instance $(a_1, \ldots, a_{2n}; B)$ of Equal-Size Partition, we construct an instance of decision version of the scheduling problem as follows.

- There are $f = 2n + 3$ families of jobs $F_1, \ldots, F_{2n+3}$.
- Each of the first $2n$ families $F_i$ $(1 \leq i \leq 2n)$ consists two jobs $J_{i1}$ and $J_{i2}$. These families are called normal families. For each $i$ with $1 \leq i \leq 2n$, we have $s_i = B + a_i$, $t_i = 2B - a_i$, $p_{i1} = 0$ and $p_{i2} = 3B$.
- Family $F_{2n+1}$, called head-family, contains only one job $J_{2n+1}$. We have $s_{2n+1} = 0$, $p_{2n+1} = 0$ and $t_{2n+1} = 3nB + 5B$. The head-family $F_{2n+1}$ guarantees that the first delivery batch starts its delivery at time 0.
- Family $F_{2n+2}$, called partition-family, contains only one job $J_{2n+2}$. We have $s_{2n+2} = 0$, $p_{2n+2} = 4nB$ and $t_{2n+2} = 4nB$. The partition-family $F_{2n+2}$ is used to partition the normal families into two parts according to their schedule.
- Family $F_{2n+3}$, called tail-family, also contains only one job $J_{2n+3}$. We have $s_{2n+3} = 0$, $p_{2n+3} = 2nB - 2B$ and $t_{2n+3} = 0$. The tail-family will guarantee that the total setup time and processing time is equal to the total delivery time.
- The threshold value is given by $Y = 15nB + B$ which is the total delivery time of the delivery batches.
- The decision asks whether there is a schedule $\pi$ such that $C_{\text{max}}(\pi) \leq Y$.

Clearly, the construction can be done in polynomial time. We show in the sequel that the instance of Equal-Size Partition has a solution if and only if there is schedule $\pi$ for the scheduling instance such that $C_{\text{max}}(\pi) \leq Y$.

Assume that $(I_1, I_2)$ is a solution of the instance of Equal-Size Partition. That is, $(I_1, I_2)$ is a partition of $S = \{1, \ldots, 2n\}$ with $|I_1| = |I_2| = n$ and $\sum_{i \in I_1} a_i = \sum_{i \in I_2} a_i = B$. Then we can define a schedule $\pi$ by the following way.

Each family $F_i$ with $i \in I_1$ is partitioned into two processing batches $F_i(0)$ and $F_i(3B)$, where $F_i(0)$ consists the job $J_{i1}$ with processing time 0, and $F_i(3B)$ consists the job $J_{i2}$ with processing time $3B$. Each of the other families acts as a single processing batch. The processing batches are processed by the order

$$F_{2n+1} \rightarrow F_{2n+1}(0), \quad i \in I_1 \rightarrow F_{2n+2}(i), \quad i \in I_2 \rightarrow F_{2n+2}(i) + F_{i}(3B), \quad i \in I_1 \rightarrow F_{2n+3}(i).$$

Since $c_i = 1$ for each family, each job acts as a delivery batch. The delivery batches are transported in the same order of the processing of the jobs. We use $M$ to denote the machine and $V$ to denote the vehicle. Then schedule $\pi$ can be indicated by Fig. 1 with the zero-time processing of $F_{2n+1}$ being omitted.

It can be verified that schedule $\pi$ has objective value $C_{\text{max}}(\pi) = Y$.

Conversely, assume that the scheduling instance has an optimal schedule $\pi$ with $C_{\text{max}}(\pi) \leq Y$. Note that the total delivery time is $\sum_{i=1}^{2n} t_{2n+1} + \sum_{i=1}^{2n} t_i = 15nB + B = Y$. Then we have $C_{\text{max}}(\pi) \geq Y$. Consequently, $C_{\text{max}}(\pi) = Y$ and the vehicle is always busy from time 0 to time $Y$. We define a partition $(I_1, I_2)$ of $S = \{1, \ldots, 2n\}$ by setting

$$I_1 = \{i : 1 \leq i \leq 2n, \quad F_i \text{ is processed as two batches in } \pi\},$$

$$I_2 = \{i : 1 \leq i \leq 2n, \quad F_i \text{ is processed as a single batch in } \pi\}.$$

As above, we suppose that $F_i$ with $i \in I_1$ is partitioned into two processing batches $F_i(0)$ and $F_i(3B)$, where $F_i(0)$ consists the job $J_{i1}$ with processing time 0, and $F_i(3B)$ consists the job $J_{i2}$ with processing time $3B$. Note that the processing batch structure has been determined, by Remark 3.3 and the simple arguments, we can assume that $\pi$ satisfies the following conditions:

(a) Family $F_{2n+1}$, with $s_{2n+1} = p_{2n+1} = 0$, is processed first starting at time 0, since the first delivery batch must be delivered at time 0.

(b) Family $F_{2n+3}$ is processed last. Otherwise we can interchange it with the jobs from other families without increase the objective value, since $t_{2n+3} = 0$.

(c) The jobs in $F_i, 1 \leq i \leq 2n$, and $F_{2n+2}$ are processed in the following order (by Remark 3.3 and algorithm GT):

$$F_i(0), \quad i \in I_1 \rightarrow F_i, \quad i \in I_2 \rightarrow F_{2n+2} \rightarrow F_i(3B), \quad i \in I_1.$$

(d) The vehicle transports jobs in the interval $[0, 15nB + B]$ without idle time.

(e) The sum of processing times of jobs and the setup time of processing batches on the machine cannot exceed $15nB + B = Y$. 

\[\begin{array}{cccc}
M & F_i(0), i \in I_1 & F_i, i \in I_2 & F_{i}(3B), i \in I_1 \\
F_{2n+1} & F_{2n+1} & & \\
V & F_{2n+1}; i \in I_1 & F_{2n+2}; i \in I_2 & F_{2n+2}; i \in I_1 \\
0 & 3nB + 1B & 3nB + 4B & 9nB + 2B & 15nB + 2B & 15nB + B \\
\end{array}\]

**Fig. 1.** Configuration of the optimal schedule in Theorem 4.1.
Set $T = \sum_{i=1}^{2n} s_i + \sum_{i=1}^{3} p_{2n+i} + \sum_{i=1}^{2n} p_{2i} = 14nB$. We are ready to show that Equal-Size Partition has a solution. Suppose that $|I_1| = m$. Then $\sum_{i\in I_1} s_i = \sum_{i\in I_1} (a_i + B) = mB + \sum_{i\in I_1} a_i$. So the sum of processing times of jobs and the setup times of processing batches is $T + \sum_{i\in I_1} s_i = 14nB + mB + \sum_{i\in I_1} a_i$ which is less than or equal to $Y = 15nB + B$. Hence, we have

$$|I_1| = m \leq n \quad \text{and} \quad \sum_{i\in I_1} a_i \leq B. \quad (6)$$

Now, the processing completion time of family $F_{2n+2}$ is

$$Y_1 = p_{2n+1} + \sum_{i\in I_1} (s_i + p_{i+1}) + \sum_{i\in I_2} (s_i + p_{i+1} + p_{i/2}) + p_{2n+2}$$

$$= \sum_{i\in I_1\cup I_2} s_i + p_{2n+2} + \sum_{i\in I_2} p_{i/2}$$

$$= 2nB + 2B + 4nB + (2n - m)3B$$

$$= (2 + 12n - 3m)B.$$ 

But the sum of the delivery times of jobs processed before $F_{n+2}$ is

$$Y_2 = t_{2n+1} + \sum_{i\in I_1} t_i + \sum_{i\in I_2} 2t_i$$

$$= t_{2n+1} + \sum_{i\in I_1\cup I_2} t_i$$

$$= 7nB + 3B + \sum_{i\in I_2} (2B - a_i)$$

$$= (3 + 11n - 2m)B - \sum_{i\in I_2} a_i.$$ 

Since there is no idle time in the transportation, we have $Y_1 \leq Y_2$, and so $(n - m - 1)B + \sum_{i\in I_2} a_i \leq 0$. This implies

$$|I_1| = m \geq n \quad \text{and} \quad \sum_{i\in I_2} a_i \leq B. \quad (7)$$

By noting that $\sum_{i\in I_1} a_i + \sum_{i\in I_2} a_i = 2B$, from (6) and (7), we conclude that

$$|I_1| = m = n \quad \text{and} \quad \sum_{i\in I_1} a_i = \sum_{i\in I_2} a_i = B.$$ 

Consequently, $I_1$ and $I_2$ define a solution of the instance of Equal-Size Partition. The result follows. $\square$

4.2. A general dynamic programming algorithm

In this subsection, we establish a general dynamic programming algorithm for the scheduling problem $1 \rightarrow D, f|v = 1, s_i, c_i = 1, t_i|^C_{\text{max}}$. From Lemma 2.1, suppose that the jobs within each family are indexed in SPT rule.

For $f + 1$ integers $k_1, k_2, \ldots, k_f, m$ with $0 \leq k_i \leq n_i, 1 \leq i \leq f$, and $1 \leq m \leq f$, we consider the problem $1 \rightarrow D, f|v = 1, s_i, c_i = 1, t_i|^C_{\text{max}}$ on the set of jobs $\{f_j : 1 \leq i \leq f, 1 \leq j \leq k_i\}$ under the restriction that the job in the last delivery batch belongs to family $F_m (1 \leq m \leq f)$. Such a problem is denoted by $\mathcal{P}(k_1, k_2, \ldots, k_f, m)$. The following notations are used in our discussion.

- $R(k_1, k_2, \ldots, k_f, m)$ is the optimal objective value of $\mathcal{P}(k_1, k_2, \ldots, k_f, m)$.
- $\Pi(k_1, k_2, \ldots, k_f, m)$ is the set of all optimal SPT schedules of $\mathcal{P}(k_1, k_2, \ldots, k_f, m)$ assuming $R(k_1, k_2, \ldots, k_f, m)$.
- $\tau(k_1, k_2, \ldots, k_f, m)$ is the set of $m'$ such that there is a schedule in $\Pi(k_1, k_2, \ldots, k_f, m)$ such that the second last job is in $F_{m'}$, $1 \leq m' \leq f$.
- $Y(k_1, k_2, \ldots, k_f, m)$ is the minimum processing completion time of last job among all schedules in $\Pi(k_1, k_2, \ldots, k_f, m)$.

The dynamic programming recursion will calculates all values of

$$R(k_1, k_2, \ldots, k_f, m), \quad \tau(k_1, k_2, \ldots, k_f, m) \quad \text{and} \quad Y(k_1, k_2, \ldots, k_f, m)$$

in this order.

The boundary conditions for the dynamic programming are

$$\tau(k_1, k_2, \ldots, k_f, m) = \begin{cases} \{m\}, & \text{if } k_i = 0 \text{ for all } i \neq m \text{ and } k_m = 1, \\ \emptyset, & \text{otherwise.} \end{cases}$$

$$Y(k_1, k_2, \ldots, k_f, m) = \begin{cases} s_m + p_{m1}, & \text{if } k_i = 0 \text{ for all } i \neq m \text{ and } k_m = 1, \\ +\infty, & \text{otherwise.} \end{cases}$$
Without loss of generality, we assume that the processing sequence of families is only the jobs in family \( C \) associated 3-tuple \((a, b, c, \lambda)\). Theorem 4.2. \( C^*_\text{max} \geq \max \left\{ \sum_{i=1}^{f} a_i + \sum_{i=1}^{f} b_i, \ max_{1 \leq j \leq f} \{ C_\text{max}(\pi_i) \} \right\} \).

Proof. To justify the above inequality is a valid lower bound, we note that the first two terms in the maximization are the smallest total loads on the machine and the vehicle, while \( C_\text{max}(\pi_i) \) is the optimal objective value obtained by considering only the jobs in family \( F_i \).

Theorem 4.3. \( C_\text{max}(\sigma)/C^*_\text{max} \leq 3/2 \).

Proof. Without loss of generality, we assume that the processing sequence of families is \((F_1, F_2, \ldots, F_f)\) in \( \sigma \). From the execution of algorithm GT, we can see that, in schedule \( \sigma \), each family forms only one processing batch and the jobs in each family are consecutively processed on the machine and delivered by the vehicle in the same order. By the definition of \((a_i, b_i, \lambda_i)\) and the execution of algorithm GT, the objective value of \( \sigma \) can be written as

\[
C_\text{max}(\sigma) = \max_{1 \leq k \leq f} \left\{ \sum_{i=1}^{k} a_i - \lambda_i + \sum_{i=k}^{f} b_i \right\}.
\tag{8}
\]

Suppose that the maximum in (8) is attained at \( k = c \). From (8), we have

\[
C_\text{max}(\sigma) = \sum_{i=1}^{c} a_i - \lambda_c + \sum_{i=c}^{f} b_i.
\tag{9}
\]

If \( F_c \in A_1 \), then \( a_c \leq b_c \). By Lemma 4.2 and Eq. (1), we have \( a_c - \lambda_c \leq \frac{1}{2} (a_c - \lambda_c + b_c - \lambda_c) \leq \frac{1}{2} C_\text{max}(\pi_c) \leq \frac{1}{2} C^*_\text{max} \). From the execution of Step 3.2 in algorithm GT, we have \( a_i \leq b_i \) for \( 1 \leq i \leq c - 1 \). By Lemma 4.2, we obtain

\[
C_\text{max}(\sigma) = \sum_{i=1}^{c} a_i - \lambda_c + \sum_{i=c}^{f} b_i \leq \sum_{i=1}^{f} b_i + a_c - \lambda_c \leq (3/2)C^*_\text{max}.
\]

If \( F_c \in A_2 \), then \( a_c > b_c \). By Lemma 4.2 and Eq. (1), we have \( b_c - \lambda_c < \frac{1}{2} (a_c - \lambda_c + b_c - \lambda_c) \leq \frac{1}{2} C_\text{max}(\pi_c) \leq \frac{1}{2} C^*_\text{max} \). From the execution of Step 3.2 in algorithm GT, we have \( a_i > b_i \) for \( c + 1 \leq i \leq f \). By Lemma 4.2, we obtain

\[
C_\text{max}(\sigma) = \sum_{i=1}^{c} a_i - \lambda_c + \sum_{i=c}^{f} b_i \leq \sum_{i=1}^{f} a_i + b_c - \lambda_c \leq (3/2)C^*_\text{max}.
\]

The result follows. \( \square \)
References