This is the Pre-Published Version.

Batching in a two-stage flowshop with dedicated machines in the second stage

T.C.E. CHENG[1], M.Y. KOVALYOV[2] and K.N. CHAKHLEVICH[1]

[1]*Department of Management, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong*

[2]*Faculty of Economics, Belarus State University, Skorini 4, 220050 Minsk, Belarus*

Abstract

The problem of batching and scheduling n identical jobs of F, $F \geq 2$, part types in a shop made up of $F+1$ machines is studied. The processing of each job comprises two stages. The first stage is undertaken on the machine common to all jobs and the second stage is undertaken on the machine specific to a particular part type. Setup times are necessary at the first stage to switch processing from a job of one part type to a job of another part type. Jobs of the same part type processed contiguously at the first stage form a batch. The objective is to find a batch schedule minimizing makespan. We show that this problem is equivalent to a special case of a single machine family scheduling problem to minimize maximum lateness and, therefore, it can be solved in $O(n^F)$ time by a known algorithm. Furthermore, for the case $F = 2$, we present an iterative exact algorithm with $O(k^0 \log L)$ running time, where k^0 is the maximum number of batches in a schedule created in any iteration of the algorithm and L is the problem input length in unary encoding. The algorithm finds a schedule with the minimum number of batches k^* in any optimal solution. Computational experiments were conducted to investigate the relationship between k^* and k^0. For all tested examples, $k^0 = \max\{k_1^*, k_2^*\}$, where k_1^* (k_2^*) was the minimum number of batches in any optimal schedule starting with a batch of one part type (the other part type).

Keywords: Batching; Scheduling.

1 Introduction

The batch scheduling problem studied in this paper can be formulated as follows. A set of independent non-preemptive jobs each belonging to one of F, $F \geq 2$, part types $f = 1, \dots, F$ has to be scheduled for processing in a shop comprising $F + 1$ machines $M_0, M_1, \dots, M_F$. Each job is ready for processing at time zero and has to be processed first on machine M_0, and then on machine M_f if it is of part type f. A job can start its processing on machine M_f, $f \geq 1$, immediately after its completion on machine M_0. Machine M_0 requires a setup time S_f whenever there is a switch of processing from a job of part type $g \neq f$ to a job of part type f or when a job of part type f is the first job to start processing on this machine. The processing requirements for all jobs are identical on each machine but they differ among the machines. The processing time of any job on machine M_l is v_l, $l \in \{0, 1, \dots, F\}$. The number of jobs of part type f is equal to $q_f \geq 1$, $f = 1, \dots, F$. All parameters are assumed to be non-negative integers.

A maximum set of jobs of the same part type scheduled contiguously on machine M_0 is called a *batch*. The objective is to partition the jobs into batches and find a batch schedule minimizing the makespan $C_{\max}$, i.e., the maximum completion time of all jobs. We denote this problem as CMAX.

Since all jobs are identical, the sequence of jobs on each machine is immaterial. Therefore, only the batching decision is essential. Recent reviews of research on batch scheduling and scheduling with setup times have been provided by Potts and Van Wassenhove (1991), Webster and Baker (1995), Allahverdi, Gupta and Aldowaisan (1999) and Potts and Kovalyov (2000).

Motivation of problem CMAX comes from scheduling jobs at a bottleneck facility, after which the jobs will be processed on separate production lines. Each production line is dedicated to producing a part type. When the bottleneck facility is changed over from jobs of one part type to jobs of another part type, a setup time is incurred since the part types manufactured by the bottleneck facility for the subsequent production lines are different. The objective is to batch jobs of different part types so as to keep all the production lines busy with few setups on the bottleneck facility, which is equivalent to the minimization of total production time. Ching, Liao and Wu (1997) observed this production situation in a leading PC manufacturer in Taiwan, where various kinds of circuit boards were assembled on surface mounting technology facilities to feed subsequent production lines, each of which was dedicated to the production of a specific PC family.

There exist the following results for the case $F = 2$ of problem CMAX. Ching, Liao and Wu presented an integer programming formulation as well as a heuristic algorithm. Cheng and Kovalyov (1998) presented a dynamic programming algorithm with time complexity $O((q_1 + q_2)q_1^4 q_2^3)$.

In this paper, we show that problem CMAX is equivalent to a special case of the problem of scheduling jobs of F families on a single machine to minimize maximum lateness. Though the computational complexity of this special case is unknown, it can be solved to optimality in $O(q_1 q_2 \cdots q_F)$ time by the algorithm of Ghosh and Gupta (1997).

For the case $F = 2$ of problem CMAX, we present an iterative exact algorithm with $O(k^0 \log L)$ running time, where k^0 is the maximum number of batches in a schedule created in any iteration of the algorithm and L is the problem input length in unary encoding.

The algorithm finds a schedule with the minimum number of batches k^* in any optimal solution. This property is important for situations when there is a secondary criterion to minimize a total time or cost associated with setups.

Let k_f^* denote the minimum number of batches in any optimal schedule in the class of schedules starting with a batch of part type f, $f = 1, 2$. Observe that $k^* \in \{k_1^*, k_2^*\}$. Computational experiments were conducted to investigate the structure of an optimal schedule and the relationship between k^0, k_1^* and k_2^*. For all the tested examples, $k^0 = \max\{k_1^*, k_2^*\}$.

2 Equivalence to a single machine family scheduling problem

The single machine family scheduling problem can be formulated as follows. The jobs each belonging to one of F families have to be scheduled for processing on a single machine. Family f comprises the jobs $(1, f), (2, f), \ldots, (n_f, f)$, $f = 1, \ldots, F$. Each job (j, f) has a processing time $p_{(j,f)}$ and a due date $d_{(j,f)}$. A machine setup time S_f is needed whenever the machine switches from processing a job of family $g \neq f$ to a job of family f or at time zero when the machine starts with a job of family f. The objective is to find a schedule that minimizes maximum lateness $L_{\max} = \max_{(j,f)}\{C_{(j,f)} - d_{(j,f)}\}$, where $C_{(j,f)}$ is the completion time of job (j, f). In this section, it is assumed that each maximum is taken over all jobs (j, f). We denote this problem as LMAX.

Given an instance IC of problem CMAX, we construct the following instance IL of problem LMAX. The machine in problem LMAX is the machine M_0. The processing times of the jobs are all equal to v_0. Family f comprises the q_f jobs of part type f indexed as $(1, f), (2, f), \dots, (q_f, f)$. The job due dates are determined as follows:

$$d_{(j,f)} = -(q_f + 1 - j)v_f,\ f = 1, \dots, F,\ j = 1, \dots, q_f. \tag{1}$$

We now prove that the two instances are equivalent. The following statements are needed for these purposes.

Lemma 1 *Consider instance IC of problem CMAX. Given D, there exists a schedule such that $C_{\max} \le D$ if and only if the job of part type f sequenced j-th completes on machine M_0 by time*

$$D - (q_f + 1 - j)v_f,\ f = 1, \dots, F,\ j = 1, \dots, q_f.$$

Proof. Consider a schedule for which $C_{\max} \le D$. Since after the completion of the j-th job of part type f on machine M_0, this job and the remaining $q_f - j$ jobs of part type f must be processed on machine M_f by time D, so this job must be completed on machine M_0 by time $D - (q_f + 1 - j)v_f$. This statement holds for all f and j.

Now assume that the j-th job of part type f completes on machine M_0 by time $D - (q_f + 1 - j)v_f$. Then this job and the remaining $q_f - j$ jobs of part type f can be processed on machine M_f in the time interval $[D - (q_f + 1 - j)v_f, D]$ of length $(q_f + 1 - j)v_f$. This statement holds for all f and j. Therefore, for the corresponding schedule, $C_{\max} \le D$. ■

Let σ^* and C^* be an optimal batch schedule and the corresponding makespan value for the instance IC of problem CMAX.

Corollary 1 *According to σ^*, the j-th job of part type f completes on machine M_0 by time $D^*_{(j,f)} := C^* - (q_f + 1 - j)v_f$ and there is no batch schedule σ such that $\max_{(j,f)}\{C_{(j,f)}(\sigma) - D^*_{(j,f)}\} < 0$.*

Now we prove the main result of this section.

Theorem 1 *A batch schedule σ^* that is optimal for the instance IC of problem CMAX is an optimal batch schedule for the instance IL of problem LMAX with the value $L_{\max}(\sigma^*) = C^*$.*

Proof. For an arbitrary batch schedule σ, we have

$$L_{\max}(\sigma) = \max_{(j,f)}\{C_{(j,f)}(\sigma) - d_{(j,f)}\} =$$

$$\max_{(j,f)}\{C_{(j,f)}(\sigma) - (C^* - (q_f + 1 - j)v_f)\} + C^* = \max_{(j,f)}\{C_{(j,f)}(\sigma) - D^*_{(j,f)}\} + C^*.$$

From Corollary **??**, it follows that $\max_{(j,f)}\{C_{(j,f)}(\sigma) - D^*_{(j,f)})\} \geq 0$ and $\max_{(j,f)}\{C_{(j,f)}(\sigma) - D^*_{(j,f)})\} = 0$ for $\sigma = \sigma^*$. Therefore, $L_{\max}(\sigma^*) = C^*$ is the minimum objective function value for the instance IL of problem LMAX. ■

Bruno and Downey (1978) proved that problem LMAX is NP-hard. However, our problem CMAX is equivalent to a special case of problem LMAX, the computational complexity of which is unknown.

Ghosh and Gupta (1997) derived a dynamic programming algorithm for problem LMAX with running time $O(q_1 q_2 \cdots q_F)$. This algorithm can be used to solve our problem CMAX if $d_{(j,f)}$ are calculated according to (**??**) and the job processing times are all equal to v_0.

3 Problem CMAX for $F = 2$

For $F = 2$, problem CMAX can be solved in $O(q_1 q_2)$ time by the algorithm of Ghosh and Gupta (1997). In this section, a different algorithm is derived for this case, which appears to be more efficient.

3.1 Lower bounds and trivially solvable cases

As above, let C^* and σ^* denote the minimum value of $C_{\max}$ for problem CMAX and the corresponding optimal schedule, respectively. Since $F = 2$, a schedule is completely characterized by the sequence of batch sizes and the part type of the first batch. In this subsection, we establish lower bounds LB_1, LB_2 and LB_3 for C^*, which are based on the following observations.

We first note that minimum delays of $S_1 + v_0$ and $S_2 + v_0$ time units are needed before the processing of any job can start on machines M_1 and M_2, respectively. Therefore, the last jobs on machines M_1 and M_2 cannot be completed at time earlier than

$$LB_1 = S_1 + v_0 + q_1 v_1 \text{ and } LB_2 = S_2 + v_0 + q_2 v_2,$$

respectively.

Furthermore, the last job on machine M_0 cannot complete at time earlier than $S_1 + S_2 + (q_1 + q_2)v_0$. Since after finishing on machine M_0 this job has to be processed on machine M_1 or M_2, we have

$$LB_3 = S_1 + S_2 + (q_1 + q_2)v_0 + \min\{v_1, v_2\}.$$

We now identify special cases of the batching problem, in which C^* coincides with one of the above lower bounds.

In the sequel, assume without loss of generality, that $v_1 \leq v_2$.

Case $v_1 \leq v_2 \leq v_0$. Construct batch sequence σ_2 in which there are two batches and the first batch is of part type 2 (see Fig. **??**).

Insert Fig. **??** here

From Fig. **??** it is easy to see that σ_2 is optimal because $C_{\max}(\sigma_2) = LB_3$.

Case $v_1 \leq v_0 < v_2$. In this case, batch sequence σ_2 is again optimal. The maximum completion time is reached either on machine M_1 or M_2 (see Fig. **??** and **??**).

Insert Fig. **??** and **??** here in this order

Thus, for the case $v_1 \leq v_0 < v_2$, we have $C_{\max}(\sigma_2) = \min\{LB_2, LB_3\} = C^*$.

We conclude this subsection by noting that the only non-trivial case of the problem is when the processing requirements on machines M_1 and M_2 are greater than that on machine M_0, i.e., $v_0 < v_1 \leq v_2$.

3.2 The non-trivial case $v_0 < v_1 \leq v_2$

We start with partitioning the set of feasible batch sequences into three subsets Σ_0, Σ_1 and Σ_2. All sequences in Σ_0 consist of two batches. Clearly, there are only two sequences in Σ_0. All sequences in Σ_1 consist of more than two batches and start with a batch of part type 1, and all sequences in Σ_2 consist of more than two batches and start with a batch of part type

2. Let $\sigma^{(0)}, \sigma^{(1)}$ and $\sigma^{(2)}$ denote the batch sequences delivering the minimum $C_{\max}$ values in Σ_0, Σ_1 and Σ_2, respectively. It is clear that $\sigma^* \in \{\sigma^{(0)}, \sigma^{(1)}, \sigma^{(2)}\}$.

Let σ_1 denote a batch sequence comprising two batches and starting with a batch of part type 1. We have $C_{\max}(\sigma^{(0)}) = \min\{C_{\max}(\sigma_1), C_{\max}(\sigma_2)\}$.

Now assume that there are at least three batches in an optimal batch sequence. In this case, we may use the inequalities $LB \le C^* \le UB$, where $LB = \max\{LB_1, LB_2, LB_3\}$ and $UB = C_{\max}(\sigma^{(0)}) - 1$.

We first develop an algorithm that either finds batch sequence $\sigma^{(1)}$ or detects that $\Sigma_1 = \phi$. At the upper level of this algorithm, a bisection search over the range $[LB, UB]$ is conducted.

At the beginning, the question "Is there a batch sequence $\sigma \in \Sigma_1$ such that $C_{\max}(\sigma) \le C$?" is answered for $C = LB$ and $C = UB$. The procedure stops if the answer for $C = LB$ is "yes" or the answer for $C = UB$ is "no". In the former case, we find σ and set $\sigma^* = \sigma$. In the latter case, $\Sigma_1 = \phi$. Otherwise, we set $T = LB$, $V = UB$ and conduct a general iteration of the bisection search by answering the above question for $C = \lfloor (T+V)/2 \rfloor$. If the answer is "yes", then we find and keep the corresponding batch sequence, reset $V = C$, retain T unchanged and go to the next iteration of the bisection search. If the answer is "no", then we reset $T = C$, retain V unchanged and go to the next iteration of the bisection search. The procedure stops when the length of the current interval $[T, V]$ to be partitioned is equal to zero. In this case, $\sigma^{(1)}$ is the last found batch sequence.

It remains to provide an algorithm for solving the problem of finding a batch sequence $\sigma \in \Sigma_1$ such that $C_{\max}(\sigma) \le C$, $C \in [LB, UB]$. This algorithm plays a central role in our study.

Let the sequence σ to be found be $(A_1, B_1, A_2, \dots)$, where A_j and B_j are batches of part types 1 and 2, respectively, appearing j-th among batches of the same part type in σ. Let b_1^j and b_2^j denote the sizes of batches A_j and B_j, respectively: $|A_j| = b_1^j$ and $|B_j| = b_2^j$.

The following algorithm CHECK either finds the above batch sizes and, hence, the corresponding batch sequence σ or determines that such a sequence does not exist. The main idea of this algorithm is to assign as many jobs to the current batch of part type f as permitted by the latest possible starting time for the unassigned jobs of part type $(3-f)$ on machine M_0. This latest time is equal to $C - (x \cdot v_{3-f} + v_0)$, where x is the number of the unassigned jobs of part type $(3-f)$. If the starting time of the current batch and the latest possible starting time for the next batch are known, the size of the current batch can easily be calculated.

In iteration k of algorithm CHECK, batch sizes b_1^k and b_2^k are computed and, hence, the corresponding batches are identified. The total numbers of jobs of part types 1 and 2 assigned to batches up to iteration k are denoted by T_1^k and T_2^k, respectively. The total processing time of the batches plus the corresponding setup times assigned to machine M_0 up to iteration k plus two extra setup times S_1 and S_2 is denoted by P_k. Thus, we have $T_f^k = \sum_{j=1}^k b_f^j$, $f \in \{1, 2\}$, and $P_k = (k+1)(S_1 + S_2) + v_0 \sum_{j=1}^k (b_1^j + b_2^j) = (k+1)(S_1 + S_2) + v_0(T_1^k + T_2^k)$, $k = 1, 2, \ldots$

Algorithm CHECK.

Input: S_1, S_2, q_1, q_2, v_0, v_1, v_2, and $C \in [LB, UB]$.

Output: batch sizes b_1^1, b_2^1, $b_1^2, b_2^2, \ldots$ or determination that sequence $\sigma \in \Sigma_1$ such that $C_{\max}(\sigma) \le C$ does not exist.

Step 1. (Initialization) Set $T_1^0 = T_2^0 = 0$, $P_0 = S_1 + S_2$ and $k = 1$.

Step 2. (Recursive computation of batch sizes) Compute the following.

$$b_1^k = \lfloor \frac{C - (q_2 - T_2^{k-1})v_2 - P_{k-1}}{v_0} \rfloor - 1.$$

If $b_1^k \le 0$, then stop: sequence σ does not exist.

If $T_1^{k-1} + b_1^k \ge q_1$, then A_k is the last batch of part type 1 and, hence, B_k is the last batch of part type 2. Therefore, reset $b_1^k = q_1 - T_1^{k-1}$ and compute $b_2^k = q_2 - T_2^{k-1}$. Output σ that is determined by batch sizes b_1^1, b_2^1, $b_1^2, b_2^2, \ldots, b_1^k, b_2^k$.

If $T_1^{k-1} + b_1^k < q_1$, then compute $T_1^k = T_1^{k-1} + b_1^k$ and $P_k' = P_{k-1} + S_1 + b_1^k v_0$.

Compute

$$b_2^k = \lfloor \frac{C - (q_1 - T_1^k)v_1 - P_k'}{v_0} \rfloor - 1.$$

If $b_2^k \le 0$, then stop: sequence σ does not exist.

If $T_2^{k-1} + b_2^k \ge q_2$, then B_k is the last batch of part type 2 and, hence, A_{k+1} is the last batch of part type 1. Therefore, reset $b_2^k = q_2 - T_2^{k-1}$ and compute $b_1^{k+1} = q_1 - T_1^k$. Output σ that is determined by batch sizes b_1^1, b_2^1, $b_1^2, b_2^2, \ldots, b_2^k, b_1^{k+1}$.

If $T_2^{k-1} + b_2^k < q_2$, then compute $T_2^k = T_2^{k-1} + b_2^k$ and $P_k = P'_k + S_2 + b_2^k v_0$. Repeat Step 2 for $k = k+1$. ■

Theorem 2 *Algorithm* CHECK *either finds a batch sequence* $\sigma \in \Sigma_1$ *such that* $C_{\max}(\sigma) \leq C$, $C \in [LB, UB]$, *or establishes that such a sequence does not exist. Moreover, the sequence* σ *found by the algorithm has the minimum number of batches among schedules satisfying the above conditions.*

Proof. We first show that if algorithm CHECK finds a batch sequence σ, then $C_{\max}(\sigma) \leq C$. The makespan value of σ can be calculated as follows.

$$C_{\max}(\sigma) = \max\{S_1 + v_0 + q_1 v_1, S_1 + S_2 + (b_1^1 + 1)v_0 + q_2 v_2,$$
$$2S_1 + S_2 + (b_1^1 + b_2^1 + 1)v_0 + (q_1 - b_1^1)v_1, \ \dots \}.$$

We have $S_1 + v_0 + q_1 v_1 = LB_2 \leq LB \leq C$. Furthermore, from

$$(b_1^1 + 1)v_0 = \lfloor \frac{C - q_2 v_2 - (S_1 + S_2)}{v_0} \rfloor v_0 \leq C - q_2 v_2 - (S_1 + S_2),$$

we obtain

$$S_1 + S_2 + (b_1^1 + 1)v_0 + q_2 v_2 \leq C,$$

and from

$$(b_1^1 + b_2^1 + 1)v_0 = b_1^1 v_0 + \lfloor \frac{C - (q_1 - b_1^1)v_1 - (2S_1 + S_2 + b_1^1 v_0)}{v_0} \rfloor v_0 \leq$$
$$b_1^1 v_0 + C - (q_1 - b_1^1)v_1 - (2S_1 + S_2 + b_1^1 v_0),$$

we obtain

$$2S_1 + S_2 + (b_1^1 + b_2^1 + 1)v_0 + (q_1 - b_1^1)v_1 \leq C.$$

By repeating the above calculations for the remaining batches in σ, we prove that $C_{\max}(\sigma) \leq C$.

Furthermore, σ comprises at least three batches. This statement is easily deduced from $C_{\max}(\sigma) \leq C$, $C \in [LB, UB]$, $\Sigma_0 = \{\sigma_1, \sigma_2\}$ and $C_{\max}(\sigma_2) > UB$, $C_{\max}(\sigma_1) > UB$.

Let us now consider an arbitrary batch sequence $\sigma' \in \Sigma_1$ such that $C_{\max}(\sigma') \leq C$, $C \in [LB, UB]$. We show that σ' can be transformed into the batch sequence σ constructed by the algorithm CHECK with a fewer or equal number of batches. The proof is illustrated by a diagram given in Fig. **??**.

Insert Fig. ?? here

Let $\sigma' = (G_1, H_1, G_2, \dots)$, where G_j and H_j are batches of part types 1 and 2, respectively, appearing j-th among batches of the same part type in σ', and let $|G_j| = a_1^j$ and $|H_j| = a_2^j$, $j = 1, 2, \dots$ Denote the times when machines M_1 and M_2 finish their processing by F_1 and F_2, respectively.

Consider time $t(H_1) = S_1 + a_1^1 v_0 + S_2 + v_0$, when the first job of batch H_1 starts its processing on machine M_2. If $C - q_2 v_2 - t(H_1) \geq v_0$, then move a job from G_2 to G_1. Value F_1 will not increase and value F_2 may increase by at most v_0 due to the starting time of batch H_1 being postponed by v_0 time units after the expansion of batch G_1. Therefore, for the new batch sequence, we have $C_{\max} \leq C$. Continue to move jobs from $G_2, G_3, \dots$ to G_1 and stop when $C - q_2 v_2 - t(H_1) < v_0$. Let us retain the same notation σ' for the new batch sequence and leave all other related notations unchanged. It is easy to verify that after the moves are stopped, the number of jobs in batch G_1 is equal to

$$b_1^1 = \lfloor \frac{C - q_2 v_2 - (S_1 + S_2)}{v_0} \rfloor - 1.$$

This value is also determined in iteration $k = 1$ of algorithm CHECK. Therefore, in the new sequence, $G_1 = A_1$. Clearly, the number of batches is not increased.

Notice that $b_1^1 < q_1$ because otherwise there should be two batches in σ'. It was shown above that in this case $C_{\max}(\sigma') > UB$, which contradicts our assumption about σ'.

Now assume that $G_1 = A_1$. Consider time $t(G_2) = \max\{S_1 + v_0 + b_1^1 v_1, 2S_1 + S_2 + (b_1^1 + a_2^1 + 1)v_0\}$, at which the first job of batch G_2 starts its processing on machine M_1. If $C - (q_1 - b_1^1)v_1 - t(G_2) \geq v_0$ and H_1 is not the last batch of part type 2, then move a job from H_2 to H_1. Value F_2 will not increase and value F_1 may increase by at most v_0. Again, for the new batch sequence, $C_{\max} \leq C$. Continue moving jobs from $H_2, H_3, \dots$ to H_1 and stop when $C - (q_1 - b_1^1)v_1 - t(G_2) < v_0$. Leave the notation for the new batch sequence unchanged. After the moves are stopped, the number of jobs in batch H_1 is equal to

$$b_2^1 = \lfloor \frac{C - (q_1 - b_1^1)v_1 - (2S_1 + S_2 + b_1^1 v_0)}{v_0} \rfloor - 1.$$

This value is determined in iteration $k = 1$ of algorithm CHECK. Therefore, we have obtained $H_1 = B_1$. The number of batches is not increased.

If H_1 is the last batch of part type 2, i.e., $|H_1| = q_2$, then it is easy to verify that the initial value b_2^1 found by algorithm CHECK, i.e., the one calculated before resetting, is equal to q_2 if $C - (q_1 - b_1^1)v_1 - t(G_2) < v_0$, and it is greater than q_2 if $C - (q_1 - b_1^1)v_1 - t(G_2) \geq v_0$. The final value b_2^1 is equal to q_2.

Thus, the sequence σ' can be transformed so that $G_1 = A_1$, $H_1 = B_1$ and the number of batches is not increased. Repeating this argument for the remaining batches in σ' completes the proof. ■

Algorithm CHECK incorporated in the binary search procedure described above allows us to find the batch sequence $\sigma^{(1)}$ that is optimal in the set Σ_1 or determine $\Sigma_1 = \phi$. It can also be used to find the batch sequence $\sigma^{(2)}$ or determine $\Sigma_2 = \phi$. The only difference in the description of algorithm CHECK is that the notation for part types must be switched. If $\sigma^{(1)}$ is found, then the upper bound UB can be reset to be equal to $C_{\max}(\sigma^{(1)})$.

Recall that k_f^* denotes the number of batches in the sequence $\sigma^{(f)}$, $f = 1, 2$. An optimal batch sequence σ^* with the minimum number of batches k^* for the case $v_0 < v_1 \leq v_2$ can be found lexicographically from

$$(C_{\max}(\sigma^*), k^*) = \text{lexmin}\{(C_{\max}(\sigma^{(0)}), 2), (C_{\max}(\sigma^{(1)}), k_1^*), (C_{\max}(\sigma^{(2)}, k_2^*)\},$$

where $\text{lexmin}\{(a_1, k_1), (a_2, k_2), (a_3, k_3)\} = (a_j, k_j)$ if $a_j = \min\{a_1, a_2, a_3\}$ and $k_j = \min\{k_l \mid a_l = a_j,\ l = 1, 2, 3\}$.

It is easy to see that algorithm CHECK runs in $O(k)$ time, where k is the number of batches in the final complete or partial schedule created by this algorithm. It is used by the bisection search procedure $O(\log UB)$ times. Since UB is bounded by a polynomial of L, the complexity of finding σ^* is equal to $O(k^0 \log L)$, where k^0 is the maximum number of batches in a schedule created in any application of algorithm CHECK.

If algorithm CHECK is applied for the set $\Sigma_f \neq \phi$ and a trial value $C \geq C_{\max}(\sigma^{(f)})$, then it follows from Theorem **??** that the number of batches created by this algorithm does not exceed k_f^*, $f = 1, 2$. We conjecture that this statement also holds if $C < C_{\max}(\sigma^{(f)})$, $f = 1, 2$. If the conjecture is true, then the running time of our algorithm can be estimated as $O((k_1^* + k_2^*) \log L)$.

4 Computational experiments

In this subsection, we observe the results of computational experiments conducted to investigate the structure of an optimal schedule with the minimum number of batches and the relationships between k^0, k_1^* and k_2^*.

Only the non-trivial case $v_0 < v_1 \leq v_2$ was analyzed. About 500 problem instances were tested for various configurations of the parameters. The main observations of the experiments are the following.

1) For the examples with $S_1 = S_2 = 0$, $v_0 = 1$, $v_1 = v_2 = 2$ and $q_1 = q_2$, we have $k^* = q_1 + 1$. The latter equation can easily be proved analytically. It follows that, in general, k^* is not bounded by a polynomial in $\log L$. However, for the above examples, the sizes of the first and last batches were equal to 1 and all the other batch sizes were equal to 2. Therefore, it might be the case that for any instance of the problem, either $k^* \leq \log L$, or σ^* has a structure that allows the problem to be solved in polynomial time.
2) Batch sizes were non-increasing or non-decreasing for each part type, with the possible exception for the first or last batch of one part type.
3) The value of k^* increased, decreased or increased and then decreased while any of the problem input parameters r, $r \in \{S_1, S_2, q_1, q_2, v_0, v_1, v_2\}$, increased assuming that the other parameters did not change, i.e., the dependency $k^* = f(r)$ is Λ-*shaped*.
4) For all tested instances, $k^0 = \max\{k_1^*, k_2^*\}$.

Note that the integer programming algorithm by Ching, Liao and Wu (1997) required up to 2.5 hours on a Sun 3/60 workstation to find an optimal solution for an example with $q_1 + q_2 = 1200$ while our algorithm found the same solution in less than 0.01 second on a 260 Mhz Pentium-II. Moreover, the maximum number of batches of each part type was bounded by $m = 6$ in their experiments because the integer programming formulation of the batching problem included $2m$ variables and $10m - 1$ constraints. We had shown that there were examples with $2m \geq k^* = q_1 + 1$.

5 Conclusions

The problem of batching and scheduling jobs of F part types in a shop made up of $F+1$ machines is shown to be equivalent to a special case of a single machine family scheduling problem. An $O(q_1 \cdots q_F)$ time algorithm for the latter problem is known in the literature.

An iterative algorithm with $O(k^0 \log L)$ running time has been presented to solve the problem with $F = 2$ part types. Here k^0 is the maximum number of batches in a schedule created in any iteration of the algorithm and L is the problem input length in unary encoding. The algorithm finds a schedule with the minimum number of batches k^* in any optimal solution. Computational experiments were conducted. For all the tested examples, $k^0 = \max\{k_1^*, k_2^*\}$, where k_f^* was the minimum number of batches in any optimal schedule in the class of schedules starting with a batch of part type f, $f = 1, 2$. The algorithm outperformed the previous algorithms suggested in the literature both in the quality of the solution with respect to the number of batches created and in running time.

Further research can be undertaken to establish the computational complexity of the general case of the problem. Relevant practical extensions of this model are also of interest. They can include the presence of jobs with distinct processing times, ready times or due dates, transportation between the machines, total cost criteria, etc.

Acknowledgments

This research was supported in part by The Hong Kong Polytechnic University under grant number G-S818 and the Croucher Foundation under a Croucher Senior Research Fellowship for the first author. The second author was also supported by INTAS (Projects 96-0820 and 00-217).

References

Allahverdi, A., Gupta, J.N.D. and Aldowaisan, T. (1999) A review of scheduling research involving setup considerations. *Omega*, **27**, 219-239.

Bruno, J. and Downey, P. (1978) Complexity of task sequencing with deadlines, set-up times and changeover costs. *SIAM Journal on Computing,* **7**, 393-404.

Ching, C.Y., Liao, C.-J. and Wu, C.J. (1997) Batching to minimize total production time for two part types. *International Journal of Production Economics*, **48**, 63-72.

Cheng, T.C.E. and Kovalyov, M.Y. (1998) An exact algorithm for batching and scheduling two part types in a mixed shop: a technical note. *International Journal of Production Economics*, **55**, 53-56.

Ghosh, J.B. and Gupta, J.N.D. (1997) Batch scheduling to minimize maximum lateness. *Operations Research Letters*, **21**, 77-80.

Potts, C.N, and Kovalyov, M.Y., 2000. Scheduling with batching: a review, *Europ. J. Oper. Res.*, **120**: 228-249.

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

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

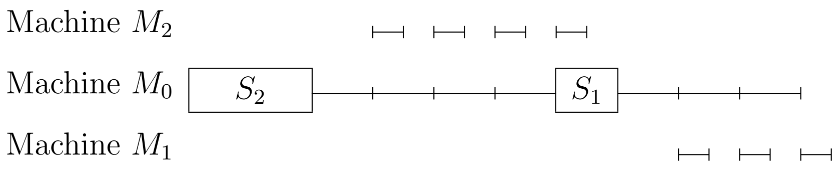

Figure 1: Sequence σ_2 for the case $v_1 \leq v_2 \leq v_0$

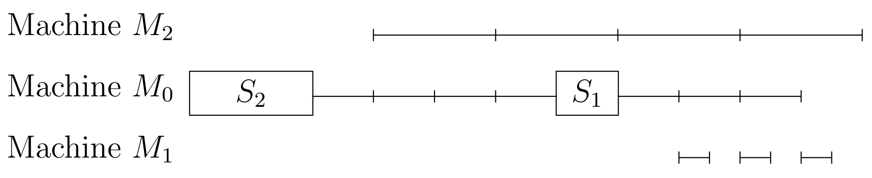

Figure 2: Case $v_1 \leq v_0 < v_2$. The value of $C_{\max}$ is reached on machine M_2

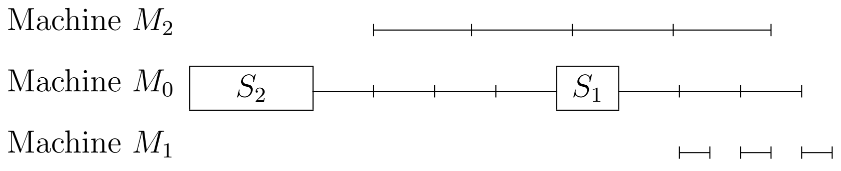

Figure 3: Case $v_1 \leq v_0 < v_2$. The value of $C_{\max}$ is reached on machine M_1

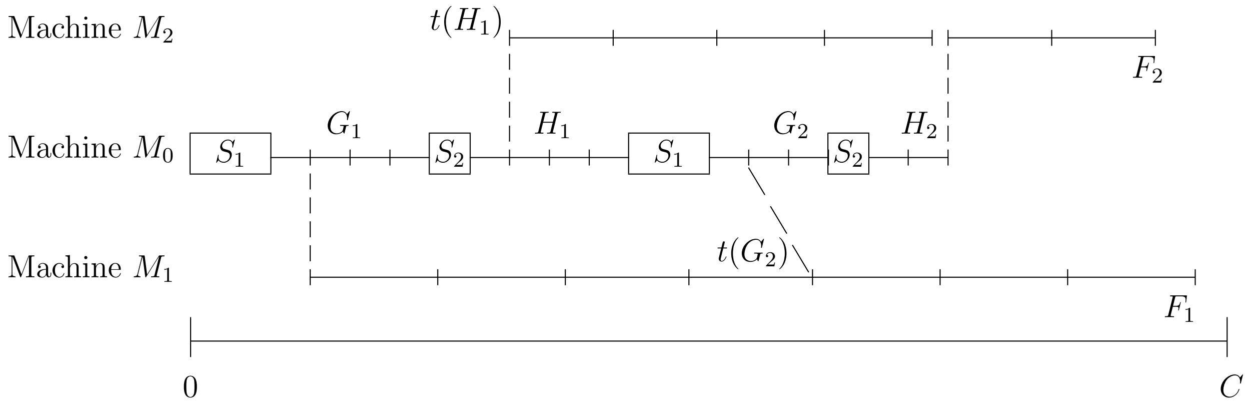

Figure 4: Sequence σ'