UNIVERSITE PARIS DAUPHINE

Laboratoire d'Analyse et Modélisation de Systèmes pour l'Aide à la Décision

CNRS UMR 7024

CAHIER DU LAMSADE

290

Decembre 2009

The two-machine flow-shop serial-batching scheduling problem with limited batch size

M. A. ALOULOU, A. BOUZAIENE, N. DRIDI, D. VANDERPOOTEN

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

The two-machine flow-shop serial-batching scheduling problem with limited batch size

Mohamed Ali Aloulou[1], Afef Bouzaiene[1,2], Najoua Dridi[2], and Daniel Vanderpooten[1]

[1]LAMSADE, Université Paris Dauphine, France
(aloulou,bouzaiene,vdp)@lamsade.dauphine.fr
[2]OASIS - Ecole Nationale d'Ingénieurs de Tunis, Tunisia
Najoua.Dridi@enit.rnu.tn

Abstract

We consider the the two-machine flow-shop serial-batching scheduling problem where the batches have limited size. Two criteria are considered here. The first criterion is to minimize the number of batches. This criterion reflects situations where processing of any batch induces a fixed cost, which leads to a total cost proportional to the number of batches. The second criterion is the makespan. We study the complexity of the problem and propose polynomial-time algorithms for some particular cases and an approximation algorithm with a guaranteed performance for the general case.

Keywords: two-machine flow-shop, serial batching, limited batch size, makespan, batch cost.

1 Problem formulation and motivation

The following two-machine flow-shop scheduling problem is considered. We have a set of n jobs to be processed. Each job $i, i = 1, \ldots n$, is made up of two operations. The first operation is handled on machine M_1 and its processing time is denoted by a_i. The second operation is handled on machine M_2 and its processing time is denoted by b_i. M_1 and M_2 are two serial-batch (or sum-batch) machines with a limited capacity, denoted by c, in terms of the number of jobs (each job i has a unit size $s_i = 1$). The following assumptions are made: (i) preemption is not allowed, (ii) each batch in both machines does not contain more than c operations, (iii) operations within a batch can be processed in any order, (iv) the total processing time of a batch is equal to the sum of the processing times of the operations in the batch, (v) an operation is assumed to be completed if all the operations in its batch are completed (batch availability constraint). Two criteria are considered here. The first criterion is to minimize the number of batches $\#batch$. This criterion reflects situations where processing of any batch induces a fixed cost, which leads to a total cost proportional to $\#batch$. The second criterion is the makespan $C_{\max}$, i.e. the completion time of the last job on machine M_2. This bicriteria scheduling problem is denoted by $F2|\text{sum-batch}, c|(\#batch, C_{\max})$.

Our model can be seen as a generalization of the model proposed by [1], where the authors studied the problem of finding group-schedules for problem $F2||C_{max}$. A group-schedule is a sequence of groups of permutable operations (or batches) defined on each machine. Two

conflicting criteria are of interest: flexibility of a solution and its makespan. The flexibility is measured by the number of batches in the solution and the makespan is computed as for serial-batch models. The only difference with our model is that the number of jobs in a batch is not limited, i.e. $c = n$. Referring to previous results of [2] for $F2|\text{sum-batch}|C_{max}$ (with setup times), they conclude that the constrained problems $F2|\text{sum-batch}, C_{max} \leq \alpha|\#batch$ and $F2|\text{sum-batch}, \#batch = k|C_{max}$ are NP-hard in the strong sense. When k is fixed, the second problem is NP-hard in the ordinary sense. They also provide polynomial-time approximation algorithms for these problems. Unfortunately, these algorithms cannot be extended to our model.

If the number of jobs n is a multiple of c, i.e. $n = ch$, then the solution with a minimum number of batches contains exactly h full batches. Here, a batch is called full if it contains exactly c jobs. The lexicographic bicriteria problem $F2|\text{sum-batch}, c|Lex(\#batch, C_{\max})$ is equivalent to problem $F2|\text{sum-batch}, c\text{-in-1}|C_{\max}$, where notation c-in-1 imposes to have exactly c jobs in a batch insuring that the number of batches is minimum. This notation has been first introduced by [3] for single-machine batch scheduling. The authors considered problem $1|\text{sum-batch}, c\text{-in-1}, w_i = p_i|\sum w_i C_i$ and proved that it is NP-hard in the strong sense when $c \geq 3$ and polynomially solvable when $c = 2$. They also provide a polynomial time algorithm when the jobs are inversely agreeable, i.e. $p_i < p_j$ implies $w_i \geq w_j$.

If the number of jobs is not multiple of c, we can add $c \times \lceil \frac{n}{c} \rceil - n$ dummy jobs with zero processing times in both machines and solve problem $F2|\text{sum-batch}, c\text{-in-1}|C_{\max}$ to determine the solution minimizing the makespan under the constraint that the number of batches is minimum. By the same way, if we impose a number of batches $\#b \in \{\lceil \frac{n}{c} \rceil, \ldots, n\}$, in order to determine a solution with minimum makespan, we add $c \times \#b - n$ dummy jobs with zero processing time in both machines and solve problem $F2|\text{sum-batch}, c\text{-in-1}|C_{\max}$. Hence, solving $(n - \lceil \frac{n}{B} \rceil + 1)$ times problem $F2|\text{sum-batch}, c\text{-in-1}|C_{\max}$ allows us to solve problem $F2|\text{sum-batch}, c|(\#batch, C_{\max})$.

In this paper, we first study the complexity of problem $F2|\text{sum-batch}, c\text{-in-1}|C_{\max}$. Then we propose polynomial-time algorithms for some particular cases and an approximation algorithm with a guaranteed performance for the general case.

2 Complexity

Theorem 1 *Problem $F2|$sum-batch, c-in-1$|C_{\max}$ is NP-hard in the strong sense for $c \geq 3$, even if $a_i = b_i$ for all $i = 1, \ldots, n$.*

Proof : First remark that in this case all batches have the same duration on both machines and consequently if the batches are formed the makespan of any batch sequence is equal to the sum of $a_i, i = 1, \ldots, n$ plus the duration of the largest batch. Hence to solve problem $F2|\text{sum-batch}, c\text{-in-1}, a_i = b_i|C_{\max}$, we only have to constitute the batches such that the duration of the largest one is minimum.

We use a polynomial transformation from the 3-Partition problem: Given $3m+1$ positive integers $c_1, \ldots, c_{3m}$ and C such that $C/4 < c_j < C/2$, $j = 1, \ldots, 3m$, and $\sum_{j=1}^{3m} c_j = mC$, is there a partition of the set $\{1, \ldots, 3m\}$ into m subsets $X_1, \ldots, X_m$, for which $\sum_{j \in X_l} c_j = C$, $l = 1, \ldots, m$? Given an instance of this problem, construct the following instance of our problem. There are $3m$ jobs i such that $a_i = b_i = c_i$. We show that 3-partition problem has a solution if, and only if, there exists a solution to the constructed instance such that $\#batch = m$ and $C_{\max} \leq y := (m+1)C$.

Suppose that 3-partition problem has a solution $X = \{X_1, \ldots, X_m\}$. Constitute m batches according to solution X. The makespan of any batch sequence is equal to $\sum_{j=1,\ldots,m} X_j + \max_{j=1,\ldots,m} X_j = mC + C = (m+1)C$.

Assume now that there exists a schedule such that $\#batch = m$ and $C_{\max} \leq (m+1)C$. Let $X_1, \ldots, X_m$ be the duration of the corresponding batches. We have $C_{\max} = \sum_{j=1,\ldots,m} X_j + \max_{j=1,\ldots,m} X_j = mC + \max_{j=1,\ldots,m} X_j \leq (m+1)C$. Then $\max_{j=1,\ldots,m} X_j = C$ and $X_1 = \ldots = X_m = C$, which means that $X = \{X_1, \ldots, X_m\}$ is solution to 3-partition problem. □

However, problem $F2|\text{sum-batch}, \text{2-in-1}|C_{\max}$ is open.

3 Polynomial-time cases

Solving the batching problem for a given job sequence: Given a job sequence π, we propose a polynomial-time dynamic-programming algorithm, named $DP1(\pi)$, allowing us to solve the bicriteria problem $F2|\text{sum-batch}, c|(\#batch, C_{\max})$, i.e. determine the Pareto optimal solutions set $\mathcal{E}_\pi$ subject to the constraint that the jobs follow sequence π. We first renumber the jobs according to sequence π. For each partial solution in which we have already scheduled i jobs, we associate a $state(i,k,q)$, where k is the total number of jobs in the last batch and q is the number of batches. Let $C(i,k,q)$ denote the minimum makespan among all partial solutions corresponding to the same state $state(i,k,q)$. Algorithm $DP1(\pi)$ recursively calculates values $C(.)$. The initialization is given by :

$C(i,k,q) = \infty$ $(i = 0, ..., n;\ k = 0, ..., n,\ q = 0, ..., n.)$ and $C(0,0,0) = 0$.

The recursion for $i = 1, ..., n$, $k = 1, ..., i$ and $q = 1, ..., i$ is given by:

$$C(i,k,q) = \begin{cases} \infty, & \text{if } k > c, \\ \max(C(i-1,k-1,q) - \sum_{j=1}^{k-1} b_{i-j}, \sum_{j=1}^{i} a_j) + \sum_{j=0}^{k-1} b_{i-j}, & \text{if } 1 < k \leq c, \\ \min_{p=0,\ldots,i-1}\{\max(C(i-1,p,q-1), \sum_{j=1}^{i} a_j)) + b_i\}, & \text{if } k = 1. \end{cases}$$

In order to enumerate all Pareto optimal solutions following the given sequence, we determine, for each value $q \in \{1, ..., n\}$, the optimal makespan, which is given by $\min_{k=1,\ldots,n} C(n,k,q)$. The corresponding solutions are obtained by backtracking. The running time of algorithm $DP1(\pi)$ can be evaluated as $O(n^3)$.

Equal processing times and batches of size two: We consider here the problem $F2|\text{sum-batch}, 2-in-1, a_i = b_i|C_{\max}$. First renumber the jobs such that $a_1 \leq a_2 \leq ... \leq a_n$. We have the following results.

Lemma 1 *If n is even, then there exists an optimal solution for problem $F2|$sum-batch, 2-in-1$, a_i = b_i|C_{\max}$ in which jobs i and $(n-i+1)$, for $i = 1, \ldots, n/2$, are in the same batch.*

Proof : Consider an optimal solution π characterized by batches $B_1, \ldots, B_h$ (with $h = n/2$). Suppose that jobs 1 and n are not in the same batch, i.e. job 1 is with some job j_1 in a batch B_k and job n is with another job j_2 in a batch B_l. Construct a new solution π' in which jobs 1 and n are in the same batch B'_k, jobs j_1 and j_2 are in the same batch B'_l, and the other batches are the same as in π. Denote by $p(B_r) = \sum_{i \in B_r} a_i, r = 1, \ldots, h$. We have $p(B'_k) = a_1 + a_n \leq a_{j_2} + a_n = p(B_l)$ and $p(B'_l) = a_{j_1} + a_{j_2} \leq a_n + a_{j_2} = p(B_l)$. Consequently,

$\max_{r=1,\ldots,h} p(B'_r) \leq \max_{r=1,\ldots,h} p(B_r)$ and we have $C_{\max}(\pi') \leq C_{\max}(\pi)$, which means that π' is also optimal. By repeating the same procedure at most $n/2$ times, we obtain a solution satisfying the property. □

If the number of jobs is odd, we can easily prove that job n is alone on a batch and jobs i and $(n-i)$, for $i = 1, \ldots, \lfloor n/2 \rfloor$, are in the same batch. Hence, we get the following result.

Theorem 2 *Problem $F2|$sum-batch, 2-in-1$, a_i = b_i|C_{\max}$ can be solved in $O(n \log n)$ time.*

As a consequence, problem $F2|\text{sum-batch}, c = 2, a_i = b_i|(\#batch, C_{\max})$ can be solved in $O(n^2)$ time by adding some dummy jobs and solve $O(n)$ 2-in-1 problems. Remark that imposing a number of batches $\#b, n/2 \leq \#b \leq n$, implies that there are $(2 \times \#b - n)$ batches with one job each and $(n - \#b)$ batches containing each 2 jobs. In an optimal solution, the latter batches will contain the jobs with largest processing time, i.e. jobs $(2n-2\#b+1), \ldots, n$, and the former batches are constructed according to lemma 1. Consequently, we have the following result.

Corollary 1 *The bicriteria problem $F2|$sum-batch$, c = 2, a_i = b_i|(\#batch, C_{\max})$ can be solved in $O(n^2)$ time.*

Constant processing time for the first or the second machine: We can prove the following result by the job interchange argument.

Lemma 2 *Any Pareto optimal solution of problem $F2|$sum-batch$, c, a_i = a|(\#batch, C_{\max})$ can be transformed into a solution with the same performance measure values such that the jobs are processed in LPT order with respect to their processing times on machine M_2.*

Using algorithm $DP1(\pi^{LPT(M_2)})$, we get the following result.

Theorem 3 *$F2|$sum-batch$, c, a_i = a|(\#batch, C_{\max})$ can be solved in $O(n^3)$ time.*

Similarly, we can prove that any Pareto optimal solution of problem $F2|\text{sum-batch}, c, b_i = b|(\#batch, C_{\max})$ can be transformed into a solution with the same performance measure values such that the jobs are processed in SPT order with respect to their processing times on machine M_1. Using algorithm $DP1(\pi^{SPT(M_1)})$ allows to solve the problem.

Theorem 4 *$F2|$sum-batch$, c, b_i = b|(\#batch, C_{\max})$ can be solved in $O(n^3)$ time.*

4 A polynomial-time approximation algorithm

We consider here that the number of jobs n is such that $n = ch, h > 0$. Otherwise, we add $(ch - \lfloor n/c \rfloor)$ dummy jobs with zero processing times in both machines. We propose an approximation algorithm, named $A(c)$, to solve problem $F2|\text{sum-batch}, c\text{-in-1}|C_{\max}$ in polynomial time when c is constant. In this algorithm, batches are formed according to Johnson sequence. When the condition in instruction 6 is not verified, and when $h \geq 2c - 2$, jobs of batch B_k are separated and mixed with the jobs of the following (or previous) $c - 1$ batches when $k \leq c - 1$ (or $k \geq c$) (see instructions 10-17). We have the following result.

Algorithm 1: Algorithm A(c)

Number the jobs according to Johnson sequence $\pi^J = (1, 2, \dots ch)$
Compute $C^J_{\max}$ the optimal makespan of the $F2||C_{\max}$ problem
Group jobs into h batches of c jobs according to sequence π^J to form solution $S = (B_1, \dots, B_h)$
Compute the makespan $C_{\max}(S)$ of solution S
Identify a batch B_k such that $\sum_{i=1}^{ck} a_i + \sum_{i=c(k-1)+1}^{ch} b_i = C_{\max}(S)$
if $\min_{0\leq t\leq c-1}\{\sum_{i=c(k-1)+2+t}^{ck} a_i + \sum_{i=c(k-1)+1}^{c(k-1)+t} b_i\} \leq \frac{C^J_{\max}}{2}$ **then**
    Return solution S
else
    if $h \geq 2c-2$ **then**
        if $1 \leq k \leq c-1$ **then**
            for $r \leftarrow 0$ *to* $c-1$ **do**
                Replace batch B_{k+r} by batch $B'_{k+r} = (c(k-1)+1+r, i^2_r, \dots, i^c_r)$ where $i^l_r = c(k+r) - r - 1 + l, l = 2, \dots, c$
            Return the resulting solution $S' = (B_1, \dots, B_{k-1}, B'_k, \dots, B'_{k+c-1}, B_{k+c}, \dots, B_h)$
        else
            for $r \leftarrow 0$ *to* $c-1$ **do**
                Replace batch $B_{k-c+1+r}$ by batch $B''_{k-c+1+r} = (c(k-1)+1+r, j^2_r, \dots, j^c_r)$ where $j^l_r = c(k-c+r) - r - 1 + l, l = 2, \dots, c$
            Return the resulting solution $S'' = (B_1, \dots, B_{k-c}, B''_{k-c+1}, \dots, B''_k, B_{k+1}, \dots, B_h)$
    else
        Select and return a best solution among all possible solutions

Theorem 5 *Algorithm $A(c)$ is a polynomial-time approximation algorithm for problem $F2|$sum-batch, c-in-1$|C_{\max}$ with tight approximation ratio $\rho = \frac{3}{2}$ when c is constant.*

Proof : Denote by $C^{*b}_{\max}$ the optimal makespan for problem $F2|$sum-batch, c-in-1$|C_{\max}$.

In algorithm $A(c)$ batch B_k defines $C_{\max}(S)$, then we have $C_{\max}(S) = \sum_{i=1}^{ck} a_i + \sum_{i=c(k-1)+1}^{ch} b_i$ that can be rewritten, for all $t = 0, \dots, c-1$,

$$C_{\max}(S) = \sum_{i=1}^{c(k-1)+1+t} a_i + \sum_{i=c(k-1)+1+t}^{ch} b_i + \sum_{i=c(k-1)+2+t}^{ck} a_i + \sum_{i=c(k-1)+1}^{c(k-1)+t} b_i. \quad (1)$$

Hence, we have

$$C_{\max}(S) \leq C^J_{\max} + \min_{t=0,\dots,c-1}\{\sum_{i=c(k-1)+2+t}^{ck} a_i + \sum_{i=c(k-1)+1}^{c(k-1)+t} b_i\}. \quad (2)$$

Consequently, if

$$\min_{t=0,\ldots,c-1}\{\sum_{i=c(k-1)+2+t}^{ck} a_i + \sum_{i=c(k-1)+1}^{c(k-1)+t} b_i\} \leq \frac{1}{2}C^J_{\max},$$

then $C_{\max}(S) \leq \frac{3}{2}C^J_{\max} \leq \frac{3}{2}C^{*b}_{\max}$. (instructions 6 and 7 in the algorithm).

Otherwise, since $C^J_{\max} \geq \sum_{i=1}^{l} a_i + \sum_{i=l}^{ch} b_i, l = 1, \ldots, ch$, we get

$$\forall l = 1, \ldots, c(k-1)+1, ck, \ldots, ch, \quad \sum_{\substack{i=1 \\ i \notin B_k \setminus \{c(k-1)+1\}}}^{l} a_i + \sum_{\substack{i=l \\ i \notin B_k \setminus \{ck\}}}^{ch} b_i < \frac{1}{2}C^J_{\max}. \tag{3}$$

We distinguish two cases:

1. Case where $1 \leq k \leq c-1$. Algorithm $A(c)$ provides solution S' (see instructions 11-13). Let B'_l be the batch determining the makespan of solution S'. Then:

   - if $l < k$ or $l \geq k + c$ then

   $$\begin{aligned} C_{\max}(S') &= \sum_{i=1}^{cl} a_i + \sum_{i=c(l-1)+1}^{ch} b_i \\ &= \sum_{i=1}^{cl} a_i + \sum_{i=cl}^{ch} b_i + \sum_{i=c(l-1)+1}^{cl-1} b_i \end{aligned}$$

   According to equations (3), we have $\sum_{i=c(l-1)+1}^{cl-1} b_i < \frac{1}{2}C^J_{\max}$. Then, $C_{\max}(S') \leq \frac{3}{2}C^J_{\max} \leq \frac{3}{2}C^{*b}_{\max}$.

   - if $k \leq l \leq k + c - 1$, let $l = k + r, 0 \leq r \leq c - 1$. We have

   $$\begin{aligned} C_{\max}(S') = \sum_{i=1}^{c(k-1)} a_i &+ a_{c(k-1)+1} + a_{i_0^2} + \ldots + a_{i_0^c} \\ &+ \ldots \\ &+ a_{c(k-1)+1+r} + a_{i_r^2} + \ldots + a_{i_r^c} \\ &+ b_{c(k-1)+1+r} + b_{i_r^2} + \ldots + b_{i_r^c} \\ &+ \ldots \\ &+ b_{ck} + b_{i^2_{c-1}} + \ldots + b_{i^c_{c-1}} \\ &+ \sum_{i=c(k+c-1)+1}^{ch} b_i \end{aligned}$$

   Hence, we have the following

   $$C_{\max}(S') = \sum_{i=1}^{c(k-1)+1+r} a_i + \sum_{i=i_0^2}^{i_r^c} a_i + \sum_{i=c(k-1)+1+r}^{ch} b_i - \sum_{i=i_0^2}^{i^c_{r-1}} b_i. \tag{4}$$

According to equations (3), we have $\sum_{i=i_0^2}^{i_r^c} a_i < \frac{1}{2}C^J_{\max}$. Then, $C_{\max}(S') \leq \frac{3}{2}C^J_{\max} \leq \frac{3}{2}C^{*b}_{\max}$.

2. Case where $k \geq c$. Algorithm $A(c)$ provides solution S'' (see instructions 15-17) Let B''_l be the batch determining the makespan of solution S''. Then:

   - if $l \leq k-c$ or $l > k$ then

$$\begin{aligned} C_{\max}(S'') &= \sum_{i=1}^{cl} a_i + \sum_{i=c(l-1)+1}^{ch} b_i \\ &= \sum_{i=1}^{cl} a_i + \sum_{i=cl}^{ch} b_i + \sum_{i=c(l-1)+1}^{cl-1} b_i \end{aligned}$$

   According to equations (3), we have $\sum_{i=c(l-1)+1}^{cl-1} b_i < \frac{1}{2}C^J_{\max}$. Then, $C_{\max}(S'') \leq \frac{3}{2}C^J_{\max} \leq \frac{3}{2}C^{*b}_{\max}$.

   - if $k-c+1 \leq l \leq k$, let $l = k-c+1+r, 0 \leq r \leq c-1$. We have

$$\begin{aligned} C_{\max}(S'') = \sum_{i=1}^{c(k-c)} a_i &+ a_{c(k-1)+1} + a_{j_0^2} + \ldots + a_{j_0^c} \\ &+ \ldots \\ &+ a_{c(k-1)+1+r} + a_{j_r^2} + \ldots + a_{j_r^c} \\ &+ b_{c(k-1)+1+r} + b_{j_r^2} + \ldots + b_{j_r^c} \\ &+ \ldots \\ &+ b_{ck} + b_{j_{c-1}^2} + \ldots + b_{j_{c-1}^c} \\ &+ \sum_{i=ck+1}^{ch} b_i \end{aligned}$$

   Hence, we have the following

$$C_{\max}(S'') = \sum_{i=1}^{c(k-1)+1+r} a_i + \sum_{i=c(k-1)+1+r}^{ch} b_i + \sum_{i=j_r^2}^{j_{c-1}^c} b_i - \sum_{i=j_{r+1}^2}^{j_{c-1}^c} a_i. \qquad (5)$$

   According to equations (3), we have $\sum_{i=j_r^2}^{j_{c-1}^c} b_i < \frac{1}{2}C^J_{\max}$. Then, $C_{\max}(S'') \leq \frac{3}{2}C^J_{\max} \leq \frac{3}{2}C^{*b}_{\max}$.

If $h < 2c-2$, then we select a best solution, with makespan equal to $C^{*b}_{\max}$ among all possible solutions (instruction 19 in the algorithm). The number of all possible solutions is less than $(2c(c-1))!$ that is polynomial since c is constant.

To summarize, in all cases algorithm $A(c)$ provides a solution with a makespan less or equal to $\frac{3}{2}C^{*b}_{\max}$.

Table 1: An example with a ratio equal to 3/2

	i	a_i	b_i
$a_i \leq b_i$	1	1	$4M$
	$i = 2, \ldots, c-1$	i	$i+1$
$a_i > b_i$	c	$2M$	$c+1$
	$c+1$	$2M$	c
	$i = c+2, \ldots, 2c$	$2c-i+2$	$2c-i+1$

In order to prove that the approximation ratio is tight, consider the following instance composed of $n = 2c$ jobs and described in table 1. Johnson order is $\pi^J = (1, 2, \ldots, 2c)$ and $C^J_{\max} = 4M + (c+1)^2 - 2$. Algorithm $A(2)$ provides a solution $S = (B_1, B_2)$ such that $B_1 = (1, \ldots, c)$ and $B_2 = (c+1, \ldots, 2c)$. We have $C_{\max}(S) = 6M + (c+1)^2 + \frac{c(c-1)}{2} - 3$. The optimal solution $S^* = (B^*_1, B^*_2)$ is such that batch $B^*_1 = (1, \ldots, c-1, 2c)$ and $B^*_2 = (c, c+1, \ldots, 2c-1)$. We have $C^{*b}_{\max} = 4M + (c+1)^2 + \frac{c(c-1)}{2} - 1$. Consequently, we get:

$$\frac{C_{\max}(S)}{C^{*b}_{\max}} = \frac{6M + (c+1)^2 + \frac{c(c-1)}{2} - 3}{4M + (c+1)^2 + \frac{c(c-1)}{2} - 1} \longrightarrow \frac{3}{2} \text{ when } M \text{ tends to } \infty$$

□

5 Future research

The following questions are yet to be answered: (1) What is the complexity of problem $F2|\text{sum-batch}, 2\text{-in-}1|C_{\max}$? (2) Can we extend algorithm $A(c)$ when c is part of the problem input ? (3) In practice, is it difficult to solve this strong NP-hard problem ?

References

[1] Esswein C., J.C. Billaut and V.A. Strusevich, 2005 "Two machine shop scheduling : compromise between flexibility and makespan value", *European Journal of Operational Research*, Vol. 167(3), pp. 796-809.

[2] Glass C.A., C.N. Potts and V.A. Strusevich, 2001 "Scheduling batches with sequential job processing for two-machine flow and open shops", *INFORMS Journal on Computing*, Vol. 13(2), pp. 120-137.

[3] Yuan J.J., Y.X. Lin , T.C.E. Cheng and C.T. Ng , 2007 "Single machine serial-batching scheduling problem with a common batch size to minimize total weighted completion time", *International Journal of Production Economics*, Vol. 105(2), pp. 402-406.