Scheduling chains of operations on a batching machine with disjoint sets of operation compatibility

Nadia Brauner, Guyslain Naves

Scheduling chains of operations on a batching machine with disjoint sets of operation compatibility

Nadia Brauner, Guyslain Naves

March 17, 2008

Abstract

We consider a scheduling problem that arises from an industrial application in chemical experimentations, where a single machine can process a fixed number of compatible jobs simultaneously. The precedence graph is restricted to be a disjoint union of chains, and the compatibility constraints are given by a partition of the tasks. Nevertheless, with these restrictions we prove the NP-completeness of the problem when the machine has a capacity of two, implying the difficulties for greater capacities. We also present a short proof for an infinite capacity. Our results also show the NP-completeness of the D-Supersequence problem, even when there are only two kinds of strings. We show polynomiality results when the number of chains in the precedence graph is fixed or when each chain has only two jobs.

1 A scheduling problem

This study started with an industrial project that we carried out with the *Institut Français du Pétrole* (IFP) (see [6] for a detailed description). Lengthy and costly chemical experiments had to be conducted. The problems of scheduling those experiments had many features that could be addressed by the classical scheduling approaches, for instance, scheduling with machine unavailability or parallel machine scheduling minimizing the total processing time. But we were also faced with a new aspect.

At an intermediate step of the experimentations, we were confronted with the scheduling of chains of operations coming from the different experimentations: we have a set of tasks T_i given as a sequence of ordered experiments or operations of various lengths (*i.e.*, the precedence graph is a chain). The machines are so-called *batch machines*, *i.e.*, machines that can handle simultaneously several operations [1]. We call operations *batch-compatible* if they can be put in the same batch. Moreover, we consider that the batch compatibility constraint partitions the operations in disjoint sets. Indeed, in our application, two operations are batch-compatible if they have exactly the same length which is then also the length of the batch. So in the scheduling context, the problem is to schedule operations on a batch machine with precedence constraints as chains and disjoint sets of batch compatibility. The objective is to minimize the makespan, *i.e.*, to finish the most rapidly all operations.

Example 1 The following example is composed of two tasks with respective sequences of duration of the operations (the arrows indicate the precedence constraints):

$$T_1 : 21 \to 5 \to 14 \quad \text{and} \quad T_2 : 5 \to 14 \to 21$$

A possible schedule of length 61 on a single batch machine with capacity 2 is:

T_1^{21}	$T_1^5\ T_2^5$	$T_1^{14}\ T_2^{14}$	T_2^{21}

where the operations of length 5 and 14 are grouped in batches and the operations of length 21 are scheduled alone. An optimal schedule of length 59 is:

T_2^5	T_2^{14}	$T_1^{21}\ T_2^{21}$	T_1^5	T_1^{14}

$\square$

In the classical classification of scheduling problems [4] as (resource, task, objective), our problem can be described as follows:

- m parallel batch-machines M_j of capacity B_j (p-batch machine, *i.e.* the duration of the batch is the length of the largest operation);
- n tasks T_i composed of sets of operations with precedence graphs as chains (each task is a chain) and batch compatibility constraints on the operations (as disjoint sets of operations);
- the objective is $Cmax$.

A survey on scheduling problems with batches can be found in [8]. In this paper, we consider a single batch machine ($m = 1$) of capacity B. Some complexity results for this problem are given in [2]. Following the classical scheduling notations, we denote it by $B1|chains; comp; B > n|Cmax$ for the infinite-capacity case and $B1|chains; comp; B = c|Cmax$ for the fixed-capacity case. In Section 2 we study the case $B = \infty$ and Section 3 deals with $B = 2$.

There is an interesting analogy of this type of problems with the minimal common supersequence problems and with problems in genetics (sequence alignment) where we have the same structure [5]. This analogy helps propose solution methods and complexity results. Task T_i can be defined as a string (or a chain) on the alphabet Σ of the operations, $w : \Sigma \to \mathbb{N}$ gives the duration of each operation, and two operations correspond to the same letter if they can be put into the same batch. T_i^j denotes the j^{th} operation of task T_i.

Theorem 1 *For a fixed number of tasks n, the scheduling problem is polynomial.*

Proof. We note $\text{pref}_j(T_i)$ the sequence of the first j operations of T_i, for all i and $j \leq |T_i|$.
We use a simple dynamic programming algorithm. Let $\text{Opt}[j_1, \dots, j_n]$, with $j_i \leq |T_i|$ for all i, denote the optimal solution for the problem $\text{pref}_{j_1}(T_1), \dots, \text{pref}_{j_n}(T_n)$. We give an inductive definition of $\text{Opt}[j_1, \dots, j_n]$ for all values of $j_1, \dots, j_n$, with the order for the induction being $a_1, \dots, a_n \leq b_1, \dots, b_n$ if and only if $a_1 \leq b_1, \dots, a_n \leq b_n$. For all $a \in \Sigma$, we say that $u \in 2^{[1,n]}$ is a-compatible with $(j_1, \dots, j_n)$ if

$$\forall i \in [1, n], u_i = 1 \Rightarrow (j_i \neq 0 \wedge T_i^{j_i} = a)$$

and $1 \leq \sum_{i=1}^n u_i \leq B$. Then the inductive equations are:

$$\text{Opt}[0, \dots, 0] = 0$$

$$\text{Opt}[j_1, \dots, j_n] = \min_{\substack{a \in \Sigma \\ u\ a\text{-compatible with} \\ (j_1, \dots, j_n)}} w(a) + \text{Opt}(j_1 - u_1, \dots, j_n - u_n)$$

Thus there exists a polynomial-time algorithm that solves this problem. $\square$

2 Infinite-capacity batch machine

This section deals with the case $B = \infty$. Note that if each letter appears at most B times then, we can consider that $B = \infty$. Consider the following problem from computer science theory:

Minimal Common Supersequence
Instance. Finite alphabet Σ where each letter has a positive weight, finite set R of strings from Σ^* and a positive integer K.
Question. Is there a string $w \in \Sigma^*$ with total weight less than K (where the weight of a string is the sum of the weights of its letters) and each string $x \in R$ is a subsequence of w, *i.e.* one can get x by taking away letters from w.

This decision problem is equivalent to the decision version of the infinite capacity batch machine problem we consider: as mentioned before, the operations are the letters of Σ, the weight of a letter is the duration of the corresponding operation and the tasks are the strings of R. The solution w gives a feasible scheduling of the operations and its weight is the makespan.

Example 1 (Cont.)

$$T_1 = abc, \qquad T_2 = bca, \qquad w(a) = 21, \qquad w(b) = 5, \qquad w(c) = 14$$

This problem is also equivalent to the classical sequence alignment problem in genetics with the score S of an alignment at a position defined as:

$$S(\overbrace{p, p \dots p}^{\alpha}, \overbrace{-, - \dots -}^{n-\alpha}) = (\alpha - 1)p \qquad\qquad \text{with } p \in \sigma$$

See [5] for more details on this problem.

Theorem 1 states that this problem is solvable in polynomial time if the number of sequences $|R|$ is a constant. If all weights are equal to 1, this problem is known as the Shortest Common Supersequence (problem [SR8] in [3]). It is NP-complete even for two-letter alphabets (*i.e.* $|\Sigma| = 2$) [9], or if all strings of R are of length 2. This last result contradicts [3] and therefore, we prove it in the following Proposition (also proved in [10]).

Proposition 1 *The* Shortest Common Supersequence *problem is NP-complete even if all chains are of length 2, i.e.,* $\forall x \in R$*, one has* $|x| = 2$*.*

Proof. We reduce from Stable Set. Let $G = (V, E)$ be an undirected graph, we build the set of words upon the alphabet V defined by $\{uv, vu \mid \{u, v\} \in E\}$. Then, it is easy to see that the length of the shortest supersequence is $2|V| - \alpha(G)$, with $\alpha(G)$ the maximum cardinality of the stable sets of G (the vertices of the stable set correspond exactly to the letters that are scheduled in a single batch). □

The reduction can be improved in such a way that each word is of length at most 2 and every letter appears at most three times [10].

3 Two-capacity machine

In this section, we consider the case $B = 2$. Figure 1 is an example of four chains or tasks for the case where each letter has a weight of 1 and the letters indicate the compatibility constraint. For this example, an optimal solution is *cbdebcde*.

T_1 :	b	c	d	e
T_2 :	c	b	e	d
T_3 :	b	d	c	e
T_4 :	c	d	e	b

the tasks

				b	c	d	e
c	b		e			d	
	b	d			c		e
c		d	e	b			

an optimal solution

Figure 1: An example with 4 chains.

We shall consider several interesting sub-problems that are related to problems already studied in the literature: one chain contains half of the letters (section 3.1), each letter appears at most once in a chain (section 3.2) and the length of the chains are bounded (section 3.3).

3.1 Disjoint supersequences

In this section, we study the D-Supersequence problem which is a special case of $B1|chains; comp; B = 2|Cmax$ where one of the chains contains half of all the operations and all durations are equal to 1.

The following notations are from [7]. Consider two strings $T = (t_1 t_2 \dots t_t)$ and $S = (s_1 s_2 \dots s_s)$ on a same alphabet Σ. An *embedding* f of T in S is a strong growing function from $[1, t]$ to $[1, s]$ such that t_i and $s_{f(i)}$ are the same letter for all $i \in [1, t]$. Let $R = \{S_1, S_2, \dots S_k\}$ be a set (or a multiset) of strings. An embedding of R in S is a k-tuple $(f_1, f_2 \dots f_k)$ where f_i is an embedding of S_i in S. An embedding is *disjoint* if, for all $i \neq j \in [1, k]$ and $l_i \in [1, |S_i|]$ and $l_j \in [1, |S_j|]$, we have $f_i(l_i) \neq f_j(l_j)$. A string S is a *d-supersequence* of a set of strings R if there exists a disjoint embedding of R in S. Consider the following decision problem:

D-Supersequence
Instance. A string S over a finite alphabet Σ and a set R of strings over the same alphabet Σ.
Question. Is S a d-supersequence of R ?

As mentioned earlier, this problem is a special case of the 2-capacity scheduling problem we consider. This problem is easier than our scheduling problem, for we only want to know if there is a "perfect" scheduling and not a scheduling that is better than the objective. We shall give negative results even for a two letter alphabet using a preliminary lemma.

Lemma 1 d-supersequence *is NP-complete.*

Proof. We reduce from 3-Sat. Let $\mathcal{C} = \{C_1, \dots, C_p\}$ be a set of clauses of size 3 over variables in $X = \{x_1, \dots, x_n\}$. We can suppose that each variable appears exactly three times in $\mathcal{C}$, two times positively and one time negatively. The set of letters is $\Sigma = \{b_1, \dots, b_n\} \cup \{c_1, \dots, c_p\}$. For each $x_i \in X$, there are $j < k$ and l such that $x_i \in C_j$, $x_i \in C_k$ and $\overline{x_i} \in C_l$, we note $c_i^{1+} := c_j$ and $c_i^{2+} := c_k$, and $c_i^{-} := c_l$.
For each variable, we build three words, all these words giving R:

$$\begin{aligned} r_{3i-2} &:= b_i c_i^{1+} c_i^{2+} \\ r_{3i-1} &:= b_i c_i^{2+} \\ r_{3i} &:= b_i c_i^{-} \end{aligned}$$

We define S:

$$b_1 b_2 \dots b_n c_1 c_2 \dots c_p b_1 b_1 b_2 b_2 \dots b_n b_n c_1 \dots c_1 c_2 \dots c_2 \dots c_p \dots c_p$$

We use as many c_i in S as there are in R. Now, we prove that $\mathcal{C}$ is satisfiable if and only if S is a d-supersequence of R.
Suppose that $\mathcal{C}$ is satisfiable. We take a solution, and for each clause C_k, choose a variable that makes the clause satisfied. Then, for each variable x_i, there are five cases:

- either x_i was taken twice (thus positively), then we define $f_{3i-2}(1) = i$, $f_{3i-2}(2)$ is the index of the first c_i^{1+} in S and $f_{3i-2}(3)$ of the first c_i^{2+},
- or x_i was taken once, for the clause encoded by c_i^{1+}, then $f_{3i-2}(1) = i$ and $f_{3i-2}(2)$ is the index of the first c_i^{1+} in S,
- or x_i was taken once, for the clause encoded by c_i^{2+}, then $f_{3i-1}(1) = i$ and $f_{3i-1}(2)$ is the index of the first c_i^{2+},
- or x_i was taken once, for the clause encoded by c_i^-, then $f_{3i}(1) = i$ and $f_{3i-1}(2)$ is the index of the first c_i^-,
- or x_i was not taken, $f_{3i-2}(1) = i$, and we do nothing.

It is straightforward to construct the rest of the solution, as the first $n+p$ letters of S are already embedded, and the others do not create any difficulty.
Reciprocally, a solution of the d-supersequence problem gives us a solution for the satisfiability problem: if $f_{3i-2}(1) \leq n$ or $f_{3i-1}(1) \leq n$ then set x_i to $true$, else to $false$, and each clause will be satisfied. This Karp reduction can be computed in polynomial time, and the problem is trivially in NP, thus NP-complete. □

Lemma 2 D-SUPERSEQUENCE *is NP-complete, even if* $R = \{S_1, S_2, \ldots S_k\}$ *contains only one single chain with multiplicity* k, i.e. $S_1 = S_2 = \ldots S_k$.

Proof. We reduce from D-SUPERSEQUENCE. Let S, $R = \{S_1, S_2, \ldots S_k\}$ be an instance of D-SUPERSEQUENCE. We introduce k new letters $\alpha_0, \alpha_1, \ldots, \alpha_{k-1}$. We define a new instance (S', S_R) of our problem by (where S_R is taken with multiplicity k):

$$B = \alpha_0^k (S_1.\alpha_1)^{k-1}.(S_2.\alpha_2)^{k-2} \ldots (S_{k-1}.\alpha_{k-1})^1$$

$$E = (\alpha_1.S_2)^1.(\alpha_2.S_3)^2 \ldots (\alpha_{k-1}.S_k)^{k-1}$$

$$S' = B.S.E$$

$$S_R = \alpha_0.S_1.\alpha_1.S_2.\alpha_2 \ldots S_k$$

We note that the letters of B must be matched with α_0 in the first word S_R, $\alpha_0.S_1.\alpha_1$ in the second, $\ldots \alpha_0.S_1.\alpha_1 \ldots \alpha_{k-1}$ in the last word, up to a permutation of the words, as the α_i are new letters. Moreover, the letters of E are matched with $\alpha_1.S_2 \ldots \alpha_k$ in the first word, $\ldots$, α_k in the last word. Thus, the letters of S must be matched with the word $S_1, \ldots S_k$, proving the reduction. □

The following theorem proves that, even for a 2-letter alphabet, deciding whether a string is a disjoint supersequence of k times another one is an NP-complete problem.

Theorem 2 D-SUPERSEQUENCE *is NP-complete even if* $|\Sigma| = 2$ *and* $R = \{S_1, S_2, \ldots S_k\}$ *contains only one single chain with multiplicity* k

Proof. We reduce from the preceding problem (Lemma 2): we give an encoding of the letters of $\Sigma = \{a_0, \dots, a_n\}$. let $\varphi : \Sigma^\star \to \{0,1\}^\star$ be the morphism defined by $\varphi(a_i) = 01^i01^{2n+1}$. It can be computed in polynomial time, thus we only have to show that this encoding φ preserves the existence of a solution. Trivially, if there is a solution for the initial problem, there is one for the encoding. Reciprocally, it suffices to show that the encoding of the first letter of S is embedded exactly by the encoding of a letter of R. To see this, remark that before reading the encoding of the second letter of S, we need to read at least $2n + 1$ times '1' but only twice '0', and this is possible only by reading twice '0' on the same word of $\varphi(R)$. Thus the encoding is correct. □

3.2 Each letter at most once in a chain

In the practical problem of scheduling experiments, each duration appears at most once for each task. In this section, we consider the corresponding case where a letter cannot appear more than once in each chain.

Theorem 3 *The scheduling problem $B1|chains; comp; B = 2|Cmax$ is NP-complete even if each type of operation appears at most once in each chain (all letters are different in one chain).*

Proof. We reduce from the general case, which is NP-complete (Theorem 2). As long as there is a chain C with a letter, say a, that appears at least twice, we do the following operation: replace the second a of C by $\alpha\beta$, where α and β are two new letters, and add a word $\alpha a \beta$. This is trivially correct. It gives a polynomial Karp reduction, and the problem being in NP, the proof is complete. □

We now consider the case where the number of letters, $|\Sigma|$, is a constant. For this case, the total number of different chains is fixed. Therefore, an instance can be described by the 'types' of chain and a multiplicity for each type. We prove that this problem is easily solvable using the following lemma.

Lemma 3 *Let I be a multiset of n words $w_1, \dots, w_n \in \Sigma^\star$ with multiplicity $i_1, \dots i_n$ respectively. We note $|\Sigma| = m$. Suppose that each letter appears at most once in each word of I. If there is some $1 \le k \le n$ such that $i_k \ge (3m.m!)^m$, then the minimum number of letters that are not matched on the scheduling of I is equal to this minimum on the scheduling of $I \cup \{w_k, w_k\}$.*

Proof. First, we show that the maximum number of distinct words in I is less than $3m!$. Indeed, the number of words w of length j such that each letter appears at most once in w is exactly $\binom{m}{j}j!$. Thus, the number of different words in I verifies

$$n \le \sum_{j=1}^{m} \binom{m}{j} j! = \sum_{j=1}^{m} \frac{m!}{(m-j)!} \le m!e < 3m!$$

Note that it is easy to match all the letters of two identical words. We prove the lemma by induction on m. If $m < 2$, it is obvious.

Let m be an integer, suppose the lemma is true for all ranks $m' < m$. We prove it for rank m. Let I be a multiset as in the lemma, w.l.o.g. $k = 1$. Let $\Sigma_1 = \{a \in \Sigma : a \in w_1\}$ and $\overline{\Sigma_1} = \{a \in \Sigma : a \notin w_1\}$, and $m' = |\overline{\Sigma_1}| \le m - 1$. We define a morphism φ upon words by $\varphi(a) = a$ if $a \in \overline{\Sigma_1}$, $\varphi(a) = \epsilon$ otherwise, *i.e.* it erases all letters of Σ_1. For all $j \in 2 \dots n$, we pose $i'_j = \min(\{i_j\} \cup \{l \in \mathbb{N} : l \equiv i_j[2] \wedge l \ge (3m'.m'!)^{m'}\})$, *i.e.* if i_j is greater than $(3m'.m'!)^{m'}$, then i'_j is the smallest integer having the same parity as i_j and still greater than $(3m'.m'!)^{m'}$.

Let I' be the multiset of words $w_2, \dots, w_n$ with multiplicity $i'_2, \dots i'_n$ respectively, and I'' the multiset of the same words with multiplicity $i_2, \dots i_n$. By induction hypothesis, the minimum number of letters of $\overline{\Sigma}_1$ not matched on the scheduling of $\varphi(I')$ is equal to this minimum on $\varphi(I'')$. Let S be an optimal scheduling for $\varphi(I')$. We can interpret S as a scheduling upon I' that minimizes the number of letters of $\overline{\Sigma}_1$ that are not matched (the letters of Σ_1 are not matched). Now, we can match the words of $I'' \setminus I'$ perfectly, as i_j and i'_j have the same parity for all j. Thus, we obtain a scheduling S' of the words $w_2, \ldots, w_n$ of I that minimizes the number of letters of $\overline{\Sigma}_1$ that are not matched. Moreover, for each letter $a \in \Sigma_1$, a appears at most $\sum_{j=2}^{n} i'_j \leq (n-1)(3(m-1)(m-1)!)^{m-1}$ times not matched.

Now, given two occurrences of $w_1 = u_1 \ldots u_l$, we can easily match all the letters of these two occurrences except two identical letters, say u_p. These two identical letters can be match with any letter of I without violating the precedence constraints, as we can suppose that in the two occurrences, $u_1, \ldots, u_{p-1}$ are done at the beginning of the scheduling, and $u_{p+1}, \ldots, u_l$ at the end. Thus, with $2m.3m!(3(m-1)(m-1)!)^{m-1}$ words w_1 we can free $2.3m!(3(m-1)(m-1)!)^{m-1}$ occurrences for each letter in Σ_1 that can be matched with the letters not matched in S'. Thus, we obtain a scheduling S'' such that the number of $\overline{\Sigma}_1$ letters that are not matched is minimum, and there is at most one occurrence of each letter of Σ_1 that is not matched, depending on the parity of the number of occurrences of this letter. Observe that, for $m \geq 2$,

$$2m.3m!(3(m-1)(m-1)!)^{m-1} \leq (3m.m!)^m$$

Adding two occurrences of w_1 changes neither the minimum over $\overline{\Sigma}_1$ nor the parity of the number of occurrences of the letters of Σ_1, thus does not change the minimum number of letters not matched. □

The next theorem follows easily.

Theorem 4 *The scheduling problem $B1|chains; comp; B = 2|Cmax$ is polynomial when there is a fixed number of different operations, and each duration for the operations appears at most once in each task (all letters are different in one chain).*

Proof. Whenever the number m of operations is fixed, then the number of different words that have at most one of each letters is fixed. By Lemma 3, for each word of the instance, if this word appears more than the fixed bound $(3m.m!)^m$, then we can reduce it to $(3m.m!)^m$ or $(3m.m!)^m + 1$, depending on the parity, without affecting the total loss due to operations scheduled alone. Then we have a fixed number of different tasks, and each tasks has a bounded multiplicity. Hence, the number of instance of the problem is then fixed. Thus, the time of the algorithm is just the time of bounding the multiplicities, which can be done in linear-time. □

3.3 Bounded chain lengths

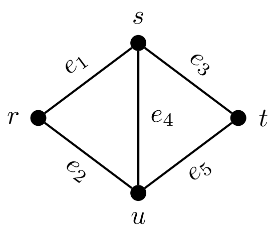
In this section, we consider the special case where the lengths of the chains are bounded. Proposition 2 considers the case of 3-length chains and the number of occurrences of each letter in the strings, the *orbit*, is smaller than 2. Notice that the shortest common supersequence problem with the orbit of the letters smaller than a given C is a special case of the scheduling problem $B1|chains; comp; B = C|Cmax$. The shortest common supersequence problem with bounded orbits for the letters has already been studied in [10]. Proposition 2 is equivalent to a theorem in [10] (where it is denoted by "case $n = 3$, $r = 2$"). We present a simple proof.

Proposition 2 *[10] The scheduling problem $B1|chains; comp; B = 2|Cmax$ is NP-complete even if each operation appears at most two times, and the length of each chain is at most three and all durations are equal to 1.*

Proof. We reduce from Stable Set. Let $G = (V, E)$ be an undirected graph. We can suppose that each vertex of G has a degree at least 2. Then, we replace each edge $\{u, v\}$ by two opposite arcs (u, v) and (v, u), giving the set of arcs A
The alphabet is given by $\Sigma = A \cup \bigcup_{v \in V}\{v_0, \dots, v_{2d(v)-2}\}$, where $d(v)$ is the degree of v in the initial graph. For each vertex v, if $a_1, \dots, a_k$ are the arcs entering v, $k = d(v)$, and $a'_1, \dots, a'_k$ are the arcs leaving v, we add the set of words (see Figure 2):

$$\begin{array}{l} a_1v_0 \\ a_2v_0v_1 \\ a_3v_1v_2 \\ \dots \\ a_kv_{k-2}v_{k-1} \\ v_{k-1}v_ka'_k \\ \dots \\ v_{2k-4}v_{2k-3}a'_3 \\ v_{2k-3}v_{2k-2}a'_2 \\ v_{2k-2}a'_1 \end{array}$$

We prove that it gives a polynomial Karp reduction: given a stable set S of G, for each $v \in V \setminus S$, we schedule the two occurrences of $v_{d(v)-1}$ in two different batches. Remark that removing the two occurrences of $v_{d(v)-1}$ allows us to schedule all the letters $a_1, \dots, a_{d(v)}$, $a'_1, \dots, a'_{d(v)}$ and $v_0 \dots v_{2d(v)-2}$ by pairs. Then, for the nodes in S, all the letters corresponding to arcs are already scheduled, thus the remaining letters can be easily scheduled.
Reciprocally, for the same reason, we can suppose that the only letters that are not scheduled by pairs in a solution of our problem are of the form $v_{d(v)-1}$ for $v \in V' \subset V$. Then, $S = V \setminus V'$ is a stable set of G. For otherwise, there would be an edge $uv \in E$ such that the letters of the words introduced by u and v are all scheduled by pairs, which is not possible. $\square$



	a'_1s_0	a'_2u_0	
a_1r_0	$a'_4s_0s_1$	$a_4u_0u_1$	a_3t_0
$a_2r_0r_1$	$a'_3s_1s_2$	$a'_5u_1u_2$	$a_5t_0t_1$
$r_1r_2a'_2$	$s_2s_3a_3$	$u_2u_3a_5$	$t_1t_2a'_5$
$r_2a'_1$	$s_3s_4a_4$	$u_3u_4a'_4$	$t_2a'_3$
	s_4a_1	u_4a_2	

Figure 2: Example of the reduction

In the following part, we deal with the case where all chains are of length at most two.

Definition 1 *We call an obstruction a set of k letters $a_1, \dots, a_k$ verifying:*

(i) *each letter appears in at most two chains;*

(ii) *there exists k chains $C_1, \dots, C_k$ of length 2 such that for all $1 \le i \le k-1$, a_i is the last operation of C_i and the first of C_{i+1}, and a_k is the first operation of a_1 and the last of C_k.*

Obviously, for each obstruction, there is at least one type of operation that must be done in two different batches. Moreover, for each letter that appears an odd number of times, at least one of the occurrences of this letter must be done isolated. The next theorem shows that these are the only two cases for which an operation must be scheduled alone.

Theorem 5 *In an optimal schedule of $B1|chains; comp; B = 2|Cmax$ where each chain is of length at most two, the number of operation that are done alone, is exactly two times the number of obstructions plus the number of operations that appear an odd number of times.*

Proof. Because of the preceding remarks, one inequality is already proved, thus we must show that there exists a scheduling with at most this number of operations done alone.

First, we schedule the tasks that appear in obstructions, by choosing for each obstruction one of the letters, the one with minimal weight, and by doing it in two different batches, every other type of operation of the obstruction can then be done in a single batch.

Next, let M be any maximal matching upon the remaining letters, the number of operations not matched is exactly the number of letters that appear an odd number of times. This matching can violate some constraints of precedence. Let G_M be the digraph whose vertices are the couples of M, and there is an arc from m_1 to m_2 if m_1 contains an operation that precedes one of the operations of m_2. Note that the degree of each vertex of G_M is at most 2, thus G_M is the union of some paths and some cycles. Now, choose M such that the number of directed cycles of G_M is minimum. Suppose that there is a cycle in G_M.

Let $C = \{m_1 = \{a_1, b_1\}, \ldots, m_n = \{a_n, b_n\}\}$ be the vertex set of one of these cycles, such that $a_1b_2, a_2b_3, \ldots, a_nb_1$ are chains of the problem. Suppose that a_i and a_j are batch-compatible, then by replacing m_i and m_j by $\{a_i, a_j\}$ and $\{b_i, b_j\}$, we have reduced the number of directed cycles in M without changing the maximality of M. Now suppose that the a_i are distinct. Note that at least one of the letters of $a_1, \ldots, a_n$ has at least three occurrences in the scheduling problem, otherwise C would be an obstruction, w.l.o.g. a_1 has at least 2 other batch-compatible operations. One of these operations is not in C, call it c_1. If c_1 is not matched, then replace m_1 by $\{a_1, c_1\}$ if c_1 is the first operation of its chain, $\{b_1, c_1\}$ otherwise. We have reduced the number of directed cycle of G_M by one. Suppose now that c_1 is matched with d_1 and that c_1 and d_1 appear at the first (resp. last) position of their respective tasks. Then, we replace m_1 and $\{c_1, d_1\}$ in M by $m' = \{a_1, c_1\}$ and $m'' = \{b_1, d_1\}$, giving M'. Now, m' has an in-degree of 0 (resp. m'' has an out-degree of 0) in $G_{M'}$, thus is not in the vertex set of any directed cycle of $G_{M'}$, proving that we have again reduced the number of directed cycle.
Finally, suppose that c_1 appears at the first position of its task, and d_1 at the last. Then, we replace m_1 and $\{c_1, d_1\}$ by $m' = \{a_1, c_1\}$ and $m'' = \{b_1, d_1\}$. m' has an in-degree of 0 and m'' has an out-degree of 0, thus the number of directed cycle is reduced by at least one.
We have proved that G_M is acyclic, then by taking any topological ordering of the couples of M, we obtain a scheduling that respects the precedence constraints and that reaches our bound. □

It follows:

Corollary 1 *The scheduling problem $B1|chains; comp; B = 2|Cmax$ is polynomial when the length of each chain is at most two.*

4 Conclusion

In this paper, we have considered a scheduling problem where the tasks are composed of ordered operations (chains of operations) to be scheduled on a batch machine with compatibility constraints. An analogy is made with supersequence problems where the tasks are the chains and the operations are the letters. The capacity of the machine indicates the maximum number of letters that can be matched and the duration of the operations is the weight of the letters.

Table 1 summarizes the results presented in this paper. The first column describes constraints on the chains ($|x| \leq c$ means that all chains are of length less than c; 'each letter once' means that a chain contains each letter at most once). The second column describes constraints on the number of chains (constant or not). The third column concerns the capacity B of the batch machine. The fourth column indicates whether the durations of all operations are equal or not and NP-C means ''NP-Complete".

chains	n	B	$\|\Sigma\|$	durations	
-	constant	-	-	-	polynomial (Theorem 1)
-	-	∞	2	$= 1$	NP-C [9]
$\|x\| \leq 2$	-	∞	-	$= 1$	NP-C (Proposition 1) [10]
half op. in T_1; $T_2 = \ldots = T_n$	-	2	2	$= 1$	NP-C (Theorem 2)
each letter once	-	2	-	$= 1$	NP-C (Theorem 3)
each letter once	-	2	cst	-	polynomial (Theorem 4)
$\|x\| \leq 3$	-	2	-	=1	NP-C (Proposition 2)
$\|x\| \leq 2$	-	2	-	-	polynomial (Corollary 1)

Table 1: Known results

Acknowledgments. The authors are grateful to András Sebő for several useful discussions regarding the topic of this paper.

References

[1] M. Boudhar and G. Finke. Scheduling on a batch machine with job compatibilities. *Belgian Journal of Operations Research, Statistics and Computer Science - JORBEL*, 40(1-2):69–80, 2000.

[2] P. Brucker, A. Gladky, J.A. Hoogeveen, M.Y. Kovalyov, C.N. Potts, T. Tautenhahn, and S.L. van de Velde. Scheduling a batching machine. *Journal of Scheduling*, 1:31–54, 1998.

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

[4] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 4:287–326, 1979.

[5] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge Univ Pr, 1997.

[6] V. Lebacque, N. Brauner, B. Celse, G. Finke, and C. Rapine. Planification d'expériences dans l'industrie chimique. In *Colloque IPI 2006*, Allevard, France, 2006.

[7] M. Middendorf. Supersequences, runs, and CD grammar systems. *Theoretical Computer Science, Topics in Computer Science*, 6:101–114, 1994.

[8] Chris N. Potts and Mikhail Y. Kovalyov. Scheduling with batching: A review. *European Journal of Operational Research*, 120(2):228–249, 2000.

[9] K.-J. Raiha and E. Ukkonen. The shortest common supersequence problem over binary alphabet is NP-complete. *Theoretical Computer Science*, 16:187–198, 1981.

[10] V.G. Timkovsky. Complexity of common subsequence and supersequence problems and related problems. *Cybernetics*, 25:565–580, 1990. English Translation from Kibernetika.