ELSEVIER Theoretical Computer Science 191 (1998) 205–214

Theoretical Computer Science

Note

Shortest common superstrings and scheduling with coordinated starting times

Martin Middendorf*

Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe, D-76128 Karlsruhe, Germany

Received July 1996; revised May 1997
Communicated by M. Nivat

Abstract

In the first part of this paper we show that the Shortest Common Superstring problem is NP-complete even if all strings are of the simple form 10^p10^q, $p,q \in \mathbb{N}$. This result closes the gap left between the polynomial cases where all strings are of the form 0^p10^q or all strings are of the form 10^p1 and NP-complete cases when strings have a more complicated structure. In the second part of the paper we use the above result to investigate the complexity of 2-machine flow-shop and open-shop problems with machines that have to coordinate their starting times, i.e. when one machine starts an operation the other machine also starts an operation or has to be idle at that time.

Keywords: Superstrings; Flow-shop; Job-shop; NP-completeness

1. Introduction

Given a set $\mathscr{S}$ of strings over an alphabet Σ the Shortest Common Superstring problem is to find a shortest string that contains each string in $\mathscr{S}$ as a substring (i.e. a consecutive block of characters). It is well known that the Shortest Common Superstring problem is NP-complete for strings over an alphabet of size 2 [4]. Moreover, the problem remains NP-complete if all given strings have a simple structure, e.g. if every given string is of one of the forms $0^p10^q10^r1$ or $10^p10^q10^r$, $p,q,r \in \mathbb{N}$ [7] (i.e. each string contains exactly three ones). When every given string is of the form 0^p10^q or every string is of the form 10^p1 the problem can be solved in polynomial time [7]. In the first half of this paper we close the remaining gap showing that even the case that every given string is of the form 10^p10^q is NP-complete. The motivation for this

* Tel.: +49 (721) 608 3; fax: +49 (721) 693 7; e-mail: middendorf@aifb.uni-karlsruhe.de.

study is that this version of the Shortest Common Superstring problem has applications in Scheduling and the planning of experiments. This is elaborated in the second half of this paper where we study the complexity of 2-machine flow shop and open shop problems with machines that have to coordinate their starting times.

2. Basic definitions and notation

For a string $S = s_1 s_2 \ldots s_n$ the length n of S is denoted by $|S|$. A string S is a *superstring* of a string S' if $S = TS'T'$ where T and T' are (possibly empty) strings. In this case S' is called *substring* of S. A string S is a (*common*) *superstring* of a set $\mathscr{L}$ of strings if S is a superstring of every string in $\mathscr{L}$. A superstring S of $\mathscr{L}$ is *minimal* if no proper substring of S is a superstring of $\mathscr{L}$. An *embedding* of $\mathscr{L}$ in $S = s_1 s_2 \ldots s_n$ is a function f from $\mathscr{L}$ to $[1:|S|]$ such that if $f(T) = i$ for $T \in \mathscr{L}$ then $T = s_i s_{i+1} \ldots s_{i+|T|-1}$. Let S be a superstring of a set $\mathscr{L} = \{S_1, S_2, \ldots, S_k\}$ of strings and f be an embedding of $\mathscr{L}$ in S. A substring S' of S is a *component* of S (with respect to f) iff

(i) $S = TS'T'$ and each string in $\mathscr{L}$ is embedded completely into one of the substrings T, S', or T', and

(ii) S' is minimal with property i), i.e. no substring of S' is a component of S.

If f is injective and w.l.o.g. $S_1, S_2, \ldots, S_k$ are the strings in $\mathscr{L}$ ordered by their f-values, then S_i, $i \in [1:k]$ is the ith *string* of f and S_{i+1} is the *successor* of S_i with respect to f, $i \in [1:n-1]$.

For a set $\mathscr{L} = \{S_1, S_2, \ldots, S_k\}$ of strings let $S = S_1 \rightarrow S_2 \rightarrow \cdots \rightarrow S_k$ denote the shortest string such that there exists an injective embedding from $\mathscr{L}$ into S such that S_i is the ith string of f.

In this paper we consider the problem to schedule a set of jobs on two identical machines M_1 and M_2. A job $J = (P, Q)$ consists of two operations where the first operation P has to be scheduled on M_1 and the second operation Q on M_2. Operation P (Q) has processing time $t(P)$ ($t(Q)$). The two operations of a job cannot be processed at the same time. A machine can process at most one operation at a time.

In the *open shop* problem we have no precedence relations between the operations of a job. In the *flow shop* problem it is required that for each job the operation on M_1 has to be finished before the operation on M_2 can start. In all our scheduling problems we want to minimize the makespan *Cmax*, i.e. the time interval between the start time of the first operation and the finish time of the last operation. In the *no wait* versions of our problems we require that the two operations of a job are processed directly one after the other. We use the classification that is widely used in the literature for scheduling problems (see e.g. [1]): $F2 \,||\, Cmax$ is the two machine flow-shop problem and $F2 \,|\, no\ wait \,|\, Cmax$ is the no wait version. The corresponding open shop problems are denoted by $O2 \,||\, Cmax$ and $O2 \,|\, no\ wait \,|\, Cmax$.

3. Shortest common superstring

In this section we present our result concerning the Shortest Common Superstring problem.

Theorem 1. *The Shortest Common Superstring problem is* NP-*complete even if each string is of the form* 10^p10^q, $p,q \in \mathbb{N}$.

Proof. Clearly, the problem is in NP. To show the completeness let a set $\mathscr{C} = \{C_1, C_2, \ldots, C_m\}$ of clauses each of size three over a set $V = \{v_1, v_2, \ldots, v_n\}$ of variables be an instance of the 3-SAT problem. Recall that the 3-SAT problem asks whether there exists a $\mathscr{C}$ satisfying truth assignment of V. We will construct a set $\mathscr{L}$ of strings over the alphabet $\{0,1\}$ as follows: Let $X = (n+1)\cdot(m+1)+1$. For each variable $v_i \in V$ let $s_i = (2i-1)\cdot X + m + 1$, $t_i = 2i \cdot X + m + 1$, and define the strings

$$T_i = 10^{t_i}10^{s_i}, \qquad T_i' = 10^{s_i}10^{t_i}.$$

For each clause $C_l = \{x_h, x_i, x_j\}$ with $h<i<j$ and $x_h = v_h \vee x_h = \bar{v}_h$, $x_i = v_i \vee x_i = \bar{v}_i$, and $x_j = v_j \vee x_j = \bar{v}_j$ let

$$p_l = \begin{cases} (2h-1)\cdot X + l & \text{if } x_h = v_h, \\ 2h \cdot X + l & \text{if } x_h = \bar{v}_h, \end{cases}$$

$$q_l = \begin{cases} (2i-1)\cdot X + l & \text{if } x_i = v_i, \\ 2i \cdot X + l & \text{if } x_i = \bar{v}_i, \end{cases}$$

$$r_l = \begin{cases} (2j-1)\cdot X + l & \text{if } x_j = v_j, \\ 2j \cdot X + l & \text{if } x_j = \bar{v}_j, \end{cases}$$

and define the strings

$$Q_l^1 = 10^{r_l}10^{p_l}, \qquad Q_l^2 = 10^{p_l}10^{q_l}, \qquad Q_l^3 = 10^{q_l}10^{r_l}.$$

For each $i \in [1:2n]$ define $R_i = 10^i10^{i\cdot X}$. Now, let $\mathscr{L} = \{T_i, T_i' \mid i \in [1:n]\} \cup \{Q_l^1, Q_l^2, Q_l^3 \mid l \in [1:m]\} \cup \{R_i \mid i \in [1:2n]\}$. Let $\alpha = \sum_{i=1}^{n}(s_i + t_i + 2)$, $\beta = \sum_{l=1}^{m}(p_l + q_l + r_l + 3)$, $\gamma = \sum_{i=1}^{2n}(i+1)$, $\delta = \sum_{i=1}^{2n}(i\cdot X + 1)$. Set $k = \alpha + \beta + \gamma + \delta + n\cdot(m+1)$.

Assume that we have a $\mathscr{C}$ satisfying truth assignment of V. Let exactly one true literal in each clause be marked. For $l \in [1:m]$ consider the clause $C_l = \{x_h, x_i, x_j\}$ with $h<i<j$ and define the string

$$G_l = \begin{cases} Q_l^1 \to Q_l^2 \to Q_l^3 & \text{if } X_h \text{ is marked}, \\ Q_l^2 \to Q_l^3 \to Q_l^1 & \text{if } X_i \text{ is marked}, \\ Q_l^3 \to Q_l^1 \to Q_l^2 & \text{if } X_j \text{ is marked}. \end{cases}$$

For $i \in [1:n]$: If v_i is true and $C_{l_1}, C_{l_2}, \ldots, C_{l_t}$ are the clauses where v_i is marked define

$$S_{2i-1} = R_{2i-1} \to G_{l_1} \to G_{l_2} \to \cdots \to G_{l_t} \to T_i' \to T_i, \qquad S_{2i} = R_{2i}$$

else let $C_{l_1}, C_{l_2}, \ldots, C_{l_t}$ be the clauses where $\bar{v}_i$ is marked and define

$$S_{2i-1} = R_{2i-1}, \qquad S_{2i} = R_{2i} \to G_{l_1} \to G_{l_2} \to \cdots \to G_{l_t} \to T_i \to T_i'.$$

Now it is easy to verify that the string $S = S_1 S_2 \ldots S_{2n}$ is a superstring of $\mathscr{L}$ with length k.

On the other hand, let S be a superstring of $\mathscr{L}$ of size $\leqslant k$ and f be an embedding of $\mathscr{L}$ in S. Without loss of generality let S be a minimal superstring. Clearly each component S' of S is of the form $10^{h_1}10^{h_2}\ldots 10^{h_j}$ for a $j \geqslant 2$ and there are strings $S_1, S_2, \ldots, S_{j-1}$ in $\mathscr{L}$ such that $S_i = 10^{h_i}10^{k_i}$ with $k_i \leqslant h_{i+1}$. Observe, that for each string 10^p10^q in $\mathscr{L}$ we have $q \geqslant X > 2n$. Hence, it follows that each string $R_i = 10^i10^{iX}$, $i \in [1:2n]$ must be the first string of a component of S.

Note, that the lengths of the first (respectively second) 0-runs of two different strings in $\mathscr{L}$ always differ. Hence for each two different strings in $\mathscr{L}$ the prefixes (suffixes) that consist of the first (respectively second) 1-run and 0-run are embedded on disjoint substrings of S. The sum of the lengths of these prefixes (suffixes) of all strings in $\mathscr{L}$ is $\alpha + \beta + \gamma$ (respectively $\alpha + \beta + \delta$). Thus, there can be at most $k - \alpha - \beta - \gamma = \delta + n \cdot (m+1)$ characters in S into which only characters of suffixes of the form 10^q of the strings in $\mathscr{L}$ are embedded. Hence, the sum of the lengths of the suffixes of the form 10^q of all components of S is at most $\delta + n \cdot (m+1)$. For $i \in [1:2n]$ set

$$\mathscr{L}_i = \{10^p10^q \in \mathscr{L} \mid i \cdot X \leqslant q \leqslant i \cdot X + m + 1\},$$

$$\mathscr{L}_i' = \{10^p10^q \in \mathscr{L} \mid i \cdot X \leqslant p \leqslant i \cdot X + m + 1\}.$$

Observe that $\bigcup_{i=1}^{2n} \mathscr{L}_i = \mathscr{L}$, $\bigcup_{i=1}^{2n} \mathscr{L}_i' = \mathscr{L} - \{R_i \mid i \in [1:2n]\}$, and $|\mathscr{L}_i| = |\mathscr{L}_i'| + 1$ for $i \in [1:2n]$. We need the following claims.

Claim 2. *S has exactly $2n$ components and for each $i \in [1:2n]$: One of the strings in $\mathscr{L}_i$ is the last string in a component of S.*

Proof of Claim 2. Since each string $R_i \in [1:2n]$ is the first string of a component of S it follows that S has at least $2n$ components. Since $|\mathscr{L}_i| = |\mathscr{L}_i'| + 1$ it is easy to see that for each $j \in [1:2n]$ there are at least $2n - j + 1$ strings in $\bigcup_{i=j}^{2n} \mathscr{L}_i$ that are the last string of a component of S. Hence, there exists $2n$ pairwise different strings $S_1, S_2, \ldots, S_{2n} \in \mathscr{L}$ each of which is the last string of a component of S and such that $S_j \in \bigcup_{i=j}^{2n} \mathscr{L}_i$. Since the sum of the lengths of the suffixes of the form 10^q of all components of S is at most $k - \alpha - \beta - \gamma = \delta + n \cdot (m+1)$ a simple calculation shows that $S_j \in \mathscr{L}_j$ must hold for each $j \in [1:2n]$. A similar calculation shows that S cannot have more than $2n$ components. Altogether, it follows that S has exactly $2n$ components. $\square$

Claim 3. *For $i \in [1:n]$: Each string in $\mathscr{L}_i$ that is not the last string in a component of S has a successor in $\mathscr{L}_i'$.*

Proof of Claim 3. Assume the Claim does not hold for an $i \in [1:2n]$, i.e. there exists a string $S^* \in \mathscr{L}_i$ that has a successor $S^{**} \in \mathscr{L}'_j$ with $j > i$. But then there must be at least $2n - j + 2$ strings in $\bigcup_{h=j}^{2n} \mathscr{L}_h$ that are last strings of a component of S. Similar to the proof of Claim 1 this would imply that S has length at least $\alpha + \beta + \gamma + \delta + X - m - 1 > k$ which is a contradiction. □

Claim 4. *For $i \in [1:n]$: The string T_i is the last string of a component of S and T'_i is its predecessor or vice versa.*

Proof of Claim 4. For an $i \in [1:n]$ consider the string $T_i = 10^{t_i}10^{s_i} \in \mathscr{L}_{2i-1}$. Observe that T'_i is the only string of the form 10^p10^q in $\mathscr{L}'_{2i-1}$ with $p = (2i-1) \cdot X + m + 1 = s_i$. All other strings in L'_{2i-1} have a first 0-run of length $< (2i-1) \cdot X + m + 1 = s_i$. From Claim 2 we derive that T_i must have T'_i as a successor if it is not the last string in a component of S. Analogously, it follows that T_i is the successor if T'_i is not the last string in a component of S. Hence, one of T_i and T'_i must be the last string in a component of S. □

Proof of Theorem 1 (*continued*). Claim 1 shows that for each $i \in [1:2n]$ there is a component of S with a suffix of the form 10^p with $i \cdot X \leqslant p \leqslant i \cdot X + m + 1$. Claim 3 implies that the sum of the lengths of the suffixes of the form 10^p of the components of S is at least $\delta + n \cdot (m+1)$. It was observed above that this sum cannot be larger than $\delta + n \cdot (m+1)$. The following claim follows easily:

Claim 5. *For each $i \in [1:n]$: Either T_i and R_{2i} or T'_i and R_{2i-1} are both last strings of a component of S.*

Proof of Theorem 1 (*continued*). Now we define a truth assignment for V. Let v_i be true iff T_i is the last string of a component of S.

Claim 6. *For each clause $C_l = \{x_{i_1}, x_{i_2}, x_{i_3}\}$, $l \in [1:m]$, $i_1 < i_2 < i_3$: If a literal $x_{i_j} \in C_l$ is false then the string $Q_l^{j+1 \bmod 3}$ is the successor of Q_l^j.*

Proof of Claim 6. First consider the case $x_{i_j} = v_{i_j}$. Since x_{i_j} is false the strings $T'_{i_j} \in \mathscr{L}'_{2i_j-1}$ and $R_{2i_j-1} \in \mathscr{L}_{2i_j-1}$ are last strings in a component of S. Claim 3 shows that $T_{i_j} \in \mathscr{L}_{2i_j-1}$ is the predecessor of T'_{i_j}. Assume that $Q_l^{j+1 \bmod 3}$ is not the successor of Q_l^j. Then there must be another string $Q_{l^*}^{j^*} \in \mathscr{L}'_{2i_j-1}$ with $l^* > l$ that is a successor of Q_l^j. This implies that there must exist a string $Q_{l^{**}}^{j^{**}} \in \mathscr{L}_{2i_j-1}$ with $l^{**} > l$ that has a successor in a set $\mathscr{L}'_{j'}$ with $j' > 2i_j - 1$. This is a contradiction to Claim 2. The case $x_{i_j} = \bar{v}_{i_j}$ is similar. □

Proof of Theorem 1 (*conclusion*). From Claim 5 we derive that each clause contains at least one true literal. □

4. Scheduling with coordinated starting times

This section contains the results about 2-machine flow shop and open shop problems with machines that have to coordinate their starting times. It is well known that $F2 \,||\, Cmax$ ([6]), $F2 \,|\, no\ wait \,|\, Cmax$ ([5]; the problem becomes NP-complete if jobs with only one operation for M_2 are allowed (Sahni, Cho 1977 cited in [3])), $O2 \,||\, Cmax$ (see e.g. [1]), and $O2 \,|\, no\ wait \,|\, Cmax$ (which is essentially the same problem as $F2 \,|\, no\ wait \,|\, Cmax$) can all be solved in polynomial time. Here we consider these problems under additional constraints concerning the starting times of operations on the machines. We will see that in most cases the problems become NP-complete under such constraints. Let us introduce the following three constraints

(i) R_1: A machine can start processing an operation at a time only when the other machine is idle or also starts processing an operation at the same time.

(ii) R_2: M_1 can start processing an operation at a time only when M_2 is idle or also starts processing an operation at the same time.

(iii) R_3: Same as R_2 but with M_1 interchanged with M_2.

Observe that every schedule for the no wait two machine flow shop problem fulfills constraint R_3. Thus, we consider only constraint R_2 for this problem.

Theorem 7. *$F2 \,|\, no\ wait \,|\, Cmax$ with restriction R_2 is* NP-*complete.*

Proof. It is easy to see that each instance of Shortest Common Superstring which is substringfree and where each string is of the form 10^p10^q is essentially an instance of our flow-shop problem. Each string $S = 10^p10^q$ represents a job J with operations O_1, O_2 where O_1 has to be processed on M_1 in time $p+1$ on M_1 and O_2 has to be processed on M_2 in time $q+1$. □

Theorem 8. *$F2 \,||\, Cmax$ is* NP-*complete under each of the constraints R_1, R_2, and R_3.*

Proof. (a) *Proof for R_1*: The construction is similar to that used in the proof of Theorem 7: Let a set $\mathscr{C} = \{C_1, C_2, \ldots, C_m\}$ of clauses each of size three over a set $V = \{v_1, v_2, \ldots, v_n\}$ of variables be an instance of the 3-SAT problem. We construct a set of jobs $\mathscr{J}$ as follows: Let $X = (n+1)(m+1)+2$. For each variable $v_i \in V$ let $s_i = (2i-1)\cdot X + m + 1$ and $t_i = 2i \cdot X + m + 1$, and define the jobs

$$J_i^v = \{P_i^v, Q_i^v\} \text{ with } t(P_i^v) = t_i,\ t(Q_i^v) = s_i,$$
$$J_i^{v'} = \{P_i^{v'}, Q_i^{v'}\} \text{ with } t(P_i^{v'}) = s_i,\ t(Q_i^{v'}) = t_i.$$

For each clause $C_l = \{x_h, x_i, x_j\}$ with $h<i<j$ and $x_h = v_h \vee x_h = \bar{v}_h$, $x_i = v_i \vee x_i = \bar{v}_i$, and $x_j = v_j \vee x_j = \bar{v}_j$ let

$$p_l = \begin{cases} (2h-1)\cdot X + l & \text{if } x_h = v_h, \\ 2h \cdot X + l & \text{if } x_h = \bar{v}_h, \end{cases}$$

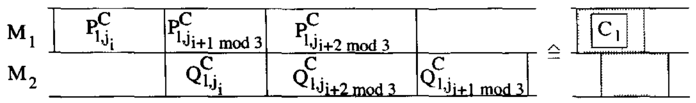
M1
M2
P^C_{l,j_i}
$P^C_{l,j_{i+1 \bmod 3}}$
$P^C_{l,j_{i+2 \bmod 3}}$
Q^C_{l,j_i}
$Q^C_{l,j_{i+2 \bmod 3}}$
$Q^C_{l,j_{i+1 \bmod 3}}$
≙
C_l

Fig. 1. Scheduling operations corresponding to C_l.

$$q_l = \begin{cases} (2i-1)\cdot X + l & \text{if } x_i = v_i, \\ 2i \cdot X + l & \text{if } x_i = \bar{v}_i, \end{cases}$$

$$r_l = \begin{cases} (2j-1)\cdot X + l & \text{if } \ x_j = v_j, \\ 2j \cdot X + l & \text{if } \ x_j = \bar{v}_j, \end{cases}$$

and define the jobs

$$J^C_{l,1} = \{P^C_{l,1}, Q^C_{l,1}\} \text{ with } t(P^C_{l,1}) = r_l,\ t(Q^C_{l,1}) = p_l,$$

$$J^C_{l,2} = \{P^C_{l,2}, Q^C_{l,2}\} \text{ with } t(P^C_{l,2}) = p_l,\ t(Q^C_{l,2}) = q_l,$$

$$J^C_{l,3} = \{P^C_{l,3}, Q^C_{l,3}\} \text{ with } t(P^C_{l,3}) = q_l,\ t(Q^C_{l,3}) = r_l.$$

For $i \in [1:2n]$ define the jobs $J_i = \{P_i, Q_i\}$ and $J'_i = \{P'_i, Q'_i\}$ with $t(P_i) = 1$, $t(Q_i) = iX$, $t(P'_i) = iX$, $t(Q'_i) = 1$.

Let $\mathscr{J} = \{J^v_i, J^{v'}_i \mid i \in [1:n]\} \cup \{J^C_{l,1}, J^C_{l,2}, J^C_{l,3} \mid l \in [1:m]\} \cup \{J_i, J'_i \mid i \in [1:2n]\}$. Let $\mathscr{P}$ (respectively $\mathscr{Q}$) be the set of first (respectively second) operations of the jobs in $\mathscr{J}$. Let $\alpha = \sum_{i=1}^{n} (t(P^v_i) + t(P^{v'}_i)) + \sum_{l=1}^{m} (t(P^C_{l,1}) + t(P^C_{l,2}) + t(P^C_{l,3})) + \sum_{i=1}^{2n} (t(P_i)) + t(P'_i))$ be the sum of the processing times of all (first) operations in $\mathscr{P}$. Observe that α equals also the sum of the processing times of all (second) operations in $\mathscr{Q}$. Let $k = \alpha + n(m+1) + 1$.

Assume that we have a $\mathscr{C}$ satisfying truth assignment of V. Let exactly one true literal in each clause be marked. For a clause $C_l = \{x_{j_1}, x_{j_2}, x_{j_3}\}$ with $1 \leqslant j_1 < j_2 < j_3 \leqslant n$ and $x_{j_h} = v_{j_h} \vee x_{j_h} = \bar{v}_{j_h}$, $h \in [1:3]$ let x_{j_i} be the marked true literal for an $i \in [1:3]$. We schedule the jobs $J^C_{l,1}, J^C_{l,2}, J^C_{l,3}$ corresponding to C_l as follows. $P^C_{l,j_i}, P^C_{l,j_{i+1 \bmod 3}}, P^C_{l,j_{i+2 \bmod 3}}$ are scheduled in this order on M_1 and $Q^C_{l,j_i}, Q^C_{l,j_{i+1 \bmod 3}}, Q^C_{l,j_{i+2 \bmod 3}}$ in this order on M_2 as indicated in Fig. 1.

If v_h is a true and $C_{l_1}, C_{l_2}, \ldots, C_{l_h}$, $l_1 < l_2 < \cdots < l_h$ are the clauses which contain v_h as a marked literal we schedule the operations corresponding to this clauses and this variable as follows (cf. Fig. 2): P_{2h-1}, the operations in $\{P^C_{l_1,1}, P^C_{l_1,2}, P^C_{l_1,3}\}$, the operations in $\{P^C_{l_2,1}, P^C_{l_2,2}, P^C_{l_2,3}\}, \ldots$, the operations in $\{P^C_{l_h,1}, P^C_{l_h,2}, P^C_{l_h,3}\}$, $P^{v'}_h$, P^h_v, and P'_{2h-1} in one block on M_1. In another block the operations P_{2h} and P'_{2h} on M_1. On M_2 we schedule Q_{2h-1}, operations in $\{Q^C_{l_1,1}, Q^C_{l_1,2}, Q^C_{l_1,3}\}$, operations in $\{Q^C_{l_2,1}, Q^C_{l_2,2}, Q^C_{l_2,3}\}, \ldots$, operations in $\{Q^C_{l_h,1}, Q^C_{l_h,2}, Q^C_{l_h,3}\}$, $Q^{h'}_v$, Q^v_h, and Q'_{2h-1} in one block and the operations Q_{2h} and Q'_{2h} in another block.

If v_h is a false and $C_{l_1}, C_{l_2}, \ldots, C_{l_h}$ are the clauses which contain $\bar{v}_h$ as a marked literal we schedule the jobs in a similar way as above but with the roles of $P_{2h-1}, P'_{2h-1}, Q_{2h-1}, Q'_{2h-1}$ interchanged with $P_{2h}, P'_{2h}, Q_{2h}, Q'_{2h}$. Moreover, the order of $P^{v'}_h$ and P^v_h ($Q^{v'}_h$ and Q^v_h) is reversed.

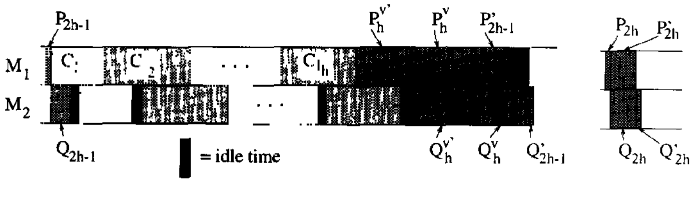

Fig. 2. Scheduling operations in case that v_h is true.

Observe that the sum of the idle times of M_2 during the two blocks of operations corresponding to v_h (i.e. between Q_{2h-1} and Q'_{2h-1} and between Q_{2h} and Q'_{2h}) is $(m+1)$. Altogether we have at most $n(m+1)$ idle time steps on M_2 during the blocks of operations corresponding to all variables v_1 to v_n. It is easy to see that we can combine the blocks of operations such that the whole schedule needs time at most $k = \alpha + n(m+1) + 1$.

On the other hand let a schedule S for $\mathscr{J}$ on M_1 and M_2 be given with $Cmax \leqslant k$. W.l.o.g. let S be such that at no time step $\leqslant Cmax$ both processors are idle. For $i \in [1:2n]$ set

$$\mathscr{P}_i = \{P \in \mathscr{P} \mid iX \leqslant t(P) \leqslant iX + (m+1)\},$$

$$\mathscr{Q}_i = \{Q \in \mathscr{Q} \mid iX \leqslant t(Q) \leqslant iX + (m+1)\}.$$

Note, that $|\mathscr{P}_i| = |\mathscr{Q}_i|$. Clearly, no operation in a set P_i, $i \in [1:2n]$ can start on M_1 at a time when no operation starts at the same on M_2 or when an operation in a set Q_j with $i \neq j$ starts on M_2 at the same time (this would cause at least $X - m + 1 > \alpha - k$ idle time steps on one of the machines). Thus, when an operation in P_i starts on M_1 there must be an operation from Q_i starting on M_2 at the same time and vice versa. For each $h \in [1:n]$ one of the following cases must hold: Q_h^v and $Q_h^{v'}$ start after P_h^v and or Q_h^v and $Q_h^{v'}$ start after $P_h^{v'}$. W.l.o.g. let us assume that for an $h \in [1:n]$ Q_h^v and $Q_h^{v'}$ start after $P_h^{v'}$. Then there must exist a sequence of operations $P_{l_1,i_1}^C, P_{l_2,i_2}^C, \ldots, P_{l_k,i_k}^C, P_h^{v'}$ with $P_{l_1,i_1}^C, P_{l_2,i_2}^C, \ldots, P_{l_k,i_k}^C \in \mathscr{P}_{2h-1}$ such that Q_{2h-1} starts at the same time as P_{l_1,i_1}^C, Q_{l_1,i_1}^C starts at the same time as $P_{l_2,i_2}^C, \ldots, Q_{l_k,i_k}^C$ starts at the same time as $P_h^{v'}$.

During these operations there are at least $m+1$ idle time steps on M_2 when M_1 is busy. For all $h \in [1:n]$ we have at least $n(m+1)$ idle time steps on M_2 when M_1 is busy (plus at least one additional such idle time of M_2 when the whole schedule begins). We conclude that for each $h \in [1:n]$ there exist an $f_h \in \{2h-1, 2h\}$ such that each operation in $\mathscr{P}_i$ starts at the same time as it's corresponding operation with the same processing time in $\mathscr{Q}_i$.

Now we define a truth assignment for V. Let $v_h \in V$ be true iff $f_h = 2h-1$. The last conclusion and our construction imply that in each clause at least one literal must be true.

(b) The proofs for constraints R_2 and R_3 are easy reductions of 3-Partition (see [3]): Let $B \in \mathbb{N}$, $A = \{a_1, a_2, \ldots, a_n\}$ a set of integers with $n = 3m, m \in \mathbb{N}$ and for all $i \in [1:n]$ $\frac{1}{3}B < a_i < \frac{1}{2}B$ be an instance of 3-Partition. Recall that 3-Partition asks whether there exists a partition of A into sets $A_1, A_2, \ldots, A_m$, each of size 3, such that $\sum_{a_j \in A_i} a_j = B$ holds for all $i \in [1:m]$. We construct a set of jobs $\mathscr{J} = \{J_1, J_2, \ldots, J_{n+m}\}$ as follows: For $i \in [1:n+m]$ $J_i = \{P_i, Q_i\}$ and $t(P_i) = 1$, $t(Q_i) = a_i \cdot (n+m+1)$ if $i \in [1:n]$ and $t(P_i) = B \cdot (n+m+1)$, $t(Q_i) = 1$ if $i \in [N+1 : n+m]$. Now it is easy to verify that there exists a schedule with $Cmax \leqslant B \cdot (n+m+1) + n + m$ iff there exists a 3-Partition of A.

The proof for R_3 is similar to the proof for R_2 and is omitted here. □

Theorem 9. *$O2 \mid no\ wait \mid Cmax$ is* NP*-complete under each of the constraints R_1, R_2, and R_3.*

Proof. (a) *Proof for R_1*: Let $\{S_1, S_2, \ldots, S_n\}$, k be an instance of Shortest Common Superstring from the proof of Theorem 1. For each string $S_i = 10^{p_i}10^{q_i}$, $i \in [1:n]$ introduce two jobs J_i and J_i'. J_i has operations O_{1i} with processing time $p_i + 1$ and O_{2i} with processing time ik, $i \in [1:n]$. J_i' has operations O_{1i}' with processing time $ik + 1$ and O_{2i}' has processing time $q_i + 1$, $i \in [1:n]$. Now it is easy to see that it is not possible to schedule $\{J_1, J_2, \ldots, J_n, J_1', J_2', \ldots, J_n'\}$ in time $\leqslant k + \sum_{i+1}^{n} ik + 2$ if for one job J_i or J_i' O_{2i} is scheduled before O_{1i} (respectively O_{2i}' before O_{1i}'). Hence, essentially we have to solve the corresponding flow shop problem which is NP-complete.

(b) Proof for R_2 and R_3: With the same construction as in (a) we can reduce $F2 \mid no\ wait \mid Cmax$ under constraint R_i to $O2 \mid no\ wait \mid Cmax$ under constraint R_i, $i \in \{2,3\}$. □

Theorem 10. *$O2 \parallel Cmax$ with constraint R_1 is polynomial time solvable.*

Proof. The problem can be transformed into the Maxweight Matching problem on bipartite graphs which is solvable in polynomial time (see e.g. [2]). Let $\mathscr{J} = \{J_1, J_2, \ldots, J_n\}$ be a set of jobs, $J_i = \{P_i, Q_i\}$, $i \in [1:n]$ where P_i (Q_i) has to be scheduled on M_1 (respectively M_2). Define the edge weighted bipartite graph $G = (V, E)$ with $V = \mathscr{P} \cup \mathscr{Q}$ and $E = \mathscr{P} \times \mathscr{Q}$ where the weight of an edge $w(\{P_i, Q_j\}) = \min\{t(P_i), t(Q_j)\}$ if $i \neq j$ and $w(\{P_i, Q_i\}) = 0$, $i, j \in [1:n]$. A matching is a subset E' of E such that no two edges in E' share a common vertex. The weight E' is the sum of the weights of the edges in E'. It is easy to see that there exists a schedule with $Cmax = k$ for $\mathscr{J}$ iff there exists a matching E' with weight $\sum_{i \in [1:n]} t(P_i) + \sum_{i \in [1:n]} t(Q_i) - k$. □

Theorem 11. *$O2 \parallel Cmax$ is* NP*-complete under each of the constraints R_2 and R_3.*

Proof. Similar to the proof of NP-completeness of $F2 \parallel Cmax$ under constraints R_2 and R_3. □

Acknowledgements

I would like to thank Vadim Timkovsky for helpful discussions and pointing out the relation between our special superstring problem and the flow-shop problem with constraint R_1.

References

[1] P. Brucker, Scheduling Algorithms, Springer, Berlin, 1995.
[2] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press, Cambridge, MA, 1989.
[3] M.R. Garey, D.S. Johnson, Computers and Intractability, W.H. Freeman, San Francisco, 1979.
[4] J. Gallant, D. Maier, J.A. Storer, On finding minimal length superstrings, JCCS 20 (1980) 50–58.
[5] P.C. Gilmore, R.E. Gomory, Sequencing a one-state variable machine: a solvable case for the travelling salesman problem, Oper. Res. 12 (1964) 655–679.
[6] S.M. Johnson, Optimal two- and three-stage production schedules with setup times included, Naval Res. Logistic Quart. 1 (1954) 61–68.
[7] M. Middendorf, More on the complexity of common superstring and supersequence problems, Theoret. Comput. Sci. 124 (1994) 205–228.