RC22359 (W0203-017) March 7, 2002
Computer Science

IBM Research Report

Thirteen Notes on Equal--Execution--Time Scheduling

Philippe J. Baptiste
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Vadim G. Timkovsky
CGI Group Inc., 1 Dundas St. W., Suite 2700
Toronto, Ontario M5G 1Z3, Canada, and
Department of Computing & Software, McMaster University
Hamilton, Ontario L8S 4L7, Canada

Research Division
Almaden - Austin - Beijing - Delhi - Haifa - India - T. J. Watson - Tokyo - Zurich

Thirteen Notes on Equal–Execution–Time Scheduling

Philippe Baptiste
IBM T. J. Watson Research Center
Mathematical Sciences, Office 35-214
P. O. Box 218, Yorktown Heights, NY 10598
baptiste@us.ibm.com

Vadim G. Timkovsky
CGI Group Inc., 1 Dundas St. W., Suite 2700
Toronto, Ontario M5G 1Z3, Canada, and
Department of Computing & Software, McMaster University
Hamilton, Ontario L8S 4L7, Canada
timko@cgi.ca

March 5, 2002

Abstract

We show how 13 equal–execution–time scheduling problems whose complexity status has been unknown before can be solved in polynomial time by well known and recently found techniques applicable to problems of this kind. We consider single–machine, parallel–machine, parallel–batch, open–shop and job–shop machine environments.

Keywords: scheduling, computational complexity, polynomial time.

Introduction

Looking through the listing of complexity results for scheduling problems [6] one can observe that the vast majority of problems whose complexity status is unknown involve jobs with equal–execution–time operations. Hence, the state–of–the–art in studying the complexity of scheduling problems has reached the level at which approaches and techniques successfully working for equal–execution–time scheduling become especially important.

This paper is devoted to polynomial–time solutions to equal–execution–time scheduling problems in parallel–machine, parallel–batch, open–shop and job–shop environments, the area where a certain progress has been recently made. When the results we present here are variations of earlier results that are explicitly discussed in cited papers, we give only short sketches on how to obtain polynomial–time algorithms for either problems by using certain modelling techniques such as reductions to similar problems, flows in networks, linear programming, dynamic programming and transversal graphs. Given such sketches the reader can restore necessary details.

All denotations follow the notation of Graham *et al.* [10]: C_j, w_j, r_j and d_j are the completion time, the weight, the release date and the due date of a job J_j, $j = 1, 2, \ldots, n$; $L_j = C_j - d_j$, $T_j = \max\{0, L_j\}$ and U_j, where $U_j = 0$ if $T_j = 0$ or $U_j = 1$ if $T_j > 0$, are the lateness, the tardiness and the unit penalty per late job, respectively; p_j is the execution time of a single–operation job that can be executed on any of the parallel machines M_i, $i = 1, 2, \ldots, m$; p_{ij} is the execution time of the ith operation, O_{ij}, of a multi–operation job, which is a chain of operations $O_{1j}O_{2j}\ldots O_{m_j j}$ in a job shop or an unordered set of operations $O_{1j}, O_{2j}, \ldots, O_{mj}$ in an open shop; O_{ij} has to be executed on a specified machine $M_{\mu_{ij}}$; the case of a job shop with $m_j = m$ and $\mu_{ij} = i$ is a flow shop; symbols P, Q, R, O, F, J in problem denotations stand for identical (equal–speed), uniform (different–speed) and unrelated parallel machines, an open shop, a flow shop and a job shop, respectively; the appended symbol m, for example Pm, means that the number of machines m is fixed; abbreviations *pmtn* and *nowait* mean that jobs can be preempted and that multi–operation jobs cannot be interrupted between operations, respectively.

$J_j \prec J_k$ means that J_j is an immediate predecessor of J_k in accordance with specified precedence constraints. Words *chains, intree* and *prec* denote chain–like, intree–like and arbitrary precedence constraints, respectively. We set $a_{\min} = \min_{1 \le j \le n} a_j$ and $a_{\max} = \max_{1 \le j \le n} a_j$ for numbers $a_1, a_2, \ldots, a_n$.

Without loss of generality, we assume that $r_{\min} = 0$. All numerical parameters are assumed to be integers. Among minimization criteria we consider $L_{\max}$, $\sum C_j$, $\sum T_j$, $\sum U_j$, $\sum w_j U_j$ and $\sum f_j$, where f_j is a nondecreasing cost function. The last criterion is the most general.

A key observation we use to obtain polynomial–time solutions to nonpreemptive problems with equal execution times $p_j = p$ or $p_{ij} = p$ is that there are very few possible start times. Indeed, since an optimal nonpreemptive schedule is active, *i.e.*, no operation can be shifted to the left without violation of release dates or precedence constraints, start times and completion times of jobs in it belong to the set

$$\mathcal{T} = \{t : \exists j \in [1, n] : \exists l \in [0, nm_{\max}] : t = r_j + lp\},$$

where $m_{\max} = 1$ in the case of single–operation jobs. Thus $\mathcal{T}$ contains at most $n(nm_{\max} + 1)$ time points. $\mathcal{T}_{\text{pmtn}}$ will denote a time space for a preemptive problem. In comparison with $\mathcal{T}$ it depends on preemptions and hence requires a special consideration.

1 $Pm|intree, p_j = 1|\sum C_j$

Using the fact that the precedence relation is an intree, it is easy to make sure by contradiction that the number of busy machines is not increasing in time in any optimal schedule. Hence, an optimal schedule defines the time unit $[t, t + 1]$ that contains a set L with less than m jobs such that either $t = 0$ or the time unit $[t - 1, t]$, where $t > 0$, contains m jobs. It is evident that L is the set of leaves of the subtree S scheduled in $[t, C_{\max}]$, and the set F of other jobs is a forest with mt jobs scheduled in $[0, t]$. Let C_j denote the completion time of J_j in an optimal schedule. Then the total completion time of the schedule can be written as $m(1 + 2 + \ldots + t) + \sum_{J_j \in S} C_j$.

Note that any active schedule for S is optimal because it requires less than m machines. Once L is given, F and S can be obviously restored in linear time. The subschedule for F and the time point t can also be found in linear time by Hu's algorithm [12]. To find a set L we need to check all $O(n^m)$ subsets with less than m jobs. Since m is fixed, we obtain a solution in polynomial time.

2 $Om|nowait, intree, p_{ij} = 1|\sum C_j$

There is a polynomial–time reduction to $Pm|intree, p_j = m|\sum C_j$ [5].

3 $P|chains, r_j, p_j = 1|L_{\max}$

Without loss of generality we assume that release dates and due dates are adjusted to precedence constraints by replacing r_k by $\max\{r_j + p, r_k\}$ and d_j by $\min\{d_j, d_k - p\}$ for each immediate precedence $J_j \lessdot J_k$. Hence, we can assume that release dates are increasing and due dates are decreasing along each chain of precedence constraints.

We consider only the decision version of the problem of finding a schedule meeting given release dates r_j and deadlines $D_j = d_j + L_{\max}$. If we show that it can be solved in polynomial time for a fixed $L_{\max}$, we can find a schedule with minimum $L_{\max}$ in polynomial time as well by a binary search.

Let us create a network with a source S, job vertices J_j, $1 \leq j \leq n$, chain–time vertices (c, t) for each chain c and integral time point $t \in \mathcal{T}$, time vertices $t \in \mathcal{T}$ and a sink T. Since the number of chains is at most n and $|\mathcal{T}| = O(n^2)$, the number of vertices in the network is $O(n^3)$. The arcs in the network will be the arcs in all possible paths

$$S \to J_j \to (c, t) \to t \to T$$

under the condition that arcs $J_j \to (c, t)$ exist if and only if $r_j \leq t < D_j$ and $J_j \in c$. Assign capacity m to arcs $t \to T$, and 1 to the other arcs.

It can be easily shown that a feasible schedule exists if and only if the maximal flow in the network is n. The unit flow through the arc $J_j \to (c, t)$ models an assignment of the job J_j belonging to the chain c to be executed in the time unit $[t, t + 1]$ inside the the time interval $[r_j, D_j]$. The flow of value at most m through the arc $t \to T$ models an assignment of at most m jobs to be executed in the time unit $[t, t + 1]$.

The network–flow model observes all precedence constraints with the exception of possibly those between jobs J_i and J_j that are both assigned to the time interval $[\max\{r_i, r_j\}, \min\{D_i, D_j\}]$. However, transposing such jobs we can obtain a schedule that observes all precedence constraints in time $O(n^2)$. The overall time complexity of finding a feasible schedule is at most $O(n^9)$, because finding a maximal flow in a network takes at most cubic time in the number of its vertices [9].

4 $O|chains, r_j, p_{ij} = 1|L_{\max}$

There is a polynomial–time reduction to $P|chains, r_j, p_j = 1|L_{\max}$ [21].

5 $1|pmtn, prec, r_j, p_j = p|\sum C_j$

As we show, there is no advatage to preemptions in this problem. Thus it can be solved in polynomial time by the algorithm of Simons [20] for $1|prec, r_j, p_j = p|\sum C_j$. Let t be the earliest preemption time, and let J_j be the job interrupted at time t, started at time S_j and completed at time C_j in a preemptive schedule. Let $\mathcal{K}$ and k denote the set and number of jobs, respectively, that are started not earlier than t and completed earlier than C_j. We can reconstruct the schedule in the time interval $[t, C_j]$ by moving all parts of J_j into the time interaval $[t, S_j + p]$, so that J_j becomes uninterrupted, and placing all parts of other jobs in $[t, C_j]$ without changing their order in time into the time interaval $[S_j + p, C_j]$. The reconstruction obviously observes precedence constraints and release dates. It is easy to make sure that the loss of J_j in completion time is at least kp and that the gain of jobs of $\mathcal{K}$ in total completion time is at most $k(p - t + S_j)$. This, the reconstruction decreases the total completion time by at least $k(t - S_j)$. Recursively applying the reconstruction procedure we obtain a nonpreemptive schedule that is not worse that the original preemptive schedule.

6 $P|pmtn, p_j = p|\sum T_j$

Let us consider the class of problems $P|pmtn, p_j = p|\sum f_j$, where f_j are convex nondecreasing functions such that differences $f_i - f_j$ are all monotone functions. Since tardinesses $T_j = \max\{0, C_j - d_j\}$ meet these conditions, $P|pmtn, p_j = p|\sum T_j$ is in the class. Note that $\sum T_j$ is a piecewise linear function. We assume that the jobs are in a linear order where for each pair of jobs J_i and J_j the functions $f_i - f_j$ are either strictly increasing or constant if $i < j$. As it is shown in [2], such an order always exists.

It can be trivially shown by the exchange argument that for any problem in the class there exists an optimal schedule where $i \le j \Rightarrow C_i \le C_j$. Let us consider completion times C_j for all $j = 1, 2, \ldots, n$ as deadlines. It is known [11] that a feasible schedule for the decision problem $P|pmtn$ with deadlines

C_j exits if and only if

$$\forall j : \sum_{i=1}^{n} \max\{0, p_i - \max\{0, C_i - C_j\}\} \leq C_j m \quad \text{and} \quad C_j \geq p_j.$$

Under the conditions $p_j = p$ and $i \leq j \Rightarrow C_i \leq C_j$ this predicate is

$$\forall j : \sum_{i=j+1}^{n} \max\{0, p - C_i + C_j\} \leq mC_j - jp \quad \text{and} \quad C_j \geq p.$$

Introducing additional variables $X_{ij} = \max\{0, p - C_i + C_j\}$ for all $i = 2, 3 \ldots, n$ and $j = 1, 2, \ldots, n$ we finally obtain the convex program of minimizing $\sum_{j=1}^{n} f_j(C_j)$ under the linear constraints

$$\begin{array}{rrcl} \forall j : & \sum_{i=j+1}^{n} X_{ij} \leq mC_j - jp & \text{and} & C_j \geq p, \\ \forall i : \forall j : & X_{ij} \geq p - C_i + C_j & \text{and} & X_{ij} \geq 0. \end{array}$$

For any convex piecewise linear objective function it can be solved in polynomial time [18]. Once the optimal completion times are known, the optimal schedule can be produced by Sahni's algorithm [19].

7 $Q|pmtn, p_j = p|\sum U_j$

Since late jobs can be arbitrarily scheduled we can consider schedules of only early jobs. It is easy to see that the maximum number of early jobs can be reached on jobs with latest due dates. In other words, if $d_1 \geq d_2 \geq \ldots \geq d_n$, then $J_1, J_2, \ldots, J_i$ are early jobs in an optimal schedule if and only if there is no schedule of the jobs $J_1, J_2, \ldots, J_i, J_{i+1}$ which are all early. We can test each integer among $0, 1, 2, \ldots, n$ on the role of i checking by the polynomial–time algorithm of Lawler and Labetoulle [16] for $R|pmtn, r_j$ whether there exists a schedule where all jobs among $J_1, J_2, \ldots, J_i$ are early.

8 $Pm|pmtn, p_j = p|\sum w_j U_j$

Let us consider the extension $Pm|pmtn|\sum w_j U_j$. Applying the dynamic programming approach to this problem, we use an algorithm of finding a feasible schedule for $P|pmtn$ that meets due dates d_j. The following recurrent procedure we call a staircase algorithm is just a variation of Sahni's algorithm [19]. Let us introduce an artificial job J_0 with execution time $p_0 = 0$ and due

date $d_0 = 0$. We assume that J_0 is scheduled at time 0 on each machine and that the jobs $J_0, J_1, \ldots, J_n$ are ordered by nondecreasing due dates.

Let after scheduling jobs $J_0, J_1, \ldots, J_{j-1}$ the machines become available at times a_i, $i = 1, 2, \ldots, m$, such that $a_1 \le a_2 \le \ldots \le a_m$. Then define an availability profile to be the vector $a = (a_1, a_2, \ldots, a_m)$. Initially $a_0 = (0, 0, \ldots, 0)$. To schedule J_j the algorithm in loop on $i = m, m-1, \ldots, 1$ assigns a part of J_j of length $\ell_{ij} = \max\{0, \min\{p_j, d_j - a_i\}\}$ to start at time a_i on M_i and sets

$$a_i = a_i + \ell_{ij}, \quad d_j = d_j - \ell_{ij}, \quad p_j = p_j - \ell_{ij}.$$

If $p_j = 0$ after the loop, then J_j becomes scheduled. The resulted availability profile we denote as a^j. It is easy to see that components of a^j are nondecreasing. If $p_j > 0$ after the loop, then J_j cannot be early, and the algorithm terminates. Using induction it can be shown that the algorithm terminates if and only if there is no feasible schedule. The proof, closely related to Sahni's result [19], can be found in [3].

Let $W_j(a)$ denote the minimum weighted number of late jobs among $J_j, J_{j+1}, \ldots, J_n$ for schedules with the availability profile a. It is easy to check that then

$$W_j(a) = \begin{cases} \min\{W_{j+1}(a) + w_j, W_{j+1}(a^j)\} & \text{if } a_j + p_j \le d_j, \\ W_{j+1}(a) + w_j & \text{otherwise.} \end{cases}$$

Setting $W_{n+1}(a) = 0$ for all availability profiles a, which number is at most $|\mathcal{T}_{\text{pmtn}}|^m$, we can solve the problem by the dynamic programming approach in time $O(n \cdot |\mathcal{T}_{\text{pmtn}}|^m)$.

To evaluate how much time space $\mathcal{T}_{\text{pmtn}}$ is needed in the case of equal execution times $p_j = p$ let us figure out how many different availability profiles the staircase algorithm can produce. Obviously, components of availability profiles can be computed as completion times of nonpreemptive parts of jobs. Let us find for each machine M_i a set of time points $T_{\text{pmtn},i}$ that contains all possible completion times in staircase schedules. If M_i runs a part of J_j in a staircase schedule, then C_{ij} will denote the completion time of the part. Thus $C_{ij} \in T_{\text{pmtn},i}$.

It is easy to see that $\mathcal{T}_{\text{pmtn},m}$ is contained in the set of positive integers in the time interval $[0, np]$ that are equal to due dates modulo p. Now let $i < m$. Notice that for any job J_j in a staircase schedule there exists $v_j \le m$ such that $M_1, M_2, \ldots, M_{v_j - 1}$ do not run J_j while $M_{v_j}, M_{v_j+1}, \ldots, M_m$ run J_j

in succession. Herein, if J_j passes M_i and M_{i+1} without waiting, then C_{ij} is also the completion time of another job on M_{i+1}. Let C_{ik}, where $k < j$, be the latest completion time earlier than C_{ij} such that $C_{ik} = C_{i+1,k-1}$. Since there are no idling machines in a staircase schedule, all jobs $J_{k+1}, \ldots, J_j$ are executed in the time intervals

$$\underbrace{[C_{ik}, C_{ij}]}_{\text{machine } M_i}, \underbrace{[C_{i+1,k}, C_{i+1,j}]}_{\text{machine } M_{i+1}}, \ldots, \underbrace{[C_{mk}, C_{mj}]}_{\text{machine } M_m}.$$

In other words, $C_{ik} = C_{i+1,k-1}$ and $\sum_{u=i}^{m}(C_{uj} - C_{uk}) = p(j-k)$. Extracting C_{ij} with $i < m$ gives

$$C_{ij} = p(j-k) + C_{i+1,k-1} - \textstyle\sum_{u=i+1}^{m}(C_{uj} - C_{uk}).$$

From this expression we have $|\mathcal{T}_{\text{pmtn},i}| \le n \cdot |\mathcal{T}_{\text{pmtn},i+1}| \cdot \prod_{u=i+1}^{m} |\mathcal{T}_{\text{pmtn},u}|^2$. Obviously, $|\mathcal{T}_{\text{pmtn},u+1}| \le |\mathcal{T}_{\text{pmtn},u}|$. Therefore, $\prod_{u=i+1}^{m} |\mathcal{T}_{\text{pmtn},u}|^2 \le |\mathcal{T}_{\text{pmtn},i+1}|^{2m-2i} < |\mathcal{T}_{\text{pmtn},i+1}|^{2m}$ and hence

$$|\mathcal{T}_{\text{pmtn},i}| < n \cdot |\mathcal{T}_{\text{pmtn},i+1}|^{2m+1}.$$

Resolving this inequality with the initial condition $|\mathcal{T}_{\text{pmtn},m}| = O(n^2)$ it is not hard to make sure that $|\mathcal{T}_{\text{pmtn},1}| < n^{2^m m^m}$. This implies that the problem can be solved by dynamic programming in time $O(n^{2^m m^{m+1}+1})$.

9 $Pm|r_j, p_j = p| \sum w_j U_j$

We describe a decomposition scheme that follows the technique used earlier for $1|r_j, p_j = p| \sum w_j U_j$ [1], $Pm|r_j, p_j = p| \sum w_j C_j$, $Pm|r_j, p_j = p| \sum T_j$ [2] and a suggestion in [8].

Define a resource profile to be a vector $a = (a_1, a_2, \ldots, a_m)$ with integral components in $\mathcal{T}$ such that $a_{\max} - a_{\min} \le p$, and the components are associated with the machines $M_1, M_2, \ldots, M_m$. We assume that $a_{\min}$ is the completion time of a job, $a_i = a_{\min}$ implies that a_i is the completion time of a job or an idle time on M_i, and $a_i \ne a_{\min}$ implies that a_i is the completion time of a job that is in process on M_i at time $a_{\min}$. Let $\preceq$ and $\prec$ denote the coordinate order and the strict coordinate order on resource profiles, respectively, *i.e.*,

$$a \preceq b \;\Leftrightarrow\; \forall i \in [1,m] : a_i \le b_i \;\text{ and }\; a \prec b \;\Leftrightarrow\; a \preceq b \;\wedge\; a \ne b.$$

Let $t_{\max} = \max_{t \in \mathcal{T}} t$. For a resource profile $x \neq (t_{\max}, t_{\max}, \ldots, t_{\max})$ define the set of next resource profiles

$$next(x) = \{x' : \exists h \in [1, m] : x_h = x_{\min} \ \wedge \ x_h + p = x'_h \ \wedge \ i \neq h \Rightarrow x_i = x'_i\}.$$

In what follows, we assume that jobs are ordered by nondecreasing due dates, *i.e.*, $d_1 \leq d_2 \leq \ldots \leq d_n$. We say that a schedule of jobs is feasible on an interval of resource profiles $a \prec b$ if the jobs are scheduled between their release dates and due dates, no more than m jobs are processed simultaneously, and M_i is idling before a_i and after b_i for all $i = 1, 2, \ldots, m$.

For positive integer $k \leq n$ and an interval of resource profiles $a \prec b$, let $W_k[a, b]$ denote the maximum weight of early jobs J_j with $j \leq k$ and $a_{\max} - p \leq r_j < b_{\min}$ in schedules that are feasible on $a \prec b$. Then it is not hard to make sure that the minimum weighted number of late jobs sought for is

$$\textstyle\sum_{j=1}^{n} w_j - W_n[(0, 0, \ldots, 0), (t_{\max}, t_{\max}, \ldots, t_{\max})]$$

and

$$W_k[a, b] = \begin{cases} \max\{W'_k[a, b], W_{k-1}[a, b]\} & \text{if } a_{\max} - p \leq r_k < b_{\min}, \\ W_{k-1}[a, b] & \text{otherwise,} \end{cases}$$

where

$$W'_k[a, b] = \max_{\substack{r_k \leq x_{\min} \leq d_k - p \\ x' \in next(x) \\ a \preceq x \\ x' \preceq b}} \{W_{k-1}[a, x] + w_k + W_{k-1}[x', b]\}.$$

If $a_{\max} - p > r_k$ or $r_k \geq b_{\min}$, then J_k is not taken into account in $W_k[a, b]$, so $W_k[a, b] = W_{k-1}[a, b]$. If $a_{\max} - p \leq r_k < b_{\min}$, then we try all possible positions of J_k by scheduling it between the ressource profiles x and x' on M_h with $x_h = x_{\min} = x'_h - p$. Since J_k is early, we add the cost w_k. As shown in [2], the problem of finding $W_k[a, b]$ is a decomposition of two independent subproblems of finding $W_{k-1}[a, x]$ and $W_{k-1}[x', b]$.

Finally, we add initial conditions $W_0[a, b] = 0$ for all pairs of resource profiles a and b with $a \prec b$. Since the number of resource profiles is at most $|\mathcal{T}|^m$, where $|\mathcal{T}| = O(n^2)$, the number of triplets a, x, b is at most $(|\mathcal{T}|^m)^3 = O(n^{6m})$. The recurrence formula should be applied for each $k = 1, 2, \ldots, n$, therefore, we can calculate all weights $W_k[a, b]$ and related schedules in time $O(n^{6m+1})$ using the dynamic programming approach [13].

10 $Om|nowait, r_j, p_{ij} = 1|\sum w_j U_j$

There is a polynomial–time reduction to $Pm|r_j, p_j = m|\sum w_j U_j$ [5].

11 $1|p\text{-}batch, r_j, p_j = p|\sum f_j$

Let us consider a more general problem, $1|p\text{-}batch, r_j|\sum f_j$, where execution times (lengths) of jobs can be different [4]. The jobs are to be executed in parallel batches, *i.e.*, jobs in a batch start and complete at the same time, and the length of a batch is the maximum length of its jobs. We assume that batches can contain an unbounded number of jobs.

It is easy to show by contradiction that there exists an optimal schedule where, for any batch, jobs that are shorter than the batch and released not later than the batch starts are assigned to the batch or a previous batch. Hence, if start time of a longest batch is known, then the problem can be decomposed into two subproblems that appear before and after completion time of the batch.

Let the jobs be ordered by nondecreasing lengths, *i.e.*, $p_1 \le p_2 \le \ldots \le p_n$, and let $\left[\begin{smallmatrix} a \\ x \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} b \\ y \end{smallmatrix}\right]$ denote a left batch and a right batch with start times a and b and lengths x and y, respectively. Define a subproblem for $k = 1, 2, \ldots, n$ as finding a minimum–cost schedule of jobs J_j with $j \le k$ and $a < r_j \le b$ that are assigned to the batch $\left[\begin{smallmatrix} b \\ y \end{smallmatrix}\right]$ or a batch following $\left[\begin{smallmatrix} a \\ x \end{smallmatrix}\right]$. Let $F_k\left[\begin{smallmatrix} a \\ x \end{smallmatrix}\right]\left[\begin{smallmatrix} b \\ y \end{smallmatrix}\right]$ denote the minimum cost of the schedule, and let $G_k\left[\begin{smallmatrix} a \\ x \end{smallmatrix}\right]\left[\begin{smallmatrix} b \\ y \end{smallmatrix}\right] = F_{k-1}\left[\begin{smallmatrix} a \\ x \end{smallmatrix}\right]\left[\begin{smallmatrix} b \\ y \end{smallmatrix}\right] + f_k(b + y)$. Then F_k can be expressed via G_k and F_{k-1} by the recurrence formula

$$F_k\left[\begin{smallmatrix} a \\ x \end{smallmatrix}\right]\left[\begin{smallmatrix} b \\ y \end{smallmatrix}\right] = \begin{cases} F'_k\left[\begin{smallmatrix} a \\ x \end{smallmatrix}\right]\left[\begin{smallmatrix} b \\ y \end{smallmatrix}\right] & \text{if } a < r_k \le b, \\ F_{k-1}\left[\begin{smallmatrix} a \\ x \end{smallmatrix}\right]\left[\begin{smallmatrix} b \\ y \end{smallmatrix}\right] & \text{otherwise,} \end{cases}$$

$$\text{where } F'_k\left[\begin{smallmatrix} a \\ x \end{smallmatrix}\right]\left[\begin{smallmatrix} b \\ y \end{smallmatrix}\right] = \min\left\{ G_k\left[\begin{smallmatrix} a \\ x \end{smallmatrix}\right]\left[\begin{smallmatrix} b \\ y \end{smallmatrix}\right], \min_{\max\{a+x, r_k\} \le t \le b - p_k} \left\{ G_k\left[\begin{smallmatrix} a \\ x \end{smallmatrix}\right]\left[\begin{smallmatrix} t \\ p_k \end{smallmatrix}\right] + F_{k-1}\left[\begin{smallmatrix} t \\ p_k \end{smallmatrix}\right]\left[\begin{smallmatrix} b \\ y \end{smallmatrix}\right] \right\} \right\}.$$

The interior minimum models the appearance of a new batch $\left[\begin{smallmatrix} t \\ p_k \end{smallmatrix}\right]$ capturing J_k between the batches $\left[\begin{smallmatrix} a \\ x \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} b \\ y \end{smallmatrix}\right]$. $G_k\left[\begin{smallmatrix} a \\ x \end{smallmatrix}\right]\left[\begin{smallmatrix} b \\ y \end{smallmatrix}\right]$ models the appearance of J_k in the batch $\left[\begin{smallmatrix} b \\ y \end{smallmatrix}\right]$. Setting $F_0\left[\begin{smallmatrix} a \\ x \end{smallmatrix}\right]\left[\begin{smallmatrix} b \\ y \end{smallmatrix}\right] = 0$ for all possible batches $\left[\begin{smallmatrix} a \\ x \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} b \\ y \end{smallmatrix}\right]$ we can calculate the minimum $\sum f_j$ as $F_n\left[\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right]\left[\begin{smallmatrix} r_{\max} + n p_{\max} \\ p_{\max} \end{smallmatrix}\right]$.

Since the number of quintets (t, a, b, x, y) is at most $|\mathcal{T}|^5$ in the case of equal–length jobs, and the recurrence formula should be applied for each $k = 1, 2, \ldots, n$, the problem can be solved by dynamic programming in time $O(n \cdot |\mathcal{T}|^5)$. Since $|\mathcal{T}| = O(n^2)$, we obtain the time bound $O(n^{11})$.

12 $J|prec, r_j, p_{ij} = p, n = k|\sum f_j$

Following [17] we reduce the problem to finding a shortest path in a related transversal graph. Vertices in it are vectors

$$u = (v_1, v_2, \ldots, v_k; t_1, t_2, \ldots, t_k),$$

where $v_j \in \{0, 1, 2, \ldots, m_j\}$ counts operations $O_{v_j j}$ in J_j if $v_j > 0$, and t_j is 0 if $v_j = 0$ or completion time of $O_{v_j j}$ if $v_j > 0$. A vertex v presents the right front of a partial schedule at time $t_{\max}$ that involves v_jth operations of jobs J_j with $v_j > 0$ and does not involve jobs J_j with $v_j = 0$. Vertices

$$(0, 0, \ldots, 0; 0, 0, \ldots, 0) \text{ and } (m_1, m_2, \ldots, m_k; t_1, t_2, \ldots, t_k)$$

are the source and a sink, respectively. A pair (u', u) is an arc if and only if $\exists i \in [1, k] : \forall j \neq i : (1) \wedge (2) \wedge (3) \wedge (4)$, where

$$\begin{array}{ll}
(1) & v_i = v'_i + 1 \ \wedge \ v_j = v'_j \ \wedge \ t_j = t'_j \ \wedge \ t_i \geq t_j, \\
(2) & \mu_{v_i i} = \mu_{v_j j} \Rightarrow t_i \geq t_j + p, \\
(3) & v_i > 1 \Rightarrow t_i \geq t'_i + p, \\
(4) & v_i = 1 \Rightarrow [\,[\, J_j \lessdot J_i \Rightarrow v_j = m_j \,] \ \wedge \ t_i \geq \max\{r_i, \max\limits_{J_j \lessdot J_i} t_j\} + p \,].
\end{array}$$

The predicates above mean that: (1) the front u is at time t_i and reached from the front u' by adding O_{1i} or exchanging $O_{v_i - 1, i}$ into $O_{v_i i}$ if $v_i > 1$, (2) $O_{v_i i}$ does not overlap in time with other operations that require the same machine $M_{\mu_{v_i i}}$, (3) start time $t_i - p$ of $O_{v_i i}$ is not earlier than completion time of the preceding operation $O_{v'_i i}$, (4) J_i starts only after all its immediate predecessors are completed and start time $t_i - p$ of J_i is not earlier than release date r_i or completion times of immediate predecessors of J_i. Define the length of the arc (u', u) to be

$$\ell(u', u) = \begin{cases} f_i(t_i) & \text{if } v_i = m_i, \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to make sure that source–sink paths in the graph are in a one–to–one correspondence with schedules of costs equal to the lengths of the correspondent paths. Hence, the problem reduces to searching a shortest path in the graph. Since the number of vertices is at most $m_{\max}^k \cdot |\mathcal{T}|^k$, where $|\mathcal{T}| = O(m_{\max})$, and a shortest path can be found in square time [7], the problem can be solved in polynomial time $O(m_{\max}^{4k})$.

13 $J|nowait, prec, r_j, p_{ij} = p, n = k|\sum f_j$

A polynomial–time solution to the no–wait counterpart of the previous problem and even to $J|nowait, prec, r_j, n = k|\sum f_j$ can be obtained by exchanging "$\geq$" into "=" in Predicate (3) and replacing "p" by "$p_{v_i-1,i}$" in Predicates (2), (3) and (4). The number of vertices in the transversal graph in this case is at most $m_{\max}^k \cdot |\mathcal{T}_{\text{nowait}}|^k$, where $\mathcal{T}_{\text{nowait}}$ is the time space required for saving all active no–wait job–shop schedules. Since no–wait jobs are not interrupted in such schedules,

$$\mathcal{T}_{\text{nowait}} = \{t : \exists l \in [1,k] : \forall j \in [1,k] : \exists x_j \in [1,m_j] : \exists y_j \in [1,m_j] : \\ x_j \leq y_j \quad \wedge \quad t = r_l + \textstyle\sum_{j=1}^{k} \sum_{i=x_j}^{y_j} p_{ij} \Big\} .$$

Thus $|\mathcal{T}_{\text{nowait}}| \leq k \cdot \prod_{j=1}^{k} m_j = O(m_{\max}^k)$. Hence, a shortest path in this case can be found in polynomial time $O(m_{\max}^{2k^2+2k})$.

Conclusion

Though we did not find an extension of the technique described in Note 1 to unit–time flow shops, we conjecture that $Fm|intree, p_{ij} = 1|\sum C_j$ can also be solved in polynomial time. Notes 8 and 11 present pseudopolynomial–time solutions to $Pm|pmtn|\sum w_jU_j$, $1|p\text{-}batch, r_j|\sum w_jU_j$, $1|p\text{-}batch, r_j|\sum w_jT_j$ that prove the ordinary NP–hardness of these problems since $1|pmtn|\sum w_jU_j$, $1|p\text{-}batch|\sum w_jU_j$, $1|p\text{-}batch|\sum w_jT_j$ are NP–hard [14, 4].

On the other hand, though $P|pmtn|\sum U_j$ [15] and $1||\sum w_jU_j$ [14] are NP–hard, it is still unclear whether $P|pmtn|\sum U_j$ or $Pm|r_j|\sum w_jU_j$ can be solved in pseudopolynomial time. The complexity status of $P|pmtn|\sum T_j$ remains unknown.

References

[1] Ph. Baptiste, Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine when processing times are equal, *Journal of Scheduling* **2** (1999) 245–252.

[2] Ph. Baptiste, Scheduling equal–length jobs on identical parallel machines, *Discrete Applied Mathematics* **103** (2000) 21–32.

[3] Ph. Baptiste, *Preemptive Scheduling of Identical Machines*, Technical Report, Université de Technologie de Compiègne, Compiégne, 2000.

[4] P. Brucker, A. Gladky, H. Hoogeveen, M. Kovalyov, C. Potts, T. Tautenhahn and S. van de Velde, Scheduling a batching machine, *Journal of Scheduling* **1** (1998) 31–54.

[5] P. Brucker, B. Jurisch and M. Jurisch, Open shop problems with unit time operations, *Zeitschrift für Operations Research* **37** (1993) 59–73.

[6] P. Brucker and S. Knust, *Complexity Results for Scheduling Problems*, `http://www.mathematik.uni-osnabrueck.de/research/OR/class/`.

[7] B. V. Cherkassky, A. V. Goldberg and T. Radzik, Shortest paths algorithms: theory and experimental evaluation, *Mathematical Programming* **73** (1996) 129–174.

[8] J. Chuzhoy, R. Ostrovsky and Y. Rabani, Approximation algorithms for the job interval selection problem and related scheduling problems, *Proc. 42nd Annual Symposium on Foundations of Computer Science*, Las Vegas, Nevada, 2001.

[9] A. V. Goldberg, É. Tardos and R. E. Tarjan, Network flow algorithms, in *Algorithms and Combinatorics* **9**: *Flows, Paths, and VLSI–Layout*, (B. Korte, L. Lovász, H. J. Prömel, and A. Schrijver, eds.), Springer–Verlag (1990) 101–164.

[10] 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* **5** (1979) 287–326.

[11] W. Horn, Some simple scheduling problems, *Naval Research Logistics Quarterly* **21** (1974) 177–185.

[12] T. C. Hu, Parallel sequencing and assembly line problems, *Operations Research* **9** (1961) 841–848.

[13] L. Hubert, Ph. Arabie and J. Meulman, *Combinatorial Data Analysis: Optimization by Dynamic Programming* (SIAM Monographs on Discrete Mathematics and Applications), SIAM 2001.

[14] R.M. Karp, Reducibility among combinatorial problems, in *Complexity of Computer Computations* (R.E. Miller and J.W. Thatcher, eds.) Plenum Press, New York, 1972, 85–103.

[15] E.L. Lawler, Recent results in the theory of machine scheduling, in *Mathematical Programming: the State of the Art* (A. Bachem, M. Grötschel and B. Korte, eds.), Springer, Berlin, 1983, 202–234.

[16] E. L. Lawler and J. Labetoulle, On preemptive scheduling of unrelated parallel processors by linear programming, *Journal of the Association for Computing Machinery* **25** (1978) 612–619.

[17] M. Middendorf and V. G. Timkovsky, Transversal graphs for partially ordered sets: sequencing, merging and scheduling problems, *Journal of Combinatorial Optimization* **3** (1999) 417–435.

[18] A. S. Nemirovskii and A. Ben–Tal, *Lectures on Modern Convex Optimization : Analysis, Algorithms, and Engineering Applications* (MPS–SIAM Series on Optimization), SIAM 2001.

[19] S. Sahni, Preemptive scheduling with due dates, *Operations Research* **27** (1979) 925–934.

[20] B. Simons, Multiprocessor scheduling of unit–time jobs with arbitrary release times and deadlines, *SIAM Journal on Computing* **12** (1983) 294–299.

[21] V. G. Timkovsky, *Identical Parallel Machines vs. Unit–Time Shops and Preemptions vs. Chains in Scheduling Complexity*, Technical Report, McMaster University, Hamilton, 1998.