Scheduling of Two Parallel Machines with Linear Decreasing Time Slot Costs to Minimize Total Weighted Completion Time *

Alexander Kononov[1,2] and Irina Lushchakova[3,4]

[1] Sobolev Institute of Mathematics,
4, Akad. Koptyug avenue, 630090, Novosibirsk, Russia.
[2] Novosibirsk State University, 2, Pirogova str., 630090, Novosibirsk, Russia
[3] Belarusian State University of Informatics and Radioelectronics, Belarus.
[4] UIIP NAS of Belarus, Belarus.
alvenko@math.nsc.ru,
IrinaLushchakova@yandex.ru

Abstract. We consider a scheduling problem with two parallel machines to minimize the sum of total weighted completion time and total machine time slot cost. In this paper we focus on the case of the constant or linear decreasing sequences of time slot costs. We suggest an exact pseudopolynomial DP algorithm for the case of arbitrary integer processing times of jobs. If all jobs have unit processing times, we modify our approach to obtain a polynomial algorithm.

Introduction

There are two parallel machines and a set $N = \{1, 2, ..., n\}$ of jobs that should be processed on these machines. An integer processing time $p_i > 0$ and a weight $w_i > 0$ are given for each job $i \in N$. It is assumed that every job is processed on at most one machine at a time, and each machine processes no more than one job at a time. In a schedule s, job i starts at time $r_i(s)$ and completes at time $C_i(s) = r_i(s) + p_i$, being processed on a machine without interruption. We shall consider the total weighted completion time $F_1(s) = \sum_{i=1}^{n} w_i C_i$ which is one of the traditional scheduling criteria.

Suppose that the planning horizon consists of K time slots with unit length. Assume that $p_i \leq K$ for each job $i \in N$ and $P = \sum_{i=1}^{n} p_i \leq 2K$. We shall call the interval $(k-1; k]$ by the kth slot, $k = 1, 2, ..., K$. Denote π_k^L the cost of using the kth time slot by machine L, $1 \leq L \leq 2$. Let $N_L(s)$ be a set of jobs assigned to machine L in a schedule s. For job $i \in N_L(s)$ denote $\bar{\pi}^L_{r_i(s),C_i(s)}$ its total time slot cost in a schedule s, i.e. $\bar{\pi}^L_{r_i(s),C_i(s)} = \bar{\pi}^L_{r_i(s),r_i(s)+p_i} = \pi^L_{r_i(s)+1} + \pi^L_{r_i(s)+2} + ... + \pi^L_{r_i(s)+p_i}$, where $r_i(s)$ is the starting time of processing job i in this schedule. Then the total time slot cost for a given planning horizon and a schedule s is the function $F_2(s) = \sum_{L=1}^{2} \sum_{i \in N_L(s)} \bar{\pi}^L_{r_i(s),C_i(s)}$.

* partially supported by the Project BRFFR Ф15 CO-043

In: A. Kononov et al. (eds.): DOOR 2016, Vladivostok, Russia, published at http://ceur-ws.org

A schedule s is determined by assigning for each job $i \in N$ its start moment $r_i(s)$ and machine L on which it will be processed. Notice that in a schedule s there may be some machine idle times between jobs.

The objective is to find a schedule s^* minimizing the function

$$F_1(s) + F_2(s) = \sum_{i=1}^{n} w_i C_i + \sum_{L=1}^{2} \sum_{i \in N_L(s)} \bar{\pi}^L_{r_i(s),C_i(s)}.$$

We denote this problem by $P2|slotcost|\sum(w_iC_i + \bar{\pi}^L_{r_i,C_i})$. In [1] the analogous problem for single machine is denoted by $1|slotcost|\sum(w_iC_i + \bar{\pi}_{r_i,C_i})$.

Proposition 1 ([1]). *If the sequence $\{\pi_k\}$ of time slot costs, $k = 1, 2, ..., K$, is non-decreasing, the problem $1|slotcost|\sum(w_iC_i + \bar{\pi}_{r_i,C_i})$ reduces to the classical problem $1||\sum w_iC_i$.*

The same conclusion can be done for the problem $P2|slotcost|\sum(w_iC_i + \bar{\pi}^L_{r_i,C_i})$.

Theorem 1. *[2] If the sequence $\{\pi_k\}$, $k = 1, 2, ..., K$, is non-increasing, the problem $1|slotcost|\sum(w_iC_i + \bar{\pi}_{r_i,C_i})$ is NP-hard in the strong sense.*

Consider the case of the problem $1|slotcost|\sum(w_iC_i + \bar{\pi}_{r_i,C_i})$ where the time slot cost decreases as a linear function of the time index k, i.e. $\pi_k - \pi_{k+1} = \varepsilon$, $\varepsilon > 0$, for $k = 1, 2, ..., K-1$.

Lemma 1. *[2] If the sequence $\{\pi_k\}$, $k = 1, 2, ..., K$, is linear decreasing, then in a optimal schedule for the problem $1|slotcost|\sum(w_iC_i + \bar{\pi}_{r_i,C_i})$ the jobs are scheduled in the order $\frac{w_1}{p_1} \geq \frac{w_2}{p_2} \geq ... \geq \frac{w_n}{p_n}$.*

Lemma 2. *[2] For a job i, if $\frac{w_i}{p_i} > \varepsilon$, then it is optimal to process this job as early as possible. If $\frac{w_i}{p_i} < \varepsilon$, then it is optimal to process job i as late as possible. If $\frac{w_i}{p_i} = \varepsilon$, then the job has the same cost no matter when it is processed.*

Based on the above lemmas, an $O(nlogn)$ algorithm is suggested in [2] for the problem $1|slotcost|\sum(w_iC_i + \bar{\pi}_{r_i,C_i})$ with linear decreasing time slot cost sequence.

1 Algorithm DP-LinDec-2 for the case of arbitrary processing times

Suppose that two parallel machines have either constant or linear decreasing time slot cost sequences $\{\pi^1_k\}$ and $\{\pi^2_k\}$: $\pi^1_k - \pi^1_{k+1} = \varepsilon_1$ and $\pi^2_k - \pi^2_{k+1} = \varepsilon_2$ for $k = 1, 2, ..., K-1$, where $0 \leq \varepsilon_1 \leq \varepsilon_2$.

Divide the set N of jobs into three subsets: $J_1 = \{i : w_i/p_i > \varepsilon_2\}$, $J_2 = \{i : w_i/p_i < \varepsilon_1\}$, $J_3 = \{i : \varepsilon_1 \leq w_i/p_i \leq \varepsilon_2\}$. For a nonempty set J_j, $1 \leq j \leq 3$, we set $n_j = |J_j|$, $P_j = \sum_{i \in J_j} p_i$; otherwise let $n_j = 0$, $P_j = 0$.

Let us order and renumber the jobs in such way that we have:

$J_1 = \{i_1, i_2, ..., i_{n_1}\}$ and $\frac{w_{i_1}}{p_{i_1}} \geq \frac{w_{i_2}}{p_{i_2}} \geq ... \geq \frac{w_{i_{n_1}}}{p_{i_{n_1}}} > \varepsilon_2$;

$J_2 = \{k_1, k_2, ..., k_{n_2}\}$ and $\frac{w_{k_1}}{p_{k_1}} \leq \frac{w_{k_2}}{p_{k_2}} \leq ... \leq \frac{w_{k_{n_2}}}{p_{k_{n_2}}} < \varepsilon_1$;

$J_3 = \{l_1, l_2, ..., l_{n_3}\}$ and $\varepsilon_2 \geq \frac{w_{l_1}}{p_{l_1}} \geq \frac{w_{l_2}}{p_{l_2}} \geq ... \geq \frac{w_{l_{n_3}}}{p_{l_{n_3}}} \geq \varepsilon_1$.

For the problem under consideration we present an algorithm **DP-LinDec-2** (Dynamical Programming- Linear Decreasing time slot costs - 2 machines) which proceeds in three stages.

At **Stage 1**, algorithm **DP-LinDec-2** schedules jobs from the set J_1.

Let $J_1 \neq \emptyset$. Denote $\bar{p}_m^{(1)} = p_{i_1} + p_{i_2} + ... + p_{i_m}$ for $m = 1, 2, ..., n_1$; $P_1 = \bar{p}_{n_1}^{(1)}$.

Consider a subproblem with jobs $i_1, i_2, ..., i_m$. Suppose that machines start at time 0. Let $f_1(x, m)$ be the minimal value of the objective function when machine 1 completes the processing at time x, $0 \leq x \leq \min\{P_1, K\}$. Then machine 2 should complete the processing at time $(\bar{p}_m^{(1)} - x)$. Due to the WSPT property, job i_m is the last job either on machine 1 or on machine 2. Hence, we have the following dynamic programming recursion:

$$f_1(x, m) = \min\{f_1(x - p_{i_m}, m-1) + w_{i_m} x + \bar{\pi}^1_{x-p_{i_m},x},$$
$$f_1(x, m-1) + w_{i_m}(\bar{p}_m^{(1)} - x) + \bar{\pi}^2_{\bar{p}_{m-1}^{(1)}-x, \bar{p}_m^{(1)}-x}\}.$$

The boundary conditions are:

$f_1(x, m) = +\infty$ for $x < 0$, or $x > \min\{K, \bar{p}_m^{(1)}\}$, or $\bar{p}_m^{(1)} - x > K$.

The initial conditions are

$f_1(x, 1) = w_{i_1} p_{i_1} + \bar{\pi}^2_{0,p_{i_1}}$, if $x = 0$,

$w_{i_1} p_{i_1} + \bar{\pi}^1_{0,p_{i_1}}$, if $x = p_{i_1}$, $+\infty$ otherwise.

For each $x = 0, 1, ..., \min\{P_1, K\}$ we keep the value $f_1(x, n_1)$ and its corresponding schedule $\sigma'(x)$. If $J_1 = \emptyset$, we set $x = 0$, $f_1(0, 0) = 0$. In this case the corresponding schedule $\sigma'(0)$ is a dummy one. The time complexity of **Stage 1** is $O(n_1 P_1)$.

At **Stage 2**, algorithm **DP-LinDec-2** schedules jobs from the set J_2.

Let $J_2 \neq \emptyset$. Denote $\bar{p}_m^{(2)} = p_{k_1} + p_{k_2} + ... + p_{k_m}$ for $m = 1, 2, ..., n_2$; $P_2 = \bar{p}_{n_2}^{(2)}$.

Consider a subproblem with jobs $k_1, k_2, ..., k_m$ from the set J_2. Let $f_2(y, m)$ be the minimal value of the objective function when machine 2 starts the processing of some of these jobs at time y. Then machine 1 starts the processing of the remaining part of these jobs at time $2K - y - \bar{p}_m^{(2)}$. Both machines complete the processing at time K. Actually, from the symmetry between the set J_1 and the set J_2 we can use the similar dynamic programming approach in the opposite direction. Formally, we have the following dynamic programming recursion:

$$f_2(y, m) = \min\{f_2(y + p_{k_m}, m-1) + w_{k_m}(y + p_{k_m}) + \bar{\pi}^2_{y,y+p_{k_m}},$$
$$f_2(y, m-1) + w_{k_m}(2K - y - \bar{p}_{m-1}^{(2)}) + \bar{\pi}^1_{2K-y-\bar{p}_m^{(2)}, 2K-y-\bar{p}_{m-1}^{(2)}}\}.$$

The boundary conditions are:

$f_2(y, m) = +\infty$ for $y > K$, or $y < \max\{0, K - \bar{p}_m^{(2)}\}$, or $2K - y - \bar{p}_m^{(2)} < 0$.

The initial conditions are

$f_2(y, 1) = w_{k_1} K + \bar{\pi}^2_{K-p_{k_1},K}$, if $y = K - p_{k_1}$,

$w_{k_1} K + \bar{\pi}^1_{K-p_{k_1},K}$, if $y = K$, $+\infty$, otherwise.

For each $y = K, K-1, ..., \max\{0, K - P_2\}$ we keep the value $f_2(y, n_2)$ and its corresponding schedule $\sigma''(y)$. If $J_2 = \emptyset$, we set $y = K$, $f_2(K, 0) = 0$. In this case the

corresponding schedule $\sigma''(K)$ is a dummy one. The time complexity of **Stage 2** is $O(n_2P_2)$.

Stage 3 of algorithm **DP-LinDec-2** is realized for all values of the variables x and y, $0 \le x \le \min\{P_1, K\}$, $\max\{0, K - P_2\} \le y \le K$.

Let us consider the particular values of x and y from the above ranges. At first the algorithm checks, if there exists a feasible schedule $\sigma(x, y)$ such that in this schedule jobs from the sets J_1 and J_2 are processed in the same time intervals and on the same machines as in the schedules $\sigma'(x)$ and $\sigma''(y)$. A schedule $\sigma(x, y)$ is feasible if each machine processes no more than one job at any time slot. To make sure that a feasible schedule exists, it is sufficient to verify that the inequality $P_1 - y \le x \le 2K - y - P_2$ holds. If a schedule $\sigma(x, y)$ is feasible, the algorithm inserts the jobs from the set J_3 into it.

Let $J_3 \ne \emptyset$. Denote $\bar{p}_m^{(3)} = p_{l_1} + p_{l_2} + ... + p_{l_m}$ for $m = 1, 2, ..., n_3$; $P_3 = \bar{p}_{n_3}^{(3)}$.

Consider a subproblem with jobs $l_1, l_2, ..., l_m$ from the set J_3. We know that machine 1 starts the processing of some of these jobs at time x and completes at time z, while machine 2 starts the processing of the remaining part of these jobs at time u. Let $f_3(x, y, z, u, m)$ be the minimal value of the objective function when machine 1 completes the processing of the first part of the set J_3 at time z. Then machine 2 should complete the processing of the remaining jobs of the set J_3 at time $u + \bar{p}_m^{(3)} - (z - x)$. Due to the WSPT property, job l_m is the last job from the set J_3 either on machine 1 or on machine 2. Hence, we have the following dynamic programming recursion:

$$f_3(x, y, z, u, m) = \min\{f_3(x, y, z - p_{l_m}, u, m - 1) + w_{l_m} z + \bar{\pi}^1_{z - p_{l_m}, z},$$

$$f_3(x, y, z, u, m - 1) + w_{l_m}(u + \bar{p}_m^{(3)} - (z - x))) + \bar{\pi}^2_{u + \bar{p}_{m-1}^{(3)} - (z-x), u + \bar{p}_m^{(3)} - (z-x)}\}.$$

The boundary conditions are:

$$f_3(x, y, z, u, m) = +\infty \text{ for } z < x, \text{ or } z > \min\{2K - y - P_2, x + \bar{p}_m^{(3)}\},$$
$$\text{or } u + \bar{p}_m^{(3)} - (z - x) > y.$$

The initial conditions are:

$$f_3(x, y, z, u, 1) = w_{l_1}(u + p_{l_1}) + \bar{\pi}^2_{u, u + p_{l_1}}, \text{ if } z = x \text{ and } u + p_{l_1} \le y,$$
$$w_{l_1}(x + p_{l_1}) + \bar{\pi}^1_{x, x + p_{l_1}}, \text{ if } z = x + p_{l_1} \le 2K - y - P_2,$$
$$+\infty, \text{ otherwise.}$$

Thus, for any feasible schedule $\sigma(x, y)$ algorithm **DP-LinDec-2** constructs a family of feasible schedules $\tilde{\sigma}(x, y, z, u)$, $x \le z \le \min\{x + P_3, 2K - y - P_2\}$, $\max\{y - P_3, P_1 - x\} \le u \le y$. Each schedule $\tilde{\sigma}(x, y, z, u)$ has the value $f_3(x, y, z, u, n_3)$ of the objective function. If $J_3 = \emptyset$ we set $z = x$, $u = y$, $f_3(x, y, x, y, 0) = 0$. In this case the corresponding schedule $\tilde{\sigma}(x, y, x, y)$ coincides with the schedule $\sigma(x, y)$.

At each step of **Stage 3** we keep only the current schedule with the minimal value of the function $f_1(x, n_1) + f_2(y, n_2) + f_3(x, y, z, u, n_3)$ that is found over all values of the variables x, y, z, u enumerated till the moment. After finishing **Stage 3** the algorithm finds an optimal schedule. The time complexity of **Stage 3** is $O(n_3P_1P_2P_3^2)$.

The whole time complexity of algorithm **DP-LinDec-2** is $O(n_1P_1 + n_2P_2 + n_3P_1P_2P_3^2)$. It should be mentioned that at all stages of algorithm the space complexity is $O(nK)$.

2 Algorithm DP-LinDec-2-UT for the case of unit processing times

Consider the particular case of the problem under consideration when all jobs have unit processing times, i.e. all $p_i = 1$. Since in this case $P_j = n_j$, $1 \le j \le 3$, the algorithm **DP-LinDec-2** becomes polynomial algorithm with the complexity $O(n_1^2 + n_2^2 + n_1 n_2 n_3^3)$. However, we can suggest the modification of our approach to reduce the complexity to $O(n_1 n_2 n_3)$. Let us consider an algorithm **DP-LinDec-2-UT**(Dynamical Programming- Linear Decreasing time slot costs - 2 machines-Unit Times)which consists of four stages.

Let L_1, L_2 be the numbers of machines, $L_1 \in \{1, 2\}$, $L_2 = 3 - L_1$.

At **Stage 1**, algorithm **DP-LinDec2-UT** calculates the contribution of jobs from the set J_1 into the optimal value of the objective function. Let $J_1 \neq \emptyset$. Suppose that machines start at time 0. Let $g_1(x, L_1)$ be the minimal contribution of jobs from the set J_1 when machine L_1 completes the processing at time x. Then machine L_2 should complete the processing at time $(n_1 - x)$. Jobs will be scheduled on machines in the non-increasing order of their weights. Hence, we have the following dynamic programming recursion:

$g_1(0, L_1) = \bar{\pi}^{L_2}_{0,n_1} + \sum_{j=1}^{n_1} j w_{i_j}$,

$g_1(x, L_1) = g_1(x-1, L_1) + \bar{\pi}^{L_1}_{x-1,x} - \bar{\pi}^{L_2}_{n_1-x,n_1-(x-1)} - \sum_{j=2x}^{n_1} w_{i_j}$,

where $x = 1, 2, ..., [\frac{n_1}{2}]$ and $L_1 \in \{1, 2\}$.

We keep all values $g_1(x, 1)$, $g_1(x, 2)$, where $x = 0, 1, ..., [\frac{n_1}{2}]$. If $J_1 = \emptyset$, we set $x = 0$, $g_1(0, 1) = 0$, $g_1(0, 2) = 0$. The time complexity of **Stage 1** is $O(n_1)$.

At **Stage 2**, algorithm **DP-LinDec2-UT** calculates the contribution of jobs from the set J_2 into the optimal value of the objective function. Let $J_2 \neq \emptyset$ and $g_2(y, L_1)$ be the be the minimal contribution of jobs from the set J_2 when machine L_1 starts the processing at time $K - y$. Then machine L_2 should start the processing at time $K - (n_2 - y)$. Both machines complete the processing at time K. We have the following dynamic programming recursion:

$g_2(0, L_1) = \bar{\pi}^{L_2}_{K-n_2,K} + \sum_{j=1}^{n_2} (K - n_2 + j) w_{k_j}$,

$g_2(y, L_1) = g_2(y-1, L_1) + \bar{\pi}^{L_1}_{K-y,K-y+1} - \bar{\pi}^{L_2}_{K-(n_2-y+1),K-(n_2-y)} +$

$\sum_{j=1}^{n_2-2y+1} w_{k_j}$, where $y = 1, 2, ..., [\frac{n_2}{2}]$ and $L_1 \in \{1, 2\}$.

We keep all values $g_2(y, 1)$, $g_2(y, 2)$, $y = 0, 1, ..., [\frac{n_2}{2}]$. If $J_2 = \emptyset$, we set $y = 0$, $g_2(0, 1) = 0$, $g_2(0, 2) = 0$. The time complexity of **Stage 2** is $O(n_2)$.

At **Stage 3**, algorithm **DP-LinDec-2-UT** calculates the contribution of jobs from the set J_3 into the optimal value of the objective function under the condition that jobs from the sets J_1 and J_2 have been already assigned to machines.

Let $J_3 \neq \emptyset$ and $g_3(z, x, y)$ be the minimal contribution of jobs from the set J_3 when machine 1 starts the processing of some of these jobs at time x and completes their processing at time $x + z$, while machine 2 starts the processing of the remaining part of these jobs at time $K - y - (n_3 - z)$ and completes their processing at time $K - y$. We have the following dynamic programming recursion:

$g_3(0, x, y) = \bar{\pi}^{2}_{K-y-n_3,K-y} + \sum_{j=1}^{n_3} (K - y - n_3 + j) w_{l_j}$,

$g_3(z, x, y) = g_3(z-1, x, y) + \bar{\pi}^{1}_{x+z-1,x+z} - \bar{\pi}^{2}_{K-y-(n_3-(z-1)),K-y-(n_3-z)} +$

$(x+z)w_{l_z} - (K-y-(n_3-z))w_{l_z}$,
where $x = 0, 1, 2, ..., [\frac{n_1}{2}]$, $y = 0, 1, 2, ..., [\frac{n_2}{2}]$, $z = 1, 2, ..., n_3$.

We keep all values $g_3(z, x, y)$, $x = 0, 1, ..., [\frac{n_1}{2}]$, $y = 0, 1, ..., [\frac{n_2}{2}]$, $z = 0, 1, ..., n_3$. If $J_3 = \emptyset$, we set $z = 0$, $g_3(0, x, y) = 0$ for $x = 0, 1, ..., [\frac{n_1}{2}]$, $y = 0, 1, ..., [\frac{n_2}{2}]$. The time complexity of **Stage 3** is $O(n_1 n_2 n_3)$.

At **Stage 4**, algorithm **DP-LinDec-2-UT** finds an optimal schedule. The total cost of an optimal schedule corresponds to the value $G = \min\{G_1, G_2, G_3\}$, where

$G_1 = \min\{g_1(x, 1) + g_2(y, 1) + g_3(0, x, y) | x = 0, 1, ..., [\frac{n_1}{2}]; y = 0, 1, ..., [\frac{n_2}{2}]; n_1 - x \leq K - (n_2 - y + n_3)\}$;

$G_2 = \min\{g_1(x, 2) + g_2(y, 2) + g_3(n_3, n_1 - x, y) | x = 0, 1, ..., [\frac{n_1}{2}]; y = 0, 1, ..., [\frac{n_2}{2}]; n_1 - x + n_3 \leq K - (n_2 - y)\}$;

$G_3 = \min\{g_1(x, 2) + g_2(y, 1) + g_3(z, n_1 - x, n_2 - y) | x = 0, 1, ..., [\frac{n_1}{2}]; y = 0, 1, ..., [\frac{n_2}{2}]; z = 1, 2..., n_3 - 1, n_1 - x + z \leq K - y; x \leq K - (n_2 - y) - (n_3 - z)\}$.

If $G = G_1$ then on machine 1 we reserve x slots starting from the time moment 0 and y slots starting from the moment $K - y$, while on machine 2 we reserve $n_1 - x$ slots starting from the time moment 0 and $n_2 - y + n_3$ slots starting from the moment $K - (n_2 - y) - n_3$.

If $G = G_2$ then on machine 2 we reserve x slots starting from the time moment 0 and y slots starting from the moment $K - y$, while on machine 1 we reserve $n_1 - x + n_3$ slots starting from the time moment 0 and $n_2 - y$ slots starting from the moment $K - n_2 + y$.

If $G = G_3$ then on machine 1 we reserve $n_1 - x + z$ slots starting from the time moment 0 and y slots starting from the moment $K - y$, while on machine 2 we reserve x slots starting from the time moment 0 and $n_2 - y + n_3 - z$ slots starting from the moment $K - n_2 + y - n_3 + z$.

In the reserved slots we assign jobs of the sets J_1, J_3 and J_2 in nonincreasing order of their weights. The time complexity of algorithm **DP-LinDec-2-UT** is $O(n_1 n_2 n_3)$.

If for each job i we have $p_i = 1$ and $w_i = 1$, then all jobs are contained only in one of the sets J_1, J_2, or J_3. In this case the complexity of the algorithm will be $O(n)$.

References

1. Wan G., Qi X.: Scheduling with Variable Time Slot Costs. Naval Research Logistics. (2010) V. 57, N 2. P. 159-171.
2. Zhao Y., Qi X., Li M.: On scheduling with non-increasing time slot cost to minimize total weighted completion time. Journal of Scheduling. -DOI 10.1007/s10951-015-0462-9.