

Dedicated Scheduling of Biprocessor Tasks to Minimize Mean Flow Time

Krzysztof Giaro, Marek Kubale, Michał Małafiejski, and Konrad Piwakowski

Technical University of Gdańsk, Foundations of Informatics Dept.
Narutowicza 11/12, 80-952 Gdańsk, Poland
kubale@pg.gda.pl

Abstract. This paper investigates the complexity of scheduling biprocessor tasks on dedicated processors to minimize mean flow time. Since the general problem is strongly NP-hard, we assume some restrictions on task lengths and the structure of associated scheduling graphs. Of particular interest are acyclic graphs. In this way we identify a borderline between NP-hard and polynomially solvable special cases.

1 Introduction

In this paper we assume that applications (or programs) of a computer operating system are multiprocessor tasks. Multiprocessor task model assumes that some tasks may require several processors at the same time. Examples of multiprocessor tasks include file transfers which require two corresponding processors simultaneously: the sender and the receiver [1], programs executed on several processors in parallel which vote for a reliable final result [6] and mutual testing of processors in biprocessor diagnostic links [12]. Another example is the problem of resource scheduling in batch manufacturing where jobs simultaneously require two resources for processing [3]. More precisely, this paper is devoted to the complexity of the following class of multiprocessor scheduling problems. A collection $J = \{J_1, \dots, J_j, \dots, J_n\}$ of n tasks has to be executed by m identical processors $M_1, \dots, M_i, \dots, M_m \in M$. Each task requires the simultaneous use of two prespecified (unique) processors for its execution (repetitions of tasks is not allowed) but each processor can execute at most one such task at a time. These tasks are referred to as biprocessor tasks. Task J_j ($j = 1, \dots, n$) requires processing during a given time p_j . The completion time of task J_j is denoted by C_j . In contrast to the previous papers on the subject we are interested in minimization of the *total completion time* $\sum C_j = \sum_{j=1}^n C_j$, which differs from the mean flow time by a factor of $1/n$ independent of the schedule. This optimization goal favors the user of computer system rather than the owner of computing facility.

In general we follow the notation and terminology of [11]. In particular, we use the well-known three-field notation scheme $\alpha|\beta|\gamma$. The first field is simply P , since we assume that the number of processors is arbitrary in all models under consideration. The second field β describes the task characteristics. In particular, the word $fix_j = 2$ is used to denote that each task is biprocessor. We also use

the notation $M = graph$ if we wish to emphasize that the incidence matrix of task system has a particular structure, e.g. $M = tree$ means that the associated scheduling graph is a tree. The third field γ stands for the optimality criterion, which in our case is $\sum C_j$. The problem considered here can be modeled as an edge-weighted multigraph $G = (V, E)$. There is a one-to-one correspondence between the vertex set $V = \{v_1, \dots, v_m\}$ and the processor set M as well as the edge set $E = \{e_1, \dots, e_n\}$ and the task set J , i.e. $e_j = v_i v_k \in E$ if and only if there is a task J_j to be executed on processors M_i and M_k . Moreover, the weight of e_j is p_j for all $j = 1, \dots, n$. Such a weighted graph will be called a scheduling graph. Any solution to the $P|fix_j = 2, M = graph|\sum C_j$ problem in k time units is equivalent to a k -coloring of the edges of the scheduling graph G with intervals of size p_j that minimizes the (interval) edge sum of G . Therefore, we will speak indifferently of colorings and schedules.

In the literature little attention has been focused on the complexity of scheduling biprocessor tasks on dedicated processors. On the whole the papers are devoted to minimizing completion time C_{\max} (cf. an overview paper by Drozdowski [4]). For example, Kubale [13] has shown that the $P|fix_j = 2|C_{\max}$ problem is NP-hard even if all processing times are equal to 1. A polynomially solvable case $P|fix_j = 2, p_j \in \{1, l\}, M = tree|C_{\max}$ is addressed by Kubale and Piwakowski [15].

2 Arbitrary Execution Times

The general $P|fix_j = 2|\sum C_j$ problem is clearly NP-hard. Therefore, we consider herein some special cases involving highly structured scheduling graphs. However, before we assume that the number of processors is part of the problem instance, following [14] we give two complexity results concerning the case $m \leq 4$.

Proposition 1. *The $P4|fix_j = 2|\sum C_j$ problem is NP-hard in the ordinary sense.*

Proposition 2. *The $P3|fix_j = 2|\sum C_j$ problem can be solved in $O(n \log n)$ time.*

The second proposition follows from the fact that in the 3-processor case biprocessor tasks are all incompatible. A well-known approach is using the shortest processing time (SPT) rule, which requires preliminary sorting of all tasks in nondecreasing order of their processing times.

2.1 Double Stars

Now let us assume that m is arbitrary and consider some special cases of acyclic scheduling graphs. The first result is due to Hoogeveen et al. [11] and involves as simple graph as a double star, in short 2-star, i.e. a tree with $m - 2$ leaves.

Theorem 1. *The $P|fix_j = 2, M = 2 - star|\sum C_j$ problem is NP-hard in the ordinary sense.*

Proof. The proof is based upon a reduction from

EVEN-ODD PARTITION (EOP):

Given a set of $2n$ positive integers $A = \{a_1, \dots, a_{2n}\}$ such that $a_i < a_{i+1}$, $i = 1, \dots, 2n - 1$. Is there a partition of A into two disjoint subsets B and $A \setminus B$ with equal sum and such that B contains exactly one of $\{a_{2i-1}, a_{2i}\}$ for each $i = 1, \dots, n$?

2.2 Paths and Cycles

The first algorithm for optimal coloring of paths was given in [10]. In the following we give our own solution because of two reasons. First, the method leads to a simple algorithm for cycles, both even and odd. Second, our algorithm leads to a simple generalization to a weighted total completion time criterion of optimality [9].

Let J_1, \dots, J_n be a sequence of tasks forming a path P_m , $m = n + 1$. The aim is to find for each J_j a starting time b_j such that:

- (1) $b_j + p_j \leq b_{j+1}$ or $b_j \leq b_{j+1} + p_{j+1}$ ($j = 1, \dots, n - 1$) and
- (2) the total completion time is as small as possible.

Without loss of generality we introduce two dummy tasks J_0 and J_{n+1} with $p_0 = p_{n+1} = 0$ for which $b_0 = b_{n+1} = 0$.

Given a schedule $B = (b_1, \dots, b_n)$ we denote $d(B) = \sum_{i=1}^n b_i$ and say that

- J_i is preceded on the left, if $b_i = C_{i-1}$;
- J_i is preceded on the right, if $b_i = C_{i+1}$;
- J_i is initial, if $b_i = 0$.

Two initial tasks J_i, J_j are said to be *adjacent*, if there is no job J_k , $i < k < j$ that is initial. Note that in any optimal schedule any task J_i is either preceded on the left or preceded on the right or else it is initial.

Lemma 1. *For any optimal schedule and any two adjacent initial tasks J_i, J_j ($i < j$) there is k , $i < k \leq j$ such that*

- (i) *all tasks J_{i+1}, \dots, J_{k-1} are preceded on the left,*
- (ii) *all tasks J_k, \dots, J_{j-1} are preceded on the right.*

Proof. By assumption no tasks J_{i+1}, \dots, J_{j-1} are initial. Thus each of them is either preceded on the left or preceded on the right. If they are all preceded on the left, the thesis follows with $k = j$. If this is not the case, let $k = \min\{r : i < r < j, J_r \text{ is preceded on the right}\}$. Property (i) is obvious. Property (ii) follows from the fact that for any two tasks J_r, J_{r+1} it is impossible that J_r is preceded on the right and simultaneously J_{r+1} is preceded on the left.

Note that it is not true that for all indices $i < k \leq j$ there is a schedule fulfilling conditions (1) and (i), (ii) of Lemma 1. It may happen that J_{k-1} and J_k would have to perform simultaneously. This situation takes place if and only if $b_{k-1} < C_k$ and $b_k < C_{k-1}$, where

$$b_{k-1} = \sum_{r=i}^{k-2} p_r, C_{k-1} = \sum_{r=i}^{k-1} p_r, b_k = \sum_{r=k+1}^j p_r, C_k = \sum_{r=k}^j p_r.$$

If $p_i = 0$ then $k = i + 1$, since otherwise J_{i+1} would be initial. Similarly, $p_j = 0$ implies $k = j$.

Let us define a predicate P excluding these cases:

$$P(i, j, k) \iff \left(\sum_{r=i}^{k-2} p_r \geq \sum_{r=k}^j p_r \vee \sum_{r=k+1}^j p_r \geq \sum_{r=i}^{k-1} p_r \right) \wedge (p_i \neq 0 \vee k = i + 1) \wedge (p_j \neq 0 \vee k = j).$$

Let

$$U(a, b) = \begin{cases} \sum_{i=0}^{b-a-1} (b-a-1)p_{a+i} & \text{if } a < b \\ \sum_{i=0}^{a-b-1} (a-b-1)p_{a-i} & \text{if } a > b \\ 0 & \text{if } a = b \end{cases}$$

The following lemma follows immediately from the definitions.

Lemma 2. $U(a, b) = b_a + \dots + b_b$ if J_a is initial and one of the following holds:

- (1) $a < b$ and J_{a+1}, \dots, J_b are preceded on the left, or
- (2) $a > b$ and J_b, \dots, J_{a-1} are preceded on the right.

Let

$$W(a, b) = \begin{cases} \min \{U(a, c-1) + U(b, c) : a < c \leq b \wedge P(a, b, c)\} & \text{for } 0 \leq a < b \leq n+1 \wedge (p_a \neq 0 \vee p_b \neq 0) \\ \infty & \text{otherwise.} \end{cases}$$

and let $D = (V, A)$ be a weighted digraph with vertex set $V = \{0, 1, \dots, n+1\}$ and arc set $A = V \times V$, where arc vw has weight $W(vw)$. For a given set of indices $I = \{0, a_1, \dots, a_k, n+1\}$, where $a_j - a_i > 1$ for $1 \leq i < j \leq k$, we define schedule B_I as a schedule having the set of indices of initial tasks equal to I and fulfilling conditions (1) and (2).

Lemma 3. *The length of path $I = (0, a_1, \dots, a_k, n+1)$, where $a_j - a_i > 1$ for $1 \leq i < j \leq k$, in digraph D is $d(B_I)$.*

Proof. Let $B = (b_1, \dots, b_n)$ be an optimal schedule with the set of indices of initial tasks $I = \{0, a_1, \dots, a_k, n+1\}$. Let $a_0 = 0, a_{k+1} = n+1$. By Lemmas 1, 2 and the definition of function W we obtain $\sum_{j=a_i}^{a_{i+1}} b_j = W(a_i, a_{i+1})$ for any two adjacent initial tasks with indices a_i, a_{i+1} , $0 \leq i \leq k$. Thus

$$\sum_{i=0}^k W(a_i, a_{i+1}) = \sum_{i=1}^n b_i + \sum_{i=0}^{k+1} b_{a_i} = \sum_{i=1}^n b_i.$$

Let B be a schedule fulfilling (1) and (2) and let I be the set of indices of initial tasks in B . Then $B = B_I$. For any other set S of such indices schedule B_S fulfills (1) and since B fulfills (2) we have $d(B) \leq d(BS)$, so $d(B) = \min\{d(B_S) : S = \{0, a_1, \dots, a_k, n+1\}, \text{ where } a_j - a_i > 1 \text{ for } 1 \leq i < j \leq k\}$. Thus the schedule B corresponds to the shortest path from vertex 0 to vertex $n+1$ in D . This leads to the following

Lemma 4. *For any optimal schedule the set I of indices of initial tasks coincides with the set of vertices of any shortest path from vertex 0 to vertex $n+1$ in digraph D . Moreover, the total completion time of optimal schedule is $\sum^* C_j = d(B_I) + p_1 + p_2 + \dots + p_n$.*

Now we are in a position to state

Theorem 2. *The $P|fix_j = 2, M = path|\sum C_j$ problem can be solved in time $O(n^2)$.*

Proof. Lemmas 1-4 lead us to the following algorithm for finding a minimum flow time solution.

Step 1. Calculate the values of $U(a, b)$ for $0 \leq a, b \leq n+1$.

Step 2. Calculate the weights $W(a, b)$ for $0 \leq a, b \leq n+1$ storing in $X(a, b)$ the corresponding value x for which $W(a, b) = U(a, x-1) + U(b, x)$.

Step 3. Construct digraph D and find the shortest path I from 0 to $n+1$ in D .

Step 4. Construct the solution B as follows.

If $i \in I$ then $b_i = 0$.

If $i \notin I$ then find two adjacent indices $a, b \in I$ such that $a < i < b$ and set

$$b_i = \begin{cases} \sum_{r=a}^{i-1} p_r & \text{if } i < X(a, b) \\ \sum_{r=i+1}^b p_r & \text{if } i \geq X(a, b) \end{cases}$$

A straightforward generalization of the algorithm given above leads to the following

Theorem 3. *The $P|fix_j = 2, M = cycle|\sum C_j$ problem can be solved in time $O(n^3)$.*

Proof. In the optimal solution at least one of the tasks must begin at time 0. Assuming this for each task J_k separately we obtain n instances of the above problem for a path of length $n-1$ with a slight modification that dummy tasks J_0 and J_n get $b_0 = b_n = 0$ and $p_0 = p_n = p_k$.

2.3 Comets

Before we study the complexity of scheduling comets in full detail, we give a straightforward claim concerning the complexity of scheduling biprocessor tasks whose scheduling graph is in the shape of star.

Theorem 4. *The $P|fix_j = 2, M = star|\sum C_j$ problem can be solved in time $O(n \log n)$.*

Graph G is a *comet* if it is a junction of a star and a path in such a way that a new edge is joining a radius of the star with an endpoint of the path. Let $G = (V, E)$ be a comet like in Fig. 1 with a weighting function $p : E \rightarrow R^+$.

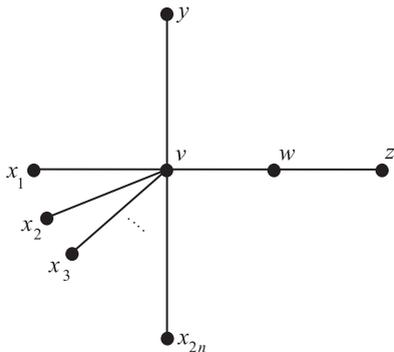


Fig. 1. One edge star extension.

Given an interval coloring $b : E \rightarrow R^+$ of G , let $S(b) = \sum_{e \in E} b(e)$. The problem of interval edge coloring of G with chromatic sum optimization is to find a coloring b^* such that $S(b^*) = \min\{S(b) : b \text{ is interval edge coloring of } G\}$. Let us consider the following problem

ONE EDGE STAR EXTENSION COLORING (1-ESEC):

Given positive integers $a_1 < a_2 < \dots < a_{2n}$, find an optimal coloring of graph G in Fig. 1 with the following weights of edges: $p(wz) = nL$, $p(vw) = \frac{2n+1}{2(n+1)}L$, $p(vy) = M$, $p(vx_i) = L + \varepsilon_i$, where for $i = 1, \dots, 2n$, $\varepsilon_i = a_i - (\frac{1}{2n} \sum_{i=1}^{2n} a_i)$, $\varepsilon = \max\{|\varepsilon_i| : 1 \leq i \leq 2n\}$, $L = 36(n+1)n^2\varepsilon$, $M = 20n^2L$.

Lemma 5. *If b is a solution to problem 1-ESEC then $b(vy) > b(vw)$ and $b(vy) > b(vx_i)$ for $i = 1, \dots, 2n$.*

Proof. Suppose b fulfills the assumptions of Lemma 5 but the thesis fails. Then for some $u \in \{x_1, \dots, x_{2n}, w\}$ we have $b(vy) < b(vu)$. Hence $S(b) > b(vu) > p(vy) = M$. Let b_2 be the coloring defined recursively: $b_2(e_1) = 0$, $b_2(e_i + 1) = b_2(e_i) + p(e_i)$ for $i = 1, \dots, 2n$, where (e_1, \dots, e_{2n+3}) is any permutation of E such that $e_{2n+3} = vy$. Then $S(b_2) < (2n + 3)b_2(vy) < (2n + 3)(3n + 1)L \leq M$. Hence $S(b_2) < S(b)$, a contradiction.

Lemma 6. *If b is a solution to problem 1-ESEC then $b(vw) > b(vz)$.*

Proof. Suppose b fulfills the assumptions of Lemma 6 but the thesis fails. Let k be the number of edges vx_i such that $b(vw) > b(vx_i)$. We then have

$$\begin{aligned} S(b) &> \sum_{i=1}^k i(L - \varepsilon) + 2 \left(k(L - \varepsilon) + \frac{(2n+1)L}{2(n+1)} \right) \\ &+ \sum_{i=k+1}^{2n} \left(i(L - \varepsilon) + \frac{(2n+1)L}{2(n+1)} \right) \\ &= L(n+1)(2n+1) - n(2n+1)\varepsilon + k \left(\frac{2n+3}{2n+2}L - 2\varepsilon \right) \end{aligned}$$

Since $L > 2\varepsilon$ so $S(b) > L(n+1)(2n+1) - n(2n+1)\varepsilon$. Let B be any n -element subset of vertices $C = \{x_i : 1 \leq i \leq 2n\}$ such that $\sum_{x \in B} p(vx) > \sum_{x \in C \setminus B} p(vx)$. Let b_2 be a coloring defined recursively: $b_2(wz) = b_2(e_1) = 0$, $b_2(e_{i+1}) = b_2(e_i) + p(e_i)$ for $i = 1, \dots, 2n+1$, where (e_1, \dots, e_n) is any permutation of B , $e_{n+1} = vw$, $(e_{n+2}, \dots, e_{2n+1})$ is any permutation of $B \setminus C$ and $e_{2n+2} = vy$. Then

$$\begin{aligned} S(b_2) &< \sum_{i=1}^n i(L + \varepsilon) + \sum_{i=n}^{2n} \left(i(L + \varepsilon) + \frac{(2n+1)L}{2(n+1)} \right) \\ &= L \left(n(2n+3) + \frac{1}{2} \right) + n(2n+2)\varepsilon \end{aligned}$$

Hence $S(b) - S(b_2) > (L - (8n^2 + 6n)\varepsilon)/2 > 0$, a contradiction.

Lemma 7. *If b is a solution to problem 1-ESEC then $b(vw) = nL$ and there are exactly n edges vx_i such that $b(vx_i) < b(vw)$.*

Proof. Suppose b is a solution to 1-ESEC. Let k be the number of edges vx_i such that $b(vx_i) < b(vw)$. Let (e_1, \dots, e_{2n}) be such a permutation of edges from $\{vx_i : 1 \leq i \leq 2n\}$ that $i < j$ iff $b(e_i) < b(e_j)$. Then $b(wz) = b(e_1) = 0$, $b(e_{i+1}) = b_2(e_i) + p(e_i)$ for $i = 1, \dots, k-1, k+1, \dots, 2n-1$, $b(e_{k+1}) = b(vw) + L(2n+1)/(2n+2)$ and $b(vy) = b(e_{2n}) + p(e_{2n})$. Let us consider two cases.

Case 1. Suppose $k < n$. Then $\sum_{1 \leq i \leq k} e_i \leq k(L + \varepsilon) \leq (n-1)(L + \varepsilon) < nL$, since $L > (n-1)\varepsilon$. Thus $b(vw) = nL$. Let b_2 be the coloring which comes from b by changing the values for two edges vw and e_{k+1} , namely: $b_2(e_{k+1}) = b_2(e_k) + p(e_k)$ and $b_2(vw) = \max\{nL, b_2(e_{k+1}) + p(e_{k+1})\}$. Hence $b_2(e_{k+1}) \leq (n-1)(L + \varepsilon)$ and $b_2(vw) \leq n(L + \varepsilon)$. Consequently, $S(b) - S(b_2) = (b(e_{k+1}) + b(vw)) - (b_2(e_{k+1}) + b_2(vw)) = (1 + L(2n+1)/(2n+2)) - (2n-1)\varepsilon > L - 2n\varepsilon > 0$, a contradiction.

Case 2. Suppose $k > n$. Then $\sum_{1 \leq i \leq k} e_i \geq k(L - \varepsilon) \geq (n+1)(L - \varepsilon) > nL$, since $L > (n+1)\varepsilon$. Thus $b(vw) = b(e_k) + p(e_k)$. Let $p = k - n$. We then have $S(b) > \sum_{i=1}^n i(L + \varepsilon) + \sum_{i=k}^{2n} (i(L + \varepsilon) + L(2n-1)/(2n+2)) = (n(2n+3) + 1/2)L - n(2n+2)\varepsilon + p(L/(2n+2) - \varepsilon) \geq (n(2n+3) + 1/2)L - n(2n+2)\varepsilon + L/(2n+2) - \varepsilon$. Let

b_2 be a coloring defined the same as in the proof of Lemma 6. Then $S(b) - S(b_2) > (L - (8n^3 + 16n^2 + 10n + 2)\varepsilon) > 0$, a contradiction. Thus we have proven that $k = n$.

Now assume that $b(vw) > nL$. Thus $p(e_1) + \dots + p(e_n) > nL$. Let b_2 be a coloring defined as follows: $b_2(wz) = b_2(e_{n+1}) = 0$, $b_2(e_{i+1}) = b_2(e_i) + p(e_i)$ for $i = 1, \dots, n - 1$, $n + 1, \dots, 2n - 1$, $b_2(e_1) = b_2(vw) + L(2n + 1)/(2n + 2)$, $b_2(vy) = b_2(e_n) + p(e_n)$ and $b_2(vw) = nL$. Then $S(b) - S(b_2) = n(b(vw) - nL) > 0$, a contradiction.

Lemma 8. *If $A = \{a_1, a_2, \dots, a_{2n}\}$, $a_1 < a_2 < \dots < a_{2n}$ and b is a solution to problem 1-ESEC for A then the answer to the EOP problem for set A is positive if and only if $S(b) = \sum_{i=1}^{n-1} (n - i)(\varepsilon_{2i-1} + \varepsilon_{2i}) + (2n^2 + 3n + 1/2)L$.*

Proof. Let (e_1, \dots, e_{2n}) be such a permutation of edge set $\{vx_i : 1 \leq i \leq 2n\}$ that $i < j$ iff $b(e_i) < b(e_j)$. Thus

$$(3) \quad S(b) = \sum_{i=1}^{n-1} (n - i)(p(e_i) + p(e_{n+1})) + (n^2 + 3n + 1/2)L + \sum_{i=1}^n p(e_{n+i}).$$

It is easy to see that

$$(4) \quad \sum_{i=1}^{n-1} (n - i)(p(e_i) + p(e_{n+1})) \geq \sum_{i=1}^{n-1} (n - i)(L + \varepsilon_{2i-1} + L + \varepsilon_{2i}).$$

By Lemma 7 we have

$$(5) \quad \sum_{i=1}^n p(e_i) \leq nL.$$

From (3), (4) and (5) it follows

$$(6) \quad S(b) \geq \sum_{i=1}^{n-1} (n - i)(L + \varepsilon_{2i-1} + L + \varepsilon_{2i}) + (n^2 + 4n + 1/2)L.$$

\Rightarrow Suppose set B is a solution to the EOP problem (defined in the proof of Theorem 1) for set A . Let (c_1, \dots, c_n) be a permutation of B such that $c_i < c_{i+1}$ for $i = 1, \dots, n - 1$. Let (d_1, \dots, d_n) be a permutation of $A \setminus B$ such that $d_i < d_{i+1}$ for $i = 1, \dots, n - 1$. Let (h_1, \dots, h_{2n}) be a permutation of $\{vx_i : 1 \leq i \leq 2n\}$ such that $p(h_i) = c_i$ and $p(h_{n+i}) = d_i$ for $i = 1, \dots, n$. Let b_2 be a coloring defined as follows: $b_2(wz) = b_2(h_1) = 0$, $b_2(h_{i+1}) = b_2(h_i) + p(h_i)$ for $i = 1, \dots, n - 1$, $n + 1, \dots, 2n - 1$, $b_2(vw) = nL$, $b_2(h_{n+1}) = b_2(bw) + L(2n + 1)/(2n + 2)$ and $b_2(vy) = b_2(h_{2n}) + p(h_{2n})$. Then $S(b_2) = \sum_{i=1}^{n-1} (n - i)(L + \varepsilon_{2i-1} + L + \varepsilon_{2i}) + (n^2 + 4n + 1/2)L$. This equality and (6) imply $S(b) = \sum_{i=1}^{n-1} (n - i)(L + \varepsilon_{2i-1} + L + \varepsilon_{2i}) + (n^2 + 4n + 1/2)L$.

\Leftarrow Suppose $S(b) = \sum_{i=1}^{n-1} (n - i)(L + \varepsilon_{2i-1} + L + \varepsilon_{2i}) + (n^2 + 4n + 1/2)L$. Thus by (3) and (4) we have $\sum_{i=1}^n p(e_{n+i}) \leq nL$. By (5) we have $\sum_{i=1}^n p(e_i) = \sum_{i=1}^n p(e_{n+i}) = nL$ and $\sum_{i=1}^{n-1} (n - i)(p(e_i) + p(e_{n+i})) = \sum_{i=1}^{n-1} (n - i)(L + \varepsilon_{2i-1} + L + \varepsilon_{2i})$. The last equality implies $\{p(e_i) : p(e_{n+i})\} = \{L + \varepsilon_{2i-1}, L + \varepsilon_{2i}\}$ for $i = 1, \dots, n$. Thus $B = \{p(e_i) : 1 \leq i \leq n\}$ guarantees that the EOP problem is answered affirmatively for set A .

Lemma 8 implies that the problem of interval sum coloring of graph G in Fig. 1 is NP-hard. More generally, for any nontrivial supergraph of a star (like a comet, for example) the problem remains NP-hard. Now, expressing our result in terms of scheduling we obtain the following

Theorem 5. *The $P|fix_j = 2, M = comet|\sum C_j$ problem is NP-hard in the ordinary sense.*

3 Complexity Classification

If $p_j = 1$ for all $j = 1, \dots, n$ then our problem reduces to optimal sum coloring of the edges of scheduling graph. We shall call this subproblem the *unit execution time (UET)* scheduling problem. In [8] Giaro and Kubale showed that optimal sum coloring of a bipartite graph is NP-hard. This implies that the borderline between P and NP-complete UET scheduling problems splits the class of bipartite graphs. Among polynomially solvable instances are trees and certain bipartite graphs. Below we give a special case of bipartite graphs that admit polynomial algorithms.

Theorem 6. *If $G = (V_1 \cup V_2, E)$ is a bipartite graph in which $\deg(x) \geq \deg(y)$ for each edge xy with $x \in V_1, y \in V_2$ then an optimal sum coloring can be found in time $O(|E|\log\Delta)$.*

In fact, every Δ -coloring of such a graph is optimal. The fastest algorithm for Δ -coloring of a bipartite graph is due to Cole et al. [2].

The strongest polynomially solvable is the case of trees. Recently Zhou and Nishizeki [16] showed

Theorem 7. *If G is a tree then an optimal sum coloring of the branches of G can be found in $O(|V|^{2.5}\log|V|)$ time.*

It is easy to see that the UET problem becomes linearly solvable if the scheduling graph is a double star or a comet.

The main results of our investigation are summarized in Table 1. Entries in the table are either "sNPh" for strongly NP-hard, "NPh" for ordinary NP-hard or $O(\cdot)$ for an upper bound on the complexity derived from the best polynomial-time optimization algorithm known for the corresponding subproblem.

Table 1. Complexity classification for dedicated scheduling of biprocessor tasks (AET - arbitrary execution times, UET - unit execution times).

Scheduling graphs	AET	UET	References
bipartite	sNPh	sNPh	[8]
trees	sNPh	$O(n^{2.5}\log n)$	[8], [16]
double stars	NPh	$O(n)$	[11], [7]
comets	NPh	$O(n)$	Theorem 5, [7]
cycles	$O(n^3)$	$O(n)$	Theorem 3
paths	$O(n^2)$	$O(n)$	Theorem 2
stars	$O(n\log n)$	$O(n)$	Theorem 4

References

1. Coffman, Jr., E.G., Garey, M.R., Johnson, D.S., LaPaugh, A.S.: Scheduling file transfers. *SIAM J. Comput.* **14** (1985) 744-780
2. Cole, R., Ost, K., Schirra, S., Edge-coloring bipartite graphs in $O(E \log D)$ time. *Combinatorica* **21** (2001) 5-12
3. Dobson, G., Karmarkar, U.S.: Simultaneous resource scheduling to minimize weighted flow times. *Oper. Res.* **37** (1989) 592-600
4. Drozdowski, M.: Scheduling multiprocessor tasks - An overview. *Euro. J. Oper. Res.* **94** (1996) 215-230
5. Drozdowski, M., Dell'Olmo, P.: Scheduling multiprocessor tasks for mean flow time criterion. *Comp. Oper. Res.* **27** (2000) 571-585
6. Gehring, E.F., Siewiorek, D.P., Segall, Z.: *Parallel Processing: The Cm* Experience*. Digital Press, Bedford (1987)
7. Giaro, K., Kubale, M., Małafiejski, M., Piwakowski, K.: Chromatic scheduling of dedicated 2-processor UET tasks to minimize mean flow time. *Proc. ETFA'99, Barcelona* (1999) 343-347
8. Giaro, K., Kubale, M.: Edge-chromatic sum of trees and bounded cyclicity graphs. *Inf. Process. Lett.* **75** (2000) 65-69
9. Giaro, K., Kubale, M., Piwakowski, K.: Complexity results on open shop scheduling to minimize weighted mean flow time of operations. (in preparation)
10. Halldórsson, M.M., Kortsarz, G., Proskurowski, A., Salman, R., Shachnai, H., Telle, J.A., Multicoloring trees. *Computing and Combinatorics Conference, Tokyo* (1999), *Lecture Notes in Computer Science* **1627** (1999) 271-280
11. Hoogeveen, J.A., van de Velde, S.L., Veltman, B.: Complexity of scheduling multiprocessor tasks with prespecified processor allocations. *Disc. Appl. Math.* **55** (1994) 259-272
12. Krawczyk, H., Kubale, M.: An approximation algorithm for diagnostic test scheduling in multicomputer systems. *IEEE Trans. Comput.* **34** (1985) 869-872
13. Kubale, M.: The complexity of scheduling independent two-processor tasks on dedicated processors. *Inf. Process. Lett.* **24** (1987) 141-147
14. Kubale, M.: Preemptive versus nonpreemptive scheduling of biprocessor tasks on dedicated processors. *Euro. J. Oper. Res.* **94** (1996) 242-251
15. Kubale, M., Piwakowski, K.: A linear algorithm for edge coloring of binomial trees. *Disc. Math.* **150** (1996) 247-256
16. Zhou, X., Nishizeki, T.: Algorithms for the cost edge-coloring of trees. *Lecture Notes in Computer Science* **2108** (2001) 288-297