Open Journal of Applied Sciences
Supplement: 2012 world Congress on Engineering and Technology

Algorithm for Cost Non-preemptive Scheduling of Partial k-Trees

Yiming Li
Center of Modern Educational Technology, Wenzhou University, Chashan Educational Area, Wenzhou, Zhejiang, 325035, P. R. China
Email: ymli@wzu.edu.cn

Zhiqian Ye
College of Biomedical Engineering and Science Instrument, Zhejiang University, 38 Zheda Road, Hangzhou, Zhejiang, 310027, P. R. China.
Email: yezhiqian@zju.edu.cn

Xiao Zhou
Graduate School of Information Sciences, Tohoku University, Aoba-yama 6-6-05, Sendai, 980-8579, Japan.
Email: zhou@ecei.tohoku.ac.jp

***Abstract*—Let G be a graph, in which each vertex (job) v has a positive integer weight (processing time) $p(v)$ and each edge (u, v) represented that the pair of jobs u and v cannot be processed in the same slot. In this paper we assume that every job is non-preemptive. Let $C = \{1, 2, \cdots\}$ be a color set. A multicoloring (scheduling) F of G is to assign each job v a set of $p(v)$ consecutive positive integers (processing consecutive time slots) in C so that any pair of adjacent vertices receive disjoint sets. Such a multicoloring is called a non-preemptive scheduling. The cost non-preemptive scheduling problem is to find an optimal multicoloring of G, that is, a multicoloring F such that $\sum_{v \in V} \omega(F(v))$ is minimum among all multicolorings of G, where ω is a cost function of processing time slots which assigns a real number to a set of consecutive positive integers. In this paper we give a polynomial-time algorithm to find an optimal multicoloring F of a given partial k-tree. The algorithm takes time $O(n|C|^{k+2})$ if n is the number of vertices and C is the color set.**

Keywords: Coloring, Non-preemptive scheduling, Partial k-tree

I. Introduction

Let $G = (V, E)$ be a graph with vertex set V and edge set E. A *vertex-coloring* of a graph G is to color all vertices so that any pair of adjacent vertices are colored with different colors. Let each vertex v of G have a positive integer weight $p(v)$. Let $C = \{1, 2, \cdots\}$ be a set of consecutive positive integers as a color set, and let 2^C be the power set of C. Then a *multicoloring* F of G is a mapping from V to 2^C which assigns each vertex $u \in V$ a set $F(u)$ of $p(u)$ consecutive integers in C so that $F(v) \cap F(w) = \phi$ for any pair of adjacent vertices $v, w \in V$. Thus the ordinary vertex-coloring is merely a multicoloring for the special case where $p(v) = 1$ for every vertex v. The *multichromatic number* $\chi_p(G)$ of G is the minimum number of colors required for a multicoloring of G, that is,

$$\chi_p(G) = \min\left\{|C| : \ G \text{ has a multicoloring } F : V \to 2^C \ \right\}.$$

The *multicoloring problem* is to compute the multichromatic number $\chi_p(G)$ of a given graph G. Since the vertex-coloring problem is NP-hard, the multicoloring problem is of course NP-hard and hence it is very unlikely that the multicoloring problem can be efficiently solved for general graphs. However, there may exist an efficient algorithm to solve the multicoloring problem for a restricted class of graphs. Indeed, the problem can be solved for trees in time $O(n)$ [3], [4], for triangulated graphs in time $O(n^2)$ [3], [4], for perfect graphs in time $O(mn)$ [4], for series-parallel graphs in time $O(nW)$ [13], and for partial k-trees in time $O(nW^{2^{2k+3}} \log_2 W)$ [5], where m is the number of edges, n is the number of vertices of a given graph G, and W is the maximum vertex weight, that is, $W = \max_{v \in V} p(v)$.

In this paper we consider a "cost multicoloring problem." Let ω be a cost function which assigns a real positive number $\omega(S)$ to each set S of consecutive integers in C, the *cost multicoloring problem* is to find an *optimal multicoloring* of G, that is, a multicoloring F such that $\sum_{v \in V} \omega(F(v))$ is minimum among all multicolorings of G. An optimal multicoloring does not always use the minimum number $\chi_p(G)$ of colors. For example (see Fig. 1), suppose that $p(v) = 1$ for every vertex $v \in V$ and that $\omega(\{1\}) = 1$ and $\omega(\{i\}) = 5$ for each index $i \geq 2$, then the graph G with $\chi_p(G) = 3$ in Fig. 1(a) can be uniquely colored by the three cheapest colors $1, 2$ and 3 as in Fig. 1(a), but this coloring is not optimal; an optimal coloring of G uses of four cheapest colors $1, 2, 3$ and 4, as illustrated in Fig. 1(b).

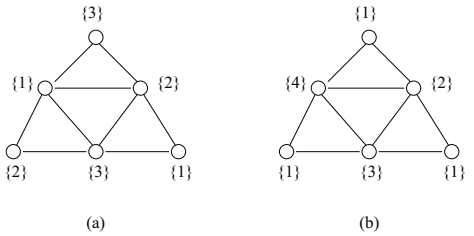

Fig. 1. (a) Multicoloring using $\chi_p(G) = 3$ colors, and (b) optimal multicoloring using $\chi_p(G) + 1 = 4$ colors, where $\omega(\{1\}) = 1$ and $\omega(\{2\}) = \omega(\{3\}) = \omega(\{4\}) = 5$.

The cost multicoloring problem has natural application in scheduling theory [6], including job scheduling, resource allocation in cloud computing (see, e.g., [9], [11]), and VLSI layout problem (see, e.g., [10]). Consider a set V of non-preemptive jobs such that each job $v \in V$ needs a total of $p(v)$ unites of time to be finished and there are several pairs

of jobs which cannot be executed simultaneously. Suppose that if a task v executed from the cth time slot takes $p(u)$ units of time, then it takes the cost $\omega([c, c + p(u) - 1])$, where $[c, c + p(v) - 1]$ is the set of $p(u)$ consecutive integers c, $c + 1$, $\cdots$, $c + p(u) - 1$. Then we wish to find a non-preemptive schedule that minimizes the total cost, and hence this corresponds to the cost multicoloring problem as follows. A multicoloring F of G corresponds to a schedule, where the vertex v receives a set $F(v)$ of $p(v)$ consecutive colors in which each color $c \in C$ represents the collection of tasks, each of which starts at the cth time slot and ends at the $(c + p(v) - 1)$th time slot with the cost $\omega(F(v))$. Our goal is to find a non-preemptive schedule of the minimum cost of executing all jobs.

The minimum sum coloring problem was introduced in [7], [8], is merely the cost multicoloring problem for the special case $\omega(\{c\}) = c$ for each integer $c \geq C$. Since the minimum sum coloring problem is NP-hard [7], the cost multicoloring problem is also NP-hard. However, one may expect that the problem can be solved efficiently for some restricted class of graphs, say partial k-trees, that is, the class of graphs of treewidth bounded by a fixed constant k. Indeed in this paper we give a polynomial-time algorithm to solve the multcoloring problem for partial k-trees.

II. Preliminaries

In this section, we give some definitions. Let $G = (V, E)$ be a simple graph without selfloops and multiple edges. Let $V(G)$ and $E(G)$ be the sets of vertices and edges, respectively. An edge joining vertices u and v is denoted by (u, v). We denote by n and m the number of vertices and edges in G, respectively. Let $C = \{1, 2, \cdots\}$ be a color set. Every vertex v is assigned to a positive integer $p(v)$ that is the number of consecutive colors assigned to v. Let ω be a cost function from consecutive integers (colors) to a positive real number. A *p-multicoloring* $F : V \to 2^C$, simply called *p-coloring*, of G must satisfy the following (a) and (b):

(a) for every vertex $v \in V$, the set $F(v)$ consists of $p(v)$ consecutive positive integers in C, and hence $|F(v)| = p(v)$; and
(b) for every edge $(u, v) \in E$, all integers in $F(u)$ differ from those in $F(v)$.

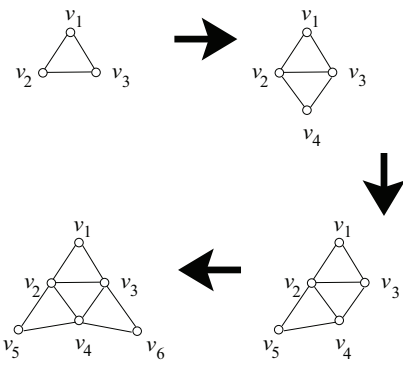

Fig. 2. A process of generating a 2-tree.

We define

$$\omega(G, F, p) = \sum_{v \in V} \omega(F(v))$$

called the *cost of the p-coloring F of G*. Let $\omega(G, p)$ be the minimum cost among all p-colorings of G. A p-coloring F is called *optimal* if $\omega(G, F, p) = \omega(G, p)$. The *p-coloring problem* is to compute $\omega(G, p)$ for a given graph G with weight $p(v)$ for each vertex $v \in V$ and a cost function ω.

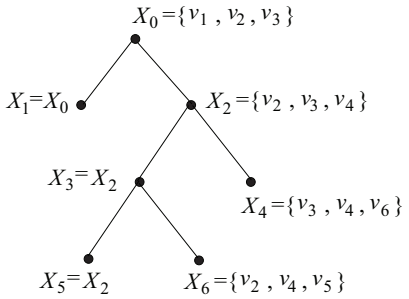

Fig. 3. Tree-decomposition of the partial 2-tree in Fig. 1.

Assume that k is a bounded positive integer. A k-tree is defined recursively as follows [1]:

(1) A complete graphs with $k + 1$ vertices is a k-tree.
(2) If G is a k-tree and k vertices induce a complete subgraph of G, then a graph obtained from G by adding a new vertex and joining it with each of the k vertices is a k-tree.

Any subgraph of a k-tree is called a *partial k-tree*. Thus a partial k-tree $G = (V, E)$ is a simple graph, and $|E| < kn$. Figure 2 illustrates a process of generating 2-trees. The graph in Fig. 1 is a partial 2-tree since it is a subgraph of the last 2-tree in Fig. 2. A binary tree $T = (V_T, E_T)$ is called a *tree-decomposition* of a partial k-tree $G = (V, E)$ if T satisfies the following conditions (a)–(e):

(a) every node $X \in V_T$ of T is a subset of V, and $|X| = k + 1$;
(b) $\bigcup_{x \in V_t} X = V$;
(c) for each edge $e = (u, v)$ of G, T has a leaf $X \in V_T$ such that $u, v \in X$;
(d) if node X_p lies on the path in T from node X_q to node X_r, then $X_q \cap X_r \subseteq X_p$; and
(e) each internal node X_i of T has exactly two children, say X_l and X_r, and $|X_l \cap X_r| = k$ and either $X_i = X_l$ or $X_i = X_r$.

We will use notions as: *leaf*, *node*, *child* and *root* in their usual meaning. Figure 3 illustrates a tree-decomposition T of the partial 2-tree in Fig. 1. We denote by X_0 the root of a tree-decomposition. Since a tree-decomposition T of a partial k-tree G can be found in linear time [1], we may assume that a partial k-tree G and its tree-decomposition T are given. The number of nodes of T constructed by the algorithm in [1] is $O(n)$.

III. Polynomial-Time Algorithm

Our algorithm for solving the cost non-preemptive scheduling problem on partial k-trees is based on the idea of "dynamic programming method." A dynamic programming (DP) method is a standard one to solve a combinatorial problem on partial k-trees [1], [12]. We also use it, and compute the minimum cost $\omega(G,p)$ of a given partial k-tree with its tree-decomposition T by the "bottom-up tree computation." We assume from now on that k is a bounded integer. Although we give an algorithm to compute $\omega(G,p)$ it can be easily modified so that it actually finds an optimal p-coloring F with the cost $\omega(G,p)$.

Let T be a tree-decomposition of a partial k-tree G. We define a vertex set $V_i \subseteq V(G)$ and an edge set $E_i \subseteq E(G)$ for each node X_i of T as follows: if X_i is a leaf, then let $V_i = X_i$ and $E_i = \{(u,v) \in E(G) : u,v \in X_i\}$; if X_i is an internal node with children X_l and X_r, then let $V_i = V_l \cup V_r$ and $E_i = E_l \cup E_r$. Note that $V_l \cap V_r \subseteq X_i$. We denote by G_i the graph with vertex set V_i and edge set E_i. Then graphs $G_l = (V_l, E_l)$ and $G_r = (V_r, E_r)$ share common vertices only in X_i because of the property (e) of a tree-decomposition.

Let $X_i = \{v_1, v_2, \cdots, v_{k+1}\}$ be a node of T. Let $\mathcal{S} = (S_1, S_2, \cdots, S_{k+1}) \in (2^C)^{k+1}$ be a color set. A p-coloring F of G satisfying $F(v_j) = S_j$ for each j, $1 \le j \le k+1$, is called an $\mathcal{S}$-*coloring*. We define

$$\omega(G_i, p; \mathcal{S}) = \min_F \omega(F,p),$$

where the minimum is taken over all $\mathcal{S}$-colorings F of G_i. Let $\omega(G_i,p;\mathcal{S}) = \infty$ if there is no such $\mathcal{S}$-coloring. Since S_j, $1 \le j \le k+1$, is a set of $p(v_j)$ consecutive integers, the number of S_j is at most $|C|$ and hence $\omega(G_i,p;\mathcal{S}) \neq \infty$ is at most $|C|^{k+1}$. If X_0 is the root of T, then clearly

$$\omega(G,p) = \omega(X_0,p) = \min_{\mathcal{S}} \omega(G_0,p;\mathcal{S}),$$

and hence $\omega(G,p)$ can be computed in time $O(|C|^{k+1})$ from all $\omega(G_0,p;\mathcal{S}) \neq \infty$. Therefore we need to compute all the values $\omega(G_i,p;\mathcal{S})$ for all $\mathcal{S} \in (2^C)^{k+1}$ from leaves to root X_0. Thus the DP table for each node X_i of T consists of at most $|C|^{k+1}$ values $\omega(G_i,p;\mathcal{S}) \neq \infty$, $\mathcal{S} \in (2^C)^{k+1}$, and has size $|C|^{k+1}$. Furthermore, one can recursively compute $\omega(G_i,p;\mathcal{S})$ as follows.

Consider the case where $X_i = \{v_1, v_2, \cdots, v_{k+1}\}$ is a leaf of T, then clearly

$$\omega(X_i,p;\mathcal{S}) = \sum_{1 \le j \le k+1} \omega(S_j)$$

if S_i consists of $p(v_i)$ consecutive integers for each $1 \le i \le k+1$ and, for each pair of vertices $v_j, v_{j'} \in X$ satisfying $(v_j, v_{j'}) \in E$, $S_j \cap S_{j'} = \emptyset$; otherwise $\omega(G_i,p;\mathcal{S}) = \infty$. Then we have the following lemma.

Lemma 3.1: Let X_i be a leaf of a tree-decomposition T of a partial k-tree $G = (V,E)$. Then all $\omega(G_i,p;\mathcal{S}) \neq \infty$, $\mathcal{S} \in (2^C)^{k+1}$, can be computed in time $O(|C|^{k+1})$.

Consider the other case where $X_i = \{v_1, v_2, \cdots, v_{k+1}\}$ is an internal node of T. In this case, let $X_l = \{v_{l1}, v_{l2}, \cdots, v_{l(k+1)}\}$ and $X_r = \{v_{r1}, v_{r2}, \cdots, v_{r(k+1)}\}$ be the children of X_i, and assume without loss of generality that $X_i = X_l$ and $v_j = v_{lj} = v_{rj}$ for each j, $1 \le j \le k$, because of the property (e) of the tree-decomposition. Then, for each pair of $\mathcal{S}_l = (S_{l1}, S_{l2}, \cdots, S_{l(k+1)})$ and $\mathcal{S}_r = (S_{r1}, S_{r2}, \cdots, S_{r(k+1)})$ in $(2^C)^{k+1}$, let

$$\omega(G_i,p;\mathcal{S}_l,\mathcal{S}_r) = \omega(G_i,p;\mathcal{S}_l) + \omega(G_r,p;\mathcal{S}_r) - \sum_{1 \le j \le k} \omega(S_{lj}) \tag{1}$$

if $S_{lj} = S_{rj}$ for each j, $1 \le j \le k$; otherwise, $\omega(G_i,p;\mathcal{S}_l,\mathcal{S}_r) = \infty$. Then we have the following lemma.

Lemma 3.2: Let X_i be an internal node of T with two children X_l and X_r. Assume that $X_i = X_l$. Then

$$\omega(G_i,p;\mathcal{S}) = \min_{\mathcal{S}' \in (2^C)^{k+1}} \omega(G_i,p;\mathcal{S},\mathcal{S}') \tag{2}$$

for each $\mathcal{S} \in (2^C)^{k+1}$. Furthermore, all $\omega(G_i,p;\mathcal{S}) \neq \infty$, $\mathcal{S} \in (2^C)^{k+1}$, can be computed in time $O(|C|^{k+2})$.

Proof: Let $\mathcal{S} = (S_1, S_2, \cdots, S_{k+1}) \in (2^C)^{k+1}$. We first prove that

$$\omega(G_i,p;\mathcal{S}) \ge \min_{\mathcal{S}' \in (2^C)^{k+1}} \omega(G_i,p;\mathcal{S},\mathcal{S}'). \tag{3}$$

If $\omega(G_i,p;\mathcal{S}) = \infty$, then trivially Eq. (3) holds true. One thus assume that $\omega(G_i,p;\mathcal{S}) \neq \infty$, and hence there is an $\mathcal{S}$-coloring F of G_i such that $\omega(F) = \omega(G_i,p;\mathcal{S})$ and $S_j = F(v_j)$ for each j, $1 \le j \le k+1$, Let $F_l(v) = F(v)$ for each vertex $v \in V(G_l)$. Since F is a p-coloring of G_i, F_l is p-colorings of G_l. Let $X_l = X_i = \{v_{l1}, v_{l2}, \cdots, v_{l(k+1)}\}$. Let $S_{lj} = F_l(v_{lj})$ for each j, $1 \le j \le k+1$, and let $\mathcal{S}_l = (S_{l1}, S_{l2}, \cdots, S_{l(k+1)})$. Then F_l is an $\mathcal{S}_l$-coloring of G_l, and hence we have

$$\omega(G_l,p;\mathcal{S}_l) \le \omega(F_l). \tag{4}$$

Similarly, let $F_r(v) = F(v)$ for each vertex $v \in V(G_r)$. Then F_r is a p-coloring of G_r. Let $X_r = \{v_{r1}, v_{r2}, \cdots, v_{r(k+1)}\}$, let $S_{rj} = F_r(v_{rj})$ for each j, $1 \le j \le k+1$, and let $\mathcal{S}_r = (S_{r1}, S_{r2}, \cdots, S_{r(k+1)})$. Then F_r is an $\mathcal{S}_r$-coloring of G_r, and hence we have

$$\omega(G_r,p;\mathcal{S}_r) \le \omega(F_r). \tag{5}$$

Since $X_i = X_l$, we have $\mathcal{S} = \mathcal{S}_l$. Since $|X_l \cap X_r| = k$, one may assume that $v_{lj} = v_{rj}$ for each j, $1 \le j \le k$. Therefore we have

$$\begin{aligned}
\omega(G_i,p,\mathcal{S}) &= \omega(F) \\
&= \omega(F_l) + \omega(F_r) - \sum_{1 \le j \le k} \omega(F(v_{lj})) \\
&\ge \omega(G_l,p,\mathcal{S}_l) + \omega(G_r,p,\mathcal{S}_r) - \sum_{1 \le j \le k} \omega(S_j) \\
&= \omega(G_i,p;\mathcal{S},\mathcal{S}_r) \\
&\ge \min_{\mathcal{S}' \in (2^C)^{k+1}} \omega(G_i,p;\mathcal{S},\mathcal{S}'),
\end{aligned}$$

completing to verify Eq. (3).

We then prove that

$$\omega(G_i, p; \mathcal{S}) \leq \min_{\mathcal{S}' \in (2^C)^{k+1}} \omega(G_i, p; \mathcal{S}, \mathcal{S}'). \quad (6)$$

If $\min_{\mathcal{S}' \in (2^C)^{k+1}} \omega(G_i, p; \mathcal{S}, \mathcal{S}') = \infty$, then trivially Eq. (6) holds true. One thus assume that

$$\min_{\mathcal{S}' \in (2^C)^{k+1}} \omega(G_i, p; \mathcal{S}, \mathcal{S}') \neq \infty,$$

and hence there are $\mathcal{S}$-coloring F_l of G_l and $\mathcal{S}'$-coloring F_r of G_r such that $\omega(G_l, p, \mathcal{S}) = \omega(F_l)$, $\omega(G_r, p, \mathcal{S}') = \omega(F_r)$ and

$$\min_{\mathcal{S}' \in (2^C)^{k+1}} \omega(G_i, p; \mathcal{S}, \mathcal{S}') = \omega(F_l) + \omega(F_r) - \sum_{1 \leq j \leq k} \omega(S_{lj}). \quad (7)$$

Let $F(v) = F_l(v)$ if $v \in V(G_l)$; otherwise, let $F(v) = F_r(v)$. Then trivially F is an $\mathcal{S}$-coloring of G_i. Therefore we have

$$\begin{aligned} \omega(G_i, p; \mathcal{S}) &\leq \omega(F) \\ &= \omega(F_l) + \omega(F_r) - \sum_{1 \leq j \leq k} \omega(S_{lj}) \\ &= \min_{\mathcal{S}' \in (2^C)^{k+1}} \omega(G_i, p; \mathcal{S}, \mathcal{S}'), \end{aligned}$$

completing to verify Eq. (6).

We finally show that all $\omega(G_i, p; \mathcal{S}) \neq \infty$, $\mathcal{S} \in (2^C)^{k+1}$, can be computed in time $O(|C|^{k+2})$. Consider $\mathcal{S} = (S_1, S_2, \cdots, S_{k+1})$ such that $\omega(G_i, p; \mathcal{S}) \neq \infty$. Then clearly $|S_j| = p(v_j)$ for each j, $1 \leq j \leq k+1$. So the number of all $S_j \subseteq C$ consisting of $p(v_j)$ consecutive integers is at most $|C|$. We thus have all $\omega(G_i, p; \mathcal{S}) \neq \infty$ at most $O(|C|^{k+1})$. For each $\mathcal{S} = (S_1, S_2, \cdots, S_{k+1})$, the number of all $\mathcal{S}' = (S'_1, S'_2, \cdots, S'_{k+1})$ satisfying $S_j = S'_j$, $1 \leq j \leq k$, is at most $|C|$. Therefore by Eqs. (1) and (2) we can compute all $\omega(G_i, p; \mathcal{S}) \neq \infty$, $\mathcal{S} \in (2^C)^{k+1}$, in time $O(|C|^{k+2})$. ■

By Lemmas 3.1 and 3.2 we obtain the following straightforward algorithm. Let T be a tree-decomposition of a partial k-tree G, and let X_i be a node in T. Then the following procedure Color(G_i) computes $\omega(G_i, p; \mathcal{S})$ for all $\mathcal{S} \in (2^C)^{k+1}$. We first call procedure Color(G_0) for the root X_0 of T, and then compute $\omega(G, p) = \min_{\mathcal{S} \in (2^C)^{k+1}} \omega(G_0, p; \mathcal{S})$, where $G = G_0$.

Lemmas 3.1 and 3.2 imply that Lines 2 and 7 in the algorithm above can be done in time $O(|C|^{k+2})$. Since T has at most n leaves and hence $O(n)$ nodes in total. Consequently the recursive calls occur at most $O(n)$ times during the execution of Color(G_0). Thus the total running time of the algorithm is $O(n|C|^{k+2})$. We thus have the following theorem.

Theorem 3.3: The cost multicoloring problem can be solved in time $O(n|C|^{k+2})$ for partial k-trees G, where n is the number of vertices in G and C is the color set.

Algorithm 1 Color(G_i)

1: **if** X_i is a leaf **then**
2: compute $\omega(G_i, p, \mathcal{S})$ for all $\mathcal{S} \in (2^C)^{k+1}$ by Lemma 3.1
3: **else**
4: let X_l and X_r be the two children of X_i of T
5: Color(G_l)
6: Color(G_r)
7: compute $\omega(G_i, p, \mathcal{S})$ for all $\mathcal{S} \in (2^C)^{k+1}$ from all $\omega(G_l, p, \mathcal{S}_l)$ and $\omega(G_r, p, \mathcal{S}_r)$ by Lemma 3.2
8: **end if**

IV. Conclusion

In this paper we introduced the cost non-preemptive scheduling problem and showed that the problem can be solved in polynomial time for partial k-trees. The cost preemptive scheduling problem can be similarly defined for the case where every job may be preemptive. However, it is still open for partial k-trees.

References

[1] H. L. Bodlaender, Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees, J. Algorithms 11, pp. 631–643, 1990.

[2] R. B. Borie, R. G. Parker and C. A. Tovey, Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families, Algorithmica 7, pp. 555–581, 1992.

[3] E. Balas and J. Xue, Minimum weighted colouring of triangulated graphs, with application to maximum weight vertex packing and clique finding in arbitrary graphs, SIAM. J. Comput., 20, pp. 209–221, 1991.

[4] C.T. Hoáng, Efficient algorithms for minimum weighted colouring of some classes of perfect graphs, Discrete Applied Mathematics, 55, pp. 133–143, 1994.

[5] T. Ito, T. Nishizeki and X. Zhou, Algorithms for multicolorings of partial k-trees, IEICE Trans. Inf& Syst., E86-D, pp 191–200, 2003.

[6] D. Karger, C. Stein and J. Wein, "Scheduling algorithms," in Algorithms and Theory of Computation, Handbook, ed. MJ. Atallah, CRC Press, 1998.

[7] E. Kubicka and A.J. Schwenk, An introduction to chromatic sum, Proc. 17th Annual ACM Computer Science Conf., pp. 39–45, 1989.

[8] A. Moukrim, K. Sghiouer, C. Lucet and Y. Li, Lower bounds for the minimal sum coloring problem, Electronic Notes in Discrete Mathematics, vol. 36, pp. 663–670, 2010.

[9] A. Oproscu and T. Kielmann, Bag-of-tasks scheduling under budget constraints, In Cloud Computing Technology and Science, 2010 IEEE Second International Conference, pp. 351–359, 2010.

[10] A. Sen, H. Deng and S. Guha, On a graph partition problem with an application to VLSI layout, Information Processing Letters, 43, pp. 87-94, 1992.

[11] W. Shi and B. Hong, Resource allocation with a budget constraint for computing independent task in the cloud. In Cloud Computing Technology and Science, 2010 IEEE Second International Conference, pp. 327–334, 2010.

[12] S. Isobe, X. Zhou and T. Nishizeki, A polynomial-time algorithm for finding total colorings of partial k-trees, International Journal of Foundations of Computer Science, 10, pp. 171–194, 1999.

[13] X. Zhou and T. Nishizeki, Multicolorings of series-parallel graphs, Algorithmica, 38, pp. 271–297, 2004.