Discussiones Mathematicae
Graph Theory 29 (2009) 361–376

EFFICIENT LIST COST COLORING OF VERTICES AND/OR EDGES OF BOUNDED CYCLICITY GRAPHS[*]

Krzysztof Giaro and Marek Kubale

Gdańsk University of Technology
Department of Algorithms and System Modeling
Narutowicza 11/12, 80–952 Gdańsk, Poland

e-mail: kubale@eti.pg.gda.pl

Abstract

We consider a list cost coloring of vertices and edges in the model of vertex, edge, total and pseudototal coloring of graphs. We use a dynamic programming approach to derive polynomial-time algorithms for solving the above problems for trees. Then we generalize this approach to arbitrary graphs with bounded cyclomatic numbers and to their multicolorings.

Keywords: cost coloring, dynamic programming, list coloring, NP-completeness, polynomial-time algorithm.

2000 Mathematics Subject Classification: 05C15.

1. Introduction

The problem that we are going to consider is strongly related to Optimum Cost Partition [4] but it also has other names. It generalizes some basic graph coloring problems like sum edge coloring for example [1]. The problem is motivated by potential applications in multiprocessor task scheduling to minimize mean flow time [2] and by minimum wire length routing in VLSI circuit design [11].

All graphs considered in this paper are simple and connected. Let $G = G(V, E)$ be such a graph with $n = |V|$ vertices and $m = |E|$ edges. By $deg(v)$ we mean the degree of vertex $v \in V$, and Δ stands for the maximum

[*]Supported in part by MNiSW grant N516 029 31/2941.

vertex degree. Symbol $E_v = \{e \in E : v \in e\}$ is used to mean the set of all edges incident to $v \in V$. By $\gamma(G)$ we denote the cyclomatic number of G, i.e., $\gamma(G) = m - n + 1$. Since we consider many different graphs and other objects simultaneously, sometimes we use the symbol of graph to which the corresponding notion is relevant, e.g. $deg_G(v)$. $P_{fin}(A)$ denotes the family of all finite subsets of set A. By N we mean the set of natural numbers and $N_0 = N \cup \{0\}$ is the set of non-negative integers. We begin with recalling standard definitions from the chromatic theory of graphs.

Definition 1.1. Let $G(V, E)$ be any graph. A function $c : V \mapsto N$ is called a *vertex coloring* of G if $c(u) \neq c(v)$ whenever $\{u, v\} \in E$.

In symbols, a function c is vertex coloring if $\forall_{\{u,v\}\in E}\ c(u) \neq c(v)$.

Definition 1.2. A function $c : E \mapsto N$ is called an *edge coloring* of G if $c(e) \neq c(f)$ for any two adjacent edges $e, f \in E$.

More formally, a function c is edge coloring if $\forall_{e,f\in E}\ e \cap f \neq \emptyset \Rightarrow c(e) \neq c(f)$.

Definition 1.3. A function c: $V \cup E \mapsto N$ is called a *pseudototal coloring* of G if its restriction $c_{|E}$ is an edge coloring and $c(e) \neq c(v)$ whenever $v \in e$.

In symbols, $\forall_{e,f\in E}\ e \cap f \neq \emptyset \Rightarrow c(e) \neq c(f)$ and $\forall_{v\in V}\forall_{e\in E}\ v \in e \Rightarrow c(e) \neq c(v)$.

Definition 1.4. A function c: $V \cup E \mapsto N$ is called a *total coloring* of G if it is pseudototal and its restriction $c_{|V}$ is a vertex coloring.

Constrains on function c are the strongest for total coloring, namely $\forall_{\{u,v\}\in E}\ c(u) \neq c(v)$, $\forall_{e,f\in E}\ e \cap f \neq \emptyset \Rightarrow c(e) \neq c(f)$ and $\forall_{v\in V}\forall_{e\in E}\ v \in e \Rightarrow c(e) \neq c(v)$.

In our general model to different elements x of graph G (vertices, edges, or vertices and edges simultaneously) there are assigned arbitrary cost functions $f_x : N \mapsto N_0$ specifying the cost of coloring x with a particular color $i \in N$. Informally, the sum of costs among all such x is the cost of coloring of a particular set (of vertices, edges, or vertices and edges). We will attempt to minimize this parameter. Another restriction that we consider in the paper is connected with lists $L(x) \subseteq N$ specifying which colors are available to x.

Definition 1.5. Let it be a graph $G(V, E)$ and a collection of lists $L : V \mapsto P_{fin}(N)$ (resp. $L : E \mapsto P_{fin}(N)$, $L : V \cup E \mapsto P_{fin}(N)$). A vertex coloring (resp. edge coloring, (pseudo)total coloring) c of G is called *L-list coloring* if $\forall_{v\in V}\ c(v) \in L(v)$ (resp. $\forall_{e\in E}\ c(e) \in L(e)$, $\forall_{x\in V\cup E}\ c(x) \in L(x)$).

Given a graph G and a collection of lists L, the problem of existence of a coloring of the above type is called the *list coloring problem* for vertices, edges, etc.

Definition 1.6. Let it be a graph $G(V, E)$ and cost functions f_x computable in time $O(1)$ for all $x \in V$ (resp. $x \in E$, $x \in V \cup E$). By the *cost* of vertex coloring (resp. edge coloring, (pseudo)total coloring) c we mean the sum $\sum_{v\in V} f_v(c(v))$ (resp. $\sum_{e\in E} f_e(c(e))$, $\sum_{x\in V\cup E} f_x(c(x))$).

Given a graph G with functions f_x on the vertices and/or edges, it is natural to ask for a coloring whose sum of costs is as small as possible. We shall call such a coloring as *optimal*. At last we combine both models, that is, we consider a graph whose elements x are assigned both available colors $L(x)$ and cost functions f_x (which are defined on $L(x)$ rather than on the whole N) and we ask for a cheapest coloring among all legal L-list colorings (if one exists). We shall call such a problem the *list cost coloring problem.*

Lemma 1.7. *Let $G(V, E)$ be a graph and L be a collection of lists. Then a coloring c exists if the cardinalities of lists fulfil the following:*

- *for vertex coloring*
$$\forall_{u\in V}|L(u)| \geq deg(u) + 1,$$
- *for edge coloring*
$$\forall_{\{u,v\}\in E}|L(\{u, v\})| \geq deg(u) + deg(v) - 1,$$
- *for pseudototal coloring*
$$\forall_{\{u,v\}\in E}|L(\{u, v\})| \geq deg(u) + deg(v) + 1 \quad \text{and} \quad \forall_{u\in V}|L(u)| \geq deg(u) + 1,$$
- *for total coloring*
$$\forall_{\{u,v\}\in E}|L(\{u, v\})| \geq deg(u) + deg(v) + 1 \quad \text{and} \quad \forall_{u\in V}|L(u)| \geq 2deg(u) + 1.$$

Proof. It is easy to see that any greedy algorithm finds a solution provided that elements x are colored one by one and $L(x)$ contains more elements than the number of colors being in a conflict with $c(x)$. ■

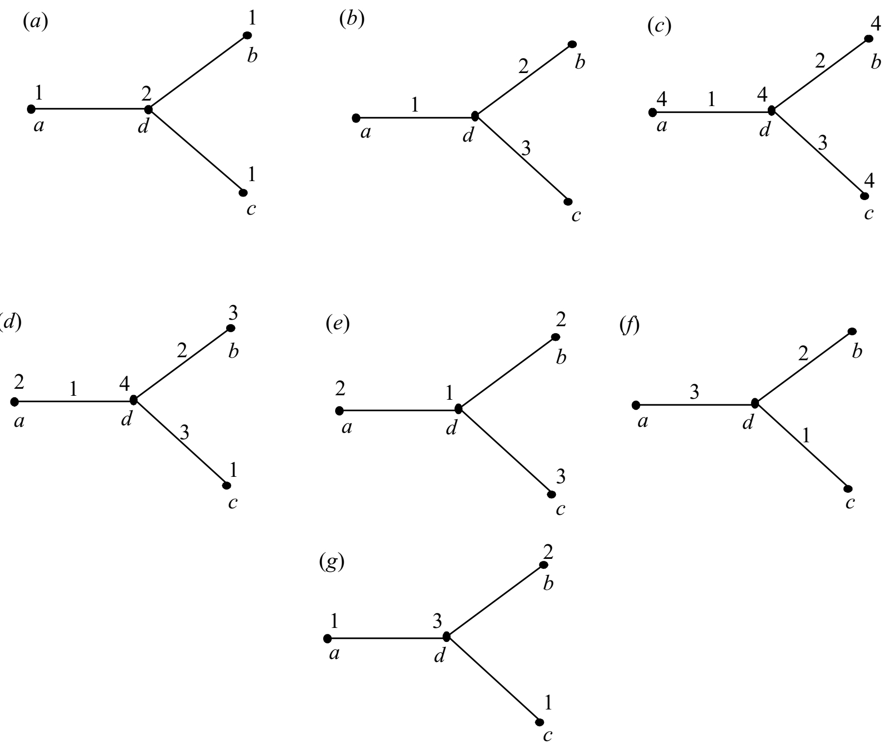

Figure 1. Colorings: (a) vertex; (b) edge; (c) pseudototal; (d) total; (e) list vertex $L(a) = \{2,3\}$, $L(b) = \{2,3\}$, $L(c) = \{1,3\}$, $L(d) = \{1,2\}$; (f) list edge $L(ad) = \{2,3\}$, $L(bd) = \{2,3\}$, $L(cd) = \{1,4\}$; (g) vertex cost optimal $f_a(x) = 5x$, $f_b(x) = 5|x - 2|$, $f_c(x) = f_d(x) = x$.

Definition 1.8. For a given graph G and a given model of coloring, the collection of lists L is called *exact* if for all elements of G being colored the corresponding bound in Lemma 1.7 holds tight.

Note that the general problem of list coloring can be reduced to that of cost coloring. It suffices to assign cost $f_x(i) = 0$ if $i \in L(x)$ and $f_x(i) = 1$ if $i \notin L(x)$. Then a list coloring of G exists if and only if there is a coloring of G (without lists) with zero cost.

Next, the cost coloring problem can be reduced to that of list cost coloring. In fact, we can create an exact collection of lists by assigning to each element x a list $L(x)$ of length mentioned in Definition 1.8, and consisting of colors with the smallest possible costs, i.e., the smallest values of functions f_x. It is easy to see that a coloring c can be reduced to an L-list coloring without increasing its cost. To the aim we successively exchange all colors $c(x) \notin L(x)$ with the cheapest available (at the moment) from $L(x)$. By Definition 1.8 we know that there is at least one such color available.

Also, note that the general version of list cost coloring problem can be polynomially reduced to the same problem but with all lists being exact. In fact, let C be the biggest of all costs on all lists of L. We construct a collection L' of lists by augmenting lists too short (in the sense of Definition 1.8) by new unique colors with cost $(m+n)C+1$. Then the existence of L-list coloring is equivalent to the fact that an optimal L'-list coloring has the cost less than $(m+n)C+1$. On the other hand, deleting the elements with highest costs without changing the optimal solution can shorten the lists too long for being exact. For this reason we may assume, unless otherwise stated, that all instances of the list cost coloring problem have an exact collection of lists.

Kroon *et al.* [6] proved that the optimal cost coloring of the vertices of G can be found in linear time if G is a tree. Using a similar method we obtain

Theorem 1.9 ([6]). *The list cost vertex coloring problem for trees is solvable in linear time.*

2. Coloring Algorithms

We begin with recalling some basic facts concerning matching.

Definition 2.1. Any set of independent edges $B \subseteq E$ of graph $G(V,E)$ is called a *matching*. A matching is *perfect* if $|B| = |V|/2$.

Lemma 2.2 ([9]). *A maximum cardinality matching can be found in time* $O(mn^{1/2})$.

Lemma 2.3 ([5]). *For a given bipartite graph* $G(V,E)$ *with weights* $w : E \mapsto N_0$ *one can find the heaviest matchings (in the sense of sum of weights) in all subgraphs induced by sets* $V - v$, $v \in V$ *in time* $O(mn^{1/2} \log(n \max_{e \in E} w(e)))$.

Note that the same time is required to find the lightest matchings having the maximum cardinality in these subgraphs. It suffices to use the above-mentioned algorithm to the same subgraphs but with weights $w'(e) = \lfloor n(G) \max_{f \in E(G)} w(f)/2 \rfloor - w(e) + 1$.

Lemma 2.4 ([8, 10]). *For every fixed $k \in N$ there exists a polynomial-time probabilistic algorithm for solving the following problem: given graph G, a family of pairwise disjoint subsets $E_i \subseteq E$, and integers $a_i \in N_0$, $i = 1, \ldots, k$; we aim at finding a perfect matching B in G such that $|B \cap E_i| = a_i$ for all $i = 1, \ldots, k$.*

The authors showed in [2] the following

Theorem 2.5. *For every fixed k the cost edge coloring problem with increasing cost functions for graphs G with the cyclomatic number at most k can be solved in time $O(n\Delta^{1.5+k} \log(nC))$, where $C = \max_{e \in E(G)} f_e(2\Delta(G) - 1)$.*

We will show that Theorem 2.5 can be generalized to list cost coloring of such graphs with arbitrary cost functions. Our approach develops a technique for finding optimal edge sum coloring of trees [1] and uses an improvement useful in reducing its time complexity [12].

Lemma 2.6. *An optimal list cost coloring of the edges of tree G can be found in time $O(n\Delta^2 \log(nC))$, where $C = \max_{e \in E(G), i \in L(e)} f_e(i)$.*

Proof. We shall sketch a procedure for determining the minimum cost of list coloring of a tree $G(V, E)$. One can easily extend it to finding the corresponding coloring. Suppose that a collection of lists L is exact for G. For a pair of adjacent vertices v and v' let H stand for the largest subtree of G such that $\{v, v'\}$ belongs to H and v is a leaf. Function $Val(v, v') : L(\{v, v'\}) \mapsto N_0$ is defined so that $Val(v, v')(i)$ is the minimal cost of $L_{|E(H)}$-list coloring of the edges of H with the same cost functions as in G, in which edge $\{v, v'\}$ gets color i. If we recursively find the values of $Val(u, u')$ for a leaf u of G then the requested optimal cost will be equal to $\min_{i \in L(\{u,u'\})} Val(u, u')(i)$.

If H contains only one edge $\{v, v'\}$ then obviously $Val(v, v') = f_{\{v,v'\}}$. So suppose that in the succeeding step of the procedure we have a subtree H as shown in Figure 2 and the functions $Val(v', w_l)$, $l = 1, \ldots, k$ are already known.

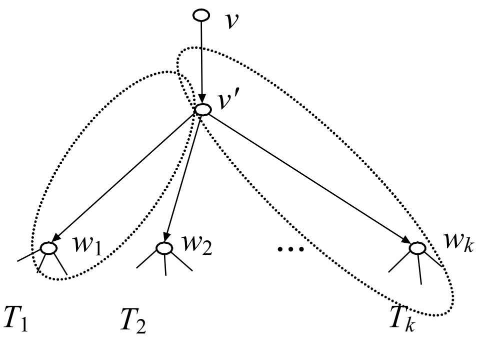

Figure 2. Tree H and its subtrees.

To find the value of $Val(v,v')$ we construct a bipartite graph $K^{v,v'}$ whose vertices in the first partition are $w_1,\dots,w_k$ and the vertices in the second partition are colors $L(\{w_1,v'\})\cup\cdots\cup L(\{w_k,v'\})$. There is an edge $\{w_l,x\}$ in $K^{v,v'}$ if and only if $x\in L(\{w_l,v'\})$ and then we put the weight $Val(v',w_l)(x)$ on it. In that case $Val(v,v')(j)$ is equal to the weight of the lightest k-edge matching in subgraph $K^{v,v'}-\{j\}$ (i.e., the cost of coloring of T_l) plus $f_{\{v,v'\}}(j)$, since edge $\{v,v'\}$ has to be given color j.

Finding the value of $Val(v,v')$ requires prior calculating the weights of k-edge matchings in all graphs $K^{v,v'}-\{j\}$ for $j\in L(\{v,v'\})$. By using the algorithm mentioned in Lemma 2.3 for graph $K^{v,v'}$, for which $n(K^{v,v'})=O(deg_G(v')\Delta(G))=m(K^{v,v'})$, we can do this in time $O(deg_G(v')^{3/2}\Delta(G)^{3/2}$ $\log(n\Delta(G)C))$. We get the overall complexity of the coloring algorithm by summing up among all vertices v', which results in $O(n\Delta^2\log(nC))$. ■

Theorem 2.7. *For every fixed k a list cost coloring of the edges of graph G with the cyclomatic number at most k can be found in time* $O(n\Delta^{2+k}$ $\log(nC))$*, where* $C=\max_{e\in E(G),i\in L(e)}f_e(i)$.

Proof. Let $G(V,E)$ be a graph with $\gamma(G)=k$, L its exact lists and let $A=\{e_1,\dots,e_k\}$ be a set of the edges whose deletion form G results in a spanning tree. Moreover, let $U=\bigcup_{e\in A}e$ be the set of all vertices incident with the edges of A. All we need is an $O(n(G)\Delta(G)^2\log(n(G)C))$-time procedure which for a given $L_{|A}$-list coloring $d:A\mapsto N_0$ of $G(U,A)$ finds its cost optimal extension to an L-list edge coloring of G. We will use the algorithm of Lemma 2.6. On the basis of graph G, the collection of lists L and cost functions we build a tree T, as follows.

Step 1. Delete the edges of A from G.

Step 2. For each deleted edge $\{u, v\} \in A$ introduce two new pendant edges $\{u, u_{new}\}$, $\{v, v_{new}\}$ with 1-element lists of the form: $L^T(\{v, v_{new}\}) = L^T(\{u, u_{new}\}) = \{d(\{u, v\})\}$. Next, define two cost functions for them, namely: one $f^T_{\{v,v_{new}\}} = f_{\{v,u\}}$, and the other $f^T_{\{u,u_{new}\}} = 0$.

Step 3. Leave costs and lists of the remaining edges unchanged.

The optimal cost of list edge coloring of T and the extension of function d to list cost coloring of G are equal: edges $\{u, u_{new}\}$ and $\{v, v_{new}\}$ correspond to the 'halves' of split edge $\{u, v\}$ and 1-element lists of these pendant edges enforce the validity of the extension of d. Finally, note that $n(T) = n(G)+2k$ and $\Delta(T) = \Delta(G)$, which completes the proof. ■

Analogously, for the pseudototal model of coloring (cf. [3]) we have

Theorem 2.8. *For every fixed k a list cost pseudototal coloring of a graph G with the cyclomatic number at most k can be found in time $O(n\Delta^{2+k} \log(nC))$, where $C = \max_{x\in V(G)\cup E(G), i\in L(x)} f_x(i)$.*

Proof. It is easy to see that a pseudototal coloring of $G(V, E)$ corresponds to edge coloring of its supergraph $G'(V', E')$, in which a new additional pendant edge e_v incident with v is introduced on each $v \in V(G)$. Both coloring problems become equivalent, if we carry over the lists and costs from v to e_v for all v, i.e., $\forall_{v\in V(G)} f^{G'}_{e_v} = f_v \wedge L^{G'}(e_v) = L(v)$. Note that $n(G') = 2n(G)$, $\gamma(G) = \gamma(G')$, $\Delta(G') = \Delta(G)+1$, so the complexity follows. ■

Now, let us consider the total model of coloring of G.

Lemma 2.9. *An optimal total list cost coloring of a tree G can be found in time $O(n\Delta^3 \log(nC))$, where where $C = \max_{x\in V(G)\cup E(G), i\in L(x)} f_x(i)$.*

Proof. Like previously, we merely sketch a procedure for determining the minimum cost of coloring a tree G. We will use a method similar to that in the proof of Lemma 2.6. Suppose that a collection of lists L is exact for G. For a pair of adjacent vertices v, v' we denote by H the same subtree as in that proof. Now let $Val(v, v') : L(v) \times L(\{v, v'\}) \mapsto N_0$ be defined so that the value of $Val(v, v')(i, j)$ for $i \neq j$ is the minimal cost of $L_{|V(H)\cup E(H)}$-list total coloring of H (with cost functions as in G), in which vertex v got color i and edge $\{v, v'\}$ obtained color j. If we succeed in calculating the value of function $Val(u, u')$ for a pendant vertex u in G and $\{u, u'\} \in E(G)$, then the requested optimal value of cost will be equal to $\min_{i \in L(u), j \in L(\{u,u'\})-\{i\}} Val(u, u')(i, j)$.

If $E(H)$ consists of $\{v, v'\}$ only and $i \neq j$, then clearly $Val(v, v')(i, j) = f_v(i) + f_{\{v,v'\}}(j) + \min_{a \in L(v')-\{i,j\}} f_{v'}(a)$. So suppose that we have a subtree H as in Figure 2 and that the values of $Val(v', w_l)$, $l = 1, \ldots, k$ are already known. For a triple of different integers $(i, j, s) \subseteq L(v) \times L(\{v, v'\}) \times L(v')$ by $Val_{i,j,s}$ we denote the minimal cost of $L_{|V(H) \cup E(H)}$-list total coloring of H in which vertices v, v' and edge $\{v, v'\}$ got colors i, s, j, respectively. In order to calculate $Val_{i,j,s}$ we construct a bipartite graph $K^{v,v'}(s)$ — the same as in the proof of Lemma 2.6, except that edge $\{w_l, x\}$ has the weight $Val(v', w_l)(s, x)$. It is easy to see that $Val_{i,j,s}$ is equal to the weight of the lightest k-edge matching in graph $K^{v,v'}(s) - \{j, s\}$ (i.e., the total cost of coloring of subtrees T_l) plus $f_v(i) + f_{\{v,v'\}}(j) - (k-1)f_{v'}(s)$ (i.e., the cost of coloring vertex v, edge $\{v, v'\}$ and a correction resulting from a $(k-1)$-fold counting of $f_{v'}(s)$ in the previous step). In the end, we have $Val(v, v')(i, j) = \min_{s \in L(v')-\{i,j\}} Val_{i,j,s}$ for $i \neq j$.

Finding the value of function $Val(v, v')$ requires prior calculation of the numbers $Val_{i,j,s}$, where $s \in L(v')$. The cardinality of the set of these numbers is bounded by $O(\Delta(G)^2 deg(v'))$, but this must be preceded by finding the lightest k-edge matchings in all graphs $K^{v,v'}(s) - \{j, s\}$ for $j \in L(\{v, v'\})$, $s \in L(v') - \{j\}$. By using the algorithm of Lemma 2.3 for graphs $K^{v,v'}(s) - \{s\}$, $s \in L(v')$ we can calculate this in time $O(deg(v')\Delta(G)^3 \log(n\Delta(G)C))$. Finally, by summing up the timing for all v' we conclude that the overall complexity of the tree coloring algorithm is bounded by $O(n\Delta^3 \log(nC))$. ■

A dynamic programming for the total cost coloring of trees was first applied in [3], however, without considering the lists of available colors. Now we shall prove two new theorems that extend Theorems 2.7 and 2.8 to total colorings.

Theorem 2.10. *For every fixed k a total list cost coloring of a graph G with the cyclomatic number at most k can be found in time $O(n\Delta^{3+2k} \log(nC))$, where $C = \max_{x \in V(G) \cup E(G), i \in L(x)} f_x(i)$.*

Proof. We apply the same notation as used in the proof of Theorem 2.7. In addition, let $U' \subseteq U$ with $|U'| \leq k$ be a vertex cover in graph (U, A). All we need is design an algorithm running in time $O(n(G)\Delta(G)^3\log(n(G)C))$, which for a given function $d: U' \cup A \mapsto N$ such that

$$\forall_{x\in U'\cup A} d(x) \in L(x),$$

$$\forall_{u,v\in U'} \{u,v\} \in E(G) \Rightarrow d(u) \neq d(v),$$

$$\forall_{e,f\in A} e \cap f \neq \emptyset \Rightarrow d(e) \neq d(f),$$

$$\forall_{v\in U'} \forall_{e\in A} v \in e \Rightarrow d(e) \neq d(v),$$

finds the minimum cost extension to list coloring of the whole G. As previously, we apply the procedure from the proof of Lemma 2.9. We transform graph G and its lists into tree T (see Steps 1 and 2 in the proof of Theorem 2.7), and define zero-valued cost functions as well as lists for new vertices u_{new} and v_{new}, namely: $L^T(u_{new}) = L(v)$ i $L^T(v_{new}) = L(u)$. Next we introduce the following modifications:

1. For each $v \in U'$ the new list is $L^T(v) = \{d(v)\}$.
2. For each new pendant vertex v_{new} belonging to edge $\{v, v_{new}\}$ introduced in the place of deleted edge $\{u,v\} \in A$, if $u \in U'$ then we set $L^T(v_{new}) = \{d(u)\}$.
3. The remaining elements of tree T inherit their lists and cost functions from G, i.e., $\forall_{x\in(V(G)\cup E(G))-(A\cup U')} f^T_x = f_x \wedge L^T(x) = L(x)$.

Again, it is easy to see that by the same arguments the minimal cost of total list coloring of T and that of list cost coloring of G extending d are equal to each other. ■

The following corollary generalizes Theorem 1.9.

Corollary 2.11. *For every fixed k the problem of list cost coloring of the vertices of G with the cyclomatic number at most k is polynomially solvable.*

Proof. List cost coloring of the vertices of graph G reduces to list total coloring of G whenever all edges are preassigned the same list of $\Delta(G)+1$ colors different from the colors available to vertices and all edges are preassigned zero-valued cost functions. ■

3. Multicoloring of Graphs with Few Cycles

Let us assume that graph G is assigned a new parameter being an integer function *size* S defined on elements to be colored (vertices, edges or both vertices and edges). This allows us to generalize all models of coloring under consideration to multicoloring. A *multicoloring* is an assignment of a set $c(x)$ of colors to every vertex and/or edge x, i.e., a set $c(x) \in P_{fin}(N)$, where $|c(x)| = S(x)$. Now the non-conflict condition $c(x) \neq c(y)$ becomes $c(x) \cap c(y) = \emptyset$ for any two adjacent or incident elements x, y. We are not going to give a formal definition of vertex multicoloring, edge multicoloring, etc., since these notions are obvious. It is easy to see that classical coloring can be regarded as a special case of multicoloring in which $S = 1$. In addition, for a list multicoloring we have $c(x) \subseteq L(x)$. We can also extend the cost criterion to the formula $\sum_{v \in V} \sum_{i \in c(v)} f_v(i)$ for the vertex multicoloring model and $\sum_{e \in E} \sum_{i \in c(e)} f_e(i)$ and $\sum_{x \in V \cup E} \sum_{i \in c(x)} f_x(i)$ for the edge and (pseudo)total multicoloring model, respectively. The extension of coloring to multicoloring causes a dramatic increase in computational complexity.

Theorem 3.1 ([7]). *It is NP-complete to decide the existence of list vertex multicoloring for a binary tree G with a size function $S : V(G) \mapsto N$.*

If we assign to the edges any 1-element pairwise different lists of available colors not appearing on the vertex lists L and we set $S(e) = 1$ to each e of a binary tree, then we transform our problem to the total list multicoloring problem.

Corollary 3.2. *It is NP-complete to decide the existence of list total multicoloring of a binary tree.*

Similarly as in Section 1 we can get rid of lists by introducing suitable zero-one cost functions. In this way we reduce the L-list multicoloring problem to the cost multicoloring (without lists).

Corollary 3.3. *The problems of vertex cost and total cost multicoloring of a binary tree are NP-hard.*

Thus the only chance for trees to have a polynomial-time algorithm for multicoloring are the models of edge and pseudototal coloring, i.e., the models not introducing the conflicts on colors of adjacent vertices. In fact, we have

Theorem 3.4 ([8]). *The list edge multicoloring problem can be solved in polynomial time for a tree G with size function $S : E(G) \mapsto N$.*

Following [8] we shall sketch the proof. For a graph $G(V, E)$ with function $S : E \mapsto N_0$ let us define another mapping $S' : V \mapsto N_0$ as follows

$$S'(v) = \sum_{e \in E_v} S(e). \tag{1}$$

This allows us to reduce L-list edge multicoloring that fulfills:

$$\begin{gathered} \forall_{e \in E} c(e) \subseteq L(e), \\ \forall_{e,f \in E} e \cap f \neq \emptyset \Rightarrow c(e) \cap c(f) = \emptyset, \\ \forall_{e \in E} |c(e)| = S(e) \end{gathered} \tag{2}$$

to deciding whether there is a *modified L-list multicoloring* c of G with appropriate numbers of colors meeting at the vertices rather than just put on the edges,

$$\begin{gathered} \forall_{e \in E} c(e) \subseteq L(e), \\ \forall_{e,f \in E} e \cap f \neq \emptyset \Rightarrow c(e) \cap c(f) = \emptyset, \\ \forall_{v \in V} \sum_{e \in E_v} |c(e)| = S'(v). \end{gathered} \tag{3}$$

As it was shown in [8], if G is a tree then function S' unambiguously determines S, then in case of trees and functions S and S' defined in (1) conditions (2) and (3) are equivalent for any multicoloring.

Now conditions (3) can be reduced to finding a perfect matching, see Figure 3 for example. Namely, we construct graph M as a sum of disjoint copies $G_l(V_l, E_l)$ of G for all $l \in \bigcup_{e \in E(G)} L(e)$. From here on by x_i we denote a copy of element x in subgraph G_i. Delete from M all edges e_i such that $i \notin L(e)$ and after that all isolated vertices. We want a perfect matchings in M to correspond to modified multicoloring c fulfilling conditions (3) with $i \in c(e)$ if and only if e_i belongs to the perfect matching. Henceforth, we have to guarantee that for each $v \in V$ in $\bigcup_{l \in (\bigcup_{e \in E(G)} L(e))} E_l$ there are exactly $S'(v)$ edges of the matching outgoing from vertices v_i ($i \in \bigcup_{e \in E_v(G)} L(e)$). To the purpose we add to graph M a set of new vertices V_v with cardinality $|\bigcup_{e \in E_v(G)} L(e)| - S'(v)$ which, together with v_i ($i \in \bigcup_{e \in E_v(G)} L(e)$) create two partitions of a complete bipartite graph, and we iterate this operation for all $v \in V(G)$. It is easy to see that each perfect matching in such M corresponds to a certain L-list multicoloring of tree G. But $n(M) = O(n(G)\Delta_L)$ and $m(M) = O(n(G)\Delta_L^2)$, where $\Delta_L = \max_{v \in V(G)} \sum_{e \in E_v(G)} |L(e)|$. Therefore, by Lemma 2.2, we can solve the problem of existence of L-list edge multicoloring for tree G in time $O(n^{3/2}\Delta_L^{5/2})$. The description of problem's instance must contain the structure of graph G, function S as well as all lists $L(e)$ for $e \in E(G)$, so the above time complexity is polynomial.

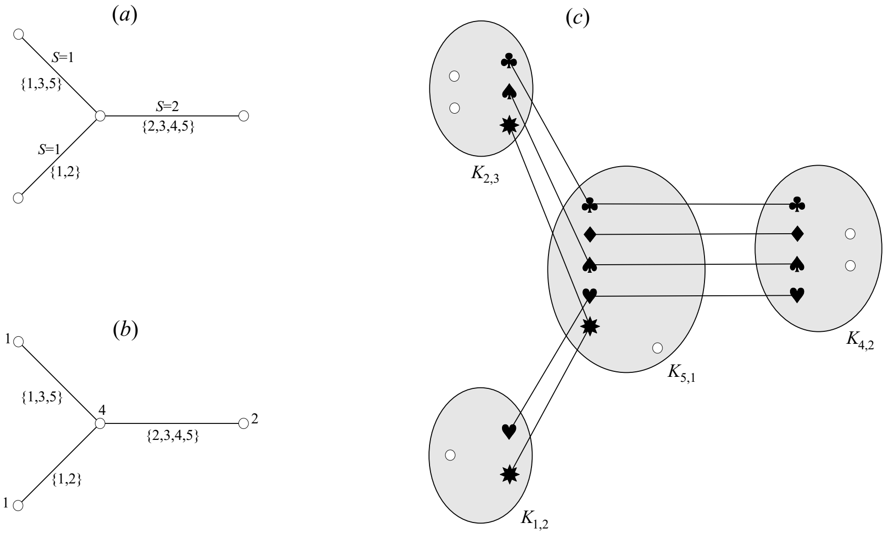

Figure 3. Reduction: (a) instance of the problem; (b) tree G with function S'; (c) graph M, where grey ellipses stand for complete bipartite graphs with white-black partitions and vertices of different subgraphs $G_1, \ldots, G_5$ are marked with five different black symbols.

Theorem 3.5. *The list cost multicoloring problem for the edges of tree G with $S : E(G) \mapsto N$ can be solved in polynomial time.*

Proof. On the edges of graph M defined in the previous proof we put weights w equal to the costs of coloring of the corresponding edges from G:

$$\forall_{e \in E(G)} \forall_{i \in L(e)} w(e_i) = f_e(i)$$

and the remaining edges have weights all equal to 0. Next, we look for a lightest perfect matching in M. Such a matching determines a minimum solution to our multicoloring problem. Using the algorithm mentioned in Lemma 2.3 we obtain the result in time $O(n^{3/2}\Delta_L^{5/2}\log(n\Delta_L C))$, where $C = \max_{e\in E(G),i\in L(e)} f_e(i)$. ■

Theorem 3.6 ([8])**.** *For every fixed k there is a randomized polynomial-time algorithm for list edge multicoloring in graphs with the cyclomatic number at most k.*

Proof. Following [8] we sketch the proof. Given graph $G(V,E)$, let $A \subseteq E$ ($|A| = k$) be a set of edges whose deletion from G results in a tree. Now function S' defined in (1) does not have to determine S unambiguously and, consequently, conditions (3) do not imply those of (2). In order for a perfect matching B in M to determine L-list multicoloring of the edges of G with size functions S, we have to add to (3) the following condition: $\forall_{e\in A}|B \cap \{e_i : i \in L(e)\}| = S(e)$ to fulfil condition (2) (see [8] for details). In this place the author of [8] refers to Lemma 2.4. ■

Now, by combining both techniques, we obtain

Theorem 3.7. *For every fixed k there is a randomized pseudopolynomial-time algorithm for list cost edge multicoloring in graphs with the cyclomatic number at most k.*

Proof. Instead of M we consider a homeomorphic graph M' in which each edge $e_i \in E(M)$, where $e \in E(G)$, $i \in L(e)$, has been replaced by a path of length $2f_e(i)+1$ consisting of consecutive edges denoted $e_{i,1}, e_{i,2}, \ldots, e_{i,2f_e(i)+1}$. Note that there is a one-to-one correspondence between a perfect matching B in M and a perfect matching B' in M'. Moreover, if $e_i \in B$ then $e_{i,1}, e_{i,3}, e_{i,5}, \ldots, e_{i,2f_e(i)-1}, e_{i,2f_e(i)+1} \in B'$, otherwise $e_{i,2}, e_{i,4}, \ldots, e_{i,2f_e(i)-2}, e_{i,2f_e(i)} \in B'$. Now let $F = \{e_{i,j} \in E(M') : e \in E(G) \wedge i \in L(e) \wedge j > 1 \wedge \neg(2|j)\}$ and $A \subseteq E(G)$ be the same as in the proof of Theorem 3.6. Note that F has exactly $f_e(i)$ edges in common with the path of M' which replaced edge $e_i \in E(M)$. Thus G has L-list edge multicoloring with cost equal to X if and only if M' has a perfect matching B' fulfilling

$$\forall_{e\in A}|B' \cap \{e_{i,1} : i \in L(e)\}| = S(e),$$

$$|B' \cap F| = X.$$

The latter problem can be solved by means of the algorithm of Lemma 2.4. The minimum cost solution to our problem can be found by a search in the interval from 0 to $\sum_{e\in E(G)} S(e) \max f_{e|L(e)}$. ■

Theorem 3.8. *All the theses of Theorems* 3.4–3.7 *remain true in pseudo-total multicoloring model.*

Proof. Similarly as in the proof of Theorem 2.8 we can translate pseudo-total multicoloring into the edge multicoloring model by replacing $G(V, E)$ with its supergraph $G'(V', E')$. In this case we also have to extend the size function by giving the size $S^{G'}(e_v) = S(v)$ to each new pendant edge e_v. ■

References

[1] K. Giaro and M. Kubale, *Edge-chromatic sum of trees and bounded cyclicity graphs*, Inf. Process. Lett. **75** (2000) 65–69.

[2] K. Giaro, M. Kubale and P. Obszarski, *A graph coloring approach to scheduling multiprocessor tasks on dedicated machines with availability constraints*, Disc. Appl. Math., (to appear).

[3] S. Isobe, X. Zhou and T. Nishizeki, *Cost total colorings of trees*, IEICE Trans. Inf. and Syst. **E-87** (2004) 337–342.

[4] J. Jansen, *Approximation results for optimum cost chromatic partition problem*, J. Alghoritms **34** (2000) 54–89.

[5] M. Kao, T. Lam, W. Sung and H. Ting, *All-cavity maximum matchings*, Proc. ISAAC'97, LNCS **1350** (1997) 364–373.

[6] L. Kroon, A. Sen. H. Deng and A. Roy, *The optimal cost chromatic partition problem for trees and interval graphs*, Proc. WGTCCS'96, LNCS **1197** (1997) 279–292.

[7] D. Marx, *The complexity of tree multicolorings*, Proc. MFCS'02, LNCS **2420** (2002) 532–542.

[8] D. Marx, *List edge muticoloring in graphs with few cycles*, Inf. Proc. Lett. **89** (2004) 85–90.

[9] S. Micali and V. Vazirani, *An $O(mn^{1/2})$ algorithm for finding maximum matching in general graphs*, Proc. 21st Ann. IEEE Symp. on Foundations of Computer Science (1980) 17–27.

[10] K. Mulmuley, U. Vazirani and V. Vazirani, *Matching is as easy as matrix inversion*, Combinatorica **7** (1987) 105–113.

[11] T. Szkaliczki, *Routing with minimum wire length in the dogleg-free Manhattan model is NP-complete*, SIAM J. Computing **29** (1999) 274–287.

[12] X. Zhou and T. Nishizeki, *Algorithms for the cost edge-coloring of trees*, LNCS **2108** (2001) 288–297.

Received 30 November 2007
Revised 26 February 2009
Accepted 26 February 2009