World Journal of Engineering and Technology, 2015, 3, 119-124
Published Online October 2015 in SciRes. http://www.scirp.org/journal/wjet
http://dx.doi.org/10.4236/wjet.2015.33C018

Cost Edge-Coloring of a Cactus

Zhiqian Ye[1], Yiming Li[2], Huiqiang Lu[3], Xiao Zhou[4]

[1]Zhejiang University, Hanzhou, China
[2]Wenzhou University, Wenzhou, China
[3]Zhejiang University of Technology, Hangzhou, China
[4]Tohoku University, Sendai, Japan
Email: yezhiqian@zju.edu.cn, ymli@wzu.edu.cn, lhq@zjut.edu.cn, zhou@ecei.tohoku.ac.jp

Received 11 August 2015; accepted 15 October 2015; published 22 October 2015

Abstract

Let C be a set of colors, and let $\omega(c)$ be an integer cost assigned to a color c in C. An edge-coloring of a graph $G=(V,E)$ is assigning a color in C to each edge $e \in E$ so that any two edges having end-vertex in common have different colors. The cost $\omega(f)$ of an edge-coloring f of G is the sum of costs $\omega(f(e))$ of colors $f(e)$ assigned to all edges e in G. An edge-coloring f of G is optimal if $\omega(f)$ is minimum among all edge-colorings of G. A cactus is a connected graph in which every block is either an edge or a cycle. In this paper, we give an algorithm to find an optimal edge-coloring of a cactus in polynomial time. In our best knowledge, this is the first polynomial-time algorithm to find an optimal edge-coloring of a cactus.

Keywords

Cactus, Cost Edge-Coloring, Minimum Cost Maximum Flow Problem

1. Introduction

Let $G=(V,E)$ be a graph with vertex set V and edge set E, and let C be a set of colors. An *edge-coloring* of G is to color all the edges in E so that any two adjacent edges are colored with different colors in C. The minimum number of colors required for edge-colorings of G is called the *chromatic index*, and is denoted by $\chi'(G)$. It is well-known that $\Delta(G) \le \chi'(G) \le \Delta(G)+1$ for every simple graph G and that $\chi'(G)=\Delta(G)$ for every bipartite (multi)graph G, where $\Delta(G)$ is the maximum degree of G [1]. The ordinary *edge-coloring problem* is to compute the chromatic index $\chi'(G)$ of a given graph G and find an edge-coloring of G using $\chi'(G)$ colors. Let ω be a cost function which assigns an integer $\omega(c)$ to each color $c \in C$, then the *cost edge-coloring problem* is to find an *optimal edge-coloring* of G, that is, an edge-coloring f such that $\sum_{e\in E}\omega(f(e))$ is minimum among all edge-colorings of G. An optimal edge-coloring does not always use the minimum number $\chi'(G)$ of colors. For example, suppose that $\omega(c_1)=1$ and $\omega(c_i)=2$ for each index $i \ge 2$, then the graph G with $\chi'(G)=3$ in **Figure 1(a)** can be uniquely colored with the three cheapest colors c_1, c_2 and c_3 as in **Figure 1(a)**, but this edge-coloring is not optimal; an optimal edge-coloring of G uses the four cheapest colors c_1, c_2, c_3 and c_4, as illustrated in **Figure 1(b)**. However, every simple graph G has an edge-coloring

How to cite this paper: Ye, Z.Q., Li, Y.M., Lu, H.Q. and Zhou, X. (2015) Cost Edge-Coloring of a Cactus. *World Journal of Engineering and Technology*, **3**, 119-124. http://dx.doi.org/10.4236/wjet.2015.33C018

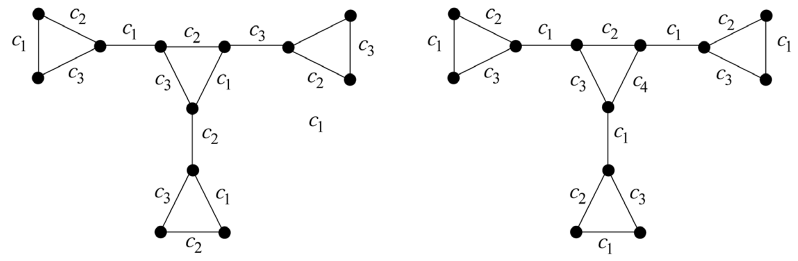

Figure 1. (a) An edge-coloring using $\chi'(G)=3$ colors, and (b) an optimal edge-coloring using $\chi'(G)+1=4$ colors, where $\omega(c_1)=1$ and $\omega(c_2)=\omega(c_3)=\omega(c_4)=2$.

using $\Delta(G)$ or $\Delta(G)+1$ colors [2] [3]. The edge-chromatic sum problem, introduced by Giaro and Kubale [4], is merely the cost edge-coloring problem for the special case where $\omega(c_i)=i$ for each integer $i\geq 1$.

The cost edge-coloring problem has a natural application for scheduling [5]. Consider the scheduling of biprocessor tasks of unit execution time on dedicated machines. An example of such tasks is the file transfer problem in a computer network in which each file engages two corresponding nodes, sender and receiver, simultaneously [6]. Another example is the biprocessor diagnostic problem in which links execute concurrently the same test for a fault tolerant multiprocessor system [7]. These problems can be modeled by a graph *G* in which machines correspond to the vertices and tasks correspond to the edges. An edge-coloring of *G* corresponds to a schedule, where the edges colored with color $c_i \in C$ represent the collection of tasks that are executed in the *i*th time slot. Suppose that a task executed in the *i*th time slot takes the cost $\omega(c_i)$. Then the goal is to find a schedule that minimizes the total cost, and hence this corresponds to the cost edge-coloring problem.

The cost edge-coloring problem is APX-hard even for bipartite graphs [8], and hence there is no polynomial-time approximation scheme (PTAS) for the problem unless $P=NP$. On the other hand, Zhou and Nishizeki gave an algorithm to solve the cost edge-coloring problem for trees *T* in time $O(n\Delta^{1.5}\log(nN_\omega))$, where *n* is the number of vertices in *T*, Δ is the maximum degree of *T*, and N_ω is the maximum absolute cost $|\omega(c)|$ of colors *c* in *C* [5]. The algorithm is based on a dynamic programming (DP) approach, and computes a DP table for each vertex of a given tree *T* from the leaves to the root of *T*. In this paper, we give a polynomial-time algorithm to solve the cost edge-coloring problem for cacti. In our best knowledge, this is the first polynomial-time algorithm to find an optimal edge-coloring of a cactus.

2. Preliminaries

In this section, we define some basic terms.

Let $G=(V,E)$ be a graph with a set *V* of vertices and a set *E* of edges. We sometimes denote by $V(G)$ and $E(G)$ the vertex set and the edge set of *G*, respectively. We denote by $n(G)$ and $m(G)$, respectively, or simply by *n* and *m*, the number of vertices and edges in *G*, that is, $n(G)=|V|$ and $m(G)=|E|$. The *degree* $d(v)$ of a vertex *v* is the number of edges in *E* incident to *v*. We denote the maximum degree of *G* by $\Delta(G)$ or simply by Δ. A cactus *G* can be represented by an under tree *T*, which is a rooted tree. In the underlay tree *T* of *G*, each node represents a block which is either a bridge (edge) of *G* or an elementary cycle of *G*. If there is an edge between nodes b_1 and b_2 of *T*, then bridges or cycles of *G* represented by b_1 and b_2 share exactly one vertex in *G*. Each node *b* of *T* corresponds to a subgraph G_b of *G* induced by all bridges and cycles represented by the nodes that are descendants of *b* in *T*. **Figure 2(a)** depicts the subgraph G_{b_1} for the child b_1 of the root *r* of *T*. Clearly $G=G_r$ and G_b is a cactus for each node *b* of *T*. One can easily find an underlay tree *T* of a given cactus *G* in linear time, and hence one may assume that an underlay tree of *G* is given. We denote by $\mathrm{ch}(b)$ the number of edges joining a node *b* and its children in *T*. Then, $\mathrm{ch}(r)=d(r)$, and $\mathrm{ch}(b)=d(b)-1$ for every vertex $b\in V\setminus\{r\}$.

Let *C* be a set of colors. An *edge-coloring* $f:E\to C$ of a graph *G* is to color all edges of *G* by colors in *C* so that any two adjacent edges are colored with different colors. Let $\omega:C\to\mathbb{R}^+$, where $\mathbb{R}^+$ is the set of real numbers. One may assume with loss of generality that ω is non-decreasing, that is, $\omega(c_i)\leq\omega(c_{i+1})$ for any

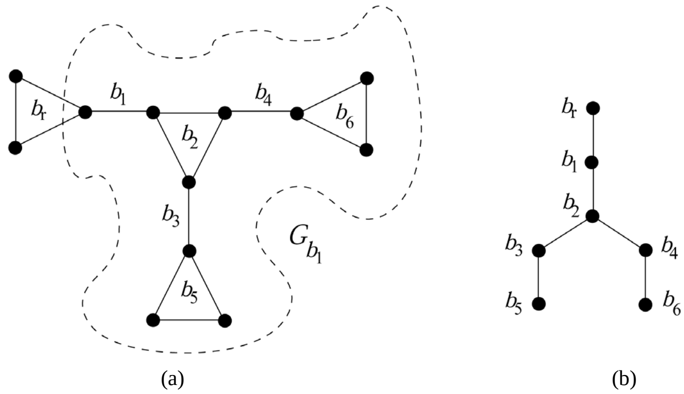

Figure 2. (a) A cactus; and (b) its under tree.

index i, $1 \le i \le |C|$. Since trivially any graph G has an optimal edge-coloring using colors at most $2\Delta(G)-1$, we assume for the sake of convenience that $|C| = 2\Delta(G)-1$, and we write $C = \{c_1, c_2, \cdots, c_{2\Delta-1}\}$. The *cost* $\omega(f)$ *of an edge-coloring f* of a graph $G = (V, E)$ is defined as follows:

$$\omega(f) = \sum_{e \in E} \omega(f(e)).$$

An edge-coloring *f* of *G* is called an *optimal* one if $\omega(f)$ is minimum among all edge-colorings of *G*. The *cost edge-coloring problem* is to find an optimal edge-coloring of a given graph *G*. The cost of an optimal edge-coloring of *G* is called the *minimum cost of G*, and is denoted by $\omega(G)$.

Let *f* be an edge-coloring of a graph *G*. For each vertex *v* of *G*, let $C_f(G, v)$ be the set of all colors that are assigned to edges incident to *v*, that is,

$$C_f(G, v) = \{f(e) \mid e \text{ is an edge incident to } v \text{ in } G\}.$$

We say that a color $c \in C$ is *missing at v* if $c \notin C(f, v)$. Let $\text{Miss}(f, v)$ be the set of all colors missing at *v*, that is, $\text{Miss}(f, v) = C \setminus C(f, v)$.

3. Algorithm

In this section we prove the following theorem.

Theorem 1. An optimal edge-coloring of a cactus can be found in polynomial time.

As a proof of Theorem 1, we give a dynamic programming algorithm in the remainder of this section to compute the minimum cost $\omega(G)$ of a given cactus *G*. Our algorithm can be easily modified so that it actually finds an optimal edge-coloring *f* of *G* with $\omega(f) = \omega(G)$.

A dynamic programming method is a standard one to solve a combinatorial problem on graphs with tree-construction. We also use it, and compute the minimum cost $\omega(G)$ of a cactus *G* with an under tree *T* by the bottom-up tree computation.

3.1. Ideas and Definitions

Let *b* be a node of *T* with its parent b', and let *v* be the vertex on both two blocks *b* and b'. Let $b_1, b_2, \cdots, b_{\text{ch}(b)}$ be the children of *b* in *T*. Then one can observe that the minimum cost $\omega(G_b)$ of the subgraph G_b rooted at *b* cannot be computed directly from the minimum costs $\omega(G_{b_j})$ of all the subgraphs G_{b_j}, $1 \le j \le \text{ch}(b)$. Our idea is to introduce a new parameter $\omega(G_b, i_1, i_2)$ defined for each node *b* of *T* and each pair of colors $c_{i_1}, c_{i_2} \in C$ as follows:

$$\omega(G_b, i_1, i_2) = \min\{\omega(f) \mid f \text{ is an edge-coloring of } G_b \text{ and } c_{i_1}, c_{i_2} \in C(f, v)\}.$$

If G_b has no such edge-coloring we define $\omega(G_b, i_1, i_2) = +\infty$. Note that $\omega(G_b, i_1, i_2) = +\infty$ if either the block *b* is an edge and $i_1 \ne i_2$ or the block *b* is a cycle and $i_1 = i_2$. Clearly,

$$\omega(G_b) = \min_{1 \le i_1, i_2 \le 2\Delta-1} \omega(G_b, i_1, i_2).$$

We compute the values $\omega(G_b, i_1, i_2)$ for all indices i_1, i_2, $1 \le i_1, i_2 \le 2\Delta - 1$, from leaves to root *r*. Thus the DP table for each node *b* consists of the $O(\Delta^2)$ values $\omega(G_b, i_1, i_2)$, $1 \le i_1, i_2 \le 2\Delta - 1$.

Our algorithm computes $\omega(G_b, i_1, i_2)$ for all pairs of colors $c_{i_1}, c_{i_2} \in C$ from the leaves to the root *r* of *T*, by means of dynamic programming. Then $\omega(G)$ can be computed at the root *r* from all the values $\omega(G_r, i_1, i_2)$ as follows:

$$\omega(G) = \begin{cases} \min\{\, \omega(G_r, i, i) \mid c_i \in C \} & \text{if the block } r \text{ is an edge;} \\ \min\{\, \omega(G_r, i_1, i_2) \mid c_{i_1}, c_{i_2} \in C \text{ and } i_1 \ne i_2 \} & \text{if the block } r \text{ is a cycle} \end{cases}$$

and it can be computed in polynomial time. Thus the remainder problem is how to compute all the values $\omega(G_b, i_1, i_2)$ for each node $b \in V(T)$ of *T* and all pairs of colors $c_{i_1}, c_{i_2} \in C$.

3.2. Algorithm

In this subsection, we explain how to compute all the values $\omega(G_b, i_1, i_2)$ for each node $b \in V(T)$ of *T* and all pairs of colors $c_{i_1}, c_{i_2} \in C$.

3.2.1. The Node *b* Is a Leaf in *T*

In this case, the block *b* is either an edge or a cycle. Therefore we have the following two cases to consider.

Case 1: the block *b* is an edge.

In this case, clearly

$$\omega(G_b, i_1, i_2) = \begin{cases} \omega(c_{i_1}) & \text{if } i_1 = i_2; \\ +\infty & \text{if } i_1 \ne i_2, \end{cases}$$

and all the values $\omega(G_b, i_1, i_2)$, $c_{i_1}, c_{i_2} \in C$, can be computed in time polynomial in $|C|$.

Case 2: the block *b* is a cycle.

In this case, we describe the following algorithm to compute $\omega(G_b, i_1, i_2)$ in time polynomial in the size of G_b and $|C|$.

Algorithm 1 AlgLeaf(G_b, i_1, i_2);

let $C = \{c_1, c_2, \cdots, C_{2\Delta-1}\}$;
let $v_1, v_2, \cdots, v_x$ be the vertices lied on the cycle of G_b in the clockwise order;
assume that v_1 is also on other blocks, that is, $d(G, v_1) \ge 2$ and $d(G, v_j) = 2$ for all j, $2 \le j \le x$;
if $i_1 = i_2$ **then**
  return $\omega(G_b, i_1, i_2) = +\infty$;
else
  if i_1 or $i_2 = 1$ **then**
    assume without loss of generality that $i_1 = 1$;
    if $i_2 \ne 2$ **then**
      return $\omega(G_b, i_1, i_2) = \omega(c_{i_2}) + \omega(c_1) * \lceil (x-1)/2 \rceil + \omega(c_2) * \lfloor (x-1)/2 \rfloor$;
    else
      if x is even **then**
        return $\omega(G_b, i_1, i_2) = \omega(c_1) * x/2 + \omega(c_2) * x/2$;
      else
        return $\omega(G_b, i_1, i_2) = \omega(c_1) * (x-1)/2 + \omega(c_2) * (x-1)/2 + \omega(c_3)$;
      end if
    end if
  else
    if i_1 or $i_2 = 2$ **then**
      assume without loss of generality that $i_1 = 2$ and $i_2 \ge 3\}$;
      return $\omega(G_b, i_1, i_2) = \omega(c_{i_2}) + \omega(c_1) * \lfloor (x-1)/2 \rfloor + \omega(c_2) * \lceil (x-1)/2 \rceil$;
    else
      return $\omega(G_b, i_1, i_2) = \omega(c_{i_1}) + \omega(c_{i_2}) + \omega(c_1) * \lceil (x-2)/2 \rceil + \omega(c_2) * \lfloor (x-2)/2 \rfloor$;
    end if
  end if
end if

3.2.2. The Node *b* Is an Internal Node

In order to compute $\omega(G_b, i_1, i_2)$ for each pair of indices i_1 and i_2, $1 \le i_1, i_2 \le |C|$, we introduce a new parameter $\omega^*(B, v, i_1, i_2)$ defined as follows.

Let $B = \{b_1, b_2, \cdots\}$ be a set of blocks of *T* such that all these blocks share exactly one vertex *v* in *G*. For each pair of colors $c_{i_1}, c_{i_2} \in C$ we define

$$\omega^*(B, v, i_1, i_2) = \min\{\omega(f) \mid f \text{ is an edge-coloring of } G_v \text{ and } c_{i_1}, c_{i_2} \in \text{Miss}(f, v)\}.$$

We show how to compute the all the values $\omega^*(B, v, i_1, i_2)$ from the $|B| \times |C|^2$ values $\omega(G_{b_j}, i_1, i_2)$, $1 \le j \le |B|$ and $1 \le i_1, i_2 \le |C|$. The problem of computing $\omega^*(B, v, i_1, i_2)$ can be reduced to the minimum cost flow problem on a bipartite graph $K(i_1, i_2)$ as follows.

We first introduce $|B| \times |C|^2$ isolated vertices $v^j_{l_1,l_2}$, $1 \le j \le |B|$ and $1 \le l_1, l_2 \le |C|$. Then add $|C|$ vertices v_l, $1 \le l \le |C|$, corresponding to colors c_l, and add a source *s* and a sink *t*. Connect the source *s* to all the $|C|$ vertices v_l, $1 \le l \le |C|$, with capacity 1 and cost 0. For each vertex v_l, $1 \le l \le |C|$ and $l \notin \{i_1, i_2\}$, connect v_l to all the vertices $v^j_{l_1,l_2}$, $1 \le j \le |B$ and $1 \le l_1, l_2 \le |C|$, satisfying $l_1 = l$ or $l_2 = l$ with capacity 1 and cost 0. Finally, for each vertex $v^j_{l_1,l_2}$, $1 \le j \le |B|$ and $1 \le l_1, l_2 \le |C|$, connect $v^j_{l_1,l_2}$ to the sink *t* with capacity 2 and cost $\omega(G_{b_j}, l_1, l_2)$. The minimum cost flow problem is to find a maximum flow from *s* to *t* with the sum of costs of edges on the flow. Clearly $\omega^*(B, v, i_1, i_2)$ is equal to the cost of the minimum cost maximum flow in $K(i_1, i_2)$.

The minimum cost maximum flow problem can be solved in time polynomial in the size of the graph [9] [10], and hence the value $\omega^*(B, v, i_1, i_2)$ for a pair of indices i_1 and i_2, $1 \le i_1, i_2 \le |C|$, can be computed in time polynomial in $|B|$ and $|C$ since $K(i_1, i_2)$ has at most $O(|B||C|^2)$ vertices and edges. Therefore the $|C|^2$ values $\omega^*(B, v, i_1, i_2)$ for all pairs of indices i_1 and i_2, $1 \le i_1, i_2 \le |C|$, can be computed total in time polynomial in $|B|$ and $|C$.

We are now ready to compute $\omega(G_b, i_1, i_2)$. Since the block *b* is either an edge or a cycle, we have the following two cases to consider.

Case 1: the block *b* is an edge $e = (u, v)$.

Let $B = \{b_1, b_2, \cdots, b_{\text{ch}(b)}\}$ be the set of blocks of the children of *b* in *T*. Then all the blocks $b_1, b_2, \cdots, b_{\text{ch}(b)}$ share exactly one vertex *v* in *G*. In this case, clearly

$$\omega(G_b, i_1, i_2) = \begin{cases} \omega^*(B, v, i_1, i_1) & \text{if } i_1 = i_2, \\ +\infty & \text{if } i_1 \ne i_2; \end{cases}$$

and it can be computed in time polynomial in the size of G_b and $|C|$.

Case 2: the block *b* is a cycle.

In this case, let $v_1, v_2, \cdots, v_x$ be the vertices lied on the cycle of G_b in the clockwise order. Assume that v_1 is the vertex shared by the block *b* and its parent block, and let $B(v_j)$, $2 \le j \le x$, be the set of blocks which shares v_j; $B(v_j) = \varnothing$ if no such blocks exist. In order to compute $\omega(G_b, i_1, i_2)$ we define

$$\omega^*_{1,j}(i_1, l_j) = \min_{1 \le l_2, l_3, \cdots, l_{j-1} \le |C|} \left\{ \sum_{2 \le p \le j} \omega^*(B(v_p), v_p, l_{p-1}, l_p) + \sum_{1 \le p \le j} \omega(c_{l_p}) \right\} \quad (1)$$

for each *j*, $2 \le j \le x$, where $l_1 = i_1$. Then clearly

$$\omega(G_b, i_1, i_2) = \omega^*_{1,x}(i_1, i_2).$$

Therefore it suffices to show how to compute $\omega^*_{1,j}(i_1, l_j)$ in polynomial time for each *j*, $2 \le j \le x$, as follows.

By Equation (1) we have

$$\begin{aligned} \omega^*_{1,j+1}(i_1, l_{j+1}) &= \min_{1 \le l_2, l_3, \cdots, l_j \le |C|} \left\{ \sum_{2 \le p \le j+1} \omega^*(B(v_p), v_p, l_{p-1}, l_p) + \sum_{1 \le p \le j} \omega(c_{l_{p+1}}) \right\} \\ &= \min_{1 \le l_j \le |C|} \left\{ \omega^*_{1,j}(i_1, l_j) + \omega^*(B(v_{j+1}), v_{j+1}, l_j, l_{j+1}) + \omega(c_{l_{j+1}}) \right\}, \end{aligned}$$

and hence $\omega^*_{1,j}(i_1,l_j)$ for all *j*, $2 \le j \le x$, can be recursively computed total in time $O(x|C|)$ if all the values $\omega^*(B(v_j),v_j,l_1,l_2)$, $1 \le l_1,l_2 \le |C|$, are given. Since we have mentioned before that all the values $\omega^*(B(v_j),v_j,l_1,l_2)$ can be computed in time polynomial in $|B(v_j)|$ and $|C|$, one can compute all $\omega^*_{1,j}(i_1,l_j)$ and hence $\omega(G_b,i_1,i_2)$ total in time polynomial in $n(G_b)$ and $|C|$.

4. Conclusion

In this paper, we show that the cost edge-coloring problem for a cactus *G* can be solved in polynomial time. It is still open to solve the problem in polynomial time for outerplanar graphs.

Supported

This work is partially supported by grants of the thousand talent plan of Zhejiang province.

References

[1] West, D.B. (2000) Introduction to Graph Theory. 2nd Edition, Prentice Hall, New Jersey.

[2] Hajiabolhassan, H., Mehrabadi, M.L. and Tusserkani, R. (2000) Minimal Coloring and Strength of Graphs. *Discrete Mathematics*, **215**, 265-270. http://dx.doi.org/10.1016/S0012-365X(99)00319-2

[3] Mitchem, J., Morriss, P. and Schmeichel, E. (1997) On the Cost Chromatic Number of Outerplanar, Planar, and Line Graphs. *Discussiones Mathematicae Graph Theory*, **17**, 229-241. http://dx.doi.org/10.7151/dmgt.1050

[4] Giaro, K. and Kubale, M. (2000) Edge-Chromatic Sum of Trees and Bounded Cyclicity Graphs. *Information Processing Letters*, **75**, 65-69. http://dx.doi.org/10.1016/S0020-0190(00)00072-7

[5] Zhou, X. and Nishizeki, T. (2004) Algorithm for the Cost Edge-Coloring of Trees. *J. Combinatorial Optimization*, **8**, 97-108. http://dx.doi.org/10.1023/B:JOCO.0000021940.40066.0c

[6] Coffman, E.G., Garey, M.R., Johnson, D.S. and LaPaugh, A.S. (1985) Scheduling File Transfers. *SIAM J. Computing*, **14**, 744-780. http://dx.doi.org/10.1137/0214054

[7] Krawczyk, H. and Kubale, M. (1985) An Approximation Algorithm for Diagnostic Test Scheduling in Multicomputer Systems. *IEEE Trans. Computers*, **34**, 869-872. http://dx.doi.org/10.1109/TC.1985.1676647

[8] Marx, D. (2009) Complexity Results for Minimum Sum Edge Coloring. *Discrete Applied Mathematics*, **157**, 1034-1045. http://dx.doi.org/10.1016/j.dam.2008.04.002

[9] Goldberg, A.V. and Tarjan, R.E. (1987) Solving Minimum Cost Flow Problems by Successive Approximation. *Proc.* 19*th ACM Symposium on the Theory of Computing*, 7-18. http://dx.doi.org/10.1145/28395.28397

[10] Goldberg, A.V. and Tarjan, R.E. (1989) Finding Minimum-Cost Circulations by Canceling Negative Cycles. *J. ACM*, **36**, 873-886. http://dx.doi.org/10.1145/76359.76368