Coloring Weighted Series-Parallel Graphs

Gašper Fijavž
Faculty of Computer and Information Science
E-mail: gasper.fijavz@fri.uni-lj.si

Keywords: graph coloring, circular coloring, weighted graphs

Received: December 31, 2003

Let G be a series-parallel graph with integer edge weights. A p-coloring of G is a mapping of vertices of G into $\mathbb{Z}_p$ (ring of integers modulo p) so that the distance between colors of adjacent vertices u and v is at least the weight of the edge uv. We describe a quadratic time p-coloring algorithm where p is either twice the maximum edge weight or the largest possible sum of three weights of edges lying on a common cycle.

Povzetek: Opisano je barvanje grafov.

1 Introduction

The motivation of the problem is twofold. An instance of coloring edge weighted graphs is the *channel assignment problem*, cf. [4]. On the other hand, traditional vertex coloring of (unweighted) graphs can be viewed as a circular one—consider the colors to lie in an appropriate ring of integer residues. *Circular colorings* of graphs, see [8] for a comprehensive survey, where we allow the vertices to be colored by real numbers (modulo p) model several optimization problems better than traditional colorings of graphs. Circular chromatic number, the minimum p for which a circular coloring exists, is a refinement of the chromatic number of a graph, and similarly NP-hard to compute.

If the largest complete minor in (an unweighted graph) G has k vertices and $k < 6$, then the valid cases of Hadwiger conjecture imply $\chi(G) \leq k$, see [7].

Let $G = (V, E, w)$ be a weighted graph (where (V, E) is the *underlying unweighted* graph) with edge weights w (and $w : E \to [1, \infty)$). We can, similarly as in the unweighted case, define the size of the largest complete minor, see [5, 6, 3]: the size of the largest weighted K_2-minor in G is twice the maximal edge weight, and for the size of the largest weighted complete K_3 minor we have also to consider the biggest possible sum of weights of three edges lying on the common cycle. If G is a series parallel graph then the largest of the above-mentioned quantities is called the *weighted Hadwiger number* of G, which we denote by $h(G)$.

The weighted case of Hadwiger conjecture is valid only for graphs satisfying $h(G) < 4$, i.e., it is true that if $h(G) < 4$, then the weighted chromatic number of G, which we denote by $\chi_w(G)$, is at most $h(G)$ [3]. If a weighted graph G is not series-parallel, then it may occur that $\chi_w(G) > h(G)$, see [3] for examples.

Hence, for series-parallel weighted graphs $h(G)$ is a natural upper bound for $\chi_w(G)$. We present an algorithm for $h(G)$-coloring weighted series-parallel graphs. As opposed to results in [3], the coloring algorithm presented here successfully colors series-parallel graphs with at most $h(G)$ colors even if the ratio between maximal and minimal edge weights exceeds 2.

2 Definitions and preprocessing

Let $\mathbb{N}$ denote the set of positive integers and let $\mathbb{Z}_p$ denote the ring of integers modulo p. If $x, y \in \mathbb{Z}_p$ then we denote the distance between x and y in $\mathbb{Z}_p$ by $|x - y|_p$. Let $G = (V, E, w)$ be a weighted series-parallel graph. Series-paralel graphs are constructed by first pasting triangles along edges (starting with a triangle), and then deleting edges [2]. In order to avoid computational difficulties concerning real numbers we shall assume that weights are integers, $w : E \to \mathbb{N}$. A *p-coloring* of G is a mapping $c : V \to \mathbb{Z}_p$ so that for every edge $e = uv$ the condition

$$|c(u) - c(v)|_p \geq w(uv)$$

is satisfied. Given a p-coloring c and an edge $e = uv$, we call $|c(v) - c(u)|_p$ the *span* of e, denoted by $\mathrm{span}(e)$, and say that e is *tight* if its span equals its weight. We shall also say that p is the *size* of the *color space* $\mathbb{Z}_p$.

2.1 Tree decomposition

Tree decomposition, see [2] for the theoretical background, of a series-parallel graph can be computed in linear-time [1]. Given a tree-decomposition $(T_G, \mathcal{V})$ of G we can by adding edges to G (and setting their weights to 1) assume that G is an edge-maximal series parallel graph. The parts $\mathcal{V}$ of the decomposition are exactly the edges and triangles of G. Two parts are adjacent (in T_G) if and only if one part is a triangle t, the other is an edge e, and e is incident with t.

Hence, G is 2-connected, and given distinct edges e and f from G, there exists a cycle containing both. We shall use both G and its tree decomposition (T_G, V) for storing the graph during the course of the coloring algorithm.

Let $e = v_1v_2$ be an edge in G. If $\{v_1, v_2\}$ is a separator in G we say that an edge e is a *separating* edge in G, and e is called *nonseparating* otherwise. If $e = v_1v_2$ is separating and $G - \{v_1, v_2\}$ consists of k components $C_1, \ldots, C_k$, then G_i $(i = 1, \ldots, k)$ denotes the graph (infact its representation) induced by vertices of C_i and $\{v_1, v_2\}$. We call G_i's $(i = 1, \ldots, k)$ the *e-splits* of G.

Throughout the algorithm we shall keep track whether an edge e is a separating edge of G. This can be easily seen from T_G, namely, an edge e is nonseparating if it is adjacent to a single triangle in T_G.

Let $t = e_1e_2e_3$ be a triangle (t contains edges e_1, e_2, and e_3) in G. Let us further assume that e_1 is a separating edge and let $G_0, G_1 \ldots, G_k$ be all e_1-splits of G, so that G_0 contains triangle t as its subgraph. Then the (graph) union $G_1 \cup \ldots \cup G_k$ is called the (t, e_1)*-fragment* of G and is denoted by $G(t, e_1)$. If e is nonseparating, and there exists a triangle t containing e (there may be at most one), then the (t, e)-fragment of G is the graph containing only e together with its endvertices. We call a graph *trivial* if it contains at most two vertices.

2.2 Heavy cycle, heavy triangle

As noted in the introduction $h(G)$, the hadwiger number of G, equals either twice the weight of the heaviest edge or the sum of three largest edge weights of edges lying on a common cycle. It is the latter option that is more appealing to our problem.

Let $t = f_1f_2f_3$ be a triangle in G. Define $G_1 = G(t, f_1)$, $G_2 = G(t, f_2)$, and $G_3 = G(t, f_3)$. If G_i $(i = 1, 2, 3)$ is trivial, then we say that $e_i = f_i$ is a realizing edge of t (in G_i). If G_i is not trivial then every heaviest edge in G_i can be chosen as a realizing edge of t (in G_i). *Weight of a triangle* t, $w(t)$, is defined as the sum of edge weights of edges realizing t. Clearly enough, the realizing edges of a triangle lie on a common cycle in G.

Let e_1 and e_2 be distinct edges with largest edge weights in G. Triangle t is called a *heavy triangle* if $w(t)$ equals $h(G)$, and both e_1 and e_2 are realizing edges of G. It may occur that no triangle is heavy in G. In this case we can by increasing weight of a single edge construct a heavy triangle in G while not increasing $h(G)$. This is the essence of the procedure heavyTriangle described in the next section.

By scanning through the edges of G, we find some heaviest edge $e_a = u_av_a$. Next we run

$$\mathsf{heavyTriangle}(G, e_a, e_a, e_a) \mapsto h(G); t, f_a, f_b, f_c; e_b, e_c, P.$$

Finally, we set $p = h(G)$, $c(u_a) = 0$, $c(v_a) = w(e_a)$ and run the main coloring procedure

$$\mathsf{color}(G, p, t; f_a, f_b, f_c; e_a, e_b, e_c; P)$$

using a heavy triangle $t = f_af_bf_c$ with its realizing edges e_a, e_b, and e_c as arguments.

3 Coloring algorithm

The coloring algorithm is recursive. Given a graph G with two precolored adjacent vertices u_a and v_a we split G along a carefully chosen edge(s) into several subgraphs, say G_0, $G_1, \ldots$ Only one of these, say G_0, contains both u_a and v_a, and it is the first one to get colored. We find colorings of $G_1, G_2, \ldots$ recursively, taking care that exactly two vertices of G_j are already colored when it is G_j's turn.

3.1 Looking for a heavy triangle

We shall first describe the routine heavyTriangle. The input for his routine consists of weighted graph G, edges e_a and e_b (e_a is heaviest in G, and e_b is either second heaviest in G or $e_a = e_b$), and a path $P \subseteq T_G$ linking edges e_a and e_b (P is trivial in case $e_a = e_b$).

The routine heavyTriangle outputs, apart from the possibly new e_b and P, also the hadwiger number $h(G)$, a heavy triangle $t = f_af_bf_c$, and its third realizing edge e_c. We set the notation so that $e_a \in E(G(t, f_a))$, $e_b \in E(G(t, f_b))$, and $e_c \in E(G(t, f_c))$, and assume that $h(G) = w(e_a) + w(e_b) + w(e_c)$.

We use the following shorthand

$$\mathsf{heavyTriangle}(G, e_a, e_b, P) \mapsto h(G); t, f_a, f_b, f_c; e_b, e_c, P.$$

The routine runs as follows:

(T1) <u>if</u> $e_a = e_b$ <u>then</u>

we find some second heaviest edge in G and adjust P so that P links e_a and the newly determined e_b. Hence $e_a \neq e_b$.

(T2) For every triangle τ we compute the realizing edges and its weight $w(\tau)$. This can be done by tracing T_G starting from e_a first. Hence e_a is one of the realizing edges in every triangle τ. By retracing towards e_a from the leaves of T_G we compute the other two realizing edges of every triangle recursively. Finally we set that e_b is one of the realizing edges in every triangle lying in P (in the direction from e_b to e_a).

(T3) Find the triangle t' with largest possible $w(t')$. Set $h(G) = \max\{w(t'), 2w(e_a)\}$.

(T4) <u>If</u> $h(G) > w(t')$ <u>or</u>

$h(G) = w(t')$ <u>and</u> e_b is not one of the realizing edges of t'

<u>then</u> do the following:

Let $t = f_af_bf_c$ be an arbitrary triangle from P so that $e_a \in G(t, f_a)$ and $e_b \in G(t, f_b)$. Set $e_c = f_c$ and increase the weight of $e_c = f_c$ by setting $w(e_c) = h(G) - w(e_a) - w(e_b)$. Note that increasing weight of e_c does not increase $h(G)$, as e_a and e_b are heaviest edges in G.

(T5) <u>If</u> $h(G) = w(t')$ <u>and</u>

e_b is one of the realizing edges in t'

<u>then :</u>

By (T2) e_a is also one of the realizing edges in t'. Set $t = t'$. Further, set e_c to be the third realizing edge in $t = f_af_bf_c$ where the notation of edges in t is chosen so that $e_a \in E(G(t, f_a))$, etc.

(T6) output $h(G); t, f_a, f_b, f_c; e_b, e_c; P$.

It is not difficult to see that heavyTriangle runs in linear time.

3.2 Recursion

We shall first describe a routine for coloring a graph with small edge weights. Let $e = uv$ be the heaviest edge in G, and assume that $p \geq 3w(e)$, where p denotes the size of the color space. Let us also assume that colors $c(u)$ and $c(v)$ are already determined so that the $\mathrm{span}(e)$ is at most $p - 2w(e)$. Procedure colorCgraph with G, p, and e as its input (satisfying the above conditions) extends the coloring c to the remaining vertices of G. This can be done by tracing along T_G starting at e, and taking care that every edge $f \in G$ satisfies $\mathrm{span}(f) \leq p - 2w(e)$ (as $w(f) \leq w(e)$). It is easy to implement colorCgraph to run in linear time.

We turn our attention to coloring the graph in case its edge weights (at least some of them) are large when compared to $h(G)$. Let G be a weighted graph, p an upper bound for $h(G)$, $t = f_a f_b f_c$ a heavy triangle, and e_a, e_b, and e_c its realizing edges (so that $e_a \in E(G(t, f_a))$, etc.). Let P be a path in T_G joining e_a and e_b, and suppose that a coloring of endvertices of e_a is given so that e_a is tight. Then

COLORING PRINCIPLE. *With the assumptions as above the procedure* color *extends the coloring c to the rest of G so that*

(i) *apart from e_a the edge e_b is also tight, and*
(ii) $\mathrm{span}(e_c) \leq p - w(e_a) - w(e_b)$.

The call

$$\mathsf{color}(G, p, t; f_a, f_b, f_c; e_a, e_b, e_c; P)$$

splits into three cases, and exactly one of them applies. These three cases will also serve as a recursive proof that a graph can indeed be colored according to the principle. The first case (C1) serves as the recursion basis, the last two cases (C2) and (C3) serve as recursion steps.

(C1) if G contains a single triangle t then.

In this case $e_a = f_a$, $e_b = f_b$, and $e_c = f_c$. Let u and v be the (colored) endvertices of e_a, and let w be the common endvertex of e_b and e_c. There exists a unique color $c(w)$ so that e_b is tight and $\mathrm{span}(e_c) = p - w(e_a) - w(e_b)$. Hence, we can extend the coloring to G according to the coloring principle.

exit

(C2) if e_a is a separating edge in G then

let $G_0, G_1, \ldots, G_k$ be the e_a-splits of G so that G_0 contains $e_b, e_c, f_a, f_b, f_c, t$, and P. We first color G_0 by calling

```
color($G_0, p, t; f_a, f_b, f_c; e_a, e_b, e_c; P$)
```

and then take care of the other splits:

```
for $i = 1$ to $k$ do
  heavyTriangle($G_i, p, e_a, e_a, e_a$) $\mapsto$
      $h(G_i), t_i; f_{ai}, f_{bi}, f_{ci}; e_{bi}, e_{ci}, P_i$
for $i = 1$ to $k$ do
  color($G_i, p, t_i; f_{ai}, f_{bi}, f_{ci}; e_a, e_{bi}, e_{ci}; P_i$)
exit
```

(C3) if e_a is nonseparating in G then

we first increase weights of f_b and f_c by setting $w(f_c) = w(e_c)$ and $w(f_b) = w(e_b)$.

Let G_a be the graph containing $G(t, e_a)$ and triangle t. Observe that either G_a contains at least two triangles or at least one of $G(t, f_b)$, $G(t, f_c)$ is not trivial (i.e. at least one of f_b, f_c is separating in G). Let P_a be the subpath of P linking f_b and e_a. Since $w(f_b) = w(e_b)$ the edge f_b is second heaviest in G_a.

```
if $e_a = f_a$ then
  color($G_a, p, t; e_a, f_b, f_c; e_a, f_b, f_c; P_a$)
```

Note that in the above case G_a contains a single triangle t as e_a is not separating in G.

else

Observe that $w(f_a) \leq w(e_b) = w(f_b)$, as e_b is second heaviest in G and $e_a \neq f_a$. Hence, we increase the weight by setting $w(f_a) = w(f_b)$, which makes f_a second heaviest in $G(t, f_a)$. Let P' be the subpath of P_a linking e_a and f_a, let G' be the graph $G(t, f_a)$, and let G'' be the subgraph of G induced by triangle t.

```
  heavyTriangle($G', p, e_a, f_a, P'$) $\mapsto$
      $h(G'), t'; f'_a, f'_b, f'_c; f_a, e'_c, P'$
  color($G', p, t'; f'_a, f'_b, f'_c; e_a, f_a, e'_c; P'$)
```

After coloring G' the edge f_a is tight and we also

```
  color($G'', p, t; f_a, f_b, f_c; f_a, f_b, f_c; f_a t f_b$)
end if
```

Note that at this point endvertices of both f_b and f_c are colored. What is more, f_b is tight, and by recursion, $\mathrm{span}(f_c) \leq p - w(e_a) - w(f_b) = p - w(e_a) - w(e_b)$.

Finally we settle the uncolored parts.

```
if $f_b$ is separating in $G$ then
  heavyTriangle($G(t, f_b), f_b, e_b, e_b P f_b$) $\mapsto$
      $h(G(t, f_b)), t_1; f_{a1}, f_{b1}, f_{c1}; e_{b1}, e_{c1}, P_1$
  color($G(t, f_b), p, t_1; f_{a1}, f_{b1}, f_{c1}; f_b, e_{b1}, e_{c1}; P_1$)
if $f_c$ is separating in $G$ then
  colorCgraph($G(t, f_c), p, f_c$)
exit
```

4 Time complexity

The last section is devoted to estimating the speed of the coloring algorithm.

TIME COMPLEXITY. *There exists a constant C so that for every weighed series parallel graph G of order n, the running time of the described coloring algorithm is bounded above by Cn^2. In other words, we can $h(G)$-color a weighted series parallel graph G in quadratic time.*

As already mentioned, the preprocessing takes linear amount of time. After preprocessing G is an edge maximal series parallel graph. If G contains $n + 3$ vertices then G contains n triangles, $2n + 1$ edges, and $3n$ lines (edge–

triangle incidencies). All these quantities are equally appropriate for measuring the size of the problem.

Let $T(n)$ denote the maximal running time for the color procedure taking a graph G with n triangles as an input. We have to show that $T(n) \leq Cn^2$ assuming $T(m) \leq Cm^2$ for every $m < n$.

Let $D_0 n$ be the upper bound for the running times of both heavyTriangle and colorCgraph if they take a graph G containing n triangles as input.

A call of color with G as its argument takes one of the three possible options: (C1), (C2), or (C3). The running time of (C1) is bounded above by a constant, say D_1.

If (C2) applies let $G_0, G_1, \dots, G_k$ be the splits. Observe that $k \geq 1$. Since G_i's together contain exactly n triangles, the recursively called procedures heavyTriangle cumulatively take no more than $D_0 n$ running time.

Let $(n_0, n_1, n_2, \dots, n_k)$ be a proper (integer) partition of n, i.e. $n_0, n_1, n_2, \dots, n_k \geq 1$, $k \geq 1$, and $n_0 + n_1 + \dots + n_k = n$. Then

$$n_0^2 + n_1^2 + \dots + n_k^2 \leq n_0^2 + (n_1 + \dots + n_k)^2 \leq (n-1)^2 + 1 \tag{1}$$

Now (1) implies that the cumulative running time of recursive calls of procedure color in (C2) is bounded from above by $C(n-1)^2 + C$. Summing it all up, the running time of (C2) is bounded from above by $C(n-1)^2 + C + D_0 n + D_2 n$ if we use at most $D_2 n$ time for running the loops (excluding time for recursive calls of heavyTriangle and color).

The case when (C3) applies is settled similarly as above. Assume that the base running time of (C3) (i.e. the running time excluding running times of recursive calls of color, colorCgraph, and heavyTriangle) is bounded by constant D_3. Recursive color-ing and colorCgraph-ing takes at most $C(n-1)^2 + C$, and heavyTriangle-s take at most $D_0 n$ time.

Combining all three possibilities yields

$$\begin{aligned} T(n) &\leq \max\{D_1,\ C(n-1)^2 + C + D_0 n + D_2 n, \\ &\qquad C(n-1)^2 + C + D_0 n + D_3\} \\ &\leq \max\{D_1,\ Cn^2 + (-2Cn + 2C + D_0 n + D_2 n), \\ &\qquad Cn^2 + (-2Cn + C + D_0 n + D_3)\}, \end{aligned}$$

which is, if C is large enough, at most Cn^2. This proves the assertion on time complexity.

Acknowledgment

The author's research was conducted while visiting University of Hannover under sponsorship of Alexander von Humboldt Foundation. Both hospitality of the university and help of the foundation are greatly acknowledged.

References

[1] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.

[2] R. Diestel. Graph theory. Springer, New York, 1997.

[3] G. Fijavž. Hadwiger's conjecture for circular colorings of edge-weighted graphs. *Discrete Math.*, to appear.

[4] C. McDiarmid. Discrete mathematics and radio channel assignment. In *Recent advances in algorithms and combinatorics*, volume 11 of *CMS Books Math./Ouvrages Math. SMC*, pages 27–63. Springer, New York, 2003.

[5] B. Mohar. Circular colorings of edge-weighted graphs. *J. Graph Theory*, 43:107–116, 2003.

[6] B. Mohar. Hajós theorem for colorings of edge-weighted graphs. *Combinatorica*, 25:65-76, 2004.

[7] B. Toft. A survey of Hadwiger's conjecture. *Congr. Numer.*, 115:249–283, 1996. Surveys in graph theory (San Francisco, CA, 1995).

[8] X. Zhu. Circular chromatic number: a survey. *Discrete Math.*, 229(1-3):371–410, 2001. Combinatorics, graph theory, algorithms and applications.