AN EDGE COLORING HEURISTIC BASED ON VIZING'S THEOREM

Tiago Januario and Sebastián Urrutia[1]

[1]Department of Computer Science
Universidade Federal de Minas Gerais
Belo Horizonte, Minas Gerais - Brazil
{januario,surrutia}@dcc.ufmg.br

Abstract. *Vizing's theorem establishes that the edges of a simple graph can be properly colored using at most $\Delta+1$ colors. Therefore, the minimum number of colors that are needed to color the edges of a graph G satisfies $\Delta \leq \chi'(G) \leq \Delta+1$ and a polynomial algorithm to obtain a $\Delta+1$ coloring of the graph is known. Heuristics for the edge coloring problem do exist in the literature and can be justified in situations where a $\Delta+1$ coloring of the graph is not good enough. In this paper, a constructive heuristic based on the proof of Vizing's theorem is proposed. The heuristic tries to find a Δ coloring of the graph, but whenever that coloring is not found, a solution using exactly $\Delta+1$ colors is returned. Therefore, the heuristic obtains solutions whose cost is, in the worst case, one color above the optimum.*
KEYWORDS. *Graph Theory, Edge Coloring, Vizing's Theorem.*

1. Introduction

An assignment of colors to the edges of a graph $G = (V, E)$ is a function $\lambda : E \rightarrow C$, where C is a non-empty set of colors. A color is present on a vertex, if an edge incident on the vertex has that color; otherwise, the color is free or available on that vertex. A conflict in an assignment of colors is the existence of two edges with the same color incident to a common vertex. A proper edge coloring of G is an assignment of colors to every edge of G without conflicts and a proper partial edge coloring is an assignment of colors without conflicts such that some edges of G may be uncolored.

Edge coloring is an important problem in graph theory. It consists in coloring the edges of a graph such that two edges incident to the same vertex get different colors. For a graph G, the minimum number of colors necessary to obtain a proper edge coloring is called the *chromatic index* $\chi'(G)$. Let Δ be the maximum vertex degree on the graph. It is easy to see that $\chi'(G) \geq \Delta$ because all the edges incident to a maximum degree vertex have to be assigned different colors. Vizing's theorem proves, in a constructive way, that every simple graph may be colored in polynomial time with no more than $\Delta+1$ colors [Vizing 1964]. The proof immediately yields a $O(|E| \cdot |V|)$ time algorithm.

A simple graph G is said to be in Class 1 if $\chi'(G) = \Delta$ and in Class 2 if $\chi'(G) = \Delta+1$. Vizing's theorem states that there are no other possibilities: all graphs are either in Class 1 or Class 2 [Vizing 1964]. It is $\mathcal{NP}$-Complete to determine whether a graph is in Class 1 or in Class 2 [Holyer 1981].

Many real-world problems may be modelled by edge coloring, including scheduling of tasks requiring the cooperation of two processors [Kosowski 2009],

file transfer operations [Nakano et al. 1995], and channel assignment in wireless networks [Hsu et al. 2006].

Metaheuristics have been applied to edge coloring. While Khuri et al. [Khuri et al. 2000] proposed using genetic algorithms, Enochs and Wainwright [Enochs and R. 2001] developed a simulated annealing heuristic. Due to the existence of Vizing's theorem, these time consuming heuristics can be only justified in cases where a $\Delta+1$ coloring of the graph is not good enough and a Δ coloring should be found whenever such a coloring exists.

A set of fast edge coloring heuristics was proposed by [Hilgemeier et al. 2003]. These heuristics do not guarantee to find a proper edge coloring of a graph but they might be useful in contexts where very solutions must be generated in a short amount of time.

In [Sanders and Steurer 2005] the authors proposed an edge coloring heuristic that attempts to use a minimum number of colors. They suggest that it is possible to speed up their algorithm by starting the coloring process with only Δ colors. Before adding more colors, the heuristic would try to color edges by shifting alternating paths. At the best of our knowledge, the idea was not implemented.

In this work, we propose a constructive heuristic for edge coloring based on a proof of Vizing's theorem that, while seeking solutions using exactly Δ colors, guarantees obtaining solutions using no more than $\Delta+1$ colors. The heuristic requires $O(|E|\cdot|V|\cdot\Delta)$ time and always find a proper edge coloring of the graph.

The rest of this paper is organized as follows. In the next section we describe a proof of Vizing's theorem as a basis for the proposed heuristic. Section 3 describes the constructive heuristic being proposed. In Section 4 we present computational results concerning the new heuristic and evaluate its performance against state of the art heuristics. In the last section we make some concluding remarks.

2. Vizing's Theorem

The main result on edge coloring came in 1964 with Vizing's theorem [Vizing 1964] which says that every graph G may be colored with at most $\Delta+1$ colors, that is,

$$\chi'(G) \leq \Delta+1 \tag{1}$$

There exist several proofs of Vizing's theorem in the literature, see for example [Diestel 2005, Dijkstra and Rao 1990, Gabow et al. 1985, Gould 1988, Misra and Gries 1992]. All proofs are based on augmenting the coloring of the graph by coloring a new edge at each iteration with color $c \in \{c_1, c_2, \ldots c_{\Delta+1}\}$. The proofs show how to color an arbitrary uncolored edge of a partially colored graph (which may require changing the color of colored edges to maintain validity) never exceeding $\Delta+1$ different colors. This procedure is repeated until all edges are colored.

The proof of the theorem in which our new heuristic is based may be found in [Gould 1988]. In the rest of this section, we partially describe the proof as a basis for our heuristic.

Let $e_0 = wv_0$ be an uncolored edge of a graph G partially colored with no more than $\Delta+1$ colors. Observe that, since both w and v_0 have at most $\Delta-1$ colored incident edges,

there are at least two available colors in both of them. Let us call $Free(v)$ the set of colors available at vertex v.

If $Free(w) \cap Free(v_0) \neq \varnothing$ we can simply choose one color from that intersection to color edge e_0.

Assume now, that the intersection is empty. Let α_0 be an available color at v_0 and β an available color at w. Let P be the maximal path that starts at v_0 and its edges are alternately colored with colors β and α_0. Two cases may arise: P ends at a vertex different from w or P ends at w. In the first case, the coloring may be augmented by switching the colors of the edges on the path P (edges colored with α_0 are recolored with β and vice-versa) and coloring edge e_0 with color β that is now available in vertex v_0 (see Fig. 1).

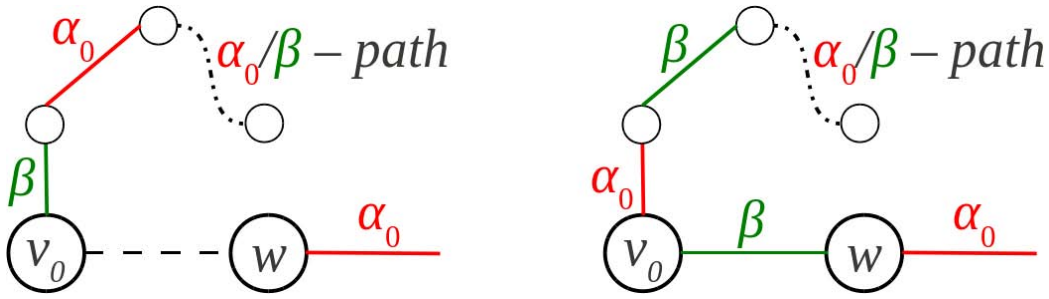

Figure 1. Color β is available at v_0 after the switch.

Assume now that the path P ends at w. If we switch the edges of path P, color β will be available in vertex v_0 but, on the other hand, it will not be available in vertex w (see Fig. 2).

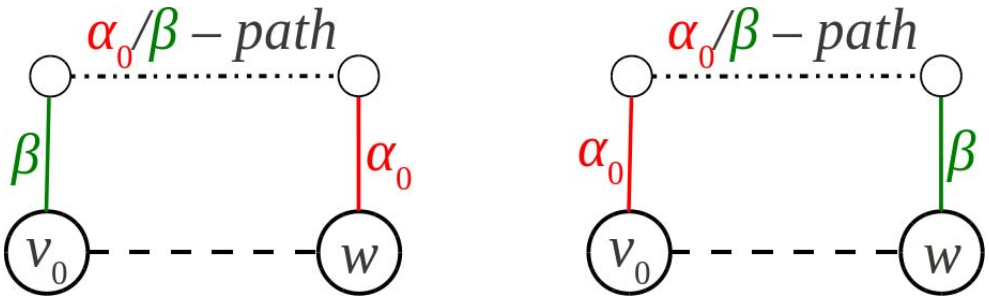

Figure 2. A situation where the path P ends at w.

Let v_1 be the vertex adjacent to w in P and $e_1 = wv_1$ which is colored with α_0. Remove color α_0 from e_1 and color e_0 with that color (see Fig. 3).

Now the problem consists in re-coloring edge e_1 and we can apply the same procedure we used with edge e_0 to try to assign a color to edge e_1. Note that if $Free(w) \cap Free(v_1)$ is also empty, when considering a color α_1 available at v_1, this color must be different from α_0 in order to avoid cycling. Such color always exists because v_1 has at most $\Delta - 1$ colored incident edges.

We may continue in this way until $Free(w) \cap Free(v_i)$ is not empty or the path P does not end at w. The rest of the proof of Vizing's theorem [Gould 1988], shows that this will be eventually the case after at most Δ iterations. This proof of Vizing's theorem immediately yields an $O(|E| \cdot |V| \cdot \Delta)$ time algorithm to obtain a $\Delta + 1$ proper coloring of a

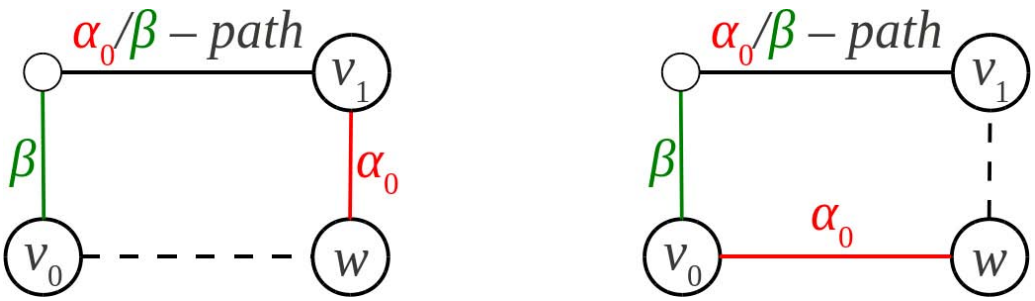

Figure 3. Color α_0 moves from wv_1 to wv_0.

simple graph. In the next section we adapt the algorithm derived from this proof to try to obtain a Δ proper coloring of the graph.

3. Edge coloring heuristic

The heuristic proposed in this paper, is an adaptation of the algorithm derived from the proof of Vizing's theorem described in the previous section. We call that algorithm, Vizing's algorithm, and its adaptation, the Δ Vizing heuristic (ΔVh).

ΔVh considers a set $\{c_1, c_2, \ldots c_\Delta\}$ of colors and tries to apply Vizing's algorithm using that set instead of the set with $\Delta+1$ colors. Note that in this case, the observation that there are at least two available colors in every vertex with an uncolored incident edge is no longer valid. Indeed, if while executing the coloring algorithm, there is a vertex whose degree is equal to Δ and, all but one of its incident edges are colored, then there is only one available color at that vertex.

Therefore, considering a set of just Δ colors, Vizing's algorithm may fail when it has to set variable α_i to a color available at v_i different from α_{i-1}. In this case, our proposed heuristic augments the set of available colors including color $c_{\Delta+1}$ and continues its execution as Vizing's algorithm.

Algorithm 1 depicts the Δ Vizing heuristic. In line 1 variable *taboo* is set to *nil*. This variable represents the color that cannot be used to recolor the current edge (e_0), and is different from *nil* only when the current edge has to be recolored (when it was uncolored in the previous iteration). In line 2 the algorithm assigns the same set of Δ different available colors to each vertex of the graph. Then, the algorithm iterates until all edges are colored. If no edge was uncolored in the previous iteration, in line 5 the algorithm chooses the current edge at random among the uncolored edges of the graph. Otherwise, the current edge is the edge that needs to be recolored. In lines 6 to 8, whenever the intersection of the sets of available colors in both ends of the current edge is not empty, the algorithm assigns a color from that intersection to the current edge and terminates the current iteration to consider another edge. Note that the procedure $SetColor(e_0, \varphi)$ besides assigning the color φ to edge e_0, removes that color from the sets of available colors at both ends of the edge.

Assume now that the intersection of the sets of available colors in both ends of the current edge is empty. If there is only one available color at v_0 and is equal to the *taboo* color (note that this may only happen while recoloring), in lines 11 to 13 the algorithm inserts the new color $c_{\Delta+1}$ to the set of available colors at every vertex of the graph, colors the current edge with the new color and terminates the current iteration to consider a new uncolored edge. Observe that, if the algorithm execution reaches lines 11 to 13 it will not

find a Δ coloring of the graph and will return a $\Delta+1$ proper coloring.

Assume now that there is at least one color available at v_0 different from *taboo*. The algorithm assigns to α_0 one of such colors chosen at random in line 15. Then, in line 17, if the current edge is not being recolored, the algorithm assigns to β a color available at w, otherwise β is already an available color at w that was selected in a previous iteration. Next, the algorithm computes the maximal path P starting in v_0 and whose edges alternate between edges colored with β and edges colored with α_0. If this path does not end at w, in lines 20 to 22 the algorithm switches the colors of the edges on the path from β to α_0 and vice-versa, assigns the color β (which is now available at v_0) to e_0 and terminates the current iteration to consider another uncolored edge.

Finally, assume that the path P ends at w. In this case, in lines 24 to 29, the algorithm removes the color α_0 from the last edge of P, assigns the color α_0 to the current edge e_0, set the new current edge as the last edge of P and sets variable taboo to α_0 in order to avoid, in the next iteration, recoloring the new current edge with the color it just lost. Observe that the procedure $UnSetColor(e_1)$, besides removing the current color from the edge e_1, inserts that color in the sets of available colors at both ends of the edge.

The asymptotic complexity of the Δ Vizing heuristic is the same as Vizing's algorithm, that is, $O(|E| \cdot |V| \cdot \Delta)$. We acknowledge the existence of other edge coloring algorithms with lower complexity [Gabow et al. 1985]. In fact, the best known proof of Vizing's theorem [Misra and Gries 1992] immediately yields a $O(|E| \cdot |V|)$ algorithm. In order to attain that complexity, the algorithm computes $Free(w) \cap Free(v_i)$ and the path P only once per iteration, which ends up hurting the effectiveness of a heuristic based on that proof of the theorem. Moreover, experimental results showed that, despite their different complexities, the algorithms derived from the two proofs are equivalent in terms of execution times.

4. Computational results

The heuristic described in Section 3 was coded in C++ and compiled with version 4.4.1 of the g++ compiler with the optimization flag -O3. The experimental results showed in this section were taken on a AMD Sempron 3600+ processor (1.8 GHz, 256kB cache) with 2 GB of RAM running Linux.

We compare ΔVh against other heuristics from the literature in terms of solution quality and execution times. The problem instances are the same considered in [Enochs and R. 2001] excluding complete graphs and multigraphs (graphs with parallel edges). The chromatic index of complete graphs is $\Delta+1$ when the number of vertices is odd and, in that case, Vizing's algorithm finds an optimal coloring. When the number of vertices is even, complete graphs have chromatic index equal to Δ and there exist several fast algorithms that find Δ colorings for such graphs (see for example [Costa et al. 2012]).

Table 1 compares ΔVh against the constructive heuristics proposed in [Hilgemeier et al. 2003] (observe that the authors considered just a subset of the instances tackled in [Enochs and R. 2001]) showing results of a unique execution of the heuristics on each problem instance. The first column shows the instance name. The next six columns show solution costs and execution times in seconds for the three constructive heuristics evaluated in [Hilgemeier et al. 2003]. The last two columns show solution costs and execution times for the heuristic proposed in this work (ΔVh). The results concerning the

Algorithm 1: Edge Coloring Heuristic

```
1  taboo ← nil ;
2  Let Free(v) ← {c_1, c_2, ... c_Δ} for every v ∈ V ;
3  while there are uncolored edges in G do
4      if taboo = nil then
5          e_0 = wv_0 ← uncolored edge of G;
6      if ∃φ ∈ {Free(v_0) ∩ Free(w)} then
7          SetColor(e_0, φ) ;
8          taboo ← nil ;
9      else
10         if {φ ∈ Free(v_0) | φ ≠ taboo} = ∅ then
11             Free(v) ← Free(v) ∪ {c_{Δ+1}} for every v ∈ V ;
12             SetColor(e_0, c_{Δ+1}) ;
13             taboo ← nil ;
14         else
15             α_0 ← a color in {φ ∈ Free(v_0) | φ ≠ taboo} ;
16             if taboo = nil then
17                 β ← a color in Free(w) ;
18             Compute P, the maximal alternating β-α_0-path starting in v_0 ;
19             if P does not end at w then
20                 Switch(P) ;
21                 SetColor(e_0, β) ;
22                 taboo ← nil ;
23             else
24                 e_1 ← last edge of P ;
25                 UnSetColor(e_1) ;
26                 SetColor(e_0, α_0) ;
27                 e_0 ← e_1 ;
28                 v_0 ← end vertex of e_0 different from w ;
29                 taboo ← α_0 ;
```

heuristics from the literature were taken from [Hilgemeier et al. 2003] and were obtained on a AMD Athlon XP2200+ processor (1.8 GHz, 256kB cache) with 512 MB RAM running Linux.

benchmark	linE		antN		ant1		ΔVh	
name	colors	secs	colors	secs	colors	secs	colors	secs
myciel6	**47**	-	**47**	-	**47**	0.01	**47**	-
myciel7	**95**	0.02	**95**	0.01	**95**	0.03	**95**	-
le450_5c	**66**	0.08	**66**	0.08	**66**	0.15	**66**	-
le450_15a	**99**	0.06	**99**	0.07	**99**	0.12	**99**	-
le450_15c	**139**	0.40	**139**	0.42	**139**	0.76	**139**	0.01
le450_15d	**138**	0.42	**138**	0.42	**138**	0.78	**138**	0.01
le450_25c	**179**	0.52	**179**	0.52	**179**	0.96	**179**	0.01
le450_25d	**157**	0.53	**157**	0.51	**157**	0.96	**157**	0.01

Table 1. Computational results for small graphs.

ΔVh clearly outperforms the constructive heuristics from the literature in terms of execution time (even considering the different hardware). In terms of solution quality, all heuristics found a Δ coloring of the graph for every benchmark instance considered.

In order to compare ΔVh against metaheuristic approaches, we developed a multi start heuristic. This heuristic loads the graph once and then executes ΔVh 1000 times. Table 2 shows the computational results. The first column shows the instance name. The next two columns show solution costs and execution times for the simulated annealing heuristic introduced in [Enochs and R. 2001] (running times were taken on a Intel Pentium III, 700Mhz computer). The next column shows solution costs for the genetic algorithm proposed in [Khuri et al. 2000] (note that the authors did not report executions times). The last three columns concern ΔVh results. The first of these columns shows the best solution cost found over the 1000 executions of the heuristic. The second column in the group shows the success rate, this is, the number of times the best solution was obtained divided by 1000. The last column shows the computational time of loading the instance once and executing the heuristic 1000 times.

benchmark	SA		GA	ΔVh		
name	colors	secs	colors	colors	suc. rate	secs
myciel3	**5**	1	**5**	**5**	1	0.01
myciel4	**11**	1	**11**	**11**	1	-
myciel5	**23**	1	**23**	**23**	1	0.03
myciel6	**47**	2	**47**	**47**	1	0.12
myciel7	**95**	3	**95**	**95**	1	0.69
le450_5a	**42**	25	43	**42**	1	1.59
le450_5b	**42**	25	43	**42**	1	1.60
le450_5c	**66**	93	67	**66**	1	3.12
le450_5d	**68**	88	69	**68**	1	3.06
le450_15a	**99**	35	**99**	**99**	1	3.03
le450_15b	**94**	36	**94**	**94**	1	3.00
le450_15c	**139**	234	**139**	**139**	1	6.25
le450_15d	**138**	231	140	**138**	1	6.31
le450_25a	**128**	33	129	**128**	1	3.32
le450_25b	**111**	38	112	**111**	1	3.16
le450_25c	**179**	243	180	**179**	1	7.01
le450_25d	**157**	440	159	**157**	1	6.76

Table 2. Computational results for small graphs.

Both, the simulated annealing approach and ΔVh were able to find Δ colorings for all instances tested. The success rate column shows that, for all instances, in each one of the 1000 executions of the heuristic, the optimal solutions was found. The genetic algorithm was unable to find optimal solutions for nine of the 17 instances. As expected, ΔVh is at least order of magnitude faster than the simulated annealing approach (even considering that the computer used in this work is at least three times faster). The most time consuming instances were solved 1000 times by ΔVh in around 7 seconds. On the other hand, the simulated annealing heuristic took as much as 440 seconds to execute once to solve a single instance problem.

5. Conclusions

In this paper we introduced a new constructive heuristic for the edge coloring problem on simple graphs based on Vizing's theorem. Being a simple modification of Vizing's algorithm, the new heuristic guarantees that, when it is not able to find a proven optimal solution to the problem (matching the Δ lower bound), it finds a solution using at most one more color than the optimal. Experimental results showed that the new heuristic was capable of finding a Δ coloring for all benchmark instances considered. In terms of computational times, the new heuristic is significantly faster than previous approaches in the literature.

As future work, we intend to extend the proposed heuristic to consider multigraphs. In fact, Vizing's theorem states that the chromatic index of a multigraph is between Δ and $\Delta + \mu$ being μ the multiplicity of the graph. Vizing's algorithm finds $\Delta + \mu$ colorings of multigraphs. A heuristic similar to the one developed in this work may be able to find better solutions in almost the same computational time.

References

Costa, F., Urrutia, S., and Ribeiro, C. (2012). An ils heuristic for the traveling tournament problem with predefined venues. *Annals of Operations Research*, 194:137–150.

Diestel, R. (2005). *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg.

Dijkstra, E. W. and Rao, J. (1990). Constructing the proof of Vizing's Theorem. Technical Report EWD1075, University of Texas at Austin.

Enochs, B. P. and R., W. (2001). Forging optimal solutions to the edge-coloring problem. In *Proceedings of the Genetic and Evolutionay Computation Conference*, GECCO '2001, pages 313–317, San Francisco. Morgan Kaufmann Publishers.

Gabow, H. N., Nishizeki, T., Kariv, O. Leven, D., and Terada, O. (1985). Algorithms for edge-coloring graphs. Technical Report TRECIS-8501, Tohoku University.

Gould, R. (1988). *Graph Theory*. Benjamin-Cummings.

Hilgemeier, M., Drechsler, N., and Drechsler, R. (2003). Fast heuristics for the edge coloring of large graphs. In *Proceedings of the Euromicro Symposium on Digital Systems Design*, DSD '03, pages 230–238, Washington, DC, USA. IEEE Computer Society.

Holyer, I. (1981). The NP-completeness of edge-coloring. *SIAM Journal on Computing*, 10:718 – 720.

Hsu, C.-C., Liu, P., Wang, D.-w., and Wu, J.-J. (2006). Generalized edge coloring for channel assignment in wireless networks. In *Proceedings of the 2006 International Conference on Parallel Processing*, ICPP '06, pages 82–92, Washington, DC, USA. IEEE Computer Society.

Khuri, S., Walters, T., and Sugono, Y. (2000). A grouping genetic algorithm for coloring the edges of graphs. In *Proceedings of the 2000 ACM symposium on Applied computing - Volume 1*, SAC '00, pages 422–427, New York, NY, USA. ACM.

Kosowski, A. (2009). Approximating the maximum 2- and 3-edge-colorable subgraph problems. *Discrete Applied Mathematics*, 157(17):3593–3600.

Misra, J. and Gries, D. (1992). A constructive proof of vizing's theorem. *Information Processing Letters*, 41(3):131–133.

Nakano, S., Zhou, X., and Takao (1995). Edge-coloring algorithms. *Lecture Notes in Computer Science*, 1000:172 – 183.

Sanders, P. and Steurer, D. (2005). An asymptotic approximation scheme for multigraph edge coloring. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '05, pages 897–906, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Vizing, V. G. (1964). On an estimate of the chromatic class of a p-graph. *Diskrete Analiz.*, 3:25 – 30.