Combinatorial Tricks and Lovász Theta Function Applied to the Graph Coloring Problem

Igor Djukanović[1] and Franz Rendl[2]

[1] University of Maribor, EPF, Razlagova 14, 2000 Maribor, Slovenia,
igor.mat@uni-mb.si

[2] Universität Klagenfurt, Institut für Mathematik, Universitätstraße 65-67, A-9020 Klagenfurt, Austria,
franz.rendl@uni-klu.ac.at

Abstract. The semidefinite programming (SDP) formulation of the Lovász theta number $\theta(\bar{G})$ does not give only one of the best polynomial bounds on the chromatic number of a graph $\chi(G)$ but also leads to a heuristic for the graph coloring problem. Karger, Motwani and Sudan have proved the best known worst case bound for the number of the colors needed by such a heuristic. Their ideas can be employed to produce a recursive heuristic based on the Lovász theta number or its strengthenings. Those recursive heuristics can benefit from the standard combinatorial tricks for the graph coloring.

1 The problem

Graph coloring and its generalizations are well-known tools for scheduling. Consider the final exams at a university. Let $V = \{1, ..., n\}$ be the set of the final exams. If there is a student who needs to take both exams $i \neq j$, then such a *conflict* is modeled by an *undirected edge* $(i, j) = (j, i)$. Let $E = \{(ij)\}$ be the set of those edges. Then $G(V, E)$ is a *simple undirected graph.*

A k*-coloring* of the graph $G(V, E)$ is any function $c : V \to \{1, ..., k\}$ satisfying $(ij) \in E \Rightarrow c(i) \neq c(j)$. The *chromatic number* $\chi(G)$ is the smallest integer k for which a k-coloring exists. If the *colors* $\{1, ..., k\}$ stand for the different exam periods, then $\chi(G)$ is the smallest number of the time intervals in which the finals can be scheduled.

Finding a $\chi(G)$-coloring of a graph is a well-known NP-hard problem. Therefore numerous heuristics for finding suboptimal colorings exist, the most promising ones for coloring large graphs are hybrid heuristics combining a cross-over operator (genetic approach) with local search techiques [2].

In this paper we present a recursive heuristic for coloring small and medium sized graphs. On the graph coloring benchmarks from `mat.gsia.cmu.edu/COLOR03`, it performs optimaly on "MYC" and "CAR" graphs (based on Mycielsky graphs) and almost optimaly on queen graphs and random "DSJ" graphs.

2 Notation

The set of *neighbors* of a vertex $v \in V$ is $N(v) = \{w|(wv) \in E\}$, its *degree* $\Delta v = |N(v)|$ is the number of its neighbors, and $G - v$ is the graph obtained by removing the vertex v and the adjacent edges from G.

Merging two vertices i and j into one produces the *topological quotient* $G_{/\{i,j\}}$ with vertices $\{v \in V|v \neq i,j\}$ and $[i] = [j]$, the last one is adjacent to all the former neighbors of i and j.

3 Breaking the graph

If the graph is not connected, colorings of components define a coloring of the whole graph. This has two advantages. Smaller graphs can be colored much faster and also more precise.

An articulation point is such a vertex a that $G - a$ is not connected. Let $V_1, ..., V_l$ be components of $G - a$. Then finding colorings of graphs induced on $V_1 \cup a$,...$V_l \cup a$ such that $c_i(a) = 1$ defines a coloring of the whole graph. The depth-first search finds all two-connected components of a graph in a linear time[1]. Generalization of this trick is an "*articulation edge*" (and an "*articulation clique*").

However our numerical experience shows that only the rather sparse graphs break.

4 Removing vertices

It can easily be checked in a linear time weather the graph is *bipartite*. Otherwise $\chi(G) \geq 3$ is a lower bound on the chromatic number. Let $k \leq \chi(G)$ be a lower bound on the chromatic number of the graph (for example $k = \theta(\bar{G})$), and let $\Delta v < k$. Then it is enough to color $G - v$ since we can always choose a k-th color different from all the colors on the neighbors of v, and apply it on v.

Even more useful proved removing dominated vertices. *Dominant* vertex v *dominates dominated* vertex u if $(uv) \notin E$ while $N(u) \subseteq N(v)$. Obviously it is enough to color $G - u$, and apply the same color $c(v)$ on u, too.

5 Recycling the tricks

Removing a vertex v decreases the degrees of all of its neighbors. So any neighbor $u \in N(v)$ can get the degree $\Delta_{G-v} u < k \leq \chi(G)$ or become dominated by a third vertex w. Or the graph can break...

6 Hard decisions

Since the graph coloring is NP-hard, it is obvious that we can not color every graph only by recycling those combinatorial tricks. To decrease the size of the problem we need to decide which (at least two) vertices to color with the same color. SDP formulation (reformulation from [6]) of Lovász theta function, a well-known lower bound on $\chi(G)$[5],

$$\theta(\bar{G}) := \min\{t|\, z_{ii} = t-1\, \forall i \in V,\ z_{ij} = -1\, \forall (ij) \in E,\ Z \succeq 0\} \tag{1}$$

can be used to find such pairs [4]. Let Z_{ij} be large. Then with high probability there is an optimal coloring of G in which i and j are colored with the same color. While not surprisingly $Z_{ij} \doteq -1$ (and also $Z_{ij} < -1$) should be interpreted as: (probably) there is an optimal coloring of G which colors the vertices i and j with two different colors.

So a natural recursive heuristic for the graph coloring finds the biggest element $Z_{ij} > -1$ in an (almost) optimal solution Z of (1), merges the two vertices, and recursively finds a coloring c' of $G_{/\{i,j\}}$. A coloring of the graph G can be defined by $c(i) = c(j) := c'([i])$ while other vertices keep their colors.

The recursive calls become faster and faster as the graph coloring problem is much easier on the smaller graphs. They stop when the graph becomes a clique.

This heuristic, of course, needs not produce an optimal coloring. The motivation for recursive algorithm is that (on almost every graph) some of the above interpretations of the elements of the matrix Z are inconsistent. So it seems safer to employ only the most sound piece of the information.

7 Simple cutting planes

The SDP lower bound on $\chi(G)$ (1) can be tightened with the simple inequalities $z_{ij} \geq -1$ $(ij) \notin E$.[6] A tight inequality should be interpreted as a "soft edge" - as do not color the vertices i and j with the same color ($z_{ij} = -1\, \forall (ij) \in E$).

Vertex i *softly dominates vertex* j, if $(ij) \notin E$ and all neighbors of i are either neighbors or soft neighbors of j. Removing all softly dominated vertices on each iteration of this heuristic is a natural generalization of the dominated-vertex trick.

8 Triangle inequalities

This SDP formulation can be further tightened with the triangle inequalities [6]

$$z_{ij} + z_{ik} - z_{jk} \leq t - 1 \tag{2}$$

which prevent z_{ij} and z_{ik} to be large while z_{jk} is small, thus preventing inconsistent interpretation to color both vertices of edges (ij) and (ik) with the same color while coloring vertices in the edge (j,k) with different colors.
Let's define

$$\begin{array}{lll} \theta^{\triangle}(G) := \min t & & \\ & z_{ii} = t-1 & i \in V \\ & z_{ij} = -1 & (ij) \in E \\ & z_{ij} \geq -1 & i,j \in V \\ & z_{ij} + z_{jk} - z_{ik} \leq t-1 & \forall i,j,k \in V \\ & Z \succeq 0. & \end{array} \tag{3}$$

Since $\theta(\bar{G}) \leq \theta^{\triangle}(G) \leq \chi(G)$ we expect the optimal matrix variable Z_3 for (3) to produce a better coloring then Z_1, the optimal matrix variable for (1). Also the tight triangle inequalities (2) in the optimal solution of (3) have an obvious interpretation: there are two vertices in a "triangle " $i, j, k \in V$ which should be colored with the same color. Much more useful information is obtained from a tight pair

$$\begin{aligned} z_{ij} + z_{ik} - z_{jk} &\leq t - 1 \\ z_{ij} + z_{jk} - z_{ik} &\leq t - 1 \end{aligned} \tag{4}$$

which is interpreted as color the vertices i and j with the same color. Since we solve (3) with a primal-dual interior point method (IPM), it is natural to check which inequality is tighter by the value of its dual variable. The largest dual variable represents the tightest inequality (which was tight in a few iterations of an IPM, and not only by chance in the last step).

This motivates the following heuristic. Find the tightest pair of triangle inequalities (4), merge the corresponding vertices i and j, and proceed recursively.

9 Speeding up the heuristic

We have saved up to one third of the time by using warm-start initial solutions for computing (1) and (3) on the merged graphs. It seems also that applying warm-start solutions improves the coloring.

However any exact IPM applyed to our heuristics has a rather large storage requirement. Therefore to dramatically speed up the heuristic we plan to utilise the spectral bundle method. [3]

References

1. AHO, A.V., HOPCROFT, J.E., ULLMAN, J.D.: The Design and Analysis of Computer Algorithms. Addison-Wesley Publishing Company (1974)
2. GALLINIER, P., J.K. HAO, J.K.: Hybrid evolutionary algorithms for graph coloring. Journal of Combinatorial Optimization, **3** (1999) 379–397
3. HELMBERG, C., RENDL, F.: A spectral bundle method for semidefinite programming. SIAM Journal on Optimization, **10** (2000) 673–696
4. KARGER, D. R. MOTWANI, R.SUDAN, M.: Approximate graph coloring by semidefinite programming. Journal of the Association for Computing Machinery, **45** (1998) 246–265
5. LOVÁSZ, L.: On the Shannon capacity of a graph. IEEE Trans. Inform. Theory, **25** (1979) 1–7
6. MEURDESOIF, P.: Strengthening the Lovasz bound for graph coloring. Mathematical Programming (to appear).