A Report on
Approximate Graph Coloring by Semidefinite Programming *

Pingke Li[a], Zhe Liu[b]
[a] *Edward P. Fitts Department of Industrial and Systems Engineering*
North Carolina State University, Raleigh, NC, 27695-7906
[b] *Graduate Program in Operations Research*
North Carolina State University, Raleigh, NC, 27695-7906
pli@ncsu.edu (P. Li) *zliu@ncsu.edu (Z. Liu)*

Instructor: Kartik K. Sivaramakrishnan

Abstract

In this report, some results on semidefinite programming relaxation of graph coloring are summarized. Two algorithms on semicoloring/coloring are described in detail for 3-colorable graphs. The relaxation between semidefinite programming relaxation and the Lovász theta function is simply introduced.

Keywords: Graph coloring, semidefinite programming.

1 Introduction and Preliminaries

In graph theory, a graph coloring is an assignment of colors to certain objects in a graph subject to certain constraints. The simplest form of graph coloring, called the proper or legal vertex coloring, is to assign colors to the vertices of a graph such that no two adjacent vertices share the same color. Similarly, a proper edge coloring assigns colors to the edges such that no two incident edges share the same color, while a proper face coloring of a planar graph assigns colors to the faces or regions such that no two adjacent faces share the same color. Typically, other coloring problems can be transformed into a vertex coloring version. For example, an edge coloring of a graph is just a vertex coloring of its adjoint graph, while a face coloring of a planar graph is just a vertex coloring of its planar dual graph. When used without any further specification, a coloring of a graph is always assumed to be a proper vertex coloring. However, non-vertex coloring problems are usually stated and studied as they are to keep things in their perspective. Graph coloring problem has found a number of applications such as scheduling, compiler allocation, frequency assignment and pattern matching. See e.g. Berge [2], Briggs *et al.* [6] and Marx [18].

*This report is a part of the course project of OR791K at NCSU under the instruction of Dr. K. Sivaramakrishnan. The report is based on the paper by Karger *et al.* [15] and related papers on graph coloring.

A proper coloring of a graph using at most k colors is called a k-coloring. Clearly, finding a k-coloring of a graph is equivalent to the problem of partitioning the vertices into k or fewer independent sets. The least number of colors needed to color a graph is called its chromatic number χ. A graph is called k-colorable if there exists a k-coloring of the graph and is called k-chromatic if its chromatic number is exactly k.

Theorem 1: *Let $G = (V, E)$ be a graph. We have*
(i) $\chi(G) = 1$ if and only if G is totally disconnected;
(ii) $\chi(G) \geq 3$ if and only if G is not bipartite, or equivalently, has an odd cycle;
(iii) $\chi(G) \geq \omega(G)$, where $\omega(G)$ is the order of a largest clique in G;
(iv) $\chi(G) \leq \triangle(G) + 1$, where $\triangle(G)$ is the largest degree over all vertices;
(v) (Brook's Theorem) $\chi(G) \leq \triangle(G)$ if G is not K_n or C_{2n+1} where K_n is the complete graph with n vertices and C_{2n+1} is the odd cycle with $2n + 1$ vertices.
(vi) $\chi(G) \leq 4$ for any planar graph G.

Given a k-colorable graph $G = (V, E)$, finding a k-coloring for G is solvable in polynomial time for $k = 2$ but NP-hard for $k \geq 3$. The decision version of graph covering problem, i.e., whether or not is there a k-coloring, is one of Karp's 21 NP-complete problems. As shown by Garey *et al.* [9], it remains to be NP-complete even on planar graphs with node degree at most 4 although it is trivial for $k \geq 4$ on planar graphs due to the four-color theorem. However, it usually suffices to find an approximately optimum graph coloring in many applications that can be formulated as graph coloring problems. Johnson [12] showed that a version of the greedy algorithm gives an $O(n/\log n)$-approximation algorithm for k-colorings where n is the number of vertices in the graph. Wigderson [21] developed an algorithm to find a $O(\sqrt{n})$-coloring for any 3-colorable graph in polynomial time, which can be extended to find $O(n^{1-1/(k-1)})$-colorings for general k. Blum [3] provided a combinatorial algorithm for coloring a 3-colorable graph using $O(n^{3/8} \log^{8/5} n)$ colors, which can be generalized to color a k-colorable graph with $O(n^{1-1/(k-4/3)} \log^{8/5} n)$ colors. Karger *et al.* [15] presented a semidefinite programming based $O(n^{1/4} \log^{1/2} n)$-coloring for 3-colorable graphs and $O(n^{1-3/(k+1)} \log^{1/2} n)$-coloring for k-colorable graphs. Blum and Karger [4] combined the techniques in Blum [3] and Karger *et al.* [15] and improved the bound to $\tilde{O}(n^{3/14})$ for 3-colorable graphs where the notation $\tilde{O}$ is used to hide lower-order multiplicative terms, such as $\log n$. So far, the best known approximation for 3-colorable graphs is due to Arora *et al.* [1], which is based on semidefinite programming and the triangular inequality.

On the hardness of the approximation, Khanna *et al.* [14] indicated that coloring a 3-colorable graph with 4 colors is NP-hard. Generally, Lund and Yannakakis [17] showed that there exists a small number constant $\epsilon > 0$ such that no polynomial time algorithm can approximate the chromatic number of a graph to within a ratio of n^ϵ unless $P = NP$. Feige and Kilian [8] and

Håstad [11] showed that approximating the chromatic number to within $n^{1-\delta}$ for any $\delta > 0$ would imply $NP = RP$ where RP is the class of probabilistic polynomial time algorithms making one-sided error.

2 Semidefinite Programming Relaxation

The semidefinite programming relaxation for graph coloring assigns a unit vector to each vertex of a graph $G = (V, E)$ such that certain separation properties are satisfied for vectors corresponding to each pair of adjacent vertices. It is known that for any integer $k \leq n+1$ there exist k unit vectors in R^n such that their pairwise inner products are $-1/(k-1)$. Moreover, given a k-coloring of the graph $G = (V, E)$, we can assign one of these k vectors to each color class such that $\langle v_i, v_j \rangle = -1/(k-1)$, $\forall\ (i,j) \in E$ where v_i is the vector assigned to the vertex $i \in V$. See Lemma 4.1 in Karger *et al.* [15] for details. These facts lead to the following vector optimization problem:

$$(P1) \qquad \begin{array}{ll} \min\ t & \\ \text{subject to :} & \\ \quad \langle v_i, v_j \rangle \leq t & \forall\ (i,j) \in E, \\ \quad \langle v_i, v_i \rangle = 1, & \forall\ i \in V, \\ \quad v_i \in R^n, & \forall\ i \in V. \end{array}$$

Lemma 1: Let t^* be the optimal value of $(P1)$.
(i) If G is k-colorable, then $t^* \leq \dfrac{-1}{k-1}$.
(ii) If G contains a k-clique, then $t^* \geq \dfrac{-1}{k-1}$.

Remark: If $t^* = \dfrac{-1}{k-1}$, the function $\theta(G) = 1 - \dfrac{1}{t^*} = k^*$. In this case, the graph G is called vector k-colorable and the corresponding optimal solution is called a vector k-coloring for G. The value of k^* is called the vector chromatic number of G. As mentioned by Szegedy [20], k^* is exactly $\vartheta_{1/2}(\bar{G})$ where $\bar{G}$ is the complement of G and $\vartheta_{1/2}$ is the variant of the Lovász ϑ-function introduced by Schrijver [19]. The gap between the vector chromatic number and the true chromatic number was demonstrated in Karger *et al.* [15] by constructing Kneser graphs with vector chromatic number 3 and chromatic number n^{ϵ}.

The problem $(P1)$ is essentially equivalent to the semidefinite programming below:

$$(P2) \qquad \begin{array}{ll} \min\ t & \\ \text{subject to :} & \\ \quad A_{ij} \bullet X \leq t & \forall\ (i,j) \in E, \\ \quad A_{ii} \bullet X = 1, & \forall\ i \in V, \\ \quad X \in S^n_+. & \end{array}$$

where $A_{ii} = e_i e_i^T$, $A_{ij} = e_i e_j^T$ and e_i is the unit vector with its ith element equal to 1. Clearly, a solution to $(P1)$ can be obtained via the Cholesky Decomposition of any solution to $(P2)$.

Remark: As in Karger *et al.* [15], if the optimal value of $(P2)$ is $t^* = \frac{-1}{k-1}$, the graph G is called matrix k-colorable and the corresponding optimal solution is called a matrix k-coloring for G.

3 Rounding Algorithms

After solving the semidefinite programming relaxation for the graph coloring problem, we need a way to round the solution into a valid coloring for the graph. It turns out that it is too much to expect an algorithm that colors the whole graph properly with high probability. Instead, we aim for an algorithm that colors the graph almost properly. This is exactly the motivation of the semicolorings of a graph presented in Karger *et al.* [15]

Definition 1: *A k-semicoloring of a graph $G = (V, E)$ is an assignment of k colors to at least half of its vertices such that no two adjacent vertices share the same color.*

It is clear that at least $n/2$ vertices are properly colored in any k-semicoloring of the graph. Therefore, if we can semicolor a graph with k colors, we can color the graph with $k \log n$ colors where $n = |V|$. The procedure is straightforward since we can semicolor the remaining half vertices of the graph with k new colors at each iteration. This will take at most $\log n$ iterations to color the whole graph properly. Based on this result, two algorithms were developed by Karger *et al.* [15] for transforming vector colorings into semicolorings.

3.1 Rounding via Hyperplane Partitions

In this subsection, we focus on vector 3-colorable graphs and outline a randomized rounding scheme for transforming a vector 3-coloring of G into an $O(\triangle^{\log_3 2})$-semicoloring and thus into an $O(\triangle^{\log_3 2} \log n)$-coloring of G. It can yield an $O(n^{0.386})$-coloring of G by combining with the techniques used by Wigderson [21]. This method is based on that of Goemans and Williamson [10] and is weaker than the method described in the next subsection. However, it introduces several ideas which will be used in the more powerful algorithm.

Definition 2: *A hyperplane H is said to separate two vectors if they do not lie on the same side of the hyperplane. For any edge $(i, j) \in E$, the hyperplane is said to cut the edge if it separates the vectors v_i and v_j associated with the vertices i and j in a vector coloring of G.*

Lemma 2: (*Goemans-Williamson [10]) Given two vectors at an angle of θ, the probability that they are separated by a random hyperplane is exactly θ/π.*

This lemma indicates a good semicoloring algorithm for the graph G. Note that with a given vector 3-coloring $\{v_i\}_{i=1}^n$ for a 3-colorable graph G, we have $\langle v_i, v_j \rangle \leq -1/2$, $\forall\ (i, j) \in E$ which implies that the angle between v_i and v_j is at least $2\pi/3$. Therefore, the following algorithm can be applied to obtain a semicoloring for G.

Algorithm KMS-1

Step 1: Solve the semidefinite programming relaxation and obtain a vector 3-coloring $\{v_i\}_{i=1}^n$.

Step 2: Let $p = 2 + \log_3 \triangle(G)$. Create random vectors $r_1, \cdots, r_p$.

Let

$$\begin{array}{rcl} R_1 & = & \{\ i:\ \langle r_1, v_i\rangle \geq 0,\ \langle r_2, v_i\rangle \geq 0, \cdots,\ \langle r_p, v_i\rangle \geq 0\ \} \\ R_2 & = & \{\ i:\ \langle r_1, v_i\rangle < 0,\ \langle r_2, v_i\rangle \geq 0, \cdots,\ \langle r_p, v_i\rangle \geq 0\ \} \\ \vdots & & \vdots \\ R_{2^p} & = & \{\ i:\ \langle r_1, v_i\rangle < 0,\ \langle r_2, v_i\rangle < 0, \cdots,\ \langle r_p, v_i\rangle < 0\ \} \end{array}$$

Step 3: Assign color i to vertices in R_i.

This algorithm creates p random hyperplanes and assigns a particular color to the vectors lying in each of the 2^p regions created by the intersecting hyperplanes.

Theorem 2: *(Karger* et al. *[15]) The algorithm KMS-1 provides an* $O(\triangle^{\log_3 2})$*-semicoloring with probability* 1/2.

Proof: Since $2^p = 4 \times 2^{\log_3 \triangle} = 4\triangle^{\log_3 2}$, we have

$$\begin{aligned} & Pr\{i,\ j \text{ get the same color and } (i,j) \in E\} \\ & \quad = \left(1 - \frac{1}{\pi} \arccos\left(\langle v_i, v_j\rangle\right)\right)^p \\ & \quad \leq \left(1 - \frac{1}{\pi} \arccos\left(-\frac{1}{2}\right)\right)^p \\ & \quad = \left(1 - \frac{1}{\pi}\frac{2\pi}{3}\right)^p \\ & \quad \leq \frac{1}{9\triangle} \end{aligned}$$

Denote $m = |E|$ and thus $m \leq n\triangle/2$. We have

$$E\{\text{number of uncuted edges }\} \leq \frac{m}{9\triangle} \leq \frac{\triangle n/2}{9\triangle} \leq \frac{n}{8}$$

and therefore

$$Pr\{\text{ more than } \frac{n}{4} \text{ uncutted edges}\} \leq \frac{1}{2}$$

If we repeat the algorithm q times, we will find an $O(\triangle^{\log_3 2})$-semicoloring with probability at least $1 - 1/2^q$.

Note that $\log_3 2 < 0.631$ and $\triangle < n$. The algorithm provides an $\tilde{O}(n^{0.631})$-coloring for the graph G which is worse than Widgerson's algorithm [21]. However, it can be improved by the following idea due to Widgerson [21].

Algorithm KMS-1'

Step 1: Fix a threshold value δ.

Step 2: Pick any vertex i such that $deg(i) \geq \delta$.

Step 3: Color the vertex i and 2-color the neighborhoods of the vertex i with 2 new colors.

Step 4: Remove the colored vertices.

Step 5: Go to Step 2 until the maximum vertex degree is below δ at the cost using at most $2n/\delta$.

Step 6: Apply KMS-1 to color the remaining graph with $O(\delta^{\log_3 2})$ colors.

By this algorithm, we can obtain a semi-coloring using $O(n/\delta + \delta^{0.631})$ colors. The optimum choice of δ is around $n^{0.613}$ which implies an $O^{0.387}$-semicoloring algorithm and hence an $\tilde{O}(n^{0.387})$-coloring algorithm. This algorithm is still worse than Blum's algorithm [3] which provides the guarantee of an $\tilde{O}(n^{0.375})$-coloring.

3.2 Rounding via Vector Projections

In this subsection, a more powerful version of Theorem 2 was provided for general vector k-colorable graphs.

Theorem 3: *For every integer function $k = k(n)$, a vector k-colorable graph with maximum degree $\triangle$ can be semicolored with at most $O(\triangle^{1-2/k}\sqrt{\ln \triangle})$ colors in probabilistic polynomial time.*

For the case of 3-colorable graphs, we have the following algorithm to semicolor the graph.

Algorithm KMS-2

Step 1: Solve the semidefinite programming relaxation and obtain a vector 3-coloring $\{v_i\}_{i=1}^n$.

Step 2: Let $c = \sqrt{2(k-2)/k \cdot \ln \triangle}$ for sufficiently large $\triangle$. Set $j = 1$.

Step 3: Create a random vector r_j.

Step 4: Denote $R_j = \{i \in V \mid \langle v_i, r_j \rangle \geq c\}$.

Step 5: Assign color j to R_j and remove the vertices in R_j. Set $j = j + 1$.

Step 6: Repeat Step 3 to Step 5 until all the vertices are colored.

Since this procedure takes $\tilde{O}(\triangle^{1/3})$ iterations to color all the vertices, we have

$$Pr\{i,\ j \text{ get the same color and } (i,j) \in E\} = \tilde{O}((\triangle^{1/3})^{-3}) = \tilde{O}(\triangle^{-1}).$$

and

$$Pr\{(i,j) \in E \text{ get the same color }\} \leq \frac{1}{9\triangle}.$$

This is exactly same as that in the previous algorithm except that KMS-2 uses $\tilde{O}(\triangle^{1/3})$ colors. Similarly, if we apply Widgerson's technique [21], we can obtain an $\tilde{O}(\triangle^{1/4})$-coloring for any 3-colorable graphs. Generally, we have the following theorem:

Theorem 4: *For every integer function $k = k(n)$, any vector k-colorable graph on n vertices can be semicolored with $O(n^{1-3/(k+1)}/\sqrt{\log n})$ colors by a probabilistic polynomial time algorithm.*

4 Other Semidefinite Programming Formulations

In Karger *et al.* [15], another formulation of semidefinite programming related with graph coloring is also studied.

$$(P3) \qquad \begin{aligned} &\min\ k \\ &\text{subject to :} \\ &\quad \langle v_i, v_j \rangle = \frac{-1}{k-1} \quad && \forall\ (i,j) \in E, \\ &\quad \langle v_i, v_i \rangle = 1, && \forall\ i \in V, \\ &\quad v_i \in R^n, && \forall\ i \in V. \end{aligned}$$

A solution to (P3) is called a strictly vector k-coloring. The optimal value k^* to (P3) is called the strict vector chromatic number of the graph. It is proved that the strict vector chromatic number of G ie equal to $\vartheta(\bar{G})$. See Theorem 8.2 in Karger *et al.* [15] for details. Kleinberg and Goemans [16] constructed a family of graphs with strict vector chromatic number $2+\epsilon$ and chromatic number $n^{\delta(\epsilon)}$. Kleinberg and Goemans [16] also presented a stronger version of semidefinite programming formulation:

$$(P4) \qquad \begin{aligned} &\min\ k \\ &\text{subject to :} \\ &\quad \langle v_i, v_j \rangle = \frac{-1}{k-1} \quad && \forall\ (i,j) \in E, \\ &\quad \langle v_i, v_j \rangle \geq \frac{-1}{k-1}, && \forall\ i,j \in V, \\ &\quad \langle v_i, v_i \rangle = 1, && \forall\ i \in V, \\ &\quad v_i \in R^n, && \forall\ i \in V. \end{aligned}$$

The optimal value k^* to (P4) is called the strong vector chromatic number. It turns out that it is exactly the function $\vartheta_2(\bar{G})$ where ϑ_2 is the function introduced by Szegedy [20]. Szegedy [20] also showed that $\vartheta_2 \leq \vartheta^2$. The gap between the optimal value of (P4) and the chromatic number is related to the well known Borsuk conjecture [5]. Some recent results can be found in Kahn and Kalai [13] and Charikar [7].

References

[1] Arora, S., Chlamtac, E., and Charikar, M., New approximating guarantee for chromatic number, *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, 2006, 215-224.

[2] Burge, C., *Graphs and hypergraphs*, North-Holland, Amsterdam, 1973.

[3] Blum, A., New approximation algorithms for graph coloring, *Journal of the ACM*, 31 (1994) 470-516.

[4] Blum, A., and Karger, D., An $\tilde{O}(n^{3/14})$-coloring algorithm for 3-colorable graphs, *Information Processing Letters*, 61 (1997) 49-53.

[5] Borsuk, K., Drei Sätze über die n-dimensionale euklidische sphäre, *Fundamenta Math.*, 20 (1993), 177-190.

[6] Briggs, P., Cooper, K.D., Kennedy, K., and Torczon, L., Coloring heuristics for register allocation, *Proceedings of the SIGPLAN 89 Conference on Programming Language Design and Implementation*, 1989, 275-284.

[7] Charikar, M., On semidefinite programming relaxations for graph coloring and vertex cover, *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, 2002, 616-620.

[8] Feige, U., and Kilian, J., Zero knowledge and chromatic number, *Proceedings of the 11th Annual Conference on Computional Complexity*, 1996, 278-287.

[9] Garey, M.R., Johnson, D.S., and Stockmeyer, L., Some simplified NP-complete problems, *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, 1974, 47-63.

[10] Goemans, M.X., and Williamson, D.P., Improved approximation algorithm for maximum cut and satisfiability problems using semidefinite programming, *journal of the ACM*, 42 (1995) 1115-1145.

[11] Håstad, J., Clique is hard to approximate within $n^{1-\epsilon}$, *Proceedings of the 37th Annual IEEE Symposium on Foundation of Compute Science*, 1996, 627-636.

[12] Johnson, D.S., Worst cas behavior of graph coloring algorithms, *Proceedings of the Fifth Southeastern Conference on Combinatorics, Graph Theory and Computing*, 1974, 513-527.

[13] Kahn, J., and Kalai, G., A counterexample to Borsuk's conjecture, *Bulletin of the American Mathematical Society*, 29 (1993) 60-62.

[14] Khanna, S., Linial, N., and Safra, S., On the hardness of approximating the chromatic number, *Proceedings of the 2nd Israeli Symposium on Theory and Computing Systems*, 1992, 250-260.

[15] Karger, D., Matwani, R., and Sudan, M., Approximate graph coloring by semidefinite programming, *Journal of the ACM*, 45 (1998) 246-265.

[16] Kleinberg, J., and Goemans, M., The Lovász theta function and a semidefinite programming relaxation of vertex cover, *SIAM Journal on Discrete Mathematics*, 11 (1998) 196-204.

[17] Lund, C., and Yannakakis, M., On the hardness of approximating minimization problems, *Proceedings of the 25th ACM Symposium on Theory of Computing*, 1993, 286-293.

[18] Marx, D., Graph coloring problems and their applications in scheduling, *Periodica Polytechnica Ser. El. Eng.*, 48 (2004) 5-10.

[19] Schrijver, A., A comparison of the Delsarte and Lovász bounds, *IEEE Transactions on Information Theory*, 25 (1979), 425-429.

[20] Szegedy, M., A note on the θ number of Lovász and the generalized Delsarte bound, *Proceedings of the 35th Annual Sympositium on Foundations of Computer Science*, 1994, 36-39.

[21] Wigderson, A., Improving the performance guarantee for approximate graph coloring, *Journal of the ACM*, 30 (1983) 729-735.