社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－AL－108 (9)
2006／9／27

半定値計画法にもとづく彩色問題の発見的解法

小野孝男，柳浦睦憲，岩城正哉，平田富夫 (名大)
{takao,yagiura,iwaki,hirata}@al.cm.is.nagoya-u.ac.jp

概要

本論文では頂点彩色問題に対する新しい発見的アルゴリズムを提案する．このアルゴリズムでは，各頂点に平面上の単位ベクトルを割当てることにより頂点彩色問題をベクトル配置問題に還元する．このベクトル配置問題は半定値計画問題であるが，求める半正定値行列のランクに制約があるため，NP-困難である．このため発見的手法により可能解を見付ける．得られたベクトル配置からは簡単に実行可能な彩色が得られる．
計算機実験により，密なグラフにおいては提案するアルゴリズムがタブー探索や DSATUR よりも色数の少ない彩色を見付けることがわかった．

An SDP-Based Heuristic Approach for the Graph Coloring Problem

Takao Ono, Mutsunori Yagiura, Masaya Iwaki, Tomio Hirata (Nagoya University)

abstract

In this paper we will propose a new heuristic algorithm for the graph coloring problem. This problem is known to be NP-hard. In theory, approximation is also hard. That is, there exists a constant $\epsilon > 0$ such that we cannot find a coloring for the graph with n vertices using $O(n^\epsilon)$ times more colors than necessary in polynomial time unless P = NP. For the graphs with bounded chromatic number, several approximation algorithms have been proposed. Some of them are based on semidefinite programming (SDP) relaxation, where there exist sophisticated SDP solvers recently. However, although SDP can be solved in polynomial time, the time bound is of high order. Thus we consider restricting the feasible region of SDP relaxation. The results in this paper show that we can accelerate an SDP-based algorithm with little loss of accuracy.

1 Introduction

A vertex coloring for graph $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges is an assignment $c: V \to \mathbb{Z}$ such that $c(i) \neq c(j)$ if $(i, j) \in E$. A k-coloring is a vertex coloring whose range is $\{1, 2, \ldots, k\}$. The graph coloring problem is the problem of finding, for a given graph G, a k-coloring of G with minimum k. Such k is called the chromatic number of G and is denoted by $\chi(G)$. This problem is known to be NP-hard. Thus in the field of theory the approximation algorithms for the graph coloring problem have been studied. The approximation algorithm is the polynomial time algorithm which finds for any graph a coloring with at most α times as many colors as the optimal coloring. This α is called the approximation ratio of the algorithm. Some NP-hard problems such as the maximum cut problem and the maximum satisfiability problem, have approximation algorithms with constant approximation ratios. However, the approximation of the graph coloring problem is harder. That is, no constant approximation ratio can be achieved unless P $\neq$ NP. Moreover, there exists a constant $\epsilon > 0$ such that it is NP-hard to find a coloring using $n^\epsilon \chi(G)$ colors.

For the graphs with the bounded chromatic number, some approximation algorithms with approximation ratio $O(n^\epsilon)$ for $\epsilon < 1$ are known. For example, Wigderson [10] showed an $O(\sqrt{n})$-approximation algorithm for 3-colorable graphs. Karger, Motwani and Sudan [8] made a significant improvement on the approximation factor by proposing a Semidefinite Programming (SDP) based approach. Their algorithm approximates the chromatic number of any 3-colorable graph with n vertices in a factor of $\tilde{O}(n^{1/4})$,

where $\tilde{O}$ hides polylogarithmic factors.

Inpired by this result, we consider applying the SDP-based approach for practical use. However, we found that even with sophisticated solvers such as CSDP and fast computers, SDP relaxation requires much time and memory to solve for the graphs with hundreds of vertices. Thus we consider reducing the feasible region of the SDP relaxation. That is, we assign a planar (2-dimensional) vector for each vertex while Karger, Motwani and Sudan used full-dimensional vectors. The restricted problem is not SDP and hence cannot be solved with any SDP solvers, but we can use a heuristic approach for finding a near-optimal solution. We will propose an algorithm for converting the solution, the assignment of the planar vectors, to the feasible coloring. As a result, our algorithm yields better performance than a simple tabu search for dense graphs.

In Section 2 we first recall SDP and its application to the graph coloring problem, and then we introduce rank-two SDP relaxation for the graph coloring problem. We provide our experimental results in Section 3. The concluding remarks is in Section 4.

2 Rank-two SDP Relaxation for the Graph Coloring Problem

First we recall the semidefinite programming (SDP) and briefly review the SDP relaxation scheme for the graph coloring problem, proposed by Karger, Motwani and Sudan [8]. After that, we propose a new relaxation scheme for the graph coloring problem. Hereafter we call our scheme "rank-two SDP" in honor to Burer, Monteiro and Zhang [3], who applied the simliar approach to the Maximum Cut Problem.

2.1 Semidefinite Programming

In the following, let us denote $\mathcal{S}^{n\times n}$ the set of all $n \times n$ symmetric real matrices. We define the inner product $A \bullet B$ for two matrices $A = (a_{ij})$, $B = (b_{ij}) \in \mathcal{S}^{n\times n}$ as

$$A \bullet B = \sum_{i,j\in\{1,2,\ldots,n\}} a_{ij}b_{ij}.$$

Semidefinite Programming is the optimization problem of the form

$$\begin{aligned} \max\ & C \bullet X \\ \text{s.t.}\ & A_i \bullet X \le b_i, \\ & X \text{ is positive semidefinite.} \end{aligned} \tag{1}$$

All the matrices C, A_i and X are assumed to be in $\mathcal{S}^{n\times n}$. That is, SDP is the optimization problem with linear constraints and linear objective function of the elements of the positive semidefinite matrix.

It is known that we can solve SDP almost optimally (within additive error ϵ) in polynomial time. There are many SDP solvers, for example, SDPA [5] and CSDP [1].

2.2 SDP Relaxation Scheme

For a k-colorable graph $G = (V, E)$ with $n = |V|$ vertices, we assign an n-dimensional unit vector $\boldsymbol{v}_i$ to each vertex $i \in V$. It is easy to assign these vectors so that for each edge $(i, j) \in E$ the inequality $\boldsymbol{v}_i \cdot \boldsymbol{v}_j \le -1/(k-1)$ holds, where $\boldsymbol{v}_i \cdot \boldsymbol{v}_j$ denotes the inner product of $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$. With this observation, Karger, Motwani and Sudan [8] introduced the notion of vector coloring: A vector k-coloring of a graph G is an assignment of n-dimensional unit vector $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n \in \mathbb{R}^n$ to the vertices $1, 2, \ldots, n$, respectively, such that $\boldsymbol{v}_i \cdot \boldsymbol{v}_j \le -1/(k-1)$ for each edge $e = (i, j) \in E$. The vector chromatic number of the graph is the smallest number k admitting the vector k-coloring. It is easy to see that every k-colorable graph has a vector k-coloring. On the other hand, its inverse does not always hold. Thus the vector chromatic number is a lower bound for the chromatic number.

Karger, Motwani and Sudan proposed the following approximation algorithm for the graph coloring problem: First, assign vector $\boldsymbol{v}_i \in \mathbb{R}^n$ for each vertex $i \in V$ and solve the problem:

$$\begin{aligned} &\min\ \alpha \\ &\text{s.t. } \boldsymbol{v}_i \cdot \boldsymbol{v}_j \le \alpha \quad \forall e = (i,j) \in E, \\ &\qquad \|\boldsymbol{v}_i\| = 1 \qquad \forall i \in V. \end{aligned} \tag{2}$$

This problem is in fact an SDP. To see this, consider the matrix $W = (w_{ij}) \in \mathcal{S}^{(n+1)\times(n+1)}$ with $w_{ij} = \boldsymbol{v}_i \cdot \boldsymbol{v}_j$ and $w_{i,n+1} = w_{n+1,i} = 0$ for all i, $j \in \{1,\ldots,n\}$, $w_{n+1,n+1} = \alpha$. Then the problem (2) has the linear constraints and linear objective function of the elements of W, which is positive semidefinite.

For a k colorable graph G with maximum degree Δ, we can obtain an optimal solution to (2) with $\alpha = -1/(k-1)$. Now we choose $O(\Delta^{1-2/k})$ random vectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots$ in $\mathbb{R}^n$. For each vertex i, we assign the vector $\boldsymbol{u}_t$ with the largest inner product with $\boldsymbol{v}_i$. With high probability there exist at most $|E|/4$ edges whose two end vertices receive the same vector $\boldsymbol{u}_t$. Now we consider that each vector $\boldsymbol{u}_t$ corresponds to a color. Then with high probability a constant fraction of the vertices are properly colored, that is, having no adjacent vertices with the same color. Fix the colors of such vertices and repeat the process to the remaining vertices until every vertex has its own color. This algorithm uses $\tilde{O}(\Delta^{1-2/k})$ colors for any k-colorable graph.

2.3 A New Heuristic Approach Based on SDP

SDP-based algorithm in Section 2.2 is good in theory. However, the time bound to solve an SDP is of high order and thus we cannot apply this algorithm for large graphs. For example, CSDP [1] uses more than $O(n^5)$ time and about $O(n^4)$ space to solve an SDP relaxation for the graph with n vertices. This motivated us to derive a new approach based on SDP.

In our approach, we use two-dimensional (planar) unit vectors instead of n-dimensional vectors. With this modification, the SDP relaxation (2) becomes:

$$\begin{aligned} &\min\ \alpha \\ &\text{s.t. } \boldsymbol{v}_i \cdot \boldsymbol{v}_j \le \alpha \qquad \forall (i,j) \in E, \\ &\qquad \|\boldsymbol{v}_i\| = 1 \text{ and } \boldsymbol{v}_i \in \mathbb{R}^2 \quad \forall i \in V. \end{aligned} \tag{3}$$

This problem can be rewritten as follows, using the positive semidefinite matrix $W = (w_{ij}) \in \mathcal{S}^{n\times n}$, where $w_{ij} = \boldsymbol{v}_i \cdot \boldsymbol{v}_j$:

$$\begin{aligned} &\min\ \alpha \\ &\text{s.t. } w_{ij} \le \alpha \qquad \forall (i,j) \in E, \\ &\quad w_{ii} = 1 \qquad \forall i \in V, \\ &\quad W \in \mathcal{S}^{n\times n} \text{ is p.s.d.}, \\ &\quad \operatorname{rank} W \le 2. \end{aligned}$$

We call our approach "rank-two SDP relaxation" in honor to Burer, Monteiro and Zhang [3], because our approach is similar to the one that they proposed for the maximum cut problem.

We note that the problem (3) is not convex optimization problem. In fact, although we omit the proof, this problem is NP-hard. However, we can apply heuristic approaches to find a reasonably good solution to (3). It is easy to see that there is no need to use vectors in (3), because the planar unit vector $\boldsymbol{v}_i$ is characterized by its angle θ_i as $\boldsymbol{v}_i = (\cos\theta_i, \sin\theta_i)$. The problem (3) can now be rewritten as follows:

$$\begin{aligned} &\max\ \beta \\ &\text{s.t. } \operatorname{diff}(\theta_i,\theta_j) \ge \beta \quad \forall (i,j) \in E, \\ &\qquad 0 \le \theta_i < 2\pi \qquad \forall i \in V, \end{aligned} \tag{4}$$

where $\operatorname{diff}(\theta_i,\theta_j) = \min\{|\theta_i - \theta_j|, 2\pi - |\theta_i - \theta_j|\}$ denotes the difference (between 0 and π) of the angles θ_i and θ_j.

2.4 Rounding Methods for Rank-Two SDP Solution to Coloring

In this section we propose two rounding methods for a solution $(\beta, \theta_1, \ldots, \theta_n)$ to (4).

The first one is quite simple: We know that if $\theta_i \sim \theta_j < \beta$ then two vertices i and j can have the

same color. Thus we can easily find a k-coloring of G, where $k = \lceil 2\pi/\beta \rceil$: Divide the unit circle into k sectors $S_1, \ldots, S_k$ with the same center angle $2\pi/k$, $S_i = \{\theta | 2\pi(i-1)/k \leq \theta < 2\pi i/k\}$ $(i = 1, 2, \ldots, k)$. We associate the color i to the sector S_i. That is, for each vertex $i \in V$ with the corresponding angle $\theta_i \in S_t$, we assign the color t of S_t. This gives the correct coloring, because for two adjacent vertices i and j the corresponding angles θ_i and θ_j are apart at least $2\pi/k$ from each other. Thus these two vectors are in different sectors, meaning that vertices i and j receive different colors.

With this method, it can be the case that we use more colors than necessary, due to numerical error. For example, if the graph G is 3-partite, then we can take $\beta = 2\pi/3$. However, in the solution to (4) β can be $2\pi/3 - \epsilon$ for a small ϵ. In this case we take $k = 4$ and find 4-coloring of G, in spite of having almost 3-coloring. We will overcome this by proposing another rounding method.

The second rounding method comes from the following observation: What we want to find is the division of unit circle into k sectors $S_1, \ldots, S_k$ such that no two adjacent vertices i and j correspond to the angle θ_i and θ_j lying in the same sector.

We further convert this problem as follows: Consider for two adjacent vertices i and j two counterclockwise circular arcs $a_{ij} = (\theta_i, \theta_j)$ and $a_{ji} = (\theta_j, \theta_i)$ on the unit circle. We assume that these arcs does not include their end points. Let $\mathcal{S}$ be the collection of these arcs for all pairs of adjacent vertices. Now what we want to find is the set $\Phi = \{\phi_1, \ldots, \phi_k\}$ of angles such that every circular arc in $\mathcal{S}$ includes at least one angle in the set Φ. Such set of angles is called a piercing set of $\mathcal{S}$, and finding a smallest piercing set of $\mathcal{S}$ is the minimum piercing set problem for arcs. Katz, Nielsen and Segal [9] proposed an algorithm to solve the minimum piercing set problem for the set of n arcs optimally in time $O(n \log n)$. Thus we can use the following method to find an optimal coloring according to the solution $(\beta, \theta_1, \ldots, \theta_n)$:

1. Construct the set of arcs $\mathcal{S}$.
2. Solve the Minimum Piercing Set Problem for $\mathcal{S}$. Let the optimal piercing set be $\Phi = \{\phi_1, \ldots, \phi_k\}$, where ϕ_i is listed in the counterclockwise order.
3. For each vertex $i \in V$ assign color t (less than k) if $\phi_t < \theta_i < \phi_{t+1}$, and assign color k if $0 \leq \theta_i < \phi_1$ or $\phi_k < \theta_i < 2\pi$.

No two adjacent vertices have the same color because we use the piercing set. We can run this method in time $O(m \log m)$ because step two dominates the running time and $|\mathcal{S}| = 2m$ implies the time $O(m \log m)$ in that step.

We note that the running time can be reduced in two reasons. First, the set $\mathcal{S}$ has unnecessary arcs. If arc a_{ij} completely contains another arc a_{ik}, then any point in a_{ik} is also in a_{ij} and thus a_{ij} is redundant. This means that no more than two arcs are necessary for each vertex. For vertex i, we only need to consider arcs a_{ij} with the nearest angle θ_j to θ_i in both clockwise and counterclockwise order. Now the number $|\mathcal{S}|$ of the member in $\mathcal{S}$ is $2n$. Second, we can use faster algorithm (in constant factor) to solve the Minimum Piercing Set Problem. In our case, the circular-arc graph $G_{\mathcal{S}}$ of $\mathcal{S}$ cannot be the complete graph. In this case a clique cover of $C_{\mathcal{S}}$ with s cliques provides a piercing set of $\mathcal{S}$ with s points, with the point in the piercing set corresponding to the endpoint of each clique. Thus we can run on the circular-arc graph $G_{\mathcal{S}}$ the minimum clique cover algorithm [7], which runs in linear time after sorting the endpoints of all the arcs in $\mathcal{S}$.

3 Experimental Results

We implement the rank-two SDP relaxation in the following way. For each vertex i we assign angle θ_i. We repeat the following process: For $i = 1, 2, \ldots, n$, we change angle θ_i to θ which maximizes the function

$$f_i(\theta) = \min_{j:\ \text{adjacent to } i} \text{diff}(\theta, \theta_j).$$

The algorithm outputs the best coloring found. We call this algorithm *2SDP*.

We made three experiments to evaluate the performance of 2SDP. In Experiment 1, we run 2SDP and the ordinary (full rank) SDP relaxation on the small graphs in the Second DIMACS Challenge. The results of the ordinary SDP relaxation works as lower bounds of the number of colors for the graphs.

In Experiment 2, we compare the results of 2SDP with other heuristics [6, 2, 4]. In this experiments we used the graphs with more vertices than the graphs in Experiment 1. We note that we cannot run CSDP on the SDP relaxation for these graphs because our PC does not have enough memory.

In Experiment 3, we used random k-partite graphs with edge density 5 %, 10 %, ..., 95 %, for $k = 5$ and 9. We run 2SDP and a tabu search on these graphs, and compare the number of used colors.

3.1 Experiment 1: Comparison Between 2SDP and Full-Rank SDP

Table 1 shows the comparison between the ordinary SDP relxation with our 2SDP relxation by running on the graph coloring instances given in the second DIMACS challenge. 2SDP relxation is run on a PC with a 1GHz Pentium 3 and 6GB of memory, while CSDP is run on a PC with a 3GHz Xeon and 8GB of memory. The Xeon PC has roughly four times faster than the Pentium 3 PC. The data in row "2SDP" are the numbers of colors used in the obtained coloring. The data in row "CSDP" are the vector chromatic numbers of the graphs, that is, the values $-1/\alpha + 1$ for the optimal value α of the relaxation problem (2). 2SDP provides an optimal solution for 7 graphs out of 11. Except for queen6_6, 2SDP finds a coloring with at most 7 % more colors than the lower bound of vector chromatic number. For one exception, queen6_6, 2SDP finds a coloring with 50 % more colors than the vector chromatic number.

3.2 Experiment 2: Comparison to Other Heuristics

Next, we examined the performance of 2SDP relaxation for larger graphs. Because CSDP requires $O(n^4)$ memory space, we cannot use the original SDP relaxation for large graphs. Thus we compare the performance of 2SDP relaxation to other heuristic algorithms, namely, a tabu search proposed by Hertz and de Werra [6], DSATUR [2], and Iterated Greedy (IG for short) [4].

The result is in Table 2. The data in columns 2SDP, HWTabu, and DSATUR are the minimum and maximum number of colors used in the colorings obtained from ten independent runs for each instances[1]. The data in column IG have been reported in Culberson and Luo [4]. The table shows that 2SDP has roughly the same performance as HWTabu and DSATUR.

3.3 Experiment 3: For Random k-partite Graphs

Finally, we compare the performance of 2SDP with those of HWTabu and DSATUR on random k-partite graphs for $k = 5$ and 9. In these graphs, each partition has 100 vertices and two vertices i and j in the different partition are adjacent with fixed probability p. Moreover, we select one vertex from each partition to form a clique, which ensures that the chromatic number of the graph is exactly k. For each set of parameter (k, p) we used 10 random graphs and for each graph we ran each of three algorithms 10 times.

Figures 1 and 2 show the average number of colors used in the obtained coloring. In both cases, 2SDP has almost the best performance for dense graphs with at least 30 % density. Contrary to the intuition, our experiment in general shows that the denser the graph is, the better the performance of all algorithms is. We can explain this result as follows: in a dense

[1]The program codes of HWTabu and DSATUR are obtained from Culberson's Web site. The URL is http://www.cs.ualberta.ca/~joe/Coloring/Colorsrc/color.tar.gz.

表 1: Results on the DIMACS challenge instances

Instance	$\|V\|$	$\|E\|$	2SDP	CSDP (lower bound)
miles250	128	387	8 colors/0.07 s	8 colors/979.21 s
miles500	128	1170	21 colors/0.10 s	20 colors/1094.70 s
miles750	128	2113	33 colors/0.12 s	31 colors/1388.32 s
miles1000	128	3216	45 colors/0.32 s	42 colors/1282.72 s
miles1500	128	5198	73 colors/0.25 s	73 colors/1371.49 s
mulsol.i.1	197	3925	49 colors/0.29 s	49 colors/19414.61 s
mulsol.i.2	188	3885	31 colors/0.25 s	31 colors/18976.04 s
mulsol.i.3	184	3916	31 colors/0.24 s	31 colors/15811.54 s
mulsol.i.4	185	3946	31 colors/0.24 s	31 colors/16180.90 s
mulsol.i.5	186	3973	31 colors/0.25 s	31 colors/16499.77 s
queen6_6	36	290	9 colors/0.03 s	6.028 colors/1.19 s

表 2: Results of 2SDP compared to those of other heuristic algorithms

Instance	$\|V\|$	$\|E\|$	2SDP	HWTabu [6]	DSATUR [2]	IG [4]
DSJC250.5	250	31336	39–46	41–43	37–39	32–34
DSJC500.5	500	125248	70–80	71–74	65–68	57–60
DSJC1000.5	1000	499652	122–136	125–129	114–119	102–106
flat300_20_0	300	21375	43–52	45–48	41–43	20–22
flat300_26_0	300	21633	43–52	45–48	41–43	36–38
flat300_28_0	300	21695	43–51	45–48	41–45	35–38
flat1000_50_0	1000	245000	121–133	124–127	114–117	50–104
flat1000_60_0	1000	245830	120–133	122–126	114–117	100–105
latin_square_10	900	307350	137–156	146–156	126–134	
le450_15a	450	8168	21–25	17–20	16–18	17–18
le450_15b	450	8169	21–25	17–20	16–17	17–18
le450_15c	450	16680	31–36	29–30	24–25	25–26
le450_15d	450	16750	31–36	28–31	24–26	25–26
mulsol.i.1	197	3925	49–49	49–49	49–49	49–49
school1	385	19095	29–45	14–15	14–29	14–14
school1_nsh	352	14612	34–45	14–15	14–25	14–15

graph, each vertex is connected to many vertices in every partition. This means that it is easy to find which partition the vertex belongs to. On the other hand, it is hard to find the hidden structure in sparse graphs.

4 Conclusion

In this paper we proposed a new heuristic algorithm, rank-two SDP, for the graph coloring problem. This is inspired from the SDP-based approximation algorithm of Karger, Motwani and Sudan. The change is that, rank-two SDP maps each vertex to a two-dimensional vector instead of n-dimensional vector. Although we cannot apply standard SDP solvers to this restricted SDP, we can always convert its solution to the legal coloring in linear time.

From our experiment, rank-two SDP solves the small DIMACS instances almost optimally. For larger instances, rank-two SDP has almost the same performance as other simple heuristic algorithms such as the tabu search of Hertz and de Werra, and DSATUR. The last experiment showed that rank-two SDP works well for the random k-partite graphs with edge density more than 30 %. The challenge is to improve the search process for finding a better solution to the SDP.

参考文献

[1] Brian Borchers. CSDP, A C library for semidefinite programming. *Optimization Methods and Software*, 11(1):613–623, 1999.

[2] D. Brélaz. New methods to color vertices of a graph. *Communications of the ACM*, 22:251–256, 1979.

[3] Samuel Burer, Renato D.C. Monteiro, and Yin Zhang. Rank-two relaxation heuristics for max-cut and other binary quadratic programs. TR00-33, Department of Computational and Applied Mathematics, Rice University, 2000.

[4] Joseph C. Culberson and Feng Luo. Exploring the k-colorable landscape with iterated greedy. In David S. Johnson and Michael A. Trick, editors, *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 245–284. American Mathematical Society, 1993.

[5] Katsuki Fujisawa, Masakazu Kojima, Kazuhide Nakata, and Makoto Yamashita. SDPA (semidefinite programming algorithm) user's manual — version 6.2.0. Research Report B-308, Dept. Math. & Comp. Sciences, Tokyo Institute of Technology, December 1995, Revised September 2004.

[6] A. Hertz and D. de Werra. Using tabu search techniques for graph coloring. *Computing*, 39:345–351, 1987.

[7] W.L. Hsu and K.H. Tsai. Linear time algorithms on circular-arc graphs. *Information Processing Letters*, 40:123–129, 1991.

[8] David Karger, Rajeev Motwani, and Madhu Sudan. Approximate graph coloring by semidefinite programming. *Journal of the ACM*, 45(2):246–265, 1998.

[9] Matthew J. Katz, Frank Nielsen, and Michael Segal. Maintenance of a piercing set for intervals with applications. *Algorithmica*, 36(1):59–73, February 2003.

[10] Avi Wigderson. Improving the performance guarantee for approximate graph coloring. *Journal of ACM*, 30(4):729–735, 10 1983.

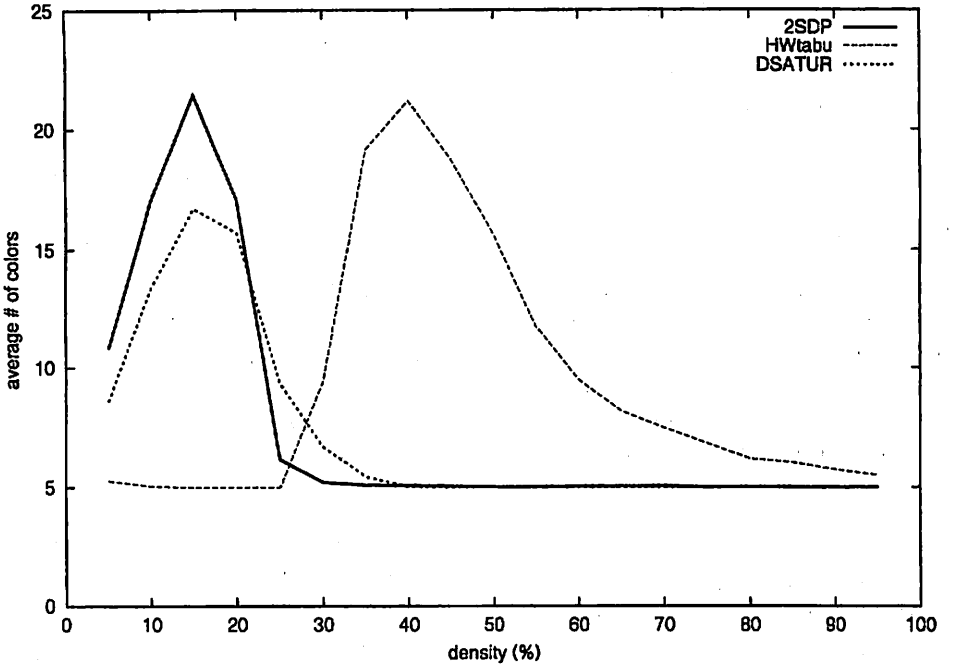

図 1: Results for 5-partite graphs

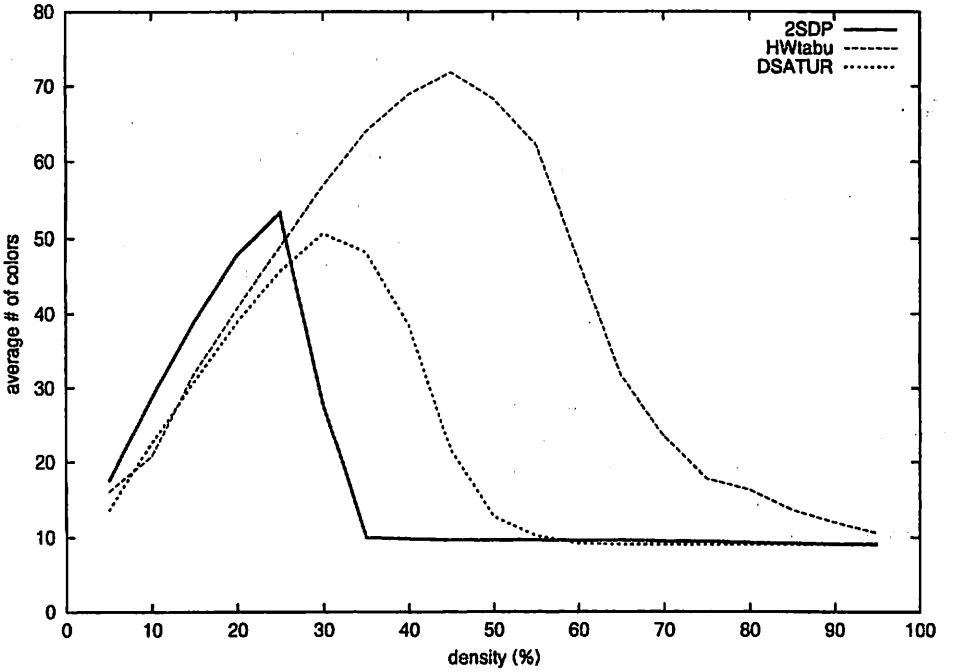

図 2: Results for 9-partite graphs