*Discussiones Mathematicae*
*Graph Theory 24 (2004) 183–195*

# NEW LOWER BOUNDS ON THE WEIGHTED CHROMATIC NUMBER OF A GRAPH

Massimiliano Caramia

*IAC – Istituto per le Applicazioni del Calcolo "M. Picone"*
*CNR – Viale del Policlinico, 137 – 00161 Roma, Italy*

**e-mail:** caramia@iac.rm.cnr.it

AND

Jiří Fiala*

*Institute of Theoretical Computer Science (ITI)*[1]
*Charles University, Faculty of Mathematics and Physics*
*Malostranské nám. 2/25, 118 00, Prague, Czech Republic*

**e-mail:** fiala@kam.mff.cuni.cz

**Abstract**

In this paper we present theoretical and algorithmic results for the computation of lower bounds on the chromatic number of a weighted graph. In particular, we study different ways of a possible improvement of the lower bound offered by a maximum weighted clique. Based on our findings we devise new algorithms and show their performance on random graphs.

**Keywords:** combinatorial analysis, computational analysis, optimization.

**2000 Mathematics Subject Classification:** 05C15, 05C85.

*Research supported in part by Czech research grants GAUK 158/99 and GAČR 201/99/0242.
[1]Supported by the Ministery of Education of the Czech Republic as project LN00A056.

## 1. Introduction

The graph weighted coloring problem generalizes the well known graph coloring problem which is *NP*-hard [5, 6], and for which a number of exact but possibly exponential-time algorithms have been presented in the literature (e.g., see [1, 2, 6, 7, 8, 9, 10]).

The weighted coloring problem asks for an assignment of sets of colors to the vertices of the given graph such that each vertex receives as many colors as is prescribed by its weight. Moreover two adjacent vertices are not allowed to share colors in the assigned sets. The objective is to find a solution with the minimum number of different colors. For the minimum *weighted* coloring problem, recently an exact algorithm appeared in the literature [3].

Coloring has received much attention in the literature as it can be used for modeling many real problems, such as scheduling, register allocation, garbage collection, frequency assignment and many others.

In the papers dealing with similar problems (especially in those related to exact algorithms), one common difficulty has emerged: the computing of a good lower bound. A good lower bound is needed to render the enumeration process more effective. On the other hand, it is also desired to reduce the gap between the lower and upper bounds to capture the distance of a given solution from the optimal one.

In particular, this necessity comes out also from the fact that the most common lower bound used for graph coloring is the size of a maximum clique. Unfortunately, this bound is associated with a negative result: Descartes [4] proved that the lower bound in terms of size of the maximum clique can be arbitrarily far from the minimum coloring.

Observe also, that for a perfect graph $G$, the weighted chromatic number of $G$, is equal to the weight of a maximum weighted clique. It means that there exists a large class of graphs that attain this lower bound.

Joining these two results, it seems that the maximum clique is a useful tool, but can be inadequate. In this paper we pose the question on a possible improvement of this lower bound. The approach used in the paper is similar to those of list coloring. We perform computation on lists of feasible colors given by a fixed coloring of the maximum weighted clique. We take into account cliques in the neighborhood of the maximum weighted one, and also all stars outside the maximum weighted clique.

In particular, we design several algorithms that are able to provide better lower bounds exploring cliques of different sizes in the neighborhood of

a maximum weighted clique. With respect to the latter case, we first give an *NP*-completeness results for the so called *star coloring extension problem*. We prove that the list coloring problem on stars is *NP*-complete. Then we show a polynomial time algorithm for the stars made up of two rays, i.e., paths of length two. We experiment with the algorithms devised and provide results on generic random graphs and on triangle free random graphs.

The paper is organized as follows: In Section 2 there are basic notations; in Section 3 there is the analysis of the lower bound divided into two subsections: Subsection 3.1, where we consider cliques to extend the lower bound offered by a maximum weighted clique, and Subsection 3.2 which describes the same approach on stars.

In Section 4 we report computational results associated with the algorithms presented in the paper.

## 2. Preliminaries

Let $G$ be a simple loopless graph, with vertex (node) set $V$ and edge set $E$.

For a set of vertices $W \subseteq V$, we define a *neighborhood* $N(W)$ as those vertices at distance one from $W$, i.e., $N(W) = \{u \in V \setminus W | \exists v \in W : (u, v) \in E\}$. For simplicity we omit braces on one-element sets, i.e., $N(u) = N(\{u\})$.

A *clique* $K$ is a set of nodes that are mutually adjacent. We also associate the word clique with the complete subgraph of $G$ induced by vertices of $K$.

For a vertex $u \in V$ we define the *star* $S_u$ as the subgraph of $G$ induced by edges incident with $u$. The neighbors of $u$ in $S_u$ are called *rays*.

A *proper k-coloring* is a function $c : V \to \{1, 2, \ldots, k\}$ such that $c(u) \neq c(v)$ for each $(u, v) \in E$, i.e., no two adjacent nodes are assigned the same color.

The smallest number $k$ for which a proper $k$-coloring of $G$ exists is called the *chromatic number* of $G$ and is denoted by $\chi(G)$.

Assume that in a graph $G$ to each node $u \in V$ is assigned a positive integer weight $w_u$. Then we call $G$ a weighted graph. For any set of nodes $W \subseteq V$ we define the weight of $W$ as the sum of weights of its elements, i.e., $w_W = \sum_{u \in W} w_u$.

A clique $K$ of weight $w_K$ is the *maximum weighted clique* of $G$ if no clique of larger weight exists in $G$.

A proper *weighted* $k$-coloring of a weighted graph $G$ is a coloring of the vertices by *sets* of colors, such that each set has as many elements as

specified by the weight of the vertex. More formally it is a mapping $C : V \to \mathcal{P}(\{1, 2, \ldots, k\})$, satisfying

- $\forall u \in V : |C(u)| = w_u$, and
- $\forall (u, v) \in E : C(u) \cap C(v) = \emptyset$.

The minimum number of colors for which there exists a proper weighted $k$-coloring is called *weighted chromatic number* of $G$ and is denoted by $\chi_w(G)$.

The classical coloring problem is equivalent to the weighted variant in the case of all weights being one.

## 3. The Analysis of the Lower Bound

Let $G$ be the given graph and assume that $K$ is its maximum weighted clique of weight $w_K$. Without loss of generality we may assume that the colors used on $K$ form an integer interval $\{1, 2, \ldots, w_K\}$. Fix a weighted $w_K$ coloring $C$ of $K$ arbitrarily.

Our method is based on the exploration of which colors are available on the neighborhood of $K$. For each vertex $u \in V \setminus K$ we define a list of feasible colors as $L_u = \{1, 2, \ldots, w_K\} \setminus \bigcup_{v \in N(u) \cap K} C(v)$.

Observe that for every vertex $|L_u| \geq w_u$, otherwise $\{u\} \cup (N(u) \cap K)$ is a clique of weight greater then $w_K$. It is also straightforward that the mapping $C$ could be extended to the entire graph $G$ if on vertices of $V \setminus K$ it uses either colors from lists $L_u$ or extra new colors above the interval $\{1, 2, \ldots, w_K\}$. Our method to estimate the lower bound on the weighted chromatic number is based on the following principle: In the subgraph of $G$ induced by $V \setminus K$ we examine two kinds of graphs, namely cliques (Section 3.1) and stars (Section 3.2) and estimate the minimum necessary number of extra colors.

### 3..1 Cliques in the neighborhood of $K$

Consider the set $\mathcal{K}$ of all cliques of $N(K)$ and any clique $K' \in \mathcal{K}$. Clearly $\chi_w(K') = w_{K'}$, and if the number of colors available by the union of list of elements of vertices of $K'$ is smaller than $w_{K'}$, we will need at least $w_{K'} - |\bigcup_{u \in K'} L_u|$ extra colors. This argument holds for every clique $K' \in \mathcal{K}$, so our first extended lower bound is expressed as the maximum over $\mathcal{K}$, i.e.,

**Theorem 1.** *For any graph $G$, its weighted chromatic number is bounded by*

$$\chi_w(G) \geq w_K + \max\left\{0, \max_{K' \in \mathcal{K}} \left( w_{K'} - \Big| \bigcup_{u \in K'} L_u \Big| \right)\right\}.$$

Observe that if we consider a clique $K'' \subseteq V \setminus K$ containing a point outside $N(K)$, then $\bigcup_{u \in K''} L_u = \{1, 2, \ldots, w_K\}$ and no extra colors are forced due to the assumption of $K$ being the maximum clique.

From the computational point of view it is hard to consider all possible cliques in graph, (there might be exponentially many such sets) so we restrict our attention to three specific tractable subsets of $\mathcal{K}$:

- $\mathcal{K}_1$ is the set of all edges in $N(K)$, providing the bound $\Lambda_1$.
- $\mathcal{K}_2$ contains all triangles in $N(K)$ with bound $\Lambda_2$.
- $\mathcal{K}_3$ is a set of cliques obtained by the the heuristic below and the corresponding bound is $\Lambda_3$.

**Algorithm**
**Input:** A weighted subgraph induced by $N(K)$
**Output:** A set of cliques $\mathcal{K}_3$ and the corresponding bound $\Lambda_3$.

**Step 1.** $\mathcal{K}_3 := \emptyset$;
$\Lambda_3 := w_K$;
order nodes $u \in N(K)$ non decreasingly by the ratio $\frac{w_u}{|L_u|}$
and number them accordingly $u_1, \ldots, u_{|N(K)|}$;

**Step 2.** for $i := 1$ to $|N(K)|$ do

**Step 2.1.** $K' := \{u_i\}$;
$L_{K'} := L_{u_i}$;

**Step 2.2.** for $j := i + 1$ to $|N(K)|$ do
if $K' \cup \{u_j\}$ is a clique in $N(K)$ then
$K' := K' \cup \{u_j\}$;
put $K'$ into $\mathcal{K}_3$;
$L_{K'} := L_{K'} \cup L_{u_j}$;
$\Lambda_3 := max\{\Lambda_3, w_K + w_{K'} - |L_{K'}|\}$;

Roughly speaking, the above algorithm orders vertices by ratio between $w_u$ and $|L_u|$ and for every vertex it greedily searches for a clique among its

successors. The order prefers vertices which might contribute by the largest portion to the difference $w_{K'} - |L_{K'}|$.

We note here that with a suitable data structure the time complexity of the algorithm is bounded by $O(|V|^3)$ and similarly the bounds $\Lambda_1$ and $\Lambda_2$ could be obtained in $O(|V|^2)$ and $O(|V|^3)$ time, respectively.

Observe also that the value $w_K + w_{K'} - |L_{K'}|$ does not necessarily form a monotone sequence as $K'$ gains new vertices, and therefore we take into $\mathcal{K}_3$ all stages of growth of the clique $K'$.

The computation of a bound $\Lambda_4$ majorizing all bounds $\Lambda_1$, $\Lambda_2$ and $\Lambda_3$ (i.e., $\mathcal{K}_4 \supseteq \mathcal{K}_1 \cup \mathcal{K}_2 \cup \mathcal{K}_3$) can be done by a common algorithm running in time $O(|V|^3|E|)$. The new algorithm differs by the beginning of Step 2, which is modified as follows:

**Step 2.** for all $(v, v') \in E(N(K))$ do

**Step 2.1.** $K' := \{v, v'\}$;
$L_{K'} := L_v \cup L_{v'}$;
$\Lambda_4 := max\{\Lambda_4, w_K + w_{K'} - |L_{K'}|\}$;
let $k$ be the higher index of $v$ an $v'$;
for $i := k + 1$ to $|N(K)|$ do
if $u_i, v, v'$ induce a triangle then
$K' := K' \cup \{u_i\}$;
$L_{K'} := L_{K'} \cup L_{u_i}$;
$\Lambda_4 := max\{\Lambda_4, w_K + w_{K'} - |L_{K'}|\}$;
etc.

Finally notice that in a triangle-free graph, all maximal cliques are edges (or isolated vertices). In this case for every edge $(u, v) \in N(K)$ we have $|L_u \cup L_v| = \{1, 2, \ldots, w_K\}$. Hence $\Lambda_1 = w_K$, even if $N(K)$ might have an interesting structure, in fact it is a bipartite graph.

### 3..2 Extending the coloring on stars

In this section we consider the set $\mathcal{S}$ of all star subgraphs $S$ in $V \setminus K$. Even if there is no simple formula for the lower bound as in the case of cliques we propose this method due to the following arguments:

- Stars in $V \setminus K$ are more easy to enumerate.
- The lower bound obtained by edge-induced subgraph is still valid also for its vertex-induced subgraph.

- The arguments for general subgraphs would be combinatorially more complicated.

As above, consider a weighted star $S$, where each vertex $u$ is assigned a list $L_u$ of feasible colors. Our question is:

Star Coloring Extension Problem: *"What is the minimum necessary number $\lambda$ of extra new colors such that after extending each list by these $\lambda$ colors the star $S$ will have a feasible weighted coloring?"*

Based on a possible answer to the Star Coloring Extension Problem we obtain the expression for the new lower bound on the weighted chromatic number.

**Theorem 2.** *The weighted chromatic number of an arbitrary graph $G$ is bounded by*

$$\chi_w(G) \geq w_K + \max_{S \in \mathcal{S}} \lambda_S.$$

We firstly bring an observation, that the possible solution of the Star Coloring Extension Problem might be of a special form. Denote the center of the star $S$ by $u$ and its rays by $R = \{a_1, a_2, \ldots, a_{\deg(u)}\}$.

Without loss of generality we may assume that lists assigned to the vertices of the star satisfy $L_u = \bigcup_{a \in R} L_a$. If not, we simplify the instance given by lists $L_u, L_a, (a \in R)$ and weights $w_u, w_a, (a \in R)$ as follows:

- Define the new list $L_u$ as $L_u$ without all colors not presented in $\bigcup_{a \in R} L_a$ and set $w_u := w_u - |L_u \setminus \bigcup_{a \in R} L_a|$. Observe that the removed colors might be used on $u$ without any restriction on coloring of any $a \in R$.
- Similarly, for each $a \in R$ remove from $L_a$ all colors not presented in $L_u$ and decrease $w_a$ by $|L_a \setminus L_u|$. As above the removed colors do not influent the coloring of $u$.

Now we define a partial order on colors of $L_u$ as follows:

$$c' \prec c \Leftrightarrow \{L_a : c' \in L_a, a \in R\} \subset \{L_a : c \in L_a, a \in R\}.$$

This partial order means that the color $c$ is more demanded than $c'$. Since the newly introduced colors are available for all rays, by definition they are more demanded than colors of $L_u$, i.e., each new color $c$ is such that $c' \preceq c$, $\forall c' \in L_u$.

The following lemma might be interesting by its own.

**Lemma 1.** *Let $S_u$ be a star, and $C$ be an optimal coloring minimizing the value $\lambda_S$. Then there exists an alternative optimal solution $C'$, satisfying:*

$$\forall a \in R, c \in C'(a), c' \in L_a \setminus C'(a)$$

*is such that $c \not\prec c'$.*

***Proof.*** Assume that in the solution $C$ for a ray $a$ such pair of colors $c$ and $c'$ satisfying $c \prec c'$ exists. If $c' \in C(u)$ we may set $C'(a) = (C(a) \setminus \{c\}) \cup \{c'\}$. Otherwise we modify $C$ as follows:

- set $C'(a_j) = (C(a_j) \setminus \{c\}) \cup \{c'\}$ for all $a_j : c \in C(a_j)$,
- and set $C'(u) = (C(u) \setminus \{c'\}) \cup \{c\}$.

The new coloring $C'$ is a feasible weighted coloring for $S_u$ that does not increase the value $\lambda_S$. We repeat this procedure until the new solution satisfies the required property. ■

Unfortunately, even the standalone STAR COLORING EXTENSION PROBLEM is computationally hard.

**Lemma 2.** *The decision version of the* STAR COLORING EXTENSION PROBLEM *is NP-complete for $\lambda = 0$.*

***Proof.*** We reduce SATISFIABILITY. Let $\Phi$ be a formula with $m$ clauses and $n$ variables. We define a weighted star $S$ with $m + n$ rays, where each vertex is assigned a list of colors, s.t. the star allows a list-weighted coloring if and only if $\Phi$ has a satisfying assignment. The set of colors used in this construction correspond to all possible literals, i.e., for each variable $x_i$, there are colors $2i - 1$ and $2i$ representing literals $x_i$ and $\neg x_i$, respectively.

The lists are assigned as follows:

- The central vertex $u$ of the star is assigned the list of all colors $L_u = \{1, 2, \ldots, 2n\}$ and $w_u = n$.
- For each variable $x_i$ there is a unique ray $u_i$ assigned $L_{u_i} = \{2i - 1, 2i\}$ and $w_{u_i} = 1$.
- For each clause $c_j$ there is a unique ray $v_j$ assigned list of colors representing the literals of $c_j$ and with weight $w_{v_j} = 1$.

If a feasible weighted coloring exists, then the colors of the variable rays determine the truth valued literals. The central vertex is then colored by

all negative valued literals and on clause rays it is possible to use only true valued literals. The opposite transformation of a truth assignment to coloring is then straightforward. ■

Observe, that the Star Coloring Extension Problem could be solved by integer linear programming approach, where the number of variables is proportional to the number of distinct inclusions of lists in the star graph $S$. On account of Lemma 2 we relax to provide the optimum solution of the Star Coloring Extension Problem for the general case, but we restrict ourselves on tractable instances. If the number of rays is bounded, we may provide an exact solution, and we would like to explain it on the simplest case of the star with two rays. Consider a star $S$ with two rays, i.e., isomorphic to a path of length two. Denote by $u$ the central vertex and by $a$ and $b$ the two rays. As discussed above we may assume without loss of generality that $L_u = L_a \cup L_b$.

Denote by $x_a, x_b$ and $x_c$ the sizes of sets $L_a \setminus L_b, L_b \setminus L_a$ and $L_a \cap L_b$, respectively. We proceed the computation of $\lambda_s$ as follows: Assign $\min\{x_c, w_a, w_b\}$ colors from $L_a \cap L_b$ to both $a$ and $b$. We distinguish two cases:

- Without loss of generality let $w_a = \min\{x_c, w_a, w_b\}$. Then $u$ could be assigned $x_a$ colors from $L_a \setminus L_b$, and $b$ will need $w_b - w_a$ more colors where only $|L_b| - w_a$ colors are available for both $u$ and $b$.
  In total the number of extra new colors is:

$$\begin{aligned} \lambda_S &= \max\{0,\ w_u - x_a + w_b - w_a - (|L_b| - w_a)\} \\ &= \max\{0,\ w_u + w_b - x_a - x_b - x_c\}. \end{aligned}$$

- If $x_c = \min\{x_c, w_a, w_b\}$, then we have $x_a + x_b$ colors at disposal and we have to saturate $w_a + w_b + w_c - 2x_c$ requests. We may use at most $\min\{w_a - x_c, w_b - x_c\}$ new colors simultaneously at $a$ and $b$, hence:

$$\begin{aligned} \lambda_S &= \max\Bigg\{0,\ x_a + x_b + 2x_c - w_a - w_b - w_c - \\ &\quad - \min\left\{\frac{x_a + x_b + 2x_c - w_a - w_b - w_c}{2},\ w_a - x_c,\ w_b - x_c\right\}\Bigg\}. \end{aligned}$$

In spirit of Theorem 2 we may apply the above procedure and compute the lower bound $\Lambda_5 = w_K + \max_{S \in \mathcal{S}_5} \lambda_S$ over the set $\mathcal{S}_5$ of all stars with two rays in $V \setminus K$.

Observe that the computation of $\lambda_S$ for a given $S$ could be done in time dependent only on computing of the sizes $x_a, x_b$ and $x_c$. Under the assumption that these times could be handled in time $t_w$, then the bound $\Lambda_5$ could be computed in $O(t_w|V|^3)$ time.

## 4. Experimental Results

In order to deeply analyze our algorithms, we have used two kinds of test problems in our data set: the first one is given by generic random graphs, the second one is given by triangle free random graphs. A weight function assigns uniformly at random to each node in a graph an integer drawn from the set $\{1, \ldots, m_w\}$, where $m_w$ is the maximum weight allowable for a node.

Table 1. $\Lambda_4$ values on random graphs

| Type | $w_K$ $m_w = 5$ | $\Lambda_4$ $m_w = 5$ | $w_K$ $m_w = 10$ | $\Lambda_4$ $m_w = 10$ | $w_K$ $m_w = 20$ | $\Lambda_4$ $m_w = 20$ |
|---|---|---|---|---|---|---|
| 30.01 | 10.2 | 10.8 | 19.8 | 20.0 | 40.4 | 41.2 |
| 30.03 | 15.4 | 16.0 | 31.0 | 32.2 | 62.8 | 65.6 |
| 30.05 | 24.6 | 25.6 | 45.0 | 47.0 | 90.8 | 95.2 |
| 30.07 | 30.2 | 31.2 | 58.0 | 60.4 | 116.8 | 122.2 |
| 30.09 | 38.6 | 40.2 | 74.0 | 77.6 | 148.8 | 156.4 |
| 50.01 | 11.2 | 11.4 | 20.8 | 21.0 | 42.4 | 43.2 |
| 50.03 | 18.4 | 19.4 | 35.4 | 37.2 | 71.6 | 75.6 |
| 50.05 | 25.8 | 28.2 | 48.2 | 51.2 | 97.2 | 103.6 |
| 50.07 | 34.8 | 37.2 | 64.8 | 67.4 | 130.4 | 136.4 |
| 50.09 | 44.6 | 46.8 | 82.6 | 85.6 | 166.0 | 172.4 |
| 70.01 | 12.4 | 12.8 | 22.8 | 23.2 | 46.4 | 47.6 |
| 70.03 | 21.2 | 22.8 | 40.6 | 42.2 | 82.0 | 85.6 |
| 70.05 | 29.6 | 31.6 | 54.8 | 57.6 | 110.4 | 116.4 |
| 70.07 | 38.8 | 40.0 | 75.4 | 78.2 | 151.6 | 157.6 |
| 70.09 | 47.0 | 49.2 | 92.4 | 95.2 | 185.6 | 191.6 |
| 80.01 | 13.2 | 13.8 | 24.4 | 25.2 | 49.6 | 51.6 |
| 80.03 | 23.4 | 26.4 | 44.0 | 47.4 | 88.8 | 96.0 |
| 80.05 | 35.6 | 37.8 | 67.2 | 70.2 | 135.2 | 141.6 |
| 80.07 | 46.8 | 50.2 | 89.2 | 93.6 | 179.2 | 188.4 |
| 80.09 | 52.0 | 56.2 | 102.8 | 107.0 | 206.4 | 215.2 |
| 90.01 | 14.6 | 15.4 | 26.2 | 27.2 | 53.2 | 55.6 |
| 90.03 | 24.8 | 27.0 | 45.2 | 48.2 | 91.2 | 97.6 |
| 90.05 | 36.0 | 39.4 | 70.8 | 74.4 | 142.4 | 150.0 |
| 90.07 | 50.2 | 53.8 | 96.4 | 100.4 | 193.6 | 202.0 |
| 90.09 | 59.2 | 63.4 | 115.8 | 120.0 | 232.4 | 241.2 |

All statistics are averages of 5 test problems.

All algorithms have been coded in the C language and run on a Pentium III PC with a 700 MHz processor and 128 MB RAM. In the columns are listed the following parameters:

- $w_K$: the weight of the maximum clique;
- $\Lambda_4$, $\Lambda_5$: the lower bound values presented in the paper.

It should be observed that the new lower bound values are always greater than the values of the corresponding maximum weighted clique. In particular, referring to $\Lambda_4$ we have an improvement ranging from 2% to 6%, and referring to $\Lambda_5$ an improvement ranging from 4% to 12%.

Table 2. $\Lambda_5$ values on random graphs

| Type | $w_K$ $m_w = 5$ | $\Lambda_5$ $m_w = 5$ | $w_K$ $m_w = 10$ | $\Lambda_5$ $m_w = 10$ | $w_K$ $m_w = 20$ | $\Lambda_5$ $m_w = 20$ |
|---|---|---|---|---|---|---|
| 30.01 | 10.2 | 11.4 | 19.8 | 20.4 | 40.4 | 42.2 |
| 30.03 | 15.4 | 17.0 | 31.0 | 32.6 | 62.8 | 66.8 |
| 30.05 | 24.6 | 26.2 | 45.0 | 47.6 | 90.8 | 98.0 |
| 30.07 | 30.2 | 32.0 | 58.0 | 60.8 | 116.8 | 124.2 |
| 30.09 | 38.6 | 40.6 | 74.0 | 78.2 | 148.8 | 158.6 |
| 50.01 | 11.2 | 12.4 | 20.8 | 22.2 | 42.4 | 45.4 |
| 50.03 | 18.4 | 20.2 | 35.4 | 38.6 | 71.6 | 77.4 |
| 50.05 | 25.8 | 29.0 | 48.2 | 52.8 | 97.2 | 105.8 |
| 50.07 | 34.8 | 39.4 | 64.8 | 69.2 | 130.4 | 139.4 |
| 50.09 | 44.6 | 48.6 | 82.6 | 90.8 | 166.0 | 175.6 |
| 70.01 | 12.4 | 14.0 | 22.8 | 25.4 | 46.4 | 49.6 |
| 70.03 | 21.2 | 24.8 | 40.6 | 44.6 | 82.0 | 82.8 |
| 70.05 | 29.6 | 34.8 | 54.8 | 59.8 | 110.4 | 118.8 |
| 70.07 | 38.8 | 44.6 | 75.4 | 80.6 | 151.6 | 159.0 |
| 70.09 | 47.0 | 52.0 | 92.4 | 99.8 | 185.6 | 194.8 |
| 80.01 | 13.2 | 15.8 | 24.4 | 27.2 | 49.6 | 53.8 |
| 80.03 | 23.4 | 29.4 | 44.0 | 50.0 | 88.8 | 99.2 |
| 80.05 | 35.6 | 41.0 | 67.2 | 73.2 | 135.2 | 144.8 |
| 80.07 | 46.8 | 54.8 | 89.2 | 96.8 | 179.2 | 192.4 |
| 80.09 | 52.0 | 60.4 | 102.8 | 110.2 | 206.4 | 220.4 |
| 90.01 | 14.6 | 17.2 | 26.2 | 28.2 | 53.2 | 58.4 |
| 90.03 | 24.8 | 29.4 | 45.2 | 51.0 | 91.2 | 102.8 |
| 90.05 | 36.0 | 42.6 | 70.8 | 74.8 | 142.4 | 154.8 |
| 90.07 | 50.2 | 56.0 | 96.4 | 102.8 | 193.6 | 208.2 |
| 90.09 | 59.2 | 66.8 | 115.8 | 125.0 | 232.4 | 246.4 |

All statistics are averages of 5 test problems.

We have generated triangle free graphs by modifying the previous random graphs by means of a procedure which takes in input a graph with edge density equal to 0.5 and eliminates those edges inducing a triangle in the

graph. In Tables 3 we report results obtained only for $\Lambda_5$ since each clique and its neighborhood in a triangle free graph induces a bipartite graph and $\Lambda_4$ cannot obtain any lower bound improvement with respect to $w_K$.

Table 3. $\Lambda_5$ values on triangle free random graphs

| Type | $w_K$ $m_w = 5$ | $\Lambda_5$ $m_w = 5$ | $w_K$ $m_w = 10$ | $\Lambda_5$ $m_w = 10$ | $w_K$ $m_w = 20$ | $\Lambda_5$ $m_w = 20$ |
|---|---|---|---|---|---|---|
| tr30 | 10.0 | 11.8 | 20.0 | 23.8 | 40.0 | 46.8 |
| tr50 | 10.0 | 12.0 | 20.0 | 24.2 | 40.0 | 48.0 |
| tr70 | 10.0 | 12.4 | 20.0 | 24.2 | 40.0 | 48.6 |
| tr80 | 10.0 | 12.4 | 20.0 | 25.2 | 40.0 | 50.2 |
| tr90 | 10.0 | 12.8 | 20.0 | 25.4 | 40.0 | 52.2 |

All statistics are averages of 5 test problems.

As for the previous tables it can be observed that $\Lambda_5$ improves on the maximum weighted clique, where this improvement ranges from 12% to 30%.

Note that from a computational point of view, we select stars with two rays, paying attention to not to choose triangles to avoid the overlapping of the contributions given by $\Lambda_5$ and $\Lambda_4$. In all our experiments the time needed to compute the proposed lower bounds, once a maximum weighted clique is obtained, is negligible (we observed at most 5 seconds of CPU time).

## 5. Conclusion

In this paper we have presented new theoretical and algorithmic results to compute lower bounds on the chromatic number of weighted graphs. The effectiveness of such algorithms have been evaluated by experiments on random instances, showing that in all the instances tested the algorithms were able to improve the clique bound.

## References

[1] D. Brelaz, *New methods to color the vertices of a graph*, Communications of the ACM **22** (1979) 251–256.

[2] M. Caramia and P. Dell'Olmo, *Iterative coloring extension of a maximum clique*, Naval Research Logistics **48** (2001) 518–550.

[3] M. Caramia and P. Dell'Olmo, *Solving the minimum weighted coloring problem*, Networks **38** (2001) 88–101.

[4] B. Descartes, *Solution to advanced problem*, No 4526. Amer. Math. Monthly **61** (1954) 532.

[5] M.R. Garey and D.S. Johnson, Computers and Intractability (Freeman and Co.: San Francisco, 1979).

[6] M. Kubale, Introduction to Computational Complexity and Algorithmic Graph Coloring (Wydawnictwo GTN, Gdańsk, 1998).

[7] M. Kubale and B. Jackowski, *A generalized implicit enumeration algorithm for graph coloring*, Communications of the ACM **28** (1985) 412–418.

[8] A. Mehrotra and M.A. Trick, *A column generation approach for graph coloring*, INFORMS J. on Computing **8** (1996) 344–354.

[9] T.J. Sager and S. Lin, *A Pruning procedure for exact graph coloring*, ORSA J. on Computing **3** (1993) 226–230.

[10] E.C. Sewell, *An Improved Algorithm for Exact Graph Coloring*, in: D.S. Johnson and M.A. Trick, eds., DIMACS Series in Computer Mathematics and Theoretical Computer Science **26** (1996) 359–373.

Received 13 May 2002
Revised 27 October 2003