AKCE Int. J. Graphs Comb., 9, No. 2 (2012), pp. 169-186

Greedy Algorithms for Triangle Free Coloring

S. Szabó and B. Zaválnij
Institute of Mathematics and Informatics
University of Pécs
Ifjúság u. 6
7624 Pécs, Hungary.
e-mail: *sszabo7@hotmail.com, bogdan@ttk.pte.hu*

Communicated by: S. Arumugam
Received 09 May 2012; accepted 22 May 2012

Abstract

According to [2] the capability of providing good bounds for the size of a maximum clique in a graph is a critical ingredient of an efficient clique search algorithm. In order to establish upper bounds of the size of a maximum clique we introduce some new coloring schemes using s-clique free partitioning. A complexity result will show that finding the optimal number of colors is an NP-complete problem. This can be interpreted loosely as a computational limitation. To overcome this difficulty for practical purposes we suggest various greedy coloring procedures.

Keywords: clique, maximum clique, independent set, vertex coloring, s-clique free set, s-clique free coloring, clique search algorithm, satisfiability solver, greedy coloring.

2010 Mathematics Subject Classification: 05C15.

1. Introduction

Let $\Gamma = (V, E)$ be a finite simple graph and let k be a positive integer.

Definition 1. *A subgraph Δ of Γ is called a k-clique if each two distinct nodes of Δ are adjacent and Δ has k nodes. A k-clique Δ in Γ is called a maximal clique if Δ is not a subgraph of any $(k+1)$-clique in Γ. A k-clique Δ in Γ is called a maximum clique if Γ does not contain any $(k+1)$-clique. The clique number $\omega(\Gamma)$ of Γ is k if Γ has a maximum k-clique.*

We refer to locating cliques in a given graph loosely as clique search problems. We describe some relevant clique search problems formally.

Problem 1. *Given a finite simple graph Γ and given a positive integer k. Decide if Γ contains a k-clique.*

Problem 1 is a decision problem and it is well-known that it belongs to the NP-complete complexity class ((See [9], [10]).

Problem 2. *Given a finite simple graph* Γ *and a positive integer* k*. List all* k*-cliques in* Γ*.*

Problem 2 is not a decision problem and so it cannot belong to the NP-complete complexity class. However, it is clear that Problem 2 cannot be computationally less demanding than Problem 1.

Determining the clique number $\omega(\Gamma)$ of Γ is not a decision problem either. But again it is clear that finding $\omega(\Gamma)$ must be computationally at least as challenging as Problem 1.

Determining the size of a maximum clique in a graph is an important problem with many practical applications. For details see for example [2], [4], [5], [6], [7]. The capability of bounding the size of the maximum clique was identified by [2] as a key ingredient of an efficient clique search algorithm. Motivated by this insight in this paper we will be interested in establishing upper bounds for $\omega(\Gamma)$.

Let $\Gamma = (V, E)$ be a finite simple graph and let s be a positive integer such that $s \geq 2$.

Definition 2. *A subset* U *of* V *is called an* s*-free set if the graph spanned by* U *in* Γ *does not contain any* s*-clique. A partition* $U_1, \ldots, U_r$ *of* V *is called an* s*-clique free partition of* V *if* U_i *is an* s*-clique free subset of* V *for each* i*,* $1 \leq i \leq r$*.*

We color the nodes of Γ such that each node receives exactly one color of the given r colors. A coloring of the nodes of Γ with r colors can be described more formally as a surjective map $f : V \to \{1, \ldots, r\}$. Here we identify the r colors with the numbers $1, \ldots, r$, respectively. The level sets of f are the so-called color classes of the coloring. The i-th color class $C_i = \{v : v \in V, f(v) = i\}$ consists of all the nodes of Γ that are colored with color i. The color classes $C_1, \ldots, C_r$ form a partition of V. Obviously, the coloring is uniquely determined by the color classes $C_1, \ldots, C_r$.

Definition 3. *A coloring of the nodes of* Γ *with* r *colors is called an* s*-clique free coloring if the color classes* $C_1, \ldots, C_r$ *are all* s*-clique free subsets of* V*.*

In particular, in a 2-clique free coloring of Γ adjacent nodes cannot receive the same color. A 2-clique free coloring is commonly referred as a legal or well coloring of Γ. In a 3-clique free coloring of Γ the nodes of a triangle in Γ cannot receive the same color. We will call these colorings triangle free colorings. There is a straightforward connection between the s-clique free colorings of Γ and $\omega(\Gamma)$.

Proposition 1. *If* Γ *has an* s*-clique free coloring with* r *colors, then* $\omega(\Gamma) \leq r(s-1)$*.*

Proof. Let Δ be a k-clique in Γ and suppose that $C_1, \ldots, C_r$ are the color classes of an s-clique free coloring of Γ with r colors. Note that Δ can have at most $s-1$ nodes in each of the color classes $C_1, \ldots, C_r$. It follows that $k \leq r(s-1)$ and therefore $\omega(\Gamma) \leq r(s-1)$. $\square$

If Γ has a 2-clique free coloring with r colors, that is, if the nodes of Γ can be legally colored with r colors, then by Proposition 1, $\omega(\Gamma) \leq r$. The smaller is r, the better is the upper estimate for $\omega(\Gamma)$. However, determining the minimum number of colors which still allows a legal coloring of the nodes of Γ is itself computationally expensive. Namely, the problem of determining if a given graph Γ can be legally colored with r colors belongs to the NP-complete complexity class. For proofs see [9] or [10].

On the other hand there is a collection of greedy algorithms for legal coloring of the nodes of a given graph. These algorithms have polynomial running times but not necessarily find the optimum number of colors.

Problem 3. *Given a finite simple graph Γ and given two positive integers r, s ($s \geq 2$). Decide if Γ has an s-clique free coloring with r colors.*

In the special case $s = 2$ Problem 3 asks for a legal coloring of the nodes of Γ with r colors. This is a well-known NP-complete problem for $r \geq 3$. In Section 2 we will show that Problem 3 is in the NP-complete class for each $s \geq 2$ whenever $r \geq 3$.

2. A complexity result

The main result of this section is the following proposition.

Proposition 2. *Problem 3 is NP-complete for $s \geq 2$, $r = 3$.*

Proof. Let s be a positive integer. Let $H = (V, E)$ be a graph with $3s - 2$ nodes and $(3s-2)(3s-3)/2 - 1$ edges. We write V in the form $A \cup \{b_1, b_2\}$, where $A, \{b_1\}, \{b_2\}$ are disjoint. We connect all the distinct nodes in H. Finally, we delete the edge connecting the nodes b_1 and b_2, as shown in Figure 1.

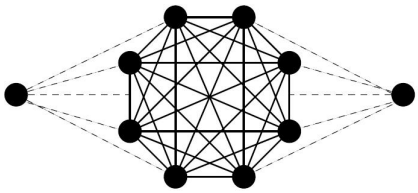

Figure 1: The H graph for $s = 4$ (dotted lines represent connections to all nodes)

Suppose H has an s-clique free coloring with 3 colors. Then the subgraph K of H spanned by A has an s-clique free coloring too. Let C_1, C_2, C_3 be the color classes. We may assume that $|C_1| \geq |C_2| \geq |C_3|$ since this is only a matter of reordering of colors 1, 2, 3. Clearly, $|C_1| \leq s - 1$. From $|C_1| + |C_2| + |C_3| = |A| = 3s - 4$ it follows that $|C_1| = |C_2| = s - 1$, $|C_3| = s - 2$.

The s-clique with nodes $\{b_1\} \cup C_1$ shows that b_1 cannot receive color 1. The s-clique whose nodes are in $\{b_1\} \cup C_2$ provides that b_1 neither can receive color 2. Thus b_1 must

receive color 3. A similar argument gives that b_2 must receive color 3 too. The point we would like to emphasize is that b_1 and b_2 must receive the same color. We may summarize the above considerations in the following way.

(1) The graph H cannot have an s-clique free coloring with less than 3 colors.

(2) The graph H does have have an s-clique free coloring with exactly 3 colors.

(3) For each s-clique free coloring $f : V \to \{1, 2, 3\}$ of H the equation $f(b_1) = f(b_2)$ must hold.

Using $s - 1$ isomorphic copies $H_1, \ldots, H_{s-1}$ of H we construct a new graph L. Here $H_i = (V_i, E_i)$ with $V_i = A_i \cup \{b_{i,1}, b_{i,2}\}$. We solder the nodes $b_{1,1}, \ldots, b_{s-1,1}$ together to get a new node b of L. We rename the nodes $b_{1,2}, \ldots, b_{s-1,2}$ by $c_1, \ldots, c_{s-1}$ respectively and we connect all the distinct pairs among $c_1, \ldots, c_{s-1}$. The set of nodes of L can be written in form

$$A_1 \cup \cdots \cup A_{s-1} \cup \{b\} \cup \{c_1, \ldots, c_{s-1}\}.$$

The sets in this union are pair-wise disjoint and so the number of the nodes of L is $(3s - 3)(s - 1) + 1$. An example L graph for $s = 4$ is shown in Figure 2.

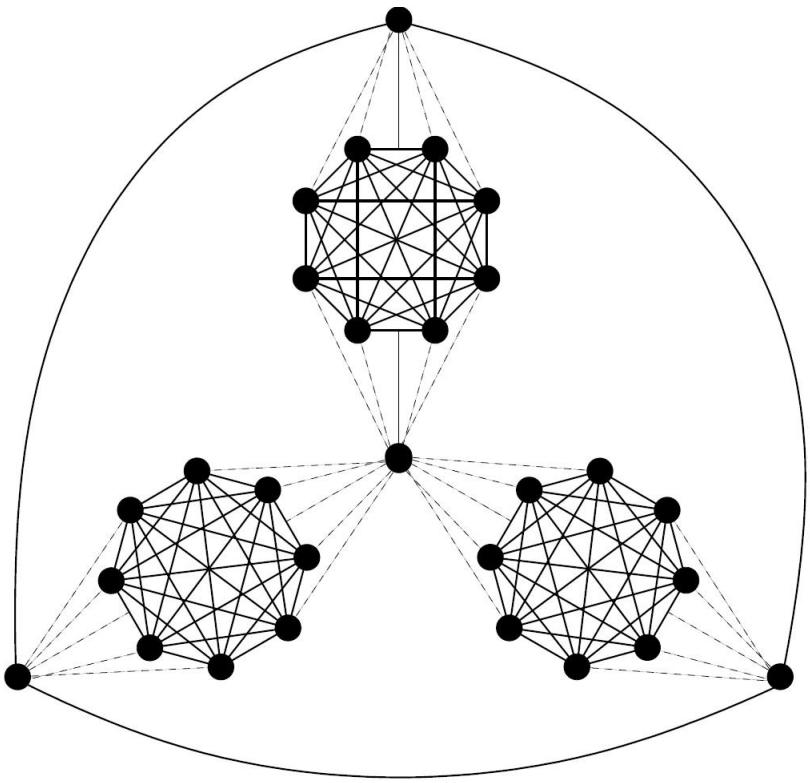

Figure 2: The L graph for $s = 4$

Suppose L has an s-clique free coloring with 3 colors. This provides an s-clique free coloring for H_i for each i, $1 \leq i \leq s-1$. The nodes $b, c_1, \ldots, c_{s-1}$ must receive the same color. We will say that the uniquely determined color of the nodes $b, c_1, \ldots, c_{s-1}$ is associated with the graph L. Sometimes we express this fact by saying that L is labeled with the color of the nodes $b, c_1, \ldots, c_{s-1}$.

Using a given graph $G = (V, E)$ we construct a new graph $G' = (V', E')$. If $v_1, \ldots, v_n$ are all the nodes of G, then we consider n distinct isomorphic copies $L_1, \ldots, L_n$ of L. Here $L_i = (V_i, E_i)$,

$$V_i = A_{i,1} \cup \cdots \cup A_{i,s-1} \cup \{b_i\} \cup \{c_{i,1}, \ldots, c_{i,s-1}\}.$$

Set $V' = V_1 \cup \cdots \cup V_n$. If v_i and v_j are adjacent nodes in G, then we add the edges $\{c_{i,k}, c_{j,l}\}$, $1 \leq k, l \leq s-1$. If v_i and v_j are not adjacent in G, then we do not add any new edge to G'. Figure 3 illustrates the construction in the special case $s = 3$.

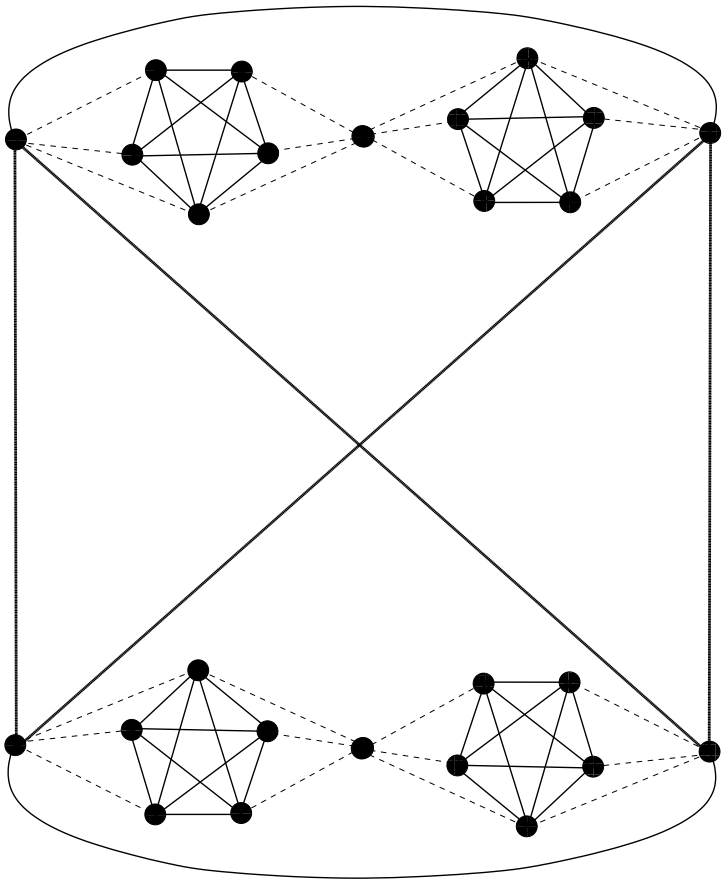

Figure 3: Two connected nodes of G represented in G' for $s = 3$

The graph G' is a blown up version of G. Each node v_i is represented by a graph L_i. If we collapse each L_i to a point, then we get back the graph G. As G has n nodes, G'

has $n[(3s-3)(s-1)+1]$ nodes. (The essential fact is that for a fixed s the number of the nodes of the graph G' is a polynomial in terms of n.)

Assume first that G has an edge free coloring $f : V \to \{1,2,3\}$. We extend f to a coloring $f' : V' \to \{1,2,3\}$ of G'. Namely, we set $f'(c_{i,1})$ to be $f(v_i)$. We know that L_i can be colored with 3 colors such that the color of the node $c_{i,1}$ is prescribed. Thus we may label the graph L_i by color $f'(c_{i,1})$. The nodes $c_{i,1}, \ldots, c_{i,s-1}$ receive the same color $f'(c_{i,1})$.

After labeling L_i for each i, $1 \leq i \leq n$ each node of G' is colored with exactly one color. We denote this coloring of G' by $f' : V' \to \{1,2,3\}$. We claim that f' is an s-clique free coloring. In order to prove this claim note that the nodes of any s-clique inside an L_i cannot receive the same color. Suppose v_i and v_j are adjacent in G. Now the colors $f(v_i)$ and $f(v_j)$ must be distinct.

Let us consider the s-clique whose nodes are $c_{i,1}, c_{j,1}, \ldots, c_{j,s-1}$. From $f'(c_{i,1}) = f(v_i)$, $f'(c_{j,1}) = \cdots = f'(c_{j,s-1}) = f(v_j)$ and $f(v_i) \neq f(v_j)$ it follows that $f'(c_{i,1}) \neq f'(c_{j,1})$, that is, the s-clique is well colored. Therefore f' is an s-clique free coloring of G'.

Next assume that G' has an s-clique free coloring $f' : V' \to \{1,2,3\}$. From f' we construct a coloring $f : V \to \{1,2,3\}$ of G. The graph L_i is labeled by color $f'(c_{i,1})$ and we set $f(v_i)$ to be $f'(c_{i,1})$. We claim that f is an edge free coloring of G. In order to verify the claim assume that v_i and v_j are adjacent nodes in G. Since the nodes $c_{i,1}, c_{j,1}, \ldots, c_{j,s-1}$ form an s-clique, they cannot receive the same color while the $(s-1)$-clique formed by the nodes $c_{j,1}, \ldots, c_{j,s-1}$ must have the same color because of the construction of L_j. It follows that $f'(c_{i,1}) \neq f'(c_{j,1})$. As $f'(c_{i,1}) = f(v_i)$ and $f'(c_{j,1}) = \cdots = f'(c_{j,s-1}) = f(v_j)$, it follows that $f(v_i) \neq f(v_j)$. Therefore f is an edge free coloring of G.

Suppose there is an algorithm A which decides if a given graph G has an s-clique free coloring and the running time of A is a polynomial of the number of the nodes of G. From G we construct the graph G' we described earlier. Submitting G' to A we get an answer if G' has an s-clique coloring with 3 colors. The time of the computation is a polynomial of the number of nodes of G' which in turn is a polynomial of the number of the nodes of G. If G' has an s-clique free coloring with 3 colors, then G has an edge free coloring with 3 colors. If G' does not have an s-clique free coloring with 3 colors, then G does not have an edge free coloring with 3 colors. □

The next theorem is an extension of Proposition 2.

Theorem 1. *Problem 3 is NP-complete for $s \geq 2$, $r \geq 3$.*

Proof. We follow the steps of the proof of Proposition 2 modifying the parameters where it is necessary.

Let s be a positive integer. Let $H = (V, E)$ be a graph with $rs - (r-1)$ nodes. We decompose V into the form $V = A \cup \{b_1, b_2\}$, where $A, \{b_1\}, \{b_2\}$ are pair-wise disjoint sets. We connect all distinct nodes in H with an edge. Finally, we delete the edge connecting the nodes b_1 and b_2, but not deleting the nodes.

Let us assume that H has an s-clique free coloring with r colors. It follows that the subgraph K of H spanned by A inherits the s-clique free coloring. We assume that $|C_1| \geq \cdots \geq |C_r|$. It is clear that $|C_1| \leq s-1$ must hold. From $|C_1|+\cdots+|C_r| = |A| = rs-(r+1)$ it follows that $|C_1| = \cdots = |C_{r-1}| = s-1, |C_r| = s-2$.

Let us watch the s-clique whose set of nodes is $\{b_1\} \cup C_i, 1 \leq i \leq r-1$. This clique shows that b_1 cannot receive color i. Therefore b_1 receives color r. An analogous argument shows that b_2 must receive color r as well. In short b_1 and b_2 must receive the same color.

The essential properties we need are the following.

(1) The graph H cannot possess an s-clique free coloring with less than r colors.

(2) The graph H does admit an s-clique free coloring with exactly r colors.

(3) For each s-clique free coloring $f : V \to \{1, \ldots, r\}$ of H the equation $f(b_1) = f(b_2)$ must hold.

It is a well known fact that the problem of colorability of a graph with r colors is in the NP-complete complexity class. Using our construction we can reduce the ordinary edge-free node colorability with r colors to the s-clique free colorability with r colors. □

3. The simplest greedy coloring procedure

Theorem 1 can be interpreted loosely such that finding the minimum number of colors for which a given graph admits a triangle free coloring is a computationally demanding problem. On the other hand it is relatively easy to construct a triangle free partition for a given graph Γ. To see why suppose that $v_1, \ldots, v_n$ is a fixed list of all the nodes of Γ. We put v_1 into the first color class C_1. Assume that at the middle of the computation the nodes $v_1, \ldots, v_{r-1}$ have already been placed to the color classes $C_1, \ldots, C_{s-1}$. Let us consider the node v_r. If x, y, v_r are not the nodes of a 3-clique in Γ for each $x, y \in C_1$, then we put v_r into C_1. If we cannot put v_r into C_1, then we try to put it into C_2. Continuing in this way either v_r can be placed into some of $C_1, \ldots, C_{s-1}$ or we open a new color class C_s for v_r. Then repeat the whole procedure for $v_{r+1}, \ldots, v_n$. The result will be a triangle free partition of Γ with color classes $C_1, \ldots, C_t$.

So Proposition 1 provides a practical way to find an upper bound for $\omega(\Gamma)$. For the sake of illustration let us work out a small example in details. The graph Γ is shown in Figure 4 and given by its adjacency matrix in Table 1.

We open a color class C_1 and put node 1 into C_1. This information is recorded in the 1-st array in Table 2. We can add node 2 to C_1. In the graph spanned by C_1 in Γ an edge appears. This is edge $\{1, 2\}$. This stage of the computation is recorded in the 2-nd array in Table 2. Node 3 is connected to the end points of the edge $\{1, 2\}$ and so node 3 cannot be put in the color class C_1. We open a new color class C_2 and put node 3 into C_2. The result can be seen in the 3-rd array of Table 2. Node 4 is not adjacent to the end

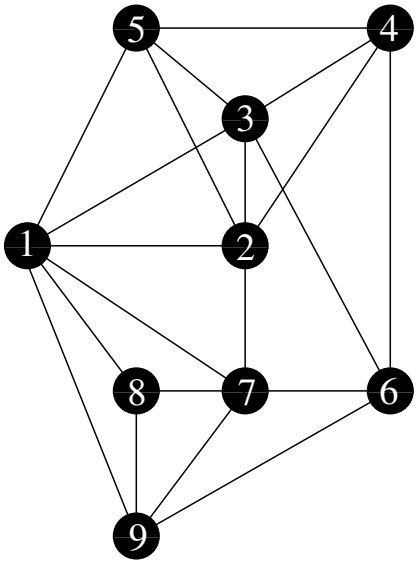

Figure 4: The graph Γ

Table 1: The adjacency matrix of the graph Γ.

	1	2	3	4	5	6	7	8	9
1		•	•		•		•	•	•
2	•		•	•	•		•		
3	•	•		•	•	•			
4		•	•		•	•			
5	•	•	•	•					
6				•	•		•		•
7	•	•				•		•	•
8	•						•		•
9	•					•	•	•	

points of the edge $\{1, 2\}$ and so we may put node 4 into C_1. In the graph spanned by C_1 a new edge $\{2, 4\}$ appears. Node 5 is connected to both end points of the edge $\{1, 2\}$ and so node 5 cannot be put into C_1. We put node 5 into C_2. The edge $\{3, 5\}$ is an edge in the graph spanned by C_2. We add this edge to the edge list associated with C_2. At this point the procedure must be rather clear and we do not comment the following steps.

Table 2: The triangle free coloring procedure.

C_1	edges
1	

C_1	edges
1	$\{1,2\}$
2	

C_1	edges	C_2	edges
1	$\{1,2\}$	3	
2			

C_1	edges	C_2	edges
1	$\{1,2\}$	3	
2	$\{2,4\}$		
4			

C_1	edges	C_2	edges
1	$\{1,2\}$	3	$\{3,5\}$
2	$\{2,4\}$	5	
4			

C_1	edges	C_2	edges
1	$\{1,2\}$	3	$\{3,5\}$
2	$\{2,4\}$	5	
4	$\{4,6\}$		
6			

C_1	edges	C_2	edges
1	$\{1,2\}$	3	$\{3,5\}$
2	$\{2,4\}$	5	
4	$\{4,6\}$	7	
6			

C_1	edges	C_2	edges
1	$\{1,2\}$	3	$\{3,5\}$
2	$\{2,4\}$	5	
4	$\{4,6\}$	7	
6	$\{1,8\}$		
8			

C_1	edges	C_2	edges
1	$\{1,2\}$	3	$\{3,5\}$
2	$\{2,4\}$	5	
4	$\{4,6\}$	7	
6	$\{1,8\}$	9	
8			

From the procedure summarized in Table 2 one can read off the color classes $C_1 = \{1,2,4,6,8\}$, $C_2 = \{3,5,7,9\}$. By Proposition 1, it follows that $\omega(\Gamma) \leq (2)(2) = 4$.

After working out an example it must be clear that constructing a triangle free coloring is more time consuming than constructing an edge free coloring. The extra work with the triangle free coloring is justified only if it provides a better estimate for the clique size. The next result shows that the extra work with the triangle free coloring sometimes pays off since it may provide a better estimate for the clique size.

Proposition 3. *For each edge free coloring of a given graph* Γ *with* $2r$ *colors there is a triangle free coloring of* Γ *with* r *colors.*

Proof. Let us choose an edge free coloring of Γ such that $C_1, \ldots, C_{2r}$ are all the color classes. Set $D_1 = C_1 \cup C_2, \ldots, D_r = C_{2r-1} \cup C_{2r}$. One can check that $D_1, \ldots, D_r$ are triangle free sets. □

4. An improved greedy coloring procedure

The procedure presented in Section 3 is a rather straightforward greedy algorithm to construct a triangle free partition. We can improve on this algorithm. To this end we define degrees of freedom for nodes. Suppose a partial coloring $C_1, \ldots, C_s$ is available and v is a node which is not in any of these classes. The degree of freedom of v with respect to the color classes $C_1, \ldots, C_s$ is d if v can be placed exactly into d of the classes $C_1, \ldots, C_s$. With some extra work we can keep track of the degrees of freedom. It looks reasonable to choose a node v with a minimum degree of freedom and to put node v into an eligible color class. We would like to point out that the heuristic of assigning color to the most constrained node is used in connection with ordinary coloring in [8].

To illustrate the procedure we work out a toy example in details. The graph Γ is defined by its adjacency matrix in Table 3.

Table 3: The adjacency matrix of Γ.

	1	2	3	4	5	6	7	8	9	10	11	12
1		•							•		•	•
2	•				•		•	•		•	•	•
3				•		•			•			•
4			•					•				
5		•				•	•	•		•		
6			•		•		•	•			•	
7		•			•	•		•		•		
8		•		•	•	•	•			•		
9	•		•								•	
10		•			•		•	•				•
11	•	•				•			•			•
12	•	•	•							•	•	

We maintain a table in the course of our work. The rows are labeled with the nodes of Γ. We open a color class C_1 and put node 1 into C_1.

Suppose that $C_1, \ldots, C_r$ are partially filled color classes. We keep a column for each of these color classes. The columns are labeled by $C_1, \ldots, C_r$, respectively. A 0 entry in the cell at the intersection of column C_i and row v will indicate that node v can be put into class C_i. If node v cannot be put into the color class C_i, then the cell will contain a 1. Counting the 1's and 0's at the intersection of columns $C_1, \ldots, C_s$ and row v we can determine the degrees of freedom of node v.

The 1-st array in Table 4 shows that node 1 is in the color class C_1. The 0's in column C_1 record the fact the any of the nodes $2, \ldots, 12$ can be put into C_1. The bracketed entry in the C_1 column expresses the fact that node 1 belongs to C_1. We choose the first node with the smallest degrees of freedom, which is represented in the column "df". In our case this is node 2. We will call it the pivot element and we will indicate our choice by bracketing the pivot element in the 1-st column. We can put node 2 into C_1. Edge $\{1, 2\}$ is an edge of the subgraph of Γ spanned by C_1 in Γ. We enter the edge $\{1, 2\}$ to the column labeled by "edges". Node 11 is a common neighbor of the nodes 1 and 2. We put a 1 into row 11 in column C_1. Similarly, node 12 is a common neighbor of the nodes 1 and 2 and we put a 1 in row 12 in column C_1. Next we can determine the degrees of freedom of the nodes $2, \ldots, 12$. Finally, we choose node 11 to be the new pivot element. The result is in the 2-nd array of Table 4.

Table 4: The first 2 steps of the triangle free coloring procedure.

nodes	C_1	edges	df
1	[0]		
[2]	0		1
3	0		1
4	0		1
5	0		1
6	0		1
7	0		1
8	0		1
9	0		1
10	0		1
11	0		1
12	0		1

nodes	C_1	edges	df
1	[0]	$\{1, 2\}$	
2	[0]		
3	0		1
4	0		1
5	0		1
6	0		1
7	0		1
8	0		1
9	0		1
10	0		1
[11]	1		0
12	1		0

Table 5: The 3-rd step.

nodes	C_1	edges	C_2	edges	df
1	[0]	$\{1,2\}$			
2	[0]				
3	0		0		2
4	0		0		2
5	0		0		2
6	0		0		2
7	0		0		2
8	0		0		2
9	0		0		2
10	0		0		2
11			[0]		
[12]	1		0		1

Table 6: The 4-th step.

nodes	C_1	edges	C_2	edges	df
1	[0]	$\{1,2\}$		$\{11,12\}$	
2	[0]				
[3]	0		0		2
4	0		0		2
5	0		0		2
6	0		0		2
7	0		0		2
8	0		0		2
9	0		0		2
10	0		0		2
11			[0]		
12			[0]		

We have to open a new color class C_2 for node 11. Node 11 in C_2 does not block any node to enter into C_2. We can determine the degrees of freedom. Then we choose node 12 to be the pivot element. We put node 12 into C_2. Continuing in this way we end up with a triangle free coloring of the nodes of the given graph. The color classes of the triangle free coloring of Γ are $C_1 = \{1,2,3,4,5,6,9\}$, $C_2 = \{7,8,11,12\}$, $C_3 = \{10\}$. The subgraph of Γ spanned by C_3 does not have any edge.

Table 7: The last step.

nodes	C_1	edges	C_2	edges	C_3	edges	df
1	[0]	$\{1,2\}$		$\{11,12\}$			
2	[0]	$\{3,4\}$		$\{7,8\}$			
3	[0]	$\{2,5\}$					
4	[0]	$\{3,6\}$					
5	[0]	$\{5,6\}$					
6	[0]	$\{1,9\}$					
7		$\{3,9\}$	[0]				
8			[0]				
9	[0]						
10					[0]		
11			[0]				
12			[0]				

Let $C_1, \ldots, C_t$ be the color classes of a triangle free coloring of a graph Γ. The example above shows that it may happen that the subgraph of Γ spanned by C_i in Γ does not have any edge. In this case C_i is not only a triangle free set but in fact it is an independent set. Proposition 1 has a sharper version.

Proposition 4. *Let $C_1, \ldots, C_t$ be the color classes of a triangle free partition of Γ such that each of the subgraphs spanned by $C_1, \ldots, C_r$ contains edges and each of the subgraphs spanned by $C_{r+1}, \ldots, C_t$ is edge free. Then $\omega(\Gamma) \leq t + r$.*

Proof. Let Δ be a maximum clique in Γ and let U be the set of nodes of Δ. The sets $C_1 \cap U, \ldots, C_t \cap U$ form a partition of U and so $|U| = |C_1 \cap U| + \cdots + |C_t \cap U|$. Note that $|C_1 \cap U| \leq 2, \ldots, |C_r \cap U| \leq 2$ and $|C_{r+1}| \leq 1, \ldots, |C_t \cap U| \leq 1$. It follows that $\omega(\Gamma) = |U| \leq 2r + (t - r) = t + r$. □

Let us consider the graph Γ in the example above. By Proposition 1, we get $\omega(\Gamma) \leq (2)(3) = 6$. By Proposition 4, we get $\omega(\Gamma) \leq 3 + 2 = 5$. The nodes 2, 5, 7, 8, 10 are the nodes of a 5-clique in Γ. Therefore $\omega(\Gamma) = 5$.

5. Coloring via greedy independent sets

Let $G = (V, E)$ be a finite simple graph. We would like to construct a triangle free coloring of the nodes of G with possibly not too many colors. Using the graph G we construct a new graph Γ. The nodes of Γ are the edges of G. Two distinct edges $\{x, y\}$ and $\{u, v\}$ of G are connected in Γ if the subgraph spanned by $\{x, y, u, v\}$ in G contains a 3-clique. Applying a greedy procedure we locate an independent set Δ in Γ. Let U be the set of nodes of Δ. (Of course U is a set of edges of G.) Set $W = \{x, y : \{x, y\} \in U\}$.

Proposition 5. *The subgraph H spanned by W in G is a triangle free graph.*

Proof. Assume on the contrary that H contains a 3-clique and let x, y, z be the nodes of this 3-clique. The edges $\{x, y\}$ and $\{y, z\}$ of G are nodes of Γ. Further these nodes are connected in Γ. Therefore $\{x, y\}$ and $\{y, z\}$ cannot be nodes of Δ. As a consequence $\{x, y, z\}$ cannot be a subset of W. □

Proposition 5 tells that locating an independent set in Γ can be used to find a triangle free set in G. Let us see an example. The adjacency matrix of the given graph G is depicted in Table 8, and the graph is shown in Figure 5.

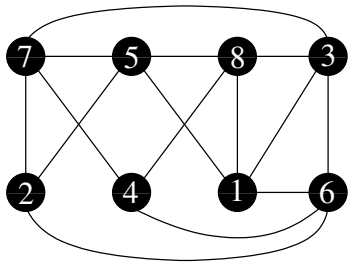

Figure 5: The graph G

Table 8: The adjacency matrix of G.

	1	2	3	4	5	6	7	8
1			•		•	•		•
2					•	•	•	
3	•					•	•	•
4						•	•	•
5	•	•					•	•
6	•	•	•	•				
7		•	•	•	•			
8	•		•	•	•			

Next we set up the adjacency matrix of the graph Γ. The graph G has 15 edges and so Γ has 15 nodes. The adjacency matrix of Γ shown in Table 9, where we label the rows and columns by pairs of nodes of G

The greedy independent set search can be carried out more conveniently if we rename the nodes of Γ. Using simply $1, \ldots, 15$ instead of $\{1, 3\}, \ldots, \{5, 8\}$. We do not include the adjacency matrix of Γ where the rows and columns are labeled by $1, \ldots, 15$.

The set $\{8, 9, 10, 11, 12, 13, 14, 15\}$ is an independent set in Γ. In other words the edges $\{3, 6\}$, $\{3, 7\}$, $\{3, 8\}$, $\{4, 6\}$, $\{4, 7\}$, $\{4, 8\}$, $\{5, 7\}$, $\{5, 8\}$ of G are pair-wise unconnected in Γ. Therefore the nodes $3, 4, 5, 6, 7, 8$ form a triangle free set in G.

Table 9: The adjacency matrix of the graph Γ.

	1,3	1,5	1,6	1,8	2,5	2,6	2,7	3,6	3,7	3,8	4,6	4,7	4,8	5,7	5,8
1,3			•	•		•		•		•	•		•		•
1,5				•			•	•		•			•		•
1,6	•						•	•	•	•					•
1,8	•	•			•			•	•	•				•	•
2,5				•			•		•			•		•	
2,6	•												•	•	
2,7		•	•		•									•	•
3,6	•	•	•	•											
3,7			•	•	•										
3,8	•	•	•	•											
4,6	•														
4,7					•										
4,8	•	•				•									
5,7				•	•	•	•								
5,8	•	•	•	•			•								

In a typical situation we would remove these nodes from G to get a new graph G_1. Then we would construct a graph Γ_1 from G_1. Locating an independent set in Γ_1 we would get a further triangle free set in G_1. Repeating this procedure leads to a triangle free partition of G.

Probably the most straightforward way to locate independent sets in a given graph Γ is to form the complement Γ' of Γ and locate a clique in Γ'. This clique need not to be necessarily a maximum clique. In fact any suboptimal clique will do. For this purpose we recommend the Carraghan-Pardalos clique search algorithm described in [3]. In [5] it is pointed out that the Carraghan-Pardalos procedure finds optimum cliques fairly efficiently but a relatively large amount of time is spent on proving the optimality of the clique. In our case we are content with a quickly identified suboptimal clique and we do not need any certificate of optimality.

6. Coloring via satisfiability

Problem 3 can be reduced to a satisfiability problem. For the sake of definiteness we deal with the special case $s = 3$. Let $G = (V, E)$ be a finite simple graph. We want to color the nodes of G with k colors without monochrome triangles. We introduce propositional variables $c(v, i)$, $v \in V$, $1 \leq i \leq k$. The intuitive meaning of $c(v, i)$ is that node v receives color i. If $n = |V|$, then the number of these variables is nk.

The clause $c(v, 1) \vee \cdots \vee c(v, k)$ makes sure that node v receives at least one of the colors

$1, \ldots, k$. Altogether there are n such clauses. The clauses $\neg[c(v,i) \wedge c(v,j)]$, $1 \leq i < j \leq k$ guarantee that node v cannot receive two distinct colors. The number of these formulas is $n[k(k-1)]/2$. Suppose u, v, w are nodes of a triangle in G. The expressions $\neg[c(u,i) \wedge c(v,i) \wedge c(w,i)]$, $1 \leq i \leq k$ will prohibit the monochrome triangles in G. The number of these expressions is tk, where t is the number of triangles in G.

Let us see an example. The graph G whose adjacency matrix is in Table 8 contains the following 4 triangles $\{1,3,6\}$, $\{1,3,8\}$, $\{1,5,8\}$, $\{2,5,7\}$. We ask if G can be colored with 2 colors such that no monochrome triangles appear. We introduce 16 propositional variables depicted in Table 10.

Table 10: The propositional variables.

1: $c(1,1)$	2: $c(1,2)$	3: $c(2,1)$	4: $c(2,2)$
5: $c(3,1)$	6: $c(3,2)$	7: $c(4,1)$	8: $c(4,2)$
9: $c(5,1)$	10: $c(5,2)$	11: $c(6,1)$	12: $c(6,2)$
13: $c(7,1)$	14: $c(7,2)$	15: $c(8,1)$	16: $c(8,2)$

Table 11: The clauses in DIMACS format.

$1, 2, 0$	$-1, -2, 0$	$-1, -5, -11, 0$
$3, 4, 0$	$-3, -4, 0$	$-2, -6, -12, 0$
$5, 6, 0$	$-5, -6, 0$	$-1, -5, -15, 0$
$7, 8, 0$	$-7, -8, 0$	$-2, -6, -16, 0$
$9, 10, 0$	$-9, -10, 0$	$-1, -9, -15, 0$
$11, 12, 0$	$-11, -12, 0$	$-2, -10, -16, 0$
$13, 14, 0$	$-13, -14, 0$	$-3, -9, -13, 0$
$15, 16, 0$	$-15, -16, 0$	$-4, -10, -14, 0$

The clauses in the 1-st column of Table 11 are responsible for that each node receives at least one color. The clauses in the 2-nd column are responsible for that each node receives at most one color. The responsibility of the clauses in the last column is that monochrome triangle does not appear. We may assume that each of $c(1,1)$, $c(3,1)$, $c(6,2)$ is satisfied since the nodes of the triangle $\{1,3,6\}$ cannot receive the same color.

Feeding the 24 clauses in Table 11 together with the above 3 singleton clauses into a satisfiability solver we may learn that the singleton clauses 1, 5, 12, 16, 3, 7, 9, 14 must be satisfied. In other words $c(1,1)$, $c(3,1)$, $c(6,2)$, $c(8,2)$, $c(2,1)$, $c(4,1)$, $c(5,1)$, $c(7,2)$ all must be true. This provides a coloring of the nodes of the given graph G.

Theorem 1 can be interpreted as a warning that no matter how efficient satisfiability solver we use we may expect instances with long running times. However, the satisfiability solvers can be used in a practicable manner. This based on the following empirical fact.

Suppose that a given graph G has a triangle free coloring with k_0 colors and G does not have any triangle free coloring with less than k_0 colors. Suppose further that we try to find a triangle free coloring of G with k colors. Numerical experiments show that the larger is k compared with k_0 the faster the solvers find a coloring. One can set a time length T and a color number k. If the solver finds a triangle free coloring in T seconds with k colors, then repeat the computation with $k-1$ colors. Of course in a typical situation the value k_0 is not available. So we cannot use k_0 to set a reasonable initial value for k. If the maximum degree of the nodes of G is d, then G has an edge free coloring with $d+1$ colors. By Proposition 3, G has a triangle free coloring with $\lceil (d+1)/2 \rceil$ colors. Thus we may set k to be $\lceil (d+1)/2 \rceil$.

There are so-called incomplete satisfiability solvers that can be used only to find solutions if the satisfiability instance is solvable but cannot be used to verify that the instance is not solvable. The complete solvers often based on exhaustive search while the incomplete solvers are typically employ stochastic local search. (For further details see [1].) In our case both the complete and incomplete solvers are applicable.

References

[1] A. Biere, M. Heule, H. van Maaren, and T. Walsh, (Eds.), *Handbook of Satisfiability*, IOS Press, 2009.

[2] I. M. Bomze, M. Budinich, P. M. Pardalos, M. Pelillo, *The Maximum Clique Problem, Handbook of Combinatorial Optimization Vol. 4*, Kluwer Academic Publisher, 1999.

[3] R. Carraghan, P. M. Pardalos, An exact algorithm for the maximum clique problem, *Operation Research Letters* **9** (1990), 375–382.

[4] J. Hasselberg, P. M. Pardalos, and G. Vairaktarakis, Test case generators and computational results for the maximum clique problem, *J. Global Optim.*, **3** (1993), 463–482. `http://www.springerlink.com/content/p2m65n57u657605n`

[5] D. Kumlander, *Some Practical Algorithms to Solve the Maximum Clique problem* PhD. Thesis, Tallin University of Technology, 2005.

[6] C. Morgan, *A Combinatorial Search with Dancing Links*, PhD. Thesis, Univ. of Warwick, 1999–2000.

[7] P. R. J. Östergård, A fast algorithm for the maximum clique problem, *Discrete Appl. Math.*, **120** (2002), 197–207.

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

[9] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, New York, 2003.

[10] C. H. Papadimitriou, *Computational Complexity*, Addison-Wesley Publishing Company, Inc., 1994.