Solving 3-Colouring via 2SAT

Guillermo De Ita, César Bautista, and Luis C. Altamirano

Computer Sciences, Universidad Autónoma de Puebla, México
{deita,bautista,altamirano}@cs.buap.mx

Abstract. The 3-Colouring of a graph is a classic NP-complete problem. We show that some solutions for the 3-Colouring can be built in polynomial time based on the number of basic cycles existing in the graph. For this, we design a reduction from proper 3-Colouring of a graph G to a 2-CF Boolean formula F_G, where the number of clauses in F_G depends on the number of basic cycles in G. Any model of F_G provides a proper 3-Colouring of G. Thus, F_G is a logical pattern whose models codify proper 3-Colouring of the graph G.

Keywords: 3-Colouring, SAT Problem, Efficient Computing.

1 Introduction

Graph vertex colouring is an active field of research, with many interesting problems [10,7,14]. In the colouring of a graph, we ask to colour the nodes properly with the smallest possible number of colours so that no two adjacent nodes receive the same colour. If such a colouring with k colours exists, the graph is k-colourable. The chromatic number of a graph G, denoted as $\chi(G)$, represents the minimum number of colours for proper colouring G.

To determine the chromatic number $\chi(G)$ of a graph G is a NP-complete problem, even for graphs with degree 3 or higher. As a consequence, there are many unanswered questions related to the colouring of a graph [10].

Many important combinatorial problems are modelled as Constraint Satisfaction Problems (CSP). CSP form a large class of combinatorial logic problems that contains many important 'real-world' problems. An instance of a CSP consists of a set V of variables, a domain D, and a set F of constraints. For example, the domain may be $\{0, 1\}$, and the constraints may be clauses of a Boolean formula in Conjunctive Form (CF). The objective is to assign values in D to the variables in such a way that all constraints are satisfied.

One application of the CSP has been to recognize combinatorial patterns in graphs and to apply techniques developed mainly for solving CSP. For example, let $G = (V, E)$ be an undirected graph where $V = \{v_1, \ldots, v_n\}$. We can associate a monotone 2-CF formula F_G with variables V, and where $F_G = \bigwedge_{(u,v)\in E}(u \vee v)$.

A set $I \subseteq V$ is called an independent set if no two of its elements are joined by an edge. Let $S_I = \{v_j : j \in I\}$ be an independent set in G, then the assignment defined by $x_i = 0$ if $i \in I$ and $x_i = 1$ otherwise, satisfies F_G. The reason is that in every clause $(x_i \vee x_j)$ (representing an edge $\{v_i, v_j\}$) at least one of the variables

J.-F. Martínez-Trinidad et al. (Eds.): MCPR 2011, LNCS 6718, pp. 50–59, 2011.

is assigned to 1, since otherwise the nodes v_i and v_j are in the independent set S_I and then there will not be an edge in G between them.

The above reduction shows us how to apply CSP for modeling different combinatorial patterns of a problem, and in general, for representing key structures which allow us to design smart algorithms. In general, the CSP has been a helpful language for modeling and represent combinatorial patterns on graphs.

1.1 Related Work

Usually, the problem of computing the chromatic number of a graph has been attacked applying dynamic programming and improving upper bounds on the number of maximal independent sets.

In the area of exact algorithms for computing the chromatic number of a graph, the first algorithm was designed by Lawler [11] and it has a running-time of $O*((1+\sqrt[3]{3})^n) = O(2.4423^n)$, where $O*(\cdot)$ means that polynomial factors are ignored. The Lawler's algorithm had not been improved for 25 years [4].

Following the line of exact algorithms and using maximal independent sets to compute the chromatic number, Eppstein established an $O*((4/3+(3/4)^{4/3})^n) = O(2.4151^n)$ time algorithm [2]. And Byskov provided an $O(2.4023^n)$ algorithm [5]. All of those algorithms utilize exponential size memory. While Bodlaender proposed an $O(5.283^n)$ running-time algorithm but using a polynomial size memory [4].

When the number of colourings is fixed, faster algorithms have been designed. The currently best known bound for k-colouring, are: $O(1.3289^n)$ for $k = 3$ [2]. For $k = 4$, the $O(1.7504^n)$ algorithm by Byskov [5], and for $k = 5$, the $O(2.1020^n)$ algorithm by Byskov and Eppstein (see [5]). Each one of those algorithms uses polynomial size memory.

There has also been some related work on approximate or heuristic 3-Colouring algorithms. Blum and Karger have shown that any 3-chromatic graph can be coloured with $O(n^{3/14})$ colours in polynomial time [3]. Alon described a technique for colouring random 3-chromatic graphs in expected polynomial time [1]. Vlasie has described a class of instances which are (unlike random 3-chromatic graphs) difficult to colour [13].

On the other hand, for $k \leq 2$ the problem is polynomially solvable, as is the general problem of colouring for many classes of graphs such as: interval graphs, chordal graphs, comparability graphs [7], the 3-Colouring for AT-free graphs [12] and more generally, for perfect graphs [9]. In these cases, special structures (patterns) have been found which identify those classes of graphs such that they also allow the design of polynomial-time algorithms on them.

We consider here the lowest value for k where the k-colouring is a NP-complete problem, that is, the 3-Colouring problem, and instead of using maximal independent sets of the input graph G, we consider here the CSP as a formulation for codifying proper 3-Colourings. We show that a subset of proper 3-Colourings of G exists which are codified as models of a 2-CF Boolean formula F_G, where the number of clauses in F_G depends on the number of basic cycles in G.

Then, to build the 2-CF F_G and to determine its models (if any exist) can be done in polynomial time on the number of basic cycles of the graph. And since 2SAT problem is polynomially solvable, our proposal changes the parameter for measure the time-complexity of the 3-Colouring problem: from the size of the input graph G to the number of basic cycles in G.

The main advantages of our proposal is to show how to build the 2-CF F_G whose models represent proper 3-Colourings of G, as well as to compute such 3-Colourings in polynomial time based on the number of basic cycles of G.

2 Preliminaries

Let $G = (V, E)$ be a simple graph (i.e., finite, undirected, loop-less and without multiple edges). $V(G)$ and $E(G)$ are also used to denote the sets V and E, respectively. Two vertices v and w are called *adjacent* if there is an edge $\{v, w\} \in E$, connecting them. Sometimes, the shorthand notation of $u\ v$ is used for denoting the edge $\{u, v\}$. The cardinality of a set A is denoted by $|A|$.

The neighborhood of a vertex $v \in V$ is the set $N(v) = \{w \in V : \{v, w\} \in E(G)\}$, and the closure neighborhood of v is $N[v] = N(v) \cup \{v\}$. The degree of a node v, denoted by $\delta(v)$, is the number of neighbors it has, that is $\delta(v) = |N(v)|$. The degree of a graph G is $\Delta(G) = \max_{x \in V}\{\delta(x)\}$.

A path from v to w is a sequence of edges: $v_0v_1, v_1v_2, \ldots, v_{n-1}v_n$ such that $v = v_0$ and $v_n = w$ and v_k is adjacent to v_{k+1}, for $0 \leq k < n$. The length of the path is n. A simple path is a path where $v_0, v_1, \ldots, v_{n-1}, v_n$ are all distinct.

A cycle is just a nonempty path such that the first and last vertices are identical, and a simple cycle is a cycle in which no vertex is repeated, except that the first and last vertices are identical. A simple cycle is even if it has an even number of edges while the cycle is odd when it has an odd number of edges. A graph G is acyclic if it has no cycles.

Definition 1. *Let G be a graph. Then G is said to be connected if for each pair $u, v \in V$ there exists a path from u to v. A* connected component *of G is a maximal subgraph of G induced by this equivalence relation, that is, a connected component is not a proper subgraph of any other connected subgraph of G.*

For example, a tree graph is an acyclic connected graph. Given an undirected connected graph $G = (V, E)$ to apply a depth-first search for traversing G produces a tree graph T_G, where $V(T_G) = V(G)$. The edges in T_G are called *tree edges*, whereas the edges in $E(G) \backslash E(T_G)$ are called *back edges*.

Let $e \in E(G) \backslash E(T_G)$ be a given back edge. The union of the path in T_G between the endpoints of e with the edge e itself forms a simple cycle, such cycle is called a basic (or fundamental) cycle of G with respect to T_G. Each back edge holds the maximum path contained in the basic cycle which is part of.

Let us consider $\mathcal{C} = \{C_1, \ldots, C_k\}$ the set of basic cycles found during the depth first search on G. Given any pair of basic cycles C_i and C_j from $\mathcal{C}$, if C_i and C_j share any edges, they are called *intersecting* cycles; otherwise they are

called *independent* cycles. We say that a basic cycle $C_i \in \mathcal{C}$ is independent if C_i and $C_j \in \mathcal{C}$ are disjoint for each $j \neq i$.

It is known that any acyclic graph is 2-colourable and therefore, 3-colourable as well. Furthermore, if a graph G has only cycles of even length then G is also 2-colourable since to alternate two colours with respect to the levels of the tree T_G builds a proper 2-colouring.

Cycles of odd length require at least 3 colours to make proper colourings. Therefore, the subgraph structures which generate conflict for the 3-Colourings are the odd cycles. We show in the following subsection how to treat those structures for building 3-Colourings.

3 A Heuristic for Building Proper 3-Colourings

Let $G = (V, E)$ be a simple connected graph. Assume an order on the vertices V over which depth-first search is based. Then, the application of the depth-first search over G builds a new depth-first graph G' and a tree graph T_G. During the depth-first search, two sets C_o and C_e are formed: C_o contains the basic cycles of odd length in G and C_e the basic cycles of even length in G. The following procedure is a heuristic for reducing the problem of searching for a 3-Colouring of G to determine the satisfiability of a 2-CF F_G.

Procedure 3_Col(G)

1. We start colouring the nodes in T_G using two colors $\{G, R\}$: the same colour for all nodes at the same level, and with different colours for different levels, that is, alternating colours with respect to the levels of the tree T_G.
2. If $C_o = \varnothing$ then G is 2-Colourable.
3. If all the odd basic cycles are independent, then G is 3-colourable.
   Two alternating colours on T_G and a third colour assigned to just one of the two nodes of each back edge of the cycles in C_o build a proper 3-Colouring of G.
4. Let $CE \subseteq C_e$ be the set of cycles of even length which are intersecting with any odd cycle in G', i.e.,

$$CE = \{C_i \in C_e \ : \text{ length of } C_i \text{ is even and } \exists C_j \in C_o,\ C_i \cap C_j \neq \varnothing\}.$$

5. We assume $k = |C_o|$, there are k odd basic cycles in G'. Let $De = \{be_1, \ldots, be_k\}$ be the set of k back edges formed from C_o, such that each $be_j \in De$ is the back edge of $C_j \in C_o, j = 1, \ldots, k$.
6. For each $be_j = \{x_j, y_j\} \in De, j = 1, \ldots, k$ the two binary clauses are built: $A_j = (x_j \vee y_j) \wedge (\overline{x}_j \vee \overline{y}_j)$.
7. For each pair of different back edges in De: $be_l = \{x_l, y_l\}$ and $be_j = \{x_j, y_j\}$, $l \neq j$ where one of its endpoints (x_j or y_j) is adjacent to (x_l or y_l), e.g. assuming that x_l and x_j are adjacent nodes, the following binary clause is built: $B_j = (\overline{x_l} \vee \overline{x_j})$.

8. For each back edge $e_l = \{x_l, y_l\}$ of a cycle in CE, the following binary clause is built: $E_l = (\overline{x_l} \vee \overline{y_l})$.
9. Let F_G be the 2-CF formed by the conjunction of the $A'_i s$ and $B'_j s$ and $E'_l s$ built in the previous steps.
10. Determine if F_G is satisfiable or not. If F_G is satisfiable then G is 3-colourable and any model of F_G represents a proper 3-Colouring of G. Otherwise, our heuristic can not determine a proper 3-Colouring of G.

We show now that any satisfiable assignment of F_G determines a proper 3-Colouring for G. First, we explain in Table 1 what it means that the clauses in F_G are satisfied.

Table 1. Relationship between the third colour and the satisfiability of the clauses

Clauses	*Intended meaning*
$A'_j s$	For each back edge of an odd cycle, just one of its two nodes must be coloured with W
$B'_j s$	Any pair of nodes with colour W must not be adjacent
$E'_j s$	Both nodes of a back edge of an even cycle can not be assigned the colour W

Theorem 1. *Any satisfiable assignment of the 2-CF F_G formed by 3_Col(G) determines a proper 3-Colouring for the input graph G.*

Proof:

If there is a satisfiable assignment s for F_G, the variables in s which are assigned the value True correspond to the nodes which are assigned the third colour W, while the variables in s which are assigned the value false remain with the same colour assigned in the step 1 of the procedure 3_Col.

The set of clauses in $A'_j s$ guarantees that all odd cycles have the colour W assigned to one of their nodes and therefore, all cycle has been 3-colourable, while the set of clauses in $B'_j s$ and $E'_j s$ prohibit assigning the third color at two adjacent nodes. As all remaining substructures in T_G have been previously coloured with G and R then s represents a proper 3-Colouring of G.

On the other hand, if F_G is unsatisfiable then there is no way to assign colour W to one of the endpoints of a back edge of an odd cycle and at the same time to guarantee that two nodes coloured with W are not adjacent. So, 3_Col can not build a proper 3-Colouring for G.

4 Example: Generalized Petersen Graphs

The generalized Petersen graph $P(n, k)$ is made of vertices $a_0, \ldots, a_{n-1}, b_0, \ldots, b_{n-1}$ and edges $\{a_i, a_{i+1}\}$, $\{a_i, b_i\}$ and $\{b_i, b_{i+k}\}$, $i = 0, \ldots, n-1$ where the subindex sum is taken inside of the additive abelian group $\mathbb{Z}_n$ of integers module n. The subgraph generated by $a_0, \ldots a_{n-1}$ is called the *outside subgraph* while the subgraph generated by $b_0, \ldots, b_{n-1}$ is the *inside subgraph*.

In the following, we are assuming that depth-first search decides what node must be visited based on the following order on the vertices:

$$b_{n-1} > \ldots > b_0 > a_{n-1} > \ldots > a_0,$$

i.e, the greatest vertex is visited first.

Note that two vertices b_i, b_j are in the same connected component of the inside subgraph if and only if $j = i+ks \pmod{n}$ for some integer s, i.e., such connected components are the elements of the quotient $\mathbb{Z}_n k/\mathbb{Z}_n$, which has order g: the greatest common divisor (gcd) of n and k. Therefore, the number of connected components of the inner subgraph of $P(n,k)$ is g and each component has the same number of elements n/g.

Let us recall that when a depth-first search is applied, the pair of numbers called discovery and finishing time are quite useful (see [6]).

Lemma 1. *For each v vertex of $P(n,2)$ let $d[v], f[v]$ the discovery time and the finishing time, respectively, after calling depth-first search on $P(n,2)$. Then*

1. *$d[v] + f[v] = 4n+1$ and $d[v] \leq 2n$, for any vertex v of $P(n,2)$;*
2. *the depth-first search tree $T_{P(n,2)}$ is a chain.*

Proof. Since g, the gcd of n and 2, is 1 or 2, then the number of connected components of the inner subgraph is 1 or 2.

1. By definition, $d[b_{n-1}] = 1$, then, since the vertices of the inside graph are greater than those in the outside graph, they are explored first. Then, the case $g = 1$ is straightforward. Otherwise, if $g = 2$: after the connected component of b_{n-1} in the inner subgraph is exhausted, at the vertex b_1, a node in the outer subgraph is reached, namely a_1, $(d[a_1] = (n/d) + 1)$ then a_2 is explored, and then b_2 in the other connected component, different from the component of b_{n-1}, is explored. This second component is exhausted at b_0 $(d[b_0] = n+2)$, then the vertices explored are $a_0, a_{n-1}, a_{n-2}, \ldots, a_3$ and $d[a_0] = n+3, d[a_{n-1}] = n+4, d[a_{n-2}] = n+5, \ldots, d[a_3] = 2n$. Now all the vertices have been explored, so $f[a_3] = 2n+1$ and we have to trace back all the vertices, so $f[a_4] = 2n+2, \ldots f[a_{n-1}] = 3n-3$ and so on. Note that $f[v] + d[v] = 4n+1$, for all vertices v of $P(n,k)$.
2. Let u, v be two different vertices of $P(n,2)$. Then $d[u] < d[v]$ or $d[u] > d[v]$. In the former case, due to 1) we have $d[u] < d[v] < 4n+1-d[v] = f[v] < 4n+1-d[u] = f[u]$. It follows that v is a proper descendant of vertex u in the depth-first forest graph $T_{P(n,2)}$ (see Theorem [6]). Similarly, in the latter case, we get that u is a proper descendant of v. Thus, for any u, v vertices in $T_{P(n,2)}$, u is a proper descendant of v, or v is a proper descendant of u. This means that $T_{P(n,2)}$ is a chain.

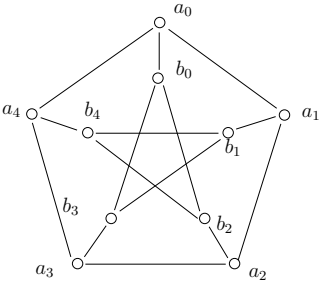
Proposition 1. *Let $n \geq 2$, the number of basic cycles of $P(n,2)$ is $O(n)$.*

Proof. Since the degree of $P(n,2)$ is 3 and $T_{P(n,2)}$ is a chain, then there are at most, $6n$ back edges.

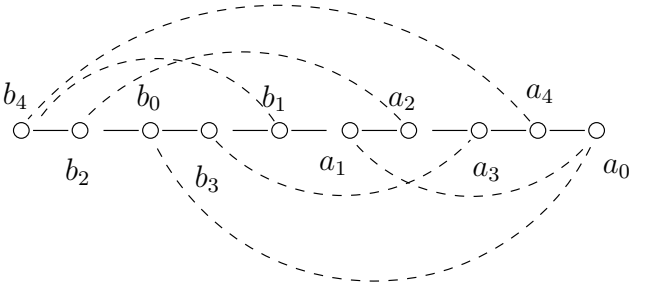
Then 3_Col$(P(n,k))$ determines 3-Colouring in polynomial time.

4.1 Applying the Heuristic on the Petersen Graph

The generalized Petersen graph $P(5,2)$ is

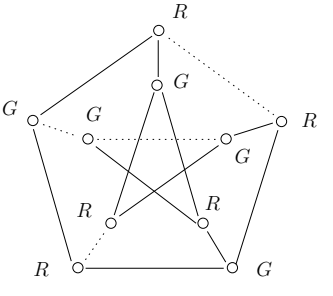


The depth-first search tree $T_{P(5,2)}$ represented by the bold lines, is:



where the dashed lines are the back edges. Then the basic cycles of odd length are $C_o = \{cycle(a_1, a_2, a_3, a_4, a_0), cycle(b_4, b_2, b_0, b_3, b_1, a_1, a_2, a_3, a_4), cycle(b_3, b_1, a_1, a_2, a_3), cycle(b_4, b_2, b_0, b_3, b_1)\}$; while the basic cycles of even length are $C_e = \{cycle(b_0, b_3, b_1, a_1, a_2, a_3, a_4, a_0), cycle(b_2, b_0, b_3, b_1, a_1, a_2)\}$.

Using $T_{P(5,2)}$ we put the alternate colours $\{G, R\}$ on $P(5,2)$. Notice the conflicting colouring at the dashed line edges.



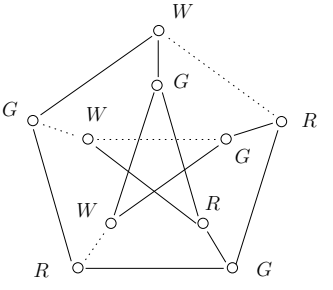
As the basic odd cycles are not independent, the sets $De = \{\{a_1, a_0\}, \{b_3, a_3\}, \{b_4, a_4\}, \{b_1, b_4\}\}$, $CE = \{cycle(b_0, b_3, b_1, a_1, a_2, a_3, a_4, a_0), cycle(b_2, b_0, b_3, b_1, a_1, a_2)\}$ are formed, $k = |C_o| = 4$.

Consider the clauses: $A_1 = (a_1 \vee a_0) \wedge (\overline{a_1} \vee \overline{a_0})$, $A_2 = (b_3 \vee a_3) \wedge (\overline{b_3} \vee \overline{a_3})$, $A_3 = (b_4 \vee a_4) \wedge (\overline{b_4} \vee \overline{a_4})$ and $A_4 = (b_1 \vee b_4) \wedge (\overline{b_1} \vee \overline{b_4})$. $B_1 = (\overline{a_0} \vee \overline{a_4})$, $B_2 = (\overline{b_1} \vee \overline{b_3})$, $B_3 = (\overline{b_1} \vee \overline{a_4})$. From CE we get $E_1 = (\overline{b_0} \vee \overline{a_0})$ and $E_2 = (\overline{b_2} \vee \overline{a_2})$.

$F_{P(5,2)}$ is $A_1 \wedge A_2 \wedge A_3 \wedge A_4 \wedge B_1 \wedge B_2 \wedge B_3 \wedge E_1 \wedge E_2$, i.e.,

$$F_{P(5,2)}: (a_1 \vee a_0) \wedge (\overline{a_1} \vee \overline{a_0}) \wedge (b_3 \vee a_3) \wedge (\overline{b_3} \vee \overline{a_3}) \wedge (b_4 \vee a_4) \wedge (\overline{b_4} \vee \overline{a_4}) \wedge$$
$$(b_1 \vee b_4) \wedge (\overline{b_1} \vee \overline{b_4}) \wedge (\overline{a_0} \vee \overline{a_4}) \wedge (\overline{b_1} \vee \overline{b_3}) \wedge (\overline{b_1} \vee \overline{a_4}) \wedge (\overline{b_0} \vee \overline{a_0}) \wedge (\overline{b_2} \vee \overline{a_2}).$$

Thus, $F_{P(5,2)}$ is satisfiable, for instance, a model is $a_0 = 1, a_1 = 0, a_2 = 0, a_3 = 0, a_4 = 0$ and $b_0 = 0, b_2 = 0, b_3 = 1, b_4 = 1$. Hence, $P(5,2)$ is 3-colourable, and the 3-Colouring is obtained by change the colors of a_0, b_3 and b_4 to W. The 3-Colouring for $P(5,2)$ is shown in the following figure.



5 Algorithm's Time-Complexity

Let $G = (V, E)$ be a simple connected graph with $n = |V|$ and $m = |E|$. A depth-first search over G is of order $O(n+m)$ and it builds an equivalent depth-first graph G'. The set of basic cycles $C = C_o \cup C_e$ is built during the depth-first search. As every cycle has no more than n nodes, then to determine the size of the cycle during the depth-first search involves $O(n+m)$ basic operations.

If the number of basic cycles in T_G is not big, e.g., it is upper bounded by a polynomial function on n or m then C is formed in order $O(poly(n+m))$, *poly* being a polynomial function. Otherwise, it is possible to consider topological graphs where their number of basic cycles grow as an exponential function over n, e.g., the class of graphs K_n (the complete graphs with n nodes), and for this class of graphs, C is built in exponential time-complexity on n.

Let us assume $nc = |C|$ as the number of basic cycles of the input graph. The time required to determine if there are intersecting cycles in a graph is of order $O(nc^2)$ since it consists of the following loop: for all $C \in \mathcal{C}$, for all $C' \in \mathcal{C}$, $C \neq C'$, test if $C \oplus C' \neq \emptyset$, $\oplus$ being the or-exclusive between the set of edges of the cycles. Steps 2, 3 and 4 in the algorithm 3_Col are done in polynomial time on nc, that is, $O(nc^2)$. Step 1 is of order $O(n)$ since it consists of assigning one of the two colours $\{G, R\}$ to each node of the graph. Step 5 runs in order $O(nc)$.

Steps 6,7 and 8 build a 2-CF F_G, the time-complexity for building F_G is related to the size $|F_G|$ which is the number of clauses in F_G. Step 6 builds $2 * |C_o|$ clauses since for each odd basic cycle there are two binary clauses in the $A'_j s$. Step 7 builds at most $|C_e|$ clauses, one clause for each intersecting even cycle, that is, the $E'_j s$ have no more than $|C_e|$ clauses. Step 8 builds one clause

for each pair of adjacent nodes u and v which are endpoints from two different back edges. For this case, at most $n-1$ clauses would be formed since at most each tree edge of T_G could hold that latter condition.

Thus $|F_G| \leq 2 * |Co| + |Ce| + (n-1) < 2 * nc + n$. Of course, for graphs holding $nc < poly(n)$ (when the number of cycles are upper bounded by a polynomial function on the number of nodes) 3_Col runs in polynomial time on n. Furthermore, 3_Col builds proper 3 colourings on that polynomial time.

Otherwise, when $nc >> n$ (the class of graphs where the number of cycles grows as an exponential function of n), we have that $|F_G| < 3 * nc$ and then, 3_Col determines the satisfiability of F_G in polynomial time based on the number of cycles that the input graph G has and so, a proper 3-Colouring of G.

6 Conclusions

We have designed an appropriate combinatorial pattern representing proper 3-Colourings of a graph. Such a pattern is codified by models of a 2-CF.

Thus, the problem of searching for proper 3-Colourings of a graph G is reduced to build satisfiable assignments of a formula F_G in 2-CF, where the number of clauses in F_G depends on the number of basic cycles in G.

The set of satisfiable assignments of F_G is a subset of all proper 3-Colouring of G. Thus, any model of F_G gives a way to 3-Colour G, although it could happen that G is 3-Colourable but F_G will be unsatisfiable.

Since 2SAT problem is polynomially soluble, our reduction shows that to build some proper 3-Colourings of a graph G can be done in polynomial time on the number of basic cycles in G.

References

1. Alon, N., Kahale, N.: A spectral technique for coloring random 3-colorable graphs. SIAM Jour. Comput. 26(6), 1733–1748 (1997), http://www.research.att.com/kahale/papers/jour.ps
2. Beigel, R., Eppstein, D.: 3-coloring in time $O(1.3289^n)$. Journal of Algorithms 54(2), 168–204 (2005)
3. Blum, A., Karger, D.: An $O(n^{3/14})$-coloring algorithm for 3-colorable graphs. Inf. Proc. Lett. 61(1), 49–53 (1997), http://www.cs.cmu.edu/http://www.cs.cmu.edu/avrim/Papers/color_new.ps.gz
4. Bodlaender H.L., Kratsch D.: An exact algorithm for graph coloring with polynomial memory. Technical Report UU-CS-2006-015 (2006), http://www.cs.uu.nl
5. Byskov J.M.: Exact Algorithms for graph colouring and exact satisifiability. Phd thesis, University of Aarbus, Denmark (2005)
6. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press, Cambridge (2009)
7. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs, 2nd edn. North-Holland, Amsterdam (2004)
8. Greenhill, C.: The complexity of counting colourings and independent sets in sparse graphs and hypergraphs. Computational Complexity (1999)

9. Grötschel, M., Lovász, L., Schrijver, A.: The ellipsoid method and its consequences in combinatorial optimization. Combinatorica 1, 169–197 (1981)
10. William, K., Kreher Donald, L.: Graphs, Algorithms, and Optimization. Chapman & Hall/CRC, Boca Raton (2005)
11. Lawler, E.: A note on the complexity of the chromatic number problem. Information Processing Letters 5, 66–67 (1976)
12. Stacho, J.: 3-Colouring AT-Free Graphs in Polynomial Time. In: Cheong, O., Chwa, K.-Y., Park, K. (eds.) ISAAC 2010, Part II. LNCS, vol. 6507, pp. 144–155. Springer, Heidelberg (2010)
13. Vlasie, R.D.: Systematic generation of very hard cases for graph 3-colorability. In: Proc. 7th-IEEE Int. Conf. Tools with Artificial Intelligence, pp. 114–119 (1995)
14. Wilson, R.A.: Graphs, Colourings and the Four-colour Theorem. Oxford University Press, Oxford (2002)