Counting the Number of Three Colourings of a Graph

Arturo Velasco, Guillermo De Ita, Adrian Pérez

Faculty of Computer Sciences, Autonomous University of Puebla
arturoczak@hotmail.com, deita@cs.buap.mx, adrian_fer2002@hotmail.com

Abstract. The problem of counting the number of three colourings of a graph G, denoted as #3-Col(G), is a #P-Complete problem for graphs of degree 3 or higher. We design a novel exact algorithm based on branch and bound technique for counting the number of three colourings of any graph and we compare our proposal with the algorithm resulting of apply the Tutte polynomial of a Graph. We show that if the input graph G does not contain intersecting cycles then our algorithm computes #3-Col(G) with a linear time complexity.

1. Introduction

The problem of counting the number of three colourings in a graph G, problem denoted as #3-Col(G), is a #P-Complete problem for graphs of degree 3 or higher [6], what means that until now, it's not know an efficient algorithm which computes #3-Col(G) for any input graph G.

Some algorithms for counting three colouring based on the maximum independent set are: The algorithms proposed by Dahllöf et. al. [4], the algorithms proposed by Angelsmark et. al. with a complexity time $O(1.8171^n)$ and $O(1.7879^n)$ [1,2], the algorithms proposed by Fürer et. al. [5], with a complexity time $O(1.7702^n)$, being n the number of nodes of the input graph. For the general case, Björklund et. al. [3] has developed an algorithm to compute $P(G,k)$ the number of *k-Colouring* of a graph G with an upper bound in time and space of $O(2^n n^{O(1)})$, based on the inclusion-exclusion method.

Those of above procedures were designed for counting colourings of a graph G based on identifying the *Maximum Independent Set* of the graph (*MIS*), a set S of vertices is a *MIS* if for every two vertices in S there is not edge of G connecting such vertices. To find a *MIS* for any graph is a NP-Complete problem, then it is not probable to find an efficient algorithm for counting #3-Col(G) based on this theory.

In following sections, we will show different procedures for computing the chromatic polynomial $P(G, \lambda)$ (the number of ways to colouring the graph G using as maximum λ colours). We design a novel algorithm based on ramification and bound method for computing #3-Col(G) for any kind of graph. We compare our proposal versus the procedure resulting of apply the Tutte polynomial of a graph [7].

We also show some graph's class for which our algorithm computes #3-Col(G) in polynomial time. Our algorithm does not need to compute the maximum independent set of the graph G instead, it computes #3-Col(G) based on the topological structure

Advances in Computer Science and Engineering.
Research in Computing Science 34, 2008, pp. 15-24

Received 19/03/08
Accepted 26/04/08
Final version 04/05/08

of G_1, being G_1 the resulting graph of apply a *Depth-First Search* (*DFS*) to the input graph G.

2. Notation

Let $G = (V, E)$ be a graph with vertex set $V = V(G)$ and set of edges $E = E(G)$. Two vertices v and w are called *adjacent* if there is an edge $\{v,w\} \in E$, connecting them.

The *Neighbourhood* for $x \in V$ is $N(x) = \{y \in V : \{x,y\} \in E\}$. We denote the cardinality of a set A by $|A|$. The degree of a vertex x, denoted by $\delta(x)$ is $|N(x)|$, and the maximum degree of G is $\Delta(G) = max\{\ \delta(x) : x \in V\}$.

Given a graph $G = (V,E)$, $S = (V',E')$ is a subgraph of G if $V' \subseteq V$ and E' contains edges $\{v,w\} \in E$ such that $v \in V'$ and $w \in V'$. If E' contains every edge $\{v,w\} \in E$, then S is called the subgraph S induced by G and is denoted by $G||S$.

Given a set of colours C, a proper colouring for vertices of a graph $G = (V,E)$ is a function $f: V(G) \rightarrow C$ such that if $\{u,v\} \in E$, then $f(u) \neq f(v)$.

A graph G is *k-colouring* if it admits a colouring with k colours. To the minimum k such that G is *k-colourings* is called the *chromatic number* of G and is denoted by $\lambda(G)$.

Let G be a graph with $\Delta(G) = k$, then $\lambda(G) \leq k + 1$ when G is a regular graph, if G is non-regular graph then $\lambda(G) \leq k$.

If G is a bipartite graph, tree or path of size $|V| = n \geq 2$ then $\lambda(G) = 2$ [6]. This is clear to see since for a bipartite graph each disjoint set V_1 and V_2 are assigned the two colours. For the case of paths, the two colours are alternated, and for the case of tree each level of the tree is colouring with one different colour.

If G is a graph that can be coloured with two colours then the #2-Col(G) problem is solved in polynomial time since the number of 2 colourings of a 2-colourable graph G is equal to 2^c, where c is the number of connected components in G [9].

The number of *k-colourings* of a graph G can be expressed by a polynomial called the *chromatic polynomial* introduced by Birkhoff in 1912 [3], it counts the number of its λ *-Colourings*, i.e. the number of ways to assign colours to the vertices of G in such a way that no two adjacent vertices share the same colour.

Let $G = (V,E)$ be a graph and let be λ a positive integer of colours, we call chromatic polynomial of G, denoted by $P(G, \lambda)$, to the polynomial in the variable λ that it gives us the number of different ways of colouring G using at most λ colours.

Some evaluations for the chromatic polynomial are:

If G is a complete graph (K_n) then: $P(K_n, \lambda) = \lambda\,(\lambda - 1)(\lambda - 2) \cdots (\lambda - n + 1)$.

Let G be a tree or path graph on n vertices then: $P(G, \lambda) = \lambda\,(\lambda - 1)^{n-1}$.

If G is a simple cycle then: $P(G, \lambda) = (\lambda - 1)^n + (-1)^n * (\lambda - 1)$, n being the number of vertices [8,10].

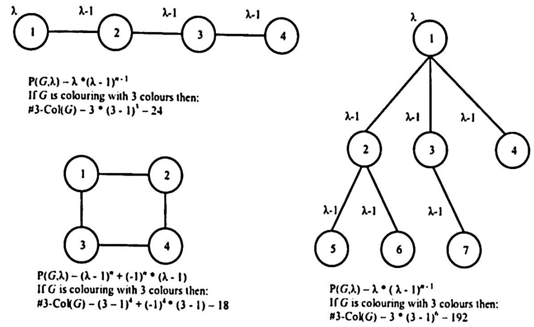


Fig. 1. Computing the Chromatic polinomial and #3-Col(G) over a path, a tree and a cycle.

3. Efficient Counting of 3-Col(G) of a Graph

Although the chromatic polynomial gives us a formula for computing the number of three colourings when the input graphs is a path, tree or cycle, the formula is not easy to combine or to apply when the graph has a combination of the above three topologies. We show here a procedural point of view for computing the number of three colourings for basic topologies such as: paths, trees, and cycles and for any combination of those basic topologies and considering that any pair of cycles of the input graph have not sharing edges.

Given a graph $G = (V,E)$ a *3-Colourings* is an assignment f: $V(G) \rightarrow \{R, A, B\}$ (colours) such that if $\{u,v\} \in E(G)$ then $f(u) \neq f(v)$. We associate to each node $u \in V$ an ordered triple: $(\alpha_u, \beta_u, \gamma_u)$ of integer numbers which carries the number of times that the node u could be properly coloured with the R, A and B colours, respectively.

3.1. Processing Paths

Let $G = P_{n-1}(V,E)$ be a path graph with n vertices: $u_1, u_2, \ldots, u_n$ and n-1 edges: $\{ u_i, u_{i+1}\}$, $1 \leq i \leq n$-1. Let A_G be the graph resulting of apply *Depth-First Search* (*DFS*) on G. The resulting graph indicates the order for processing the nodes. The first node (*leaf*) in *DFS* is calculated as $(\alpha_1, \beta_1, \gamma_1) = (1, 1, 1)$ since the node could be coloured with one of the three colours.

For a node u_i which is not a *leaf* node its triple is calculated by the equation:

$$\alpha_{i+1} = \beta_i + \gamma_i \qquad \beta_{i+1} = \alpha_i + \gamma_i \qquad \gamma_{i+1} = \alpha_i + \beta_i \tag{1}$$

i.e. α_{i+1} is the number of times that the node u_{i+1} can be coloured with the colour R, the same occurs for the values β_{i+1} and γ_{i+1} which give the number of times where the node u_{i-1} is coloured with A and B, respectively. And for the end node v_n (*root*) in the *DFS* the number of three colourings is given as #3-Col(G) = $\alpha_n + \beta_n + \gamma_n$.

3.2. Processing Trees

Let $G = (V,E)$ be a tree graph. The computing of #3-Col(G) is done while G is traversing in post-order, for the next procedure.

Algorithm: Count_3_Colourings_in_trees(G)
Input: A tree graph G
Output: #3-Col(G)
Procedure: Traversing G in post-order, and when a node $v \in G$ is visited, assign:

1) $(\alpha_v, \beta_v, \gamma_v) = (1, 1, 1)$ if v is a *leaf* node in G.
2) If v is a father node with a list of child nodes associated, i.e. $u_1, u_2, \ldots, u_k$ are the child nodes of v, then as we have already visited all the child nodes, then each triple $(\alpha_{uj}, \beta_{uj}, \gamma_{uj})$, $j = 1,\ldots,k$ has been defined based on (1). $(\alpha_{vi}, \beta_{vi}, \gamma_{vi})$ is obtained applying (1) over $(\alpha_{i-1}, \beta_{i-1}, \gamma_{i-1}) = (\alpha_{uj}, \beta_{uj}, \gamma_{uj})$. This step is iterated until computes all triple $(\alpha_{vj}, \beta_{vj}, \gamma_{vj})$, j = 1,...,$k$. And finally, let $\alpha_v = \Pi^k_{j=1}\alpha_{vj}$; $\beta_v = \Pi^k_{j=1}\beta_{vj}$; $\gamma_v = \Pi^k_{j=1}\gamma_{vj}$
3) If v is the *root* node of G then returns $(\alpha_v + \beta_v + \gamma_v)$

This procedure computes #3-Col(G) in time $O(n + m)$ which is the necessary time for traversing G in post-order.

In the fig. 2 we show two graphs, in both the node 1 is the *root* node, in the graph 1 the *leaf* node is the node 4, and in the graph 2 the *leaf* nodes are the nodes 4 and 5, notice that the step 2 is applied to node 2 in the graph G_2.

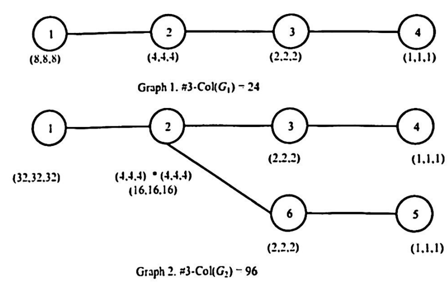


Fig. 2. Computing the *#3-Col(G)* for a path and a tree.

If G is any graph with one vertex, then G can be coloured with anyone of the three colours. If we add a vertex to G and we connect it to the other vertex, then #3-Col(G) = 6 since for each colour of the first vertex we can colouring the second vertex with any colour different to the first vertex, and by the rule of product #3-Col(G) = 3 * 2.

So every time that you add a vertex to G forming a path (or tree) #3-Col(G) is duplicated, then #3-Col(G) = $3 * 2^{n-1}$, being n the number of vertices.

3.3. Processing Graphs with Simple Cycles

Consider a simple cycle $G = (V(G), E(G))$, it can be decomposed as: $G = G' \cup \{c_m\}$, being G' a path with $V(G') = V(G)$. Then, #3-Col(G) = #3-Col(G') − $\Sigma^3_{k=1}$ $|\{f \in$ 3–Col$_k(G')$: $f(u_1) = f(u_m) = k\}|$ i.e. we have to subtract the number of colouring in the cases where $f(u_1)$ is the same that $f(u_m)$.

The equation (1) is apply to G', and in parallel way we compute $|\{f \in$ 3–Col(G'): $f(v_1) = f(v_m) = k\}|$, $k = 1,2,3$, we assign the initial triple $(\alpha_l, \beta_l, \gamma_l)$ to the cases starting just with the first, second and third colour, generating so three new computing threads which start with the triples: (1,0,0), (0,1,0) and (0,0,1) respectively . We represent those three series as: $(\alpha_i, \beta_i, \gamma_i)_k$, $i = 1,..,m$ and $k = 1,2,3$ the three colours. The final triple in each thread considers only the 3-colourings that are the number of colours which has been assigned the same colour in u_1 and u_m. Thus we take as last triple of each thread: $(\alpha_m, 0, 0)$; $(0, \beta_m, 0)$ and $(0, 0, \gamma_m)$ for the colours R, A and B respectively.

The fig. 3 shows the process to compute #3-Col(G) with a simple cycle and Col$_k$, k = 1,2,3 the three colours than can be coloured G. Notice that the triples $(\alpha_i, \beta_i, \gamma_i)_k$ have the same number of *threads$_k$-colourings* to eliminate i.e. Col$_1$ = Col$_2$ = Col$_3$ since are symmetrical, then we can only compute in parallel one colour and repeat it.

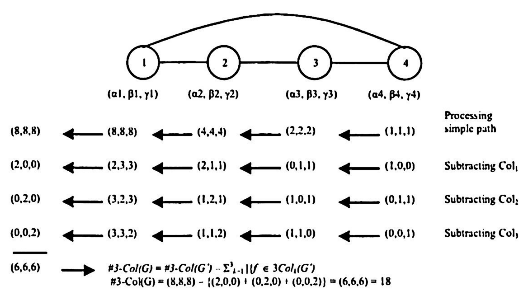


Fig. 3. Counting #3-Col(G) over simple cycle.

We can combine the above procedures, for processing paths and cycles, in one procedure called *Linear_#3-Col*(G) which allow us to processing any path or simple cycle that appear into a more complex graph. The procedure *Linear_#3-Col*(G) has a linear time complexity on the size of the path or the cycle which it processes since we have seen in this section how to process such topologies on linear time.

4. Computing #3-Col(G) in the General Case

If $G = (V,E)$ is any graph, e.g. G has intersecting cycles then an exponential algorithm to compute $P(G, \lambda)$ can be computed based on the Tutte polynomial [7]. The branch and bound is based on the *contraction and deletion of edges* rule. The graph G / e is obtained of G removing the edge e and identifying both incident vertices and merge them, this procedure is called a contraction of the two vertices. The mere deletion of the edge e in G results in the graph $G \setminus$ e with same vertex set V and new edge set E – {e}. The algorithm consists in apply the recurrence:

$$P(G, \lambda) = P(G \setminus \mathrm{e}, \lambda) - P(G / \mathrm{e}, \lambda) \qquad (2)$$

Provided that G is connected and that e is neither a loop nor a bridge edge (i.e. an edge whose deletion does not disconnect the graph).

Let u, $v \in E$, be a pair of vertices in G=(V,E) such that u, v are joined by the edge e. Then $P(G \setminus$ e, $\lambda)$ ($P(G)$ without edge e) can be coloured from 2 ways: when a different colour is assigned to u and v, and when the same colour is assigned to both. Note that the first case is: $P(G, \lambda)$ because the edge e are present and therefore u and v both can not have same colour. The second case is $P(G$ / e, $\lambda)$ because in the contraction we force that u and v both have the same colour.

So the equation (3) is derived of (2).

$$P(G \setminus \mathrm{e}, \lambda) = P(G, \lambda) + P(G / \mathrm{e}, \lambda) \qquad (3)$$

Finally, the chromatic polynomial of a (possibly disconnected) graph is the product of the chromatic polynomials of its connected components.

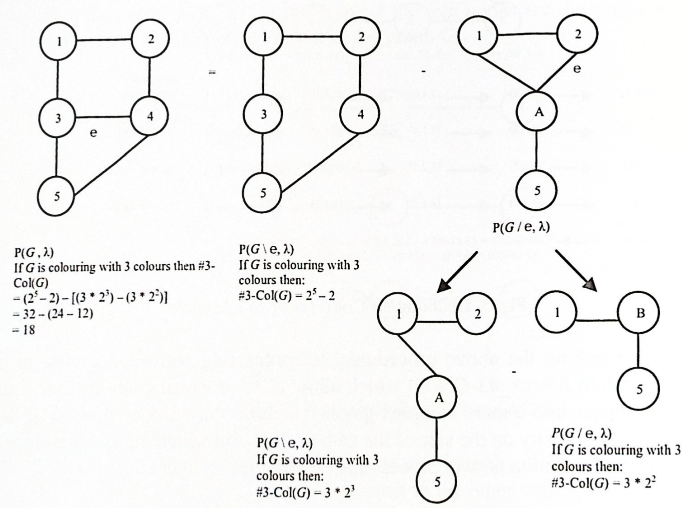


Fig. 4. Apply contraction and deletion algorithm (2).

The fig. 4 shows a graph G with intersecting cycles and the form to apply (2) in order to obtain in this case #3-Col(G). You can see that the recursion is applied twice since the chromatic polynomial neither can compute a cycle and a path in a single recursion.

On the other hand, we design a novel way to compute #3-Col(G) for any graph, we present here our proposal. We design a new split rule for choosing the best node to assign a colour, this split rule follows the idea developed in the classic Davis and Putnam method [2].

Let $G = (V,E)$ be a graph with $|V| = n$, $|E| = m$ and such that $\Delta(G) \geq 2$. Let T_G be the resulting *DFS* graph of G. We assume that T_G is connected and has intersecting cycles. Let $C = \{C_1, C_2,\ldots, C_k\}$ be the set of fundamental cycles stored in the matrix *CM*. The idea is to choose a node v that is part of intersecting cycles, to assign it a colour and eliminate it marking the neighbourhood nodes to restrict them to another colour different to the assigned to v.

Algorithm *General_3Colourings*(T_G)
Input: T_G: a *DFS* graph of a cyclic graph G
Output: #3-Col(G): the number of three colourings of G.

1. Apply *Linear_#3-Col*(G) to T_G in order to process every simple paths and simple cycles that T_G could have. This step processes e.g. any node of degree 1. We consider that after to processing a subgraph type: path, tree or cycle, this subgraph is contracted into a single fat node with the total triple value of such substructure associated to such fat node.

2. Take $S = \{v \in V$: v is part of an intersecting cycle in $T_G\}$. We search for the node $v \in S$ such that v is part of a back edge e_1 and $\delta(v) = max\{\ \delta(u) : u \in S\}$. If $\delta(v) = \delta(u)$ and v and u have maximum degree in S, we select the node $v \in S$ such that its dual degree $\delta(N(v)) = \Sigma_{u\in N(v)}\ \delta(u)$ is maximum into the nodes of S. Notice that the edge e_1 determines a base cycle C_1 and there is at least other cycle C_2 determined by other back edge e_2 since C_1 and C_2 are intersected in T_G.

3. We apply a splitting reduction rule, being v the selected node for performing the splitting. The rule generates three new subgraphs from T_G: G_1, G_2 and G_3, which are formed as $T_G - \{v\}$. We consider that v was assigned one of three colours, and for any node $u \in N(v)$, u is restricted to not set the same colour assigned to v.
Since v is part of a back edge e_1, v is not an articulation point in T_G given that every path crossing by v in T_G, goes now by the other back edge e_2 in G_i = 1, 2, 3. Thus, each G_i, i = 1, 2, 3 is still a connected graph. And for each subgraph G_i = 1, 2, 3 we have that $|V_i| = n_i = n\text{-}1$, $|E_i| = m_i \leq m\text{-}3$, and the number of base cycles in G_i is $nc_i = m_i - n_i + 1 \leq m - 3 - (n - 1) + 1 = m - n - 1$, this is, $nc_i \leq nc - 2$ for i = 1, 2, 3. Then, in each G_i, i = 1, 2, 3 there are at least two cycles less than in T_G.
The application of the splitting rule reduces the number of intersecting cycles by at least two, and builds also an enumerative ternary tree E_G, where each of its nodes has associated a subgraph of T_G.

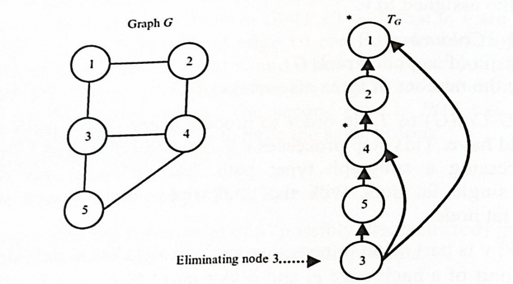
4. Once the splitting rule is applied on T_G, the linear procedure *Linear_#3-Col*(G) is employed again on each subgraph G_i, i = 1, 2, 3 for processing every new sequence of nodes (*paths*) and *simple cycles* that could appear.

5. The splitting rule is applied repeatedly on each subgraph associated with each node of E_G, whenever in such subgraph remain intersecting cycles.

6. When the associated graph G_h of a node of E_G does not have intersecting cycles, then $\#3\text{-}Col(G_h)$ is computed applying the procedures showed in previous sections. And, in such case the node is a leaf node of E_G and does not require the application of the splitting rule. We now have $H(E_G) = \{G_h : G_h$ is the graph associated to a *leaf* node of $E_G\}$.

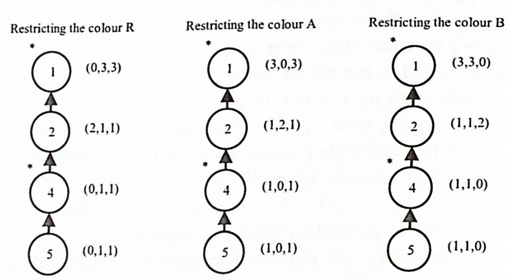
7. After E_G has been built, we have that $\#3\text{-}Col(G) = \Sigma_{G' \in H(E_G)} \#3\text{-}Col(G')$.

The fig. 5 shows our procedure for counting #3-Col(G) on a graph G with intersecting cycles. Notice that each subgraph generated by the splitting rule is symmetric.



Step 2. Node 3 contains a back edge of an elemental cycle and $\delta(v) = \max\{\delta(u): u \in S\}$.
The '*' indicates the constraint for assign the same colour.

Fig. 5a. Appling splitting reduction on T_G (the resulting graph of apply *DFS* of G).



Step 3. Splitting rule generates three subgraphs considered that the node 3 was coloured with the colours R, A and B respectively. The #3-Col(G) = (0 + 3 + 3) + (3 + 0 + 3) +(3 + 3 +0) = 18

Fig. 5b. Colouring each subgraph with the colours *R*, *A*, and *B* respectively.

5. Time Complexity of the Algorithm

Let $G = (V,E)$ be a graph with $|V| = n$, $|E| = m$ and such that $\Delta(G) \geq 2$ for the algorithm *General_3Colourings*(T_G) the steps 1, 2, 4, 6 and 7 have linear time complexity, in fact they are $O(m + n)$. The recursive procedure of splitting when G has intersecting cycles is applied, and its reduction generates three new subgraphs H_i.

The splitting rule selects a node v, such that $\delta(v) \geq 3$ and which is eliminated in every subgraph H_i, either the colour: $\{R, A, B\}$ is assigned to the nodes of $N(v)$. In fact, for each node $v \in N(v)$ at least one of its incident edges has been removed from G to Hi. The time behaviour of the algorithm resides in step 5 and corresponds to the number of intersecting cycles on the graph associated with each node of E_G. Let nc be the variable used for denoting the number of cycles in a graph associated with a node of E_G. For example the number of cycles in a node graph G is given as $nc = m - n + 1$.

The splitting rule opens three new branches with three associated subgraphs Hi and how the node v is deleted in each subgraph then $n_i = n - 1$, $m_i \leq m - 3$, so $nc_i = m_i - n_i + 1 \leq m - 3 - (n - 1) + 1 = m - n - 1 = nc - 2$, $i = 1, 2, 3$. Thus each subgraph has at least two intersecting cycles less than G.

Then, the recurrence for the splitting rule (step 5) is: $T(nc) = 3 * T(nc - 2) = 3^k * \mathrm{T}(nc - 2 * k)$. Such recurrence ends when $nc - 2 * k = 1$, that is when $k = (nc - 1) / 2 = (m - n + 1 - 1) / 2 = (m - n) / 2$. In consequence, the time complexity of our proposal is $\mathrm{T}(nc) \in O(3^{(m - n)/2} * poly(n, m))$. For the worst case, we consider that every pair of cycles appearing in any subgraph are intersecting, then $nc = m - n + 1$ and so:

$$\mathrm{O}(3^{(m-n)/2} * poly(n, m)) \approx \mathrm{O}(1.732^{(m-n)} * poly(n, m)) \tag{4}$$

is an upper bound for the time complexity of our algorithm, n and m being the number of nodes and number of edges of the input graph, respectively.

6. Conclusions

The k-colouring of a graph is a classical problem that it is not computed in polynomial time for any $k \geq 3$. We have shown a novel and exact algorithm to compute the number of 3-*colourings* based on the topological structure of G. When G is a path, cycle or tree we have designed polynomial algorithms for computing #3-Col(G) in linear time, in fact with an upper bound for the worst case of $O(m + n)$, n and m being the number of nodes and edges of the input graph.

We have presented two different strategies for processing graphs with any topology. The second of the strategies is a novel idea that we propose, for computing the number of three colourings for any input graph.

References

1. O. Angelsmark, P. Jonsson, S. Linusson, J. Thapper, Determining the number of solutions to a binary CSP instance, LNCS 2470:327-340, 2002.
2. O. Angelsmark, P. Jonsson, Improved Algorithms for counting Solutions in Constraint Satisfaction Problems, *Int. Conf. on Constraint Programming*, LNCS 2003.
3. A. Björklund, T. Husfeldt, Inclusion-Exclusion Based Algorithms for Graph Colouring, ECCC Report TR06-044, March 2006
4. V. Dahllöf, P. Jonsson, M. Wahlström, Counting models for 2SAT and 3SAT formulae. *Theorical Computer Sciences*, 332(1-3): 265-291, 2005.
5. M. Fürer, S.P. Kasiviswanathan, Algorithms for Counting 2-SAT Solutions and Colourings with Applications, Electronic Colloquium on Computational Complexity, No. 33, 2005.
6. C. Greenhill, The complexity of counting colourings and independent sets in sparse graphs and hypergraphs", *Computational Complexity*, 9(1), 2000, pp. 52-72.
7. M. Noy, Tutte Polynomials in Square Grids, Algorithms Seminar, 2000, available online in http://algo.inria.fr/seminars/
8. N. Alon and M. Tarsi. Colourings and orientations of graphs, Combinatorica, 12:125-134, 1992.
9. F. Jaeger, D.L. Vertigan, D.Welsh, On the Computational Complexity of the Jones and Tutte polynomials, Mathematical Proceedings of the Cambridge Philosophical Society, 108, 1990, p.p.35-53.
10. http://en.wikipedia.org/wiki/Graph_coloring