Short Note

A Las Vegas Graph Colouring Algorithm

It is known that for some NP-complete problem, most instances of the problem can be solved in polynomial time. The classic graph colouring problem is an example for which *Monte Carlo* algorithms have been given. These algorithms always terminate in polynomial time, and *almost always* give the correct result. That is to say, as the size of the graph tends to infinity, the probability of the algorithm giving a wrong answer on a random graph tends to zero. Here we describe a *Las Vegas* algorithm for the same problem. This kind of algorithm always gives the correct result and almost always terminates in polynomial time. The algorithm is based on a traversal of a Zykov tree derived from the graph and uses results previously obtained for a Monte Carlo algorithm for the same problem.

Received June 1988; accepted February 1989

1. Introduction

Let $G = (V, E)$ be a simple undirected graph. A k-colouring of G is a mapping $c: V \to \{1, 2, \ldots, k\}$ and c is a proper colouring if $c(u) \neq c(v)$ for all $\{u, v\}$ in E. The *chromatic number* of G, denoted by $\chi(G)$, is the smallest positive integer k for which a proper k-colouring exists. The *graph colouring* problem is to determine, for a given G and k, whether $\chi(G) \leqslant k$. The problem is a classic, **NP**-complete, decision problem, and so, unless **P** = **NP**, at least some distances are not solvable in polynomial time. However, it has been shown[6] that the intractable instances are vanishingly rare.

In the usual random graph model one considers each possible edge to be present with probability p. We say that some property holds for *almost all* graphs if the probability that the property holds for a graph selected at random from the set of all graphs on n nodes tends to 1 as n tends to infinity. One way of circumventing the **NP**-completeness of a problem may be to create an algorithm that works, perhaps not on all instances of the problem, but on almost all instances. There are two variations on this theme.

A *Monte Carlo* decision algorithm is one which always terminates in polynomial time and which is correct for almost all instances of the problem, i.e., there is a vanishingly small chance of a wrong answer. A *Las Vegas* algorithm is one which always gives a correct answer and almost always terminates in polynomial time, i.e., there is a vanishingly small chance that it may require super polynomial time. Las Vegas algorithms may be more attractive to those who prefer failure to terminate in polynomial time to an unreliable answer.

Following Turner, we call the set of all graphs on n nodes that are k-colourable the *k-colourable graphs*. It can be shown that, for fixed k, almost all graphs contain a k-clique. Consequently, almost all graphs are not $k-1$ colourable. It is then easy to give a Monte Carlo algorithm for the colouring problem, a procedure that always says 'no' will do. It has also been shown that the expected time for backtracking colouring algorithms on random graphs is a constant.[1] To avoid these unhelpful results, one requires that the algorithm perform well on almost all the k-colourable instances and on almost all the not k-colourable instances, taken separately. The *no choice* algorithm described by Turner is a Monte Carlo algorithm that has this characteristic. In Ref. 4 it is shown that one can colour the k-colourable graphs in expected polynomial time. It is the purpose of this paper to describe a Las Vegas algorithm that behaves well on both kinds of instances of the graph colouring problem.

This work was supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada.

2. Zykov Trees

Suppose there exist two non-adjacent vertices u and v in G. Let G_1 denote the graph obtained from G by adding an edge $\{u, v\}$ and let G_2 denote the graph obtained by identifying u and v, i.e., replacing u and v by a single node that is adjacent to each node which was adjacent to either u or v. Repeated application of this construction, until all G_i are cliques, produces a Zykov tree. It is well known, see Ref. 5 for example, that $\chi(G)$ is the size of a smallest clique among the leaves of a Zykov tree derived from G. Figure 2.1 shows an example of a Zykov tree.

Corneil and Graham[2] described an algorithm that computes $\chi(G)$ by finding the smallest clique in a Zykov tree derived from G. The algorithm speeds the search by pruning the tree, which is done by looking for cliques in the graphs corresponding to its nodes. If α is a known upper bound on $\chi(G)$, then there is clearly no need to expand any node which contains a clique of size $\alpha + 1$. The result is of course exact but still requires exponential time, in spite of the pruning.

Dutton and Brigham[3] suggested using the Zykov tree as the basis for an approximation algorithm. One chooses a path down the tree in such a way as to have a good chance of hitting a leaf of minimum size. The path is of course dependent on the choice of node pairs for identification and edge addition. In an attempt to forestall the formation of a clique, the authors suggested that a pair with the maximum number of common neighbours be chosen. No general analysis of this method was given, but some good results were obtained on sample graphs.

Turner[6] gives a Monte Carlo algorithm, called the *no choice* algorithm, and a proof that the algorithm gives the correct result on almost all k-colourable graphs. The Zykov tree is not explicitly used in the description of this algorithm, but we will show that it can be viewed in that context. In this paper we show that one can combine all these results and obtain a Las Vegas colouring algorithm. The fact that this algorithm almost always terminates in polynomial time follows almost immediately from the proof for the *no choice* algorithm.

3. A Las Vegas Colouring Algorithm

In this section we give an almost always polynomial time algorithm for determining whether, for graph G and integer k, G is k-colourable. The algorithm creates and traverses a Zykov tree derived from G. The significant features of the algorithm are:

(1) The procedure can, if necessary, search the entire tree. If a leaf, i.e., clique, is reached which is too large, the procedure does not terminate, it backtracks and continues.

(2) At each branch of the tree, formed either by adding an edge or by identifying nodes, a test is made for the presence of a $(k+1)$-clique. The test procedure is fast because it is restricted to the one portion of the graph at which the edge additions and node identifications are taking place. Consequently a 'No' answer may be incorrect, i.e., there may exist a $(k+1)$-clique, somewhere in the graph, but the test missed it. Hence it may allow unnecessary exploration to continue. The purpose of the test is twofold. Firstly it has a tendency to prune the search tree. Secondly, it ensures that the procedure terminates correctly on graphs that are *not* k-colourable.

(3) The way in which a pair of nodes is selected, for the identification and edge addition operations, is crucial and is described below.

```
function colourable (G:graph, k:integer):
Boolean;
  procedure colour (G:graph);
  var $G_1$, $G_2$:graph;
  if G has $\leqslant k$ nodes then foundcolour ← true
  else select(u,v); $G_2$ ← identify(u,v);
       $G_1$ ← addedge(u,v);
  if not testclique($G_2$, k) then colour($G_2$)
       endif;
  if not foundcolour and not
       testclique($G_1$, k) then colour($G_1$) endif
  endif
  endprocedure
initialise; foundcolour ← false; colour(G);
colourable ← foundcolour
endfunction;
```

The functions *identify* and *addedge* carry out exactly the operations named as previously defined. The procedure *initialise* sets up necessary data structures and finds a *focus clique* for the edge addition and node identification operations. This is done by the function *clique* described below and taken from Turner,[6] who claims linear time complexity, given suitable data representation. The function chooses at random some maximum degree node x and computes a set of nodes comprising a clique containing x. We note that it is not necessarily the largest clique containing x, but, as shown in Ref. 6, it is a largest clique in the graph, almost always.

```
function clique(V:set of nodes of G):graph;
var K, S:set of vertices;
S ← V; K ← $\varnothing$;
while $S \neq \varnothing$ do
    Randomly select $x \in S$ of maximum
    degree; $K \leftarrow K \cup \{x\}$; $S \leftarrow S \cap Neighbours(x)$
endwhile;
clique ← K
endfunction;
```

The procedure *select* selects two non-adjacent nodes for the edge addition and node identification operations so that one of the nodes is in the focus clique and the other is one that has the most neighbours in the clique. If there is more than one such pair, a pair with the most common neighbours is chosen. The edge addition or node identification operations can cause the focus clique to grow. In any

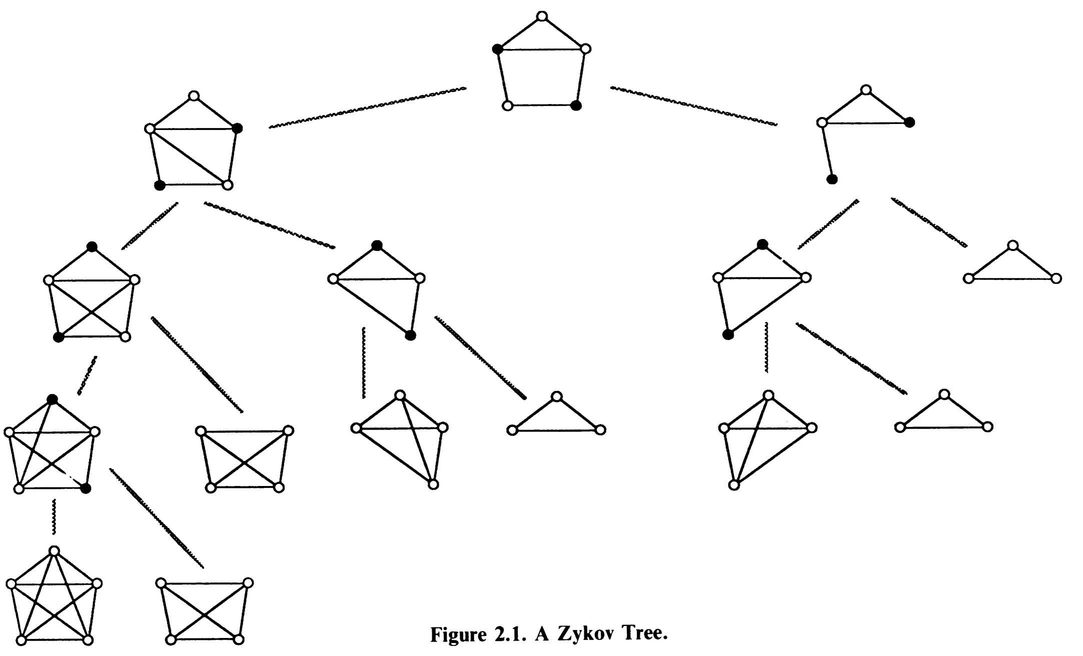

Figure 2.1. A Zykov Tree.

case, the current focus clique is passed down the tree. The function *testclique* checks to see whether the focus clique has become larger than k.

4. Analysis

We will show, by using results from Ref. 6, that, for almost all k-colourable graphs, the first leaf in the Zykov tree encountered by our algorithm corresponds to a clique with no more than k nodes. Since a single traversal from the root to a leaf of the tree clearly can be done in polynomial time, we have our desired result.

4.1. k-Colourable Graphs

The *no choice* algorithms defined in Ref. 6 is as follows:

(1) Randomly select a vertex x and find a k-clique containing x; Properly assign the available k colours to the nodes in this clique;

(2) Repeatedly find a node adjacent to $k-1$ previously coloured nodes and assign the only remaining colour to it, until all nodes are coloured;

We note that, even if a graph is k-colourable, this algorithm may fail at Step 1, or at one of the repetitions of Step 2. It is shown in Ref. 6 (Corollary 3.1), that the algorithm is in fact successful on almost all k-colourable graphs for all $2 \leqslant k \leqslant (1-\varepsilon)\log_2 n$, and fixed ε, $0 < \varepsilon < 1$.

We can deduce immediately that, for almost all k-colourable graphs, the first leaf encountered by our algorithm, will be a clique of size k or less. We note that identifying two nodes in one step of the Zykov tree construction is equivalent to assigning the same colour to those nodes. The k nodes of a k-clique leaf in the tree represent k sets of independent nodes. All the nodes that have been identified with a given clique node are in the same set.

We then note that the first steps in each algorithm are identical and that our selection process will select for identification exactly one of the nodes that is a 'no-choice' node in Turner's algorithm and the one node in the k-clique with which it is not adjacent. Since we follow the identification branch before the edge addition branch, a sequence of successful steps in the *no choice* algorithm is equivalent to an uninterrupted sequence of node identifications in our algorithm.

4.2. Not k-Colourable Graphs

We finish by noting that our algorithm also performs well on *not k-colourable* graphs. It yields the correct result because all branches of the Zykov tree will eventually terminate in leaves that are cliques of size greater than k, and the algorithm will terminate at these leaves, or earlier, with the detection of a $(k+1)$-clique. Let a random graph on n nodes be one in which each edge exists with independent probability p. With $p = \frac{1}{2}$, all graphs on n nodes exist with equal probability. For such a random graph and values of k that are not too large, the probability of any node not being part of a $(k+1)$-clique tends to zero as n tends to infinity, see Lemma 4.1 below. Consequently, on almost all graphs, k-colourable or not, the initialisation almost always finds a $(k+1)$-clique and the process need continue no further.

Let G be a random graph on n nodes, i.e., such that each possible edge exists with a probability p. Let $0 < \varepsilon < 1$, $0 < p < 1$ be fixed, $n \to \infty$ and $2 \leqslant k \leqslant (1-\varepsilon)\lambda(p)$, where $\lambda(p) = -\ln n/\ln p$. Let $1 < r \leqslant k$, and let K_r be any clique of size r in G.

Lemma 4.1

For almost all G, K_r can be extended to a clique of size $r+1$.

Proof

Let u be any node of K_r. The probability that u is adjacent to every node in K_r is p^r. Hence the probability that u is not adjacent to every node in K_r is $(1-p^r)$ and the probability that there is no node adjacent to all nodes in K_r is $(1-p^r)^{n-r}$. There are less than n^r ways of selecting K_r and so the probability that there exists an r-clique that cannot be extended is $\leqslant \sum_{r=1}^{k} n^r(1-p^r)^{n-r} \leqslant kn^k(1-p^k)^{n-k} \leqslant kn^k e^{-(n-k)p^k} \leqslant \exp(\ln k + k\ln n - n^{\varepsilon}) \to 0$, since $k = O(\log n)$. □

5. Experimental Results

The analysis does not tell us how large the graphs must be before the algorithm becomes effective. Let A *random k-colourable* graph be one generated by the following process:

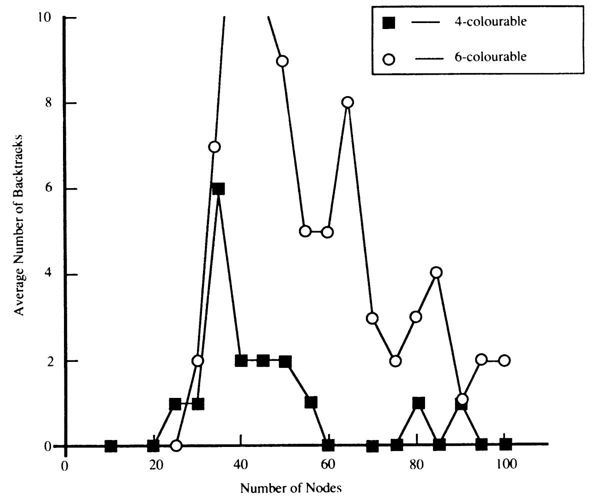

Figure 5.1.

(1) Assign random colours from the set $\{1, 2, \ldots, k\}$ to the n nodes.

(2) For each pair u, v that have been assigned different colours, create an edge $\{u, v\}$ with probability p.

As noted in Ref. 6, this process does not generate all k-colourable graphs with equal probability. However, the probability distribution is closely related to the uniform distribution and this generation method creates a very reasonable test set for a colouring algorithm.

Several experiments were performed on random k-colourable graphs, with edge probability $p = \frac{1}{2}$. The results indicate that the algorithm is effective for relatively small graphs. Figure 5.1 shows typical results. The figure shows the number of backtracks necessary to reach an optimal leaf in the Zykov tree *vs.* size of the graph. Each point is the mean of about 20 random k-colourable graphs of the given size. The largest value for k which we used was 10. In that case the algorithm was effective for graphs of more than 80 nodes.

There are two interesting points. Firstly, there is a range of small n, for which the algorithm may not be effective, i.e., may not terminate within a reasonable time. Secondly, backtracking is often necessary to reach an optimal leaf. For very small n, the algorithm is also effective, but this may have little significance and we do not attempt any analysis.

6. Future Work

We did two other kinds of experiments with results suggesting that further experimentation and analysis might be warranted. Firstly we created an approximate colouring algorithm that traversed the leaves of the Zykov tree in exactly the order defined by our algorithm. Each leaf is of course defining a proper colouring of the graph. By traversing the leaves in the order defined by our algorithm, it is possible that one is looking at an area most likely to contain leaves which correspond to optimal colourings. The approximation algorithm terminates the search at some arbitrary point and takes the best result found so far.

Secondly, we did experiments on graphs generated in the same way as the random k-colourable graphs, except that the edges were sparse, i.e., the number of edges was linearly proportional to the number of nodes. Although more backtracking was required, the algorithm was still effective for $k \leqslant 5$, but not for larger k.

J. A. ELLIS
Department of Computer Science, University of Victoria, Victoria, British Columbia V8W 2Y2, Canada
and P. M. LEPOLESA
National University of Lesotho, Lesotho

References

1. E. A. Bender and H. S. Wilf, A Theoretical Analysis of Backtracking in the Graph Coloring Problem, *Journal of Algorithms* **6**, 275–282 (1985).
2. D. G. Corneil and B. Graham, An Algorithm for Determining the Chromatic Number of a Graph, *SIAM Journal on Computing* **2** (4), 311–318 (1973).
3. R. D. Dutton and R. C. Brigham, A New Graph Colouring Algorithm, *The Computer Journal* **24**, 85–86 (1981).
4. M. E. Dyer and A. M. Frieze, Fast Solution of some Random NP-Hard Problems, *Proc 27th Sym. Foundations of Computer Science*, 331–336 (1986).
5. F. Harary, *Graph Theory*, Addison-Wesley, Reading, Mass. (1969).
6. J. S. Turner, Almost All k-Colorable Graphs Are Easy to Color, *Journal of Algorithms* **9**, 63–82 (1988).