THE COMPUTATIONAL COMPLEXITY

OF EDGE COLOURING RESTRICTED GRAPHS

by

LEIZHEN CAI

B.Sc., Zhejiang University, CHINA, 1982

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

in the Department of Computer Science

[redacted]

We accept this thesis as conforming
to the required standard

[redacted]

(Dr. John A. Ellis)

[redacted]

(Dr. Donald Miller)

[redacted]

(Dr. Frank Ruskey)

[redacted]

(Dr. Arthur Liestman)

UNIVERSITY OF VICTORIA

April 1988

Supervisor: Dr. John A. Ellis

ABSTRACT

Edge colouring a graph is a classic algorithmic graph problem which has many applications. The problem of determining the chromatic index is, in general, NP-complete. This thesis investigates the computational complexity of edge colouring restricted graphs.

Previous work has established that the chromatic index problem restricted to k-regular graphs for any fixed $k \geq 3$ remains NP-complete, whereas the problem restricted to bipartite graphs and partial k-trees for fixed k is in P.

We show that the chromatic index problem restricted to comparability graphs, perfect graphs, line graphs, claw-free graphs, triangle-free graphs and some others remains NP-complete. We present linear time optimal edge colouring algorithms for complete graphs, the line graphs of trees and the line graphs of unicyclic graphs. We also present a linear time approximation algorithm, which uses at most four colours, for cubic graphs, and an $O(1.682^{|V|})$ algorithm for determining the chromatic index of any cubic graph.

Examiners:

[redacted]

(Dr. John A. Ellis)

[redacted]

(Dr. Donald Miller)

[redacted]

(Dr. Frank Ruskey)

[redacted]

(Dr. Arthur Liestman)

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I wish to express my gratitude to my supervisor, Dr. John A. Ellis, for suggesting the topics of thesis to me and for his genial help and valuable guidance throughout the course of my Master's program. I would also like to thank all those people who helped me in various ways.

TO MOTHERLAND

Across the ocean,

there lives my soul.

CHAPTER 1

INTRODUCTION

Let G be a loopless undirected graph or multigraph. A proper edge colouring of G is an assignment of colours to the edges of G such that no two adjacent edges receive the same colour. The chromatic index of G, denote $\chi'(G)$, is the minimum number of colours which permit a proper edge colouring of G.

This problem has many applications. Problems such as routing in a permutation network, preemptive scheduling of an open shop, preemptive scheduling of unrelated parallel processors, the class-teacher timetable problem, the design of experiments, matrix algebra, and Latin squares can be modelled in terms of the edge colouring of a graph [3, 6, 10]. It is also related to other graph theoretic problems, such as vertex colouring, matching, and factorization.

For instance, consider an examination scheduling problem mentioned in [10]. "At the end of an academic year, each student must be examined orally by each of the professors who have taught him. How many examination periods are needed?" We can construct a bipartite graph whose vertices representing students and professors, with edges joining each student to the professors who have taught him. Therefore the chromatic index of this bipartite graph, which equals the maximum degree of the graph [10 p. 25], is just the number of examination periods required.

The origins of the theory of edge colouring graphs can be traced back to 1880, when P.G. Tait published a brief paper in the *Proceedings of the Royal Society of Edinburgh* [32]. He stated that the Four Colour Conjecture is true if and only if the edges of every bridgeless cubic planar graph are 3-colourable. Several results on the chromatic index of a graph appeared after that [10]. A significant and surprising result was obtained by V. G. Vizing who proved that the chromatic index of any simple graph is either equal to its maximum degree or to its maximum degree plus one. Thereafter, the problem of deciding whether a graph with maximum degree Δ needs Δ colours or $\Delta+1$ colours became a major concern in the study of edge colouring.

The general problem of deciding whether the chromatic index of a graph is Δ or $\Delta + 1$ is very difficult. However, it is settled for some special classes of graphs [10]. For example, the chromatic index of a bipartite graph, a complete graph of even order, and a planar graph of maximum degree $\Delta \geqslant 8$ is known to be equal to Δ, whereas both an odd cycle and a complete graph of odd order need $\Delta + 1$ colours. There are many papers on this subject, a bibliographic survey was provided by S. Fiorini [11]. Other surveys can be found in [10, 35]. Recent results in this area include [7, 9, 15, 25, 26].

Computationally, we are interested in two versions of this problem. Given a graph G, we use CHROMATIC INDEX to denote the problem of deciding whether the chromatic index of G is equal to its maximum degree, and EDGE COLOURING to denote the problem of finding an optimal proper edge colouring of G.

In view of the potential applications, it would be very useful to have an efficient algorithm to solve these two versions of the problem. Unfortunately, it was shown by Holyer [17] that CHROMATIC INDEX is NP-complete, even for cubic graphs. This directly implies that EDGE COLOURING is NP-hard. Therefore, both CHROMATIC INDEX and EDGE COLOURING can not be solved by any polynomial time algorithms unless P = NP.

Two approaches are commonly used in dealing with NP-complete (NP-hard in general) problems. One is to restrict the instance of the problem, and the other is to design polynomial time heuristic algorithms to generate approximate solutions. When the instances of an NP-complete problem are restricted, the problem may remain NP-complete or become polynomial time solvable. Approximation algorithms for NP-hard optimization problems are very useful in practice, some of them not only very efficient but also very close to the optimal solution of the problem. For the theory of NP-completeness, we refer readers to [14].

Consider the restrictions on the instances of CHROMATIC INDEX (EDGE COLOURING). D. Leven and Z. Galil [23] proved that CHROMATIC INDEX remains NP-complete for k-regular graphs for any $k \geqslant 3$. However, various polynomial algorithms have been found for EDGE COLOURING, and therefore for CHROMATIC INDEX too, restricted to certain instances. Linear time algorithms were presented for trees and unicyclic graphs [24, 27], linear time algorithms were also presented for Halin graphs [30] and outplanar graphs [28]. An $O(|V||E|)$ general purpose algorithm which works for series-parallel

graphs and planar graphs with maximum degree $\Delta \geqslant 8$ was presented in [33, 34]. Several polynomial time algorithms for bipartite graphs appeared in [8, 12]. The best one takes time $O(|E| log |V|)$. Recently, an $O(n^{1+2^{2(k+1)}})$ time algorithm for CHROMATIC INDEX restricted to partial k-trees was proposed by H. L. Bodlaender [5], which is thus polynomial for fixed k. Since almost trees with parameter k, graphs with bandwidth k, and k-outplanar graphs are partial k'-trees for some k', CHROMATIC INDEX restricted to any of these classes is polynomial time solvable.

Efficient approximation algorithms for the unrestricted case also appeared in the literature. Three approximation algorithms were proposed in [13], which use at most one extra colour and have complexity $O(|E||V|), O(|E|\Delta log |V|)$ and $O(|E|\sqrt{|V| log |V|})$ respectively. E. Arjomandi [1] presented an $O(MIN\{|E||V|, |V|\Delta + \sqrt{|V| log |V|}\})$ time approximation algorithm which uses at most one extra colour.

In this thesis, we mainly study the complexity of CHROMATIC INDEX (EDGE COLOURING) under various restrictions on the instances. Unless otherwise specified, all graphs in this thesis are simple undirected graphs. When the chromatic index of a graph G is k, we say that G is k-chromatic or k-colourable, and call a proper edge colouring of G using k colours a proper k-edge-colouring of G. We usually drop words "proper" and "edge" in this thesis.

In Chapter 2, our main concern is the complexity of CHROMATIC INDEX restricted to special instances. We show that CHROMATIC INDEX remains NP-complete for perfect graphs, comparability graphs, line graphs and claw-free

graphs, thus resolving four open problems in [20]. We also prove the NP-completeness of CHROMATIC INDEX restricted to some other classes of graph.

In Chapter 3, we consider EDGE COLOURING for special line graphs. We show that optimal edge colourings of complete graphs, which are line graphs of stars, line graphs of trees, or line graphs of unicyclic graphs can all be found in time linear in the number of edges in the graph.

In Chapter 4, we consider CHROMATIC INDEX and EDGE COLOURING restricted to cubic graphs. We present a linear approximation edge colouring algorithm, which uses at most four colours, for cubic graphs. We also present an algorithm to determine the chromatic index of cubic graphs.

In Chapter 5, we summarize the known results on the computational complexity of restricted edge colouring problems.

Finally, in the Appendix, we include some basic definitions in graph theory.

CHAPTER 2

NP-COMPLETENESS OF EDGE COLOURING RESTRICTED GRAPHS

The problem of determining the chromatic index of a cubic graph has been proved to be NP-complete by Holyer [17], which immediately implies that the chromatic index problem is NP-complete for general graphs. Moreover, the chromatic index problem restricted to regular graphs of fixed degree k has also been shown to be NP-complete [23]. In this chapter, we present some NP-completeness results on deciding the chromatic index of some restricted graphs. Membership in NP for all of the following problems is clear, since the general problem is in NP.

We will use CHRIND(P) to denote the chromatic index problem restricted to graphs with property P. For example, CHRIND(line graph) denotes the following problem:

INSTANCE: A line graph G.

QUESTION: $\chi'(G) = \Delta$?

We begin by listing the known NP-complete problems concerning the chromatic index of a graph. These results will be used later in our proofs.

Theorem 2.1.[Holyer] CHRIND(cubic graph) is NP-complete.

Theorem 2.2.[Leven, Galil] CHRIND(k-regular graph) for fixed k is NP-complete.

The following lemma is very useful in dealing with the colouring of regular graphs.

Lemma 2.1. (Parity Lemma [18]) Let G be a graph whose vertex degrees are either k or 1, and let $[S, \bar{S}]$ be a cut set of G. Then, in any k colouring of G, if the number of edges coloured i in $[S, \bar{S}]$ is n_i, we have $n_1 \equiv n_2 \equiv \cdots \equiv n_k \ (mod\ 2)$.

2.1. Edge Colouring Perfect Graphs

In this section, we show that CHROMATIC INDEX restricted to comparability graphs remains NP-complete. Since a comparability graph is a perfect graph, we conclude that CHROMATIC INDEX restricted to perfect graphs remains NP-complete.

Definition 2.1. A *comparability graph* is a graph for which there exists an orientation of the edges of the graph such that the resulting digraph represents a transitive closure.

Theorem 2.3.. CHRIND(comparability graph) is NP-complete.

Proof. We present a polynomial transformation from CHRIND(cubic graph) to CHRIND(comparability graph). Given an instance G of CHRIND(cubic graph), we construct an instance G' of CHRIND(comparability graph) by replacing each edge of G with a copy of H as shown in Figure 2.1.

The component H is a comparability graph since a suitable orientation exists for H, which is also illustrated in Figure 2.1. If we give each copy of H

in G' an orientation as in Figure 2.1, we get suitable orientation for G', since there is no directed path between two degree one vertices of each H. Therefore G' is a comparability graph.

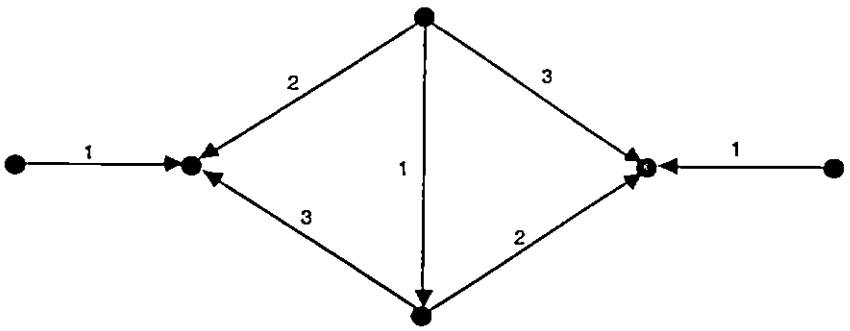


Figure 2.1. Component H and its orientation.

It is easily seen that H is 3-colourable and that for any 3-colouring of H, its two pendants must have the same colour. It follows that G is 3-colourable if and only if G' is 3-colourable. Finally, the transformation is obviously polynomial time. □

Definition 2.2. A *perfect graph* is a graph such that the chromatic number of every induced subgraph equals its maximum clique size.

Corollary 2.1. CHRIND(perfect graph) is NP-complete.

Proof. The comparability graphs form a subset of the perfect graphs. □

Theorem 2.4. CHRIND([$2r$-1]-regular comparability graph) for fixed r is NP-complete.

Proof. By Theorem 2.2, CHRIND(k-regular graph) is NP-complete for fixed k. Thus we present a polynomial time transformation from CHRIND([$2r$-1]-

regular graph) to CHRIND([2r-1]-regular comparability graph). The component H used in the transformation is $K_{2r}-uv+\{ux, vy\}$ as shown in Figure 2.2. H is obviously $2r$-1 colourable, since K_{2r} is $2r$-1 colourable. For any $2r$-1 colouring of H, the two pendants ux, vy always receive the same colour by the parity lemma.

To see that H is a comparability graph, we label the vertices of H by numbers from 0 to $2r$. Vertices u and v are labeled by $2r$-1 and $2r$ respectively; and x, y are labeled 0. Other vertices are labeled distinctly by a number from 1 to $2r$-2. Then the orientation F is given by

$$u_1u_2 \in F \text{ iff } u_1 < u_2 \text{ and } u_1u_2 \in E(H)$$

This orientation represents a transitive closure, therefore H is a comparability graph.

Let G be an instance of CHRIND([2r-1]-regular graph), then we construct an instance G' of CHRIND([2r-1]-regular comparability graph) by replacing each edge of G by a copy of H. It is easy to see that G' can be constructed in polynomial time. We then assign the orientation of each copy of H in G' as specified in Figure 2.2. It is easily seen that this defines a transitive closure orientation for G'. Therefore G' is a comparability graph.

Because of the colouring property of H, it is easy to extend a $2r$-1 colouring of G to a ($2r$-1)-colouring of G'. Conversely, a ($2r$-1)-colouring of G' can be used to derive a ($2r$-1)-colouring of G by deleting each copy of H in G', replacing it by an edge and assigning the colour used by the pendants of the deleted H to this edge. Therefore G is 3-colourable if and only if G' is 3-

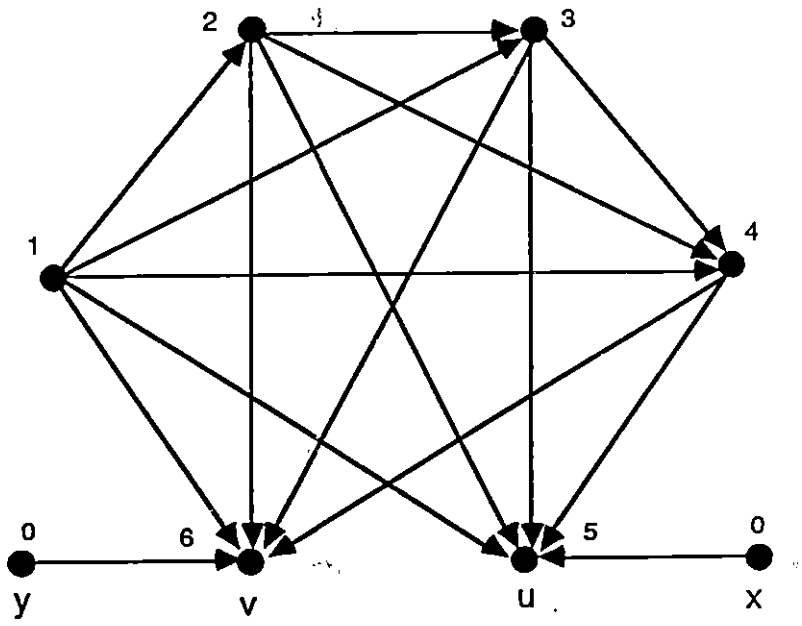


Figure 2.2. A k -regular comparability graph with two outlets.

colourable. □

2.2. Edge Colouring Cycle Restricted Graphs

In this section, we consider CHROMATIC INDEX restricted to C_k-free graphs and graphs of fixed girth.

Definition 2.3. A *triangle-free graph* is a graph which does not contain C_3 as an induced subgraph.

Theorem 2.5. CHRIND(cubic triangle-free graph) is NP-complete.

Proof. We transform CHRIND(cubic graph) to CHRIND(cubic triangle-free graph). The basic component is the graph H as shown in Figure 2.3. By the

parity lemma, H is 3-colourable if and only if the three pendant edges receive distinct colours. Figure 2.3 also shows a 3-colouring of H.

Let G be a cubic graph. If G contains no triangle as an induced subgraph, then let G' be G. Otherwise, we construct a sequence of cubic graphs

$$G = G_0, G_1, \ldots, G_r = G'$$

such that G_i, $1 \leqslant i \leqslant r$, is constructed from G_{i-1} by replacing an induced triangle with its three pendants in G_{i-1} by a copy of H. It is easy to see that G' can be constructed from G in polynomial time.

It suffices to show that G_{i-1} is 3-colourable if and only if G_i is 3-colourable for $1 \leqslant i \leqslant r$. Suppose G_{i-1} is 3-colourable, then for each triangle in G_{i-1} its three pendants receive three distinct colours. We preserve this colouring for the three pendant edges of the component H, and extend it to a

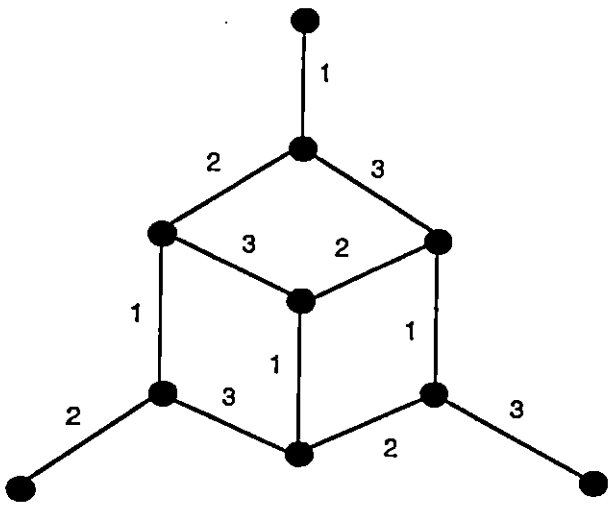


Figure 2.3. A triangle-free component.

3-colouring of H and thus get a 3-colouring of G_i. Conversely, for any 3-colouring of G_i, three pendants of each H must receive distinct colours by the parity lemma. Therefore we can colour the corresponding triangle in G_{i-1} properly by 3 colours and obtain a 3-colouring of G_{i-1}.

Definition 2.4. A graph is said to be a *C_k-free graph* for fixed k iff it contains no induced k-cycle.

Theorem 2.6. CHRIND(cubic C_k-free graph) for fixed $k \geqslant 3$ is NP-complete.

Proof. We have shown in Theorem 2.5 that CHRIND(cubic C_k-free graph) for $k = 3$ is NP-complete. We now present a polynomial time transformation from CHRIND(cubic graph) to CHRIND(cubic C_k-free graph) for fixed $k \geqslant 4$. Given an arbitrary instance G of CHRIND(cubic graph), we construct an instance G' of CHRIND(cubic C_k-free graph) for fixed $k \geqslant 4$ so that G is 3-colourable if and only if G' is 3-colourable.

The technique is local replacement. The substitute H_k consists of a chain of k copies of component H_1. An example is shown in Figure 2.4. Note that H_k is C_k-free for all $k \geqslant 4$. It is easy to check that H_k is 3-colourable and in any 3-colouring of H_k its two pendants always receive the same colour by the parity lemma.

G' is constructed from G by replacing each edge of G with a copy of H_k. It is obvious that the length of any induced cycle in G' is either 3 or no less than $6k$. Thus G' is a C_k-free graph. Clearly, the construction takes

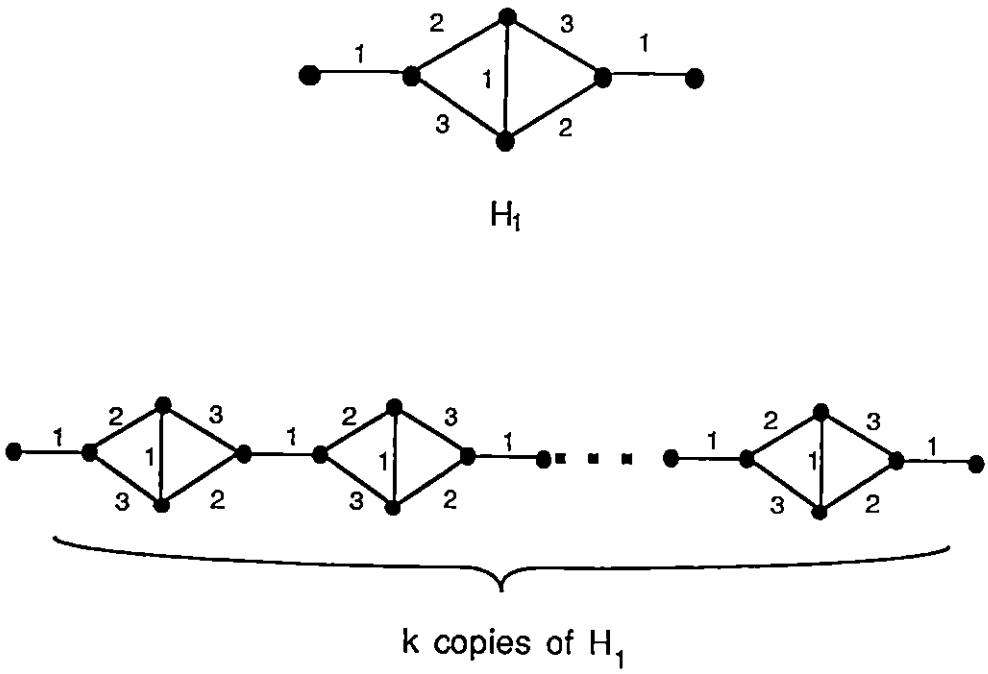


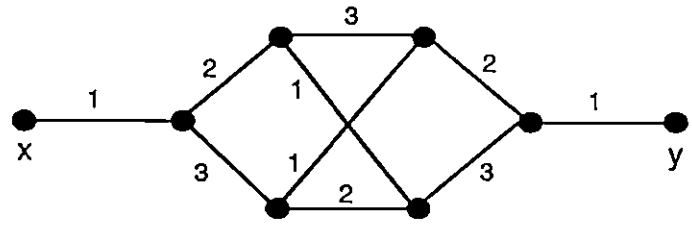
Figure 2.4. A component for constructing a C_k-free graph.

polynomial time.

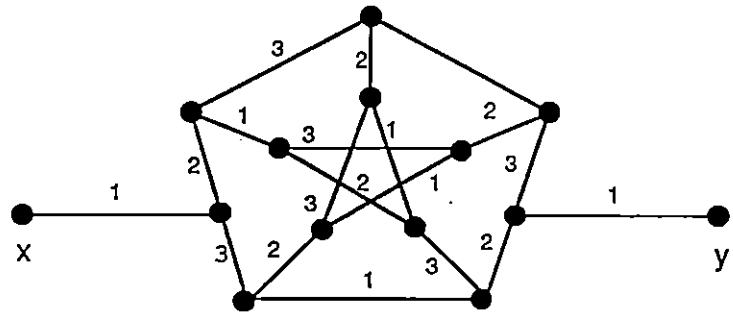
According to the construction of G' and the property of the component H, it is easy to see that G is 3-colourable if and only if G' is 3-colourable, since in any 3-colouring of H_k its two pendants always receive the same colour. Hence CHRIND(C_k-free graph) is NP-complete for fixed $k \geqslant 3$. □

Theorem 2.7. CHRIND(cubic graph of girth k) for $k = 3, 4, 5, 6, 7, 8$ is NP-complete.

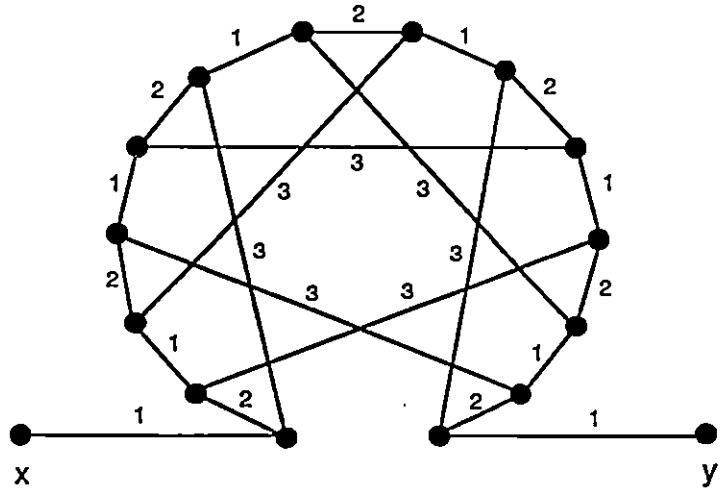
Proof. Notice that, because the components used in the proof of Theorem 2.6 contain C_3, it follows that the theorem is true for $k = 3$.



H_4

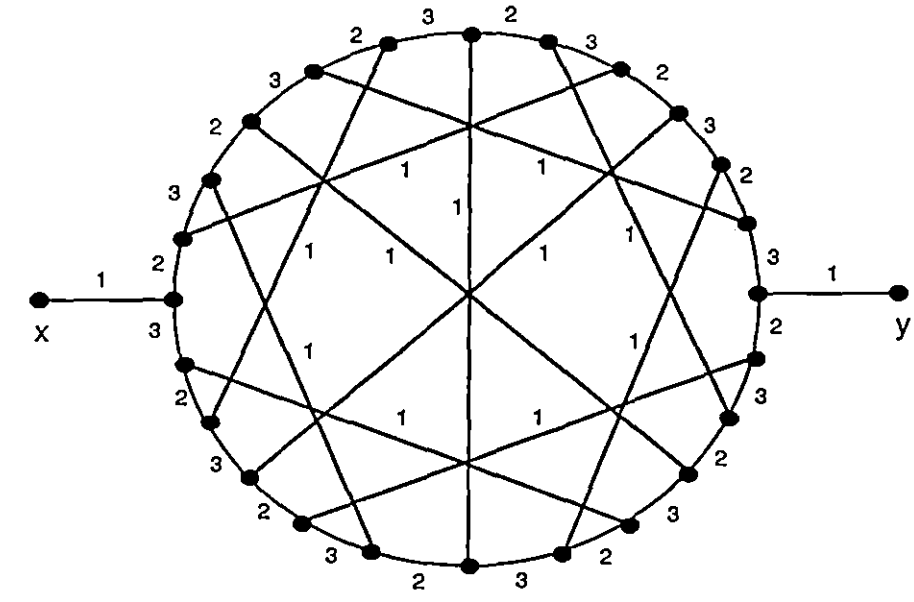


H_5

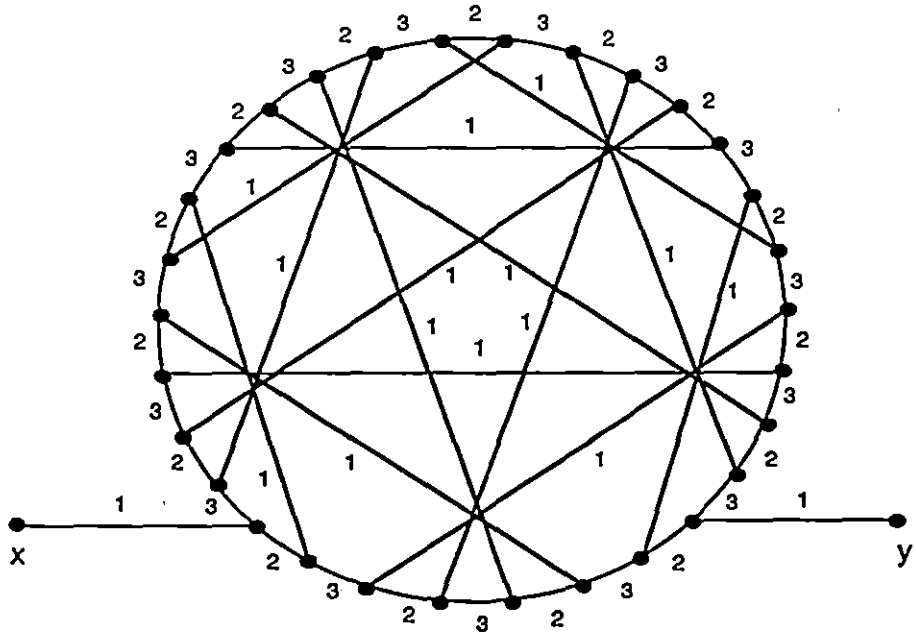


H_6

Figure 2.5. Components H_4, H_5 and H_6.



H_7



H_8

Figure 2.6. Components H_7 and H_8.

From an arbitrary instance G of CHRIND(cubic graph), we construct an instance G' of CHRIND(cubic graph of girth k) for $k = 4, 5, 6, 7, 8$ by replacing each edge of G with the component H_k shown in Figure 2.5 and Figure 2.6.

Notice that H_k is obtained from a k-cage, the girth of H_k is k and the length of the shortest path between vertices x and y is at least k. It follows that G' is a cubic graph of girth k. Figure 2.5 and Figure 2.6 show that H_k for $k = 4, 5, 6, 7, 8$ is 3-colourable. Moreover, by the parity lemma, the two pendants of H_k receive the same colour for any 3-colouring of H_k. Therefore it is easy to see that G is 3-colourable if and only if G' is 3-colourable. The transformation is clearly polynomial time. □

We conjecture that Theorem 2.7 is true for all fixed $k \geqslant 3$.

Conjecture 2.1. CHRIND(graph of girth k) for fixed $k \geqslant 3$ is NP-complete.

2.3. Edge Colouring Line Graphs

In this section, we consider CHROMATIC INDEX restricted to claw-free graphs which are a generalization of line graphs and the line graphs of various graphs.

Definition 2.5. A *claw-free graph* is a graph which does not contain $K_{1,3}$ as an induced subgraph.

Definition 2.6. The *centre* of a claw is the vertex of degree 3 in the claw.

Theorem 2.8. CHRIND(cubic claw-free graph) is NP-complete.

Proof. We present a polynomial time transformation from CHRIND(cubic graph) to CHRIND(cubic claw-free graph). For an arbitrary instance G of CHRIND(cubic graph), we construct an instance G' of CHRIND(cubic claw-free graph) such that G is 3-colourable if and only if G' is 3-colourable.

If G contains no claw as an induced subgraph, then let G' be G. Otherwise, we construct a cubic graph G' by replacing the centre of each induced claw in G with a triangle and linking 3 edges which were incident with the centre of the claw to three distinct vertices of the triangle. Since the number of induced claws in G is at most n, G' can be constructed in time $O(n)$. An example of the transformation is shown in Figure 2.7.

Suppose G is 3-colourable, then edges of each added triangle in G' can be 3-coloured since the three edges incident with it received three distinct colours

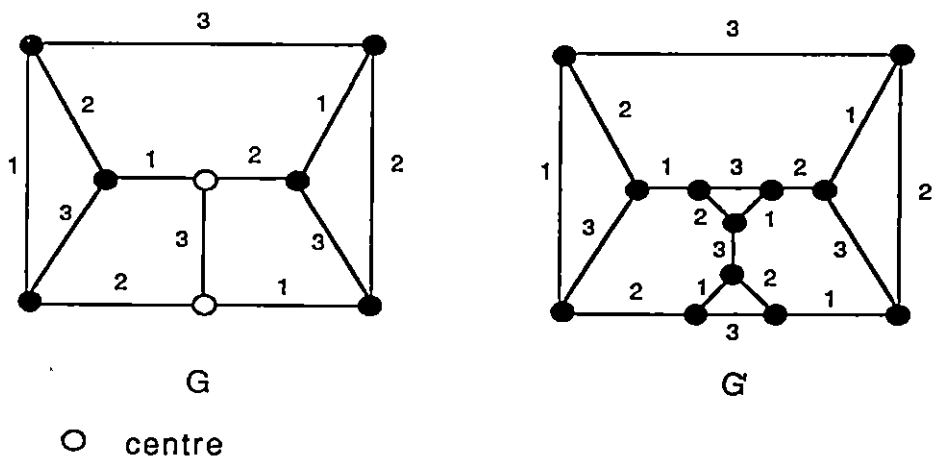


Figure 2.7. Graph G and its corresponding claw-free graph G'.

in the 3-colouring of G. Conversely, if G' is 3-colourable, then we can obtain a 3-colouring of G simply by contracting these added triangles in G' to a single vertex and preserving the colouring of the resulting graph, which is G, since the three edges incident with a triangle must receive distinct colours in any 3-colouring. □

Theorem 2.9. CHRIND(line graph) is NP-complete.

Proof. We present a polynomial time transformation from CHRIND(cubic graph) to CHRIND(line graph of the subdivision of a cubic graph). Let G be a cubic graph and G' be the line graph of the subdivision of G. G' can be constructed by replacing each vertex of G by a triangle and linking 3 edges incident with the vertex to three distinct vertices of the triangle. Figure 2.8 illustrates an example of this transformation and the corresponding 3-colourings.

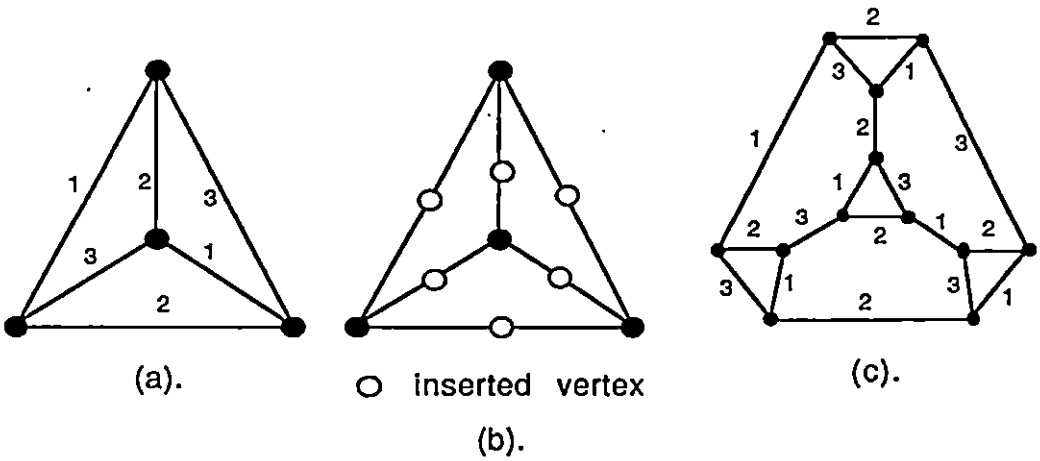


Figure 2.8. (a). Graph G. (b). Subdivision of G. (c). G'.

By an similar argument to that in Theorem 2.8, it is easy to show that G is 3-colourable if and only if G' is 3-colourable. The transformation is clearly polynomial time. □

Note that the subdivision of a cubic graph is a bipartite graph, so we obtain the following corollary.

Corollary 2.2. CHRIND(line graph of bipartite graph) is NP-complete.

Theorem 2.10. CHRIND([$2r-1$]-regular line graph) for fixed r is NP-complete.

Proof. We present a polynomial transformation from CHRIND([$2r-1$]-regular graph) to CHRIND([$2r-1$]-regular line graph). Let G be a [$2r-1$]-regular graph. Replace each vertex of G by a clique on $2r-1$ vertices and link $2r-1$ edges incident with the vertex to $2r-1$ distinct vertices of the clique to form a graph G'. The construction of G' is obviously polynomial time.

It is easy to see that G' is the line graph of the subdivision of G and [$2r-1$] regular. We need to show that G is $2r-1$ colourable if and only if G' is. Note that for any $2r-1$ colouring of K_{2r-1}, the colours of the $2r-1$ pendants of K_{2r-1} are distinct by the parity lemma. Therefore, a $2r-1$ colouring of G can easily be extended to a $2r-1$ colouring of G'. Conversely, a $2r-1$ colouring of G can be derived from a $2r-1$ colouring of G' by contracting each K_{2r-1} in G' to a single vertex and preserve the colouring of the pendants for the edges incident with that vertex. □

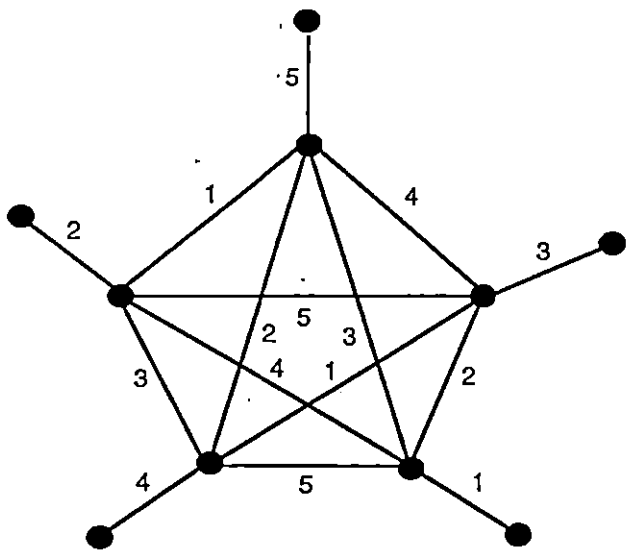


Figure 2.9. Substitute used in the proof of Theorem 2.10.

Now we are going to show that the chromatic index problem restricted to some special classes of line graphs remains NP-complete. The following lemma gives a relation between the colourability of cubic graphs and its line graphs. It was discovered by Jaeger [19], Seymour [29] and Kotzig [22].

Lemma 2.2. Let G be a cubic graph with an even number of edges, and $L(G)$ be its line graph, then G is 3-edge-colourable if and only if $L(G)$ is 4-edge-colourable.

From this lemma, it is easy to see that the NP-completeness of chromatic index problem restricted to some special cubic graphs with an even number of edges immediately implies the NP-completeness of edge colouring the line graphs of this kind of cubic graph.

Theorem 2.11. CHRIND(line graph of claw-free graph) is NP-complete.

Proof. By Lemma 2.2, it suffices to show a polynomial time transformation from CHRIND(cubic claw-free graph) to CHRIND(cubic claw-free graph with an even number of edges). Let G be an instance of CHRIND(cubic claw-free graph), then it must contain a triangle. Therefore, an instance G' of CHRIND(cubic claw-free graph with an even number of edges) can be formed from G by replacing one triangle and its three adjacent edges with the component H as shown in Figure 2.10.

G' has 9 more edges than G, thus it has an even number of edges. Notice that H is 3-colourable and that its three pendants must receive distinct colours in any 3-colouring. It is easy to see that G is 3-colourable if and only if G' is 3-colourable. □

Theorem 2.12. CHRIND(line graph of C_k-free graph) for fixed $k \geqslant 3$ is NP-

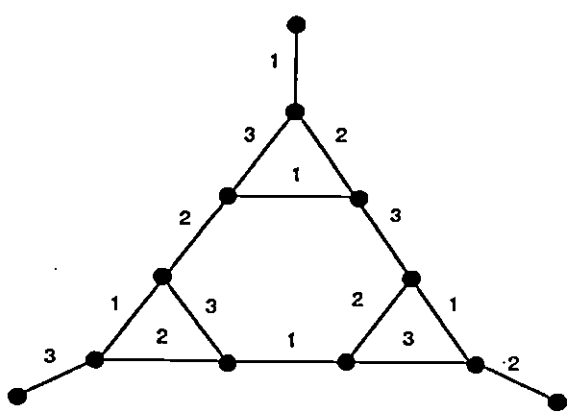


Figure 2.10. A claw-free substitute with 15 edges.

complete.

Proof. It suffices to present a polynomial time transformation from CHRIND(cubic C_k-free graph) to CHRIND(cubic C_k-free graph with an even number of edges) for fixed $k \geqslant 3$.

Let G be a cubic C_k-free graph, where $k \geqslant 3$. If G has an even number of edges, let G' be G. Otherwise, we use components H_k as shown in Figure 2.11 to replace an edge of G and so form a cubic C_k-free graph G' with an even

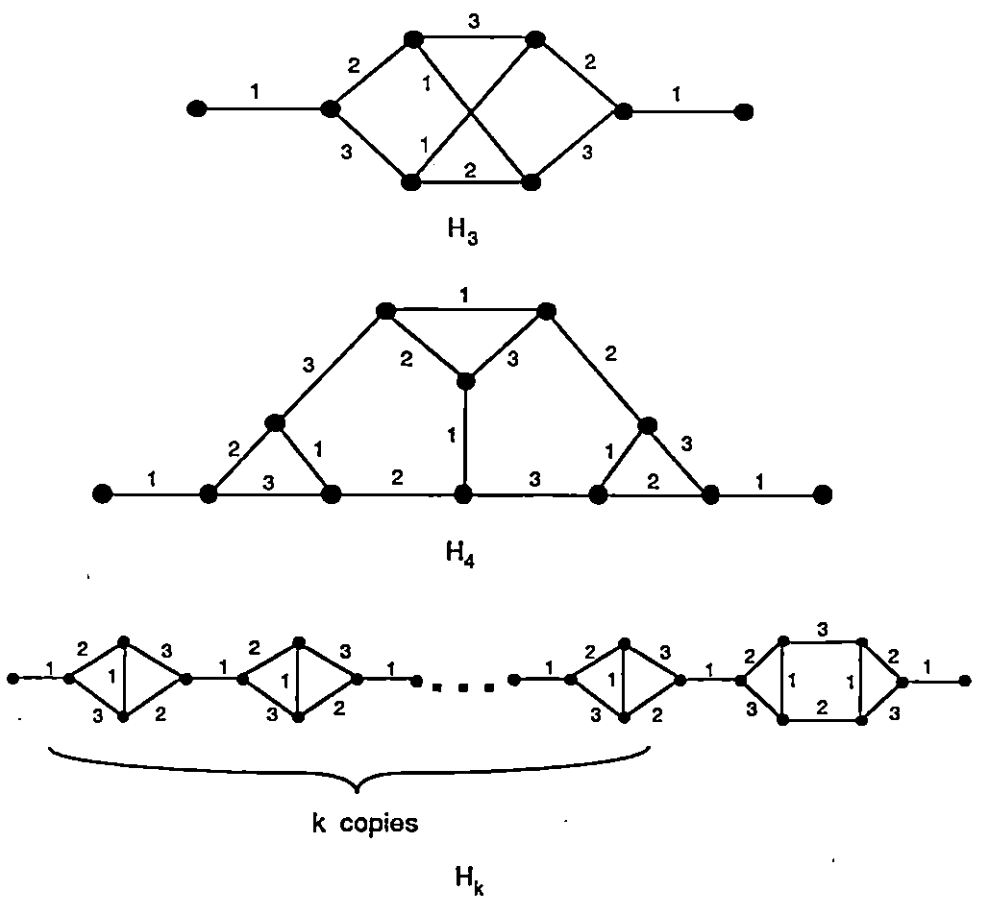


Figure 2.11. Components used in the proof of Theorem 2.12.

number of edges. Since each H_i, for $i \geqslant 3$, contains an even number of edges, then the number of edges in G' is even. It is also easy to see that G' is still C_k-free, so G' is a desired instance.

Clearly, the transformation can be done in time $O(k)$ for $k \geqslant 3$, and thus is a polynomial time transformation. Each component is 3-colourable, see Figure 2.11. By the parity lemma, the two pendants of each component always receive the same colour in any 3-colouring of each component. Therefore it follows immediately that G is 3-colourable if and only if G' is 3-colourable. □

2.4. Edge Colouring Clique-Free Graphs

Definition 2.7. A graph is a *k-clique-free graph* iff it contains no K_k as a subgraph.

Theorem 2.13 CHRIND(k-clique-free graph) for fixed odd k is NP-complete.

Proof. We have already shown that CHRIND(triangle-free graph) is NP-complete, therefore we only need to prove the theorem for $k \geqslant 5$. We present a polynomial time transformation from CHRIND(k-regular graph) to CHRIND(k-regular k-clique-free graph) for odd $k \geqslant 5$.

Given an instance G of CHRIND(k-regular), we use local replacement to get an instance G' of CHRIND(k-regular k-clique-free). We use a construction by Izbicki [18] to define a graph $H_k = (V_k, E_k)$ for $k \geqslant 5$ as our basic component. $H_k = (V_k, E_k)$ for $k \geqslant 5$ is defined as follows:

$V_k = \{x_s, y_t, z_t \mid s = 1,...,k-3, t = 1,...,k\}$

$E_k = \{x_s y_t, y_t y_{t+1}, y_t z_t \mid s = 1,...,k-3, t = 1,...,k\}$.

all indices are taken modulo k.

Note that H_k is k-colourable. One k-colouring of H_k is given as follows:

colour $x_s y_t$ with colour $t + s - 1 \ mod \ k$ for $t = 1,...,k$, $s = 1,...,k-3$.

colour $y_t z_t$ with colour $t + k - 1 \ mod \ k$ for $t = 1,...,k$.

colour $y_t y_{t+1}$ with colour $t + k - 2 \ mod \ k$ for $t = 1,...,k$.

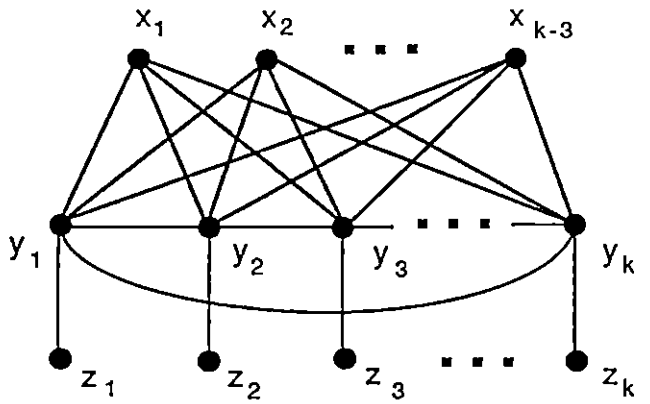
It can be shown that in any k-colouring of H_k, its k pendants $y_t z_t$ must be coloured in k distinct colours [23]. H_5 and a proper 5-colouring of it is shown in Figure 2.12.

Let G be a k-regular graph. If G contains no k-clique as a subgraph, then let G' be G. Otherwise, we construct a sequence of k-regular graphs

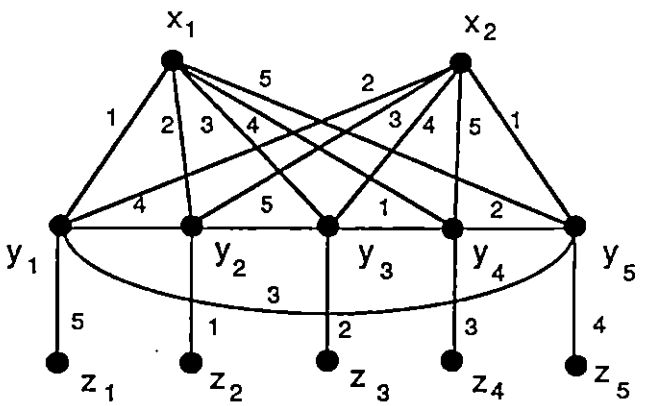
$$G = G_0, G_1, \ldots, G_r = G'$$

such that G_i, $1 \leqslant i \leqslant r$, is constructed from G_{i-1} by replacing a k-clique in G_{i-1} by H_k and linking k pendants of H_k to k distinct vertices of the clique. It is easy to see that G_i can be constructed from G_{i-1} in polynomial time, since a clique of size k can be found in polynomial time for fixed k.

Note that the number of disjoint k-cliques in G_i is at least one less than that in G_{i-1}, and that the number of disjoint k-cliques in G is at most $O(\frac{n}{k})$, so G' can be constructed from G in polynomial time.



(a).



(b).

Figure 2.12. (a) H_k . for $k \geqslant 5$. (b). H_5 and a 5-colouring.

It suffices to show that G_{i-1} is k-colourable if and only if G_i is k-colourable for $1 \leqslant i \leqslant r$. Suppose G_{i-1} is k-colourable, then since k is odd, the k incident edges of a k-clique must be coloured with distinct colours. We can preserve the colouring of these edges for k pendants of H_k and then extend a k-colouring of H_k to a k-colouring of G_i. Conversely, since k-pendants of H_k must receive distinct colours in any k-colouring of H_k, we can preserve the

colouring of these pendants for the edges incident with the clique and extend a k-colouring of the clique to obtain a k-colouring of G_{i-1}. □

2.5. Edge Colouring Complementary Graphs

The following theorems give the complexity of deciding the chromatic index of the complement of some classes of graphs.

Definition 2.8. A graph is a *co-claw-free graph* iff its complement is a claw-free graph.

Lemma 2.3. Triangle-free graphs $\subseteq$ co-claw-free graphs.

Proof. We only need to show that the complement $\overline{G}$ of a triangle-free graph G is a claw-free graph. Suppose $\overline{G}$ contains a claw induced by 4 vertices u, v_1, v_2, v_3, where u is the centre of the claw (Figure 2.13(a)), then v_1, v_2, v_3 induce a triangle in G, since $v_i, v_j \notin \overline{E}$ for $1 \leqslant i < j \leqslant 3$, a contradiction. □

Theorem 2.14. CHRIND(co-claw-free graph) is NP-complete.

Proof. Immediate from Lemma 2.3 and Theorem 2.5. □

Definition 2.9. A graph is co-C_5-free iff its complement is a C_5-free graph.

Lemma 2.4 C_5-free graphs = co-C_5-free graphs.

Proof. Suppose $\overline{G}$ has a 5-cycle induced by v_1, v_2, v_3, v_4, v_5 (Figure 2.13(b)), then v_1, v_3, v_5, v_2, v_4 would be an induced 5-cycle in G. Conversely, if G has a 5-cycle induced by v_1, v_2, v_3, v_4, v_5, then v_1, v_3, v_5, v_2, v_4 would be an induced 5-cycle in $\overline{G}$. □

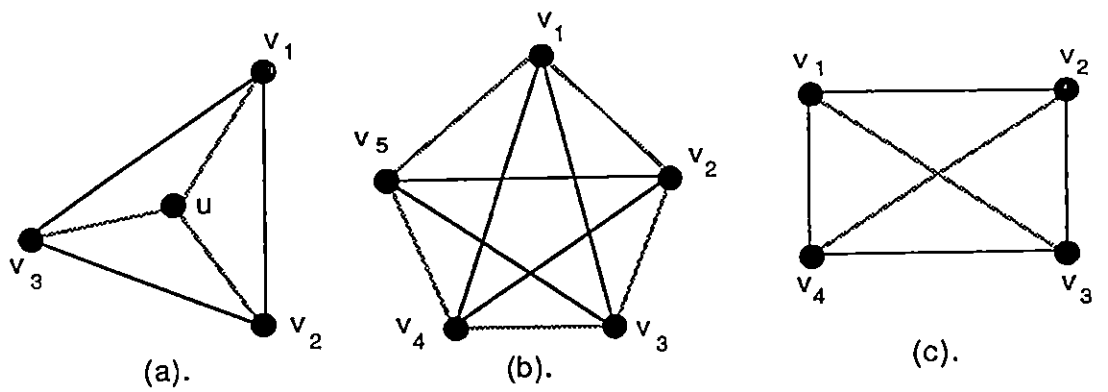


Figure 2.13. Illustration for (a). Lemma 2.3. (b). Lemma 2.4. (c). Lemma 2.5.

Theorem 2.15 CHRIND(co-C_5-free) is NP-complete.

Proof. Immediate from Theorem 2.6 and Lemma 2.4. □

Definition 2.10. A graph is a co-$2K_2$-free graph iff its complement is a $2K_2$-free graph.

Lemma 2.5. co-$2K_2$-free graph = C_4-free graph.

Proof. Suppose G has a 4-cycle induced by v_1, v_2, v_3, v_4 (Figure 2.13(c)), then v_1v_3 and v_2v_4 would induce a $2K_2$ in $\overline{G}$. Conversely, if $\overline{G}$ contains a $2K_2$ induced by v_1, v_2 and v_3, v_4, then v_1, v_3, v_2, v_4 would induce a 4-cycle in G. □

Theorem 2.16. CHRIND(co-$2K_2$-free) is NP-complete.

Proof. Immediate from Theorem 2.6 and Lemma 2.5. □

CHAPTER 3

EDGE COLOURING ALGORITHMS FOR LINE GRAPHS OF TREES AND UNICYCLIC GRAPHS

The NP-completeness of the CHROMATIC INDEX problem does not exclude the existence of polynomial time algorithms for some instances of the problem. In fact, polynomial time algorithms have been found for determining the chromatic index of some restricted graphs, such as trees, unicyclic graphs, [24] [27] , outplanar graphs [28] , series parallel graphs [31] , Halin graphs [30] , bipartite graphs [8] , [12] and, recently, partial k-trees for fixed k [5] . This chapter will investigate the CHROMATIC INDEX problem restricted to complete graphs, line graphs of trees, and line graphs of unicyclic graphs. We will show that these subproblems of CHROMATIC INDEX can be solved in time linear in the number of edges. Moreover, their optimal edge colourings can also be found in time linear in the number of edges.

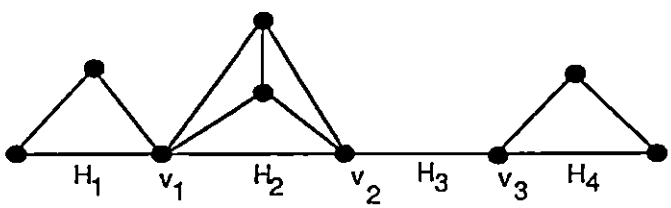
Definitions are given in Section 3.1. In Section 3.2, we present a linear time algorithm for optimally edge colouring complete graphs, which is the kernel of the other linear time algorithms in this chapter. The chromatic index of the line graphs of trees will be determined and a linear time optimal edge colouring algorithm will be given in Section 3.3. In Section 3.4, we present a linear time optimal edge colouring algorithm for line graphs of unicyclic graphs. A remark on the possible extensions of our methods is given in Section 3.5.

3.1. Definitions

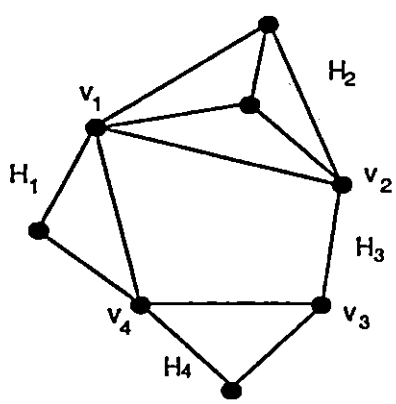
In this section we give definitions which will be used in this chapter. Basic definitions of graph theory can be found in [16]. Some of them are included in the Appendix.

Definition 3.1. A *clique chain* is a sequence of cliques and vertices $H_1 v_1 H_2 v_2 \cdots v_{n-1} H_n$, such that

1. each H_i, $1 \leqslant i \leqslant n$ is a clique;
2. each v_i, called a *common vertex*, $1 \leqslant i \leqslant n-1$ is the only vertex shared by H_i and H_{i+1};



(a)



(b)

Figure 3.1. (a). A clique chain. (b). A clique cycle.

3. H_i and H_j do not share any vertices unless $|i-j|=1$.

It is easy to see that each H_i, $1 \leqslant i \leqslant n$ forms a block and each v_i, $1 \leqslant i \leqslant n-1$ is a cut point.

Definition 3.2 A *clique cycle* is formed from a clique chain by identifying one vertex $u \neq v_1$, in H_1 with one vertex $v \neq v_{n-1}$ in H_n.

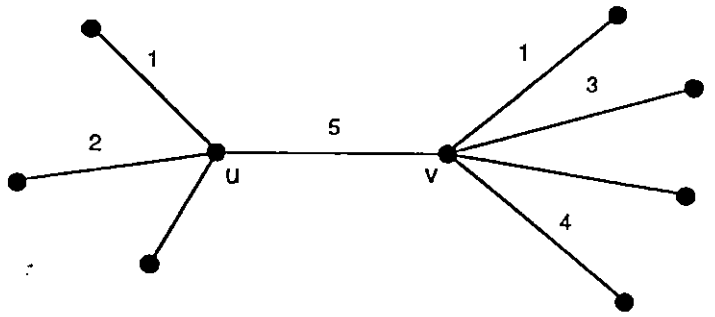
Definition 3.3 The *length* of a clique chain (cycle) is the number of maximal cliques in the clique chain (cycle). A clique chain and a clique cycle are illustrated in Figure 3.1.

Definition 3.4. A *cross edge* is an edge both of whose ends are common vertices.

Definition 3.5. Given a graph $G=(V,E)$ and a set of colours C, a *proper partial edge colouring* of G is a mapping $f: E \rightarrow C$ such that $f(e_1) \neq f(e_2)$ for any two adjacent edges e_1 and e_2. When f is a function, then we call it a *proper edge colouring* of G.

Definition 3.6. Given a proper (partial) edge colouring f of G,

1. the *used colours* at vertex v, denoted U_v, is the set of colours used in f for edges incident with v;
2. the *missing colours* at vertex v, denoted M_v, is the set of colours which are not used in f for edges incident with vertex v;
3. the *missing colours* at edge e, denoted M_e, is the set of colours which are not used in f for edges adjacent to e.



$C = \{1,2,3,4,5\}$, $M_u = \{3,4\}$, $U_u = \{1,2,5\}$,
$M_v = \{2\}$, $U_v = \{1,3,4,5\}$, $M_{uv} = \emptyset$.

Figure 3.2. A partial colouring and missing colours at vertices and edges.

Figure 3.2 illustrates a partial colouring and the meanings of terms defined in Definition 3.6.

3.2. Edge Colouring Complete Graphs

The chromatic index of the complete graph K_n has been determined by several authors using different approaches [2, 3 p. 249, 10 p. 23]. However, these methods do not immediately imply an efficient optimal edge colouring algorithm. Here we give a constructive proof which immediately implies an efficient optimal edge colouring algorithm. In order to implement our method efficiently, we will consider the adjacency matrix of K_n instead of its drawing. Suppose A is the adjacency matrix of some graph G, then a k-colouring of G is equivalent to a function f from each "1" entry of A to $\{1,2, ..., k\}$ such that

$f(a_{i,j}) = f(a_{j,i})$ for $1 \leqslant i, j \leqslant n$, and $f(a_{i,j}) \neq f(a_{i,k})$ if $j \neq k$.
Thus we also call f a k-colouring of A.

Theorem 3.1. The chromatic index of K_n $(n>1)$ is $n-1$ if n is even, and n if n is odd.

Proof. It is obvious that $\chi'(K_n) \geqslant n-1$, since the maximum degree of K_n is $n-1$.

If n is odd, then there are at most $\frac{(n-1)}{2}$ edges which share no common vertex, therefore at most $\frac{(n-1)}{2}$ edges can receive the same colour. This implies that K_n has at most $\chi'(K_n)\frac{(n-1)}{2}$ edges, and hence $\chi'(K_n) \geqslant n$. To prove that $\chi'(K_n) = n$, we will construct an n-colouring of K_n for n odd. Actually, an n-colouring of its adjacency matrix can be constructed as follows:

$$\begin{bmatrix} 1 & 2 & \cdots & n-1 & n \\ 2 & 3 & \cdots & n & 1 \\ . & . & \cdots & . & . \\ n & 1 & \cdots & n-2 & n-1 \end{bmatrix}.$$

Note that $f(a_{i,j}) = [(i+j-2) \bmod n] + 1$, and $f(m_{i,i}) \neq f(m_{j,j})$ for $i \neq j$ in this n-colouring.

If n is even, we show that $\chi'(K_n) = n-1$ by explicitly constructing an $(n-1)$-colouring of A. It is trivial if $n = 2$. If $n > 2$, we can construct an $(n-1)$-colouring f for the adjacency matrix of K_{n-1} as shown above and then move the diagonal to the last row and to the last column of A.

$$\begin{vmatrix} 0 & 2 & \cdots & n-2 & n-1 & 1 \\ 2 & 0 & \cdots & n-1 & 1 & 3 \\ . & . & \cdots & . & . & . \\ n-1 & 1 & \cdots & n-3 & 0 & n-2 \\ 1 & 3 & \cdots & n-4 & n-2 & 0 \end{vmatrix}$$

Since $f(a_{i,i}) \neq f(a_{j,j})$ for all $i \neq j$, all elements in the last row are distinct, and so are the elements in the last column. Hence it is an $(n-1)$-colouring of K_n. □

From the above arguments, an efficient algorithm follows immediately.

Algorithm 3.1 Finding an optimal edge colouring of a complete graph.

Input : The adjacency list of a complete graph G.

Output : An optimal edge colouring of G.

Method : It finds an optimal edge colouring of complete graph G as described in Theorem 3.1.

The linearity of the algorithm is quite evident, since each edge is visited only once.

Theorem 3.2. An optimal edge colouring of a complete graph G can be found in time $O(|V| + |E|)$.

3.3. Edge Colouring Line Graphs of Trees

The *line graph* of G, denoted $L(G)$, is a graph formed in the following way: the vertices of $L(G)$ represent the edges of G, two vertices of $L(G)$ are

adjacent whenever the corresponding edges of G are adjacent. If G is some tree T, then $L(G)$ is the line graph of tree T. A tree and its line graph are illustrated in Figure 3.3.

Though line graphs have more structure than arbitrary graphs, CHROMATIC INDEX remains NP-complete, even for line graphs of bipartite graphs, as we showed in the previous chapter. However, it will be shown in this section that an optimal edge colouring of the line graph of a tree can be found in time linear in the size of the graph.

The following characterization of the line graph of a tree is due to G. T. Chartrand, its proof can be found in [16 p.78] .

Theorem 3.3. A graph G is the line graph of a tree if and only if it is a connected graph which satisfies the following conditions:

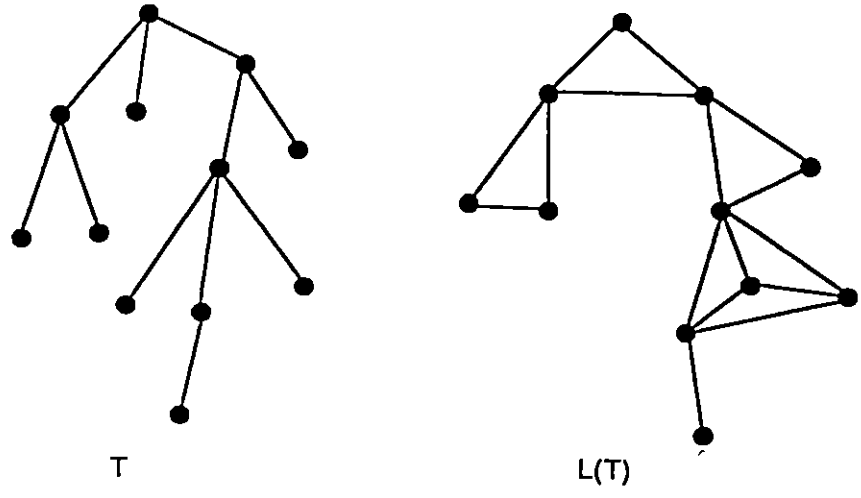


Figure 3.3. A tree T and its line graph $L(T)$.

(a) each block is a complete graph,

(b) each cut point lies on exactly two blocks.

Hence, it is easy to devise an algorithm to recognize the line graphs of trees. We only need to use depth first search algorithm to find blocks of a graph, and then check if each block is a complete graph and if each cut point lies on exactly two blocks. It is apparent that this recognition algorithm takes linear time.

Theorem 3.4. Let G be a line graph of some tree T, and Δ be the maximum degree of G, then the chromatic index of G is given by

$$\chi'(G) = \begin{cases} \Delta+1 & \text{if } T \text{ is an odd star } K_{1,n} \\ \Delta & \textit{otherwise} \end{cases}$$

Proof. It is clear that G is a clique when $T = K_{1,n}$. Hence by Theorem 3.1, $\chi'(G) = \Delta+1$ if n is odd, and $\chi'(G) = \Delta$ if n is even.

Now the proof proceeds by induction on b, the number of blocks of G. We should note that if v is a vertex of degree Δ, then it must be a cut point.

Basis: We should consider $b = 2$. Let K_n, K_m be the two blocks of G and v be the cut point between them. We need to consider three cases.

Case 1. even(n) and even(m).

We use colours $\{1, 2, ..., n-1\}$ for K_n and $\{n, ..., n+m-2\}$ for K_m.

Case 2. (even(n) and odd(m)) or (odd(n) and even(m))

Without loss of generality, we may assume that n is odd. We use colours

$C_1 = \{1, 2, .., n\}$ for K_n, and $\{\alpha, n+1, ..., n+m-2\}$ for K_m, where $\alpha \in C_1$ and is not used for any edges of K_n which are incident with v.

Case 3. odd(n) and odd(m).

We use $C_1 = \{1, 2, ..., n-1, \alpha\}$ and $C_2 = \{\beta, n, ..., n+m-2\}$ for K_n and K_m respectively, where $\alpha \in C_2$, $\beta \in C_1$ and neither of them are used for any edges which are incident with v.

Observe that the maximum degree of G is $n+m-2$ and we only use $n+m-2$ colours to properly colour G, therefore the theorem is true for $b = 2$. Figure 3.4 illustrates a proper colouring for each of the above three cases.

Hypothesis: Suppose the theorem is true for all $2 \leqslant b \leqslant k$.

Induction: Let G be the line graph of a tree with $k+1$ blocks, and K_n be an end block of G. Let G' be the graph obtained from G by deleting K_n, and v be the cut point which separates K_n from G. By Theorem 3.3, it is clear that G' is still the line graph of a tree. By the induction hypothesis, G' is Δ'-colourable, since G' has k blocks.

Since $deg(v) \leqslant \Delta$, there are at least $n-1$ colours missing at vertex v in any Δ'-colouring of G'. If n is even, then we can obtain a Δ-colouring of G by using these $n-1$ colours for K_n. If n is odd, we can use these $n-1$ colours plus a colour α which is used for some edge incident to v in G' to colour K_n in such a way that α is not used for any edges in K_n which are incident with v. Hence

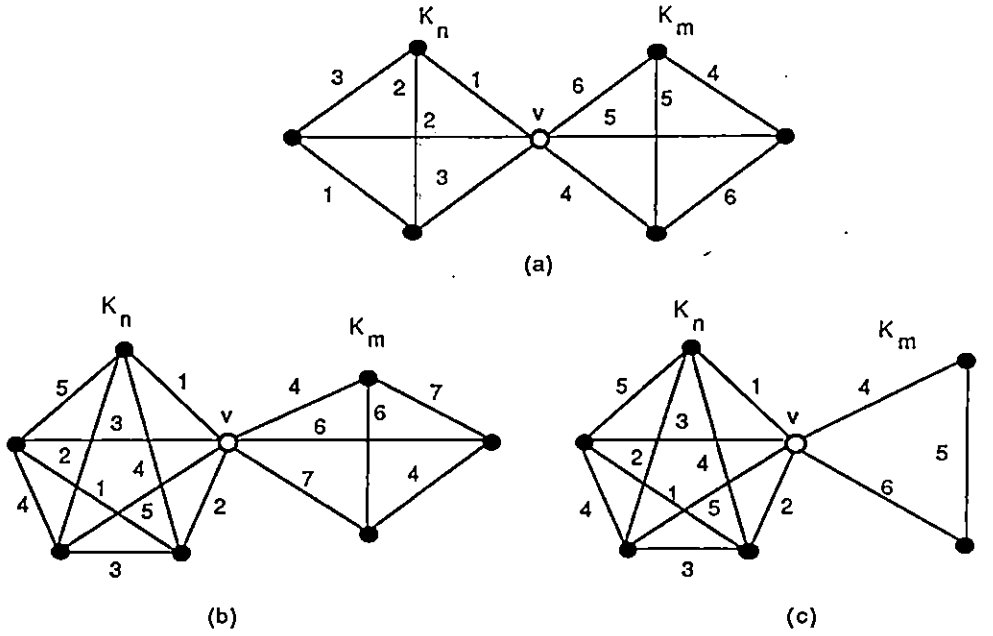


Figure 3.4. Optimal edge colourings of clique chain $K_n\, vK_m$. (a). Both n and m are even. (b). n is odd, m is even. $\alpha = 4$. (c). Both n and m are odd. $\alpha = 5$, $\beta = 4$.

G is always Δ-colourable when it has more than two blocks. □

Corollary 3.1. The chromatic index of a clique chain of length greater than 1 is equal to its maximum degree.

Proof. Since such a clique chain is the line graph of a tree, it follows from Theorem 3.4 immediately. □

The proof of Theorem 3.4 implies a method for finding an optimal edge colouring of the line graph of a tree. We can use depth first search to recognize blocks of the line graph of a tree, and then colour each block separately in such a way that all edges incident with each cut point receive distinct colours.

Obviously, the colouring procedure can be embedded into a depth first search algorithm. However, in order to clarify the colouring procedure, we separate it from the depth first search algorithm.

Let G be the line graph of a tree T, let $B(G)$ be the block graph of G, and for each vertex v in $B(G)$ let B_v be its corresponding block in G. Clearly, $B(G)$ is a tree, since each cut point of G lies on exactly two blocks. Let v be an arbitrary vertex v in $B(G)$. Then each connected component of $B(G)-v$ is a tree. On the other hand, each connected component of $G-B_v$ is the line graph of a tree and corresponds to a connected component of $B(G)-v$. We let H_i represent each connected component in $G-B_v$, and $B(H_i)$ be its

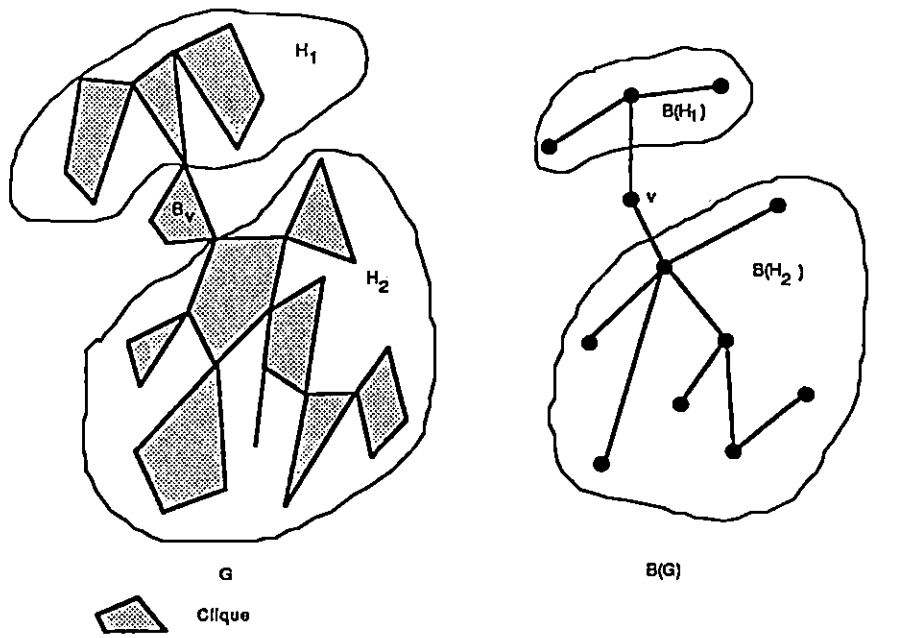


Figure 3.5. A line graph of tree G and its block graph $B(G)$.

corresponding connected component in $B(G)-v$. An example is shown in Figure 3.5.

The following procedure *ColourLT* finds an optimal colouring of the line graph of a tree whose block graph contains vertex v. It first finds an optimal colouring for the corresponding block B_v of v, and then recursively colours each connected component H_i of $G - B_v$ whose block graph $B(H_i)$ contains v_i, where v_i is a vertex of $B(G)$ which is adjacent to v.

Algorithm 3.2 Finding an optimal edge colouring of the line graph of a tree.

Input : The adjacency list of the line graph of a tree G.

Output : An optimal edge colouring of G.

```
Procedure ColourLT (ColSet , v );
    ColourBlock ( |V (B_v )| , ColSet , B_v , v );
    if v has adjacent vertices then
      for each adjacent vertex v_i of v do
        Let x be the cut point separating B_v and H_i ;
        Let M be the set of missing colours at vertex x ;
        Colour (M , v_i );
      endfor
    endif
end ColourLT ;

begin
   Construct the block graph B (G ) of G ;
   Let ColSet := { 1, 2, ..., Δ(G ) + 1 };
   Let v be a vertex of B (G );
   ColourLT ( ColSet , v )
end.
```

In this algorithm, *ColourBlock* (*VertexNum* , *ColSet* , *Block* ,*Vertex*), where *ColSet* contains the missing colours at vertex *Vertex* , optimally colours block

B_v in such a way that the set of colours used for edges in *Block* which are incident with *Vertex* are a subset of *ColSet*, thus ensuring that all edges incident with *Vertex* receive distinct colours. Notice that since each block is a clique, *Block* needs only at most |*ColSet*| colours if the number of vertices in *Block* is even, but one extra colour otherwise. In former case, we only need to call the optimal colouring procedure for complete graphs mentioned in the previous section. In the latter case, however, we need one extra colour and use this colour only for edges not adjacent to *Vertex*. Observe that when a clique is coloured by procedure *ColourClique* the first colour in *ColSet* is always missing at the first vertex, therefore we can also use procedure *ColourClique* if we let *Vertex* be the first vertex in the vertex list of the clique and let the extra colour be the first colour in the missing colour list. This procedure is specified as follows:

```
Procedure ColourBlock ( VertexNum , ColSet , Block , Vertex );
  if even (VertexNum ) then ColourClique ( VertexNum , ColSet , Block )
  else
    Let α be a colour not in ColSet ;
    Swap vertex Vertex with the first vertex in Block ;
    ColSet [ VertexNum + 1 ] := ColSet [1];
    ColSet [1] := α;
    ColourClique ( VertexNum , ColSet , Block )
  endif
end ColourBlock ;
```

The correctness of the algorithm is implied by the proof of Theorem 3.4. To see the linearity of the algorithm, we note that the optimal colouring of each block of the graph can be found in linear time and the total time spent in

calculating the missing colour is also linear in the size of the graph. Thus we have the following result.

Theorem 3.5. An optimal edge colouring of a line graph of a tree can be found in time $O(|V| + |E|)$.

3.4. Edge Colouring Line Graphs of Unicyclic Graphs

A *unicyclic graph* is a connected graph which contains exactly one cycle. It can be constructed from a tree by adding one edge between two vertices in the tree. Figure 3.6 illustrates a unicyclic graph and its line graph. Necessary and sufficient conditions for a graph to be the line graph of a unicyclic graph are given by Theorem 3.6.

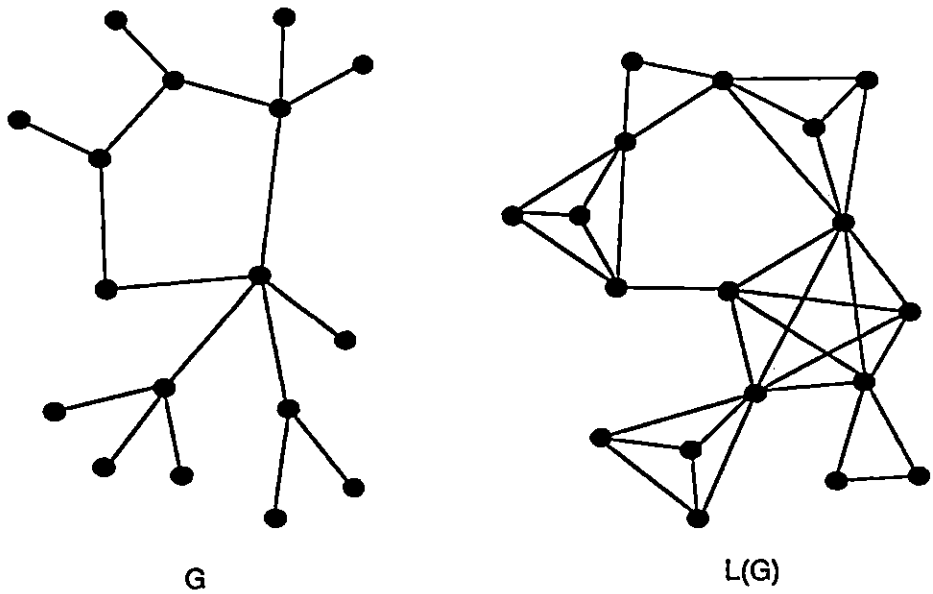


Figure 3.6. A unicyclic graph G and its line graph $L(G)$.

Theorem 3.6. A graph G is the line graph of a unicyclic graph if and only if it is either a cycle or a connected graph satisfying all the following conditions:

(a) one block is a clique cycle which is not a cycle;

(b) all remaining blocks are complete graphs;

(c) each cut point lies on exactly two blocks;

(d) each cut point in the clique cycle is not a common vertex between two cliques in the clique cycle.

Proof. Suppose G is the line graph of a unicyclic graph H. If H is a cycle then so is G. Otherwise, let $e = uv$ be an edge on the unique cycle of H. In this case, $T = H - e$ is a tree, and its line graph $L(T)$ satisfies the conditions of Theorem 3.3. Note that there is a unique clique chain in $L(T)$ whose two end cliques contain u and v respectively. G can be constructed from $L(T)$ by adding a vertex e to $L(T)$ and linking it to every vertex in the two end cliques in the unique clique chain. It is now easy to see that this unique clique chain plus the added edges introduces a block which is a clique cycle. This confirms condition (a). Conditions (b) and (c) are satisfied since we do not change any other blocks. Condition (d) is also satisfied because each common vertex in the clique cycle was a cut point in $L(T)$, therefore it could not lie on more than three cliques. This proves the necessity of the condition.

To prove the sufficiency of the conditions, let G be a graph satisfying all four conditions. Consider the graph $G' = G - u$ where u is a common vertex between two cliques in the unique clique cycle in G. G' is still connected since

u is not a cut point. Each block of G which was a clique is still a block of G', whereas the block which was a clique cycle of G now becomes a clique chain and therefore each block of it is also a clique. Note that each cut point of the clique chain was a common vertex of the clique cycle, so that each cut point of G' lies on exactly two blocks. Hence, $G - u$ is a line graph of some tree T. Observe that a star in a line graph $L(H)$ corresponds to a single edge in H, therefore the removed vertex u together with its incident edges corresponds to an edge in H. Hence, G is a line graph of some tree T plus an edge, which is just a unicyclic graph. □

From the above theorem, it is also easy to recognize the line graph of a unicyclic graph. We already know how to recognize blocks and cliques, thus we only need to demonstrate how to recognize a clique cycle CC. Observe that a vertex v of maximum degree in a clique cycle must be a common vertex. Therefore we can construct $CC' = CC - v$ and check if CC' is a clique chain, which can be done easily as described in the previous section. Finally, we check if v is adjacent to all vertices in the two end cliques in the clique chain. It is not difficulty to see that this recognition takes linear time in the size of CC, and so the whole graph can be recognized in linear time.

Since all line graphs of unicyclic graphs contain a clique cycle as a subgraph, we will determine the chromatic index of a clique cycle first.

Theorem 3.7. Let G be a clique cycle, and Δ be its maximum degree. Then the chromatic index of G is given by

$$\chi'(G) = \begin{cases} 3 & \text{if } G \text{ is an odd cycle} \\ \Delta & \textit{otherwise} \end{cases}$$

Proof. Let G be a clique cycle of length n. If G is a cycle, then its chromatic index is either 2 or 3 according to whether n is even or odd. Otherwise, let u be a vertex of degree Δ, which must be a common vertex between two adjacent cliques H_1 and H_2. Without loss of generality, we may assume that $|V(H_1)| \geqslant |V(H_2)|$, then $|V(H_1)| \geqslant 3$ since G is not a cycle. Since a clique cycle contains at least three maximum cliques, H_1 must be adjacent to another clique, say H_3; we let the common vertex between H_1 and H_3 be v. Clearly, $|V(H_2)| \geqslant |V(H_3)|$, since u is a vertex of maximum degree. Then $G - H_1$ is a clique chain of length greater than 1. Let Δ' be the maximum degree of $G - H_1$, then $G - H_1$ is Δ'-chromatic by Corollary 3.1. Since $\Delta' \leqslant \Delta$, a Δ'-colouring of $G - H_1$ gives a partial colouring of G in which Δ' colours are used and all cliques except H_1 are properly coloured. Let $C = \{1, 2, ..., \Delta\}$ be the colour set used for G, then we have

$$|M_u| + |M_v| = [\Delta - (|V(H_3)| - 1)] + [\Delta - (|V(H_3)| - 1)] \qquad (1)$$

since $|M_v| = \Delta - (|V(H_3)| - 1)$ and $|M_v| = \Delta - (|V(H_3)| - 1)$. We also have that

$$|M_u| + |M_v| \geqslant 2(|V(H_1)| - 1) \geqslant [|V(H_2)| - 1] + [|V(H_3)| - 1] \qquad (2)$$

since H_1 is the biggest clique in the clique cycle.

We will show that this partial colouring of G can always be extended to a proper Δ-colouring of G. Observe that in order to give the cross edge of H_1 a proper colouring by extending the partial colouring of G, we require that $M_u \cap M_v \neq \varnothing$. We show that this condition can always be achieved by rearranging the partial colouring of G.

Suppose $M_u \cap M_v = \varnothing$, then we must have

$$|M_u| + |M_v| \leqslant |C| = \Delta \tag{3}$$

Combining with (1), we obtain that

$$\Delta \geqslant (|V(H_2)| - 1) + (|V(H_3)| - 1); \tag{4}$$

combining with (2), we obtain that

$$\Delta \leqslant (|V(H_2)| - 1) + (|V(H_3)| - 1) \tag{5}$$

Therefore we have

$$\Delta = (|V(H_2)| - 1) + (|V(H_3)| - 1) \tag{6}$$

On the other hand,

$$\Delta = (|V(H_1)| - 1) + (|V(H_2)| - 1). \tag{7}$$

since u is the vertex of maximum degree. Combining (6) and (7), we obtain that $|V(H_3)| = |V(H_1)|$. Similarly, we can derive that $|V(H_2)| = |V(H_1)|$. Thus $|V(H_1)| = |V(H_2)| = |V(H_3)| \geqslant 3$, since G is not a cycle. Therefore H_3 has at least one edge vw which is incident with v but is not a cross edge. Notice that edge vw is adjacent to $2|V(H_3)| - 4$ coloured edges in H_3 and $|C| \geqslant 2|V(H_3)| - 1$, we have that $|M_{vw}| \geqslant 1$. So let α be the colour of edge vw and $\gamma \in M_{vw}$, and change the colour of vw to γ. This gives a new partial colouring of G, and the missing colours at vertices u

and v have exactly one colour α in common in this partial colouring. Thus we only need to consider the case that $M_u \cap M_v \neq \emptyset$.

Since $M_u \cap M_v \neq \emptyset$, we let $\alpha \in M_u \cap M_v$. Two situations need to be considered depending on whether $|V(H_1)|$ is even or odd.

Case 1. $|V(H_1)|$ *is even.* Since H_1 is always $|V(H_1)| - 1$ colourable, we can use M_u to colour H_1 in such a way that α is used for the cross-edge of H_1. Certainly, colour conflict may happen at vertex v. Therefore we need to recolour those edges of H_1 which are incident with v and conflict with H_3 at vertex v. Notice that because $|M_v| \geq |M_u|$, for each of these edges, we can always change its colour to a distinct colour in M_v and hence get a proper colouring for G.

Case 2. $|V(H_1)|$ *is odd*. In this case, if $M_u \neq M_v$, then $M_v - M_u \neq \emptyset$, since $|M_v| \geq M_u$. We can choose a colour $\beta \in M_v - M_u$ and colour H_1 in such a way that α is used for the cross edge and β is not used for any edges of H_1 which are incident with u. We then recolour those edges of H_1 which are incident with v and have colour conflict with H_3 at vertex v. The recolouring procedure is the same as that in the previous case.

If $M_u = M_v$, then $|V(H_2)| = |V(H_3)|$. Suppose $|V(H_3)| \geq 3$, then H_3 has at least one non-cross edge vw. As we argued before, we can change the colour of vw to a missing colour at edge vw and obtain a new partial colouring of G. It is clear that under this new partial

colouring, M_u is neither disjoint from nor the same as M_v, since $|V(H_1)| \geqslant 3$. Then we can apply the argument in the previous paragraph to obtain a Δ-colouring of G.

Finally, if $|V(H_2)| = |V(H_3)| = 2$, it suffices to show that the only edge of H_3 can always be changed to a new colour without affecting the colour of the only edge of H_2. Consider the other neighbour clique, denoted H_4, of H_3, and let w be the common vertex between H_3 and H_4. Suppose $|V(H_4)| < |V(H_1)|$, then there exists a missing colour γ at vertex w, thus we can change the colour of vw to γ. Otherwise, since $|V(H_4)| = |V(H_1)|$, H_4 can not be identical to H_2, and we can rearrange the colouring of H_4 to get the colour of vw changed. This can be done easily, since the missing colour at each vertex of an odd clique is distinct.

Hence in all cases except G being an odd cycle, Δ colours are sufficient. □

In the above theorem, we showed constructively that an optimal colouring of a clique cycle can be found by removing one clique from the clique cycle, optimally colouring the remaining clique chain and then extending the partial colouring to an optimal colouring of the clique cycle. Therefore, we can use this method to devise an optimal edge colouring algorithm for a clique cycle.

We can scan the whole adjacency list of G to locate a vertex of maximum degree Δ. Suppose w is a vertex of degree Δ, then we can construct $G - w$. From $G - w$, we can recognize all blocks in $G - w$ and hence H_1, H_2, H_3.

Therefore, we can derive graph $G - H_1$ and its block graph. Since $G - H_1$ is a line graph of some tree, we can use the procedure *ColourLT* in the previous section to obtain an optimal edge colouring of it. Finally, we extend the colouring of $G - H_1$ to a proper colouring of G according to rules described in the proof of Theorem 3.7. The following procedure *ColourCC* finds an optimal edge colouring for a clique cycle which is not a cycle.

```
Procedure ColourCC(ColSet , G );
    Find H1, H2, H3 and construct G − H1;
    Let u , v be common vertices between H1, H2 and H1, H3 respectively;
    ColourLT(ColSet , G − H1);
    Let Mu and Mv be the set of missing colours at u and v respectively;
    if Mu ∩ Mv = ∅
      Recolour H3 as described in Theorem 3.7;
      Compute new Mv ;
    endif;
    if H1 is an even clique then
      Let α be a colour in Mu ∩ Mv ;
      Colour H1 properly as described in Theorem 3.7;
    else if Mu = Mv
            then if V(H3) ⩾ 3 then
                   Recolour H3 as described in Theorem 3.7;
                   Compute new Mv ;
                 else
                   Let H4 be the adjacent clique of H3;
                   Let x be the common vertex between H3 and H4;
                   Recolour H4 as described in Theorem 3.7;
                   Let γ be a missing colour at vertex x ;
                   Colour edge vx by γ ;
                   Compute new Mv ;
                 endif;
         Let α be a colour in Mu ∩ Mv ;
         Colour H1 properly as described in Theorem 3.7;
    endif
end ColourCC ;
```

Observe that an optimal edge colouring of $G - H_1$ can be found in linear time, and recolouring occurs only in H_i, $1 \leqslant i \leqslant 4$. However, the recolouring process takes time proportional to the number of edges in these cliques. Since each edge in the clique cycle is visited only a constant number of times, we have the following theorem:

Theorem 3.8. An optimal edge colouring of a clique cycle can be found in time $O(|V| + |E|)$.

By Theorem 3.6, we know that each block of the line graph of a unicyclic graph is a complete graph except for one which is a clique cycle, therefore the chromatic index of each block is easy to determine. We now show that an optimal colouring of G can be derived from an optimal colouring of each block. Therefore we can determine the chromatic index of the line graph of a unicyclic graph as stated by Theorem 3.8.

Theorem 3.9. Let G be the line graph of a unicyclic graph H, and Δ be the maximum degree of G, then the chromatic index of G is given by

$$\chi'(G) = \begin{cases} 3 & \text{if } H \textit{ is an odd cycle} \\ \Delta & \textit{otherwise} \end{cases}$$

Proof. If $H = C_n$ and n is odd, then G is also C_n. Hence it requires 3 colours. Otherwise, we use induction on the number of blocks in G.

Basis: If G has only one block, then it must be a clique cycle. By Theorem 3.7, $\chi'(G) = \Delta$, since G is not an odd cycle.

Hypothesis: Assume the theorem is true for $1 \leqslant b \leqslant k$, where b is the number of blocks in G.

Induction Step: Suppose G has $k+1$ blocks, then the block which is a clique cycle is not a cycle by Theorem 3.5. Let K_r be an end block of G. Let $G' = G - K_r$, and the maximum degree of G' be Δ'. It is clear from Theorem 3.5 that G' is the line graph of a unicyclic graph with k blocks. Therefore, by the induction hypothesis, $\chi'(G') = \Delta' \leqslant \Delta$. Let u be the cut point between G' and K_r, and let $C = \{1,2,...,\Delta\}$. This Δ'-colouring of G' gives a partial colouring of G such that $|M_v| \geqslant r - 1$. If r is even, then obviously M_v is sufficient to colour K_r properly. Otherwise, we can use $M_v \cup \{\alpha\}$, where $\alpha \in U_v$, to colour K_r in such a way that α is not used for any edges in K_r which are incident with v. Hence we get a Δ colouring of G. □

From the structure of the line graph of a unicyclic graph, we can see that its block graph is also a tree. Therefore, a method similar to edge colouring the line graphs of trees can be used to find an optimal colouring of the line graph of a unicyclic graph. We will choose the clique cycle as the first block to be coloured. The algorithm is as follows:

Algorithm 3.3 Finding an optimal edge colouring of the line graph of a unicyclic graph.

Input : The adjacency list of the line graph of a unicyclic graph G.

Output : An optimal edge colouring of G.

```
begin
    if G is a cycle then colour it properly
     else
      Construct the block graph B(G) of G;
      Let C = {1, 2, ..., Δ} be colour set;
      Let v be the vertex of B(G) corresponding to the clique cycle;
      ColourCC( C, B_v );
      if v has adjacent vertices then
       for each adjacent vertex v_i do
         Let a_i be the cut point separating B_v and H_i;
         Let M_i be the set of missing colours at vertex a_i;
         ColourLT( M_i, v_i )
       endfor
      endif
    endif
end.
```

It is evident that this algorithm takes linear time, since each edge in the graph is visited a constant number of times.

Theorem 3.10. An optimal colouring of the line graph of a unicyclic graph can be found in time $O(|V| + |E|)$.

3.5. Summary

In this chapter, we discussed the problem of finding an optimal colouring of complete graphs, line graphs of trees, and line graphs of unicyclic graphs. We presented linear time algorithms for optimally edge colouring each of these three classes of graphs.

It is interesting to note that a good optimal colouring algorithm for complete graphs plays a central role in our algorithms. We break a graph into cliques whose optimal colourings are easy to determine, then we rearrange these optimal colourings to get an optimal colouring of the entire graph. We suggest

that this kind of technique might be applied for finding an optimal colouring of those graphs in which an optimal colouring of each block is easy to find.

Observe that every graph is the union of some clique cycles, therefore an interesting question is to ask whether our linear optimal edge colouring clique cycle algorithm can be applied for some special graphs by breaking these graphs into clique cycles.

CHAPTER 4

EDGE COLOURING CUBIC GRAPHS

In this chapter, we consider the edge colouring of cubic graphs. We define a cubic graph to be a loopless graph which is regular of degree 3 and may have multiple edges. We call an edge of multiplicity i an i-edge. It was shown by Holyer [17] that the chromatic index problem is NP-complete for cubic graphs. Therefore it is unlikely that fast algorithms exist for finding an optimal edge colouring of an arbitrary cubic graph. We will present two algorithms in this chapter. In Section 4.1, a linear time approximation edge colouring algorithm is presented which uses at most one extra colour. In Section 4.2, we present an algorithm for determining the chromatic index of general cubic graphs and give an upper bound for the algorithm.

4.1. An Approximate Edge Colouring Algorithm

Though it is unlikely that polynomial time optimal edge colouring algorithms exist for an arbitrary graph, even for an arbitrary cubic graph, efficient algorithms have been proposed for finding a "nearly optimal" edge colouring of an arbitrary graph. In fact, the constructive proof of Vizing's theorem suggests a polynomial time algorithm which uses at most one extra colour to properly colour an arbitrary graph. Several other approximation algorithms have been found, see [1, 13, 34]. Nevertheless, all these algorithms will take more than linear time even if the graph is cubic. In this section, we

present a linear time edge colouring algorithm which properly colours an arbitrary cubic graph with at most four colours.

Most approximation edge colouring algorithms require changing colours which were previously assigned to some edges in order to obtain a proper colouring. However, in the case of cubic graphs, we will show that we can avoid recolouring edges.

The principle of the algorithm is to reduce a cubic graph G to a smaller cubic graph G' such that a four colouring of G' can be extended to a four colouring of G efficiently. We will see that such a reduction always exists unless G consists of only a 3-edge.

Theorem 4.1. For any loopless cubic graph G which is not a 3-edge, there exists a loopless cubic graph G' with fewer vertices such that a four edge colouring of G' can be extended to a four edge colouring of G in constant time.

Proof. We pick an arbitrary edge e in G. It is either a 3-edge, or a 2-edge, or a 1-edge. We consider reductions for these three cases.

Case 1. e is a 3-edge.

$G' = G - e$. Since e is a component of G, G' is obviously a loopless cubic graph. Since a colouring of e is independent of G', by assigning an arbitrary three colours to e and preserving the four colouring of G' we obtain a four colouring of G.

Case 2. e is a 2-edge.

Suppose $e = uv$ is the 2-edge under consideration. There are two subcases to be considered.

Case 2.1. u and v are both adjacent to a vertex x.

$G' = G - \{x, w\} + uw_1 + vw_2$, where w is a vertex adjacent to x, and w is adjacent to, not necessarily distinct, vertices w_1 and w_2. It is easy to see that G' is cubic and loopless. In order to extend a four colouring of G' to a four colouring of G, we should consider two situations depending on whether uw_1 received the same colour as vw_2 in G' or not. The corresponding four colourings of G and G' for these two cases are illustrated in Figure 4.1. Notice that even if w_1 is identical to w_2, the above colouring scheme works as well.

Case 2.2. u and v are adjacent to two distinct vertices

$G' = G - \{u, v\} + u_1v_1$, where u_1 and v_1 are two distinct vertices adjacent to u and v respectively. It is clear that G' is a loopless cubic graph. A four colouring of G can be obtained from a four colouring of G' by assigning edges uu_1 and vv_1 the same colour as u_1v_1 in G', and any two of the remaining three colours to the 2-edge uv.

Case 3. e is a 1-edge.

Let $e = uv$ be the 1-edge under consideration. We assume that e is adjacent to

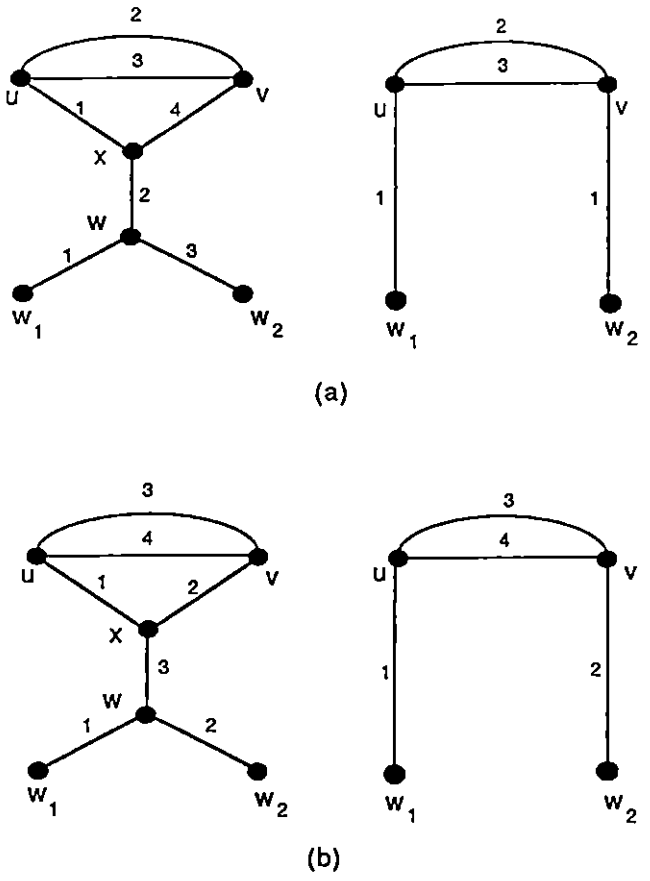


Figure 4.1. Corresponding four-colouring between G and G' in Case 2.1.
(a) uw_1 and vw_2 receive the same colour. Assume $M_{w_2} = \{3\}$.
(b) uw_1 and vw_2 receive different colours.

no 2-edges, since otherwise we can choose one of its adjacent 2-edge as the edge under consideration, which is considered already in Case 2. We should consider two subcases.

Case 3.1. u and v are both adjacent to a vertex w.

$G' = G - \{u, v\} + wu_1 + wv_1$, where u_1 and v_1 are two vertices, not necessary distinct, adjacent to u and v respectively. Since w is different from u_1 and v_1, no loop will be added into G', thus G' is a loopless cubic

graph. A four colouring of G can be obtained from a four colouring of G' as illustrated in Figure 4.2.

Case 3.2. u and v are adjacent to four distinct vertices.

$G' = G - \{u, v\} + u_1v_1 + u_2v_2$, where $u_i, i = 1, 2$ are two distinct vertices adjacent to u and $v_i, i = 1, 2$ are two distinct vertices adjacent to v. Clearly, G' is a loopless cubic graph. The four colouring of G corresponding to a four colouring of G' is illustrated in Figure 4.3.

Finally, notice that for each case, the number of edges that need to be accessed is bounded above by a constant. Therefore it takes only constant time to construct G' from G and to obtain a four colouring of G from a four colouring of G'. □

From the above theorem, we see that the problem of finding a four colouring of a graph G can be reduced to the same problem on a smaller graph

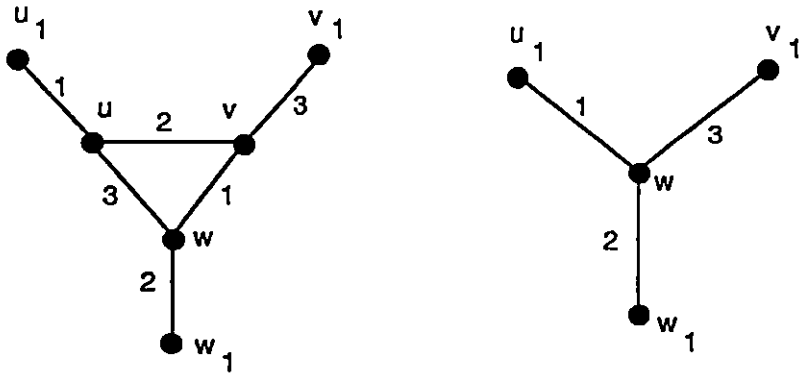


Figure 4.2. Corresponding colouring between G and G' in Case 3.1.

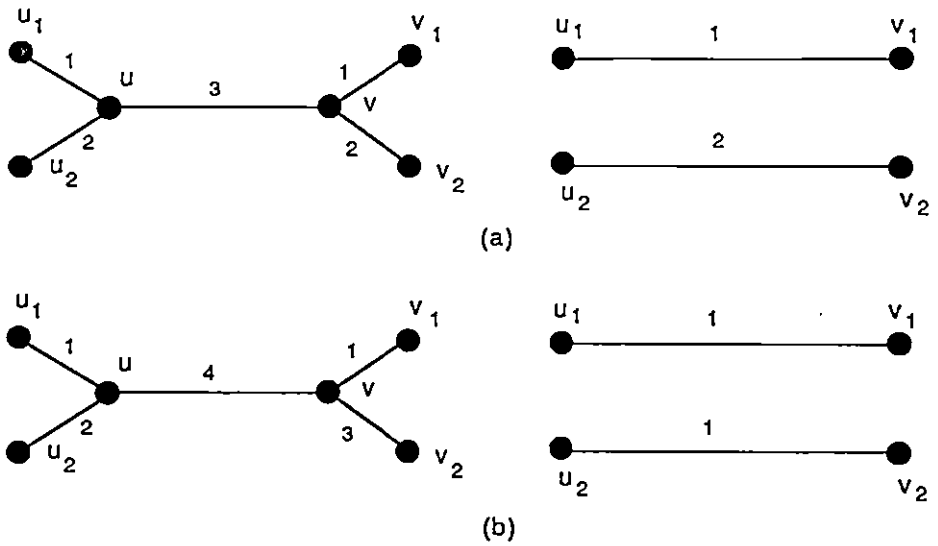


Figure 4.3. Corresponding four-colouring between G and G' in Case 3.2.
(a). u_1v_1, u_2v_2 receive different colours.
(b). u_1v_1, u_2v_2 receive the same colour. Assume $M_{u_2} = \{2\}$, $M_{v_2} = \{3\}$.

G'. A recursive algorithm follows immediately. It recursively reduces a loopless cubic graph until it becomes a 3-edge. This is coloured arbitrarily with three colours and the reinstated edges are four coloured properly as indicated in the proof of Theorem 4.1.

Algorithm 4.1 Finding a four colouring of a loopless cubic graph.

Input : The adjacency list of G.

Output : A proper edge colouring of G which uses at most four colours.

Method : A single call to the procedure *Colour*(G) given below. *Colour*(G) recursively calls itself to reduce the graph to a 3-edge. Reinstated edges are coloured properly.

The procedure *Colour* (G) is as follows:

```
Procedure Colour ( G );
 If G is empty
   then return
   else
       Choose an edge e from G;
       If e is adjacent to a 2-edge e1 then e := e1;
       case
       1. e is a 3-edge:
           Construct G' as specified in Theorem 4.1 by Case 1;
           Colour ( G');
           Assign three colours to e;
       2. e is a 2-edge:
           Let two ends of e be u and v;
           if N(u) ∩ N(v) ≠ ∅ then
           Construct G' as specified in Theorem 4.1 by Case 2.1
           Colour ( G');
           Assign proper colours to deleted edges as shown in Figure 4.3;
           else
           Construct G' as specified in Theorem 4.1 by Case 2.2
           Colour ( G');
           Assign proper colours to deleted edges as shown in Figure 4.3;
           endif
       3. e is a 1-edge:
           Let two ends of e be u and v;
           if N(u) ∩ N(v) ≠ ∅ then
           Construct G' as specified in Theorem 4.1 by Case 3.1
           Colour ( G');
           Assign proper colours to deleted edges as shown in Figure 4.3;
           else
           Construct G' as specified in Theorem 4.1 by Case 3.2
           Colour ( G');
           Assign proper colours to deleted edges as shown in Figure 4.3;
           endif
       endcase
 endif
end Colour;
```

The correctness of the algorithm follows the proof of Theorem 4.1. To see the linearity of the algorithm, we note that G' has at least two edges less than

G. Therefore the reductions are carried out at most m, the number of edges in the original graph, times. Since we have shown in Theorem 4.1 that a reduced graph G' can be constructed from G in constant time, and a four colouring of G can be obtained from a four colouring of G' in constant time, the time complexity of the algorithm is proportional to the number of edges in the original graph. Hence, since the number of edges in a cubic graph is linear in the number of vertices in a cubic graph, we have the following result.

Theorem 4.2. Procedure *Colour* (G) uses at most four colours to properly colour the edges of a cubic graph in linear time.

We should note that the above algorithm can be easily extended to a linear approximation edge colouring algorithm, using at most four colours, for graphs of maximum degree 3. Given a graph G of maximum degree 3, we can construct a loopless cubic graph H which is supergraph of G. Then we can apply the above algorithm to H to obtain a four colouring of H; therefore the restriction of the four colouring of H to G yields a four colouring of G, since G is a subgraph of H.

Suppose G is a graph of maximum degree 3. Let S_1, S_2 be the set of vertices in G of degree 1 and 2 respectively. If $|S_1|$ is even, then pair vertices in S_1 and add a 2-edge between each pair of vertices; otherwise take a vertex v out of S_1, add two new vertices u and w, and add edges vu, vw and a 2-edge uw. For vertices in $S_1 - \{v\}$, pair them and add a 2-edge between each pair of vertices. Similarly, if $|S_2|$ is even, pair vertices in S_2 and add a 1-edge between

each pair of vertices. Otherwise take a vertex v out of S_1, add three new vertices x, u and w, and add edges vx, xu, xw and a 2-edge uw. For vertices in $S_2 - \{v\}$, pair them and add a 1-edge between each pair of vertices. An example is illustrated in Figure 4.4.

Since the number of vertices in H is at most five more than that in G, the size of H is still linear in the number of vertices in the original graph G. It is clear that H can be constructed from G in linear time, and by Theorem 4.2, a four colouring of H can be found in linear time. Therefore a four colouring of G can be obtained in linear time.

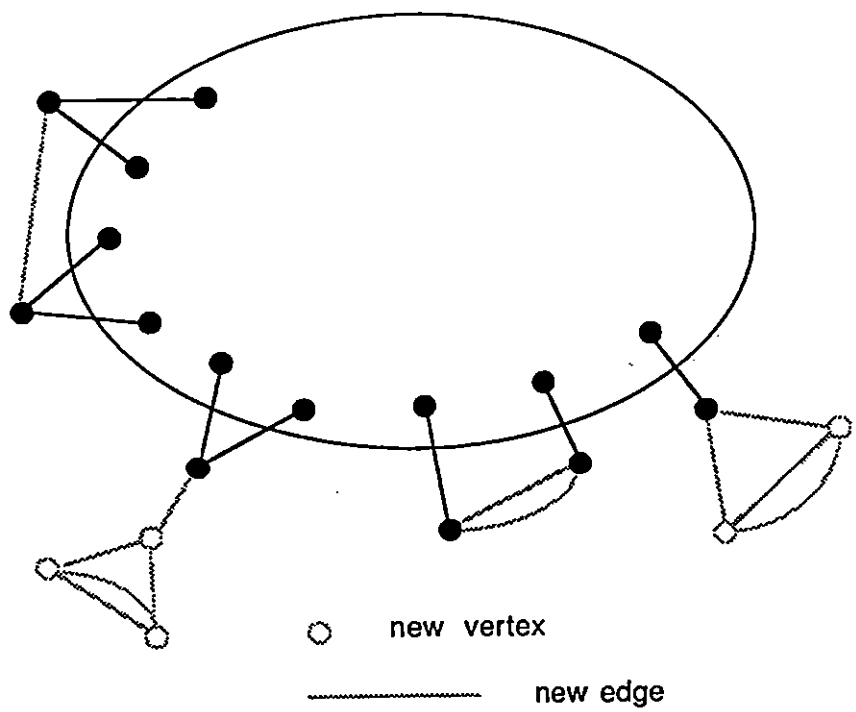


Figure 4.4. Constructing a cubic graph from a graph of maximum degree 3.

4.2. An Algorithm for Deciding the Chromatic Index of Cubic Graphs

Since CHROMATIC INDEX restricted to cubic graphs remains NP-complete, any algorithm for determining the chromatic index of an arbitrary cubic graph will take super polynomial time, on some instances of the problem, unless P = NP. A naive method of exhaustively trying all possible combinations of 3-colourings of the edges of a cubic graph will take time $O(3^{|E|}) = O(5.196^{|V|})$ to find the answer. In this section, we will show that with the time complexity can be reduced to $O(1.682^{|V|})$.

Without loss of generality, we may assume that G is a connected graph. We arbitrary choose a vertex v as the first vertex and use breadth first search to give the vertices an ordering such that $v_1 < v_2$ if and only if v_1 is visited earlier than v_2 in the breadth first search. A 3-colouring of the edges of G, if it exists, can be found as follows:

1. Assign colours 1, 2, 3 to the three edges incident with v.
2. For uncoloured edges incident with each of the remaining vertices, sequentially assign the smallest "feasible" colour to each of them.

Let v_i be a vertex such that no feasible colours are available for the uncoloured edges incident with it, then backtracking takes place from the vertex just before it. If we reach the final vertex in the sequence and find a "feasible" colouring for its incident uncoloured edges, we know that G is 3-chromatic. We also obtain a 3-colouring of G in this case. Otherwise, when backtracking reaches the root vertex v, the algorithm terminates and reports that the graph is

not 3-colourable. In this case, the chromatic index of G is 4. If we need to produce a 4-colouring, we can use the linear approximation algorithm in the previous section to obtain one.

Note that in this vertex ordering, for all v_i in the vertex sequence, at least one edge incident with it is coloured when we reach v_i. If v_i has only one uncoloured edge incident with it, then the colouring of this edge is determined. Otherwise, let the missing colours at vertex v_i be α and β. There are two possible ways to properly 3-colour the two uncoloured edges incident with it: using α for one edge and β for another or vice versa. Thus, we can reduce the number of uncoloured edges by two in this case.

Let $T(\ m\ )$ denote the time complexity of the algorithm on a graph of m uncoloured edges, then in the former case, we have

$$T(m) = T(m-1) + C_1;$$

while in the latter case, we have

$$T(m) = 2T(m-2) + C_2.$$

Therefore, in the worst case, the time complexity of the algorithm is

$$T(m) = MAX\{T(m-1) + C_1, 2T(m-2) + C_2\}.$$

Solving this recurrence relation, we obtain

$$T(m) = O(2^{\frac{m}{2}}).$$

Since for a cubic graph, $|E| = \frac{3|V|}{2}$, and initially $m = |E|$, we have $T(|E|) = O(1.682^{|V|})$.

Theorem 4.3. The chromatic index of a cubic graph can be determined in time $O(1.682^{|V|})$.

CHAPTER 5

SUMMARY

In this thesis, we studied the computational complexity of the edge colouring problem under various restrictions on the instances of the problem.

In Chapter 2, we proved that CHROMATIC INDEX restricted to each of the following classes of graphs remains NP-complete.

Claw-free graphs

Comparability graphs

k-regular comparability graphs for fixed odd k

Perfect graphs

C_k-free graphs for fixed k

Graphs of girth 3,4,5,6,7,8

Triangle-free graphs

k-clique-free graphs for fixed odd k

Complement of claw-free graphs

Complement of C_5-free graphs

Complement of $2K_2$-free graphs

Line graphs

k-regular line graphs for fixed odd k

Line graphs of claw-free graphs

Line graphs of bipartite graphs

Line graphs of C_k-free graphs for fixed k

In Chapter 3, we determined the chromatic index of line graphs of trees and unicyclic graphs. The chromatic index of the line graph of a tree T of maximum degree Δ, denoted $L(T)$, is given by

$$\chi'(L(T)) = \begin{cases} \Delta + 1 & T \text{ is an odd star } K_{1,n} \\ \Delta & \text{otherwise} \end{cases}$$

and the chromatic index of the line graph of a unicyclic graph H of maximum degree Δ, denoted $L(H)$, is given by

$$\chi'(L(H)) = \begin{cases} 3 & H \text{ is an odd cycle} \\ \Delta & \text{otherwise} \end{cases}$$

Linear time optimal edge colouring algorithms for complete graphs, line graphs of trees and line graphs of unicyclic graphs were presented.

In Chapter 4, a linear time approximation algorithm was introduced for edge colouring general cubic graphs with at most four colours. We also presented an optimal edge colouring algorithm for cubic graphs. Although its complexity is exponential, it is superior to previous results.

Finally, we illustrate the relations among various classes of graphs and indicate the known complexity of CHROMATIC INDEX (EDGE COLOURING) restricted to each class in the following pictures [4, 20, 21]. Each box in the picture represents CHROMATIC INDEX restricted to a class of graphs.

References are shown besides the box.

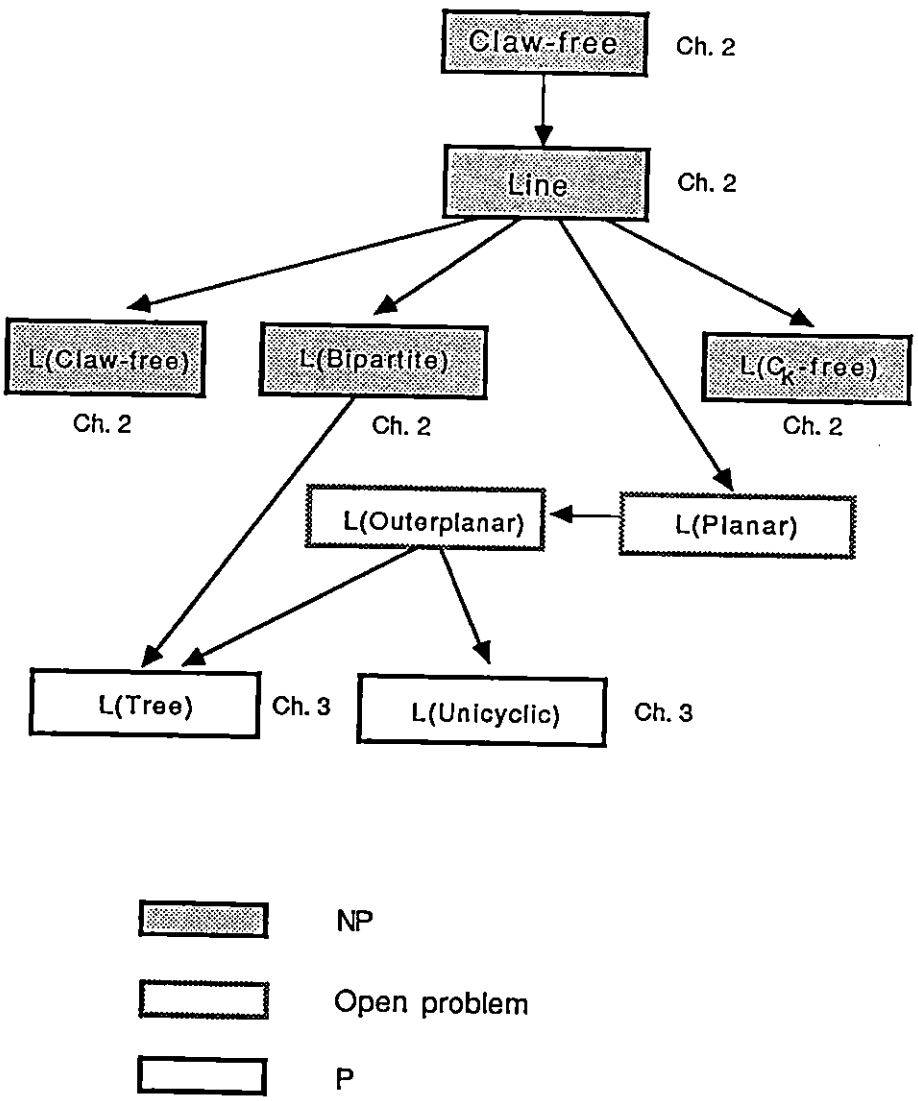


Figure 5.1. Containment relations for classes of restricted line graphs and the complexity of CHROMATIC INDEX restricted to them.

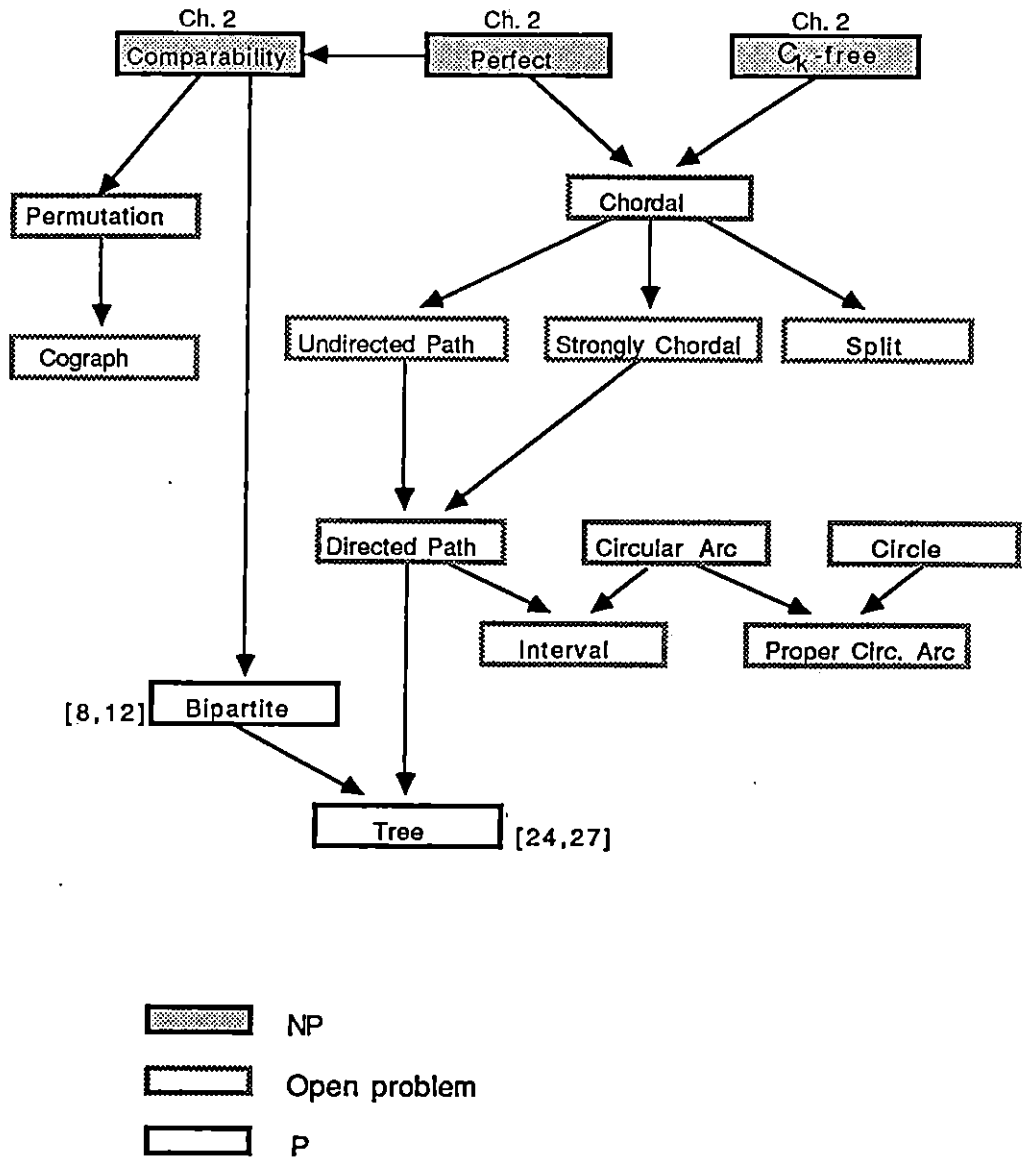


Figure 5.2. Containment relations for classes of perfect graphs and intersection graphs and the complexity of CHROMATIC INDEX restricted to them.

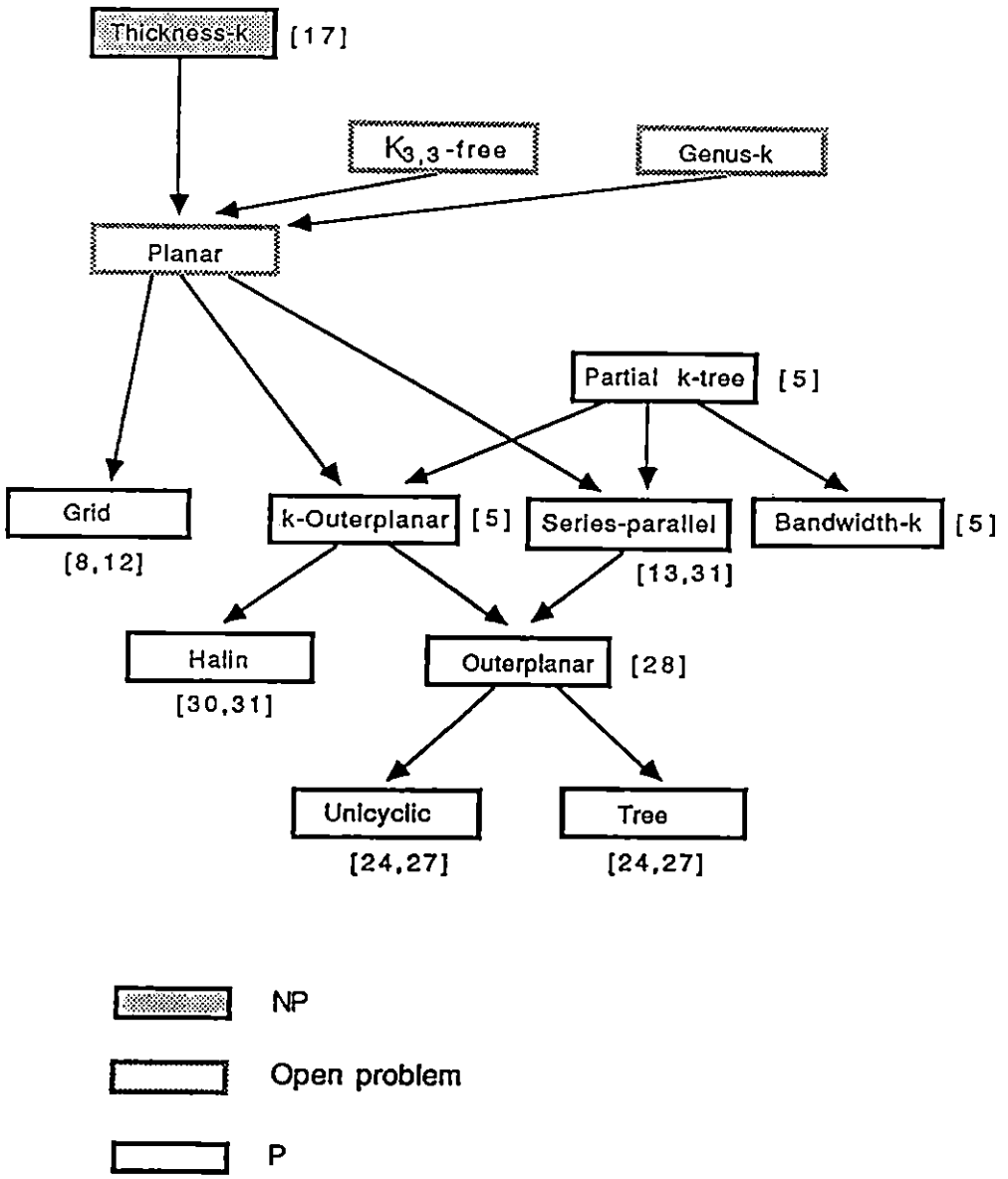


Figure 5.3. Containment relations for classes of graphs related to planarity and the complexity of CHROMATIC INDEX restricted to them.

BIBLIOGRAPHY

[1] E. Arjomandi, An efficient algorithm for colouring the edges of a graph with d+1 colours, *Discrete Mathematical Analysis and Combinatorial Computation* , Fredericton N.B., 1982, 108-132.

[2] M. Behzad, G. Chartrand and J. K. Cooper, The colour numbers of complete graphs, *J. London Math. Soc. 42*, (1967), 226-228.

[3] C. Berge, *Graphs and Hypergraphs*, North-Holland, Great Britain, 1970.

[4] H. L. Bodlaender, Classes of graphs with bound tree-width, *Report No. RUU-CS-86-22* , Dept. of Computer Science, Univ. of Utrecht, The Netherlands, Dec 1986.

[5] H. L. Bodlaender, Polynomial algorithms for chromatic index and graph isomorphism on partial k-trees, *Technical Report RUU-CS-87-17* , Dept. of Computer Science, Univ. of Utrecht, The Netherlands, Oct 1987.

[6] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, Macmillan, 1976.

[7] A. G. Chetwynd and A. J. W. Hilton, The chromatic index of graphs of even order with many edges, *J. of Graph Theory 8*, (1984), 463-470.

[8] R. Cole and J. Hopcroft, On edge colouring bipartite graphs, *SIAM J. Comput. 11*, 3 (1982), 540-546.

[9] A. Ehrenfeucht, V. Faber and H. A. Kierstead, A new method of proving theorems on chromatic index, *Discrete Mathematics 52*, (1984), 159-164.

[10] S. Fiorini and R. J. Wilson, *Edge-colourings of graphs*, *Research Notes in Mathematics 16* , London, 1977.

[11] S. Fiorini, A bibliographic survey of edge-colourings, *J. of Graph Theory 2*, (1978), 93-106.

[12] H. Gabow and O. Kariv, Algorithms for edge colouring bipartite graphs and multigraphs, *SIAM. J. Comput. 11*, (Feb 1982), 117-129.

[13] H. Gabow, T. Nishizeki, O. Kariv, D. Leven and O. Terada, *Algorithms for edge-colouring graphs*, manuscript.

[14] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, San Francisco, 1979.

[15] M. K. Goldberg, Edge-colouring of multigraphs: recolouring technique, *J. of Graph Theory 8*, (1984), 123-137.

[16] F. Harary, *Graph Theory*, Addison-Wesley, 1969.

[17] I. Holyer, The NP-completeness of edge-colouring, *SIAM J. Comput. 10*, (1981), 718-720.

[18] H. Izbicki, An edge colouring problem, *Theory of Graphs and its Applications* , New York/London, 1963, pp. 53-61.

[19] F. Jaeger, Sur L'Indice Chromatique du Graphe Représentatif des Arêtes d'un Graphe Régulier, *Discrete Mathematics 9*, (1974), 161-172.

[20] D. S. Johnson, The NP-completeness column: an ongoing guide (16), *J. of Algorithms 6*, (1985), 434-451.

[21] D. S. Johnson, The NP-completeness column: an cngoing guide (20), *J. of Algorithms 8*, (1987), 438-448.

[22] A. Kotzig, Transformations of Edge-Colouring of Cubic Graphs, *Discrete Mathematics 11*, (1975), 391-399.

[23] D. Leven and Z. Galil, NP-completeness of finding the chromatic index of regular graphs, *J. of Algorithms 4*, (1983), 35-44.

[24] S. Mitchell and S. Hedetniemi, Linear algorithms for edge-colouring trees and unicyclic graphs, *Information Processing Letters 9*, (Oct 1979), 110-112.

[25] T. Nishizeki and K. Kashiwagi, An upper bound on the chromatic index of multigraphs, *Graph Theory and Its Applications to Algorithms and Computer Science* , 1985, 595-604.

[26] M. Plantholt, A generalized construction of chromatic index critical graphs from bipartite graphs, *J. of Graph Theory 8*, (1985), 371-379.

[27] A. Proskurowski and M. Syslo, Edge-colouring of Trees and Unicyclic Graphs, *ARS Combinatorica 16*, (1983), 17-20.

[28] A. Proskurowski and M. Syslo, Efficient vertex- and edge-colouring of outerplanar graphs, *SIAM J. Alg. Disc. Math.* , 1985.

[29] P. D. Seymour, On Multi-colourings of cubic graphs and conjectures of Fulkerson and Tutte, *Proc. London Math. Soc. (3) 38*, (1979), 423-460.

[30] M. Skowronska, Efficient vertex- and edge-colouring of Halin graphs, *Proc. of the 3rd Czechoslovak Symp. on Graph Theory* , Prague, 1982.

[31] M. Syslo, NP-complete problems on some tree-structured graphs: a review, *Proc. WG '83 International Workshop on Graphtheoretic Concepts in Computer Science* , West Germany, 1983, 342-353.

[32] P. G. Tait, Remarks on the colouring of maps, *Proc. Roy. Soc. 10*, (1880), 729.

[33] O. Terada and T. Nishizeki, Approximate algorithms for the edge-colouring of graphs (in Japanese), *Trans. Institute of Electronics and Communication Engineers of Japan J65-D*, (Nov 1982), 1382-1389.

[34] O. Terada and T. Nishizeki, Approximate algorithms for the edge-colouring of graphs (abstracts), *AMS 3*, (June 1982), 286.

[35] R. J. Wilson, Edge-colourings of graphs — A survey, *Theory and Applications of Graphs* , Springer-Verlag, New York, 1978, 608-619.

APPENDIX A

DEFINITIONS IN GRAPH THEORY

Graph, Vertex and Edge

A *graph* G is a pair $(V(G), E(G))$, where $V(G)$ is a finite nonempty set of elements called *vertices* and $E(G)$ is a finite set of distinct unordered pairs of distinct elements of $V(G)$ called *edges*. $V(G)$ is called the vertex set of G, and $E(G)$ the edge set of G. We use $|V(G)|$ and $|E(G)|$ to denote the number of elements in $V(G)$ and $E(G)$ respectively. $|V(G)|$ is usually called the *order* of G. When it is clear from the context, we denote the vertex and edge sets of G by V and E, respectively.

If $e = \{u, v\}$ is an edge, then e is said to *join* the vertices u and v, and u and v are said to be *adjacent*. We also say that u and v are *neighbours* and are *ends* of e, and that e is *incident* to u and v. If two edges e_1 and e_2 are incident with a common vertex, then they are *adjacent edges*. For convenience, we denote an edge by uv or vu rather than $\{u, v\}$. For a vertex v of a graph G, the *neighbourhood* $N(v)$ of v consists of all the vertices of G adjacent to v. Similarly the *neighbourhood* $N(e)$ of an edge e of G consists of all edges of G adjacent to e.

Subgraphs

A *subgraph* of a graph $G = (V(G), E(G))$ is a graph $H = (V(H), E(H))$ such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$, in this case, we also say that G

is a *supergraph* of H. Let S be a nonempty subset of the vertex set $V(G)$ of a graph G, then the subgraph $G[S]$ *induced* by S is the graph having vertex set S and whose edge set consists of those edges of G incident with two elements of S. Given a graph H, an *H-free graph* is one which does not contain H as an induced subgraph.

Degree

The *degree* of a vertex v in graph G, denoted $deg(v)$, is the number of edges incident with v. The maximum degree of a graph G is denoted by $\Delta(G)$ or Δ when it is clear from the context which graph it refers to.

Path and Cycle

A *path* is an alternating sequence of distinct vertices and edges of G such that each edge is immediately preceded and succeeded by the two vertices with which it is incident. If two ends of a path coincide, then it is called a *cycle*. The *length* of a path (cycle) is the number of edges in the path (cycle). The *girth* of a graph containing cycles is the length of the smallest cycle in the graph. A *k-cage*, $k \geqslant 3$, is a cubic graph of girth k with the minimum number of vertices.

Connectivity

A graph G is *connected* if there is a path joining each pair of vertices. A maximal connected subgraph of G is called a *connected component* or simply a *component* of G. A *cutpoint* is a vertex whose removal increases the number of components, and a *bridge* is such an edge. A graph is *2–connected* iff it is a

connected graph of order $\geqslant 3$ and has no cutpoints, a *block* is a maximal 2-connected subgraph. An *end block* of a graph G is a block which contains exactly one cutpoint of G. An *edge cut set* $[S, \bar{S}]$ is a set of edges with one end in S and the other end in $\bar{S}$, where S is a subset of the vertex set.

Special Graphs

A *complete graph* K_n is a graph in which every two vertices are adjacent. A subgraph which is a complete graph is called a *clique*. A *k-clique* is a clique with k vertices. A *bipartite graph* is one whose vertex set can be partitioned into two sets such that each edge joins a vertex in one set to some vertex in the other set. A *complete bipartite graph* is a bipartite graph in which every vertex in one set is adjacent to every vertex in the other set. A *k-cycle* is a graph induced by a cycle of length k. A *regular graph* or *k-regular graph* is one in which each vertex has degree k. The *complement graph* $\bar{G}$ of G is the graph with the same vertex set as G, but where two vertices are adjacent iff they are not adjacent in G.

Operations on Graphs

The *union* of two graphs G_1 and G_2 is a graph G such that $V = V_1 \cup V_2$ and $E = E_1 \cup E_2$. We use mK_n to represent m disjoint copies of K_n. The *subdivision* of an edge is the insertion of a new vertex into that edge, and the *subdivision* of a graph is a graph obtained by subdividing each edge in the graph. Let $e_1, e_2, ..., e_k$ be edges, we use $G - \{e_1, e_2, ..., e_k\}$ to denote the graph obtained from G by removing the edges $e_1, e_2, ..., e_k$. Let $v_1, v_2, ..., v_k$ be

vertices, we use $G - \{v_1, v_2, ..., v_k\}$ to denote the graph obtained from G by removing the vertices $v_1, v_2, ..., v_k$ and all edges incident to any of them. Let H be a subgraph of G, we use $G - H$ to denote the graph obtained from G by removing all edges which are in H from G and all vertices which have the same degree in H as in G from G.

Intersection Graphs

The *line graph* of G, denoted $L(G)$, is a graph formed in the following way: the vertices of $L(G)$ represent the edges of G, two vertices of $L(G)$ are adjacent whenever the corresponding edges of G are adjacent. The *block graph* of G, denoted $B(G)$, is a graph formed by taking each block of G as a vertex and two vertices of $B(G)$ are adjacent whenever their corresponding blocks are.

VITA

Surname: Cai Given Names: Leizhen

Place of Birth: Zhejiang, China Date of Birth: July 27, 1962

Educational Institutions Attended, with Dates of Entering and Leaving:

Zhejinag University, Hangzhou, Zhejiang, China	1978 to 1982
Southern Methodist University, Dallas, Texas, USA	1985 to 1986
University of Victoria, Victoria, B.C. Canada	1986 to 1988

Degrees, Diplomas, Etc., Awarded, with Dates and Names of Institutions:

B.Sc.	Zhejiang University, Hangzhou, China	1982

Honors and Awards:

Zhejiang University Scholarship	1979	
Zhejinag University Research Awards	1984	
University of Victoria Fellowship	1986-1987	1987-1988

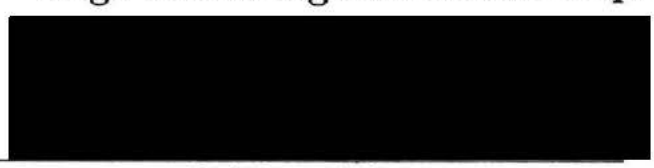
Publications:

(1) **Data Structures and Algorithms for the Generation and Display of Curves and Surfaces,** (with Feng Shuchun), Journal of Zhejiang University, May, 1985

PARTIAL COPYRIGHT LICENSE

I hereby grant the right to lend my thesis (the title of which is shown below) to users of the University of Victoria Library, and to make *single copies only* for such users or in response to a request from the library of any other university, or similar institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by me or a member of the University designated by me. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

The Computational Complexity of Edge Colouring Restricted Graphs

Author 

Leizhen Cai

April 25, 1988

Date