R.M.R. Lewis

A Guide to Graph Colouring

Algorithms and Applications

Springer

A Guide to Graph Colouring

R.M.R. Lewis

A Guide to Graph Colouring

Algorithms and Applications

Springer

R.M.R. Lewis
Cardiff School of Mathematics
Cardiff University
Cardiff
UK

ISBN 978-3-319-25728-0 ISBN 978-3-319-25730-3 (eBook)
DOI 10.1007/978-3-319-25730-3

Library of Congress Control Number: 2015954340

Springer Cham Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

For Fifi, Maiwen, Aoibh, and Macsen.
Gyda cariad.

Preface

Graph colouring is one of those rare examples in the mathematical sciences of a problem that is very easy to state and visualise, but that has many aspects that are exceptionally difficult to solve. Indeed, it took more than 160 years and the collective efforts of some of the most brilliant minds in nineteenth and twentieth century mathematics just to prove the simple sounding proposition that "four colours are sufficient to properly colour the vertices of a planar graph".

Ever since the notion of "colouring" graphs was first introduced by Frances Guthrie in the mid-1800s, research into this problem area has focussed mostly on its many theoretical aspects, particularly concerning statements on the chromatic number for specific topologies such as planar graphs, line graphs, random graphs, critical graphs, triangle free graphs, and perfect graphs. Excellent reviews on these matters, together with a comprehensive list of open problems in the field of graph colouring, can be found in the books of Jensen and Toft (1994) and Beineke and Wilson (2015).

In this book, our aim is to examine graph colouring as an *algorithmic* problem, with a strong emphasis on practical applications. In particular, we take some time to describe and analyse some of the best-known algorithms for colouring arbitrary graphs and focus on issues such as (a) whether these algorithms are able to provide optimal solutions in some cases, (b) how they perform on graphs where the chromatic number is unknown, and (c) whether they are able to produce better solutions than other algorithms for certain types of graphs, and why.

This book also devotes a lot of effort into looking at many of the real-world operational research problems that can be tackled using graph colouring techniques. These include the seemingly disparate problem areas of producing sports schedules, solving Sudoku puzzles, checking for short circuits on printed circuit boards, assigning taxis to customer requests, timetabling lectures at a university, finding good seating plans for guests at a wedding, and assigning computer programming variables to computer registers.

This book is written with the presumption that the reader has no previous experience in graph colouring, or graph theory more generally. However, an elementary knowledge of the notation and concepts surrounding sets, matrices, and enumerative

combinatorics (particularly combinations and permutations) is assumed. The initial sections of Chapter 1 are kept deliberately light, giving a brief tour of the graph colouring problem using minimal jargon and plenty of illustrated examples. Later sections of this chapter then go on to look at the problem from an algorithmic point of view, looking particularly at why this problem is considered "intractable" in the general case, helping to set the ground for the remaining chapters.

Chapter 2 of this book looks at three different well-established constructive algorithms for the graph colouring problem, namely the GREEDY, DSATUR, and RLF algorithms. The various features of these algorithms are analysed and their performance (in terms of running times and solution quality) is then compared across a large set of problem instances. A number of bounds on the chromatic number are also stated and proved.

Chapters 3 and 4 then go on to look at some of the best-known algorithms for the general graph colouring problem. Chapter 3 presents more of an overview of this area and highlights many of the techniques that can be used for the problem, including backtracking algorithms, integer programming methods, and metaheuristics. Ways in which problem sizes can be reduced are also considered. Chapter 4 then goes on to give an in-depth analysis of six such algorithms, describing their relevant features, and comparing their performance on a wide range of different graph types. Portions of this chapter are based on the research originally published by Lewis et al. (2012).

Chapter 5 considers a number of example problems, both theoretical and practical, that can be expressed using graph colouring principles. Initial sections focus on special cases of the graph colouring problem, including map colouring (together with a history of the Four Colour Theorem), edge colouring, and solving Latin squares and Sudoku puzzles. The problems of colouring graphs where only limited information about a graph is known, or where a graph is subject to change over time, are also considered, as are some natural extensions to graph colouring such as list colouring, equitable graph colouring and weighted graph colouring.

The final three chapters of this book look at three separate case studies in which graph colouring algorithms have been used to solve real-world practical problems, namely the design of seating plans for large gatherings, creating schedules for sports competitions (Lewis and Thompson, 2010), and timetabling events at educational establishments (Lewis and Thompson, 2015). These three chapters are written so that, to a large extent, they can be read independently of the other chapters of this book, though obviously a full understanding of their content will only follow by referring to the relevant sections as instructed by the text.

A Note on Pseudocode and Notation

While many of the algorithms featured in this book are described within the main text, others are more conveniently defined using pseudocode. The benefit of pseudocode is that it enables readers to concentrate on the algorithmic process without

worrying about the syntactic details of any particular programming language. Our pseudocode style is based on that of the seminal textbook *Introduction to Algorithms* by Cormen, Leiserson, Rivest and Stein, often simply known as the "The Big Book of Algorithms" (Cormen et al., 2009). This particular pseudocode style makes use of all the usual programming constructs such as while-loops, for-loops, if-else statements, break statements, and so on, with indentation being used to indicate their scope. To avoid confusion, different symbols are also used for assignment and equality operators. For assignment, a left arrow ($\leftarrow$) is used. So, for example, the statement $x \leftarrow 10$ should be read as "x becomes equal to 10", or "let x be equal to 10". On the other hand, an equals symbol is used only for equality *testing*; hence a statement such as $x = 10$ will only evaluate to true or false (x is either equal to 10, or it is not).

All other notation used within this book is defined as and when the necessary concepts arise. Throughout the text, the notation $G = (V,E)$ is used to denote a graph G comprising a "vertex set" V and an "edge set" E. The number of vertices and edges in a graph are denoted by n and m respectively. The colour of a particular vertex $v \in V$ is written $c(v)$, while a candidate solution to a graph colouring problem is usually defined as a partition of the vertices into k subsets $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$. Further details can be found in the various definitions within Chapters 1 and 2.

Additional Resources

This book is accompanied by a suite of nine graph colouring algorithms that can be downloaded from www.rhydlewis.eu/resources/gCol.zip. Each of these heuristic-based algorithms are analysed in detail in the text and are also compared and contrasted empirically through extensive experimentation. These implementations are written in C++ and have been successfully compiled on a number of different compilers and platforms. (See Appendix A.1 for further details.) Readers are invited to experiment with these algorithms as they make their way through this book. Any queries should be addressed to the author.

In addition to this suite, this book's appendix also contains information on how graph colouring problems might be solved using commercial linear programming software and also via the free mathematical software Sage. Finally, an online implementation of the table planning algorithm presented in Chapter 6 can also be accessed at www.weddingseatplanner.com.

Cardiff University, Wales. *Rhyd Lewis*
August 2015

Contents

Chapter 1
Introduction to Graph Colouring

In mathematics, a graph can be thought of as a set of objects in which some pairs of objects are connected by links. The interconnected objects are usually called *vertices*, with the links connecting pairs of vertices termed *edges*. Graphs can be used to model a surprisingly large number of problem areas, including social networking, chemistry, scheduling, parcel delivery, satellite navigation, electrical engineering, and computer networking. In this chapter we introduce the graph colouring problem and give a number of examples of where it is encountered in real-world situations. Statements on the complexity of the problem are also made.

The graph colouring problem is one of the most famous problems in the field of graph theory and has a long and illustrious history. In a nutshell it asks, given any graph, how might we go about assigning "colours" to all of its vertices so that (a) no vertices joined by an edge are given the same colour, and (b) the number of different colours used is minimised?

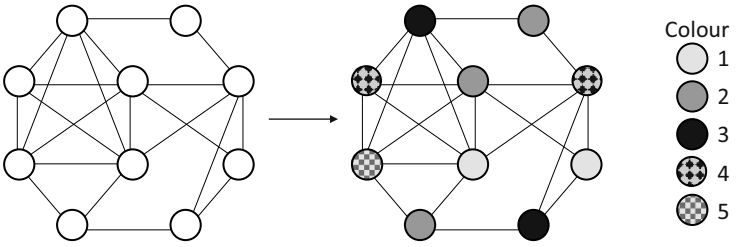

Fig. 1.1 A small graph (a), and corresponding 5-colouring (b)

Figure 1.1 shows a picture of a graph with ten vertices (the circles), and 21 edges (the lines connecting the circles). It also shows an example colouring of this graph that uses five different colours. We can call this solution a "proper" colouring because all pairs of vertices joined by edges have been assigned to different colours, as required by the problem. Specifically, two vertices have been assigned to colour 1, three vertices to colour 2, two vertices to colour 3, two vertices to colour 4, and one vertex to colour 5.

R.M.R. Lewis, *A Guide to Graph Colouring*,
DOI 10.1007/978-3-319-25730-3_1

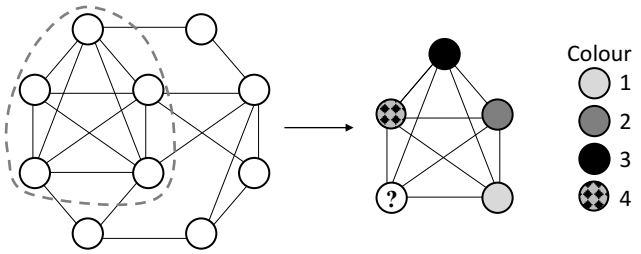

Fig. 1.2 If we extract the vertices in the dotted circle, we are left with a subgraph that clearly needs more than four colours

Actually, this solution is not the only possible 5-colouring for this example graph. For example, swapping the colours of the bottom two vertices in the figure would give us a different proper 5-colouring. It is also possible to colour the graph with anything between six and ten colours (where ten is the number of vertices in the graph), because assigning a vertex to an additional, newly created, colour still ensures that the colouring remains proper.

But what if we wanted to colour this graph using *fewer* than five colours? Is this possible? To answer this question, consider Figure 1.2, where the dotted line indicates a selected portion of the graph. When we remove everything from outside this selection, we are left with a subgraph containing just five vertices. Importantly, we can see that every pair of vertices in this subgraph has an edge between them. If we were to have only four colours available to us, as indicated in the figure we would be unable to properly colour this subgraph, since its five vertices all need to be assigned to a different colour in this instance. This allows us to conclude that the solution in Figure 1.1 is actually *optimal*, since there is no solution available that uses fewer than five colours.

1.1 Some Simple Practical Applications

Let us now consider four simple practical applications of graph colouring to further illustrate the underlying concepts of the problem.

1.1.1 A Team Building Exercise

An instructive way to visualise the graph colouring problem is to imagine the vertices of a graph as a set of "items" that need to be divided into "groups". As an example, imagine we have a set of university students that we want to split into groups for a team building exercise. In addition, imagine we are interested in divid-

ing the students so that no student is put in a group containing one or more of his friends, and so that the number of groups used is minimal. How might this be done?

Consider the example given in the table in Figure 1.3(a), where we have a list of eight students with names A through to H, together with information on who their friends are. From this information we can see that student A is friends with three students (B, C and G), student B is friends with four students (A, C, E, and F), and so on. Note that the information in this table is "symmetric" in that if student x lists student y as one of his friends, then student y also does the same with student x. This sort of relationship occurs in social networks such as Facebook, where two people are only considered friends if both parties agree to be friends in advance. An illustration of this example in graph form is also given in the figure.

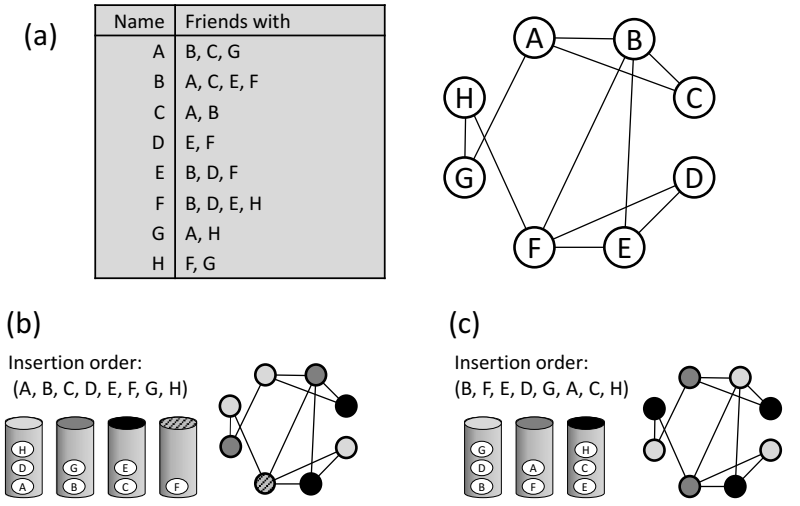

Fig. 1.3 Illustration of how proper 5- and 4-colourings can be constructed from the same graph

Let us now attempt to split the eight students of this problem into groups so that each student is put into a different group to that of his friends'. A simple method to do this might be to take the students one by one in alphabetical order and assign them to the first group where none of their friends are currently placed. Walking through the process, we start by taking student A and assigning him to the first group. Next, we take student B and see that he is friends with someone in the first group (student A), and so we put him into the second group. Taking student C next, we notice that he is friends with someone in the first group (student A) and also the second group (student B), meaning that he must now be assigned to a third group. At this point we have only considered three students, yet we have created three separate groups. What about the next student? Looking at the information we can see that student D is only friends with E and F, allowing us to place him into the first group alongside student A. Following this, student E cannot be assigned to the first group because he