University of London
Imperial College of Science and Technology
Department of Management Science

GRAPH COLOURING AND RELATED PROBLEMS IN OPERATIONS RESEARCH

by

SAMUEL MOSES KORMAN, B.Sc., M.Sc.

A thesis submitted for the
Degree of Doctor of Philosophy and
the Diploma of Imperial College

July, 1975.

ABSTRACT

This thesis is concerned with the Graph Colouring Problem, and in particular with the development of new algorithmic methods for this problem. Two other problems which are closely involved with the Graph Colouring Problem, viz. the Set-Covering and Clique-Generating problems,are also investigated in some detail.

Chapter 1 introduces the Graph Colouring Problem. Some basic definitions and known results are given. Two integer programming formulations are detailed, and the use of Set-Covering methods in solving the Graph Colouring Problem is described.

In Chapter 2, the Set Colouring Problem is investigated in its own right, and a survey of the main previously known algorithms for this problem is included. A new method for solving the Set Covering Problem is developed, and the chapter concludes with a computational comparison of these algorithms.

Chapter 3 is concerned with the generation of the maximal independent sets of a graph, which are needed for the solution of the Graph Colouring Problem as a Set Covering Problem. Various algorithms for generating these sets are described and compared. Finally, methods of finding the maximal independent sets of a graph from certain of its related graphs are derived.

Chapter 4 develops some theoretical results in the properties of optimal colouring which, in addition to leading to some new algorithms for graph-colouring, are also interesting in their own right. Theoretical consequences of these results are discussed and a one-pass algorithm for colouring perfect graphs is described.

Chapters 5 and 6 are devoted to direct graph-colouring algorithms based on tree-search principles. Chapter 5 considers colour-by-colour algorithms, and a new algorithm is developed which is based on the results of the previous chapter. Chapter 6

considers the vertex-by-vertex colouring algorithms; new bounds are developed and the results of Chapter 4 are used as dominance conditions to limit the search.

Chapter 7 concludes the thesis by summarising the results of the previous chapters, and indicating possible future areas for research.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. N. Christofides, of the Department of Management Science, Imperial College, for his advice and encouragement during the course of this work; and also Professor S. Eilon, Head of Department, and other staff of the department, for their helpfulness.

I am very grateful to Professor K. Wolfenden, Department of Statistics and Computer Science, University College, London, for his general encouragement and advice over the years; and I wish particularly to thank Mrs. S. Fry for her excellent work of typing this thesis.

The financial support of the Science Research Council for a period of three years is gratefully acknowledged.

TABLE OF CONTENTS

Chapter I: Introduction to Graph Colouring 1

1. The Graph Colouring Problem and its Applications 2
2. Elements of Graph Colouring 6
   2.1 General Graph Theory Definitions 6
   2.2 Colouring-related Defintions and Elementary Consequences 12
   2.3 Some Known Results on Colouring 14
      2.3.1 Further Bounds on the Chromatic Number 14
      2.3.2 Comments on the Previous Bounds 15
      2.3.3 Two Classical Theorems 16
3. Alternative Formulation of the Graph Colouring Problem 17
   3.1 Formulation as a General (0,1) Integer Program 17
   3.1 Formulation on a Set Partitioning / Covering Problem 22
   3.3 Other Concepts Related to Graph Colouring 27
4. Elementary Reductions in Problem Size 28
5. References 32

Chapter II: The Set Covering Problem 34

1. Introduction 35
   1.1 Problem Formulation 37
   1.2 Problem Reduction 38
2. Methods of Solution 39
   2.1 Tree Search Methods 39
   2.2 Cutting Plane Approaches 49
   2.3 Lawler's Duality Method 55
3. Some Modifications and Improvements 57
   3.1 Dynamic Application of Reduction Tests 57
   3.2 Further Dominance Tests 58
   3.3 A New Lower Bound 60
4. Computational Results 63
5. References 71
   Appendix 78

CHAPTER III: The Generation of Cliques and Maximal Independent Sets of Graphs 81

1. Introduction 82
2. Algorithms for Clique Determination 85
   2.1 The Straight-forward Approach 85
   2.2 Improvements to the Straight-forward Method 86
   2.3 A Breadth-first Algorithm 88
   2.4 A Point-Removal Method 90
3. Computational Comparisons 91

4. Cliques of Related Graphs 93
4.1 The Importance of Certain Related Graphs 93
4.2 Preliminary Theory 94
4.3 The Main Theorems 102
4.4 Clique Generating Algorithms based on Theorem 3 104
4.5 Two Specializations 109
5. MIS of Related Graphs 110
5.1 Preliminary Theory 110
5.2 Main Theorems 115
5.3 MIS-Generating Algorithms based on Theorem 6 116
5.4 Two Specializations 119
6. References 121
Appendix 122

CHAPTER IV: Theoretical Aspects of Graph Colouring 141

1. Results for General Graphs 142
1.1 A Basic Theorem 142
1.2 The Intersection Theorem and Some Corollaries 144
1.3 Graph Colouring and Subset Condensation 147
2. Perfect Graphs and P-Critical Graphs 154
2.1 Introduction to Perfect Graphs 154
2.2 Some Consequences of the Intersection Theorem 157
3. Colouring Perfect Graphs 159
3.1 A One-Pass Perfect-Graph Colouring Algorithm 160
3.2 Illustration of the Algorithm 161
4. References 163

CHAPTER V: Colour-by-Colour Graph Colouring 164

1. Introduction 165
2. A Complete Depth-First Approach 166
2.1 Preliminaries 166
2.2 Description of the Algorithm 167
2.3 Demonstration of the Algorithm 168
2.4 Simple Modifications to the Above Algorithm 172
3. Pruning the Search-Tree 175
3.1 Theoretic Justification 175
3.2 An Equivalent Depth-First Approach 178
4. A Multiple Algorithm for Optimal Graph Colouring 183
4.1 Introduction 183
4.2 Some Theoretical Considerations 184
4.3 An Overview of the Multiple Algorithm 186
4.4 Implementation of the Multiple Algorithm 187

4.5 Description of the Multiple Depth-First Algorithm for Optimal Graph Colouring 194
4.6 Computational Considerations of the Multiple Algorithm 202
4.7 Variants of the Multiple Algorithm 204
4.8 SCP and GCP 206
5. Computational Results 207
6. References 210

CHAPTER VI: Vertex-by-Vertex Graph Colouring 211

1. The Basic Algorithm and Some Variations 212
1.1 A Simple Vertex-by-Vertex Colouring Algorithm 212
1.2 Improvements to the Simple Algorithm 216
1.3 Further Modifications and a Lower Bound 220
1.4 An Improved Vertex-by-Vertex Colouring Algorithm 222
1.5 A Dual Vertex-by-Vertex Colouring Algorithm 223
1.6 Vertex-by-Vertex Colouring and Bi-directional Search 225
2. A Dichotomous Search Algorithm 226
2.1 Zykov's Original Method 227
2.2 Improvements 228
3. Vertex-by-Vertex Colouring and the Intersection Theorem 231
3.1 Preliminaries 231
3.2 Use of the Intersection Theorem 232
3.2.1 Necessary Conditions 232
3.2.2 The Application of the I.T. at an Arbitrary Stage 234
3.3 Practical Considerations 235
3.3.1 Use of the I.T. as a Lookahead Bound 236
3.3.2 Use of the I.T. to Direct the Search 236
3.4 The Intersection Theorem and a Lower Bound 239
4. Computational Results 240
5. References 242

CHAPTER VII: Conclusions 244

1. General Summary 245
2. Conjectures Old and New 246
2.1 The Four Colour Conjecture 246
2.2 The Strong Perfect Graph Conjecture 247

CHAPTER I

INTRODUCTION TO GRAPH COLOURING

1. THE GRAPH COLOURING PROBLEM AND ITS APPLICATIONS

Suppose one wishes to schedule examinations into time periods in a situation where the particular courses chosen by any student throughout the year involve a wide range of subjects. Then it is necessary that no two examinations be scheduled into the same time period if there is any student taking both of the corresponding courses (the further complication introduced by room availability is not considered here). One could, for example, schedule each examination for a different time period. On the other hand, it is often desired to determine a schedule involving the fewest time periods, into which the examinations can take place.

The above problem is a direct application of the well-known Graph Colouring Problem, e.g. [20,22,23] . For example, suppose seven subjects are offered: English (E), French (F), Latin (L), Mathematics (M), Physics (P), Chemistry (C), and Biology (B), of which two or three must be chosen, and that, of the 35, say, students, eight take, E, F, and L, four take F and M, one takes E and C, two take E,M, and C, eleven take M,P, and C, eight take P, C, and B and one takes B and L. As it is the examinations which are to be scheduled we could then set up a matrix of subjects-by-subjects and note those pairs of subjects which are taken by one or more students as these examinations cannot be held concurrently. Alternatively, we could represent the above information in the form of a 'graph' by letting the subjects be represented by points (or vertices) of the graph and joining any two points by a line (or arc) of the graph of the subjects represented by those points are taken by one or more students, so obtaining the graph in Figure 1.

The problem of scheduling the examinations into the fewest time periods can then be solved by partitioning the set of vertices

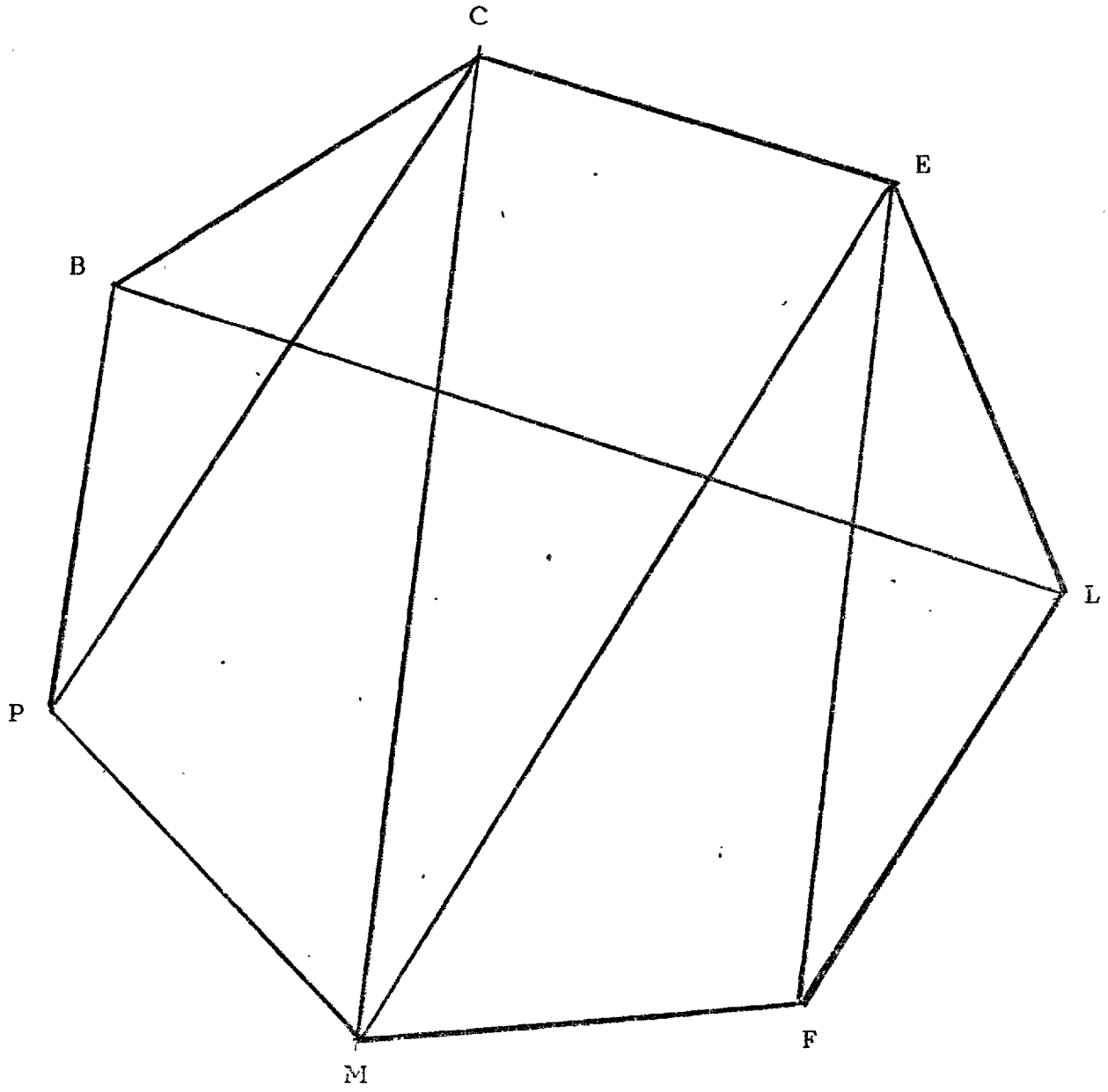


FIGURE 1

$\{E, F, L, M, P, C, B\}$ into the smallest number of disjoint subsets which together, between them, involve all the vertices and which are such that no two vertices which are joined by an arc are in the same subset. Equivalently, by letting the various time periods be represented by different colours, we wish to 'colour' the graph by assigning colours to the vertices so as to use as few colours as possible and where no two vertices which are joined by an arc can be coloured the same.

In this particular case, the minimum number of time periods is in fact four, and there are many optional schedules, a typical such schedule being $\{C \& L\}$; $\{B \& M\}$; $\{P \& E\}$; $\{F\}$.

Timetabling is only one of the many applications of the Graph Colouring Problem, which also appears, both in its own right and as a subproblem - as for example, in the case of the general timetabling problem mentioned above [6;13] - in many other fields of Operations Research.

Thus, in the field of resource allocation, [5], suppose there are n jobs to be processed, each of which involves the use of a certain subset of available resources - e.g. machines - (the processing times of the jobs being assumed to be identical). Jobs that do not require the same resource can be processed simultaneously.

The problem, then, is to allocate resources to the jobs so as to minimise the total processing time for all n jobs. This problem is equivalent to that of colouring the graph consisting of the n jobs as vertices, two vertices being adjacent if, and only if, the corresponding jobs have a common resource requirement. The optimal colouring of this graph then corresponds to the least amount of time units needed to complete all n jobs.

Another application is for the simple loading problem [7]: finding the minimum number of containers (of assumed infinite

capacity) into which to load n items, where certain pairs of items must of necessity be in separate containers due to interaction between them. This problem is again equivalent to colouring a graph with items as vertices and arcs between any pair of vertices corresponding to interacting items. The above problem can also be generated in a number of ways to deal with containers of finite capacity [7].

Yet another application of the colouring problem is in wiring printed circuits [10]. Thus, consider a printed circuit board on which we wish to join pairs of terminals with straight printed wires. As these wires are not insulated, no two wires must meet (i.e. cross each other) except at a terminal. However, as the wires are straight, and so will inevitably cross each other, it is necessary to distribute these links into a number of parallel boards, (the terminals being available on all boards). The problem then is to determine the minimum number of boards necessary to realize the configuration; again, it can be solved by (optimally) colouring the graph obtained by associating vertices with the wire links, two vertices being adjacent, if, and only if, the corresponding wires cross each other in the original configuration.

Before going on to give various formulations of the problem, we first give in the following section, the basic graph theoretic definitions which will be used throughout this work, and which generally correspond to those of Berge [3].

2. ELEMENTS OF GRAPH COLOURING

2.1 General Graph Theory Definitions

A graph G is defined as a finite set of points $\{x_1,\ldots,x_n\}$, together with a symmetric relation $\mathcal{R}$ between some pairs of these points. The points $\{x_1,\ldots,x_n\}$ are known as the vertices of G and the (unordered) pairs (x_i, x_j) are known as the arcs of G, where (x_i, x_j) is an arc of G if and only if $x_i \mathcal{R} x_j$ (or indeed $x_j \mathcal{R} x_i$). The vertices x_i and x_j are then said to be adjacent, while the arc (x_i, x_j) is said to be incident to x_i and x_j, its end-vertices.

Corresponding to these two concepts of adjacency and incidence, there are two main methods for representing a graph G with vertex set X in the form of a matrix. Thus, suppose that G contains n vertices, $\{x_1,\ldots,x_n\}$, and m arcs $\{u_1,\ldots,u_m\}$. Then each of the two following matrices completely identifies G:

(i) The $(n \times n)$ vertex-vertex adjacency matrix $A = [a_{ij}]$

defined by: $a_{ij} = 1$ if and only if vertex x_i is adjacent to vertex x_j

$= 0$ otherwise

(ii) The $(m \times n)$ arc-vertex incidence matrix $B = [b_{ki}]$

defined by: $b_{ki} = 1$ if and only if arc u_k is incident to vertex x_i

$= 0$ otherwise

The relation $\mathcal{R}$ can also be represented by a mapping $\Gamma: X \to X$

where $X = \{x_1, \ldots, x_n\}$ is the vertex set of G, and where Γx_i represents the set of vertices $x_j \in X$ for which $x_i \mathcal{R} x_j$. Thus, for any vertex x_i of G, $|\Gamma x_i|$* corresponds to the number of arcs of G which are adjacent to x_i, which is also known as the degree of x_i; denoted $d(x_i)$. We note here that, by convention $x_i \notin \Gamma x_i$, but that, as indicated above, $x_j \in \Gamma x_i \Leftrightarrow x_i \in \Gamma x_j$.** Also, as the knowledge of X and Γ uniquely identifies the graph G, we often denote the graph G as $G = (X, \Gamma)$. Finally, any graph $G = (X, \Gamma)$ can be represented by letting the vertices correspond to points on a plane and joining adjacent pairs of vertices by a line or curve.

Example: The graph $G = (X, \Gamma)$ with $X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$ and Γ defined by

$$\left.\begin{aligned} \Gamma x_1 &= \{x_2, x_4, x_5, x_6\} \\ \Gamma x_2 &= \{x_1, x_3, x_5, x_6\} \\ \Gamma x_3 &= \{x_2, x_5\} \\ \Gamma x_4 &= \{x_1\} \\ \Gamma x_5 &= \{x_1, x_2, x_3, x_6\} \\ \Gamma x_6 &= \{x_1, x_2, x_5\} \end{aligned}\right\}$$

is depicted in Figure 2. It can then be seen, for example, that x_3 is adjacent to x_2 and x_5 and so is of degree 2 ; while x_4 is adjacent only to x_1 so its degree is 1. Also both x_1 and x_6 are incident to the arc (x_1, x_6)

---

* For any set S, $|S|$ denotes the cardinality of S, i.e. the number of elements in the set S.

** In a directed graph, on the other hand, $x_j \in \Gamma x_i$ does not imply $x_i \in \Gamma x_j$.

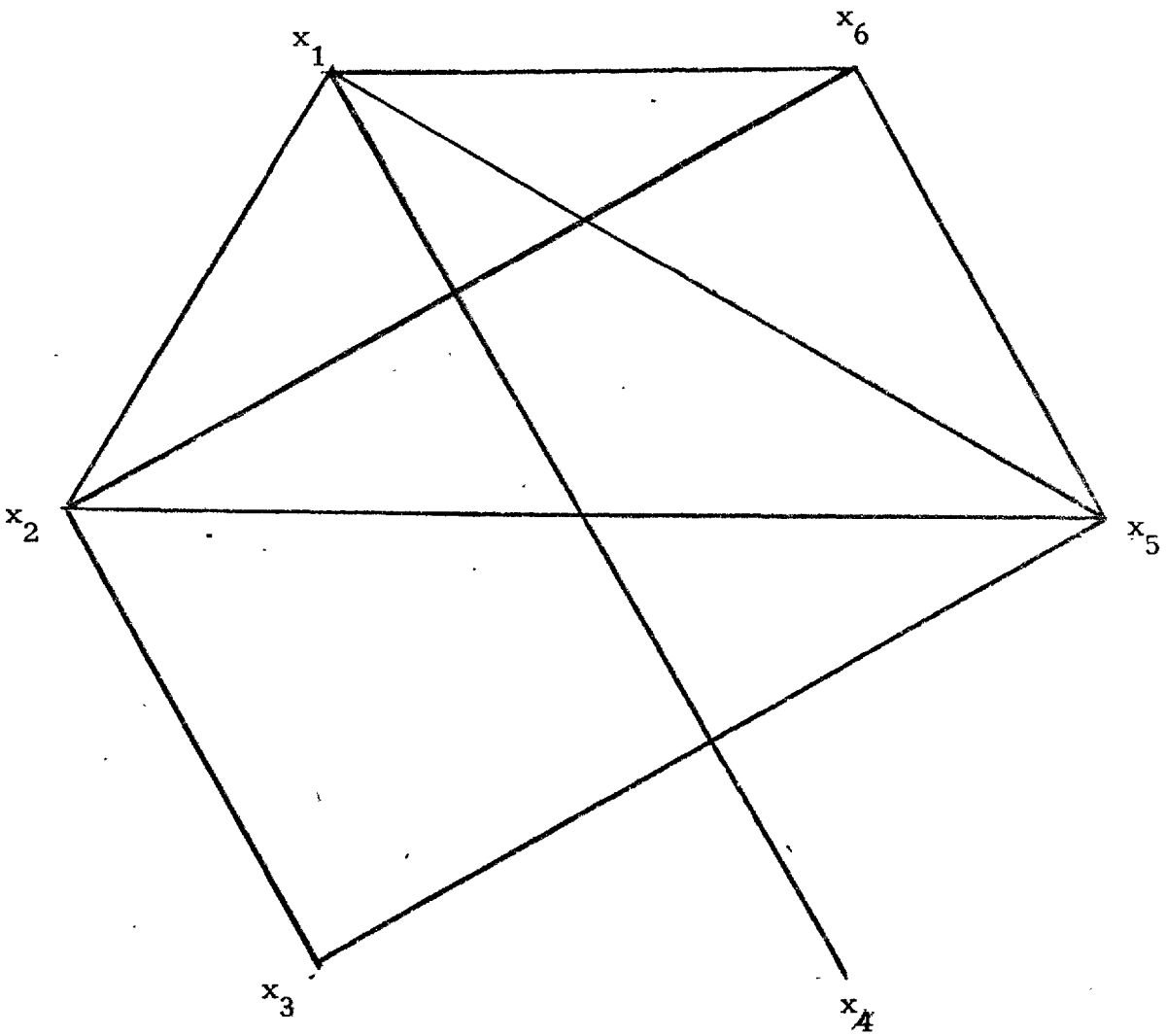


FIGURE 2

Given any graph $G = (X, \Gamma)$, the graph $\bar{G} = (\bar{X}, \bar{\Gamma})$ defined by $\bar{X} = X$ with $\bar{\Gamma}$ defined by $\bar{\Gamma}x_i = X - (\Gamma x_i \cup \{x_i\}) \forall x_i \in X$ (so that $x_j \in \bar{\Gamma}x_i$ iff $x_j \notin \Gamma x_i$) is called the complement of G.

Given a graph $G = (X, \Gamma)$, any graph $G' = (X', \Gamma')$, where $X' \subseteq X$ and $\Gamma' \subseteq \Gamma$ is called a subgraph of G, and is denoted by $G' \subseteq G$.* If, in such case, we have additionally that that $\Gamma' x_i = \Gamma x_i \cap X$, $\forall x_i \in X$, then G' is known as an induced subgraph of G. Recalling that for any graph $G' = (X', \Gamma')$ the mapping Γ' is defined only for the set X', we see that an induced subgraph with the vertex set $X' \subset X$ contains the most arcs of any subgraph of G with X' as vertex set, and is unique, being the graph remaining when the vertices of $X - X'$ and all arcs incident to these vertices are removed from G. Putting $X'' = X - X'$, the induced graph can thenequivalently be represented as $G' = ((X - X''), \Gamma')$, or, by a slight abuse of notation, as $G - X''$. (Note: the latter representation will be generally used through this work as well as its converse concept, i.e. representing G as $G' \cup X''$ where $\Gamma \cap X''$ must however be formally defined).

Example: Referring again to the graph $G = (X, \Gamma)$ of Figure 2, the graph $G' = (X', \Gamma')$ where $X' = \{x_1, x_2, x_3, x_5\}$ and Γ' is defined by $\Gamma' x_1 = \{x_2, x_5\}$; $\Gamma' x_2 = \{x_1, x_3\}$,

---

* Although this is similar to the subset notation used herein, the context will generally make it clear whether a subset or a subgraph is being defined.

$\Gamma' x_3 = \{x_2, x_5\}$, $\Gamma' x_5 = \{x_1, x_3\}$, is a subgraph of G; as is the graph $G'' = (X'', \Gamma'')$ where $X'' = \{x_2, x_4, x_5, x_6\}$ with $\Gamma'' x_2 = \{x_6\}$, $\Gamma'' x_4 = \phi$, $\Gamma'' x_5 = \phi$; $\Gamma'' x_6 = \{x_2\}$. Neither is an induced subgraph of G, due to the absence in each case of the arc (x_2, x_5). However the graph $\tilde{G} = (\tilde{X}, \tilde{\Gamma})$ obtained by adjoining the arc (x_2, x_5) to G' is an induced subgraph of G. We then have $\tilde{G} \equiv G - \{x_4, x_6\}$ and correspondingly $G \equiv \tilde{G} \cup \{x_4, x_6\}$, with $\Gamma x_4 = \{x_i\}$, $\Gamma x_6 = \{x_1, x_2, x_5\}$.

Finally, the graph $\bar{G}$ of Figure 3 is the complement of G.

A path between vertices x_i and x_j is a sequence of arcs of the form (x_i, x_k), (x_k, x_ℓ), $\cdots$ (x_s, x_t), (x_t, x_j), beginning at x_i and ending at x_j, with consecutive arcs being incident to a common vertex. The length of a path is the number of arcs that constitute the path. A cycle is a path for which the start and finish vertices are coincident (i.e. $x_i \equiv x_j$ in the illustration above).

A graph $G = (X, \Gamma)$ is said to be connected if there exists a path between any two vertices of G. A tree is a connected graph which contains no cycles. An articulating set of a connected graph $G = (X, \Gamma)$ is a set, X', of vertices whose removal disconnects G (i.e. $G - X'$ is not connected). A component of G is a maximal connected subgraph of G, (i.e. a connected subgraph of G which is not a subgraph of any other connected

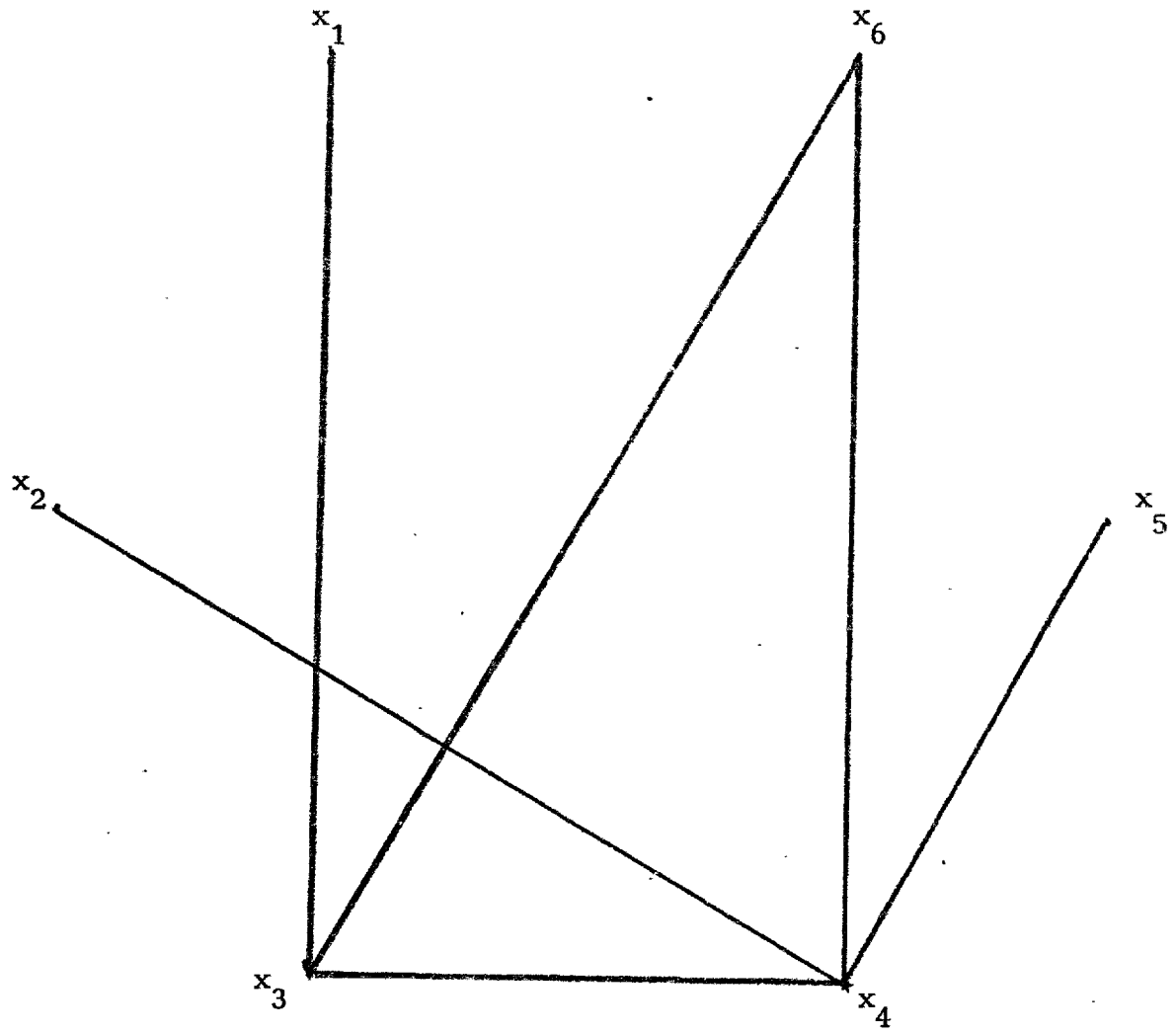


FIGURE 3

subgraph of G). Thus a graph G is connected if and only if, it consists of one component. A graph G is said to be planar if it can be drawn on a plane in such a way that arcs meet only at their (co-incident) end vertices.

Example: In the graph G of Figure 2, the sequence $P = (x_4,x_1), (x_1,x_6), (x_6,x_2), (x_2,x_5)$ is a path of G of length 4 joining x_4 and x_5, while $(x_2,x_6), (x_6, x_5), (x_5, x_2)$ is a cycle of length 3. Also G is a connected graph, while $\{x_1\}$ is an articulating set, as removing x_1 'disconnects' x_4. Finally G is planar as can be seen by moving x_4 above x_1 and moving x_6 half-way to x_2. Similarly, the graph $\bar{G}$ of Figure 3 is planar, as can be easily seen by moving x_2 below x_4.

We turn now to some definitions which are more closely related to the colouring of graphs.

2.2 Colouring-related Definitions and Elementary Consequences

An independent set of $G = (X, \Gamma)$ is a set of vertices of G, no two of which are adjacent. An independent set $X' \subseteq X$ is a maximal independent set, denoted MIS, if there is no independent set X'' with $X' \subset X'' \subseteq X$.

A complete set of $G = (X, \Gamma)$ is a set of vertices of G each pair of which are adjacent.* A maximal complete set so called a

---

* More fully, such a set is often called a completely connected set.

clique. It is seen then that a clique of G is an MIS of $\bar{G}$, its complement graph, and vice versa.

A colouring of a graph G is a manner of assigning colours to the vertices of G so that adjacent vertices do not acquire the same colour. This is done by defining a colouring function $C(\cdot)$ on the vertices of G, where $C(x_i) = j$ if and only if x_i is coloured with colour j. Two vertices which are assigned the same colour are said to be co-coloured. The set, S_j, of all vertices coloured by colour j is sometimes referred to as a coloured set.

A graph is said to be r-colourable if it can be coloured with r (or less) colours. The chromatic number of a graph G, denoted by $\gamma(G)$, is the minimum number colours required to colour G. If $\gamma(G) = q$ then G is said to be q-chromatic. (Thus if G is q-chromatic, it is q-colourable $\forall\, r \geq q$). The problem of determining the chromatic number, along with a corresponding colouring function, is known as the Graph-Colouring Problem, denoted GCP.

The independence number of G, denoted $\alpha(G)$, is the number of vertices in the largest MIS of G. The clique-number of G, denoted $\omega(G)$, is the number of vertices in the largest clique of G.

Example: For the graph G of Figure 2, $\{x_3, x_6\}$ is an independent set while $\{x_3, x_4, x_6\}$ is an MIS of G; and $\alpha(G) = 3$. Similary, $\{x_1, x_2\}$ is a complete set while both $\{x_1, x_2, x_5, x_6\}$ and $\{x_1, x_4\}$ are cliques of G; and $\omega(G) = 4$.

An optimal colouring of G can be obtained by defining

$C(x_1) = C(x_3) = 1$, $C(x_2) = C(x_4) = 2$, $C(x_5) = 3$,

$C(x_6) = 4$, so that G is 4-chromatic, i.e. $\gamma(G) = 4$.

Before proceeding further, we first discuss some consequences of the above definitions, which, although elementary will be of great use to us in later chapters.

1. Since, in any colouring of $G = (X, \Gamma)$, no co-coloured vertices can be adjacent, all vertices of any clique must be assigned different colours. Thus it is easily seen that $\gamma(G) \geq \omega(G)$. (1)

2. Similarly, any set of co-coloured vertices must be an independent set and moreover it is apparent that $\gamma(G) \geq \frac{n}{\alpha(G)}$, $(n = |X|)$ (2)

3. Finally, it is seen that no connected set of G can have two or more vertices in common with any independent set, (and vice versa) i.e. $|K \cap S| \leq 1$ (3)
for any complete set, K, and independent set, S, of G.

2.3 Some Known Results on Colouring

2.3.1 Further Bounds on the Chromatic Number

In addition to the two simple bounds of the previous section, there have been other bounds established for the chromatic number of a graph $G = (X, \Gamma)$ with n vertices and m arcs, which we list below, without proof [1, 3, 4, 14]

(i) $\gamma(G) \geq n^2 / (n^2 - 2m)$ (4)

(ii) $\gamma(G) \leq 1 + (2m(n-1) / n)^{\frac{1}{2}}$ (5)

(iii) $\gamma(G) \leq \max_{x_i \in X} \{d(x_i)\} + 1$ (6)

$$\text{(iv)} \quad \lfloor 2\sqrt{n} \rfloor \leq \gamma(G) + \gamma(\bar{G}) \leq n + 1 \qquad (7)$$

$$\text{(v)} \quad n \leq \gamma(G) \cdot \gamma(\bar{G}) \leq \lceil (n+1/2)^2 \rceil \ * \qquad (8)$$

2.3.2 Comments on the Previous Bounds

That (1) can be a very poor bound is seen from either of the following 'complementary' theorems.

Theorem 1 [12]: For any $q \geq 0$, there exists a graph G with $\omega(G) = 2$ for which $\gamma(G) > q$.

Theorem 2 [19]: For any $q \geq 0$, there exists a graph G with $\gamma(G) = q$ for which $\omega(G) = 2$.

On the other hand, (5) can be strengthened as follows.

Theorem 3 [4]: Let $h = \max_{x_i \in X} \{d(x_i)\}$. Then G is h-colourable unless (i) G contains a clique of cardinality $(h + 1)$, or (ii) $h = 2$ and G contains a cycle of odd length.

A further similar result is the following. [17, 21]

Theorem 4 [17]; If Φ is a real-valued function on G satisfying

(a) $G' \subset G \Rightarrow \Phi(G') \leq \Phi(G)$

(b) $\Phi(G) > \min_{x_i \in X} \{d(x_i)\}$

then $\gamma(G) \leq 1 + \Phi(G)$

Thus putting $\Phi(G) = \lambda(A)$ where $\lambda(A)$ is the largest eigenvalue of A, the adjacency matrix of G, gives $\gamma(G) \leq 1 + \lambda(A)$ (9)

Similarly we have: $\gamma(G) \leq 1 + \max_{G' \subset G} \{\min_{x_i \in G'} \{d(x_i)\}\}$ (10)

A bound somewhat similar to the latter is the following.

---

* $\lfloor x \rfloor$ represents the greatest integer not less than x.
$\lceil x \rceil$ represents the least integer not greater than x.

Theorem 5 [20] : Suppose $C(x_i)$ defines a q-colouring of $G = (X, \Gamma)$, and let

$$S_i = \{x_i \in X \mid C(x_i) = i\}, \quad 1 \leq i \leq q .$$

Put $d_k = \max_{x_i \in S_k} \{ d(x_i) \}$.

Then $\chi(G) \leq \max_{k \leq q} \{ \min \{ k, d_k + 1 \} \}$. (11)

2.3.3 Two Classical Theorems

Finally, we quote two long-established results on graph-colouring. Although they concern certain classes of graph (and are not in that sense general), the former represents the main result yet known concerning the characterization of certain colourable graphs, while the latter is related to the most famous conjecture of Graph Theory.

Theorem 6 [8] : A graph is 2-colourable if and only if it contains no cycles of odd length.

Theorem 7 [9] : Any planar graph is 5-colourable.

Indeed it has long been a famous conjecture (the Four-Colour Conjecture) that this latter result can be strengthened to state that every planar graph is 4-colourable, which, in view of the fact that the complete graph on 4 vertices is planar, would be the best possible. We shall return briefly to this theme in the final chapter.

3. ALTERNATIVE FORMULATIONS OF THE GRAPH COLOURING PROBLEM

3.1 Formulation of a General (0,1) Integer Program

Like many problems in computational graph theory, the GCP can be formulated as a (0,1) Integer Programming (IP) problem. This is accomplished by deriving linear inequalities equivalent to the conditions pertaining to feasible colouring and then seeking to minimize an appropriate objective function.

One of the earliest such formulations is given by Berge [2], where a sequence of IP's is required to be solved. Thus, suppose one is seeking to determine if a 'colouring' of G which uses q colours is feasible. For each colour $r = 1,2,\ldots q$ and each vertex x_i, $i = 1,\ldots n$, define the set of variables z_{ir} by

$$\left.\begin{array}{l} z_{ir} = 1 \quad \text{if } C(x_i) = r \\ \quad\ = 0 \quad \text{otherwise} \end{array}\right\} .$$

Then any assignment of integers to the z_{ir} which satisfies the following three conditions implies a feasible q-colouring of G.

$$\sum_{r=1}^{q} z_{ir} = 1, \quad \text{for all } i = 1,\ldots n \tag{12}$$

$$\sum_{i=1}^{n} b_{ki} z_{ir} \leq 1 \text{ for all } k = 1,\ldots m;\ r = 1,\ldots q \tag{13}$$

where $B = [b_{ki}]$ represents the incidence matrix of G, and m is the number of arcs in G, and

$$z_{ir} \in \{0, 1\} \qquad i = 1,\ldots n;\ r = 1,\ldots q \tag{14}$$

This is so, since (12) represents the fact that each vertex of G is assigned exactly one colour, while (13) expresses the condition that adjacent vertices of G cannot be coloured the same. Then, by finding the least value of q for which the above three conditions are satisfied, determines this value as the chromatic number of G.

This formulation, involving a series of programs containing mq + n constraints (for various q), is very unwieldly and indeed has been improved upon in [5].

Thus the system of mq inequalities (13) is replaced by the nq inequalities:

$$L\,(1 - z_{ir}) - \sum_{j=1}^{k} a_{ij}\, z_{jr} \geqslant 0 \qquad i = 1,\ldots n,\ r = 1,\ldots q \qquad (13')$$

where L is a constant greater than n and $A = [a_{ij}]$ represents the adjacency matrix of G. For, if for any i, $z_{ir} = 1$, then $\sum a_{ij}\, z_{jr}$ must be 0, so that no vertex adjacent to x_i can also be coloured r, should x_i be coloured r; on the other hand if $z_{ir} = 0$, since $L > n$, any or all of the other vertices of G can be coloured r if x_i is not coloured r. (Of course the corresponding inequality for some other i may well preclude some of these combinations)

Thus, for the graph G of Figure 1, we could let vertices 1,2,..,7 correspond to subjects C,E,B,L,P,M,F and we could similarly assign labels to the arcs, for example

$u_1 = (1,2)$, $u_2 = (1,3)$, $u_3 = (1,5)$, $u_4 = (1,6)$, $u_5 = (2,4)$ $u_6 = (2,6)$, $u_7 = (2,7)$, $u_8 = (3,4)$, $u_9 = (3,5)$ $u_{10} = (4,7)$, $u_{11} = (5,6)$, $u_{12} = (6,7)$.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
u_1	1	1					
u_2	1		1				
u_3	1				1		
u_4	1					1	
u_5		1		1			
u_6		1				1	
u_7		1					1
u_8			1	1			
u_9			1		1		
u_{10}				1			1
u_{11}					1	1	
u_{12}						1	1

FIGURE 4

The corresponding incidence matrix, B, will then be as shown in Figure 4 and a typical inequality of system (13), e.g. $k = 9$, $r = 2$ (taking $q = 4$) is $z_{32} + z_{52} \leq 1$.

Again, the adjacency matrix, A, is as shown in Figure 5, so that a typical inequality of system (13) e.g. $i = 3$, $r = 2$ (again taking $q = 4$ and $L = n + 1 = 8$) is:

$$8 - 8 z_{32} - z_{12} - z_{42} - z_{52} \geq 0 .$$

In either case, the first equation of type (12) would (for example) be $z_{11} + z_{12} + z_{13} + z_{14} = 1$ and the optimal solution given earlier for graph G would correspond to the values of the variables in the IP being as follows:

$$z_{11} = z_{41} = z_{32} = z_{62} = z_{23} = z_{53} = z_{74} = 1$$

all other z_{ir} $(1 \leq i \leq 7, \ 1 \leq r \leq 4)$, being 0. Note that for $q = 3$, no feasible solution exists for either of the IP formulations above.

Also, a formal objective function can be introduced [5], viz:

Minimise $z = \sum_{r=1}^{q} \sum_{i=1}^{n} h_r z_{ir}$, where $h_1 = 1$, $h_{r+1} > nh_r$.

This has the effect that no colour r will be considered should colours 1 to (r-1) be sufficient.

Although $nq < mq$, the above IP formulation is however still not a practicable method for solving the GCP, this being mainly due to the fact that the number of variables representing even moderately sized graphs is still too large to be efficiently solved as a general (0,1) IP, (e.g. [18]). For example, a graph with 30 vertices and chromatic number 6 would require 180 variables.

Therefore, in the next section we present a formulation as a (0,1) IP of a more specific type as a more tractable method for the GCP.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1		1	1		1	1	
x_2	1			1		1	1
x_3	1			1	1		
x_4		1	1				1
x_5	1		1			1	
x_6	1	1			1		1
x_7		1		1		1	

FIGURE 5

3.2 Formulation as a Set Partitioning / Covering Problem

We have seen previously (in section 2.2) that in any feasible colouring of G, the set of vertices which are coloured the same must be an independent set of G, so ensuring that no adjacent vertices are co-coloured. This leads naturally to the following formulation of the GCP as a (0,1) programming problem: Suppose all independent sets $S_1, S_2, \ldots S_t$ of G have been enumerated and define the 0-1 integer variables w_j, $j = 1,\ldots t$ and e_{ij}, $i = 1,\ldots n$, $j = 1,\ldots t$ respectively, as: $w_j = 1$ if set S_j is part of a feasible colouring of G, i.e. if S_j represents all the vertices of a certain colour

$= 0$ otherwise

$e_{ij} = 1$ if vertex x_i is in set S_j

$= 0$ otherwise

Then it is obvious that the GCP is equivalent to the IP

$$\text{Minimise} \quad z = \sum_{j=1}^{t} w_j$$

$$\text{s.t.} \quad \sum_{j=1}^{t} e_{ij} w_j = 1 \qquad i = 1,\ldots n$$

$$w_j = 0,1 \qquad j = 1,\ldots t \qquad (15)$$

For example, for the graph G of Figure 1 there are 16 independent sets as follows: $S_1 = \{1\}$, $S_2 = \{1,4\}$, $S_3 = \{1,7\}$, $S_4 = \{2\}$, $S_5 = \{2,3\}$, $S_6 = \{2,5\}$, $S_7 = \{3\}$, $S_8 = \{3,6\}$, $S_9 = \{3,7\}$, $S_{10} = \{4\}$, $S_{11} = \{4,5\}$, $S_{12} = \{4,6\}$,

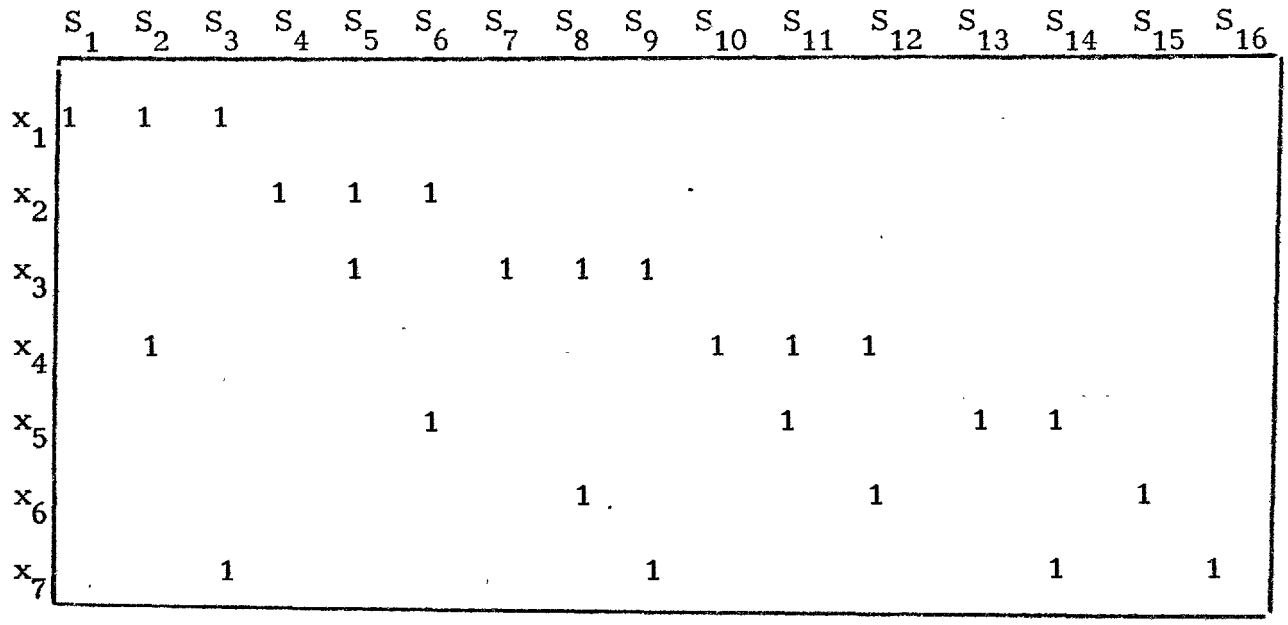

S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12 S13 S14 S15 S16
x1 1 1 1
x2 1 1 1
x3 1 1 1 1
x4 1 1 1 1
x5 1 1 1 1
x6 1 1 1
x7 1 1 1 1

FIGURE 6

$S_{13} = \{5\}$, $S_{14} = \{5,7\}$, $S_{15} = \{6\}$, $S_{16} = \{7\}$;

so that $t = 16$; and the matrix $[e_{ij}]$ would appear as shown in Figure 6.

The optimum solution mentioned earlier would then correspond to $w_2 = w_6 = w_8 = w_{16} = 1$; all other $w_j = 0$, with $z = 4$, (i.e. the vertex sets S_2, S_6, S_8 and S_{16} are coloured with colours 1 to 4 respectively.

The above is a special (unicost) case of the Set Partitioning Problem (SPP) which we consider further in the following chapter. This formulation of the GCP as an SPP is rather more efficient than the general IP formulation of the previous section, mainly due to the special structure of the SPP which can be exploited. On the other hand, the formulation involves the determination of the family of independent sets of the graph G; a problem similar to one which we shall go into more deeply in a later chapter. Moreover, the number of variables is generally larger than that needed for the IP formulation.

This latter situation can be somewhat alleviated by the use of a well-known theorem [5], which we state formally in another chapter, viz. that only maximal independent sets (MIS) of G need be considered as potential colouring sets, although in this case, some vertices may be 'over-coloured' by being contained in more than one MIS present in the colouring of G. For such vertices we could make an (arbitrary) choice of one colour among all such over-colouring.

The above analysis leads to the following formulation as a unicost version of the well-known Set Covering Problem:

Suppose that $M_1, M_2, \ldots M_p$ represent the family of MIS of G and, for each M_j define the 0-1 variable y_j as:

$y_j = 1$ if M_j is present in the optimal colouring of G

$= 0$ otherwise

Again, with $e_{ij} = 1$ if $x_i \in M_j$

$= 0$ otherwise , we have the following program :

$$\text{Minimise } z = \sum_{j=1}^{p} y_j$$

$$\text{s.t.} \sum e_{ij} y_j \geq 1 \qquad i = 1,\ldots n$$

$$y_j = 0,1 \qquad j = 1,\ldots p \qquad (16)$$

where the inequalities correspond to possible over-colourings.

Thus, for the example of Figure 1, there are 9 MIS, viz.
$M_1 = \{1,4\}$, $M_2 = \{1,7\}$, $M_3 = \{2,3\}$, $M_4 = \{2,5\}$, $M_5 = \{3,6\}$
$M_6 = \{3,7\}$, $M_7 = \{4,5\}$, $M_8 = \{4,6\}$, $M_9 = \{5,7\}$ so that
$p = 9$; with the matrix $[a_{ij}]$ being as shown in Figure 7. We note here that the optimum solution given earlier can be derived from either of the following three solutions to the SCP:

(a) $y_1 = y_2 = y_4 = y_5 = 1$; $y_3 = y_6 = y_7 = y_8 = y_9 = 0$

(b) $y_1 = y_4 = y_5 = y_6 = 1$; $y_2 = y_3 = y_7 = y_8 = y_9 = 0$

(c) $y_1 = y_4 = y_5 = y_9 = 1$; $y_2 = y_3 = y_6 = y_7 = y_8 = 0$

these corresponding respectively to the over-colouring of vertices 1,3, and 5.

This latter formulation is considerably better than those mentioned earlier, and, as we shall see later, can in fact serve as a viable method of

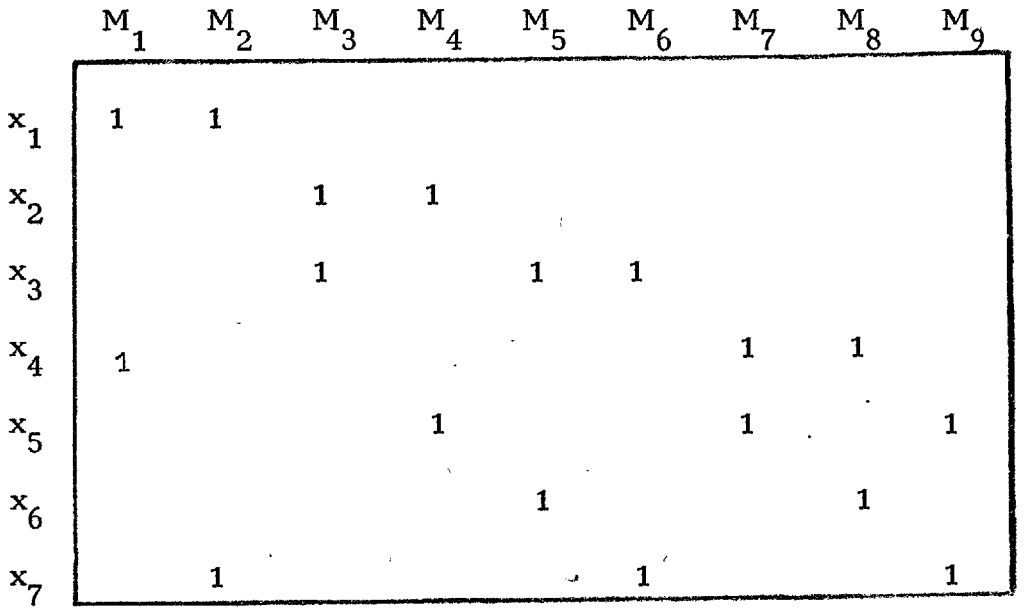

M1 M2 M3 M4 M5 M6 M7 M8 M9
x1 1 1
x2 1 1
x3 1 1 1
x4 1 1 1
x5 1 1 1
x6 1 1
x7 1 1 1

FIGURE 7

solution for the GCP. In fact, since the (general) SCP itself has many applications, and is of great importance in Operational Research we shall devote the whole of the following chapter to this problem in its own right. The solution of the GCP by means of the SCP will then follow naturally as a specific application of the (unicost) SCP.

It would of course be surprising if the best way of solving the GCP were to convert it to one of the more general problems as given above, and in Chapters 5 and 6 we shall deal in great detail with 'direct' tree search methods for solving the GCP.

3.3 Other Concepts Related to Graph Colouring

Apart from the tree-search methods just mentioned, there are also various other concepts which have been used to study graph-colouring, notably the concepts of chromatic polynomials [10,15] and Grundy functions [2,11].

Thus, the chromatic ploynomial $P(G;q)$ for a graph G corresponds to the total number of different ways whereby G can be (feasibly) coloured by using at most q colours. If N_p is the number of ways of (feasibly) colouring G by using exactly p colours (so that $N_1 = 0$ and $N_n = n!$) then it is seen that $P(G;q) = N_1 q + N_2 C_2^q + \ldots\ldots + N_n C_n^q$, a polynomial of degree n, where $n = |X|$ and where C_i^q represents the number of ways of choosing i out of q colours. Although we do not go into any detail here, the coefficients of the various factors q^i can be obtained using the combinatorial principle of inclusion and exclusion (e.g. [10]).

It is then obvious that $P(G;q) = 0$ for $q < \delta(G)$ as there are, by definition, no feasible colourings of G using less that $\gamma(G)$ colours. Hence $\gamma(G)$ is in fact, the least integer q for which

$P(G;q)$ is positive, and so $\gamma(G)$ can - theoretically - be obtained in this way.

Again, a Grundy function $g(x_i)$ is defined on the vertices of a graph G by putting $g(x_i)$ as the smallest positive integer which does not appear in the set $g(\Gamma x_i) = \{g(x_j) \mid x_j \in \Gamma x_i\}$.

That is, $g(x_i) = \min\{k \mid k \neq g(x_j)\ \forall x_j \in \Gamma x_i\}\ \forall x_i \in X$.

It is then apparent that, for any vertex x_i, $g(x_i) \leq d(x_i)+1$, the degree of x_i, and that there are, in general, various different Grundy functions for any graph G.

Also, it is seen that a graph is q-colourable if and only if it possesses a Grundy function such that $\max_{x_i \in X} \{g(x_i)\} \leq q-1$, since any Grundy function on G leads to an obvious feasible colouring for G; so that a study of Grundy functions could be useful in relation to graph-colouring.

However, as it seems unlikely at present that these approaches will lead to efficient graph-colouring algorithms, we do not further discuss them in this work.

4. ELEMENTARY REDUCTIONS IN PROBLEM SIZE

In this section we describe some simple and well-known results (e.g. [3,16]) which often enable us to reduce the amount of work involved in colouring graph G.

(i) Suppose x_s and x_t are two vertices of the graph $G = (X, \Gamma)$ with the property that $\Gamma x_s \subseteq \Gamma x_t$. That is, the set of vertices to which x_s is adjacent is a subset of the set of vertices to which x_t is adjacent.

(Note that this implies, in particular, that x_s is not adjacent to x_t, for otherwise $x_t \in \Gamma x_s$ & $x_t \notin \Gamma x_t$ contradict $\Gamma x_s \subseteq \Gamma x_t$). It is then readily seen that any colour which is feasible for vertex x_t (in the course of any algorithm) would also be feasible for x_s, so that we need only in fact use the algorithm to optimally colour the (sub)graph $G' = (X', \Gamma)$ where $X' = X - \{x_s\}$, and then assign the colour $C(x_t)$ to x_s; thereby reducing the computation time.

(ii) Suppose now that the graph G contains an articulating set which is a clique, Q, of cardinality p, say. That is removing Q from the graph will divide the graph into two, say, disjoint components (see section 2.1), $G_1 = (X_1, \Gamma_1)$ and $G_2 (X_2, \Gamma_2)$, where $X_1 \cup X_2 = X - Q$, and (a) $\Gamma_1 x_i \cap X_2 = \phi \quad \forall x_i \in X_1$

(b) $\Gamma_2 x_j \cap X_1 = \phi \quad \forall x_j \in X_2$

Therefore the colouring of X, is unaffected by the colouring of X_2, and vice versa.

Consider now the graphs $G' \equiv (X_1 \cup Q, \Gamma)$ and $G'' \equiv (X_2 \cup Q, \Gamma)$ where Γ is restricted in each case to the appropriate vertex set $(X_i \cup Q)$.

Since both G' and G'' contain a clique Q of cardinality p, we have $\chi(G') \geq p$, $\chi(G'') \geq p$. We can then colour each of G' and G'' by assigning colours $1,2,\ldots,p$ to the vertices of Q and then colouring the remainder of the graph relative to these assignments. Since, as noted above, the colouring of $X_1 - Q$ does not effect that of $X_2 - Q$ the colourings of G' and G'' can then be 'fitted together' - overlapping

at Q - to obtain an optimal colouring of G so that
$\gamma(G) = \max(\gamma(G'), \gamma(G''))$.

Thus, in this case, the problem of optimally colouring G is reduced to the simpler problems of colouring G' and G", and obviously an analagous reasoning applies should Q divide G into more than two components.

We note here that two results often used for graph reduction, viz (a) that components of a graph can be coloured independently, and (b) that bi-connected components of a graph can be independently considered [16], are simply the cases $p = 0$ and $p = 1$ of the above results.

Before concluding this chapter, we indicate here the main notational conventions which are adopted throughout the remainder of this work (except where explicitly otherwise stated).

Thus, upper-case letters, will, in general, refer to sets of elements except for F, which refers to a family of sets; G and H, which refer to graphs; and C, which is the colouring function previously defined. In particular, when dealing with graph-theoretic concepts, X will denote the vertex set of a graph G, S an independent set of G, M a maximal independent set of G and Q a clique of G. Also G' will commonly denote a subgraph of G.

Again, lower-case letters will in general refer to elements of sets. Specific uses of certain letters include using a-c to denote general colours, using h-l as indices, and subscripts, etc., using r-t to indicate various branching states or stages of algorithms, and p-q to denote the total number of colours used in graph-colouring algorithms; while, in graph - theoretic context, x will refer to a general vertex, u to a general arc, and n to the number of vertices of the graph $G = (X, \Gamma)$.

In addition to using F to denote families of sets, curly letters will also be used to denote families consisting of sets represented by the appropriate letter. Specifically, $\mathcal{M}, \mathcal{Q}, \mathcal{S}$, will refer to families of maximal independent sets, cliques and independent sets, respectively when referring to graphs.

Finally, as previously noted, the use of set-theoretic symbols $\subset, \cap, \cup, -, \bar{}$, representing the concepts of subsets, intersection of sets, union of sets, removal of sets, and complementing of sets, respectively, have been extended to the case of graphs as well.

Lastly, we briefly mention the general layout of this thesis. Thus, in each chapter the theorems and equations each run consecutively from 1, and in general these are referred to only in their own chapters so, for example, equation (10) will refer to equation (10) of that chapter. When occasion demands use of a Theorem (or equation) from a previous chapter, we preface the number of the Theorem by the number of the chapter in which it is established, so that Theorem 4-2 refers to Theorem 2 of Chapter 4.

We are now ready proceed to the next chapter.

5. REFERENCES

1. BEHZAD, M. and CHARTRAND, G. Introduction to the Theory of Graphs, Allyn and Bacon, Boston, Mass. (1971).

2. BERGE , C. Theory of Graphs, Methuen, London (1962).

3. BERGE , C. Graphs and Hypergraphs, North Holland Publishing Company, Amsterdam (1973).

4. BROOKS, R.L. "On Colouring the Nodes of a Network", Proc. Cambridge Philosophical Soc. 37 (1972), pp 194-197.

5. CHRISTOFIDES, N. Graph Theory, (in press).

6. DEMPSTER, M.A.H. "Two Algorithms for the Time-table Problem", in Combinatorial Mathematics and Its Applications (WELSH, ed) Academic Press, London (1969) pp 63-85.

7. EILON, S. and CHRISTOFIDES , N. "The Loading Problem", Management Science 17 (1971) pp 259-268.

8. HEAWOOD, P.J. "Map-Colour Theorems", Quarterly Journal of Mathematics 24 (1890), pp 332-339.

9. KONIG, D. Theorie der Endlichen un Urendlichen Graphen, Chelsea, N.Y. (1950).

10. LIU , C.L. Introduction to Combinatorial Mathematics, McGraw - Hill, N.Y. (1968).

11. MATULA, D.W. "Bounded Colour Functions on Graphs", Networks 2 (1970), pp 29-42.

12. MYCIELSKI, J. "Sur le Coloriage des Graphes", Colloq. Math 3 (1955), pp 161-162.

13. NEUFIELD, B.A. and TARTAR, J. "Graph Colouring Conditions for the Existence of Solutions to the Time-table Problem", Communications of the A.C.M. 17 (1974), pp 450-453.

14. NORDHAUS, G.A. and GADDUM, J.W. "On Complementary Graphs", American Mathematical Monthly 63 (1956), pp 175-177.

15. READ, R.C. "An Introduction to Chromatic Polynomials", J. of Combinatorial Theory 4 (1968), pp 52-71.

16. ROSCHKE, S.I. and FURTADO, A.L. "An Algorithm for Obtaining the Chromatic Number and An Optimal Colouring of a Graph", Information Processing Letters 2 (1973), pp 34-38.

17. SZEKERES, G. and WILF, H.S. "An Inequality for the Chromatic Number of a Graph", J. Combinatorial Theory 4 (1968), pp 1-3.

18. TRYPIA, M.N. "Methods for Integer Programming", Ph.D. Thesis, Dept. of Management Science, Imperial College, London (1975).

19. TUTTE, W.T. "Solution to Advance Problem No. 4526", American Mathematical Monthly 61 (1954), p 352.

20. WELSH, D.J.A. and POWELL, M.B. "An Upper Bound on the Chromatic Number of a Graph and its Applications to Time-tabling Problems", Computer Journal 10 (1967), pp 85-86.

21. WILF, H.S. "The Eigenvalues of a Graph and its Chromatic Number", J. London Mathematical Society 42 (1967), pp 330-332.

22. WILLIAMS, M.R. "A Graph Theory Model for the Solution of Timetables", Ph.D. Thesis, University of Glasgow (1968).

23. WOOD, D.C. "A System for Computing University Examinations Timetables", Computer Journal 10 (1967), pp 139-141.

CHAPTER II

THE SET COVERING PROBLEM

1. INTRODUCTION

In this chapter we deal with the well-known "Set Covering Problem", (SCP) which can be used as a tool in solving the GCP as mentioned in the previous chapter. The initial formalization of the SCP appears to have been made by circuit theorists involved in the in the simplification of logical (Boolean) expressions, for the construction of optimal logical circuits [14,15,16,17,37,39,40,43, 47,48,62]

However, many other applications of the SCP soon appeared in management science and operations research areas, especially in the fields of scheduling (and in particular aircrew scheduling) [1,2,4,5,20,33,50,57,58,59,60] , assembly line balancing [21,56] and others [17].

For the aircrew scheduling problem, for example, a set of 'flight legs', which must be flown by a particular aircraft in a given day, is given, together with a possible set of aircraft crews. Each crew has associated with it a particular cost (e.g. wages); and a compatibility table of crews to flight legs is also given (e.g. a crew at station A cannot be used on a leg B to C at that time). What is then needed is to cover, at minimal cost, all the flight legs using a (compatible) subset of the crews. This is an example of the general SCP, where crews are represented by columns of a matrix, 'flight legs' are represented by rows, and position (i,j) has an entry of 1 to 0 depending on whether crew j is compatible with leg i or not.

In the case - mentioned earlier - of simplifying a logical expression E, which is (say), given in disjunctive normal form, one need consider only the set, P, of prime implicants of E. A 'simplest' form for E is then obtained by finding the lowest - cardinality subset $P' \subseteq P$ subject to the condition that every

(conjunctive) term of E be 'covered' by (i.e. be implied by) at least one prime implicant in P'. This is an example of a unicost SCP, since any prime implicant has the same 'cost' (e.g. unity), as any other. Although most of the earlier research into the above problem dealt mainly with efficient methods for generating P , it was soon realized that the 'set covering' part of the problem was not trivial.

Other applications reported in the literature include the vehicle dispatching problem [9,18,44] , political districting [22] , information retrieval [19] , and network attack or defense, [10,12,31].

After formally stating the problem, some know techniques for reducing its dimensions are mentioned. Then, in Section 2, three basic exact methods given in the literature for solving SCP's are surveyed:

- combinatorial tree search [24,26,35,45,46]
- a specific type of (primal) integer programming [11,30] , and
- a variant of the classical Boolean algebra approach [13,34] .

We are not concerned here with the variety of herusitic procedures which have also been suggested for solving this problem.

In Section 3, the tree search algorithm is developed further by deriving a new bound and by introducing elements of an incomplete dynamic programming procedure to limit the search. Finally, in Section 4, five methods (four of which can be found in the literature and are variants of the three basic methods discussed in Section 2, together with the improved tree search procedure discussed in Section 3), are coded and their computational performance compared on the same set of unicost test problems. Although, in fact, each one of these five methods could be used to solve the general SCP,

we concentrate on the unicost SCP for the test examples, since the GCP, when solved as an SCP, is of this type.

1.1 Problem Formulation

Given a set $X = \{x_1, \ldots, x_m\}$, and a family $\mathcal{S} = \{S_1, \ldots, S_n\}$ of sets $S_j \subset X$, any subfamily $\mathcal{S}' = \{S_{j_1}, S_{j_2}, \ldots, S_{j_k}\}$ of $\mathcal{S}$ such that

$$\bigcup_{i=1}^{k} S_{j_i} = X \qquad (1)$$

is called a set-covering of X, and the S_{j_i} are called the covering sets. Also, if no $\mathcal{S}'' \subset \mathcal{S}'$ is a set-covering of X, then $\mathcal{S}'$ is called a prime set-covering. If, in addition to equation (1), $\mathcal{S}'$ also satisfies

$$S_{j_h} \cap S_{j_\ell} = \phi \quad \forall h, \ell \in \{1, \ldots, k\} \quad h \neq \ell \qquad (2)$$

i.e. if the S_{j_i} $(i = 1, \ldots k)$ are pairwise disjoint then $\mathcal{S}'$ is called a set-partitioning of X.

If, with each $S_j \in \mathcal{S}$, there is associated a (positive) cost c_j, we may wish to find that set-covering of X which has minimum cost, the cost of $\mathcal{S}' = \{S_{j_1}, \ldots, S_{j_k}\}$ being $\sum_{i=1}^{k} c_{j_i}$. The above minimisation problem is called the Set Covering Problem (SCP), the Set Partitioning Problem (SPP) being correspondingly defined.

In general, the costs c_j are positive integers, but the case often occurs, in practice, where the costs c_j are all equal (and

thus may be taken as unity); this situation giving rise to the unicost SCP/SPP. Although this chapter deals with the general SCP, we shall often make specific reference to the special case of unicost SCP as the SCP corresponding to a GCP is of this form.

Obviously the SCP can also be formulated as 0-1 linear program as:

$$\text{Minimise:} \quad z = \sum_{j=1}^{n} c_j y_j \,.$$

$$\text{subject to:} \quad \sum_{j=1}^{n} a_{ij} y_j \geq 1 \quad i = 1,2,\ldots m. \qquad (3)$$

where $c_j \geq 0$ and y_j is $1,(0)$ depending on $S_j \in \mathcal{S}^1$, $(S_j \notin \mathcal{S}^1)$, and where $a_{ij} = 1,(0)$ depending on $x_i \in S_j$, $(x_i \notin S_j)$.

For the SPP, the inequalities (3) become

$$\sum_{j=1}^{n} a_{ij} y_j = 1 \quad i = 1,2,\ldots m. \qquad (4)$$

1.2 Problem Reduction

Due to the nature of the SCP it is often possible to make certain well-known a priori deductions and reductions e.g. [8,23,24,25] [27,47,48]. Thus

(1) If for some $x_i \in X$, $x_i \notin S_j \quad \forall j = 1,\ldots n$, then the problem has no solution.

(2) If $\exists x_i \in X$ so that $x_i \in S_k$, $x_i \notin S_j$, $\forall j \neq k$, then S_k must be in all solutions and the problem may be reduced by setting $X = X - S_k$ and $\mathcal{S} = \mathcal{S} - \{S_k\}$.

(3) Let $V_i = \{j | x_i \in S_j\}$. Then, if $\exists$ $p,q \in \{1,\ldots m\}$ with $V_p \subseteq V_q$, x_q may be deleted from X, since any set that covers x_p must also cover x_q, i.e. x_q is dominated by x_p.

(4) If, for some family of sets $\bar{\mathcal{S}} \subset \mathcal{S}$ we have $\bigcup_{j \in \bar{S}} S_j \supseteq S_k$ and $\sum_{j \in \bar{\mathcal{S}}} c_j \leq c_k$ for any $S_k \in \mathcal{S} - \bar{\mathcal{S}}$ then S_k may be deleted from $\mathcal{S}$.

Henceforth we shall assume that these reductions, if applicable, have been applied and the original SCP reformulated in its irreducible form.

2. METHODS OF SOLUTION

In this section we classify and discuss various methods proposed in the literature for solving the SCP. Improvements and computational comparisons are given in later sections.

2.1 Tree Search Methods

A. A simple tree search [24, 25]

As noted earlier, the SPP is closely related to the SCP, being essentially an SCP with an additional (no-overcovering) restriction. This (and,in general,any) restriction is advantageous when we attempt to solve the problem by an impicit enumeration (tree search) method, since it may allow the early abandonment of potential branches of the tree. Therefore, we shall first discuss a tree search algorithm for

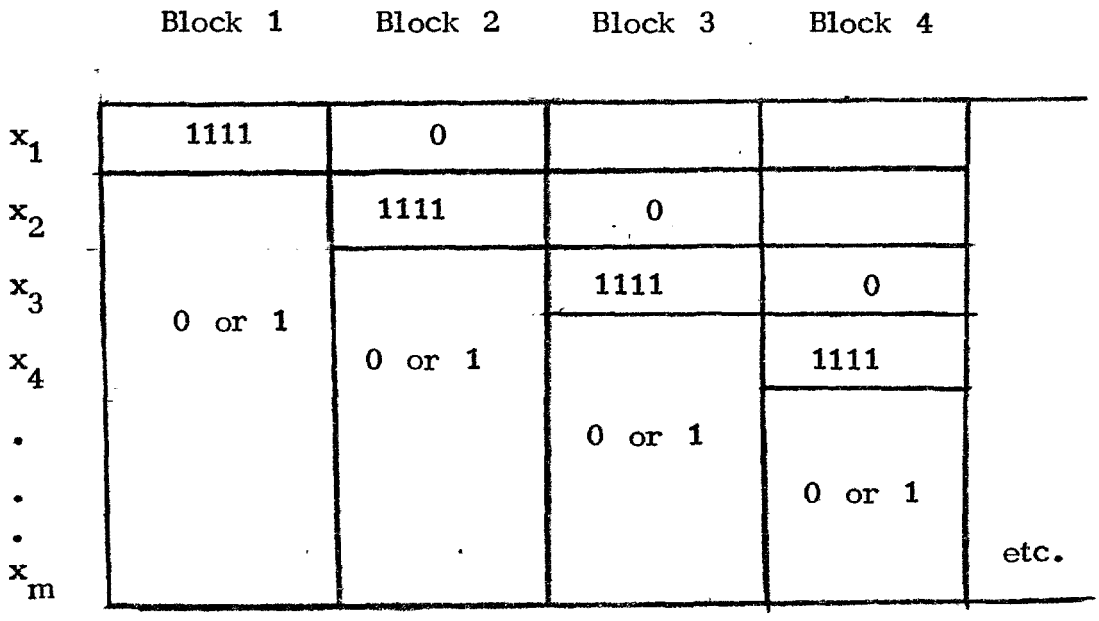
solving the SPP, and then show how it can be extended to solve the SCP.

(i) Solving the SPP

In [24] a tree search method for solving the SPP was proposed by Garfinkel and Nemhauser, and is essentially as follows:

At the start of the process, we made up m 'blocks' of columns one block for each element of X. Each block consists of those sets of $\mathcal{S}$ (represented by columns) which contain that element corresponding to the block, but which contain no lower-numbered elements. Thus block k will comprise of exactly those sets which contain x_k but do not contain any of $x_1, \ldots, x_{k-1}$. Each set, therefore, appears in exactly one block, and the totality of blocks can in general be arranged in tableau form as shown in Figure 8, although some block(s) may be nonexistent in a particular problem.

Figure 8: Initial tableau



During the course of the algorithm, blocks are searched sequentially, with block k not being searched unless every element x_i, $1 \leq i \leq k-1$, has already been covered in the partial solution. Thus, if any set in block k were to contain elements indexed less than k, the set would have to be discarded (at this stage) due to the no-overcovering requirement.

The sets within each block are arranged (heuristically) in ascending order of their cost and the sets are now renumbered so that S_j will from now on be used to mean the set corresponding to the j^{th} column of the tableau.

At any stage of the tree search a current 'best' solution $\bar{R}$ of cost $\bar{z}$ is known, where $\bar{R}$ represents the set of corresponding covering sets. If R and z represent the corresponding values in the current state of the search, and T is a set representing those elements (i.e. rows) x_i covered by the sets in R, then the algorithm proceeds as follows:

1: Initialization. Set up the initial tableau and begin with the partial solution $R = \phi$, $T = \phi$, $x = 0$ and $\bar{z} = \infty$.

2: Augmentation. Find $r = \min \{ i \mid x_i \notin T \}$. Set a marker at the top (lowest cost set) of block r.

3: Branching. Beginning at the marked position in block r examine its sets S_j^r, say, in increasing order of j. If set S_j^r is found such that $S_j^r \cap T = \phi$, and $z + c_j^r < \bar{z}$ (where c_j^r is the cost of S_j^r), go to Step 5. Otherwise, if block r is exhausted, or a set S_j^r reached

for which $z + c_j^r < \bar{z}$ go to Step 4.

4: Backtrack. R cannot lead to a better solution. If $R = \phi$ (i.e. block 1 has been exhausted), terminate with the optimal solution $\bar{R}$. Otherwise, remove the last set, S_k^ℓ say, added into R, put $r = \ell$, place a marker at set S_{k+1}, remove the previous marker in block ℓ and go to Step 3.

5: Test for new solution. Update $R = R \cup \{S_j^r\}$, $T = T \cup S_j^r$, $z = z + c_j^r$. If $T = X$, a better solution has been found; set $\bar{R} = R$, $\bar{z} = z$ and go to Step 4. Otherwise, go to Step 2.

As the search terminates with the exhaustion of block 1 (see Step 4 above), it would seem worthwhile to arrange the blocks in ascending order according to the number of columns (sets) in each. This can be achieved by renumbering the elements (rows) $x_1, \ldots, x_m$ in increasing order of number of sets in $\mathcal{S}$ containing that element, before setting up the initial tableau.

A similar algorithm for solving the SPP was also proposed by Pierce [45]. This last algorithm* incorporates additionally a lower bound, h, to the cost of covering the remaining $|X-T|$ elements of a subproblem, and if $z + h \geq \bar{z}$, backtracking can occur. The bound h is calculated by formulating a 'knapsack' problem as follows:

Suppose the search is at a stage where we are considering set

---

* In [45], a slightly different heuristic for the intra-block ordering is used i.e the sets and ordered in increasing cost per number of elements covered.

S_p as a possible augmentation of the partial solution. By summing for the subproblem, the m original constraints given by (4), we get a further single (and redundant) constraint $\sum_{j=p}^{n} \lambda_j y_j = m_p$, where λ_j is the cardinality of set S_j, and $m_p = m - |T|$, is the number of original constraints still to be satisfied. The (one-dimensional) knapsack problem is then as follows:

$$\text{Minimize} \quad \sum_{j=p}^{n} c_j y_j \quad \text{s.t.} \quad \sum_{j=p}^{n} \lambda_j y_j = m_p, \; y_j \in \{0,1\}, \; j = p, \ldots n. \tag{5}$$

A bound, h, is then found either by solving (5) (for example, by dynamic programming), or, at a lesser computational cost, by finding a lower bound to the solution of (5), an obvious one being $h = m_p \cdot \min_{j \geq p} \{c_j / \lambda_j\}$. In [46], Pierce and Lasky show how this bound can be improved by considering the sets of each block separately, (with a different initial ordering of the blocks); and the subsidiary use of an LP is also introduced.

Before showing how this SPP algorithm can be extended for the case of the SCP, we briefly indicate some of the other methods proposed for solving the SPP.

A somewaht different algorithm for the SPP, which also makes use of the block structure, is that ofMarsten [38], which is based on the fact that if $g: X \to K$ is any function from the set, X, of elements to the set, K, of blocks then $\bigcup_{k \in K} g^{-1}(k)$ induces a partition of X. It is shown in [38] that the set of feasible

solutions to the SPP corresponds to a subclass of the class of all such functions g. Since g can be built up stage by stage in terms of those elements not yet assigned to blocks, the algorithm proceeds in the standard tree search manner, but with the additional facility that, at each stage, the corresponding LP is solved to serve both as a bound and as a means of directing the search.

In [41], Michaud describes another implicit enumeration algorithm which is based on an LP problem corresponding to the SPP, with the block structure given earlier being used in a secondary role; while other algorithms involving simplex-type iterations have been proposed, both primal [6,7] and dual [53,54,55] We also point out here that this latter algorithm, which is based on Gomory's All-integer integer programming algorithm [29], can thus in fact solve the SCP as well.

In the case of the unicost SPP, a different bound based on minimal chain decomposition of a directed graph * has been incorporated, [42], into the basic search strategy of [24]. Thus given any (unicost) SPP of r elements x_i, and t covering sets S_j, an r-vertex graph G is constructed with $\{x_1,\ldots,x_r\}$ as the vertex set, with two vertices x_i, x_k $(k>i)$ being adjacent (from i to k) if and only if, there exists some set S_j which contains x_i and x_k but none of $x_{i+1},\ldots, x_{k-1}$. It can be shown, [42], that each set in a feasible set partitioning corresponds to a chain in a minimum chain decomposition but that the reverse is not always true, so that the solution to the minimum chain decomposition problem serves as a lower bound for its corresponding

---

* The minimum chain decomposition problem on a directed graph G with no directed cycles is to decompose G into a minimum number of components each of which is either a chain (a directed path or an isolated vertex, and, there exists a 'good' polynomially bounded (n^3) labelling algorithm for its solution (see [42]).

SPP. Again, as well as using the minimum chain decomposition as a bound for backtracking, in the usual manner, one could also use it to direct the search to choosing those sets S_j which correspond to chains for the next forward branching.

Finally, we briefly mention that Jensen [32] has had success with a dynamic programming method for a certain type of SPP.

(ii) Extension to the SCP

One can think of the SCP for covering the set X with the family of sets $\mathcal{S}$ as an SPP for covering X with a larger family of sets $\mathcal{S}^*$, where $\mathcal{S}^* = \{S_k^* \mid S_k^* \subseteq S_j \in \mathcal{S}\}$ and where, if a specific set S_k^* appears as a subset of more than one S_j, the cost of S_k^* is taken as the minimum cost of all S_j's for which it is a subset. Such a direct transformation of the SCP into an SPP is obviously computationally impractical since the SPP tableau would then contain the order of

$$\sum_{S_j \in \mathcal{S}} \left\{ 2^{|S_j|} - 1 \right\}$$

columns. However, if the partioning requirement, $S_j^r \cap T = \phi$, in Step 3 of the SPP algorithm just described is removed, then it is unnecessary to include very subset of each S_j of the SCP as a separate column of the tableau. Nevertheless, one could still not set up this initial tableau by entering each set S_j in just one block according to the lowest numbered element (say x_{j_1}) of

S_j, as was done previously. This is so because with the no-overcovering restriction removed, it is not now possible to exclude S_j from further consideration if the element x_{j_1} is covered by another set already in the partial solution (as would indeed be done by the SPP algorithm just described). This shortcoming can easily be alleviated by including S_j in each block corresponding to an element of S_j, so that S_j would now appear in $|S_j|$ blocks [45]. Moreover, since an element x_{j_p} of S_j would, by the sequential nature of the search, be covered before branching on the sets of block q $(q > j_p)$ is considered, it is now possible to remove all the elements x_{j_p} from a set S_j when entering S_j into any block $q > j_p$ without affecting the solution of the problem in any way. This apparently trivial removal is in fact computationally very beneficial, since the initial tableau of the SCP now set up can be reduced by the preliminary reduction test of section 1.2 to much smaller dimensions than would have been possible without the removal of the elements x_{j_p} $(j_p < q)$ from the sets of block q.

A variety of tests other than the ones just described could be incorporated into the above basic tree search with very important effects on the computational efficiency of the method. These tests are described in greater detail in Section 3.

B. An LP-directed tree search

Lenke, Salkin and Spielberg [35] give another implicit enumeration algorithm in which a linear program is solved at any node of the search tree. The solution of this LP is then used as a

lower bound during the search and is also used in determining the next forward tree branching from the current node.

Thus, suppose the search has reached some node k and let R_k be the current partial solution (i.e. the set of indices of those variables fixed to either 0 or 1); F_k be the set of indices of free (as yet unspecified) variables; and Q_k be the set of indices of those constraints not yet satisfied by R_k. At this stage we have the (covering) subproblem P_k:

$$\text{Min } z = \sum_{j \in F_k} c_j y_j \quad \text{s.t.} \sum_{j \in F_k} a_{ij} y_j \geq 1 \quad \forall i \in Q_k, \; y_j \in \{0,1\} \qquad (P_k)$$

If P_k' represents the continuous version of P_k (i.e. with the conditions of $y_j \in \{0,1\}$ replaced by $0 \leq y_j \leq 1$), then, if the solution to P_k' is infeasible, or its cost exceeds $\bar{z} - z - 1$, backtracking can occur. Also, if the solution of P_k' (say y') is all-integer, a better solution is immediately obtained by setting to 1 those variables in R_k already fixed to 1, together with those variables $y_j' = 1$ in y'. (An even better solution may result by reducing the currently obtained solution to a prime cover if it is not already one.) $\bar{z}$ is now updated and a backtracking step can be taken. Otherwise, if y' is not all integer and its cost is not greater than $\bar{z} - z - 1$, then an integer trial solution can be obtained from y' by using the following heuristic.

1. Let $J' = \{j \mid y_j' > 0\}$, $\hat{z} = 0$, and $\hat{J} = \phi$.

2. Let $\hat{I} = \left\{ i \,\middle|\, \sum_{j \in \hat{J}} a_{ij} \geq 1 \right\}$, i.e. $\hat{I}$ is the set of those constraints already satisfied by setting the variables in $\hat{J}$ to 1.

3. Choose j^* such that $c_{j^*} \Big/ \sum_{i \notin \hat{I}} a_{ij^*} = \min_{j \in J'} \left\{ c_j \Big/ \sum_{i \notin \hat{I}} a_{ij} \right\}$

i.e. j^* is the column of minimum-cost-per-constraint-satisfied.

4. Update $\hat{J}$ as: $\hat{J} = \hat{J} \cup \{j^*\}$ and $\hat{z}$ as: $\hat{z} = \hat{z} + c_{j^*}$

5. Repeat steps 2,3 and 4 until all constraints are satisfied: $\hat{J}$ then represents a feasible solution to P_k with cost $\hat{z}$. If $z + \hat{z} < \bar{z}$, then a better solution has been obtained which is stored and used to update $\bar{z}$. In any case, a forward branch is taken by augmenting R_k with j_o (y_{j_o} being set to 1) where $y'_{j_o} = \min (y'_j)$, (i.e. y'_{j_o} is the smallest positive fractional value of the variables in the LP solution).

This choice is made in the hope of perturbing the solution of P'_k as much as possible in order to improve the chances of an early backtracking step being taken, and is in principle the same rule for augmenting first introduced by Little et al., [36], for the travelling salesman problem.

Apart from the use of the LP to direct and limit the tree search, the algorithm proceeds in much the same way as that described in section 2.1(A), except for the fact that the order in which the rows are examined is not fixed, but depends on the last column (j_o) chosen for the forward branching. This has advantages because

it allows one to branch on nodes from which only a small number of forward branchings emanate and hence, hopefully, reduce the number of future nodes to be searched. However, an important disadvantage results from this additional flexibility in that the block structure of the tableau used in the tree search of section 2.1(A) is not maintained, with the result that the processing time per node searched is increased.

2.2 Cutting Plane Approaches [11,30]

In [30], House, Nelson and Rado describe a general method for the solution of the SCP which involves 'solving' a sequence of problems. Thus, given an SCP with 0-1 matrix A^p, one first finds any prime set-covering solution vector, $\tilde{y}^p$, of cost $\tilde{z}^p$. Then, if one can find a new (t^{th}) constraint (or cut),

$$\sum_{j=1}^{n} a_{t_j} y_j \geqslant 1$$

such that: (i) All solutions of cost less than $\tilde{z}^p$ satisfy it
and (ii) $\tilde{y}^p$ does not,

then matrix A^{p+1} is formed by adjoining the t^{th} constraint to the rows of the matrix A^p. If at any stage p^*, no such constraint can be generated, the procedure terminates with an optimal solution $\bar{y}$ of cost $\bar{z}$, where $\bar{z} = \min_{0 \leqslant k \leqslant p^*} \{\tilde{z}^k\}$

A. Particularisation due to House, Nelson and Rado [30]

The strategy proposed in [30] for finding a prime cover $\tilde{y}^p$ to a particular problem with matrix A^p involves choosing columns

according to the "least-cost-per-constraint-satisfied" criterion, until all rows are covered, and then reducing the result to a prime cover.

The method used for generating a cut satisfying conditions (i) and (ii) given above is as follows:

At the p^{th} iteration, consider a problem "dual" to the SCP given by equation (3) as:

$$\left.\begin{aligned} &\text{Maximize} \quad g^p = \sum_{i=1}^{m+p} w_i^p \\ &\text{subject to} \quad \sum_{i=1}^{m+p} a_{ij}^p w_i^p \leq c_j, \qquad j = 1, \ldots, n; \\ &\qquad\qquad w_i \geq 0 \qquad i = 1, \ldots, m+p \end{aligned}\right\} \quad (5)$$

and let u_i^p and v_j^p represent the slack variables for the primal and dual problems, respectively. Then, by a standard LP theorem, we have

$$z^p - g^p = \sum_{j=1}^{n} y_j^p v_j^p + \sum_{i=1}^{m+p} w_i^p u_i^p \qquad (6)$$

for any (feasible or not) solutions, y^p and w^p, to the primal and dual problems. Also, since $y_j^p, w_i^p, v_j^p, u_i^p \geq 0$ for feasible solutions, we have from (6) that $z^p \geq g^p$ for any pair of feasible solutions, and in particular,

$$\min_{y^p \text{ feasible}} \{z^p\} \geq \max_{w^p \text{ feasible}} \{g^p\} \quad . \tag{7}$$

For any solution w^p of cost g^p, we see from (6) that any feasible solution y^p of cost $z^p < g^p$ must be such that at least one y^p_j is positive for a corresponding negative v^p_j, i.e. one must have:

$$\sum_{j \in V^-} y^p_j \geq 1, \text{ where } V^- = \{j \mid v^p_j < 0\} \tag{8}$$

Now let a (feasible or infeasible) solution w^p be found with cost $g^p = \tilde{z}^p$. This w^p has a set V^- associated with it, which must be such that equation (3) is not satisfied by $\tilde{y}^p$, as required by condition (ii),

i.e. one must have $$\sum_{j \in V^-} \tilde{y}^p_j = 0. \tag{9}$$

Thus, the cutting-plane (t^{th}-constraint) during the algorithm at the p^{th} iteration is obtained by finding a 'solution' (w^p, v^p) to problem (5) which satisfies

$$\begin{aligned} w^p_i &\geq 0 \qquad i = 1, \ldots m+p \\ g^p &= \tilde{z}^p \end{aligned} \tag{10}$$

and eqn (9) i.e. $V^- \cap \tilde{Y}^p = \phi$ where $\tilde{Y}^p = \{j \mid \tilde{y}^p_j = 1\}$.

Having found V^-, the t^{th} constraint can be obtained (and appended to

the matrix A^p to form the $(p+1)^{st}$ problem) by

setting $a_{t_j} = 1$ if $j \in V^-$ according to the cut defined by (8)

$= 0$ otherwise

Obviously, if $V^- = \phi$, then all entries in the t^{th} constraint would be zero and condition (8) could not be satisfied which, according to the argument just preceding (8), implies that the optimal solution to problem p is greater than or equal to $\tilde{z}^p$, and therefore one can terminate with $\bar{z} = \min_{0 \leq k \leq p} \{\tilde{z}^k\}$ as the value of the optimal solution to the original SCP.

It should be noted that the cut defined by (8) is stronger the smaller the value of $|V^-|$, and hence it would be desirable to find a solution (w^p, v^p) satisfying (10) which leads to a small $|V^-|$. In [30], a heuristic method for obtaining such a solution is given.

An additional step which may lead to an early termination of the procedure is described in [30] where, at each step p, (before generating the cutting plane as described above), a <u>feasible</u> solution to problem (5) is found, in the hope of using condition (7) to terminate the algorithm. Thus, a feasible solution (w^p, v^p) to problem (5) of cost $\tilde{g}^p = \sum_{i=1}^{m+p} w_i^p$ is obtained heuristically by first setting $w_i^p = \min_{a_{ij}^p = 1} \{c_j\}$, $(i = 1, \ldots m+p)$, and then decreasing these w_i^p, one at a time in unit steps but maintaining non-negativity. Then, if at some stage p^*, $\max_{0 \leq k \leq p^*} \{\tilde{g}^k\} \geq \min_{0 \leq k \leq p^*} \{\tilde{z}^k\}$, one may terminate with $\bar{z}$

as the cost of the optimal cover to the initial SCP.

B. Particularisation due to Bellmore and Ratliff [11]

A method conceptually the same as the previous one is given by Bellmore and Ratliff, who describe in [11] a method for generating cutting constraints which is often computationally superior to that of section 2.2 (A) above.

Once more with $\tilde{Y}^p = \left\{ j \,\middle|\, \tilde{y}^p_j = 1 \right\}$, where $\tilde{y}^p$ is a prime feasible solution to the p^{th} SCP with matrix A^p, it can be seen that there exists, for each $j \in \tilde{Y}^p$ (according to the definition of a prime solution), at least one row i such that $a^p_{ij} = 1$ and $a^p_{ik} = 0 \;\forall k \in \tilde{Y}^p, \; k \neq j$. Then let D be a minimal set of such rows i, with only one row - if there is more than one - chosen for each column $j \in \tilde{Y}^p$. (A heuristic rule for making the choice between possible alternative rows for inclusion into D is given in [11] .)

Consider now a matrix E^p composed of the $(m+p) \times 1$ unit column vectors $-e_i$, $i \notin D$, where e_i contains a 1 in the i^{th} position and 0's everywhere else; and a matrix A^p_o composed of the column vectors β^p_j of the matrix A^p, where $j \in \tilde{Y}^p$. It can then easily be shown [11] that the matrix $B = (A^p_o, E^p)$ is a basis for the continuous LP problem corresponding to the SCP with matrix A^p. Also, considering this continuous LP problem and letting $\bar{J} = \left\{ j \,\middle|\, C_b B^{-1} \beta^p_j > c_j \right\}$, where C_b is the cost vector corresponding to basic variable, then any feasible solution of cost lower than that of the current LP solution must

introduce a variable y'_j where $j \in \bar{J}$. Thus, any 0-1 valued solution to the continuous problem with cost less than the current best must satisfy:

$$\sum_{j \in \bar{J}} y_j \geq 1 \qquad (11)$$

which therefore forms the required cutting plane constraint.

The determination of this constraint is simplified by the fact that, - as shown in Reference [11] - no matrix inversion is necessary to obtain $\bar{J}$. As this constraint satisfies the two cutting-plane conditions (i) and (ii) mentioned earlier, the basic algorithm can proceed as outlined there.

The method used in [11] for finding a solution (prime cover) $\tilde{y}^p$ to the problem with matrix A^p is not to choose columns according to the "least-cost-per-constraint-satisfied" criterion as was done in [30], but instead to make use of y'^p, the optimal LP solution to the continuous problem by setting

$$\left.\begin{aligned} \tilde{y}^p_j &= 1 \text{ if } 0 < y'^p_j \leq 1 \\ &= 0 \text{ otherwise} \end{aligned}\right\} \quad \text{and reducing the}$$

resultant $\tilde{y}^p$ to a prime cover.

Furthermore, if $\tilde{z}^p$ is the cost of $\tilde{y}^p$, and z'^p is the cost of the optimal solution to the continuous problem, then if a stage p^* is ever reached where $\min_{0 \leq k < p^*} \{z^k\} \leq \lfloor z'^{p^*} \rfloor$ where $\lfloor x \rfloor$ represents the smallest integer not less than x, then the procedure may be terminated, with $\min_{0 \leq k < p^*} \{\tilde{z}^k\}$ as the cost of the optimal solution to the initial SCP.

2.3 Lawler's "Duality" Method [34]

In [34], Lawler describes a method for solving the unicost SCP and generalizes it for the SCP. Basically, the idea is that given a unicost SCP whose optimal cover is of cost $\bar{z}$ (as yet undetermined), one generates from this a further unicost SCP whose optimal cost is $\bar{z} - 1$. Thus, by starting the process with the given SCP and terminating when a SCP is reached whose optimal cover is of cost 0 (i.e. the empty SCP), it is seen that $\bar{z}$, the optimal cost of the given unicost SCP, is equal to the number of intermediate problems generated, while the optimal cover itself can be established by means of a back-tracking method, as explained below.

Given the (0,1) matrix A of a unicost SCP (see equation (3)), let α_i, $i = 1,..m$ represent a row vector of A, and β_j, $j = 1,...n$ a column vector. If A^p represents the matrix of the p^{th} intermediate SCP, it can be shown, [34], that A^{p+1} can be obtained from A^p, where the optimal cost of A^p is 1 greater that that of A^{p+1}, by using the following steps:

(1) Choose any row α_ℓ from A^p. (Note: As the order of rows and columns in A^p is immaterial to the method, we may assume that α_ℓ is the first row, and that all its 1's are in the first r_1 columns.)

(2) For each $k = 1,2,...r_1$, let A^p_k represent the submatrix consisting of those rows i of A^p with $a^p_{ik} = 0$. (Each A^p_k thus contains n columns but has fewer rows than A^p.)

(3) Let an "or" operation, v, be defined on rows as follows: for any two rows α_s and α_t, $\alpha_s v \alpha_t = (b_1, b_2, ...b_n)$ where $b_j = \max(\alpha_{s_j}, \alpha_{t_j})$ $j = 1,...n$; and call $\alpha_1 v \alpha_2 v ... v \alpha_r$

an r-fold or-operation.

Then let $\bar{A}^{p+1}$ be the matrix whose rows are obtained by performing r_1-fold or-operations on r_1 rows, each being taken from a different A_k^p in all possible ways.

(4) Apply reduction (3) of section 1.2 to $\bar{A}^{p+1}$ to obtain A^{p+1}.

The SCP algorithm of [34] is then as follows:

(i) Set $A^o = A$, and set $p = 0$.

(ii) Choose a row α_ℓ^p of A^p and thus generate A^{p+1} as explained above. (Lawler suggests using a row with the fewest 1's so as to minimize the number of or-operations.)

(iii) If A^{p+1} has no rows, go to (iv); otherwise set $p = p+1$ and go to (ii).

(iv) Terminate the iterations. The cost of the optimal cover of the original unicost SCP is p.

The optimal cover for $A^{p+1} = (0,\ldots 0)$, and it is shown in [34] that this is also the optimal cover of A_k^p for some k. Thus, an optimal cover for A^p is $y^p = (0,0,\ldots,1,0,0,\ldots 0)$ where the entry of 1 is in the k^{th} position. y^p, in turn, is an optimal cover for some A_q^{p-1} so that $y^{p-1} = (0,0,\ldots,1,0,\ldots,1,0\ldots 0)$ with entries of 1 in the k^{th} and q^{th} positions. Continuing in this manner one eventually reaches y^o which is the solution to the initial unicost SCP.

The basic algorithm can be computationally greatly improved, and in the appendix to this chapter, we indicate some such improvements based mainly on the early detection and implicit elimination of

dominated r-fold combinations. These modifications were in fact incorporated into our computer program for this method (as described in Section 4) and resulted in improvements in computation time by orders of magnitude.

3. SOME MODIFICATIONS AND IMPROVEMENTS

In this section, we discuss in greater detail some aspects of the first of the methods described in the previous section, for which no detailed computational results, or study of its algorithmic aspects has, to our knowledge, as yet been reported.

3.1 Dynamic Application of Reduction Tests

In section 2.1.A(ii) it was noted that, in setting up the block structure, advantage was taken of the fact that when block k would be consulted during the course of the algorithm, elements $x_1, x_2, \ldots x_{k-1}$, would already have been covered by the partial solution. This led to a compression of block k by applying reduction (4) of section 1.2.

However, at any stage during the algorithm, when block k is about to be considered, some elements x_i, $i \geq k+1$, will also be covered by the current partial solution, and, therefore, an approach similar to the above can be used to compress the remaining subproblem even further during the search. Obviously this further compression has to be reversible as it is dependent on the current state of the search. (Note: this is similar to the 'local dominance' tests used in [35].)

3.2 Further Dominance Tests

We will first illustrate these dominance tests by an example before stating the general results. Consider a case where block 1 contains (among others), sets $S_1 = \{x_1, x_4, x_6\}$ and $S_2 = \{x_1, x_3, x_4\}$, of costs 3 and 4 respectively, and where block 2 contains (among others), sets $S_3 = \{x_2, x_3, x_5\}$ and $S_4 = \{x_2, x_4, x_5\}$, each of cost 2.

During the course of the algorithm of section 2.1.A, a stage will be reached where $R_a = \{S_1, S_3\}$, $T_a = \{x_1, x_2, \ldots x_6\}$, $z_a = 5$; and we will branch forward from this stage until either we find a better solution that the current $\bar{R}$, or we realize that S_1 and S_3 cannot both appear in an optimal solution.

Many steps later, we will reach a stage where

$$R_b = \{S_2, S_3\}, \quad T_b = \{x_1, \ldots, x_5\}, \quad z_b = 6.$$

At this point it becomes apparent that any further branching from this stage is unnecessary, since $T_b \subset T_a$ and $z_b > z_a$. A similar situation occurs when the stage is reached where

$$R_c = \{S_2, S_4\}, \quad T_c = \{x_1, \ldots, x_5\}, \quad z_c = 6.$$

Thus, it may be worthwhile to keep, for each value of $z = 1, 2, \ldots \bar{z}$ some (perhaps incomplete) list of the maximal T's which have already been achieved for this z, (where by maximal is meant a set not included in another set also in the list). These lists of T's can then be used to limit the search by eliminating branches that later on prove fruitless. It is generally impractical, however, to save <u>all</u> the maximal sets T, at any cost level. In fact if this is done, the method would be similar to a Dynamic Programming appraoch to the problem as a whole (or, alternatively, could be considered as a full breadth-first tree search).

Let us now assume that we have saved some list $L(z_i)$ of previous sets T that have been attained during the course of the algorithm at a summed-cost level z_i. Suppose a stage is reached where $T = T'$, $R = R'$, $z = z'$, and we are about to examine block k (i.e. $k = \min \{i \mid x_i \notin T'\}$), and to consider sets S_j^k of cost c_j^k for the next branching. If $z' + c_j^k < \bar{z}$, the original algorithm (of section 2.1.A) would have branched forward from this stage, and updated $T = T' \cup S_j^k$, $R = R \cup \{S_j^k\}$ and put $z' = z' + c_j^k$, regardless of any other considerations.

However, one could also now ensure (at step 3), that:

$$T' \cup S_j^k \notin T_\ell, \ \forall T_\ell \in L(z_i), \text{ and for all } z_i, \ z' < z_i \leq z' + c_j^k \qquad (12)$$

before branching forward. If the set S_j^k fails the above test, then it is rejected and instead the next set, S_{j+1}^k, of block k is considered, etc. If S_j^k passes test (12), one could branch forward on S_j^k, and continue as previously, after first updating $L(z)$ by introducing the set $T' \cup S_j^k$ into $L(z' + c_j^k)$.

Since, as mentioned earlier, it is practically impossible to store the complete lists $L(z_i)$, some heuristic criterion has to be used to decide on the size of these lists, and how they should be updated during the search.

(i) List size

It is obviously better to eliminate sets S_j^k which are potential

tree branchings at an early level in the tree, as this would discard larger portions of the potential search-tree. It would thus seem intuitively worthwhile to keep larger-sized lists $L(z_i)$ for the smaller z_i. This argument is further reinforced by the fact that test (12) is more likely to be effective when the sets T contain only a few elements, which is generally the case for small z_i-levels.

(ii) Updating of the lists

At a particular z_i-level, the current T-set (T' say) is more likely to be a subset of a set T'' (in the list $L(z_i)$), which was derived from the same general part of the search-tree. This is so because T' and T'' have in common a large section of the path from the start-node to their respective nodes. Thus, it would seem to be advantageous to arrange the lists $L(z_i)$ as pushdown stacks and to use a FIFO (first in, first out) policy, with the bottom-most set being discarded in the even of stack overflow.

3.3 A New Lower-Bound

In section 2.1(A) a lower bound, [45], used to limit the search was described. This bound is easy to compute but is quite loose. Since, as is well known, the tightness of a bound is one of the most effective means of limiting the size of a tree-search, we give below another bound, which is computationally more demanding, but produces, a much more efficient bounding test.

At some stage of the search, given by R', T', z', and where block k is the next block to be considered, this bound, h, can be calculated as follows:

Consider an uncovered element $x_i \in X - T'$ which does not appear in any of the sets of those blocks $k, k+1, \ldots, i-1$, corresponding to elements not yet covered by the partial solution. The element x_i cannot then be covered unless some set S_j^i of block i is chosen to add to R' at a future stage. Thus, for each such element X_i, construct a row for a matrix $M = [m_{rs}]$ and a row for a second matrix $M' = [m'_{rs}]$, where m_{rs} is the number of elements in set S_s^r and m'_{rs} is its cost.

In addition, append an extra row, θ say, to M and M' with $m_{\theta s} = s$ for all $s = 0,1,\ldots, m-|T'|$, and $m'_{\theta s} = s.\min\left\{c_j^i / |S_j^i|\right\}$ with the minimum being taken over all sets S_j^i with $x_i \notin T'$.

Note that although the number of elements in a row r_1 of M (or M') may be different from that of another row r_2, we assume here that 0's and ∞'s are entered at the end of some rows of M and M' respectively, so that all rows have ℓ (say) elements and the matrices become rectangular.

The following observations can now be made: since the optimal solution to the current subproblem must cover $m-|T'|$ elements, a choice of one entry from each row i of M, which satisfies $(m_{1s_1} + m_{2s_2} + \ldots + m_{\theta s_\theta}) \geq m-|T'|$ and which minimises the corresponding cost, $v = (m'_{1s_1} + \ldots + m'_{\theta s_\theta})$ gives v as a lower bound to the optimal cost of the (covering) subproblem. Note that the assumption is made that the sets corresponding to the

allocations in the rows of M are disjoint, which is obviously the best possible situation. The last row θ simply ensure that, if $\sum_{i=1}^{\theta-1} m_{is_i} < m-|T'|$, the remaining elements are covered in the best possible way, i.e. at minimal cost-per-additional-element-covered.

The minimal value of $\sum_{i=1}^{\theta} m'_{is_i}$ subject to $\sum_{i=1}^{\theta} m_{is_i} \geq m- T'$ can easily be derived by a dynamic programming algorithm as follows:

Let $g_\rho(v)$ be the maximum number of elements that can be covered using only the first ρ rows of M (i.e. only ρ of the blocks in the subproblem), and whose total cost does not exceed v. $g_\rho(v)$ can then be calculated iteratively as:

$$g_\rho(v) = \max_{s=1,\ldots\ell} \left\{ m_{\rho s} + g_{\rho-1}(v-m'_{\rho s}) \right\},$$

where $g_0(v)$ is initialized to 0 for all v.

Hence, the lowest value, v^*, of v, for which $g_\theta(v) \geq m-|T'|$ is then the required lower bound, h, and can be easily obtained from the dynamic programming tableau derived from the above iterative equation. One should note that only that range of v where $0 \leq v < \bar{z} - z'$ need be considered, since if $h \geq \bar{z} - z'$ (i.e. $g_\theta(v) < m-|T'|$ for $v \geq \bar{z} - z'$), a backtracking step may be taken.

The improvement of this bound over that mentioned in section 2.1(A) is due to the fact that the first $\theta-1$ rows of M correspond

to elements which can only be covered in a small number of ways, and thus individually contribute more towards a tighter lower bound.

4. COMPUTATIONAL RESULTS

Most of the methods described in the previous sections were coded in Fortran for the CDC 6400 computers, and this section presents computational comparisons of these programs on a large number of unicost set covering test problems.

The first group of 25 test problems (Tables 1 and 2) was obtained by generating random graphs, and using the set-covering methods to solve the graph-colouring problem associated with each graph. The elements (rows) for the SCP are the vertices of the graph, and the sets (columns) are the maximal independent sets of the graph, which were generated by a separate program.*

The second group of 8 test problems (Table 3) was sent to the authors by H.M. Salkin, and is a subgroup of some of the problems previously used in the literature [11,26,35,54] . The distinctive feature of these problems is that, for the most part, any column contains exactly two 1's, and no column reductions are applicable.

The third group of 275 test problems were again obtained by generating random graphs and solving the associated GCP by set-covering. Ten different graphs were generated for each of the five densities relating to any number of vertices, and the computational results averaged, in order to eliminate, as much as possible, any variations due to randomness.

---

* By definition of maximal independent sets, no column reductions (type (4) in section 1.2) are applicable to the initial SCP .

In fact, this group of problems is the main group on which the direct graph-colouring algorithms of Chapter 5 and 6 were applied, and so they serve the additional purpose of leading to a comparison between the methods of solving the GCP directly, and the methods (of this chapter) of solving the GCP by means of the SCP.

The times given in the following tables are in seconds for the CDC 6400 computer, and all programs used Boolean operations for set manipulation. Some features of the algorithms tested are apparent in these tables and will now be discussed.

A. The simple tree search

This method (A in the tables) was programmed as described in [45] but with the additional facility of a simple one-step look ahead. Also the elements were re-ordered in the manner mentioned in section 2.1.A.

As can be seen from Tables 1 and 2, this algorithm is uniformly inferior to algorithm E in all respects, and therefore this program was not run on the other groups of SCP's.

B. House, Nelson and Rado's cutting plane method

It can be seen from the above tables (Table 1 in particular), that this method behaves quite erratically. This is perhaps not very surprising when it is recalled that the method although exact, is based on a number of heuristics for generating cutting planes and resolving subproblems, and will thus inherit some of the peculiarities of these heuristics.

However, what is surprising - and totally unexpected - is the fact that this method has on a number of occasions solved very large problems e.g. problems 13, 15 in Table 1 (and in fact other

set covering problems with 500 columns), recording very small computing times. The computing times for the above mentioned two problems and also for problems 10 and 11 are actually the best times recorded for the five methods tested. On the other hand, the method failed on comparatively small problems e.g. No.7 and has found the problems of the second group very difficult (see Table 3). It is perhaps interesting to note here that the authors are aware of at least one major airline using this algorithm for "solving" aircrew scheduling problems of very large size.

C. Bellmore and Ratliff's cutting plane method

This method was programmed as described in [11] and section 2.2.B. A standard LP subroutine was used with the algorithm and the LP's were solved all in core. The blank spaces in Tables 1 and 2 corresponding to this algorithm are due to the fact that the standard LP package used could not deal with these larger problems.

As for the previous method, this algorithm is also dependent, though not to as great a degree, on a heuristic (for choosing a prime cover of any iteration), and so again some variation of performance is to be expected. It is interesting to note that very often, especially for the relatively small problems of Table 1, the round-up of the value of the LP solution did in fact correspond to the value of the optimal solution and much of the time spent is due to this heuristic.

D. Lawler's duality method

This method was first programmed in the obvious manner suggested by section 2.3, Lawler [34] not having given any detailed algorithm. However, it soon became clear from both time and storage considerations that

this program was grossly inefficient, the main trouble occurring, at step (3) of the algorithm (see section 2.4), with far too many r-fold combinations being generated and placed into $\bar{A}^P$ which were later found to be dominated. This is, of course, a well known problem when using similar Boolean approaches, i.e., an efficient 'absorption technique' is a major requirement.

With the above considerations in mind, some attempts were made to improve the algorithm, with varying degrees of success, but as shown in Table 1, the results are still not very satisfactory. However, as mentioned earlier, improvements in the results by order of magnitude were found possible by using various techniques for implicitly eliminating r-fold combinations from $\bar{A}^P$; and these are indicated in the appendix to this chapter.

E. The tree search method of Section 3

This method was programmed as described in Section 3 and the list sizes were (arbitrarily) set at 60 for the first five levels of the search-tree and 30 thereafter. As can be seen from the tables this method was the most consistently effective of the methods coded, and compares favourably with the unicost results of the other published algorithms.

It has been noted from a more detailed analysis of the results obtained by this method that the bound proved to be quite effective, cutting off more potential nodes of the tree search than did the lists when either bound or lists were acting in isolation. However, the lists tend to cut off nodes at the top part of the tree and the bound at a lower part so that their relative importance may be different from that implied above. When both the bound and lists are active together the combination produces a very effective means of limiting the tree search as can be seen from the results shown in the Tables.

Table 1 COMPUTATIONAL TIMES (CDC 6400 sec) RECORDED FOR VARIOUS METHODS

Problem	Number of rows	Number of columns	Density	Methods				
				A	B	C	D	E
1	15	20	.22	.05	.26	.30	.12	.06
2	15	25	.28	.04	.32	.45	.10	.04
3	15	25	.30	.08	.30	.35	.15	.08
4	20	60	.18	.70	2.75	2.55	.35	.35
5	20	50	.20	.15	.30	2.00	.18	.17
6	20	75	.23	.60	10.50	2.90	.26	.55
7	25	110	.17	3.50	300.+	31.10	300.+	.80
8	25	120	.19	6.75	65.00	21.15	30.00	1.80
9	25	170	.22	10.50	121.75	27.25	300.+	2.00
10	30	80	.12	18.50	2.00	195.+	30.00	1.30
11	30	180	.15	300.+	.80	20.20	300.+	14.50
12	30	250	.17	78.50	17.75	60.00	300.+	9.00
13	30	340	.21	280.00	4.50	52.§	300.+	13.20
14	30	475	.20	-	300.+	-	-	17.00
15	30	500	.23	-	27.25	-	-	32.00
16	30	700	.23	-	300.+	-	-	155.50
17	30	725	.25	-	-	-	-	47.50
18	30	875	.26	-	-	-	-	77.20
19	30	1000	.27	-	-	-	-	92.80
20	35	160	.09	-	-	-	-	28.50
21	35	290	.13	-	-	-	-	40.00
22	35	460	.16	-	-	-	-	105.50
23	35	585	.18	-	-	-	-	189.50
24	35	780	.19	-	-	-	-	201.20
25	35	1075	.20	-	-	-	-	170.00

A. PIERCE/GARFINKEL-NEMHAUSER
B. HOUSE-NELSON-RADO
C. BELLMORE-RATLIFF
D. LAWLER
E. METHOD OF SECTION 3

§ Problem ran out of core storage after generating 30 cuts.

Table 2 NUMBER OF CUTS/NODES/ITERATIONS FOR VARIOUS METHODS

Problem	Cuts/Nodes (in 1000's) / Iterations				
	A*	B+	C+	D++	E*
1	.055	2	0	6	.020
2	.030	4	0	5	.010
3	.035	3	0	4	.020
4	2.000	3	0	6	.070
5	.100	2	0	5	.020
6	.750	24	0	5	.050
7	7.000	120+	2	4+	.060
8	13.200	51	29	6	.160
9	16.900	71	4	4+	.110
10	40.700	1	70+	9	.200
11	-	0	8	4+	1.600
12	114.000	12	4	3+	.520
13	546.700	2	30§	3+	.300
14	-	70+	-	-	.090
15	-	8	-	-	.400
16	-	70+	-	-	3.020
17	-	-	-	-	.120
18	-	-	-	-	.180
19	-	-	-	-	.190
20	-	-	-	-	2.180
21	-	-	-	-	2.030
22	-	-	-	-	2.950
23	-	-	-	-	3.960
24	-	-	-	-	5.030
25	-	-	-	-	.550

Footnotes:

* Number of nodes (in thousands) in Search Tree

\+ Number of Cuts needed.

++ Number of Subproblem Generated (see section 2.3)

§ Problem ran out of core storage

Table 3 COMPUTATIONAL PERFORMANCE RECORDED FOR VARIOUS METHODS

Problem	Number of rows	Number of columns	Density	METHODS Computational Times (sec.)				METHODS Number of cuts/nodes (1000's)/iterations						
				B	C	D	E	B	C	D	E	α	β	γ
26	30	60	.07	300.+	4.2	300.+	0.7	130+	0	11+	.160	.88	18.0	10.5
27	30	60	.07	40.00	36.5	1.4	2.4	50	38	14	.650	1.22		
28	30	70	.07	300.+	3.9	100.7	2.4	130+	0	14	.620	1.96	20.4	9.3
29	30	70	.07	105.+	4.2	300.+	1.9	70+	0	11+	.430	.96		
30	30	80	.07	105.+	6.5	300.+	7.2	70+	0	11+	1.830	3.12	31.2	11.7
31	30	80	.07	105.+	7.1	200.+	0.5	70+	0	10+	.100	2.57		
32	30	90	.07	105.+	8.7	200.+	9.7	70+	0	8+	1.600	3.19	30.6	**
33	30	90	.07	105.+	200.+	200.+	10.1	70+	70+	8+	2.590	2.30		

1. α Results reported by Salkin and Koncal in [55] . Times are in Univac 1108 seconds.
2. β Results reported by Lemke, Salkin and Spielberg in [35] . Times are in IBM 360/50 seconds and represent the average over the two problems.
3. γ Results reported by Bellman and Ratliff in [11] . Times are in IBM 7094 seconds and represent an average over (about) 10 problems.
4. ** There is a misprint in Talbe 1 of [11] for these problems.

TABLE 4 SOLVING THE GCP

VERTICES	DENSITY *	TIMES CDC (6400 seconds)**			
		B	C	D	E
15	.1	.3	.7	.2	.3
	.3	.7	1.2	.3	1.4
	.5	1.4	1.0	.3	.2
	.7	.5	.6	.3	.1
	.9	.1		.3	.1
20	.1	1.2	7.2	.4	3.6
	.3	3.9	14.2	.8	2.4
	.5	8.6	5.7	1.3	.7
	.7	2.4	5.2	.7	.4
	.9	.4		.3	.2
25	.1	14.5	82.5		77.4
	.3	123.6+	33.5	41.7	17.9
	.5		25.2	148.6	7.0
	.7	88.5+	6.7	3.4	1.7
	.9	1.0		.4	.4
30	.1		154.1		
	.3		89.9		134.3
	.5		33.8		46.7
	.7		12.4		17.4
	.9	18.5+		.5	.5
35	.5		49.3		
	.7		20.3		80.4
	.9	9.4		.8	1.0
40	.7		27.5		
	.9			5.5	4.8
45	.7		44.3		
	.9			12.5	25.9

* Density refers here to the number of arcs of the graph relative to the potential number of arcs of a complete graph.

** Results are for an average over ten graphs, blanks in the table being due to time or storage excess.

\+ Includes a problem which did not terminate after 75 iterations

5. REFERENCES

1. AGARD, J. "Monthly Assignment of Stewards", Air France, AGIFORS Sympos., Killarney, 1966

2. AGARD, J., ARABEYRE, J.P. and VAUTIER, J. "Generation automatique de rotations d'equipages". RIRO 1-6 (1967) pp 107-117.

3. ANDREW, G., HOFFMAN, T. and KRABEK, C. "On the Generalized Set Covering Porblem", Joint National Meeting ORSA-TIMS, San Francisco, California (1968).

4. ARABEYRE, J.P. "Method of Crew Scheduling", AGIFORS, 1966, Air France.

5. ARABEYRE, J.P., FEARNLEY, J., STEIGER, F.C. and TEATHER, W. "The Airline Crew Scheduling Problem, A Survey", Transportation Science 3 (1969) pp 140-163.

6. BALAS, E. and PADBERG, M.W. "On the Set Covering Problem", Operations Research 20 (1972) pp 1152-1161.

7. BALAS, E. and PADBERG, M.W. "On the Set Covering Problem. II. An Algorithm", Operations Research 25 (1975).

8. BALINSKI, M. "Integer Programming: Methods, Uses, Computation", Management Science 12 (1965) pp 253-313.

9. BALINSKI, M.L. and QUANDT, R.E. "On an Integer Program for a Delivery Problem", Operations Research 12 (1964) pp 300-304.

10. BELLMORE, M., GREENBERG, H. and JARVIS, J. "Multi-Commodity Disconnecting Sets", Management Science 16 (1970) pp 427-433.

11. BELLMORE, M. and RATLIFF, H.D. "Set Covering and Involutary Bases", Management Science 18 (1971) pp 194-206.

12. BELLMORE, M. and RATLIFF, H.D. "Optimal Defense of Multi-Commodity Networks", Management Science 18 (1971) pp 174-185.

13. BERGMAN, L.G. and WEISS, D.E. "A Transformation Technique for Obtaining Least Cost Solutions to Covering Problems", SIAM J. Appl. Maths 23 (1972) pp 463-476.

14. COBHAM, A., FRIDSHAL, F. and NORTH, J.H. "An Application of Linear Programming to the Minimization of Boolean Functions", IBM Res. Rept. RC-472, IBM Research Center, 1961.

15. COBHAM, A., FRIDSHAL, F. and NORTH, J.H. "A Statistical Study of the Minimization of Boolean Functions using Integer Programming", IBM Res. Rept. RC-756, IBM Reseach Center, 1962.

16. COBHAM, A. and NORTH, J. "Extensions of the Integer Programming Approach to the Minimization of Boolean Functions", IBM Res. Rept. RC-915, IBM Research Center, 1963.

17. CROWSTON, W.B. "Decision Network Planning Models", Ph.D. Dissertation, Carnegie-Mellon University, 1968.

18. DANZIG, C.B. and RAMSER, J.H. "The Truck Dispatching Problem", Managment Science 6, pp 80-91.

19. DAY, R.H. "On Optimal Extracting from a Multiple File Data Storage System: An Application of Integer Progamming", Operations Research 13 (1965) pp 482-494.

20. EVERS, G.H.E. "Relevant Factors Aroud Crew-Utilization", AGIFORS Symp., 1965, KLM.

21. FREEMAN, D.R. and JUCKER, J.V. "The Line Balancing Problem", J. Indust. Engr. 18 (1967) pp 361-364.

22. GARFINKEL, R.S. "Optimal Political Districting", PH.D. Thesis, The John Hopkins University, 1968.

23. GARFINKEL, R.S. "Set Covering: A Survey", presented at XVII TIMS Conference, London, 1970.

24. GARFINKEL, R.S. and NEMHAUSER, G.L. "The Set Partitioning Problem: Set Covering with Equality Constraints", Operations Research 17 (1969) pp 848-856.

25. GARFINKEL, R.S. and NEMHAUSER, G.L. Integer Programming. Wiley & Sons, 1972.

26. GEOFFRION, A.M. "An Improved Implicit Enumeration Approach for Integer Programming", Operations Research 17 (1969) pp 437-454.

27. GIMPEL, J.F. "A Reduction Technique for Prime Implicant Tables", IEEE Trans. Electronic Computers EC-14 (1965) pp 535-541.

28. GLOVER, F. "A Note on Extreme Point Solutions and a Paper by Lemke, Salkin and Spielberg", Operations Research 19 (1971), pp 1023-1025.

29. GOMORY, R. "An All-integer Integer Programming Algorithm", in Industrial Scheduling (MUTH and THOMPSON, eds) Prentice-Hall,Englewood Cliffs, N.J. 1963, pp 193-206.

30. HOUSE, R., NELSON, L. and RADO, T. "Computer Studies of a Certain Class of Linear Integer Programs", in Recent Advances in Optimization Techniques, (LAVI and VOGEL eds.) Wiley, N.Y., 1965.

31. JARVIS, J. "Optimal Attack and Defense of a Command and Control Communications Network", Ph.D. Thesis, The John Hopkins University, Baltimore, Ma., 1968.

32. JENSEN, P.A. "Optimal Network Partitioning", Operations Research 19 (1971) pp 916-932.

33. KOLNER, T.K. "Some Highlights of a Scheduling Matrix Generator System", AGIFORS, 1966, Air France.

34. LAWLER, E.L. "Covering Problems: Duality Relations and a New Method of Solution", SIAM J. Appl. Maths. 14 (1966) pp 1115-1132.

35. LEMKE, C.E., SALKIN, H.M. and SPIELBERG, K., "Set Covering by Single Branch Enumeration with Linear Programming Subproblems", Operations Research 19 (1971) pp 998-1022.

36. LITTLE, J.D.C., MURTY, K.G., SWEENEY, D.W. and KAREL, C. "An Algorithm for the Travelling Salesman Problem", Operations Research 11 (1963) pp 972-989.

37. McCLUSKEY, E.J. Jr. "Minimization of Boolean Functions", Bell System Tech. Journal 35 (1956) pp 1417-1444.

38. MARSTEN. R.E. "An Algorithm for Large Set Partitioning Problems", Management Science 20 (1974) pp 774-787.

39. MAYOH, B.H. "Simplification of Logical Expressions", SIAM J. Appl. Maths 15 (1967) pp 898-905.

40. MAYOH, B.H. "On Finding Optimal Covers", International Journal of Computer Mathematics 2 (1968) pp 57-74.

41. MICHAUD, P. "Exact Implicit Enumeration Method for Solving the Set Partitioning Problem", IBM Jnl. Res. & Dev. 16 (1972) pp 573-578.

42. NEMHAUSER, G.L., TROTTER, L.E., and NAUSS, R.M. "Set Partitioning and Chain Decomposition", Management Science 20 (1974) pp 1413-1423.

43. PETRICK, S.R. "A Direct Determination of the Redundant Forms of a Boolean Function from the Set of Prime Implicants", AFCRC-TR-56-110, Air Force Cambridge Research Centre, 1956.

44. PIERCE, J.F. "On the Truck Dispatching Problem - Part I", IBM Scientific Centre Technical Report 320-2018, 1967.

45. PIERCE, J.F. "An Application of Combinatorial Programming to a Class of All-Zero-One Integer Programming Problems", Management Science 15 (1968) pp 191-209.

46. PIERCE, J.F., and LASYKY, J.S. "Improved Combinatorial Programming Algorithms for a Class of All-Zero-One Integer Programming Problems", Management Science 19 (1973) pp 528-543.

47. PYNE, J.B. and McCLUSKEY, E.J. Jr. "An Essay on Prime Implicant Tables", SIAM J. Appl. Maths. 9 (1961) pp 604-631.

48. QUINE, W.V. "The Problem of Simplifying Truth Functions", Amer. Math. Monthly 59 (1952) pp 521-531.

49. RAY-CHAUDHURI, D.K. "An Algorithm for a Maximum Cover of an Abstract Complex", Canadian Jnl. of Maths. 15 (1963) pp 11-24.

50. ROOT, J.G. "An Application of Symbolic Logic to a Selection Problem", Operations Research 12 (1964) pp 519-526.

51. ROTH, R. "Computer Solutions to Minimum-Cover Problems", Operations Research 17 (1969) pp 455-465.

52. ROY, B. "An Algorithm for a General Constrained Set Covering Problem", in Computing and Graph Theory, Academic Press, N.Y. 1972.

53. SALKIN, H.M. and KONCAL, R. "A Pseudo Dual All-Integer Algorithm for the Set Covering Problem", Dept. of O.R. Technical Memo. No. 204, Case Western Reserve University, 1970.

54. SALKIN, H.M. and KONCAL, R. "A Dual Algorithm for the Set Covering Problem", Dept. of O.R. Technical Memo. No. 250, Case Western Reserve University, 1971.

55. SALKIN, H.M. and KONCAL, R. "Set Covering by an All Integer Algorithm: Computational Experience", Journal of the Association for Computing Machinery 20 (1973) pp 189-193.

56. SALVESON, M.E. "The Assembly Line Balancing Problem", Journal of Industrial Engineering 6 (1955) pp 18-25.

57. STEIGER, F. "Optimization of Swiss Air's Crew Scheduling by an Integer Linear Programming Model", Swissair O.R. SDK 3.3.911, 1965.

58. STEIGER, F. and NEIDERER, M. "Scheduling Air Crews by Integer Programming" presented at IFIP Congress 1968, Edinburgh.

59. THIRIEZ, H.M. "Implicit Enumeration Applied to the Crew Scheduling Algorithm", Dept. of Aeronautics, M.I.T., 1968.

60. THIRIEZ, H.M. "Airline Crew Scheduling : A Group Theoretic Approach", Ph.D. Thesis, M.I.T. FTL-R69-1, 1969.

61. THIRIEZ, H.M. "The Set Covering Problem: A Group Theoretic Approach", R.I.R.O. V-3 (1971) pp 83-104.

62. WELLS, M.B. "Application of Finite Set Covering Theorem to the Simplification of Boolean Function Expressions", in Information Processing 62, (C. POPPLEWELL, ed.), North Holland, Amsterdam, 1962.

APPENDIX

Computational Considerations for Lawler's 'Duality' Method for the SCP

In section 2.3 of the preceding chapter we described Lawler's duality method for solving the SCP and we indicated that it was essentially inadequate due to the many extraneous combinations generated which are found to be dominated. In this appendix, we indicate how the above problem can be somewhat alleviated.

We recall that, at any iteration, the intermediate SCP matrix A^{p+1} is obtained from A^p by choosing a 'reducing row' α^p, containing r 1's, say; then forming r submatrices A^p_k $k = 1,\ldots r$ consisting of those rows i of A^p for which $a^p_{ik} = 0$; then applying r-fold or-operations to these A^p_k; and finally retaining, as A^{p+1}, the minimal rows so formed.

Thus, supposing submatrices A^p_k to contain t'_k rows, $1 \leq k \leq r$, we see that there are $\prod_{k=1}^{r} t'_k$ possible r-combinations, (not necessarily distinct, of course), which can be systematically enumerated by implementing (r+1) temporary vectors W_k defined recursively as follows:

$W_o = \phi$, $W_k = W_{k-1} \vee \bar{\alpha}_k$ $(k = 1,\ldots r)$ where $\bar{\alpha}_k$ is the next row in A^p_k to be considered for combining with W_{k-1} remaining unchanged until the whole of A^p_k $(k = r, r-1,\ldots 1)$ has been considered. Thus, by the end of the process, W_r will have contained each of the $\prod_{1}^{r} t^1_k$ r-combinations.

As the 'combining procedure' stops when A^p_1 has been exhausted, we would find it useful to have the 'first' submatrix as small as possible. (Although this is not immediately apparent, it will become so during the further course of the discussion). By extending the reasoning, we see that it is useful to renumber the submatrices so that

$t'_1 \leq t'_2 \leq \ldots \leq t'_r$. Henceforth we assume the A^p_k to be renumbered in the above order.

The method for generating A^{p+1} will be computationally greatly improved if we could recognize a potential non-minimal r-combination, say U, before it has been fully formed; i.e. if it is realized at stage ℓ that W_ℓ would lead to a non-minimal r-combination, we would save $\prod_{k=\ell+1} t'_k$ or-ing operations. (Note: this is one reason why we wish to have the larger t'_k among the later submatrices.

So suppose W_ℓ has just been formed, and that there exists in $A^p_{\ell+1}$ a row $\alpha_{\ell+1}$ with the property that $W_\ell \vee \alpha_{\ell+1} \leq W^*_\ell$. Let $\hat{W}_{\ell+1} = W_\ell \vee \alpha_{\ell+1}$ $(=W_\ell)$. Then, (since $\hat{W}_{\ell+1} = W_\ell$), any $(\ell+1)$-combinations appearing in $W_{\ell+1}$, say $\bar{W}_{\ell+1}$, so long as W_ℓ retains its current value will be such that $\bar{W}_{\ell+1} \geq \hat{W}_{\ell+1}$. But to each of these $t'_{\ell+1}$ values of $W_{\ell+1}$, we must adjoin a further $\prod_{k=\ell+2}^{r} t'_k$ $(r-\ell-1)$-combinations, and thus we can, for any r-combination, $\bar{W}_r$, stemming from $\bar{W}_{\ell+1}$, find a corresponding r-combination, W_r stemming from $W_{\ell+1}$ such that $W_r \leq \bar{W}_r$.

We have thus established the result below.

Lemma : For any A^p_j which contains a row α' with the property that $\alpha' \leq W_{j-1}$, (as defined above), we can ignore all other rows in A^p_j, for as long as W_{j-1} retains its value.

---

* $\alpha_x \leq \alpha_y \iff a_{xj} \leq a_{yj}$ $(j=1, \ldots, n)$

Indeed, $W_j = W_{j-1}$ during the whole of this period, so that we effectively ignore submatrix A_j^p during this time.

In particular, for any row $\bar{\alpha}$ of A^p which appears in each of the submatrices A_k^p, we have $\hat{W}_r = \hat{W}_{r-1} = \ldots = \hat{W}_1 = \bar{\alpha}$, which fact, recalling that A^p itself is a 'minimal' list, ensures that $\bar{\alpha}$ will be one of the (minimal) rows in A^{p+1}. Thus we may, before commencing the 'combining procedure', remove each of such vertices, say they are q in number, from the lists A_k^p and place them immediately into A^{p+1}. The new size of A_k^p will thus be $t_k = t_k' - q \quad k = 1, \ldots, r$, thus immediately saving $\prod_{k=2}^{r} t_i'$ or-ing operations.

The major use of the lemma is however made, dynamically, during the course of the subsequent combining procedure. Thus at any stage, k, when some row α_j of A_k^p has been incorporated by an or-combination into W_k, we now know that any submatrices A_ℓ^p $(\ell > k)$ which contain α_j will be ignored so long as α_j remains in W_k. Therefore a marker to this effect may be immediately placed in submatrix A_ℓ^p, the marker being removed only when α_j is removed from W_k.

Again, if at some stage i corresponding to an 'unmarked' submatrix A_i^p it is found that some row α_s in A_i^p is a subset of W_{i-1} (without being identical to any of its constituent rows), then again A_i^p will be ignored; and moreover, any other A_j^p $(j > i)$ which contains α_s will also be ignored, until (at least) such time as W_{i-1} has been altered; thus again leading to some computational saving.

CHAPTER III

THE GENERATION OF CLIQUES AND MAXIMAL INDEPENDENT SETS OF GRAPHS

1. INTRODUCTION

We have now seen in the preceding chapter that using a Set-Covering algorithm to solve the GCP involves, as a necessary prerequisite, the determination of the maximal independent sets, (MIS), of the graph. One might suppose that these sets could be effectively generated in an efficient manner by systematically enumerating all the independent sets of the graph G, testing (by the attempted addition of a further vertex) to see if they are maximal, and storing those sets which pass the test. However, due to the fact that during the process many independent sets will be generated only to be rejected when they are found to be non-maximal, this direct procedure does not prove to be very effective for graphs of reasonable size (e.g. $n>20$). As an example consider the portion of the graph G of Figure 9 which consists of the MIS $\{1,2,3,4\}$; the appropriate independent sets could then have been generated in the sequence $\{1\}$, $\{1,2\}$, $\{1,2,3\}$, $\{1,2,3,4\}$, $\{2\}$, $\{2,3\}$, $\{2,3,4\}$, $\{3\}$, $\{3,4\}$, of which, in particular, $\{2,3,4\}$, and $\{3,4\}$ would be rejected as being subsets of previously generated sets.*

In fact, as was mentioned in Chapter 1, the MIS of a graph G corresponds precisely to the cliques of the complement graph $\bar{G}$. Thus, since the complement of $\bar{G}$ is obviously G, the problem of determining the MIS of a graph - G - is then equivalent to that of determining the cliques of a graph - $\bar{G}$ -, and vice versa. Since most of the literature in this field (see below) is addressed to problem of generating cliques, rather than MIS, we shall follow this convention and describe the algorithms of Section 2 of this chapter in terms of cliques.

---

* For convenience, we often (in the examples) refer to a vertex x_i by its subscript i alone.

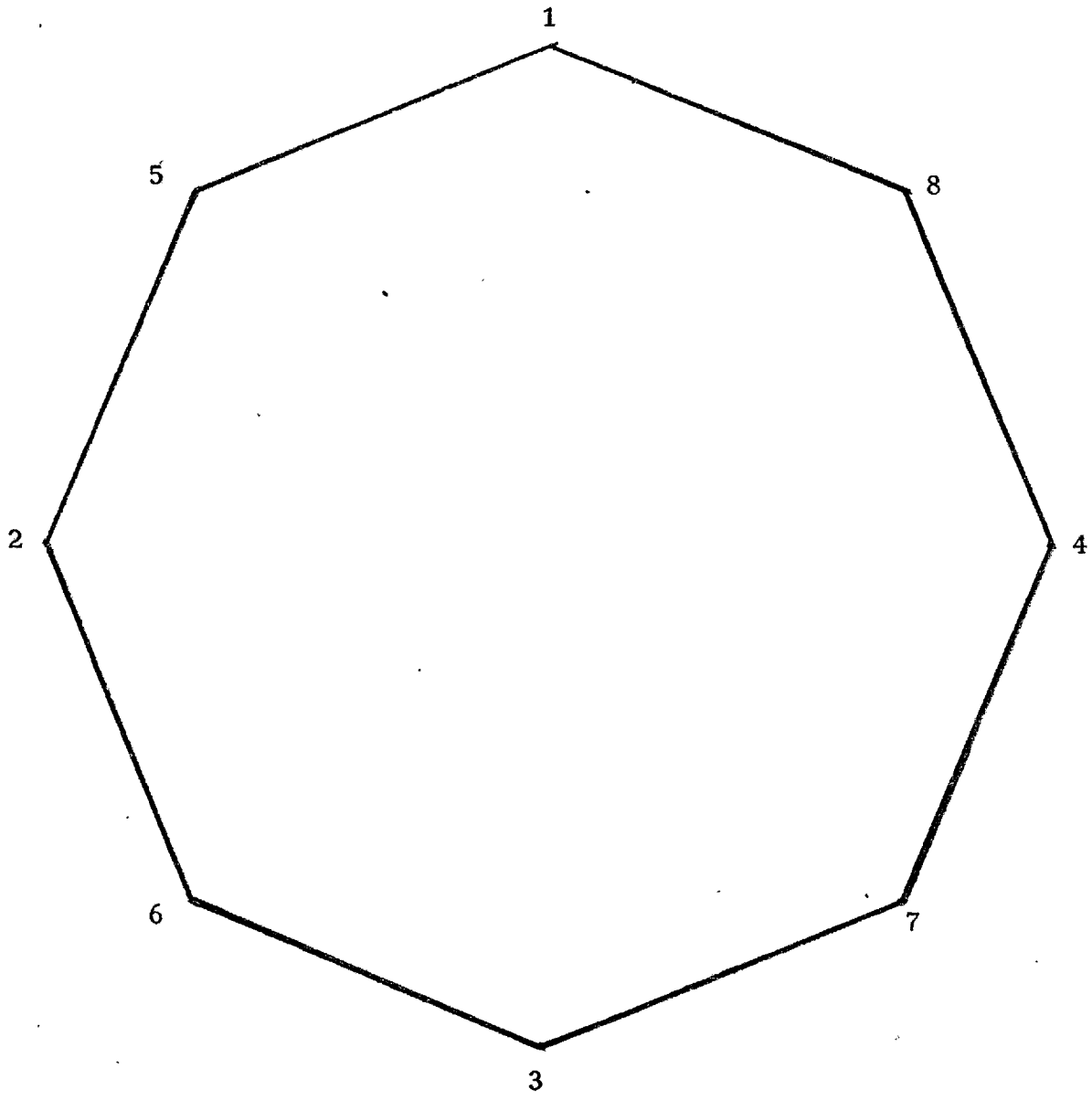


FIGURE 9

The clique-determination problem has intrinsic applications in addition to serving as part of a graph-colouring algorithm, these being mainly in the fields of taxonomy and data 'clustering', [2,8]. Thus, suppose one wishes to divide a group of objects, or attributes, into smaller subgroups, each of which is to consist of 'closely related' objects. Then one would assign numerical values to the degree of relationship (or similarity, as it is generally called) and determine a threshold value of similarity below which no relationships are said to be 'close'. A graph G is then defined with the objects as vertices, two vertices being adjacent if their similarity exceeds the threshold value; and the cliques of this graph are determined. The one could use (some of) these cliques as the required clusters or, if desired one could operate further on these cliques (perhaps by joining some together) before producing the clusters. One example of such an application is given in [7] where the objects are index terms in documents.

In the next section, we describe various algorithms which have been proposed for the clique determination problem, together with a computational comparison of these methods; while in Sections 4 and 5, we shall develop the theory, and derived algorithms, for the determination of the families of cliques and MIS of a graph from those of certain related graphs. Finally, in an appendix to this chapter, we formulate the algorithms of Section 2 in greater detail.

2. ALGORITHMS FOR CLIQUE DETERMINATION

In this section we present the main methods for determining the cliques of a graph. An overview of each method is described below while detailed descriptions of the algorithm together with a worked example to illustrate its operation is given in the Appendix to this chapter.

2.1 The Straight-forward Approach

With this method, [5], the vertices $\{x_1, x_2, \ldots x_n\}$ are considered in their indexed order, and the cliques are built up individually according to a depth-first backtrack procedure. Thus, at any level r, say, there will have been recorded,

(i) the r vertices $\{x'_1, x'_2, \ldots x'_r\}$ of the complete set , K_r, currently being generated,

(ii) the set , V_r, consisting of those vertices which have already been considered (in previous stages) for augmenting K_r, and

(iii) the set, W_r, consisting of those vertices which are candidate vertices - not previous considered - for augmenting K_r.

Sets V_r and W_r will each then consist of vertices adjacent to every vertex of K_r where, due to the indicial order whereby vertices are considered, $x'_1 < x'_2 < x'_3 < \ldots < x'_r$, and each vertex in V_r is of index less than that of x'_r while each vertex in W_r is of index greater than that of x'_r.

To branch forward, then, one chooses x'_{r+1} as the candidate vertex of lowest index in W_r, appends this vertex to the set K_r to

form K_{r+1}, and establish the sets V_{r+1} and W_{r+1} - at level (r+1) - by removing from V_r and W_r, respectively, any vertices which are not adjacent to x'_{r+1}. When a level, t, is reached at which there are no further candidate vertices (i.e. $W_t = \phi$), then if V_t is not empty, K_t must be a subset of some previous generated clique (containing some vertex of V_t as well as K_t) and so a backtracking step must be taken. Thus, the vertex x'_t is removed from W_{t-1} and appended to V_{t-1} (signifying that x'_t has now been fully considered with regard to any augmentations of K_{t-1} - which would involve x'_t -), r is reset to (t-1), and a new vertex x'_{r+1} is now sought from the updated W_r in order to branch forward once more (i.e. forming another K_{r+1}) etc.

On the other hand, if V_t be empty, then K_t is maximal, and so (the clique) K_t can be added to the list of cliques before taking a backtracking step as above.

By beginning the process with $x'_1 = x_1$ (and taking $W_o = X$, $V_o = \phi$), and terminating the procedure when due to backtrack from x_n at the first level, all the cliques of $G = (X, \Gamma)$, - and only these cliques - will be generated in this manner.

2.2 Improvements to the Straight-forward Method

The previous method can be greatly improved, [6], by incorporating a lookahead procedure (or bound) which would allow the early termination (i.e. backtracking at an early stage) of those forward

branchings which do not, in fact, lead to cliques.

Now, suppose that at some level p, the set V_p contains some vertex, x_s, which is adjacent to each of the vertices of W_p, the set of candidate vertices. It is then apparent that augmenting K_{p-1} by any (or indeed, all) vertices of W_p will still result in the appropriate V_r still containing x_s; so that, in particular, V_t will not be empty even when W_t has been exhausted, at which time it will then be seen that K_t is not maximal. Therefore, if any such vertex x_s were found in V_p, one could backtrack immedately at level p.

Again, this method can be even more dramatically improved,[6] by dynamically altering the ordering of the vertices selected throughout the course of the algorithm, with the objective of attaining such a vertex x_s (as described above) at the earliest possible stage, thus enabling backtracking steps,as above, to be taken at the earliest opportunity. This can be readily accomplished by recognizing that, if a counter is kept for each vertex of V_r to indicate the number of candidate vertices (in W_r) to which it is not connected, then moving a selected candidate x'_r from W_r into V_r (which would occur when backtracking from x'_r) would decrease by unity all counters of the vertices of V_r to which it is not connected, and introduce a new counter of its own. It is to be noted that a backtracking step can then be taken (as was shown above) immediately any counter of a vertex $x \in V_r$ (for the appropriate r) reaches 0 as this would imply that x is adjacent to each vertex remaining in W_r; and also that no

counter can ever be decreased by more than one at any one time.

Therefore we see that by determining that vertex $x_r^* \in V_r$ with the smallest counter and then selecting the candidate vertices x_t' from W_r - for augmentation to K_r, K_{r+1}, etc. - such that x_t' is not connected to x_r^* (for $t = r+1, r+2, \ldots$), will lead to our desired effect of early backtracking-by-bounding. Note that this strategy will lead to the earliest possible bounding unless, when backtracking in the normal way (i.e. in the method of the previous section) from some level s, $s > r$, the new vertex x_s' introduced into V_s (where $V_s \subseteq V_r$) has an associated counter smaller than that of the corresponding counter of x_r^*. In such a case (which occurs particularly in the first forward branchings when V_r is in fact empty) one would then choose x_s' as the 'target vertex' (so that $x_s^* = x_s'$) when branching forward again from level s.

It is reported in [6] that when the above vertex-reordering and bounding strategies were incorporated into the basic tree-search algorithm, the time-per-clique generated for any graph is independent of the size of the graph (i.e. the number of cliques in the graph) and so this method is approaching the best-possible [6]. See also Section 3 where our computational results tend to bear this out.

2.3 A Breadth-first Algorithm

This method, [8], operates by determining at each level r of the search-tree, certain categories C_r^k into which all cliques of the graph must fall, each of these categories being further divided and refined at each stage, until such time as a level is reached where each category contains a unique clique of G. At this stage, indeed, each category

will, in fact, consist of its corresponding clique, and so all the cliques of G are determined.

Thus, as each clique of G must obviously contain at least one vertex of G, the original category C_o^1 (at the 'zero'th' level of tree) is that all cliques must contain at least one vertex of $X = \{x_1, \ldots, x_n\}$. At the first level this category is subdivided into two subcategories, based on x_1, as follows:

C_1^1 : containing those cliques of G (in C_o^1) which contain x_1,
C_1^2 : containing those cliques of G (in C_o^1) which do not contain X_1

Again, it is obvious that C_1^1 is a subset of $(\Gamma x_1 \cup \{x_1\})$ while each clique in C_1^2 is a subset of $X - \{x_1\}$. At the second level, each of C_1^1 and C_1^2 are further subdivided into two categories (some of which may be empty) by branching on x_2, as follows:

C_2^1 : containing all cliques of G (in C_1^1) which contain both x_1 and x_2.
C_2^2 : containing all cliques of G (in C_1^1) which contain x_1 but not x_2.
C_2^3 : containing all cliques of G (in C_1^2) which contain x_2 but not x_1.
C_2^4 : containing all cliques of G (in C_1^2) which contain neither x_1 nor x_2.

These categories will then contain all the cliques of G as subsets of $(\Gamma x_1 \cup \{x_1\}) \cap (\Gamma x_2 \cup \{x_2\})$, $(\Gamma x_1 \cup \{x_1\}) \cap (X - \{x_2\})$, $(\Gamma x_2 \cup \{x_2\}) \cap (X - \{x_1\})$, $(X - \{x_1\}) \cap (X - \{x_2\})$ respectively. These may not all be distinct, or some may be non-maximal (i.e. subsets of others), and before proceeding to the next level, all duplicates and

non-maximal sets are removed. Continuing in this fashion level by level, it is then easily seen that all the cliques of G, and only those cliques, will be generated by the n^{th} level.

Another approach which is based on dividing the possible cliques of a graph into disjoint categories given in [1] . In this case the categories indicate that any clique in a given category must contain all vertices of some indicated set of vertices and/or at least one vertex of some other set. Also, unlike the method of [8] , the procedure is organised in a depth-first fashion to minimize storage requirements, and hence the procedure is suitable for the decomposition of large-scale problems, as indicated by its author.

2.4 A Point-Removal Method

The idea behind this approach [3 , 4] is to relate the cliques of a graph G to the cliques of a graph $G^i = G - \{x_i\}$, where x_i is a vertex of G by providing a theorem (similar to those indicated in a somewhat different context in Section 4) relating the two families. Then the algorithm involves repeatedly applying this theorem to a sequence of graphs, the first being a single point and the last being the graph G itself, so obtaining all the cliques of G. (An example is given in the Appendix to this Chapter).

In [9] , this idea is further improved to incorporate the removal of more than one vertex in certain instances by means of theorem relating the cliques of G to those of $G - X_i$ where X_i is a set of vertices containing x_i and possibly some others. A recursive algorithm is then developed, and again the results given in [9] suggest that computation time is proportional to the number of cliques in the graph.

In the following section we give computational results for four of the methods of clique- generation which we described in this section.

3. COMPUTATIONAL COMPARISONS

In this Section we introduce more fully the random graph problems which will be used throughout the remainder of this thesis to test the various algorithms, (and which have already been used in the previous chapter). For each order (number of vertices) of a graph beginning at 15 and increasing in units of 5, ten graphs are generated from each of the densities .1,.3,.5,.7,.9, where the density of a graph with n vertices and m arcs is defined as $2m/n(n-1)$, $n(n-1)/2$ being the maximum number of possible arcs in a (complete) graph. The algorithms are then seen in all ten graphs in any category and the results averaged. If not all the graphs were tested before the termination of the computer run, that particular category of (partial) results was ignored. Also only non-trivial results are recorded.

We emphasize once more that, as all the algorithms given in this thesis involve subsetting operations of some type, it is essential, when programming in FORTRAN, to use Boolean operations, if these are available; as indeed they are on the CDC 6000 Computer series. Thus all the original computational results of this thesis were obtained by running FORTRAN programs incorporating Boolean operations on the CDC 6400 at Imperial College.

Finally, it is seen from Table 5 that the results for this section are essentially self-explanatory, with by far the best method for clique-generating of those tested, being the bounding method of Section 2.2.

TABLE 5 CLIQUE-GENERATING METHODS

VERTICES	DENSITY	CLIQUES	TIMES (CDC 6400 seconds)			
			2.1	2.2	2.3	2.4
25	.1	28	.1	.1	.1	.1
	.3	52	.1	.1	.1	.3
	.5	101	.1	.1	.4	.8
	.7	178	.5	.2	1.5	2.4
	.9	291	15.7	.4	6.7	4.5
30	.1	38	.1	.1	.1	.2
	.3	80	.1	.1	.2	.5
	.5	166	.2	.2	.9	2.1
	.7	447	1.5	.6	8.4	13.2
	.9	906	95.2	1.2	61.8	36.6
35	.1	50	.1	.1	.1	.3
	.3	112	.2	.1	.3	.9
	.5	270	.4	.4	2.4	6.6
	.7	844	3.2	1.2	35.8	50.5
	.9	2466	*	3.5		
40	.1	63	.1	.1	.2	.3
	.3	155	.2	.2	.5	1.5
	.5	416	.5	.7	5.2	13.0
	.7	1580	6.8	2.5	130.0	
	.9	7461		11.4		
45	.1	82	.1	.1	.2	.6
	.3	209	.2	.3	1.0	3.1
	.5	646	.9	1.1	11.4	25.1
	.7	3011	13.5	5.1		
	.9	25106		40.2		
50	.1	92	.1	.1	.3	.9
	.3	281	.3	.5	1.7	6.0
	.5	1014	1.4	1.9	34.0	
	.7	5584	29.4	10.2		
	.9	61004		98.3		

* Blanks in the table are due to time or storage excess.

4. CLIQUES OF RELATED GRAPHS

4.1 The Importance of Certain Related Graphs

In Chapter 6, we shall describe some tree-search algorithms for optimal graph colouring, the general stage of each of which involves the derivation from G_k - the subgraph still uncoloured at stage k - of a related graph $\tilde{G}_k$ and the determination of the cliques and the MIS of this related graph $\tilde{G}_k$. Since these related graphs $\tilde{G}_k$ will be different at various stages of the search, (being derived from different graphs G_k) it is of prime importance to the effectiveness of the algorithm to be able to generate the cliques and MIS of the $\tilde{G}_k$ in a manner more efficent than that of any of the direct methods of Section 1. As, in fact, the cliques and MIS of the graph G_k will be known at any stage (having been determined at the previous stage), these sets can be used to generate (indirectly) the cliques and MIS of the graph G_k.

The actual relationship between the graph G_k and the graph $\tilde{G}_k$ is explained in greater detail in Chapter 6, but it is also mentioned here in order to elucidate and provide an understanding of the algorithms presented in this chapter.

Thus, from the given graph $G_k = (X_k, \Gamma_k)$, remove the vertices of a (given) independent set S. Augment the resultant graph, G_k', by a new vertex $\tilde{x}$ (called a pseudo-vertex) which is joined by an arc to those (and only those) vertices of G_k' which were joined (by an arc to any vertex in the set S. This defines the graph $\tilde{G}_k = (\tilde{X}_k, \tilde{\Gamma}_k)$,

where it is immediately apparent that $\tilde{X}_k = ((X_k - S) \cup \{\tilde{x}\})$ and that, in particular $\tilde{\Gamma}_k \tilde{x} = \bigcup_{x_i \in S} \Gamma_k x_i$.

In the remainder of this section, we present some theorems which lead directly to algorithms for generating all the cliques of $\tilde{G}_k$ from the cliques of G_k, and vice versa, which can then be used in the forward branching, and backtracking stages, respectively, of the graph-colouring algorithm. In the following section similar results and their derived algorithms are given for the generation of MIS. In order to simplify the notation, the subscript 'k' is dropped during the remainder of this chapter, and it is to be understood that G and $\tilde{G}$ refer to general graphs at any stage of a tree search.

4.2 Preliminary Theory

It is easily seen that the family, $F_c(G)$, of cliques of G can be partitioned into two disjoint subfamilies

(a) $F_c(G; \bar{S}) = \{ Q_i \in F_c(G) \mid Q_i \cap S = \phi \}$

(b) $F_c(G; S) = \{ Q_i \in F_c(G) \mid Q_i \cap S \neq \phi \}$

By considering these in turn, we can establish the results which will enable us to determined the family $F_c(\tilde{G})$ of cliques of $\tilde{G} \equiv (G - S) \vee \{x\}$ in an efficient manner. Before doing so, however, let us first define two concepts which will be used for ease of notation throughout the remainder of this section.

Thus, given two families E and F (not necessarily distinct) we say E is maximal with respect to (w.r.t.) F if no set of E

is a subset of any set of F. Also, let us write μF to denote the largest subfamily of F which is maximal w.r.t. itself (i.e. μF is the family of those sets of F which are not subsets of any other set of F).

We are now ready to proceed with our analysis :

(a) Consider a clique $Q_k \in F_c(G; \bar{S})$. Since $Q_k \cap S = \phi$, Q_k is wholly contained in $G' \equiv G - S$. As Q_k is a clique of G and $G' \subset G$, Q_k must therefore be a clique of G'. Now, by the definition of G we know that $\tilde{G} - \{\tilde{x}\} \equiv G - S \equiv G'$ so that $G' \subset \tilde{G}$, and in fact $\tilde{G}$ contains exactly one more vertex (viz $\tilde{x}$) than G'. Since $\tilde{\Gamma}\tilde{x} = \bigcup_{x_i \in S} \Gamma x_i$, one of the following situations must obtain:

$$\text{(i)}\ Q_k \notin \tilde{\Gamma}\tilde{x}$$

$$\text{or (ii)}\ Q_k \subseteq \tilde{\Gamma}\tilde{x}$$

In the former case, since $Q_k \cup \{\tilde{x}\}$ is not a completely connected set of $\tilde{G}$ ($\because \exists\, x_t \in Q_k$ s.t. $x_t \notin \tilde{\Gamma}\tilde{x}$) whilst Q_k is a clique of $\tilde{G} - \{\tilde{x}\}$, then Q_k must be a clique of $\tilde{G}$. In the latter case, since Q_k is a clique of $\tilde{G} - \{\tilde{x}\}$ and $Q_k \cup \{\tilde{x}\}$ is a completely connected set of $\tilde{G}$, $Q_k \cup \{\tilde{x}\}$ is obviously a clique of G.

Let us now denote by $F_c^1(\tilde{G}; \bar{S})$ the family $\left\{ Q_k \,\middle|\, Q_k \in F_c(G, \bar{S}),\ Q_k \notin \bigcup_{x_i \in S} \Gamma x_i \right\}$

by $F_c^2(\tilde{G}; \bar{S})$ the family $\left\{ (Q_k \cup \{\tilde{x}\}) \,\middle|\, Q_k \in F_c(G, \bar{S}), Q_k \subseteq \bigcup_{x_i \in S} \Gamma x_i \right\}$

and by $F^2(\tilde{G}; \bar{S})$ the family $\left\{ ((Q_k \cap \bigcup_{x_i \in S} \Gamma x_i) \cup \{\tilde{x}\}) \,\middle|\, Q_k \in F_c^1(\tilde{G}, \bar{S}) \right\}$

Then since, by definition, $F_c(G; \bar{S})$ consists of distinct cliques Q_k, the proofs of the following lemmas are immediate.

Lemma 1 $F_c^1(\tilde{G}; \bar{S})$ is a family of cliques of G which is maximal w.r.t. itself.

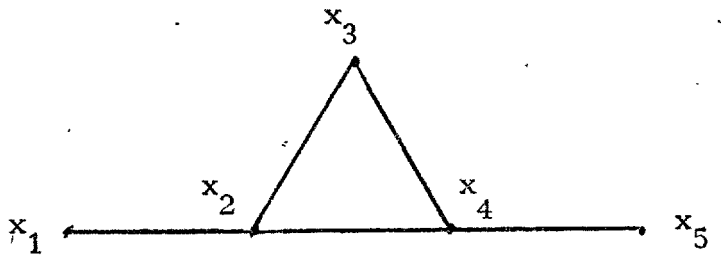
Lemma 2 $F_c^2(\tilde{G}; \bar{S})$ is a family of cliques of G which is maximal w.r.t. itself.

We can also show:

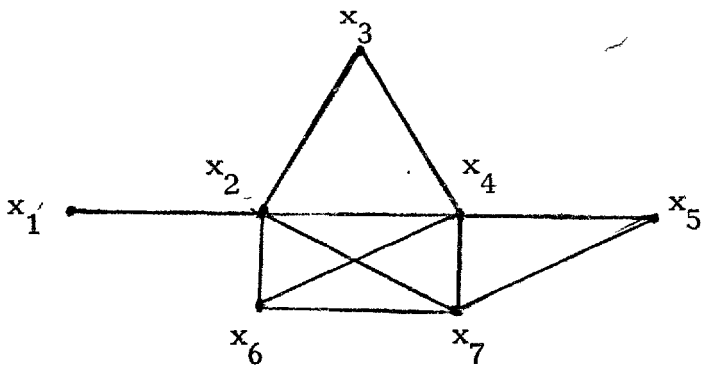
Lemma 3 Both $F_c^1(\tilde{G}; \bar{S})$ and $F_c^2(\tilde{G}; \bar{S})$ are maximal w.r.t. each other.

Proof: Since any set of $F_c^2(\tilde{G}; \bar{S})$ contains $\tilde{x}$ while no set of $F_c^1(\tilde{G}; \bar{S})$ does, it is obvious that $F_c^2(\tilde{G}; \bar{S})$ is maximal with respect to $F_c^1(\tilde{G}; \bar{S})$. On the other hand, suppose there exist cliques $Q_a \in F_c^1(\tilde{G}; \bar{S})$ and $Q_b \in F_c^2(\tilde{G}; \bar{S})$ with $Q_a \subseteq Q_b$; and let $Q_b' = Q_b - \{\tilde{x}\}$ so that $Q_b' \in F_c(G; \bar{S})$. Since $\tilde{x} \notin Q_a$, we must have $Q_a \subseteq Q_b'$. But $Q_a, Q_b' \in F_c(G; \bar{S})$, so that this contradicts the fact that $F_c(G; \bar{S})$ is maximal w.r.t itself. Thus $F_c^1(\tilde{G}; \bar{S})$ is maximal w.r.t. $F_c^2(\tilde{G}; \bar{S})$, so completing the proof ::

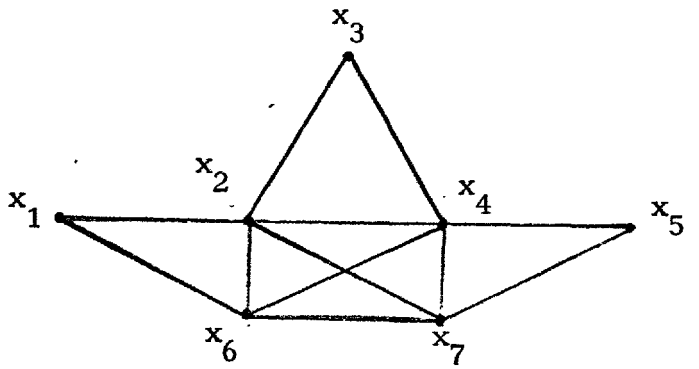
It is similarly true that any set of $F^2(\tilde{G}, \bar{S})$ may be a clique of $\tilde{G}$, as can be seen by considering the graph G of Figure 10(a), where, with S as the independent set $\{1,5\}$, the set $\{2,4,\tilde{x}\} \in F^2(\tilde{G}; \bar{S})$ is a clique of $\tilde{G}$. However, as the graphs of Figure 10(b), and 10(c), respectively, show, $F^2(\tilde{G}; \bar{S})$ is neither maximal w.r.t. itself, nor



(a)



(b)



(c)

FIGURE 10

w.r.t. $F_c^2(\tilde{G}; \bar{S})$, since, (again with $S = \{1,5\}$), in the former case $\{2,4,\tilde{x}\} \subset \{2,4,7,\tilde{x}\} \in F^2(\tilde{G}; \bar{S})$, while in the latter case $\{2,4,\tilde{x}\} \subset \{2,4,6,7,\tilde{x}\} \in F_c^2(\tilde{G}; S)$.

Let us now determine $\mu F^2(\tilde{G}; \bar{S})$, remove from this family any set which is a subset of any set of $F_c^2(\tilde{G},\bar{S})$, and denote the resultant family by $F^3(\tilde{G}; \bar{S})$. Then the following lemma is easily shown to be true.

Lemma 4 $F^3(\tilde{G}; \bar{S})$ is maximal w.r.t. itself and w.r.t. both $F_c^1(\tilde{G}; \bar{S})$ and $F_c^2(\tilde{G}; \bar{S})$.

Proof: By construction $F^3(\tilde{G}; \bar{S})$ is maximal w.r.t itself and $F_c^2(\tilde{G}; \bar{S})$. Also each set of $F^3(\tilde{G}; \bar{S})$ contains $\tilde{x}$, unlike any set of $F_c^1(\tilde{G}; \bar{S})$. Hence $F^3(\tilde{G}; \bar{S})$ is maximal w.r.t. $F_c^1(\tilde{G}; \bar{S})$. On the other hand, $F_c^1(\tilde{G}; \bar{S})$ consists of cliques of G, and each set in $F_c^2(\tilde{G}; \bar{S})$ contains a clique of G as a subset, while $F^3(\tilde{G}; \bar{S})$ consists only of proper subsets of cliques (together with $\tilde{x}$), so completing the proof of the lemma ::

(b) Let us now turn to our second alternative above and consider a clique $Q_h \in F_c(G; S)$, so that $Q_h \cap S \neq \phi$. Recalling from relation (1-3) that $|Q \cap T| \leq 1$ for any completely connected set Q and independent set T, we then have that $|Q_h \cap S| = 1$, so that we can write $\{x_t\} = Q_h \cap S$. Put $Q_h' = Q_h - \{x_t\}$. Again, since $Q_h' \subseteq \Gamma x_t \subseteq \tilde{\Gamma}\tilde{x}$, we see that $Q_h' \cup \{\tilde{x}\}$ is a completely connected

set of $\tilde{G}$. It does not, however, follow that $Q_h' \cup \{\tilde{x}\}$ is a clique of $\tilde{G}$. Indeed, let us define $F^1(\tilde{G}; S) = \{((Q_h - (Q_h \cap S)) \cup \{\tilde{x}\})\}$ for all $Q_h \in F_c(G; S)$. Then we have the following lemma.

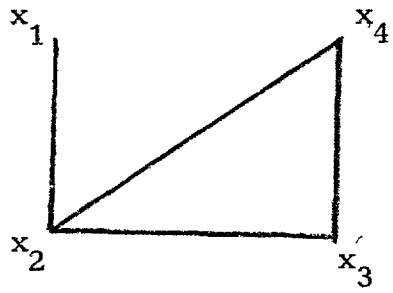
Lemma 5 $F^1(\tilde{G}; S)$ is maximal w.r.t. $F_c^1(\tilde{G}, \bar{S})$.

Proof: For, any set of $F^1(\tilde{G}; S)$ contains $\tilde{x}$, while no set of $F_c^1(G; \bar{S})$ does ::

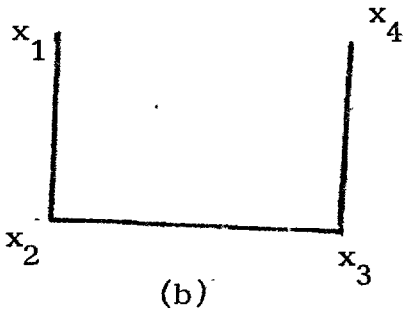
On the other hand, the graphs in Figures 11(a) and 11(b) demonstrate that $F^1(\tilde{G}; S)$ is maximal neither w.r.t. itself nor w.r.t. $F_c^2(\tilde{G}; \bar{S})$. Thus for the graph G of Figure 11(a), with $S = \{1,4\}$, we have $\{1,2\} \in F_c(G; S)$, $\{2,3,4\} \in F_c(G; S)$, so that both $\{2,\tilde{x}\}$ and $\{2,3,\tilde{x}\}$ are in $F^1(\tilde{G}; S)$, with $\{2,\tilde{x}\} \subset \{2,3,\tilde{x}\}$. Similarly, with $S = \{1,4\}$ for the graph G of Figure 11(b), $\{2,\tilde{x}\} \in F^1(\tilde{G}; S)$ while $\{2,3\} \in F_c(G; \bar{S})$ so that $\{2,3,\tilde{x}\} \in F_c^2(\tilde{G}; \bar{S})$, with $\{2,\tilde{x}\} \subset \{2,3,\tilde{x}\}$.

Again, in the graph of Figure 10(a), the sets $\{2,\tilde{x}\} \in F^1(\tilde{G}; S)$ and $\{2, 4, \tilde{x}\} \in F^3(\tilde{G}; \bar{S})$ (with $S = \{1,5\}$) demonstrate that $F^1(\tilde{G}; S)$ is not maximal w.r.t. $F^3(G; \bar{S})$ either.

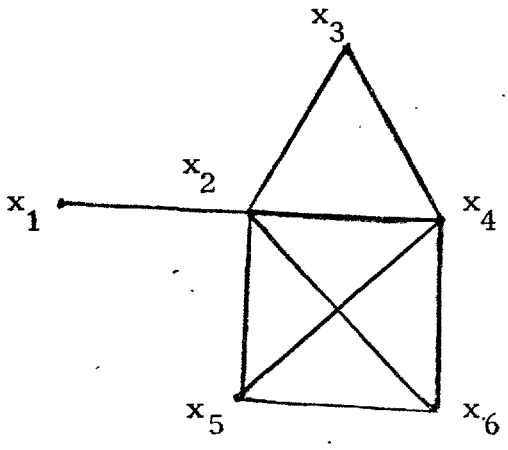
However, putting $F^2(\tilde{G}; S)$ as the largest subfamily of $F^1(\tilde{G}; S)$ which is maximal w.r.t. both $F_c^2(\tilde{G}; \bar{S})$ and $F^3(\tilde{G}; \bar{S})$ and writing $F_c^3(\tilde{G}; S) = \mu F^2(\tilde{G}; S)$ we have that $F_c^3(\tilde{G}; S)$ consists of cliques of $\tilde{G}$, and moreover, by definition, $F_c^3(\tilde{G}; S)$ is maximal w.r.t. each of $F_c^1(\tilde{G}; \bar{S})$, $F_c^2(\tilde{G}; \bar{S})$, $F^3(\tilde{G}; \bar{S})$ and



(a)



(b)



(c)

FIGURE 11

itself. We can also show the following result.

Lemma 6 Both $F_c^1(\tilde{G}; \bar{S})$ and $F_c^2(\tilde{G}, \bar{S})$ are maximal w.r.t. $F_c^3(\tilde{G}; S)$.

Proof: Suppose $Q_a \in F_c^1(\tilde{G}, \bar{S})$. Since $\tilde{x} \notin Q_a$, $\exists x_r \in Q_a$ s.t. $\tilde{x} \notin \Gamma x_r$ But every set of $F_c^3(\tilde{G}; S)$ contains $\tilde{x}$, so that no set of $F_c^3(\tilde{G}; S)$ can contain x_r (since these sets are all completely connected sets). Since some such vertex x_r can be found for any set $Q_a \in F_c^1(\tilde{G}; \bar{S})$, it follows that $F_c^1(\tilde{G}; \bar{S})$ is maximal w.r.t. $F_c^3(G; \bar{S})$.

On the other hand, suppose $\exists Q_a \in F_c^2(\tilde{G}; \bar{S})$, $Q_b \in F_c^3(\tilde{G}; S)$ with $Q_a \subseteq Q_b$.

Now, $Q_a = Q_a' \cup \{\tilde{x}\}$ with $Q_a' \in F_c(G; \bar{S}) \subseteq F_c(G)$

and $Q_b = Q_b' \cup \{\tilde{x}\}$ with $(Q_b' \cup \{x_t\}) \in F_c(G; S) \subseteq F_c(G)$

Thus $Q_a \subseteq Q_b \Rightarrow Q_a' \subseteq Q_b'$, which, since Q_b' is only a proper subset of a clique, contradicts the fact that Q_a' is a clique. Therefore $F_c^2(G; \bar{S})$ is maximal w.r.t. $F_c^3(G; S)$, so completing the proof of the lemma ::

However, as the sets $\{2,4,\tilde{x}\} \in F^3(G; \bar{S})$ and $\{2,4,6,\tilde{x}\}$ of $F_c^3(G; S)$ in the graph G of Figure 11(c) (with $S = \{1,5\}$) demonstrate, $F^3(\tilde{G}; \bar{S})$ is not maximal w.r.t. $F_c^3(\tilde{G}; S)$. Hence let us denote by $F_c^3(G; \bar{S})$ the result when those sets of $F^3(\tilde{G}; \bar{S})$ which are subsets of any sets of $F_c^3(\tilde{G}; S)$ are removed

from $F^3(\tilde{G}; \bar{S})$. Thus $F_c^3(\tilde{G}; \bar{S})$ is maximal w.r.t. $F_c^3(\tilde{G}; S)$.

We are now ready to derive the main results of this Section.

4.3 The Main Theorems

Combining lemmas 1-6 we then have the following result.

Theorem 1 $F_c^1(\tilde{G}; \bar{S}) \cup F_c^2(\tilde{G}; \bar{S}) \cup F_c^3(\tilde{G}; \bar{S}) \cup F_c^3(\tilde{G}; S) \subseteq F_c(\tilde{G})$

Proof: The left-hand side consists of the union of four disjoint families of distinct cliques of $\tilde{G}$ ::

Theorem 2 $F_c(\tilde{G}) \subseteq F_c^1(\tilde{G}; \bar{S}) \cup F_c^2(\tilde{G}; \bar{S}) \cup F_c^3(\tilde{G}; \bar{S}) \cup F_c^3(\tilde{G}; S)$

Proof: Suppose $\tilde{Q} \in F_c(\tilde{G})$. Then either $\tilde{x} \in \tilde{Q}$ or $\tilde{x} \notin \tilde{Q}$.

In the former case, $Q' = \tilde{Q} - \{\tilde{x}\}$ is a completely connected set of $G' = \tilde{G} - \{\tilde{x}\}$. Since $\tilde{x}$ is the pseudo-vertex corresponding to an independent set of G, where $G' \subset G$, we know from (1-3) that any clique of G of which Q' is a subset can contain at most one vertex of S. Assume such a vertex exists; call it x_t. Now consider any vertex x_r of $(G' - Q')$. If $((Q' \cup \{x_t\}) \cup \{x_r\})$ were a completely connected set of G, then, since $\Gamma x_t \subseteq \tilde{\Gamma}\tilde{x}$, $\tilde{Q} \cup \{x_r\}$ would be a completely connected set of $\tilde{G}$, so contradicting the fact that $\tilde{Q}$ is a clique of $\tilde{G}$. Thus in this case $((\tilde{Q} - \{\tilde{x}\}) \cup \{x_t\})$ must be a clique of G, so that $\tilde{Q} \in F_c^3(\tilde{G}; S)$.

On the other hand, if no such vertex x_t exists, that is $\nexists x_t \in S$ s.t. $Q' \subseteq \Gamma x_t$ then, since $Q' \cup \{\tilde{x}\}$ is neverthless a completely connected set of G, we must obviously have $Q' \subseteq \bigcup_{x_i \in S} \Gamma x_i$; therefore, (since $\tilde{\Gamma}\tilde{x} = \bigcup_{x_i \in S} \Gamma x_i$) either $\tilde{Q} \in F_c^1(\tilde{G}; \bar{S})$ or $\tilde{Q} \in F_c^3(\tilde{G}; \bar{S})$, depending on whether or not Q' is a clique of G'.

Finally, if $\tilde{x} \notin \tilde{Q}$ then obviously $\exists x_t \in S$ with $\tilde{Q} \subseteq \Gamma x_t$. Furthermore $\tilde{Q} \not\subseteq \bigcup_{x_i \in S} \Gamma x_i$, as otherwise $\tilde{Q} \cup \{\tilde{x}\}$ would be a completely connected set of $\tilde{G}$, contradicting the maximality of $\tilde{Q}$. Hence in this case $\tilde{Q} \in F_c^2(\tilde{G}; \bar{S})$. As all the possibilities have thus been enumerated, the theorem is proved ::

Combining Theorems 1 and 2 establishes Theorem 3 below, which in view of the preceding analysis leads directly to the algorithms that follows.

Theorem 3 $F_c(\tilde{G}) = F_c^1(\tilde{G}; \bar{S}) \cup F_c^2(\tilde{G}; \bar{S}) \cup F_c^3(\tilde{G}; \bar{S}) \cup F_c^3(\tilde{G}; S)$.

Before formally describing the algorithms which are derived from the previous theory, we briefly indicate the notation for algorithm specification which will be used throughout this thesis. Thus, the algorithms will be formally described in basically the standard (iterative) manner, by detailing the various steps of the algorithm together with an indicator at each step to direct the 'flow of control' to whichever step is to follow. However, one modification to this basic strategy is the introduction of looped sequences of substeps, similar to DO-LOOPS in FORTRAN. Such a looped sequence occuring at Step α, say, of an

algorithm would be denoted as follows.

Step α. For x = ...:

(substep) α. 1.

α. 2.

α. 3.

Step α'. Iterate

Step α - α' then represents the DO-LOOP with 'Iterate' being equivalent to the CONTINUE statement and indicating a return to substep α.1 for the next value of the loop-variable x, until such time as all values of x have been exhausted*, in which case the flow of control of the program will be to go to Step $(\alpha + 1)$.

4.4 Clique-Generation Algorithms based on Theorem 3

(i) Forward Algorithm to determine $F_c(\tilde{G})$ from $F_c(G)$:

The algorithm determines $F_c(\tilde{G})$, the family of cliques of $\tilde{G}$, given $F_c(G) = \{Q_1, \ldots, Q_t\}$, the family of cliques of G, where $\tilde{G} = (G - S) \cup \{\tilde{x}\}$ with S an independent set of G, $\tilde{x}$ the corresponding pesudo-vertex.

Step 0. Set $F_1 = \phi$, $F_2 = \phi$, $F_3' = \phi$, $F_4' = \phi$; put $L = \bigcup_{x_i \in S} \Gamma x_i$, $(t = |F_c(G)|)$.

Step 1. For i = 1,2,...,t:

1.1 If $Q_i \cap S \neq o$ go to 1.4; otherwise go to 1.2.

1.2 If $Q_i \subseteq L$, update $F_1 = F_1 \cup \{(Q_i \cup \{\tilde{x}\})\}$ and go to Step 1'; otherwise go to 1.3.

---

* As usual, if one of the substeps directs control out of the sequence, this will then override the implicit sequence.

1.3 Update $F_2 = F_2 \cup \{Q_i\}$, update $F_3' = F_3' \cup \{(Q_i \cap L) \cup \{\tilde{x}\}\}$

go to Step 1'.

1.4 Update $F_4' = F_4' \cup \{(Q_i \cup \{\tilde{x}\}) - (Q_i \cap S)\}$ and go to Step 1'.

Step 1'. Iterate.

Step 2. Remove from F_4' any set Q_a for which $\exists Q_r \in F_1$ with $Q_a \subseteq Q_r$.

Step 3. Determine $F_4 = \mu F_4'$, the family of maximal sets of F_4'.

Step 4. Remove from F_3' any set Q_a for which $\exists Q_r \in F_1$ with $Q_a \subseteq Q_r$, as well as any set Q_b for which $\exists Q_s \in F_4$ with $Q_b \subseteq Q_s$.

Step 5. Determine $F_3 = \mu F_3'$.

Step 6. Put $F_c(\tilde{G}) = F_1 \cup F_2 \cup F_3 \cup F_4$; and STOP.

Example We now apply this algorithm to find the cliques of the graph $\tilde{G}$ of Figure 12a which is obtained by replacing the independent set $S = \{x_3, x_6\}$ of the graph G in Figure 12b by the pseudo-vertex $\tilde{x}$.

Step 0. $F_c(G) = \{\{1,2,5,8\}, \{1,3,8\}, \{1,7,8\}, \{3,4\}, \{4,5,6\}, \{4,6,7\}, \{6,9\}\}$, $S = \{3,6\}$

$F_1 = \phi$, $F_2 = \phi$, $F_3' = \phi$, $F_4' = \phi$; $L = \{1,4,5,7,8,9\}$

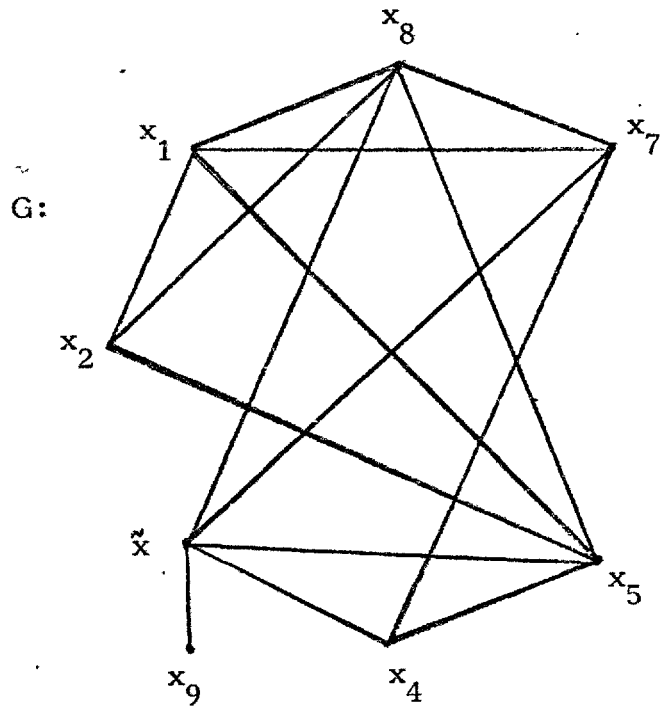
1-1' After the completion of this sequence of steps, we have:

$F_1 = \{\{1,7,8,\tilde{x}\}\}$

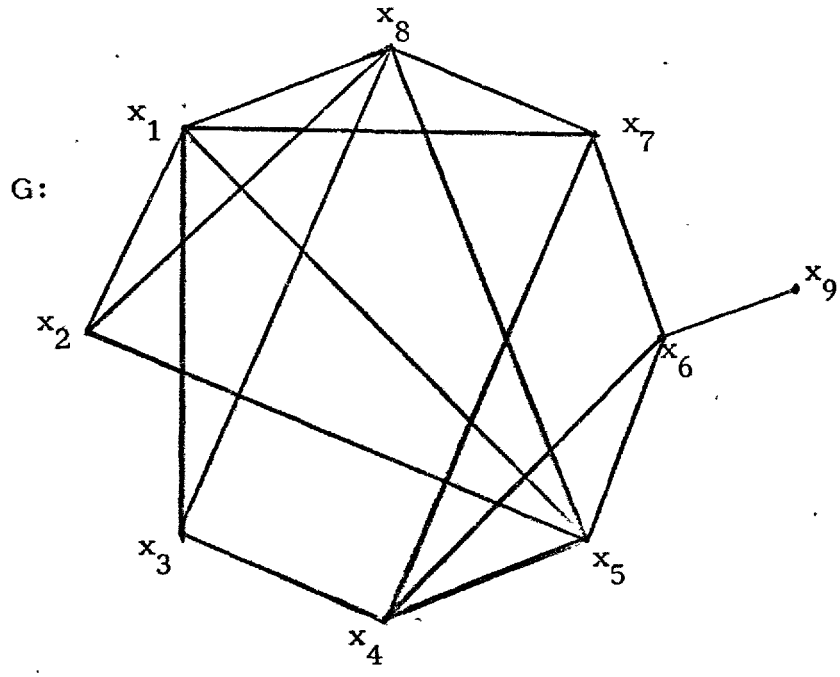
$F_2 = \{\{1,2,5,8\}\}$

$F_3' = \{\{1,5,8,\tilde{x}\}\}$

$F_4' = \{\{1,8,\tilde{x}\}, \{4,5,\tilde{x}\}, \{4,7,\tilde{x}\}, \{9,\tilde{x}\}\}$



(a)



(b)

FIGURE 12

2-3. $F_4 = \{\{4,5,\tilde{x}\}, \{4,7,\tilde{x}\}, \{9,\tilde{x}\}\}$

4-5. $F_3 = \{\{1,5,8,\tilde{x}\}\}$

6. $F_c(\tilde{G}) = \{\{1,7,8,\tilde{x}\}, \{1,2,5,8\}, \{1,5,8,\tilde{x}\}, \{4,5,\tilde{x}\}, \{4,7,\tilde{x}\}, \{9,\tilde{x}\}\}$

(ii) Backward Algorithm to determine $F_c(G)$ from $F_c(\tilde{G})$

This algorithm determine $F_c(G)$ being given $F_c(\tilde{G}) = \{\tilde{Q}_1, \ldots, \tilde{Q}_s\}$

Step 0. Set $F_1 = \phi$, $F_2 = \phi$, $F_3' = \phi$.

Step 1. For $j = 1,2,\ldots,s$:

1.1 If $\tilde{Q}_j \cap \{\tilde{x}\} = \phi$, update $F_1 = F_1 \cup \{\tilde{Q}_j\}$ and go to Step 1';

1.2 For each $x_i \in S$:

1.2.1 If $Q_j' \subseteq \Gamma x_i$, go to 1.2.2; otherwise update $F_2 = F_2 \cup \{(Q_j' \cup \{\tilde{x}_i\})\}$, set MARKER = 1, and go to 1.2'.

1.2.2 If MARKER = 1, go to 1.2.3; otherwise update $F_3' = F_3' \cup \{Q_j'\}$ and go to 1.2.3.

1.2.3 Put $Q_j'' = (Q_j' \cap \Gamma x_i) \cup \{x_i\}$.

If $\exists Q_r'' \in F_3'$ such that $Q_j'' \subseteq Q_r''$, go to 1.2';

otherwise update $F_3' = F_3' \cup \{Q_j''\}$ and go to 1.2'

1.2' Iterate.

Step 1' Iterate.

Step 2. Remove from F_3' any set Q_j'' for which $\exists Q_k \in F_2$ with $Q_j'' \subseteq \tilde{Q}_k$, as well as any set Q_i'' for which $\exists \tilde{Q}_h \in F_1$ with

$Q_i'' \subseteq Q_h$ and denote the resultant family by F_3''

Step 3. Establish $F_3 = \mu F_3''$.

Step 4. Put $F_c(G) = F_1 \cup F_2 \cup F_3$; and STOP.

Example We apply the algorithm to find the cliques of the graph G in Figure 12(b).

Step 0. $F_c(\tilde{G}) = \{\{1,7,8,\tilde{x}\}, \{1,2,5,8\}, \{1,5,8,\tilde{x}\}, \{4,5,\tilde{x}\}, \{4,7,\tilde{x}\}, \{9,\tilde{x}\}\}$

$F_1 = \phi$, $F_2 = \phi$, $F_3' = \phi$ $(s = 6)$

1. $j = 1$; $Q_1' = \{1,7,8\}$; $F_3' = \{\{1,7,8\}, \{1,3,8\}, \{6,7\}\}$; $F_2 = \phi$

1. $j = 2$; $F_1 = \{1,2,5,8\}$

1. $j = 3$

1.1 $Q_3' = \{1,5,8\}$

1.2-1.2' $F_3' = F_3' \cup \{\{1,5,8\}, \{5,6\}\}$; $F_2 = \phi$.

1. $j = 4$

1.1 $Q_4' = \{4,5\}$

1.2-1.2' $F_2 = \{\{4,5,6\}\}$; $F_3' = F_3' \cup \{\{3,4\}\}$

1. $j = 5$

1.1 $Q_5' = \{4,7\}$

1.2-1.2' $F_2 = F_2 \cup \{\{4,6,7\}\}$

1. $j = 6$

1.1 $Q_6' = \{4\}$

1.2-1.2' $F_2 = F_2 \cup \{\{6,9\}\}$

1'. $(F_1 = \{\{1,2,5,8\}\}$, $F_2 = \{\{4,5,6\}, \{4,6,7\}, \{6,9\}\}$,

$F_3' = \{\{1,7,8\}, \{1,3,8\}, \{6,7\}, \{1,5,8\}, \{5,6\}, \{3,4\}\}$

2. $F_3'' = \{\{1,7,8\}, \{1,3,8\}, \{3,4\}\}$
3. $F_3 = \{\{1,7,8\}, \{1,3,8\}, \{3,4\}\}$
4. $F_c(G) = \{\{1,2,5,8\}, \{4,5,6\}, \{4,6,7\}, \{6,9\}, \{1,7,8\}, \{1,3,8\}, \{3,4\}\}$

Specializations

ctions 4.2 and 4.3 of this chapter we analysed the correspond-
ween the cliques of a graph G and those of the graph $\tilde{G}$
by 'replacing' the vertices of some independent set S by a
udo-vertex $\tilde{x}$. In this section we consider the consequences
cializations of this operation, which can often be of use, as
see in subsequent chapters.

In the situation where S is a maximal independent set (see
), none of the results of section 4.2 would be altered; and so
nat S is an MIS would not affect the statement of Theorem 3
rived algorithms (of section 4.4).

In the situation where S is removed but not 'replaced' by
it one is seeking to determine the cliques of a subgraph (G - S)
m those of G - see Chapter 5 -), on the other hand, some mod-
to the algorithms are necessary. Thus, since $\tilde{x}$ is no longer
t of the families (corresponding to) $F_c^3(\tilde{G};\bar{S})$ and $F_c^3(\tilde{G};S)$ it
see that some sets in these families may now be subsets of some
$_k \in F_c^1(\tilde{G};\bar{S})$. Hence additional subsetting checks must be
ted into the algorithms of section 4.4, to take account of the
l possibilities. (Note that by an argument similar to that of
d part of the proof for Lemma 3, $F_c^2(\tilde{G};\bar{S})$ remains maximal
$(\tilde{G};\bar{S})$ even without the addition of $\tilde{x}$).

we shall show in Chapter 5, however, the explicit determination of
s of the subgraph (G - S) is not essential to the graph colouring
described there and so we do not discuss this clique-determination
in any further detail.

5. MIS OF RELATED GRAPHS

We mentioned in Section 4.1 the importance of efficiently determining the cliques and MIS of the graph $\tilde{G}$ (as defined earlier) with regard to some tree-search algorithms for optimal graph colouring. In the previous section we showed how the cliques of G could be used to efficiently generate those of $\tilde{G}$. In this section, we establish, in an analagous way, algorithms for generating the MIS of $\tilde{G}(G)$ from those of $G(\tilde{G})$, so enabling the efficient forward branching and backtracking, respectively, of the colouring algorithms.

5.1 Preliminary Theory

By an analysis analagous to that of the previous section, we can also establish the relationship between $F_m(G)$, the family of MIS of a graph G, and $F_m(\tilde{G})$, the family of MIS of the pseudo-graph $\tilde{G}$ which is obtained by coalescing an independent set, S, of G into the pseudo-vertex $\tilde{x}$.

Thus again the family $F_m(G)$ can be partitioned into two disjoint subfamilies:

(a) $F_m(G; S) = \{M_i \in F_m(G) \mid M_j \cap S \neq \phi\}$

(b) $F_m(G; \bar{S}) = \{M_i \in F_m(G) \mid M_j \cap S = \phi\}$

(a) Consider an MIS $M_j \in F_m(G; S)$. Since both S and M_j are independent sets, there are two possibilites:

either (i) $S \subseteq M_j$, (i.e. $S - M_j \cap S = \phi$),

or (ii) $S \not\subseteq M_j$, (i.e. $S - M_j \cap S \neq \phi$).

In the former situation we have, $S \cap M_j = S$, and, similar to the previous section, $M_j - S$ is an independent set of $G' \equiv G - S \equiv \tilde{G} - \{\tilde{x}\}$. Moreover, as M_j is an independent set, $\Gamma x_i \cap (M_j - S) = \phi \ \forall x_i \in S$;

and therefore, since $\tilde{\Gamma}\tilde{x} = \bigcup_{x_i \in S} \Gamma x_i$, we have $\tilde{\Gamma}\tilde{x} \cap (M_j - S) = \phi$. This shows that $(M_j - S) \cup \{\tilde{x}\}$ is an independent set of $\tilde{G}$. Indeed $\tilde{M}_j = (M_j - S) \cup \{\tilde{x}\}$ is an MIS of $\tilde{G}$. For, by the maximality of M_j in G, $\forall x_k \in (X - M_j)$, $\exists\, x_\ell \in M_j$ s.t. $x_k \in \Gamma x_\ell$, and so (since $M_j = (M_j - S) \cup S$) either $x_\ell \in M_j - S \subseteq \tilde{M}_j$ or $x_\ell \in S$, in which latter case $\Gamma x_\ell \subseteq \tilde{\Gamma}\tilde{x}$, so that $x_k \in \tilde{\Gamma}\tilde{x}$. Thus $\Gamma x_k \cap \tilde{M}_j \neq \phi$ $\forall x_k \in X - M_j$, and so $\tilde{M}_j$ is an MIS of G.

Again, when $S - (M_j \cap S) \neq \phi$, $M_j' = M_j - (M_j \cap S)$ is an independent set of G'. On the other hand, it must be the case that $\exists x_h \in S - (M_j \cap S)$ s.t. $\Gamma x_h \cap (M_j - (M_j \cap S) \neq \phi$ (else $(M_j \cup S)$ would be a larger MIS of G which would contain M_j as a subset); and, since $x_h \in S$, $\Gamma x_h \subseteq \tilde{\Gamma}\tilde{x}$. Thus $\tilde{\Gamma}\tilde{x} \cap (M_j - (M_j \cap S)) \neq \phi$, and so $M_j - (M_j \cap S) \cup \{\tilde{x}\}$ is not an independent set of $\tilde{G}$; although, as we shall see, M_j' need not be an MIS of $\tilde{G}$.

Let us now denote by
$F_m^1(\tilde{G};S)$, the family $\left\{(M_j - S) \cup \{x\} \mid M_j \in F_m(G) \;\&\; S \subseteq M_j\right\}$
and by $F^2(\tilde{G};S)$, the family $\left\{M_j - (M_j \cap S) \mid M_j \in F_m(G) \;\&\; S \not\subseteq M_j\right\}$.

We then have the following results.

Lemma 7 $F_m^1(\tilde{G}; S)$ is a family of MIS of $\tilde{G}$ which is maximal w.r.t itself and w.r.t. $F^2(\tilde{G}; S)$.

Proof: We have already seen that $F_m^1(\tilde{G}; S)$ consists of cliques of $\tilde{G}$.

Again, the existence of sets $\tilde{M}_j, \tilde{M}_k \in F_m^1(\tilde{G}; S)$ with $\tilde{M}_j \subseteq \tilde{M}_k$ would

imply $(M_j - S \cup \{\tilde{x}'\}) \subseteq (M_k - S \cup \{\tilde{x}\})$, M_j, $M_k \in F_m(G)$. But since $S \subseteq M_j$, M_k this would mean $M_j \subseteq M_k$, contradicting the fact that M_j and M_k are MIS of G. Thus $F_m^1(\tilde{G}; S)$ is maximal w.r.t. itself.

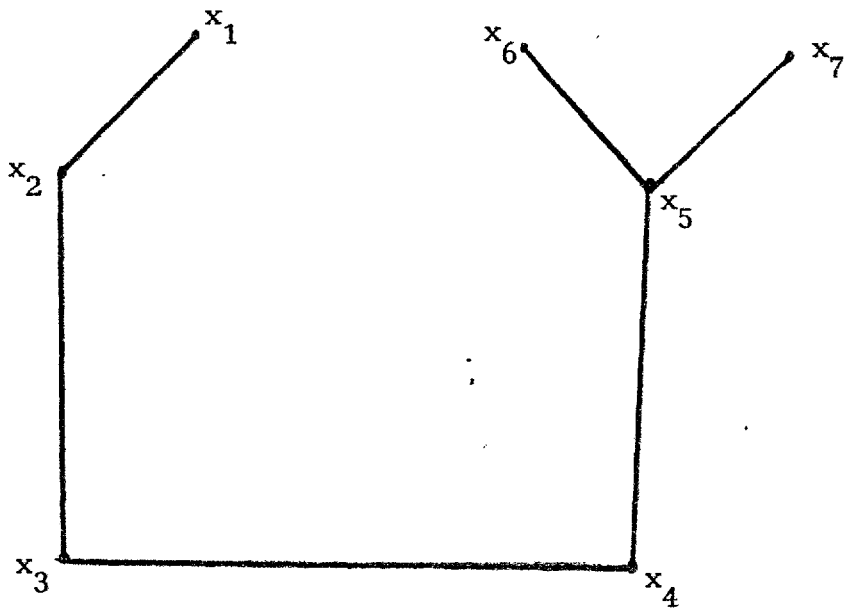
Finally, each set in $F_m^1(\tilde{G}; S)$ contains S as a subset while no set of $F^2(\tilde{G}; S)$ does. Therefore $F_m^1(\tilde{G}; S)$ is maximal w.r.t $F^2(\tilde{G}; S)$ ::

Lemma 8 $F^2(\tilde{G}; S)$ is maximal w.r.t. $F_m^1(\tilde{G}; S)$ but not w.r.t itself.

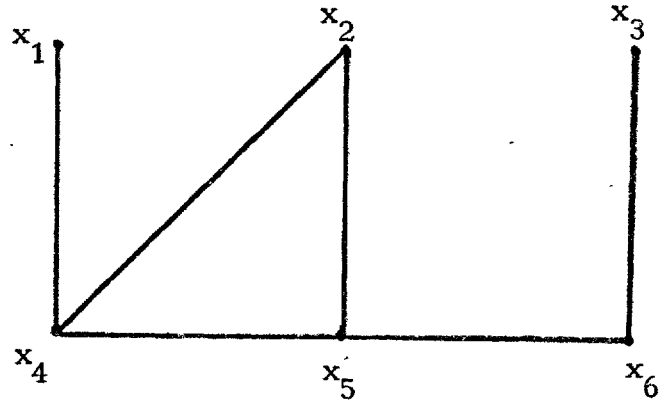
Proof: Suppose $M_j' \in F^2(\tilde{G}; S)$. Then, as we have seen above, $\exists x_k \in M_j'$ s.t. $\Gamma x_k \cap S \neq \phi$. But since S is a subset of each set of $F_m^1(\tilde{G}; S)$, no set of $F_m^1(\tilde{G}; S)$, being an independent set, can contain x_k. Thus $F^2(\tilde{G}; S)$ must be maximal w.r.t. $F_m^1(\tilde{G}; S)$.

On the other hand, consider the graph G in Figure 13(a) with S taken as $\{2,4,6\}$. It is easily seen that $\{1,3,6,7\} \in F_m(G;S)$ so $\{1,3,7\} \in F^2(\tilde{G};S)$, and similarly, that $\{1,4,6,7\} \in F_m(G;S)$, so that $\{1,7\} \in F^2(\tilde{G};S)$. Since $\{1,7\} \subset \{1,3,7\}$, it is then obvious that $F^2(\tilde{G},S)$ is not maximal w.r.t. itself, so completing the proof ::

(b) Turning now to the second alternative, consider an MIS $M_j \in F_m(G;\bar{S})$. Thus $M_j \cap S = \phi$, and as $G' = G - S$ is a subgraph of G, M_j is an MIS of G'. Again, as was shown in the



(a)



(b)

FIGURE 13

analysis leading to $F^2(\tilde{G},S)$, since $M_j - (M_j \cap S) \neq \phi$, $(M_j \cup \{\tilde{x}\})$ is not an independent set of $\tilde{G}$. Therefore, recalling that $\tilde{G} \equiv G' \cup \{\tilde{x}\}$, we see that M_j is an MIS of $\tilde{G}$.

Lemma 9 $F_m(G;\bar{S})$ is a family of MIS of $\tilde{G}$ which is maximal with respect to itself, with respect to $F_m^1(\tilde{G};S)$, and w.r.t $F^2(\tilde{G};S)$.

Proof: We have just shown that $F_m(G;\bar{S})$ consists of MIS of $\tilde{G}$. Since each set of $F_m(G;\bar{S})$ is also a (distinct) MIS of G, $F_m(G;\bar{S})$ is maximal w.r.t. itself.

Now, any set M_t' of $F^2(\tilde{G},S)$ is of the form $M_t' = M_t - (M_t \cap S)$ for some $M_t \in F_m(G;S)$, so that M_t must contain, by definition, some vertex, say x_t, of S. Since any set, M_r, of $F_m(G;\bar{S})$ is an MIS of G which does not contain x_t, M_r must contain some vertex, x_r, which is adjacent to x_t. As x_r and x_t cannot be in the same MIS, $x_r \notin M_t$; and so, since $M_t' \subset M_t$, $x_r \notin M_t'$. Thus for any pair of sets M_r, M_t', where $M_r \in F_m(G;\bar{S})$, $M_t' \in F^2(\tilde{G};S)$, M_r contains some vertex which is not in M_t'. Thus $F_m(G;\bar{S})$ is maximal w.r.t. $F^2(\tilde{G};S)$.

The same argument also shows that $F_m(G;\bar{S})$ is maximal w.r.t. $F_m^1(\tilde{G};S)$, so completing the proof ::

Lemma 10 $F_m^1(\tilde{G};S)$ is maximal w.r.t. $F_m(G;\bar{S})$.

Proof: For every set of $F_m^1(\tilde{G};S)$ contains $\tilde{x}$, while no set of $F_m(G,\bar{S})$ does ::

On the other hand, it can be seen from Figure 13(b) that $F^2(G;S)$ is not maximal w.r.t. $F_m(G;\bar{S})$. Thus, with $S = \{2,3\}$, we have $\{4,6\} \in F_m(G;\bar{S})$ whereas $\{3,4\} \in F_m(G;S)$ so that $\{4\} \in F^2(\tilde{G};S)$. Since $\{4\} \subset \{4,6\}$, the result follows.

Defining $F^3(\tilde{G}; S)$ as that subfamily of $F^2(\tilde{G}; S)$ which is maximal w.r.t $F_m(G; \bar{S})$ and putting $F_m^2(\tilde{G}; S) = \mu F^3(\tilde{G}; S)$, we see that $F_m^2(\tilde{G}; S)$ is a family of cliques of G which is maximal w.r.t each of $F_m^1(\tilde{G}; S)$, $F_m^2(\tilde{G}; S)$, and $F_m(G; \bar{S})$. Also, lemmas 7 and 9 show that the converse is also true (since maximality w.r.t any family F obviously implies maximality w.r.t. any subfamily of F).

5.2 Main Theorems

Lemmas 7 - 10 now lead immediately to the following result.

Theorem 4 $F_m^1(\tilde{G}; S) \cup F_m^2(\tilde{G}; S) \cup F_m(G; \bar{S}) \subseteq F_m(\tilde{G})$.

As expected, (cf the previous section), the converse also holds true.

Theorem 5 $F_m(\tilde{G}) \subseteq F_m^1(\tilde{G}; S) \cup F_m^2(\tilde{G}; S) \cup F_m(G; \bar{S})$.

Proof: Suppose $\tilde{M}_k \in F_m(\tilde{G})$. Then either $\tilde{x} \in \tilde{M}_k$ or $\tilde{x} \notin \tilde{M}_k$.

In the former case we must have $\tilde{M}_k - \{\tilde{x}\} \cap \tilde{\Gamma}\tilde{x} = \phi$, where $\tilde{M}_k - \{\tilde{x}\} \subseteq X' = \tilde{X} - \{\tilde{x}\} = X - S$. Thus $(\tilde{M}_k - \{\tilde{x}\}) \cap \bigcup_{x_i \in S} \Gamma x_i = \phi$, so that $(\tilde{M}_k - \{\tilde{x}\}) \cup S$ is an MIS of G, i.e. $\tilde{M}_k \in F_m^1(\tilde{G})$.

On the other hand, if $\tilde{x} \notin \tilde{M}_k$, we must have $\tilde{M}_k \cap \tilde{\Gamma}\tilde{x} \neq \phi$, with $\tilde{M}_k \subseteq X'$. Hence $\bigcup_{x_i \in S} \Gamma x_i \cap \tilde{M}_k \neq \phi$, thus

either $\Gamma x_i \cap \tilde{M}_k \neq \phi \ \forall x_i \in S$ i.e. $\tilde{M}_k \in F_m(G; \bar{S})$,

or $\Gamma x_i \cap \tilde{M}_k \neq \phi \ \forall x_i \in S' \subset S$

and $\Gamma x_i \cap \tilde{M}_k = \phi \ \forall x_i \in S - S'$ i.e. $\tilde{M}_k \in F_m^2(\tilde{G}; S)$.

Hence the theorem is proved ::

Combining theorems 4 and 5 establishes the following theorem, and again leads to the formulation of two algorithms.

Theorem 6 $\quad F_m(\tilde{G}) = F_m^1(\tilde{G}; S) \cup F_m^2(\tilde{G}; S) \cup F_m(G; \bar{S})$

5.3 MIS - Generating Algorithms based on Theorem 6

(i) Forward Algorithm to determine $F_m(\tilde{G})$ from $F_m(G)$

Let $F_m(G) = \{M_1, M_2, \ldots, M_t\}$

Step 0. Set $F_1 = \phi$, $F_2 = \phi$, $F_3 = \phi$.

Step 1. For $j = 1,2,\ldots t$

1.1 If $M_j \cap S = \phi$, update $F_1 = F_1 \cup \{M_j\}$ and go to Step 1'; otherwise go to 1.2.

1.2 If $M_j \cap S = S$, update $F_2 = F_2 \cup \{(M_j - S) \cup \{\tilde{x}\}\}$ and go to Step 1'; otherwise go to 1.3.

1.3 Update $F_3 = F_3 \cup \{M_j - (M_j \cap S)\}$ and go to Step 1'.

Step 1'. Iterate.

Step 2. Set $F_4 = \phi$ and suppose $F_3 \equiv \{M_1', M_2', \ldots, M_p'\}$

Step 3. For $i = 1,2,\ldots p$

2.1 If $\exists M_r \in F_1$ with $M_i' \subseteq M_r$, go to Step 3'; otherwise go to 2,2.

2.2 Update $F_4 = F_4 \cup \{M_i'\}$, and go to Step 3'.

Step 3'. Iterate.

Step 4. Establish $F_5 = \mu F_4$.

Step 5. Put $F_m(\tilde{G}) = F_1 \cup F_2 \cup F_5$; and STOP.

Example We now apply this algorithm to find the MIS of the graph G of the previous example (Figure 12a) from the MIS of the graph G shown in Figure 12b, again with $S = \{x_3, x_6\}$.

Step 0. $F_m(G) = \{\{1,4,9\}, \{1,6\}, \{2,3,6\}, \{2,3,7,9\}, \{2,4,9\}, \{3,5,7,9\}, \{4,8,9\}, \{6,8\}\}$, $S = \{3,6\}$

$F_1 = \phi$, $F_2 = \phi$, $F_3 = \phi$.

1-1': After the completion of this sequence of steps, we have :

$F_1 = \{\{1,4,9\}, \{2,4,9\}, \{4,8,9\}\}$

$F_2 = \{\{2,\tilde{x}\}\}$

$F_3 = \{\{1\}, \{2,7,9\}, \{5,7,9\}, \{8\}\}$

2. $F_4 = \phi$.

3-3': $F_4 = \{\{2,7,9\}, \{5,7,9\}\}$

4. $F_5 = \{\{2,7,9\}, \{5,7,9\}\}$

5. $F_m(\tilde{G}) = \{\{1,4,9\}, \{2,4,9\}, \{4,8,9\}, \{2,\tilde{x}\}, \{2,7,9\}, \{5,7,9\}\}$

(ii) Backward Algorithm to determine $F_m(G)$ from $F_m(\tilde{G})$

Let $F_m(\tilde{G}) = \{\tilde{M}_1, \tilde{M}_2, \ldots, \tilde{M}_s\}$.

Step 0. Set $F_1 = \phi$, $F_2 = \phi$, $F_3 = \phi$, $F_4' = \phi$.

Step 1. For $i = 1,2,\ldots,s$:

1.1 If $\tilde{x} \in \tilde{M}_i$, update $F_1 = F_1 \cup \{(\tilde{M}_i - \{\tilde{x}\}) \cup S\}$, and go to Step 1'; otherwise go to 1.2.

1.2 If $\Gamma x_k \cap \tilde{M}_i \neq \phi \ \forall x_k \in S$, update $F_2 = F_2 \cup \{\tilde{M}_i\}$; put $S' = S$ and go to Step 1.4; otherwise go to 1.3.

1.3 Put $S'' = \{x_k \in S \mid \Gamma x_k \cap \tilde{M}_i = \phi\}$. For each $x_k \in S''$, update $F_3 = F_3 \cup \{\tilde{M}_i \cup \{x_k\}\}$. Put $S' = S - S''$, and go to 1.4.

1.4 Put $X' = \tilde{M}_i \cup S'$, $\Gamma' = \Gamma \cap X'$, put $G' = (X', \Gamma')$, and go to 1.5.

1.5 Determine $F'_m(G', S')$, the family of MIS of G' which contain some vertex of S'_i, update $F'_4 = F'_4 \cup F'_m(G', S')$, go to Step 1'.

Step 1'. Iterate.

Step 2. Remove from F'_4 any sets which are subsets of any set in $F_1 \cup F_2 \cup F_3$.

Step 3. Establish $F = \mu F'_4$.

Step 4. Put $F_m(G) = F_1 \cup F_2 \cup F_3 \cup F_4$; and STOP.

Example We now apply this algorithm to determine the MIS of the graph G of Figure 12b from the MIS of the graph $\tilde{G}$ in Figure 12a, (with $S = \{x_3, x_6\}$).

Step 0. $F_m(\tilde{G}) = \{\{1,4,9\}, \{2,4,9\}, \{4,8,9\}, \{2,\tilde{x}\}, \{2,7,9\}, \{5,7,9\}\}$, $S = \{3,6\}$,

$F_1 = \phi$, $F_2 = \phi$, $F_3 = \phi$, $F'_4 = \phi$.

1. $i = 1$, $\tilde{M}_1 = \{1,4,9\}$

1.2 - 1.4. $F_2 = \{\{1,4,9\}\}$, $S' = \{3,6\}$; $X' = \{1,3,4,6,9\}$

1.5 $F'_m(G',S') = \{\{1,6\},\{3,6\},\{6,8\}\}$; $F'_4 = F'_m(G',S')$

1. $i = 2$, $\tilde{M}_2 = \{2,4,9\}$

1.2 - 1.4 $F_2 = F_2 \cup \{\{2,4,9\}\}$, $S' = \{3,6\}$, $X' = \{2,3,4,6,9\}$

1.5 $F'_m(G', S') = \{\{1,6\}, \{3,6\}, \{6,8\}\}$, $F'_4 = F'_m(G',S')$

1. $i = 3$, $\tilde{M}_3 = \{4,8,9\}$

1.2 - 1.4 $F_2 = F_2 \cup \{\{4,8,9\}\}$, $S' = \{3,6\}$;

$X' = \{3,4,6,8,9\}$

1.5 $F'_m(G',S') = \{\{3,6\}, \{3,9\}, \{6,8\}\}$, $F'_4 = F'_4 \cup F'(G',S')$

1. $i = 4$, $\tilde{M}_4 = \{2,\tilde{x}\}$

1.1 $F_1 = \{\{2,3,6\}\}$

1. $i = 5$, $\tilde{M}_5 = \{2,7,9\}$

1.3 $S'' = \{3\}$, $F_3 = \{\{2,3,7,9\}\}$, $S' = \{6\}$

1.4 - 1.5 $X' = \{2,6,7,9\}$, $F'_m(G',S') = \{\{2,6\}\}$,

$F'_4 = F'_4 \cup F'_m(G',S')$

1. $i = 6$, $M_6 = \{5,7,9\}$

1.3 $S'' = \{3\}$, $F_3 = F_3 \cup \{\{3,5,7,8\}\}$, $S' = \{6\}$

1.4 - 1.5 $X' = \{5,6,7,9\}$, $F'_m(G',S') = \phi$.

2. $F'_4 = \{\{1,6\}, \{6,8\}\}$

3. $F_4 = \{\{1,6\}, \{6,8\}\}$

4. $F_m(G) = \{\{2,3,6\}, \{1,4,9\}, \{2,4,9\}, \{4,8,9\}, \{2,3,7,9\}, \{3,5,7,9\}, \{1,6\}, \{6,8\}\}$.

5.4 Two Specializations

Again, in this section, we consider the specializations when S is (i) an MIS

(ii) removed from the graph but not 'replaced'.

In the former case, some simplification is now possible. For since $S \equiv M$ is an MIS of G, the only MIS M_i of G for which $M \subseteq M_i$ is M itself; which in turn means that the family $F^1_m(\tilde{G}; M) = \{M_i - M \cup \{\tilde{x}\} \mid M_i \in F_m(G) \,\&\, M \subseteq M_i\}$ degenerates simply into the single set $\{\tilde{x}\}$. Thus Theorem 6 can be restated as: $F_m(\tilde{G}) = F^2_m(\tilde{G}; M) \cup F_m(G; \bar{M}) \cup \{\tilde{x}\}$, so allowing corresponding simplification of the algorithms based on it.

Similarly, as was the case with regard to cliques, the non-existence of $\tilde{x}$ will also affect matters regarding the determination of MIS (section 5.3). Thus the family $F^1_m(G; S)$, formerly defined as $\{(M_i - S \cup \{\tilde{x}\}) \mid M_i \in F_m(G), S \subseteq M_i\}$ must be redefined as $F^1_m(G - S; S) = \{(M_i - S) \mid M_i \in F_m(G), \; S \subseteq M_i\}$, and it is easily seen that $F^1_m(G - S; S)$ is no longer maximal w.r.t. $F^2(G - S; S)$ nor w.r.t $F_m(G; \bar{S})$.

Hence all sets of $F^1_m(G - S; S)$ which are subsets of any sets in $F^2(G - S; S) \cup F_m(G; \bar{S})$ must first be removed from $F^1_m(G - S; S)$ before Theorem 6 can hold as stated. This fact will naturally also be reflected in the algorithms, but we will not go into detail for the moment. Instead, as the forward algorithm of this section is of great practical importance as far as a whole class of graph-colouring algorithms is concerned (see Chapter 5), it will be fully formulated in that chapter, while the amendments to the backward algorithm are reasonably straightforward.

6. REFERENCES

1. AKKOYUNLU, E.A. "The Enumeration of Maximal Cliques of Large Graphs", SIAM J. of Computing 2 (1972), pp 1-6.

2. AUGUSTON, J.G. and MINKER, J. "An Analysis of Some Graph Theoretical Cluster Techniques", Journal of A.C.M. 17 (1970), pp 571-588.

3. BEDNAREK, A.R. and TAULBEE, O.E., "On Maximal Chains", Rev. Romaine Math. Pures et Appliques 11 (1966), pp 23-25.

4. BIERSTONE, E. "Cliques and Generalized Cliques in a Finite Linear Graph", Unpublished Report.

5. BONNER, R.E. "On Some Clustering Techniques", IBM Journal of Research and Development 8 (1964) pp 22-32.

6. BRON, C. and KERBOSCH, J. "Finding all Cliques of an Undirected Graph", Comm. A.CM. 16 (1973), pp 575-577.

7. GOTLEIB, C.C. and KUMPR, S. "Semantic Clustering of Index Terms", Journal of A.C.M. 15 (1968), pp 493-513.

8. JARDINE, N. and SIBSON, R. Mathematical Taxonomy, Wiley and Sons, London, 1971.

9. OSTEEN, R.C. and TOU, J.T. "A Clique-Detection Algorithm based on Neighbourhoods in Graphs", Int. Journal of Computer and Information Sciences", 2 (1973), pp 257-268.

APPENDIX

Formulations and Examples of the Clique-Generating Algorithms

In this appendix, we give the detailed formulations of the main clique-generating algorithms discussed in Section 2 of the previous chapter, and we demonstrate these algorithms by determining all the cliques of the graph in Figure 14 below, which is taken from [1].

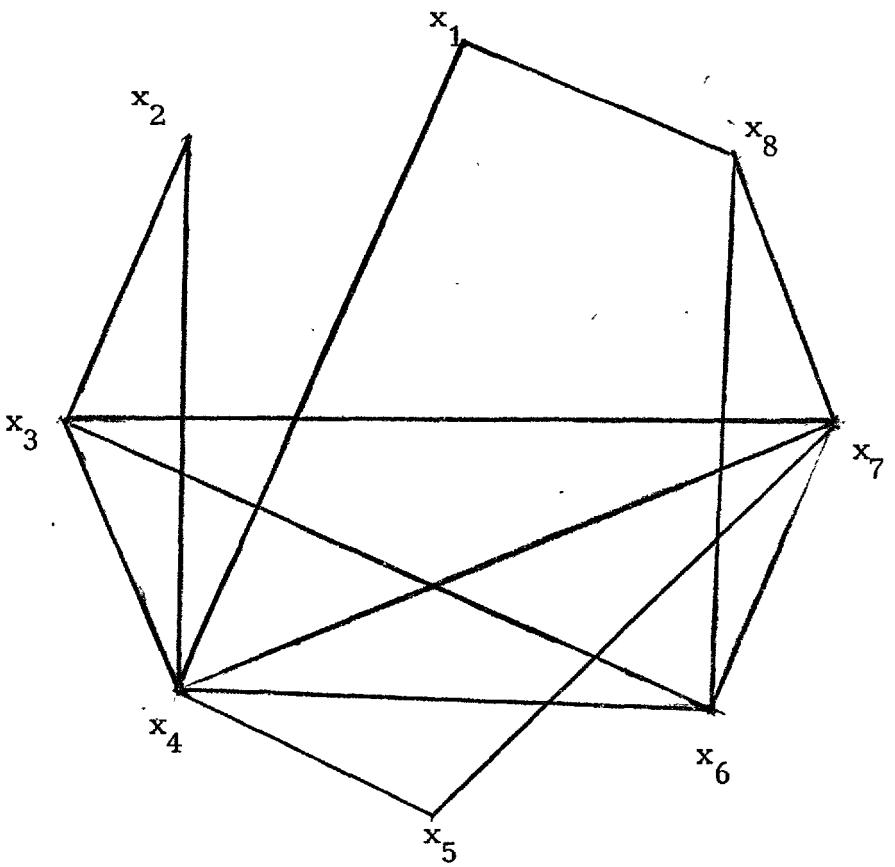


FIGURE 14

A 1. The Algorithm of Section 1.1

Step 0. Set $K_o = \phi$, $W_o = X$, $V_o = \phi$, Set $F = \phi$, and put $r = 1$

Step 1. If $W_r = \phi$ go to Step 2; otherwise find $x'_r = \min\{x_i | x_i \in W_r\}$, establish $K_{r+1} = K_r \cup \{\{x'_r\}\}$, $V_{r+1} = V_r \cap \Gamma x'_r$, $W_{r+1} = W_r \cap \Gamma x'_r$, set $r = r + 1$ and go to Step 1.

Step 2. If $V_r \neq \phi$ go to Step 3; otherwise update $F = F \cup \{K_r\}$ and go to Step 3.

Step 3. Set $r = r - 1$; if $r = 0$ go to Step 4; otherwise update $V_r = V_r \cup \{x'_r\}$, update $W_r = W_r - \{x'_r\}$, and go to Step 1.

Step 4. Put $F_c(G) = F$; and STOP.

Example We demonstrate this algorithm by determining all the cliques of the graph G in Figure 14.

Step 0. ($\Gamma x_1 = \{4,8\}$, $\Gamma x_2 = \{3,4\}$, $\Gamma x_3 = \{2,4,6,7\}$, $\Gamma x_4 = \{1,2,3,5,6,7\}$, $\Gamma x_5 = \{4,7\}$, $\Gamma x_6 = \{3,4,7,8\}$, $\Gamma x_7 = \{3,4,5,6,8\}$, $\Gamma x_8 = \{1,6,7\}$)

$K_1 = \phi$, $W_1 = \{1,2,3,4,5,6,7,8\}$, $V_1 = \phi$, $F = \phi$, $r = 1$

1. $x'_1 = 1$, $K_2 = \{1\}$, $V_2 = \phi$, $W_2 = \{4,8\}$, $r = 2$

1. $x'_2 = 4$, $K_3 = \{1,4\}$, $V_3 = \phi$, $W_3 = \phi$, $r = 3$

1. $W_3 = \phi$

2. $V_3 = \phi$; $F = \{\{1,4\}\}$

3. $r = 2,\ V_2 = \{4\},\ W_2 = \{8\}$

1. $x_2' = 8,\ K_3 = \{1,8\},\ V_3 = \phi,\ W_3 = \phi,\ r = 3$

1. $W_3 = \phi$

2. $V_3 = \phi,\ F = F \cup \{\{1,8\}\}$

3. $r = 2,\ V_2 = \{4,8\},\ W_2 = \phi$

1. $W_2 = \phi$

2. $V_2 \neq \phi$

3. $r = 1,\ V_1 = \{1\},\ W_1 = \{2,3,4,5,6,7,8\}$

1. $x_1' = 2,\ K_2 = \{2\},\ V_2 = \phi,\ W_2 = \{3,4\},\ r = 2$

1. $x_2' = 3,\ K_3 = \{2,3\},\ V_3 = \phi,\ W_3 = \{4\},\ r = 3$

1. $x_3' = 4,\ K_4 = \{2,3,4\},\ V_4 = \phi,\ W_4 = \phi,\ r = 4$

1. $W_4 = \phi$

2. $V_4 = \phi,\ F = F \cup \{\{2,3,4\}\}$

3. $r = 3,\ V_3 = \{4\},\ W_3 = \phi$

1. $W_3 = \phi$

1. $x_2' = 4,\ K_3 = \{2,4\},\ V_3 = \{3\},\ W_3 = \phi,\ r = 0$

1-2. $W_3 = \phi,\ V_3 \neq \phi$

3. $r = 2,\ V_2 = \{3,4\},\ W_2 = \phi$

1-3. $W_2 = \phi$, $V_3 \neq \phi$, $r = 1$, $V_1 = \{1,2\}$, $W_1 = \{3,4,5,6,7,8\}$

1^{5}.* $x'_1 = 3$, $K_2 = \{3\}$, $V_2 = \{2\}$, $W_2 = \{4,6,7\}$;

$x'_2 = 4$, $K_3 = \{3,4\}$, $V_3 = \{2\}$, $W_3 = \{6,7\}$

$x'_3 = 6$, $K_4 = \{3,4,6\}$, $V_4 = \phi$, $W_4 = \{7\}$

$x'_4 = 7$, $K_5 = \{3,4,6,7\}$, $V_5 = \phi$, $W_5 = \phi$

2. $V_5 = \phi$, $F = F \cup \{\{3,4,6,7\}\}$

3^{2}. $r = 4$, $V_4 = \{7\}$, $W_4 = \phi$; $r = 3$, $V_3 = \{2,6\}$, $W_3 = \{7\}$

1^{2}. $x'_3 = 7$, $K_4 = \{3,4,7\}$, $V_4 = \{6\}$, $W_4 = \phi$

2-3 - 1-3. $V_4 = \phi$, $r = 3$, $V_3 = \{2,6,7\}$, $W_3 = \phi$; $V_3 \neq o$,
$r = 2$, $V_2 = \{2,4\}$, $W_2 = \{6,7\}$

1^{3}. $x'_2 = 6$, $K_3 = \{3,6\}$, $V_3 = \{4\}$, $W_3 = \{7\}$, $r = 3$;

$x'_3 = 7$, $K_4 = \{3,6,7\}$, $V_4 = \{4\}$, $W_4 = \phi$

2-3 - 1-3. $V_4 \neq \phi$, $r = 3$, $V_3 = \{4,7\}$, $W_3 = \phi$, $V_3 \neq \phi$,
$r = 2$, $V_2 = \{2,4,6\}$, $W_2 = \{7\}$

1^{2}. $x'_2 = 7$, $K_3 = \{3,7\}$, $V_3 = \{4,6\}$, $W_3 = \phi$

2-3 -1-3. $V_3 \neq \phi$, $r = 2$, $V_2 = \{2,4,6,7\}$, $W_2 = \phi$;
$V_2 = \phi$, $r = 1$, $V_1 = \{1,2,3\}$, $W_1 = \{4,5,6,7,8\}$

1^{4}. $x'_1 = 4$, $K_2 = \{4\}$, $V_2 = \{1,2,3\}$, $W_2 = \{5,6,7\}$,

$x'_2 = 5$, $K_3 = \{4,5\}$, $V_3 = \phi$, $W_3 = \{7\}$;

$x'_3 = 7$, $K_4 = \{4,5,7\}$, $V_4 = \phi$, $W_4 = \phi$.

---

* α^{β} denotes β operations of Step α (possibly with other checking steps in between).

2. $V_4 = \phi$, $F = F \cup \{\{4,5,7\}\}$

3^2. $r = 3$, $V_3 = \{7\}$, $W_3 = \phi$; $r = 2$, $V_2 = \{1,2,3,5\}$,
$W_2 = \{6,7\}$

1^3. $x_2' = 6$, $K_3 = \{4,6\}$, $V_2 = \{3\}$, $W_3 = \{7\}$, $r = 3$;
$x_3' = 7$, $K_4 = \{4,6,7\}$, $V_4 = \{3\}$, $W_4 = \phi$

2-3 - 1-3. $V_4 \neq \phi$, $r = 3$, $V_3 = \{3,7\}$, $W_3 = \phi$;
$V_3 \neq \phi$, $r = 2$, $V_2 = \{1,2,3,5,6\}$, $W_2 = \{7\}$

1^2. $x_2' = 7$, $K_3 = \{4,7\}$, $V_3 = \{3,5,6\}$, $W_3 = \phi$.

2-3 -1-3. $V_3 \neq \phi$, $r = 2$, $V_2 = \{1,2,3,5,6,7\}$, $W_2 = \phi$;
$V_2 \neq \phi$, $r = 1$, $V_1 = \{1,2,3,4\}$, $W_1 = \{5,6,7,8\}$

1^3. $x_1' = 5$, $K_2 = \{5\}$, $V_2 = \{4\}$, $W_2 = \{7\}$, $r = 2$;
$x_2' = 7$, $K_3 = \{5,7\}$, $V_3 = \{4\}$, $W_3 = \phi$

2-3 - 1-3. $V_3 \neq \phi$, $r = 2$, $V_2 = \{4,7\}$, $W_2 = \phi$;
$V_2 \neq \phi$, $r = 1$, $V_1 = \{1,2,3,4,5\}$, $W_1 = \{6,7,8\}$

1^4. $x_1' = 6$, $K_2 = \{6\}$, $V_2 = \{3,4\}$, $W_2 = \{7,8\}$, $r = 2$;
$x_2' = 7$, $K_3 = \{6,7\}$, $V_3 = \{3\}$, $W_3 = \{8\}$, $r = 3$;
$x_3' = 8$, $K_4 = \{6,7,8\}$, $V_4 = \phi$, $W_4 = \phi$

2. $V_4 = \phi$, $F = F \cup \{\{6,7,8\}\}$

3^2. $r = 3$, $V_3 = \{3,8\}$, $W_3 = \phi$; $r = 2$, $V_2 = \{3,4,7\}$, $W_2 = \{8\}$

$1^2_{\cdot}$ $x'_2 = 8$, $K_3 = \{6,8\}$, $V_3 = \{7\}$, $W_3 = \phi$

2-3 - 1-3. $V_3 \neq \phi$, $r = 2$, $V_2 = \{3,4,7,8\}$, $W_2 = \phi$; $r = 1$,
$V_1 = \{1,2,3,4,5,6\}$, $W_1 = \{7,8\}$

$1^3_{\cdot}$ $x'_1 = 7$, $K_2 = \{7\}$, $V_2 = \{3,4,5,6\}$, $W_2 = \{8\}$, $r = 2$;
$x'_2 = 8$, $K_3 = \{7,8\}$, $V_3 = \{6\}$, $W_3 = \phi$

2-3 1-3. $V_3 \neq \phi$, $r = 2$, $V_2 = \{3,4,5,6,7\}$, $W_2 = \phi$; $r = 1$,
$V_1 = \{1,2,3,4,5,6\}$, $W_1 = \{7,8\}$

$1^3_{\cdot}$ $x'_1 = 7$, $K_2 = \{7\}$, $V_2 = \{3,4,5,6\}$, $W_2 = \{8\}$, $r = 2$;
$x'_2 = 8$, $K_3 = \{7,8\}$, $V_3 = \{6\}$, $W_3 = \phi$

2-3 - 1-3. $V_3 \neq \phi$, $r = 2$, $V_2 = \{3,4,5,6,7\}$, $W_2 = \phi$; $r = 1$,
$V_1 = \{1,2,3,4,5,6,7\}$, $W_1 = \{8\}$

$1^2_{\cdot}$ $x'_1 = 8$, $K_2 = \{8\}$, $V_2 = \{1,6,7\}$, $W_2 = \phi$

2-3. $V_2 \neq o$; $r = 1$, $V_1 = \{1,2,3,4,5,6,7,8\}$, $W_1 = \phi$

1-2 - 3. $r = 0$

4. $F_c(G) = \{\{1,4\}, \{1,8\}, \{2,3,4\}, \{3,4,6,7\}, \{4,5,7\}, \{6,7,8\}\}$

A 2. The Algorithm of Section 1.2

Step 0. Set $K_1 = \phi$, $W_1 = X_1$, $V_1 = \phi$. Set $F = \phi$ put $r = 1$, and go to Step 2.

Step 1. Determine x'_r such that $|W_r - \Gamma x'_r| = \min_{x_k \in V_r} \{|W_r - \Gamma x_k|\}$

Step 2. Determine x''_r such that $|W_r - \Gamma x''_r| = \min_{x_k \in W_r} \{|W_r - \Gamma x_k|\}$

Step 3. If $|W_r - \Gamma x''_r| \leq |W_r - \Gamma x'_r|$, go to Step 5, otherwise go to Step 4.

Step 4. If $|W_r - \Gamma x'_r| = 0$ go to Step 9; otherwise find any $x''_r \in (W_r - \Gamma x'_r)$ and go to Step 5.

Step 5. Establish $K_{r+1} = K_r \cup \{x''_r\}$, $V_{r+1} = V_r \cap \Gamma x''_r$, $W_{r+1} = W_r \cap \Gamma x''_r$, set $r = r+1$ and go to Step 6.

Step 6. If $W_r = \phi$ go to Step 7; otherwise go to Step 1.

Step 7. If $V_r = \phi$ go to Step 8; otherwise go to Step 9.

Step 8. Update $F = F \cup \{K_r\}$ and go to Step 9.

Step 9. Set $r = r - 1$. If $r = o$ go to Step 10, otherwise update $V_r = V_r \cup \{x''_r\}$, $W_r = W_r - \{x''_r\}$, set $x'_r = x''_r$ and go to Step 4.

Step 10. Put $F_c(G) = F$ and STOP.

Example Again we demonstrate this algorithm by determining all the cliques of the Graph G of Figure 14.

Step 0. $K_1 = \phi$, $W_1 = \{1,2,3,4,5,6,7,8\}$, $F = \phi$, $r = 1$.

2. $x''_1 = 4$.

3-5. $K_2 = \{4\}$, $V_2 = \phi$, $W_2 = \{1,2,3,5,6,7\}$, $r = 2$.

6-1-2. $x_2'' = 3.$

3-5. $K_3 = \{3,4\}$, $V_3 = \phi$, $W_3 = \{2,6,7\}$, $r = 3.$

6-1-2. $x_3'' = 6.$

3-5. $K_4 = \{3,4,6\}$, $V_4 = \phi$, $W_4 = \{7\}$, $r = 4.$

6-1-5. $x_4'' = 7$, $K_5 = \{3,4,6,7\}$, $V_5 = \phi$, $W_5 = \phi$, $r = 5.$

6-7. $W_5 = \phi$, $V_5 = \phi$.

8. $F = \{\{3,4,6,7\}\}.$

9^2. $r = 4$, $V_4 = \{7\}$, $W_4 = \phi$, $x_4' = 7$; $r = 3$, $V_3 = \{6\}$, $W_3 = \{2,7\}$, $x_3' = 6.$

4-5. $x_3'' = 2$, $K_4 = \{2,3,4\}$, $V_4 = \phi$, $W_4 = \phi$, $r = 4.$

6-7-8. $W_4 = \phi$, $V_4 = \phi$, $F = F \cup \{\{2,3,4\}\}.$

9^2. $r = 3$, $V_3 = \{2,6\}$, $W_3 = \{7\}$, $x_3' = 2$; $r \neq 2$, $V_2 = \{3\}$, $W_2 = \{1,2,5,6,7\}$, $x_2' = 3.$

4-5. $x_2'' = 1$, $K_3 = \{1,4\}$, $V_3 = \phi$, $W_3 = \phi$, $r = 3.$

6-7. $W_3 = \phi$, $V_3 = \phi$.

8. $F = F \cup \{\{1,4\}\}$

9. $r = 2$, $V_2 = \{1,3\}$, $W_2 = \{2,5,6,7\}$, $x_2' = 1.$

4-5. $x_2'' = 2$, $K_3 = \{2,4\}$, $V_3 = \{3\}$, $W_3 = \phi$, $r = 3.$

6-7. $W_3 = \phi$, $V_3 \neq \phi$.

9. $r = 2$, $V_2 = \{1,2,3\}$, $W_2 = \{5,6,7\}$, $x_2' = 2.$

4-5. $x_2'' = 5$, $K_3 = \{4,5\}$, $V_3 = \phi$, $W_3 = \{7\}$, $r = 3.$

6-1-2. $x_3'' = 7.$

3-4-5. $K_4 = \{4,5,7\}$, $V_4 = \phi$, $W_4 = \phi$, $r = 4$.

6-7-1. $W_4 = \phi$, $V_4 = \phi$, $F = F \cup \{\{4,5,7\}\}$

9^2. $r = 3$, $V_3 = \{7\}$, $W_3 = \phi$, $x_3' = 7$; $r = 2$,

$V_2 = \{1,2,3,5\}$, $W_2 = \{6,7\}$, $x_2' = 5$.

4-5. $x_2'' = 6$, $K_3 = \{4,6\}$, $V_3 = \{3\}$, $W_3 = \{7\}$, $r = 3$.

6-1-2. $x_3' = 3$, $x_3'' = 7$.

3-4. $|W_3 - \Gamma x_3'| = \phi$.

9^2. $r = 2$, $V_2 = \{1,2,3,5,6\}$, $W_2 = \{7\}$, $x_2' = 6$; $r = 1$,

$V_1 = \{4\}$, $W_1 = \{1,2,3,5,6,7,8\}$, $x_1' = 4$.

4-5. $x_1'' = 8$; $K_2 = \{8\}$, $V_2 = \phi$, $W_2 = \{1,6,7\}$, $r = 2$.

6-1-2. $x_2'' = 1$.

3-4-5. $K_3 = \{1,8\}$, $V_3 = \phi$, $W_3 = \phi$, $r = 3$.

6-7-8. $W_3 = \phi$, $V_3 = \phi$, $F = F \cup \{\{1,8\}\}$.

9. $r = 2$, $V_2 = \{1\}$, $W_2 = \{6,7\}$, $x_2' = 1$.

4-5. $x_2'' = 6$, $K_3 = \{6,8\}$, $V_3 = \phi$, $W_3 = \{7\}$, $r = 3$.

6-1-2. $x_3'' = 7$.

3-4-5. $K_4 = \{6,7,8\}$, $V_4 = \phi$, $W_4 = \phi$, $r = 4$.

6-7-8. $W_4 = \phi$, $V_4 = \phi$, $F = F \cup \{\{6,7,8\}\}$

9^2. $r = 3$, $V_3 = \{7\}$, $W_3 = \phi$, $x_3' = 7$; $r = 2$,

$V_2 = \{1,6\}$, $W_2 = \{7\}$, $x_2' = 6$.

4. $|W_2 - \Gamma x_2'| = \phi$.

9. $r = 1$, $V_1 = \{4,8\}$, $W_1 = \{1,2,3,5,6,7\}$, $x_1' = 8$.

4-5. $x_r'' = 2$, $K_2 = \{2\}$, $V_2 = \{4\}$, $W_2 = \{3\}$, $r = 2$.

5-1-2. $x_2' = 4$, $x_2'' = 3$.

3-4. $|W_2 - \Gamma x_2'| = \phi$.

9. $r = 1$, $V_1 = \{2,4,8\}$, $W_1 = \{1,3,5,6,7\}$, $x_1' = 2$.

4-5. $x_2' = 1$, $K_2 = \{1\}$, $V_2 = \{4,8\}$, $W_2 = \phi$.

6-7-9. $W_2 = \phi$, $V_2 \neq \phi$; $r = 1$, $V_1 = \{1,2,4,8\}$, $W_1 = \{3,5,6,7\}$
$x_1' = 1$.

4-5. $x_2' = 3$, $K_2 = \{3\}$, $V_2 = \{2,4\}$, $W_2 = \{7\}$.

6-1-4. $|W_2 - \Gamma x_2'| = \phi$.

9. $r = 1$, $V_1 = \{1,2,3,4,8\}$, $W_1 = \{5,6,7\}$, $x_1' = 3$.

4-5. $x_1'' = 5$, $K_2 = \{5\}$, $V_2 = \{4\}$, $W_2 = \{7\}$, $r = 2$.

6-1-4. $|W_2 - \Gamma x_2'| = \phi$.

9. $r = 1$, $V_1 = \{1,2,3,4,5,8\}$, $W_1 = \{6,7\}$, $x_1' = 5$.

4. $|W_1 - \Gamma x_5| = \phi$.

9. $r = 0$.

10. $F_c(G) = \{\{3,4,6,7\}, \{2,3,4\}, \{1,4\}, \{4,5,7\}, \{1,8\}, \{6,7,8\}\}$

A 3. The Algorithm of Section 1·3

Step 0. Set $C_o = \{x_1, x_2, \ldots x_n\}$, $C_1 = C_o \cap (\Gamma x \cup \{x_1\})$,
$C_2 = C_o - \{x_1\}$, $F = \{C_1, C_2\}$

Step 1, For i = 2,3,...n :

1.1 Set $F_1 = \phi$, $F_2 = \phi$, $F_3 = \phi$ Put $\Gamma' x_i = \Gamma x_i \cup \{x_i\}$

1.2 For each $C_h \in F$:

1.2.1 If $x_i \notin C_h$, update $F_1 = F_1 \cup \{C_h\}$ and go to 1.2';
otherwise put $C = C_h \cap \Gamma' x_i$ and go to 1.2.2

1.2.2 If $\exists C'_k \in F_2$ s.t. $C \subseteq C'_k$ go to 1.2.3; otherwise
update $F_2 = F_2 \cup \{C\}$, and go to 1.2.3

1.2.3 Update $F_2 = F_3 \cup \{(C_h - \{x_i\})\}$, and go to 1.2'

1.2' Iterate

1.3 Delete from F_2 any set C'_h which is a subset of any other
set of F_2,
delete from F_3 any set C'_j which is a subset of any other
set of F_1,
and delete any set C'_k which is a subset of any other set of F_2,
and go to 1.4

1.4 Update $F = F_1 \cup F_2 \cup F_3$

Step 1'. Iterate

Step 2. Put $F_c(G) = F$, and STOP

Example: We again use the graph G of Figure 14 to demonstrate this algorithm.

Step 0. $C_o = \{1,2,3,4,5,6,7,8\}$, $C_1 = \{1,4,8\}$, $C_2 = \{2,3,4,5,6,7,8\}$

$F = \{\{1,4,8\}, \{2,3,4,5,6,7,8\}\}$, $|F| = 2$

1. $i = 2$

1.1 $F_1 = \phi$, $F_2 = \phi$; $\Gamma' x_2 = \{2,3,4\}$

1.2 $C_1 = \{1,4,8\}$

1.2.1 $F_1 = \{\{1,4,8\}\}$

1.2 $C_2 = \{2,3,4,5,6,7,8\}$

1.2.1 $C = \{2,3,4\}$

1.2.2 $F_2 = \{\{2,3,4\}\}$

1.2.3 $F_3 = \{\{3,4,5,6,7,8\}\}$

1.3 $F_1 = \{\{1,4,8\}\}$, $F_2 = \{\{2,3,4,5\}\}$, $F_3 = \{\{3,4,5,6,7,8\}\}$

1.4 $F = F_1 \cup F_2 \cup F_3$, $|F| = 3$

1. $i = 3$

1.1 $F_1 = \phi$, $F_2 = \phi$, $F_3 = \phi$; $\Gamma' x_3 = \{2,3,4,6,7\}$

1.2 $C_1 = \{1,4,8\}$

1.2.1 $F_1 = \{\{1,4,8\}\}$

1.2 $C_2 = \{2,3,4\}$

1.2.1 $C = \{2,3,4\}$

1.2.2 $F_2 = \{\{2,3,4\}\}$

1.2.3 $F_3 = \{\{2,4\}\}$

1.2 $C_3 = \{3,4,5,6,7,8\}$

1.2.1 $C = \{3,4,6,7\}$

1.2.2 $F_2 = F_2 \cup \{\{3,4,6,7\}\}$

1.2.3 $F_3 = F_3 \cup \{\{4,5,6,7,8\}\}$

1.3 $F_1 = \{\{1,4,8\}\}$, $F_2 = \{\{2,3,4\}, \{3,4,6,7\}\}$

$F_3 = \{\{4,5,6,7,8\}\}$.

1.4 $F = F_1 \cup F_2 \cup F_3$, $|F| = 4$

1. $i = 4$

1.1 $F_1 = \phi$, $F_2 = \phi$, $F_3 = \phi$, $\Gamma^1 x_4 = \{1,2,3,4,5,6,7\}$

1.2 $C_1 = \{1,4,8\}$

1.2.1 - 1.2.3 $F_2 = \{\{1,4\}\}$, $F_3 = \{\{1,8\}\}$

1.2 $C_2 = \{2,3,4\}$

1.2.1 - 1.2.3 $F_2 = F_2 \cup \{\{2,3,4\}\}$, $F_3 = F_3 \cup \{\{2,3\}\}$

1.2 $C_3 = \{3,4,6,7\}$

1.2.1 - 1,2,3 $F_2 = F_3 \cup \{\{3,4,6,7\}\}$,

$F_3 = F_3 \cup \{\{3,6,7\}\}$

1.2 $C_4 = \{4,5,6,7,8\}$

1.2.1 - 1.2.3 $F_2 = F_2 \cup \{\{4,5,6,7\}\}$

$F_3 = F_3 \cup \{\{5,6,7,8\}\}$

1.3 $F_1 = \phi$, $F_2 = \{\{1,4\}, \{2,3,4\}, \{3,4,6,7\}, \{4,5,6,7\}\}$

$F_3 = \{\{1,8\}, \{5,6,7,8\}\}$

1.4 $F = F_2 \cup F_3$, $|F| = 6$

1. $i = 5$

1.1 $F_1 = \phi, F_2 = \phi, F_3 = \phi, \Gamma'x_5 = \{4,5,7\}$

1.2 After the first three applications of 1.2, we have:

$F_1 = \{\{1,4\}, \{2,3,4\}, \{3,4,6,7\}\}$

1.2 $C_4 = \{4,5,6,7\}, F_2 = \{\{4,5,7\}\}, F_3 = \{\{4,6,7\}\}$

1.2 $C_5 = \{1,8\}, F_1 = F_1 \cup \{\{1,8\}\}$

1.2 $C_6 = \{5,6,7,8\}, F_2 = F_2 \cup \{\{5,7\}\},$

$F_3 = F_3 \cup \{\{6,7,8\}\}$

1.3 $F_1 = \{\{1,4\}, \{2,3,4\}, \{3,4,6,7\}, \{1,8\}\},$

$F_2 = \{\{4,5,7\}\}, F_3 = \{\{6,7,8\}\}$

1.4 $F = F_1 \cup F_2 \cup F_3, |F| = 6$

1. $i = 6$

1.1 $F_1 = \phi, F_2 = \phi, F_3 = \phi, \Gamma'x_6 = \{3,4,6,7,8\}$

1.2 $C_1 = \{1,4\}; F_1 = \{\{1,4\}\}$

1.2 $C_2 = \{2,3,4\}; F_1 = F_1 \cup \{\{2,3,4\}\}$

1.2 $C_3 = \{3,4,6,7\}; F_2 = \{\{3,4,6,7\}\}, F_3 = \{\{3,4,7,8\}\}$

1.2 $C_4 = \{1,8\}; F_1 = F_1 \cup \{\{1,8\}\}$

1.2 $C_5 = \{4,5,7\}; F_1 = F_1 \cup \{\{4,5,7\}\}$

1.2 $C_6 = \{6,7,8\}; F_2 = F_2 \cup \{\{6,7,8\}\}, F_3 = F_3 \cup \{\{7,8\}\}$

1.3 $F_1 = \{\{1,4\}, \{2,3,4\}, \{1,8\}, \{4,5,7\}\},$

$F_2 = \{\{3,4,6,7\}, \{6,7,8\}\}, F_3 = \{\{3,4,7,8\}\}$

1.4 $F = F_1 \cup F_2 \cup F_3, \quad |F| = 7$

1. i = 7

1.1 $F_1 = \phi$, $F_2 = \phi$, $F_3 = \phi$, $\Gamma' x_7 = \{3,4,5,6,7,8\}$

1.2 After the first three applications of 1.2 we have:

$F_1 = \{\{1,4\}, \{2,3,4\}, \{1,8\}\}$

1.2 After the remaining four applications of 1.2 we have:

$F_2 = \{\{4,5,7\}, \{3,4,6,7\}, \{6,7,8\}, \{3,4,7,8\}\}$;

1.3 $F_1 = F_1 \quad F_2 = F_2, F_3 = \phi$.

1.4 $F = F_1 \cup F_2$, $|F| = 7$

1. i = 8

1.1 $F_1 = \phi$, $F_2 = \phi$, $F_3 = \phi$, $\Gamma' x_8 = \{1,6,7,8\}$

1.2 After the first three applications of 1.2 we have:

$F_1 = \{\{1,4\}, \{2,3,4\}\}$, $F_2 = \{\{1,8\}\}$, $F_3 = \{\{1\}\}$

1.2 After the remaining four applications of 1.2 we have:

$F_1 = F_1 \cup \{\{4,5,7\}, \{3,4,6,7\}\}$, $F_2 = F_2 \cup \{\{6,7,8\}\}$,

$F_3 = F_3 \cup \{\{6,7\}, \{3,4,7\}\}$

1.3 $F_1 = \{\{1,4\}, \{2,3,4\}, \{4,5,7\}, \{3,4,6,7\}\}$

$F_2 = \{\{1,8\}, \{6,7,8\}\}$

$F_3 = \phi$

1.4 $F = F_1 \cup F_2$

2 $F_c(G) = \{\{1,4\}, \{2,3,4\}, \{4,5,7\}, \{3,4,6,7\}, \{1,8\}, \{6,7,8\}\}$

A 4. The Algorithm of Section 2 4

Step 0. Set $F = \phi$, and define, for each $x_i \in X$, $\Gamma' x_i = \Gamma x_i \cap \{x_{i+1}, \ldots x_n\}$. (Thus $\Gamma' x_n = \phi$); Set $j = n$, where $n = |X|$

Step 1. Set $j = j - 1$

Step 2. If $\Gamma' x_j = \phi$, go to Step 1; otherwise for each $x_k \in \Gamma' x_j$, update $F = F \cup \{\{x_j, x_k\}\}$

Step 3. For $i = n - 2, n - 3, \ldots 1$: set $F' = \phi$, suppose $F = \{Q_i, \ldots Q_s\}$

3.1 If $\Gamma' x_i = \phi$, go to Step 3'; otherwise put $W = \Gamma' x_i$, and go to 3.2

3.2 For $j = 1,2, \ldots s$:

3.2.1 If $|Q_j \cap \Gamma' x_i| < 2$, go to Step 3.2'; otherwise put $Q = Q_h \cap \Gamma' x_i$, update $W = W - Q \cap W$ and go to 3.2.2.

3.2.2 If $Q_h \subseteq \Gamma' x_j$ go to 3.2.7; otherwise go to 3.2.3.

3.2.3 If $\Gamma' x_j \subseteq Q_h$, update $F' = F' \cup \{Q \cup \{x_i\}\}$ and delete from F' any other set $Q' \in F'$ which is a subset of $(Q \cup \{x_i\})$ and go to Step 3.4; otherwise go to 3.2.4.

3.2.4 If $\exists Q_k \in F$ $(k \neq j)$ s.t. $Q \subseteq Q_k$ go to 3.2'; otherwise go to 3.2.5.

3.2.5 If $\exists Q'_\ell \in F'$ s.t. $Q \subseteq Q_\ell$ go to 3.2'; otherwise go to 3.2.6.

3.2.6 Update $F' = F' \cup \{(Q \cup \{x_i\})\}$, delete from F' any other set $Q' \in F'$ which is a subset of $(Q \cup \{x_i\})$, and go to 3.2'.

3.2.7 Replace Q_h by $(Q_h \cup \{x_i\})$ in F, delete from F any set $Q_k \in F$ $(k \neq h)$ which is a subset of the new Q_h, delete from F' any set $Q'_k \in F'$ which is a subset of the new Q_h, and go to 3.2.8.

3.2.8 If $Q = \Gamma' x_i$, update $F = F \cup F'$ and go to Step 3'; otherwise go to 3.2'.

3.2' Iterate.

3.2 For each $x_k \in W$, update $F' = F' \cup \{x_i, x_k\}$. Go to 3.4

Step 3' Iterate

Step 4. Put $F_c(G) = F$; and STOP.

Example We again demonstrate this algorithm with reference to the graph G depicted in Figure 14.

Step 0 ($n = 8$; $\Gamma'(1) = \{4,8\}$, $\Gamma'(2) = \{3,4\}$, $\Gamma'(3) = \{4,6,7\}$, $\Gamma'(4) = \{5,6,7\}$, $\Gamma'(5) = \{7\}$, $\Gamma'(6) = \{7,8\}$, $\Gamma'(7) = \{8\}$, $\Gamma'(8) = \phi$);

$F = \phi$, $j = 8$

1 $j = 7$

2 ($\Gamma'(7) = \{8\}$), $F = \{\{7,8\}\}$

3 $s = 6$

3.1 $\Gamma'(6) = \{7,8\}$, $W = \{7,8\}$, $F' = \phi$, $(s = 1)$

3.2 $h = 1$

3.2.1 $Q = \{7,8\}$, $W = \phi$

3.2.2 $\{7,8\} \subseteq \{7,8\}$

3.2.7 $F = \{\{6,7,8\}\}$

3.2.8 $\{7,8\} = \{7,8\}$

3. i = 5

3.1 ($\Gamma'(5) = \{7\}$), $W = \{7\}$, $F' = \phi$, (s = 1)

3.2 h = 1

3.2.1 $|\{6,7,8\} \cap \{7\}| < 2$

3.3 $F' = \{5,7\}$

3.4 $F = \{\{6,7,8\}, \{5,7\}\}$

3. i = 4

3.1 ($\Gamma'(4) = \{5,6,7\}$), $W = \{5,6,7\}$, $F' = \phi$ (s = 2)

3.2 h = 1

3.2.1 $Q = \{6,7\}$, $W = \{5\}$

3.2.2 -3.2.6 $F' = \{4,6,7\}$

3.2 h = 2

3.2.1 $Q = \{5,7\}$, $W = \{6\}$

3.2.2 $\{5,7\} \subseteq \{5,6,7\}$

3.2.7 $F = \{\{6,7,8\}, \{4,5,7\}\}$

3.2.8 $F = \{\{6,7,8\}, \{4,5,7\}, \{4,6,7\}\}$

3. i = 3

3.1 ($\Gamma'(3) = \{4,6,7\}$, $W = \{4,6,7\}$, $F' = \phi$, (s = 3)

3.2 h = 1

3.2.1 $Q = \{6,7\}$, $W = \{4\}$

3.2.4 $Q \subseteq Q_2 = \{4,6,7\}$

3.2 h = 2

3.2.1 $Q = \{4,6,7\}$, $W = \phi$

3.2.2 $\{4,6,7\} \subseteq \{4,6,7\}$

3.2.7 $F = \{\{6,7,8\}, \{4,5,7\}, \{3,4,6,7\}\}$

3.2.8 $\{4,6,7\} = \{4,6,7\}$

3. i = 2

3.1 ($\Gamma'(2) = \{3,4\}$), $W = \{3,4\}$, $F' = \phi$, (s = 3)

3.2 h = 1

3.2.1 $|\{3,4\} \cap \{6,7,8\}| < 2$

3.2 h = 2

3.2.1 $Q = \{3,4\}$, $W = \phi$

3.2.2 - 3.2.3 $\{3,4\} \subseteq \{3,4,6,7\}$. $F' = \{2,3,4\}$

3.4 $F = \{\{6,7,8\}, \{4,5,7\}, \{3,4,6,7\}, \{2,3,4\}\}$

3. i = 1

3.1 ($\Gamma'(1) = \{4,8\}$), $W = \{4,8\}$, $F' = \phi$, ($s = 4$)

3.2-3.2' Each of $\{6,7,8\}$, $\{4,5,7\}$, $\{3,4,6,7\}$, $\{2,3,4\}$ has but a single-element intersection with $\{4,8\}$

3.3 $F' = \{\{1,4\}, \{1,8\}\}$

3.4 $F' = \{\{6,7,8\}, \{4,5,7\}, \{3,4,6,7\}, \{2,3,4\}, \{1,4\}, \{1,8\}\}$

4. $F_c(G) = \{\{6,7,8\}, \{4,5,7\}, \{3,4,6,7\}, \{2,3,4\}, \{1,4\}, \{1,8\}\}$

CHAPTER IV

THEORETICAL ASPECTS OF GRAPH COLOURING

1. RESULTS FOR GENERAL GRAPHS

We mentioned in Chapter 2 that the GCP could be solved by direct tree search algorithms. In this chapter we introduce the main theoretical results upon which are based the graph-colouring algorithms which we describe in the two following chapters.

1.1 A Basic Theorem

The following theorem on the practical colouring of graphs was proved by Christofides [2], who also developed a direct colouring algorithm based thereon.

Theorem 1 Any graph $G = (X, \Gamma)$ can be optimally coloured by colouring with the first colour some maximal independent set, M_1, of G; then colouring with the second colour some maximal independent set, M_2, of G_1; and so on; untill all vertices have been coloured, where $G_k = (X_k, \Gamma)$ and $X_k = X - \bigcup_{j=1}^{k} M_j$ (1) and $G_o = G$.

Proof: Suppose $\{S_1^*, S_2^*, \ldots S_{\gamma(G)}^*\}$ represents the coloured sets of an optimal colouring of G in which not all S_i^* are maximal as above. Without loss of generality, we can assume S_1^* is not an MIS of G; let M be the MIS of G of which S_1^* is a subset. Since no vertex of $M-S_1^*$ is adjacent to any vertex of S_1^* (as M is an independent set), the vertices of $M-S_1^*$ can then be coloured colour 1 together with the vertices of S_1^*. Since also removing the

vertices of $M-S_1^*$ from the graph $G-S_1^*$ cannot increase the chromatic number of that graph, the same procedure can then be applied to the graph G-M, etc. so proving the theorem::

The above result has been used directly to construct a recursive algorithm for optimal graph-colouring, which we describe in the following chapter. Additionally, as we have seen in Chapter 2, it can also be used to construct a second colouring algorithm based on the Set Covering Problem. The major computational short-coming of these algorithms (both from the storage and the computing times points of view) is that it is not known a priori which set $M_1 \in F_m(G)$ should be coloured with 1, which set $M_2 \in F_m(G_1)$ should be coloured with 2 etc, and hence all possible choices of M_1, M_2 must eventually be considered by the algorithms either explicitly or implicitly. (We recall that $F_m(G)$ is the family of MIS of the graph (G)). Thus, any result which would enable one to further restrict the choices of $M_1, M_2 \ldots$ to sets which are members of families smaller than $F_m(G), F_m(G_1) \ldots$ etc. would have immediate effect on the computational efficiency of such graph-colouring algorithms.

In the next section we establish a theorem which makes such further restrictions possible and which results, as we shall demonstrate in the following chapter, in appreciable improvements to algorithmic efficiency when implemented. Again, as we shall see in a later section of this chapter, the theorem also leads to some interesting results in its own right.

1.2 The Intersection Theorem and Some Corollaries

Let us denote the family of cliques of G of maximum size (hereafter termed the maximum cliques of G) as:

$$\hat{F}_c(G) = \{ Q_i | Q_i \in F_c(G) , |Q_i| = \omega(G) \} \qquad (2)$$

Theorem 2 (The Intersection Theorem): If for a graph G, $\gamma(G) = \omega(G)$, then an optimum colouring of G can be obtained by first colouring with colour 1 an independent set of G, say S_1, such that:

$$|S_1 \cap Q_i| = 1 \quad \forall Q_i \in \hat{F}_c(G),$$

then colouring with colour 2 an independent set of the graph G, say S_2, such that:

$$|S_2 \cap Q_i| = 1 \quad \forall Q_i \in \hat{F}_c(G_1)$$

etc, until all the vertices of G are coloured; where $G_1, G_2, \ldots$ are the graphs given by expressions similar to (1) above.

Proof: Let us denote any optimum colouring of G by $\{S_1^*, S_2^*, \ldots, S_{\gamma(G)}^*\}$, where S_k^* is the set of vertices coloured with colour k.
Suppose that we start by colouring the vertices of S_1^*. Then if there exists a set $Q \in \hat{F}_c(G)$ such that $S_1^* \cap Q = \phi$, the total number of colours needed to colour the vertices of G would be $1 + \gamma(G_1) \geq 1 + \omega(G_1)$. However, since Q remains a clique (of size $\omega(G)$) in G_1, we have $\omega(G_1) = \omega(G)$ and hence we have that $\gamma(G) = 1 + \gamma(G_1) \geq 1 + \omega(G)$, which contradicts the assumption

that $\gamma(G) = \omega(G)$.

Hence,

$$|S_1^* \cap Q_i| \geq 1 \quad \forall Q_i \in \hat{F}_c(G) \tag{3}$$

However, since S_1^* is an independent set of G, it can have at most one vertex in common with any clique of G

$$\text{i.e.} \quad |S_1^* \cap Q_i| \leq 1 \quad \forall Q_i \in F_c(G) \tag{4}$$

From (3) and (4) we then have (since $\hat{F}_c(G) \subseteq F_c(G)$):

$$|S_1^* \cap Q_i| = 1 \quad \forall Q_i \in \hat{F}_c(G) \tag{5}$$

Therefore, $\omega(G_1) = \omega(G) - 1$ and since $\gamma(G_1) = \gamma(G) - 1$, we have that $\gamma(G_1) = \omega(G_1)$. The theorem then follows by induction, applying to G_1 the same argument as was done for graph G ::

Theorem 1 shows that an optimum colouring $\{M_1^*, \ldots, M_{\gamma(G)}^*\}$ of a graph G can be obtained where M_k^* is an MIS of graph G_{k-1} defined by expressions (1) above. Corollary 1 then follows immediately from Theorem 1.

Corollary 1: If for a graph G, $\gamma(G) = \omega(G)$, then an optimum colouring $\{M_1^*, \ldots, M_{\gamma(G)}^*\}$ can then be found which satisfies:

$$|M_k^* \cap Q_i| = 1 \quad \forall Q_i \in \hat{F}_c(G_{k-1}), \quad k = 1, \ldots, \gamma(G)$$

where $M_k^* \in F_m(G_{k-1})$. (We assume $G_o \equiv G$).

Corollary 2: If for a graph G, $\gamma(G) = \omega(G)$, then an optimum colouring $\{M_1^*, \ldots, M_{\gamma(G)}^*\}$ can be found which satisfies :

$$\left| M_k^* \cap Q_i \right| = 1 \quad \forall Q_i \in \hat{F}_c(G) \ , \quad k = 1, \ldots, \gamma(G).$$

Proof: Consider the the transition from G_{k-1} to G_k and let $Q_i \in \hat{F}_c(G_{k-1})$. From Corollary 1 we have

$$\left| M_k^* \cap Q_i \right| = 1 \ \forall Q_i \in \hat{F}_c(G_{k-1}) \text{ and hence :}$$

$$\omega(G_k) = \omega(G_{k-1}) - 1 \text{ , and}$$

$$\left| Q_i - (M_k^* \cap Q_i) \right| = \omega(G_k).$$

Therefore,

$$Q_i - (M_k^* \cap Q_i) \in \hat{F}_c(G_k) \qquad (6)$$

Let us now define $\hat{F}'_c(G_k) = \left\{ Q_i - \left(M_k^* \cap Q_i\right) \middle| Q_i \in \hat{F}_c(G_{k-1}) \right\}$

From expression (6) we can then write $\hat{F}'_c(G_k) \subseteq \hat{F}_c(G_k)$ (7)

From Corollary 1 we have:

$$\left| M_{k+1}^* \cap Q_j \right| = 1 \quad \forall Q_j \in \hat{F}_c(G_k)$$

$$\Rightarrow \left| M_{k+1}^* \cap Q_j \right| = 1 \quad \forall Q_j \in \hat{F}'_c(G_k) \quad \text{(from (7))}$$

$$\Rightarrow \left| M_{k+1}^* \cap Q_j \right| = 1 \quad \forall Q_j \in \hat{F}_c(G_{k-1})$$

$$\text{(since } M_k^* \cap M_{k+1}^* = \phi \text{)}$$

$$\Rightarrow \left| M_{k+1}^* \cap Q_j \right| = 1 \quad \forall Q_j \in \hat{F}'_c(G_{k-1}) \quad \text{(using (7))}$$

etc. until we finally get:

$$\left| M_{k+1}^* \cap Q_j \right| = 1 \quad \forall Q_j \in \hat{F}_c(G_o).$$

and as $G_o \equiv G$, the proof is completed ::

We shall consider some further consequences of these results in the next section, but first we prove one further result, for general graphs, which can be used to simplify the implementation of the algorithms - described in the following chapter - based on the Intersection Theorem.

1.3 Graph Colouring and Subset Condensation

Consider a graph $G_k = (X_k, \Gamma_k)$ and a MIS, $S_{k+1} \subseteq X_k$. By a condensation of S_{k+1}, we mean the replacement of all vertices in S_{k+1} by a single extra vertex y_{k+1}, the adjacencies of y_{k+1} being the totality of the adjacencies of all vertices in S_{k+1}; all other adjacencies of the graph G_k not involving S_{k+1} remaining unaffected. Thus the condensation of S_{k+1} produces the graph $G_{k+1} = (X_{k+1}, \Gamma_{k+1})$ where:

$$X_{k+1} = (X_k - S_{k+1}) \cup \{y_{k+1}\}$$

and, for every $x_i \in (X_k - S_{k+1})$:

$$\begin{aligned} &\text{if } \Gamma_k x_i \cap S_{k+1} = \phi, \text{ then } \Gamma_{k+1} x_i = \Gamma_k x_i \\ &\text{if } \Gamma_k x_i \cap S_{k+1} \neq \phi, \text{ then } \Gamma_{k+1} x_i = \Gamma_k x_i \cup \{y_{k+1}\} \\ &\text{and } \Gamma_{k+1} y_{k+1} = \bigcup_{v_j \in S_{k+1}} (\Gamma_k v_j - \Gamma_k v_j \cap S_{k+1}) \end{aligned} \quad (8)$$

An illustration of the process is provided by Figure 15.

We note here that the condensation vertex y_{k+1} corresponds to a pseudo-vertex (defined in the previous chapter) of a special type, viz. by referring to an MIS of the graph.

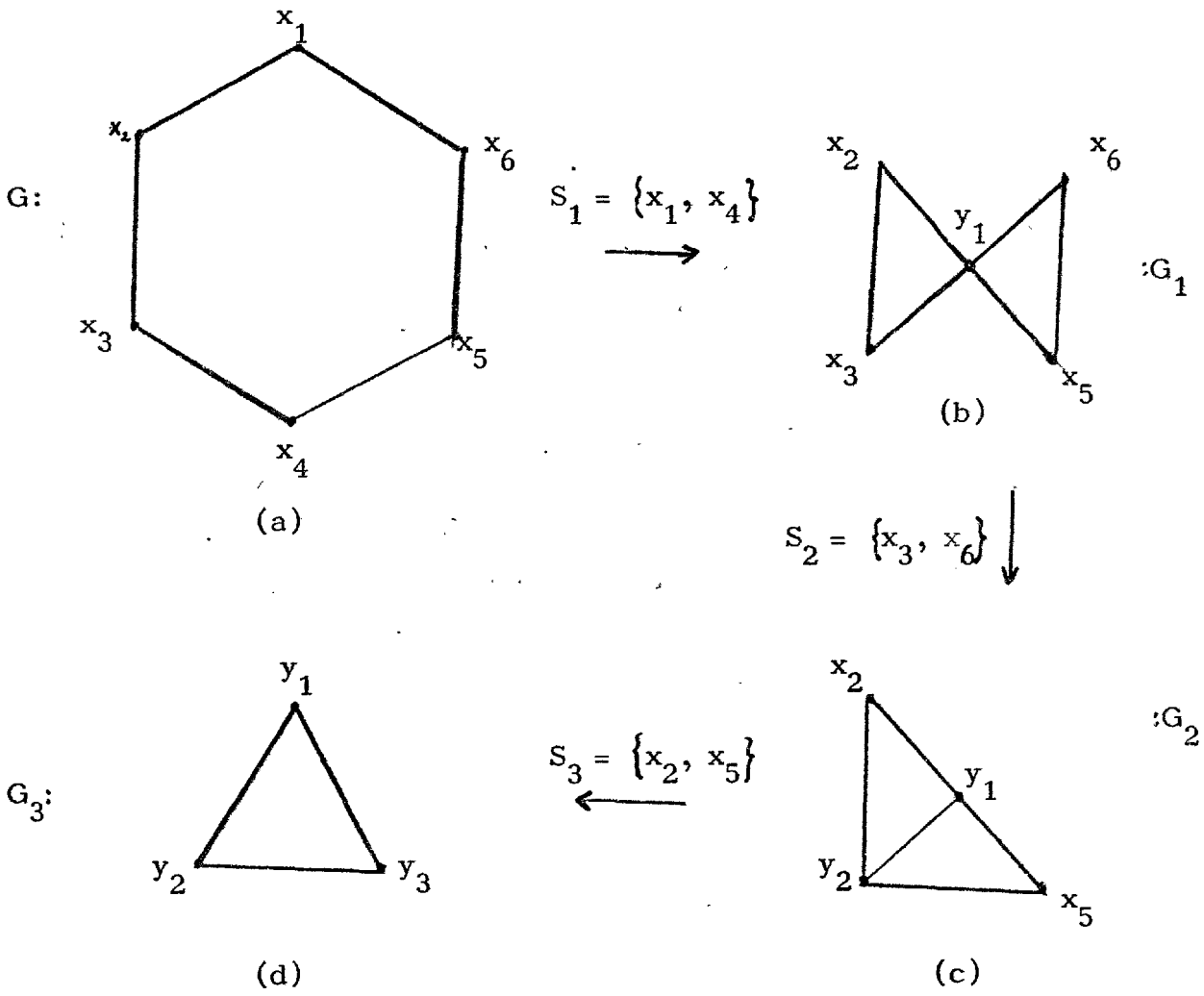

G:
x1
x2
x3
x4
x5
x6
(a)
S1 = {x1, x4}
x2
x6
y1
x3
x5
(b)
:G1
S2 = {x3, x6}
y1
y2
y3
G3:
(d)
S3 = {x2, x5}
x2
y1
y2
x5
(c)
:G2

FIGURE 15

Theorem 3 : Consider any arbitrary sequence of sets $\{S_1, S_2, \ldots S_p\}$ where S_k is an MIS of the graph G_{k-1} in the sequence of graphs $G_o, G_1, \ldots, G_p$ defined recursively by equations (8) and starting with $G_o = G$. This graph sequence then has the property that:

$$\omega(G_{k-1}) \leq \omega(G_k) \leq \omega(G_{k-1}) + 1 \quad \text{for } k = 1, \ldots, p.$$

Moreover, $\omega(G_k) = \omega(G_{k-1})$ if and only if

$$\left| S_k \cap Q_i \right| = 1, \quad \forall Q_i \in \hat{F}_c(G_{k-1})$$

Proof: Consider graph G_{k-1} in the sequence $G_o, G_1, \ldots, G_p$. Since S_k is an MIS of this graph, every vertex in $X_{k-1} - S_k$ is adjacent to at least one vertex of S_k, else S_k would not be maximal. Hence the condensation of S_k produces a vertex y_k in the graph G_k which is adjacent to every other vertex of G_k. From the independence property (equation 4) of S_k we also have:

$$\left| S_k \cap Q_i \right| \leq 1 \quad \forall Q_i \in F_c(G_{k-1}) \tag{9}$$

Thus, the removal of S_k from G_{k-1} to (partially) form G_k decreases the cardinality of all cliques of G_{k-1} by at most unity, whilst the addition of the condensation vertex y_k to form G_k increases the cardinality of all cliques of G_{k-1} by exactly unity. We therefore obtain the result

$$\omega(G_{k-1}) \leq \omega(G_k) \leq \omega(G_{k-1}) + 1 \tag{10}$$

where obviously $\omega(G_k) = \omega(G_{k-1})$ if and only if equality applies in expression (9) ::

Corollary 1: If $S_1, S_2, \ldots, S_q$ represent the coloured sets in an optimal colouring of G, so that $q = \gamma(G)$, then in the chain of inequalities:

$$\omega(G_0) \leq \omega(G_1) \leq \cdots \leq \omega(G_q) \quad (G_o = G) \qquad (11)$$

there are $\omega(G)$ equalities and $(\gamma(G) - \omega(G))$ strict inequalities.

Proof: Since the sets $S_1, S_2, \ldots S_q$ constitute a colouring of $G \equiv G_o$, it is apparent (cf Theorem 1) that the vertex set $X_{q-1} - S_q$ is empty. Therefore, by the way the graphs G_k were defined, G_q consists of q condensation vertices adjacent to each other, so that $\omega(G_q) = q$. Hence $\omega(G_q) = \gamma(G)$. Also, we have $\omega(G_o) = \omega(G)$ (since $G_o \equiv G$). Therefore, we have $\omega(G_q) - \omega(G_o) = \gamma(G) - \omega(G)$, so that the sequence of steps corresponding to the relations in (11) must span $\gamma(G) - \omega(G)$, and by Theorem 1, these steps can only be made in units of +1 or 0. Therefore there must be $(\gamma(G) - \omega(G))/1 = \gamma(G) - \omega(G)$ strict inequalities, and hence, since there are exactly $q = \gamma(G)$ relations, there must be $\gamma(G) - (\gamma(G) - \omega(G)) = \omega(G)$ equalities, so proving the result ::

A simple illustration of this result is provided by the sequence of graphs in Figure 15. In fact, Corollary 1 can be slighted strengthened

as below:

Corollary 2: Given the sequence (11) derived from any graph G and a set of MIS (of the appropriate G_k), $\{S_1, S_2, \ldots S_p\}$ corresponding to any colouring (optimal or not) of G, then the last relation in the sequence is an equality, i.e. $\omega(G_{p-1}) = \omega(G_p)$.

Proof: It can be seen from the definition above of any graph G_k in the sequence $G_o, G_1, \ldots,$ that G_k consists of the induced subgraph $G - \bigcup_{i=1}^{k} S_i$ augmented by a clique (of condensation vertices) of cardinality k, each condensed vertex being adjacent to each vertex of $G - \bigcup_{i=1}^{k} S_i$ (else the corresponding S_j $(1 \leqslant j \leqslant k)$ would not have been maximal). In particular, G_{p-1} consists of an independent set (viz S_p), each vertex of which is adjacent to a clique of cardinality (p-1). Then $\omega(G_p) = (p-1) + 1$, and S_p intersects each maximum clique of G_p. Hence by the last statement of Theorem 3, $\omega(G_p) = \omega(G_{p-1})$ ::

Corollary 3: For any graph $G = (X, \Gamma)$, the generation of the sequence of the MIS $S_1, S_2, \ldots S_p$ from the graphs $G_o, G_1, \ldots G_p$, - given by (8) - terminates once $\omega(G)$ equalities have been obtained in the series

$$\omega(G_o) \leqslant \omega(G_1) \leqslant \cdots \leqslant \omega(G_p)$$

since the union of all the corresponding S_i up to that stage will equal X.

Proof: Suppose the $\omega(G)^{th}$ equality is $\omega(G_{h-1}) = \omega(G_h)$.

Then, since, by Cor 1, only $\omega(G)$ equalities are to be obtained in the whole of the series, any relations in the series

$\omega(G_h) \leq \omega(G_{h+1}) \leq \cdots \leq \omega(G_p)$ must all be strict inequalities. But by Cor 2, the final relation must be an equality. Therefore $\omega(G_{h-1}) = \omega(G_h)$ must be the last relation in the series ::

Corresponding to Theorem 3, we can also prove the following result relating the $\gamma(G_k)$ to one another.

Theorem 4 : In the same circumstances as Theorem 3, the sequence of graphs $G_o, G_1, \ldots G_p$ has the property that:

$$\gamma(G_{k-1}) \leq \gamma(G_k) \leq \gamma(G_{k-1}) + 1 .$$

Moreover $\gamma(G_k) = \gamma(G_{k-1})$ if and only if S_k is a member of an optimal colouring of G_{k-1}.

Proof: Suppose $\gamma(G_k) > \gamma(G_{k-1}) + 1$. Removing condensation vertex y_k from G_k leaves a graph which requires at least $\gamma(G_{k-1}) + 1$ colours. But G_{k-1} is a graph which contains $G_k - \{y_k\}$ as a subgraph so that $\gamma(G_{k-1}) \geq \gamma(G_{k-1}) + 1$, a contradiction.

On the other hand, the colour assigned to y_k in G_k can be used to colour S_k in G_{k-1} (as S_k is an independent set). Therefore $\gamma(G_{k-1}) \leq \gamma(G_k)$ so completing the proof ::

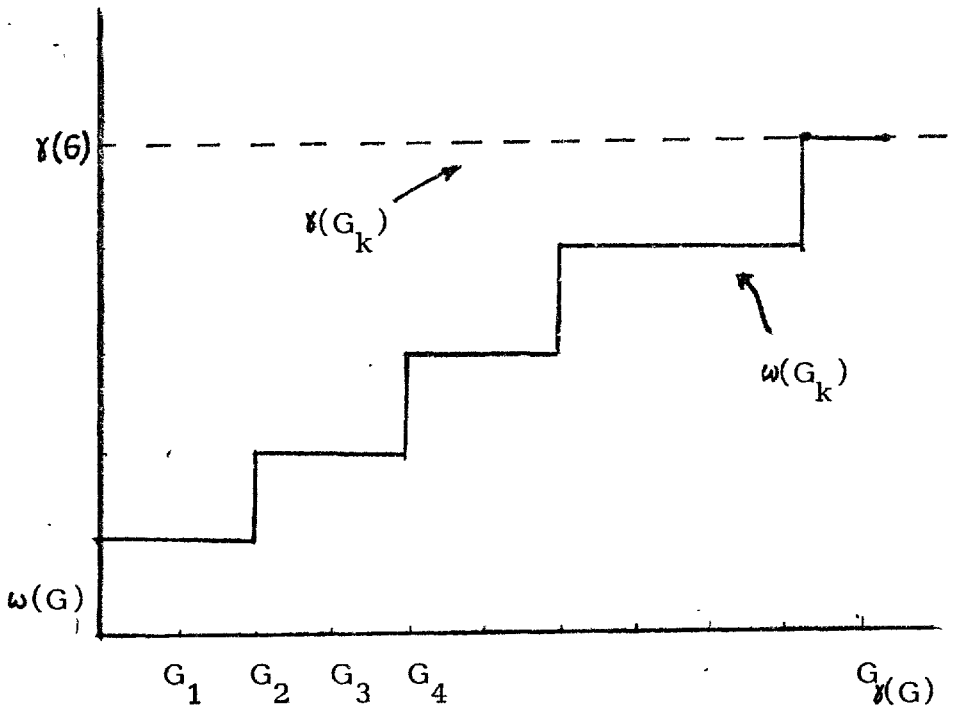


FIGURE 16

Theorem 4 implies that if $S_1, S_2, \ldots, S_q$ represent the coloured sets of an optimal colouring of G, then $\gamma(G_o) = \gamma(G_1) = \cdots = \gamma(G_q)$. Recalling that $G_o \equiv G$ (so that $\gamma(G_o) = \gamma(G)$), and using Theorem 3 and its corollaries we can then deduce the interesting profile of the graphs, G_k shown in Figure 16, which may possibly lead to a tightening of the bound: $\gamma(G) \geq \omega(G)$, mentioned in Chapter 1.

2. PERFECT GRAPHS AND P-CRITICAL GRAPHS

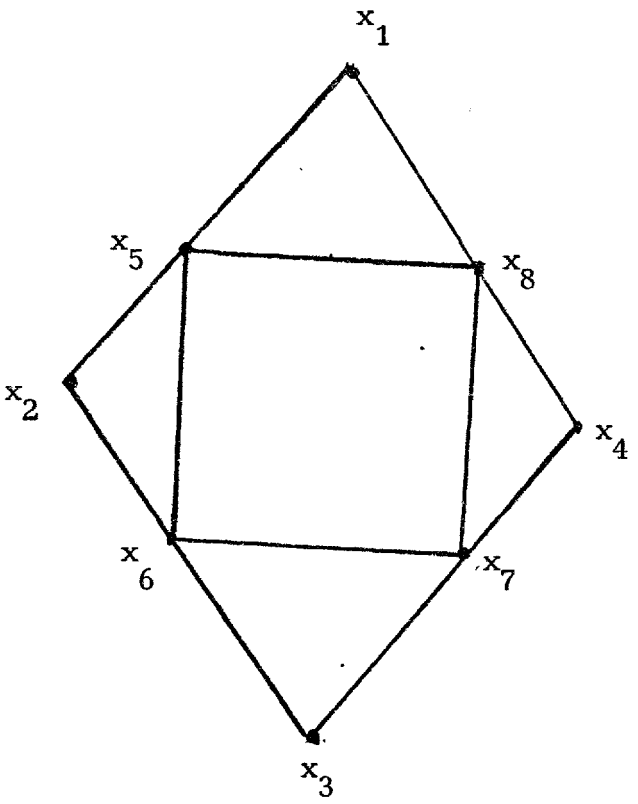
2.1 Introduction to Perfect Graphs

We saw in Chapter 1 that, for any graph G, $\gamma(G) \geq \omega(G)$, and so it is natural to study those graphs G for which $\gamma(G) = \omega(G)$. The main result to date on such graphs is Konig's theorem [4] (see Chapter 1) giving a complete characterization for the case where $\omega(G) = 2$, so Berge introduced the concept of perfect graphs as a subclass of the graphs with $\gamma(G) = \omega(G)$. [1]

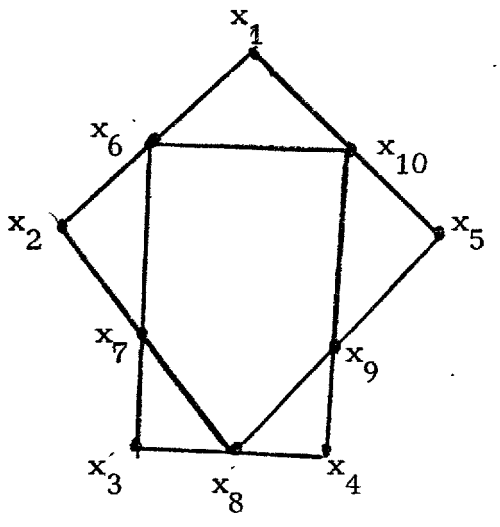
Thus, a perfect graph is defined as a graph G for which $\gamma(G') = \omega(G')$ for any induced subgraph G', of G. (In particular of course $\gamma(G) = \omega(G)$).

For example, the graph in Figure 17a is perfect while that of Figure 17b is not perfect, since its subgraph defined by the vertices $\{x_6, x_7, \ldots x_{10}\}$ together with all arcs joining these vertices has $\gamma(G) = 3$; $\omega(G) = 2$.

The main result relating to perfect graphs is the following:



(a)



(b)

FIGURE 17

Perfect Graph Theorem [5]: G is perfect if and only if $\bar{G}$ is perfect. The following conjecture, again due to Berge [1], is still unproven but which, if true, would characterize perfect graphs:

Strong Perfect Graph Conjecture: A graph G is perfect if and only if it does not contain an induced subgraph which either is a cycle (without chords)* of odd length, or the complement of such a cycle.

However, the following weaker result has been proved by Lovasz [6]:

A graph $G = (X, \Gamma)$ is perfect if and only if $\omega(G') \cdot \omega(\bar{G}') \geq |X|$ for every induced subgraph, G', of G.

Finally, let us define a p-critical graph as a graph which itself is not perfect but for which every proper induced subgraph is perfect. Thus any graph G is not perfect if and only if G contains an (induced) p-critical subgraph.

Now let G be any p-critical graph containing n vertices and let x_i be any vertex in G.

Then the following results can be easily shown (e.g. [7])

1. G is connected and contains at least 5 vertices
2. $\bar{G}$ is p-critical.
3. $\omega(G) \cdot \omega(\bar{G}) = n - 1$
4. $2 \leq \omega(G) \leq \lceil n-1 / 2 \rceil$, $2 \leq \omega(\bar{G}) \leq \lceil n-1 / 2 \rceil$
5. No independent set of G intersects all maximum cliques of G.
6. Every vertex of G is contained in at least one maximum clique.

In the next section we use Theorem 2 to derive a strengthening of this last result.

---

* A chord is an arc joining two non-consecutive vertices of a cycle.

2.2 Some Consequences of the Intersection Theorem

Theorem 5: If $G = (X, \Gamma)$ is p-critical then each vertex $x_i \in X$ is contained in at least $\omega(G)$ maximum cliques.

Proof: Consider the perfect graph $G' = G - \{x_i\}$, where x_i is any vertex of G. Since G is a p-critical graph we obviously have $\gamma(G) = \omega(G) + 1$ and $\omega(G') = \omega(G)$. Since $\gamma(G') = \omega(G')$, we can find $\omega(G') = \omega(G)$ independent sets of G', each of which intersects all maximum cliques of G' (by Corollary 2). Call these sets $S_1, S_2, \ldots, S_{\omega(G)}$.

Now, none of these independent sets can intersect all the maximum cliques of G, since the removal of such a set (say S_k) would produce $G - S_k$ with $\gamma(G - S_k) \geq \gamma(G) - 1$ and $\omega(G - S_k) = \omega(G) - 1$, which, (in view of the fact that $\gamma(G) = \omega(G) + 1$), contradicts that $G - S_k$ be perfect.

Therefore, there must be at least one maximum clique of G which contains x_i and which does not include any element of S_k.

Hence, we have that x_i must belong:

to at least one maximum clique (say Q_1) which does not contain any element of S_1,

to at least one maximum clique (say Q_2) which does not contain any element of S_2,

etc.

But we also know that Q_ℓ can contain at most one vertex from any one S_k, $k = 1, 2, \ldots \omega(G)$, i.e. Q_ℓ can contain at most one vertex from each of the sets $S_1, S_2, \ldots, S_{\ell-1}, S_{\ell+1}, \ldots S_{\omega(G)}$,

so that the only possible way to form the maximum clique Q_ℓ is by taking exactly one vertex from each S_k, $k \neq \ell$, and adding x_i. This shows that the Q_ℓ, $1 \leq \ell \leq \omega(G)$, are distinct, since for any $j \neq h$, Q_j contains a vertex of S_h while Q_h does not.

Hence, x_i belongs to at least $\omega(G)$ maximum cliques, and since x_i is any vertex, the theorem follows.

Corollary 1: Any p-critical graph $G = (X, \Gamma)$ contains at least n maximum cliques, where $n = |X|$.

Proof: Let q represent the number of maximum cliques of G. By Theorem 5, every vertex $x_i \in X$ belongs to $(\omega(G) + a_i)$ maximum cliques, where $a_i \geq 0$.

Therefore, the total number of vertices contained in the maximum cliques of G, including replications of vertices, is :

$$\sum_{i=1}^{n} (\omega(G) + a_i) = n.\omega(G) + \sum_{i=1}^{n} a_i$$

But as each maximum clique of G contains, by definition, $\omega(G)$ vertices, and as there are q such maximum cliques, we must have:

$$q.\omega(G) = n.\omega(G) + \sum_{i=1}^{n} a_i$$

i.e. $$q = n + \sum a_i / \omega(G) ,$$

But $\sum_{i=1}^{n} a_i \geq 0$, $\omega(G) \geq 0$, so that $q \geq n$. Hence the corollary ::

Corollary 2: Any p-critical graph $G = (X, \Gamma)$ contains at least n maximum independent sets, where $n = |X|$.

Proof: By the Perfect Graph Theorem, we have that a graph $G' = (X', \Gamma')$ is perfect if and only if its complementary graph $\bar{G}' = (X', \bar{\Gamma}')$ is perfect; so that it is easily seen that $G = (X, \Gamma)$ is p-critical if and only if $\bar{G} = (X, \bar{\Gamma})$ is p-critical.

Therefore, from Corollary 1, $\bar{G}$ contains at least n maximum cliques. But as any clique of $\bar{G}$ corresponds exactly to an independent set of G, the result follows immediately ::

3. COLOURING PERFECT GRAPHS

It might be supposed that, although graphs for which $\gamma(G) = \omega(G)$ can not be easily characterized, colouring such graphs would be a straightforward matter. Consider however, the graph G in Figure 17(b)Although $\gamma(G) = \omega(G) = 3$ (and moreover each MIS of G intersects all maximum cliques of G) choosing to colour the the (unique) largest MIS (viz $\{x_1, x_2, x_3, x_4, x_5\}$) with the first colour will not lead to an optimum colouring, hence backtracking becomes necessary; so that no simple (one-pass) algorithm for such graphs is apparent.

However, in [3] , a simple algorithm for the optimal colouring of a 'chordal' graph -- a graph for which every cycle of length greater than 3 contains a chord - is given; and it is noted there (see also [1]) that all chordal graphs are perfect.

In fact, with the aid of Theorem 2, it is possible to develop a one-pass colouring algorithm for the whole of the class of perfect graphs, and before going on to consider the general graph colouring algorithms in the remaining chapters, we conclude this chapter by establishing this perfect-graph colouring algorithm.

3.1 A One-Pass Perfect-Graph Colouring Algorithm

Theorem 6 : If G is a perfect graph, then every MIS of G which intersects all maximum cliques of G is in some optimum colouring of G.

Proof: Suppose that the statement is false, and let M be any MIS of G which satisfies the intersection condition but which is not in any optimum colouring of G. Then it is seen that

$$\gamma(G - M) = \gamma(G) \tag{12}$$

For if $\gamma(G-M) = \gamma(G) - 1$, then optimally colouring the graph $G' = G - M$ with $\gamma(G) - 1$ colours and assigning colour $\gamma(G)$ to the vertices of M will result in an optimum colouring of G involving M, which is a contradicition. Also, since M intersects all the maximum cliques of G, we have

$$\omega(G - M) = \omega(G) - 1 \tag{13}$$

Moreover, since G is perfect, we have

$$\gamma(G) = \omega(G) \tag{14}$$

so that combining (12) and (13) and using (14) we obtain

$$\gamma(G - M) > \omega(G - M) \tag{15}$$

But G-M is a non empty subgraph of G so that (15) contradicts the definition of G as a perfect graph. Hence the theorem ::

In view of the fact that any subgraph of a perfect graph is itself perfect, Theorem 6 leads directly to the following simple one-pass algorithm for determining an optimal colouring of a perfect graph G.

Step 0. Put $G' = G$

Step 1. Determine the family $\hat{F}_c(G')$ of maximum cliques of G', and find an MIS of G', say M', which intersects each set in $\hat{F}_c(G')$

Step 2. Put $G' = G' - M'$. If G' is the empty graph, go to Step 3 otherwise go to Step 1.

Step 3. Stop with the sets M' selected at Step 1 of each stage as the optimal vertex colouring.

The termination of the algorithm is ensured by Corollary 1, of Theorem 2.
(Indeed, as the proof of Theorem 6 nowhere makes use of the maximality of the MIS M, by Theorem 2 an equivalent algorithm can be formulated using independent sets in place of MIS's).

3.2 Illustration of the Algorithm

Finally we illustrate the above algorithm with reference to the perfect graph G in Figure 1(a).

Step 0. $G' = G$

Step 1. $\hat{F}_c(G') = \{\{1,5,8\}, \{2,5,6\}, \{3,6,7\}, \{4,7,8\}\}$.
$\{1,2,3,4\} \in F_m(G')$ and $\{1,2,3,4\} \cap Q_i \neq \phi$
$\forall Q_i \in \hat{F}_c(G')$; $M' = \{1,2,3,4\}$

Step 2. $G' = G' - \{1,2,3,4\}$; $G' \neq \phi$

Step 1. $\hat{F}_c(G') = \{\{5,6\}, \{5,8\}, \{6,7\}, \{7,8\}\}$;

$\{5,7\} \in F_m(G')$ and $\{5,7\} \cap Q_i \neq \phi \quad \forall Q_i \in \hat{F}_c(G')$;

$M' = \{5,7\}$

Step 3. $G' = G' - \{5,7\}$; $G' \neq \phi$

Step 1. $\hat{F}_c(G') = \{\{6\}, \{8\}\}$;

$\{6,8\} \in F_m(G')$ and $\{6,8\} \cap Q_i \neq \phi \quad \forall Q_i \in \hat{F}_c(G')$;

$M' = \{6,8\}$

Step 2. $G' = G' - \{6,8\}$; $G' = \phi$

Step 3. STOP with the optimal colouring $\{1,2,3,4\}, \{5,7\}, \{6,8\}$.

4. REFERENCES

1. BERGE, C. Graphs and Hypergraphs, North Holland Publishing Compnay, Amsterdam (1973).

2. CHRISTOFIDES, N. "An Algorithm for the Chromatic Number of a Graph", Computer Journal 14 (1971), pp 38-39.

3. GAVRIL, F. "Algorithms for Minimum Colouring, Maximum Clique, Minimum Covering by Cliques and Maximum Independent Set of a Chordal Graph", SIAM Journal of Computing 1 (1972), pp 180-187.

4. KONIG, D. Theorie der Endlichen un Urendlichen Graphen, Chelsea, N.Y. (1950).

5. LOVASZ, L. "Normal Hypergraphs and the Perfect Graph Conjecture", Discrete Mathematics 2 (1972), pp 253-268.

6. LOVASZ, L. "A Characterization of Perfect Graphs", J. Combinatorial Theory (B) 13 (1972), pp 95-98.

7. PADBERG, M.W. "Perfect Zero-One Matrices", Mathematical Programming 6 (1974), pp 180-196.

CHAPTER V

COLOUR-BY-COLOUR GRAPH COLOURING

1. INTRODUCTION

We have seen in Chapter 1 that the Graph Colouring Problem (GCP) can be formulated, and so solved, as a more general Set Covering Probelm (SCP) by using the maximal independent sets (MIS) of the given graph as the covering sets. However, Theorem 4-1, on which the SCP approach is based, can in fact be used to establish a tree-search method to solve the GCP directly, again by using the MIS of the graph. This method involves, at any stage of the search, the simultaneous colouring of all the vertices for which it is proposed to assign the particular colour at hand. Thus one may equivalently treat this approach as the assignment of sets of vertices to the colours - in sequential order of colours -.

In this chapter we shall trace the development of such an algorithm, from its inception as a simple breadth-first tree-search, through a sequence of modifications which lead to a depth-first tree-search method which is related to one of the SCP algorithms given in Chapter 2. Finally, we describe two variants of an original GCP algorithm which makes use of Theorem 4-2.

In order to somewhat simplify the notation in the raminder of this chapter, we make some modifications to the notation used previously. Thus the family of MIS of a graph G will henceforth be denoted by $\mathcal{M}(G)$ (instead of $F_m(G)$) while the family of cliques of G will be denoted by $\mathcal{Q}(G)$ (instead of $F_c(G)$); the families of maximum MIS and maximum cliques then being $\hat{\mathcal{M}}(G)$ and $\hat{\mathcal{Q}}(G)$ respectively.

2. A COMPLETE DEPTH-FIRST APPROACH

2.1 Preliminaries

One of the earliest methods for using the above colour-by-colour strategy for graph-colouring is given by Christofides [1], who describes a dynamic programming (or complete breadth-first tree-search) algorithm for optimal graph colouring.

This method itself can be more easily described by first defining a maximal r-colourable subset, S_r, of a graph G as any maximal subset of vertices of G which requires no more than r colours for a feasible colouring. Thus an independent set of G is 1-colourable, and only independent sets are 1-colourable while $\gamma(G)$ is obviously the last integer r for which X, the vertex set of G, is r-colourable; the family of MIS of G represent the maximal 1-colourable subsets of G, while it is easily seen that X itself is the unique maximal γ-colourable subset, a fact which shall be used later. Theorem 4-1 then implies, [1], that only maximal r-colourable subsets need be considered for potential colouring. Furthermore, as can be easily proved by contradiction, the family, $\mathcal{S}_r$, of maximal r-colourable subsets, S_r^j, of G can be determined from the family $\mathcal{S}_{r-1}$ by the recurrence relation, (where $G'_{r-1} = G - S^i_{r-1}$):

$$S_r^j = S^i_{r-1} \cup S_1^k (G'_{r-1}) \text{ for some } i^{th} \text{ set } S^i_{r-1} \in \mathcal{S}_{r-1} \text{ and } k^{th} \text{ set } S_1^k \in \mathcal{S}_1(G'_{r-1}) \quad (1)$$

Recalling that $\mathcal{S}_1 = \mathcal{M}(G)$, the family of MIS of G, the algorithm can now be basically described as follows:

After first determining $\mathcal{S}_1 = \mathcal{M}(G)$, by using one of the MIS-determination algorithms described in Chapter 3, for example, one then calculates $\mathcal{S}'_2$, the family of all sets S_2^j obtained from relation (1)

by using all possible values of i and k, and then derives the family $\mathcal{S}_2$ by removing from $\mathcal{S}_2'$ any sets which are subsets of other sets of $\mathcal{S}_2'$, (so that $\mathcal{S}_2 = \mu\mathcal{S}_2'$). One then forms $\mathcal{S}_3$ from $\mathcal{S}_2$, $\mathcal{S}_4$ from $\mathcal{S}_3$, etc. until a stage is reached where, for some value of r, $X \in \mathcal{S}_r'$, at which time the algorithm terminates with $\gamma(G) = r$ with a corresponding associated colouring.

We now describe the algorithm more formally.

2.2 Description of the Algorithm

Step 1. Find $\mathcal{M}(G) = \{M_1, M_2, \ldots, M_{t_1}\}$, the family of MIS of the graph $G = (X, \Gamma)$. For each $j = 1,2,\ldots,t_1$ put $S_1^j = M_j$, and denote by $\mathcal{S}_1$ the family of sets S_1^j.

Put $r = 0$.

Step 2. Put $r = r + 1$, $i = 0$, Set $\mathcal{S}_{r+1} = \phi$.

Step 3. For $j = 1,2,\ldots,t_r$:

3.1 Find $\mathcal{M}(G-S_r^j) = \{M_1, M_2, \ldots, M_p\}$, the family of MIS of the graph $G = (X - S_r^j, \Gamma)$.

3.2 For $k = 1,2,\ldots,p$:

3.2.1 Calculate $T = S_r^j \cup M_k$. If $T = X$ go to Step 5, otherwise go to 3.2.2.

3.2.2 For $h = 1,2,\ldots,i$:

If $T \subseteq S_{r+1}^h$ go to Step 3.2'; if $T \supset S_{r+1}^h$ replace S_{r+1}^h by T and go to 3.2';

otherwise go to 3.2.2.

3.2.2. Put $S_{r+1}^{i+1} = T$, $i = i + 1$.

3.2'. Iterate.

Step 4. Remove from $\mathcal{S}_{r+1} = \{S_{r+1}^1, \ldots, S_{r+1}^i\}$ any set S_{r+1}^k for which $\exists\, h < k$ s.t. $S_{r+1}^k = S_{r+1}^h$, renumber the indices of the remaining sets to run consecutively from 1 to t_{r+1}, say, and replace $\mathcal{S}_{r+1}$ by the resultant family of sets. Go to Step 2.

Step 5. Stop with $(r + 1)$ as the chromatic number of G, with an optimal colouring being given by the subsets M_k which were introduced at the various stages of the building up of the current set T.

2.3 Demonstration of the Algorithm

We shall demonstrate the workings of the above algorithm, and indeed of the other algorithms in this chapter, by finding an optimal colouring for the graph G, taken from [3], illustrated in Figure 18.

Step 1. $\mathcal{S}_1 = \{\{1,3\}, \{1,4,7\}, \{2,4,7\}, \{2,5,7\}, \{2,5,8\}, \{3,5\}, \{3,6\}, \{6,8\}\}$, $(t = 8)$; $r = 0$.

2. $r = 1$, $i = 0$, $\mathcal{S}_2 = \phi$, $t_1 = 8$.

3. $j = 1$, $S_1^1 = \{1,3\}$

3.1 $\mathcal{M}(G - \{1,3\}) = \{\{2,4,7\}, \{2,5,7\}, \{2,5,8\}, \{6,8\}\}$ $(p = 4)$

3.2 - 3.2' $\mathcal{S}_2 = \{\{1,3;2,4,7\}, \{1,3;2,5,7\}, \{1,3;3,5,8\}, \{1,3;6,8\}\}$, $i = 4$

3. $j = 2$, $S_1^2 = \{1,4,7\}$

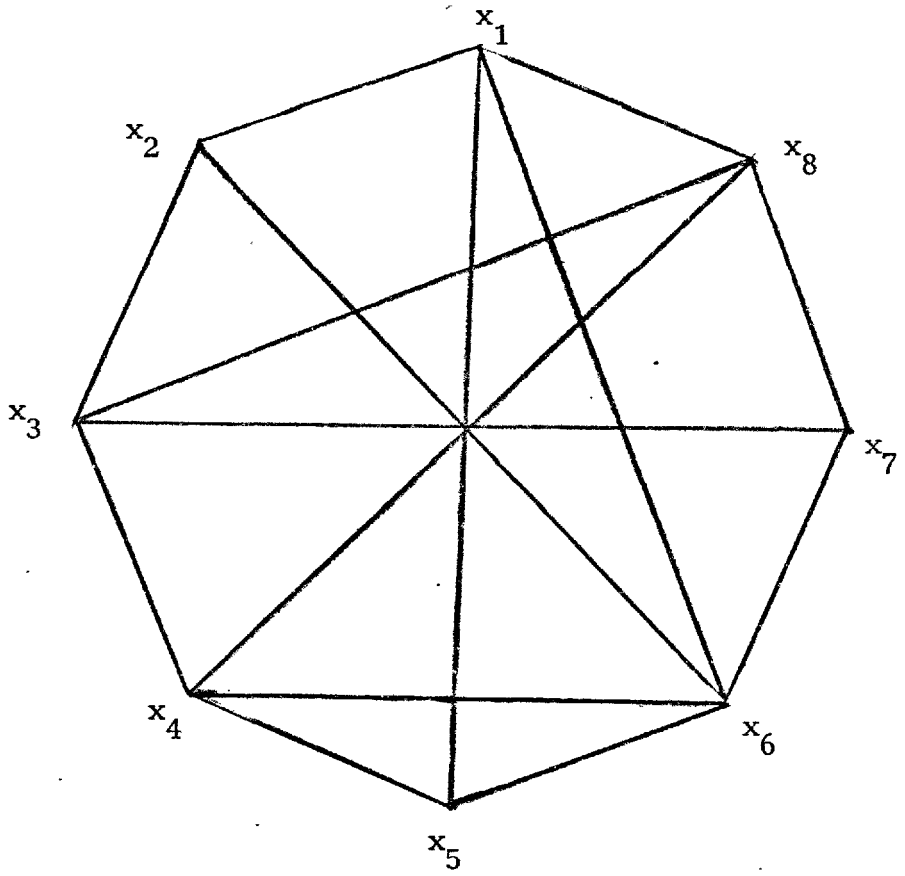


FIGURE 18

3.1 $\mathcal{M}(G - \{1,4,7\}) = \{\{2,5,8\}, \{3,5\}, \{3,6\}, \{6,8\}\}$, $(p = 4)$.

3.2 - 3.2' $\mathcal{S}_2 = \mathcal{S}_2 \cup \{\{1,4,7;2,5,8\}, \{1,8,7;3,5\}, \{1,4,7;3,6\}, \{1,4,7;6,8\}\}$, $i = 8$

3. $j = 2$, $S_1^2 = \{1,4,7\}$

3.1 $\mathcal{M}(G - \{1,4,7\}) = \{\{2,5,8\}, \{3,5\}, \{3,6\}, \{6,8\}\}$, $(p = 4)$

3.2 - 3.2' $\mathcal{S}_2 = \mathcal{S}_2 \cup \{\{1,4,7;2,5,8\}, \{1,4,7;3,5\}, \{1,4,7;3,6\}, \{1,4,7;6,8\}\}$, $i = 8$

3. $j = 3$, $S_1^3 = \{2,4,7\}$

3.1 $\mathcal{M}(G - \{2,4,7\}) = \{\{1,3\}, \{3,5\}, \{5,8\}, \{6,8\}\}$, $(p = 4)$

3.2 - 3.2' $\mathcal{S}_2 = \mathcal{S}_2 \cup \{\{2,4,7;3,5\}, \{2,4,7;6,8\}\}$, $i = 10$

3. $j = 4$, $S_1^4 = \{2,5,7\}$

3.1 $\mathcal{M}(G - \{2,5,7\}) = \{\{1,3\}, \{1,4\}, \{3,6\}, \{6,8\}\}$ $(p = 4)$

3.2-3.2' $\mathcal{S}_2 = \mathcal{S}_2 \cup \{\{2,5,7;3,6\}, \{2,5,7;6,8\}\}$, $i = 12$

3. $j = 5$, $S_1^5 = \{2,5,8\}$

3.1 $\mathcal{M}(G - \{2,5,8\}) = \{\{1,3\}, \{1,4,7\}, \{3,6\}\}$, $(p = 3)$

3.2-3.2' $\mathcal{S}_2 = \mathcal{S}_2 \cup \{\{2,5,8;3,6\}\}$, $i = 13$

3. $j = 6$, $S_1^6 = \{3,5\}$

3.1 $\mathcal{M}(G - \{3,5\}) = \{\{1,4,7\}, \{2,4,7\}, \{2,8\}, \{6,8\}\}$, $(p = 4)$

3.2-3.2' ($\mathcal{S}_2 = \mathcal{S}_2$, $i = 13$)

3. $j = 7$, $S_1^7 = \{3,6\}$

3.1 $\mathcal{M}(G - \{3,6\}) = \{\{1,4,7\}, \{2,4,7\}, \{2,5,7\}, \{2,5,8\}\}$, $(p = 4)$

3.2-3.2' $(\mathcal{S}_2 = \mathcal{S}_2, i = 13)$

3. $j = 8$, $S_1^8 = \{6,8\}$

3.1 $(G - \{6,8\}) = \{\{1,3\}, \{1,4,7\}, \{2,4,7\}, \{2,5,7\}\}$, $(p = 4)$

3.2-3.2' $(\mathcal{S}_2 = \mathcal{S}_2, i = 13)$

4. $\mathcal{S}_2 = \{\{1,3;2,4,7\}, \{1,3;2,5,7\}, \{1,3;2,5,8\}, \{1,3;6,8\}$
$\{1,4,7;2,5,8\}, \{1,4,7;3,5\}, \{1,4,7;3,6\}, \{1,4,7;6,8\}$,
$\{2,4,7;3,5\}, \{2,4,7;6,8\}, \{2,5,7;3,6\}, \{2,5,7;6,8\}$,
$\{2,5,8;3,6\}\}$, $t_2 = 13$

2. $r = 2$, $i = 0$, $\mathcal{S}_3 = \phi$.

3. $j = 1$, $S_2^1 = \{1,3;2,4,7\}$

3.1 $\mathcal{M}(G - \{1,2,3,4,7\}) = \{\{5,8\}, \{6,8\}\}$, $(p = 2)$

3.2-3.2' $\mathcal{S}_3 = \{\{1,3;2,4,7;5,8\}, \{1,3;2,4,7;6,8\}\}$, $i = 2$

3. $j = 2$, $S_2^2 = \{1,3;2,5,7\}$

3.1 $\mathcal{M}(G - \{1,2,3,5,7\}) = \{\{4\}, \{6,8\}\}$, $(p = 2)$

3.2-3.2' $\mathcal{S}_3 = \mathcal{S}_3 \cup \{\{1,3;2,5,7;6,8\}\}$, $i = 3$.

3. $j = 3$, $S_2^3 = \{1,3;2,5,8\}$

3.1 $(G - \{1,2,3,5,8\}) = \{\{4,7\}, \{6\}\}$ $(p = 2)$

3.2-3.2' $\mathcal{S}_3 = \mathcal{S}_3$, $(i = 3)$

3. $j = 4$, $S_2^4 = \{1,3;\ 6,8\}$

3.1 $(G - \{1,3,6,8\}) = \{\{2,4,7\},\{2,5,7\}\}$, $(p = 2)$

3.2-3.2' $\mathcal{S}_3 = \mathcal{S}_3$, $(i = 3)$

3. $j = 5$, $S_2^5 = \{1,4,7;2,5,8\}$

3.1 $(G - \{1,2,4,5,7,8\}) = \{3,6\}$, $(p = 1)$

3.2 $T = \{1,2,3,4,5,6,7,8\}$, $(T = X)$

5. $\chi(G) = r + 1 = 3$, an optimal colour being $\{1,4,7\}$, $\{2,5,8\}$, $\{3,6\}$

This algorithm, however, is relatively costly in terms of both computation time and storage, and a number of simple improvements can be made to the algorithm.

2.4 Simple Modifications to the Above Algorithm

(i) Determination and ordering of the sets S_r^j

As can be easily seen from the example above, ordering the sets S_r^j of the family $\mathcal{S}_r$ in order of decreasing cardinality, [2], would tend to decrease the amount of branching involved during the tree-search. Again, Step 3.1 of the algorithm involves the calculation of all MIS of a graph $G' = G - S_r^j$, which is quite time-consuming. However, as G' is a subgraph of G, any independent set, say S, of G leads to an independent set, $(S - S_r^j)$, of G', and all independent sets of G' are of this form; so that, as has been noted (e.g. [2]), the family $\mathcal{M}(G - S_r^j)$ may be determined, indirectly from the family $\mathcal{M}(G)$.

Moreover, any MIS say M, of G for which $M \cap S_r^j = \phi$ is necessarily an MIS of $G - S_r^j$, so that an efficient routine for determining $\mathcal{M}(G - S_r^j)$, assuming that $\mathcal{M}(G)$ has been saved, can be outlined as follows, (see also section 5.4 of Chapter 3):

Routine for determining $\mathcal{M}(G - S_r^j)$ from $\mathcal{M}(G)$

Step 1. Establish $\mathcal{M}_1 = \{M_i \in \mathcal{M}(G) \mid M_i \cap S_r^j = \phi\}$

Step 2. For each $M_i \in \mathcal{M}(G) - \mathcal{M}_1$, calculate $M_i' = M_i - M_i \cap S_r^j$; and denote by $\mathcal{M}'$ the family of sets M_i'.

Step 3. Determine $\mathcal{M}'' = \{M_i' \in \mathcal{M}' \mid \nexists\, M_k \in \mathcal{M}_1$ with $M_i' \subseteq M_k\}$

Step 4. Establish $\mathcal{M}_2 = \mu\mathcal{M}''$ by removing from $\mathcal{M}''$ any sets which are not maximal and retaining only one copy in the event of duplication of sets.

Step 5. Set $\mathcal{M}(G - S_r^j) = \mathcal{M}_1 \cup \mathcal{M}_2$, and STOP.

(ii) Establishment of the $\mathcal{S}_r$

The reasoning above also shows that $\mathcal{S}_{r+1}$ could similarly have been produced from $\mathcal{S}_r$ by finding all maximal sets of the form $S_r^j \cup M_k$ (for all j and k) where $S_r^j \in \mathcal{S}_r$ and $M_k \in \mathcal{M}(G)$, with the previous routine being used only to simplify the elimination of the non-maximal sets.

Similarly, as can be readily seen from the definition of S_r^i as a maximal subset of G that can be coloured with r colours, we

also have that:

$S_r^i = S_a^j \cup S_b^k$ for some $S_a^j \in \mathcal{S}_a$, $S_b^k \in \mathcal{S}_b$, where $(a+b) = r(a,b \geq 0)$,

the above being true for any $S_r^i \in \mathcal{S}_r$. In particular,

$S_{2p}^i = S_p^j \cup S_p^k$ for any $p \leq \gamma(G)/2$. Thus, the speed of the algorithm can be greatly increased by 'doubling-up' on p-maximal subsets so that instead of determining successively the families $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \ldots$, we would determine $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_4, \mathcal{S}_8, \ldots$.

It is to be noted that when the state is reached for which $X \in \mathcal{S}_{2^k}$, when the doubling-up strategy is being used, this does not imply that $\gamma(G) = 2^k$ but rather that $2^{k-1} < \gamma(G) \leq 2^k$. Therefore, having attained the situation $X \in \mathcal{S}_{2^k}$, one could either revert to the original strategy and determine, with $p = 2^{k-1}$, the families $\mathcal{S}_{p+1}, \mathcal{S}_{p+2}, \ldots$ until such time as $X \in \mathcal{S}_{p+h}$, thus showing that $\gamma(G) = p + h$; or alternatively one could adopt a binary-search strategy between the limits p and 2p, finding $\mathcal{S}_{3p/2}$, then either $\mathcal{S}_{5p/4}$ or $\mathcal{S}_{7p/4}$ etc.

Again, by a judicious choice of the stage at which to commence using the doubling-up strategy, in place of the sequential strategy of the original algorithm, it would be possible to make greater savings. For example, if it is suspected (from a heuristic perhaps) that $\gamma(G)$ is approximately 13, then instead of doubling-up to find $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_4, \mathcal{S}_8, \mathcal{S}_{16}, \mathcal{S}_9$, etc. once could alternatively calculate $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_6, \mathcal{S}_{12}, \mathcal{S}_{24}, \mathcal{S}_{13}$, by not doubling-up until the third level of the search has been reached.

Finally, let us remark that, as is easily seen, the larger the actual chromatic number of the graph, the greater, in general, will be the

savings obtained using the above doubling-up process.

3. PRUNING THE SEARCH-TREE

3.1 Theoretical Justification

Given any optimal colouring of a graph G for which any vertex x_i is assigned coloured $C(x_i)$, one can obtain an equivalent optimal colouring for which some specific vertex x_o say, is assigned colour c say. This is most simply achieved by assigning colour c to all vertices originally assigned colour $C(x_o)$, at the same time assigning colour $C(x_o)$ to all vertices originally coloured c. This idea can be easily applied to the tree- search algorithm of the previous section, in order to reduce the size of the search-tree, by predetermining, at any stage, some specific vertex which is to be assigned the next colour in the sequence of colours.

More specifically, having in mind Theorem 4-1, we see that at the first stage of the algorithm one need only consider (at Step 1) those MIS, M_j, which contain some specific vertex x_1 of X and this fact, when reflected in the algorithm, leads to a drastic reduction in the size of the search tree, with accompanying savings in computing time and storage. Similarly, at any level (r + 1), we need only consider for further branching (at Step 3.1) those MIS of $G - S_r^j$ which contain some specific vertex $x_{r+1} \in X - S_r^j$. Indeed, an obvious choice for the x_{r+1} is, at any stage, to select that vertex which belongs to the fewest MIS of $G - S_r^j$, $(S_o^j = \phi)$.

The above ideas have been incorporated into the basic tree-search algorithm ([2], [3]) where a great increase in efficiency relative to the basic algorithm is reported. It is to be noted that ([2]), when using this modification, as not all r-maximal sets of the graph are generated, it is essential that the portion of the search-tree stemming from any particular S_r^j must now be treated separately from that stemming from any other set S_r^j on level r. In the following example, we shall represent by $\mathcal{S}_{r+1}^j$ the family of (r+1)-maximal subsets stemming from S_r^j, and by x_{r+1} the specific vertex which is to be coloured r + 1.*

Example We now demonstrate this modification with reference to the graph G of Figure 18, (p169), for which we have previously obtained an optimal colouring via the complete breadth-first search algorithm.*

Step 1. $\mathcal{S}_1 = \{\{1,3\}, \{1,4,7\}, \{2,4,7\}, \{2,5,7\}, \{2,5,8\}, \{3,5\}$ $\{3,6\}, \{6,8\}\}$, r = o.

Step 1a. $\underset{x_i \text{ uncoloured}}{\text{Min}} \{|S_1^j \in \mathcal{S}_1 \,\&\, x_i \in S_1^j|\} = 2$; $x_1 = 1$,

$\mathcal{S}_1^1 \{\{1,3\}, \{1,4,7\}\}$.

2a. r = 1, $t_1^1 = 2$, $\mathcal{S}_2^1 = \phi$, $\mathcal{S}_2^2 = \phi$

3a. j = 1, $S_1^1 = \{1,3\}$

3.1 $\mathcal{M}(G - \{1,3\}) = \{\{2,4,7\}, \{2,5,7\}, \{2,5,8\}, \{6,8\}\}$

3.2-3.2' $\mathcal{S}_2^1 = \{\{1,3;2,4,7\}, \{1,3;2,5,7\}, \{1,3;2,5,8\},$ $\{1,3;6,8\}\}$

3a. j = 2, $S_1^2 = \{1,4,7\}$

---

* We shall represent by 2a,3a,4a, the steps corresponding to steps 2,3,4 of the previous algorithm.

3.1 $\mathcal{M}(G - \{1,4,7\}) = \{\{2,5,8\}, \{3,5\}, \{3,6\}, \{6,8\}\}$;

3.2-3.2' $\mathcal{S}_2^2 = \{\{1,4,7;2,5,8\}, \{1,4,7;3,5\}, \{1,4,7;3,6\}, \{1,4,7;6,8\}\}$

4a. $\mathcal{S}_2^1 = \{\{1,3;2,4,7\}, \{1,3;2,5,7\}, \{1,3;2,5,8\}, \{1,3;6,8\}\}$

1a. Min $\{|S_2^j \in \mathcal{S}_2^1 \ \& \ x_i \in S_2^j|\} = 1$; $x_2 = 4$, $\mathcal{S}_2^1 = \{\{1,3;2,4,7\}\}$
x_i uncoloured

2a. $r = 2$, $t_2^1 = 1$, $\mathcal{S}_3^1 = \phi$

3a. $j = 1$, $S_2^1 = \{1,3;2,4,7\}$

3.1 $\mathcal{M}(G - \{1,2,3,4,7\}) = \{\{5,8\}, \{6,8\}\}$

3.2 $\mathcal{S}_3^1 = \{\{1,3;2,4,7;5,8\}, \{1,3;2,4,7;6,8\}\}$

4a. $\mathcal{S}_2^2 = \{\{1,4,7;2,5,8\}, \{1,4,7;3,5\}, \{1,4,7;3,6\}, \{1,4,7;6,8\}\}$

1a. Min $\{|S_2^j \in \mathcal{S}_2^2 \ \& \ x_i \in S_2^j|\} = 1$, $x_2 = 2$,
x_i uncoloured

$\mathcal{S}_2^2 \equiv \{\{1,4,7;2,5,8\}\}$

2a. $r = 2$, $t_2^2 = 1$, $\mathcal{S}_3^2 = \phi$.

3a. $j = 1$, $S_2^2 = \{1,4;7;2,5,8\}$

3.1 $\mathcal{M}(G - \{1,2,4,5,7,8\}) = \{\{3,6\}\}$

3.2 $T = 1,2,3,4,5,6,7,8$. $(T = X)$

5. $\gamma(G) = r + 1 = 3$, an optimal colouring being $\{1,4,7\}$, $\{2,5,8\}$, $\{3,6\}$

Thus we see that the pruning modification has indeed had a marked effect. In the following section we describe a depth-first tree-search which is equivalent to the above pruned breadth-first strategy.

3.2 An Equivalent Depth-First Approach

We have described in the previous section a breadth-first tree search method which could be easily adopted in the form of a depth-first tree search [3]. Although the structure of the search-tree remains basically the same, the order in which the nodes of the tree are generated and then considered is different, the effect of which will be discussed shortly. The main strategy of the depth-first algorithm can be described as follows:

Suppose that we are at a stage of the algorithm where we are about to consider the (maximal) r-colourable subset S_r^j (at level r) to produce maximal (r + 1)-colourable subsets. As before, we determine x_{r+1} so as to belong to the fewest MIS of $G - S_r^j$ and calculate $\mathcal{S}_{r+1}(x_{r+1}, S_r^j) = \{S_{r+1}^1, S_{r+1}^2, \ldots, S_{r+1}^{t_{r+1}}\}$, the family of maximal (r + 1)-colourable subsets all of which contain x_{r+1}. But now, instead of considering the next subset S_r^{j+1} at level r, and not proceeding to the next level until all the r-colourable subsets S_r^j at level r have been considered, we now proceed immediately to level (r + 1) and use the first (maximal) (r + 1)-colourable subset $S_{r+1}^1 \in \mathcal{S}_{r+1}(x_{r+1}, S_r^j)$ for further branching, etc. continuously branching forward until such time as we reach some level, r' say, where for some maximal r'-colourable subset $S_{r'}^{j'}$ we have either

a) $S_{r'}^{j'} \neq X$ and $r' = \bar{r} - 1$, where $\bar{r}$ is the number of colours used in the best colouring of G found so far,

or b) $S_{r'}^{j'} = X$, and $r' < \bar{r}$.

In the former case, it is apparent that the set $S_{r'}^{j'}$ cannot lead to a colouring of G using less than $\bar{r}$ colours, so a backtracking step is taken, that is we consider next $S_{r'}^{j'+1}$ and proceed as before. In the latter case, we have obtained a better colouring of G using only r' colours, so we note down the component sets of $S_{r'}^{j'}$ as the actual best colouring to date, update $\bar{r} = r'$ and take a backtracking step by considering next that $(r' - 1)$-colourable subset $S_{r'-1}^{k}$ of least 'index' k which has not yet been branched from, and then proceed forward as before. If, during a backtracking step, there are no further sets to consider at the appropriate level, a backtracking step is taken to the previous level and so forth.

By initializing $\bar{r}$ as $|X| + 1$ (which is obviously greater than $\chi(G)$, and terminating the search when we are due to backtrack beyond the last set S_1^t, of level 1, we see that the algorithm will terminate with $\bar{r} = \chi(G)$, the corresponding colouring having been stored at the appropriate time.

Example We now illustrate the above depth-first tree-search algorithm by again finding an optimal colouring of the graph G of Figure 18 (p169). In the following trace, $\bar{q}$ represents the value of the best colouring found up until that stage of the algorithm.

Step 1. $\mathcal{S}_1 = \{\{1,3\}, \{1,4,7\}, \{2,4,7\}, \{2,5,7\}, \{2,5,8\}, \{3,5\}, \{3,6\}, \{6,8\}\}$, $r = o$, $\bar{q} = 8$.

1a. $\underset{x_i \text{ uncoloured}}{\text{Min}} \{|S_1^j \in \mathcal{S}_1 \ \& \ x_i \in S_1^j|\} = 2$, $x_1 = 1$, $t_1 = 2$

2a. $r = 1$, $\mathcal{S}_2 = \phi$.

3a. $j_1 = 1$, $S_1^1 = \{1,3\}$

3.1 $\mathcal{M}(G - \{1,3\}) = \{\{2,4,7\}, \{2,5,7\}, \{2,5,8\}, \{6,8\}\}$

3.2-3.2' $\mathcal{S}_2 = \{\{1,3;2,4,7\}, \{1,3;2,5,7\}, \{1,3;2,5,8\}, \{1,3;6,8\}\}$

4a. $\mathcal{S}_2 = \{\{1,3;2,4,7\}, \{1,3;2,5,7\}, \{1,3;2,5,8\}, \{1,3;6,8\}\}$

1a. $\underset{x_i \text{ uncoloured}}{\text{Min}} \{|S_2^j \in \mathcal{S}_2 \ \& \ x_i \in S_2^j|\} = 1$, $x_2 = 4$, $t_2 = 1$

2a. $r = 2$, $\mathcal{S}_3 = \phi$.

3a. $j_2 = 1$, $S_2^1 = \{1,3;2,4,7\}$

3.1 $\mathcal{M}(G - \{1,2,3,4,7\}) = \{\{5,8\}, \{6,8\}\}$

3.2-3.2' $\mathcal{S}_3 = \{\{1,3;2,4,7;5,8\}, \{1,3;2,4,7;6,8\}\}$

4. $\mathcal{S}_3 = \{\{1,3;2,4,7;5,8\}, \{1,3;2,4,7;6,8\}\}$

1a. $\underset{x_i}{\text{Min}} \{|S_3^j \in \mathcal{S}_3 \ \& \ x_i \in S_3^j|\} = 1$, $x_3 = 5$, $t_3 = 1$

2a. $r = 3$ $\mathcal{S}_4 = \phi$

3a. $j_3 = 1$, $S_3^1 = \{1,3;2,4,7;5,8\}$

3.1 $\mathcal{M}(G - \{1,2,3,4,5,7,8\}) = \{\{6\}\}$

3.2 $T = \{1,2,3,4,5,6,7,8\} = X$.

5a. $r = 4$. $\bar{q} = r = 4$, associated colouring $\{1,3;2,4,7;5,8;6\}$

5b. $r = 3$; $j_3 = t_3$

5b. $r = 2$; $j_2 = t_2$

5b. $r = 1$; $j_1 < t_1$

3a. $j_1 = 2$, $S_1^2 = \{1,4,7\}$

3.1 $\mathcal{M}(G - \{1,4,7\}) = \{\{2,5,8\}, \{3,5\}, \{3,6\}, \{6,8\}\}$

3.2-3.2' $\mathcal{S}_2 = \{\{1,4,7;2,5,8\}, \{1,4,7;3,5\}; \{1,4,7;3,6\}, \{1,4,7;6,8\}\}$

4a. $\mathcal{S}_2 = \{\{1,4,7;2,5,8\}, \{1,4,7;3,5\}, \{1,4,7;3,6\}, \{1,4,7;6,8\}\}$

1a. $\underset{x_i \text{ uncoloured}}{\text{Min}} \{|S_2^j \in \mathcal{S}_2 \;\&\; x_i \in S_2^j|\} = 1$, $x_2 = 4$, $t_2 = 1$

2a. $r = 2$, $\mathcal{S}_3 = \phi$

3a. $j = 1$, $S_2^1 = \{1,4,7;2,5,8\}$

3.1 $\mathcal{M}(G - \{1,2,4,5,7\}) = \{\{3,6\}\}$

3.2 $T = \{1,2,3,4,5,6,7,8\} = X$

5a. $r = 3$; $\bar{q} = r = 3$; associated colouring $\{1,4,7;2,5,8;3,6\}$

5b. $r = 2$, $j_2 = t_2$

5b. $r = 1$, $j_1 = t_1$

6. $\chi(G) = \bar{q} = 3$ with associated colouring $\{1,4,7;2,5,8;3,6\}$.

As is indicated by the above example, the depth-first tree-search algorithm described above is essentially the same as the breadth-first

method of section 3.4, the main trade-off between the two being that of some decrease in storage vs. some decrease in computation.

Thus, in the case of the depth-first search only the single family $\mathcal{S}_{r+1}$ stemming from current set S_r^j need be stored (at Step 4a), for any level, r, of the search; while for the breadth-first search, the (maximalised) union of all the families $\mathcal{S}_{r+1}^j$ corresponding to each of the sets S_r^j of $\mathcal{S}_r$ (in that portion of the search-tree being currently processed) are stored at Step 4a. On the other hand, because of this very fact, any set S_{r+1}^h - stemming from S_r^ℓ, say, - which is dominated by some set S_{r+1}^k - stemming from S_r^i, say, where $i \neq \ell$ - will not be (explicitly) considered by the breadth-first algorithm as it would have been found to be dominated in the formation of $\mathcal{S}_{r+1}$ from the various $\mathcal{S}_{r+1}^j$ at Step 4a. Since dominated sets of this type would in fact be branched from - needlessly - in the case of the depth-first search it is seen that breadth-first search involves, in general, some computational savings.

All in all, however, a tree-search algorithm of the depth-first variety is generally more useful, especially in view of the following facts:

(i) The extra storage requirements of the breadth-first search of themselves lead to an increase in computation time (particularly so if it necessitates list-processing techniques).

(ii) More importantly, the depth-first search quickly establishes a feasible solution which can then be used for branching purposes throughout the remainder of the tree-search.

For this reason, we shall consider mainly the depth-first variants of the graph-colouring algorithms to be discussed in the following

section. Before proceeding to the next section, however, we make the important observation that it is not valid to incoporate the doubling-up procedure of section 2.3 into either of the pruned tree-search algorithms described in the current section. This is because doubling up at any level r would generate only those 2r-colourable subsets of G which are composed of the unions of two r-colourable subsets, each of which contains x_r, rather than all 2r-colourable subsets of G which contain x_r. Therefore not all possiblities would be considered throughout the course of the tree-search, so invalidating the algorithm.

4. A MULTIPLE ALGORITHM FOR OPTIMAL GRAPH COLOURING

4.1 Introduction

We have seen above that progressing from Christofides' original variant of the basic method, the number of branchings at any specific stage of the algorithm has been methodically reduced, so increasing its efficiency. Nevertheless the algorithm is still computationally relatively inefficient, as we shall see in Section 5, where the various graph-colouring algorithms are computationally evaluated.

One major reason for this deficiency can be seen by considering more closely the depth-first variant of the algorithm. In common with so many depth-first algorithms, the optimal solution to the problem is found, in general, at a relatively early stage of the search, and by far the greater portion of the total search effort is expended in verifying that the solution found is indeed optimal. This is achieved, as we have seen earlier, by considering (via forward branchings) all other possible combinations which have not hitherto been considered,

and all of which must eventually be rejected as they cannot lead to a better solution.

It is then apparent that any algorithm (of a depth-first type) which would enable one to terminate the search immediately an optimal solution has been attained (assuming of course, that only one optimal colouring is required if more than one exists) would be of great computation significance.

(Note that although any breadth-first tree search algorithm seemingly satisfies the above criterion, it does so merely by the artifice of considering the 'extra' portion of the search-tree before, instead of after, the attainment of the optimal solution, and so leads to no real gain.)

In this section we describe such an algorithm which indeed compares very favourably with the methods of the previous sections.

4.2 Some Theoretical Considerations

In the previous chapter we proved (Theorem 4-2) that for the special case of graph G whose colouring number is equal to the cardinality of its maximum clique, one need consider, for inclusion an optimal colouring, only those MIS of G which satisfy the further condition of intersecting all the maximum cliques of G. This is in general a considerable refinement of Theorem 4-1, on which the algorithms described in the previous section are based. Unfortunately, it is rarely the case that, for a given graph G, $\gamma(G) = \omega(G)$, so that Theorem 4-2 cannot be applied directly. However, after establishing the following Lemma, we can construct an algorithm for optimally colouring any graph G which makes use of Theorem 4-2.

Lemma 1: Let $G = (X, \Gamma)$ be any graph of order n for which $\gamma(G) > \omega(G)$, and let Q be any maximum clique of G. Then the graph $G' = (X', \Gamma')$ defined by

$$\left.\begin{aligned} X' &= X \cup \{x_{n+1}\}, && x_{n+1} \notin X \\ \Gamma' x_i &= \Gamma x_i \cup \{x_{n+1}\} && \text{for } x_i \in Q \\ \Gamma' x_i &= \Gamma x_i && \text{for } x_i \in X - Q \\ \Gamma' x_{n+1} &= Q \end{aligned}\right\}$$

satisfies (a) $\omega(G') = \omega(G) + 1$; (B) $\gamma(G') = \gamma(G)$

Proof: (a) Since x_{n+1} is adjacent to every vertex of the clique Q, it is obvious that $(Q \cup \{x_{n+1}\})$ must be a clique of G' of cardinality $|Q| + 1$. However, as Q is a maximum clique of G, we have $|Q| = \omega(G)$, while the fact that x_{n+1} adjacent only to vertices of Q implies that no other clique of G' can be of cardinality greater than $\omega(G)$. Thus $\omega(G') = \omega(G) + 1$ *.

(b) Because Q is a completely connected set in G', the vertices of Q will be assigned ($\omega(G)$) different colours in any colouring of G', call these colours $1, 2, \ldots, \omega(G)$, while the vertices of $(G - Q)$ can obviously be coloured with (at most) $\gamma(G)$ colours. Again, since it is not adjacent to any vertex of $(G - Q)$, x_{n+1} can be assigend colour $\omega(G) + 1$. Therefore, since $\omega(G) < \gamma(G)$, we have $\gamma(G') = \gamma(G)$, so completing the proof ::

Cor 1: $\gamma(G') - \omega(G') = \gamma(G) - \omega(G) - 1$

---

* Indeed, the reasoning above shows that $(Q \cup \{x_{n+1}\})$ is the unique maximum clique of G'.

Proof: Immediate.

Recalling that $\omega(G)$ is obviously a lower bound for $\gamma(G)$, the rationale of the algorithm can now be described.

4.3 An Overview of the Multiple Algorithm

We first assume that $\gamma(G) = \omega(G)$ so that Theorem 4-2 would apply to graph G and we embark on a depth-first search which is bascially similar to that described in the previous section, but in which the number of branchings at any stage, k, is considerably reduced by the application of Theore 4-2 to the (sub) graph G_k corresponding to those vertices of G not yet coloured at that stage. If at any stage there exists an uncoloured vertex, x_o, such that none of the MIS of G_k containing x_o intersect all maximum cliques of G_k, then a backtracking step is taken. Thus two possibilites can occur.

(a) A feasible colouring is eventually obtained (which necessarily satisfies Theorem 4-2 at every stage); in which case, by Theorem 4-2, the colouring is optimal and the procedure can be terminated at this point.

(b) We backtrack past level 1 without attaining a feasible colouring; in which case, (again by Theorem 4-2), the assumption $\gamma(G) = \omega(G)$ is shown to be false and $\gamma(G) \geq \omega(G) + 1$. We then suppose that $\gamma(G) = \omega(G) + 1$, or equivalently, as seen from Lemma 1, we construct the graph $G' = (X', \Gamma')$ from the graph G by adjoining a 'dummy' vertex to a maximum clique of G to form a clique of cardinality $(\omega(G) + 1)$. For this graph G' we have then by assumption $\gamma(G') = \omega(G')$ so that we can repeat the whole

procedure using graph G' instead of graph G, until either
(i) a feasible (and so optimal) colouring of G' is obtained which induces an optimal colouring of G in $\omega(G) + 1$ colours, by assigning to the vertices of G the colours assigned to them by the optimal colouring of G' just found, or
(ii) no feasible colouring can be obtained in which case $\gamma(G') > \omega(G')$ (i.e. $\gamma(G) > \omega(G) + 1$), so that we construct the graph $G^2 = (X^2, \Gamma^2)$ by adjoining a further dummy vertex to the (unique) maximum clique of G'. Then by assuming $\gamma(G^2) = \omega(G^2)$ (i.e. $\gamma(G) = \omega(G) + 2$) we repeat the whole procedure on G^2 ... etc.

Eventually, since $\gamma(G) \leq n$ where $n = |X|$, we must reach the points where $\gamma(G^h) = \omega(G^h)$ for some $h \geq 0$, $(G^o \equiv G)$, in which case an optimal colouring (using $(\omega(G) + h$ colours) is obtained in the obvious manner.

We shall now discuss the implementation of this algorithm in somewhat greater detail.

4.4 Implementation of the Multiple Algorithm

As mentioned in the previous subsection, there are two phases to the algorithm: (a) the main depth-first strategy applied to a graph G^i, and
(b) the transition from the graph G^i to the graph G^{i+1}.

We deal first with the transition phase of the algorithm.

(i) The Transition Phase

It can be shown that by actually augmenting the graph by a dummy vertex at each transitional step, one would successively,

double* the number of MIS of each graph G^i, so that it would be impractical to implement the Multiple Algorithm according to the above straight forward approach. We therefore describe now an equivalent, mor practicable approach.

A. An Equivalent Approach

The following result can easily be shown:

Lemma 2: Let G^i be any graph which contains a unique maximum clique, Q_o, of which a dummy vertex x' is an element. Then any subgraph G', of G, which includes x' satisfies the property that each MIS of G' intersects Q_o.

Proof: Let M' be any MIS of G'. If M contains a vertex of $(Q_o - \{x'\})$, then obviously M' intersects Q_o. On the other hand, if M' does not contain any vertex of $(Q_o - \{x'\})$, then, since M' is maximal - and recalling the fact that x' is adjacent only to vertices of $(Q_o - \{x'\})$ -, M' must contain x' so that again M' intersects Q_o (since $x' \in Q_o$) ::

Considering again the transition from graph G^i to graph $G^{i+1} \equiv (G^i \cup \{x_{n+i+1}\})$, we recall that if Q_o is any maximum clique of G^i, $(Q_o \cup \{x_{n+i+1}\})$ is the unique maximum clique of G^{i+1}. Removing from G^{i+1} any MIS M_1 which intersects all maximum cliques of G^i (so that, in particular, $x_{n+i+1} \notin M_1$) will then obviously still leave the clique $(Q_o \cup \{x_{n+i+1}\}) - (M_1 \cap Q_o)$ as the unique maximum clique of the graph $(G^{i+1} - M_1)$ (since $|M_1 \cap Q_o| = 1$). Similarly removing from $(G^{i+1} - M_1)$ any MIS M_2 which intersects

---

* More precisely, the transition from G to G' more than doubles the number of MIS.

all maximum cliques of $(G^i - M_1)$ (i.e. $G^{i+1} - (M_1 \cup \{x_{n+i+1}\})$ will leave $(Q_o \cup \{x_{n+i+1}\}) - ((M_1 \cup M_2) \cap Q_o)$ as the unique maximum clique of $(G^{i+1} - (M_1 \cup M_2))$ ... etc. Recalling that the formation of G^{i+1} is necessary only because $\delta(G^i) > \omega(G^i)$ and making use of Theorem 4-2, we know that we will eventually reach a a state where we can find no MIS M_k of $(G^{i+1} - \bigcup_{j=1}^{k-1} M_j)$ which is also an MIS of $G' \equiv (G^i - \bigcup_{j=1}^{k-1} M_j)$ and which intersects all maximum cliques of G'. We then, have in the sequence corresponding to 4-3 of Chapter 4 :

$$\omega(G^i) = \omega(G_1^i) = \ldots = \omega(G_{k-1}^i) < \omega(G_k^i)$$

On the other hand, $G'' \equiv (G^{i+1} - \bigcup_{j=1}^{k-1} M_j)$ will contain a unique maximum clique - viz. $((Q_o \cup \{x_{n+i+1}\}) - \bigcup_{j=1}^{k-1} (M_j \cap Q_o))$ - which also includes the dummy vertex x_{n+i+1}. Therefore by Lemma 2, we can find an MIS, M_k, of G'' which intersects 'all' maximum cliques of G''; and M_k must, (by the previous sentence) contain x_{n+i+1}.

Therefore, $\omega(G^{i+1}) = \omega(G_1^{i+1}) = \ldots = \omega(G_k^{i+1})$

Furthermore, since $x_{n+i+1} \in M_k$, we have that:

$$G^{i+1} - (\bigcup_{j=1}^{k-1} M_j \cup M_k) \equiv G^i - (\bigcup_{j=1}^{k-1} M_j \cup (M_k \cap X^i)),$$

so that we see that the sole effect of the addition of x_{n+i+1} to G^i to form G^{i+1} is to create one fewer strict inequality in the sequence

$\omega(G^{i+1}) \leq \ldots \leq \omega(G_\ell^{i+1})$ relative to the sequence $\omega(G^i) \leq \ldots \leq \omega(G_\ell^i)$.

B. Implementing the Transition Phase

In the context of the algorithm, then, one need in fact never add any dummy vertices. The conceptial equivalence of G^i is merely to allow exactly i nodes on every particular path from the root of the search tree for which we do not restrict the choice of those MIS being considered for further branching to satisfy the Intersection Theorem (I.T.) with respect to all maximum cliques of G_u, the subgraph of as yet uncoloured vertices, (thus relaxing the Intersection condition). Instead we require the satisfaction of the Intersection Test for all but one maximum clique, the exception being that clique of G_u deriving from Q_o. The transition from G^i to G^{i+1} is then simply to allow one further node of this type on every path from the root of the search tree. (Note that, for the graph corresponding to G^i, there will be i stages at which the clique deriving from Q_o will be the sole maximum clique, in which case the intersection condition is in fact non-existent).

In a similar fashion, it can now be seen that, going from G^i to G^{i+1}, we need not calculate the cliques of G^{i+1} from those of G^i by using the results of 3-5. Instead, having originally chosen some particular maximum clique of G, we add unity to the cardinality of the clique (signifying the addition of the new dummy vertex) and reduce by unity its cardinality for any node of the search-tree for which we do not satisfy the I.T. (so signifying the use of a dummy vertex as part of an MIS). We note that the only additional 'bookeeping' associated with this is that we must remember to nullify the reduction in cardinality - by adding unity - when a backtracking step is taken from such a node (signifying the return of an (unused) dummy vertex).

This completes our discussion of the implementation of the transition phase of the Multiple Algorithm. We now discuss the main depth-first phase of the algorithm.

(ii) The Depth First Phase

A. Practical Considerations

The main complication introduced by depth-first phase of the Multiple Algorithm relative to the depth-first algorithm of section 3 4 is that, at every stage of the depth-first search, where the maximal r-colourable subset S_r^j is about to be considered for branching purposes, we must be able to determine, for any candidate MIS of the graph $G' = (G^i - S_r^j)$, whether that MIS intersects all maximum cliques of G' (G^i is, as before, the graph on which the depth=first procedure is applied).

The obvious method of achieving this is by calculating, at the start of the depth-first process, the family, $\mathcal{Q}^i$, of cliques of G^i, either from the family, $\mathcal{Q}^{i-1}$, of cliques of G^{i-1}, by using a result analagous to Theorem 3-3 (see also section 4,4 of Chapter 3), or, in the case of $i = 0$, directly by using one of the clique-generation algorithms of Chapter 3 on the original graph $G^o \equiv G$. Thereafter, parallel to the routine employed to determine the MIS of the subgraphs (see section 2.2 of this chapter), one would find all the cliques of $G' = (G^i - S_r^j)$ from the family $\mathcal{Q}^i$ and then determine the family of cliques of G' which are of maximum cardinality.

However, this latter process would effectively double the running time of the algorithm. Furthermore, in accordance with the analysis of the previous section, it is not necessary to form any G^{i+1} explicitly, in which case it would not be efficient to calculate its cliques. Therefore it is of prime importance to calculate actual families of cliques as few times as possible. In fact, as we shall show presently, one need calculate only one family of cliques, viz. the cliques of the original graph G, throughout the course of the Multiple Algorithm.

B. Implementation of the Intersection Theorem

We showed in section 3-5 that the MIS of the graph $G' = G - S_r^j$ are simply the maximal sets of the family $\mathcal{M}' = \{(M_i - S_r^j) | (M_i \in \mathcal{M}(G)\}$. The sole purpose of determining $\mathcal{M}(G - S_r^j)$, instead of simply using the family $\mathcal{M}'$, for the purposes of branching from S_r^j, was then to eliminate from further consideration, dominated (independent) sets (i.e. those sets of $\mathcal{M}'$ which are subsets of other sets of $\mathcal{M}'$). This is useful because it is obvious that any dominated set cannot lead to a better colouring than that attained by using its dominating set; eliminating dominated sets, as soon as possible, will then reduce the size of the search-tree. Hence it is worthwhile to calculate the family $\mathcal{M}(G - S_r^j)$ explicitly.

In the case of the cliques, on the other hand, the family $\mathcal{Q}(G - S_r^j)$, of cliques of $G - S_r^j$ is used for checking purposes only (viz. the I.T.) and not for further branching per se. Therefore, as the search-tree is in no way affected, it is not computationally worthwhile to calculate $\mathcal{Q}(G - S_r^j)$ but rather to use the family $\mathcal{Q}' = \{(Q_i - S_r^j) \mid Q_i \in \mathcal{Q}(G)\}$, instead, where, as before, $\mathcal{Q}' \supseteq \mathcal{Q}(G - S_r^j)$; provided, of course that the checking criterion is not altered.

This latter fact is, however readily established, for the I.T. concerns only those cliques of $\mathcal{Q}(G - S_r^j)$ which are of maximum size. Therefore, since $\mathcal{Q}(G - S_r^j)$ is formed from $\mathcal{Q}'$ by removing only non-maximal subsets it is apparent that any maximum cliques in $\mathcal{Q}'$ would also be $\mathcal{Q}(G - S_r^j)$, and vice versa.

Indeed, we make use of the I.T. involving $\mathcal{Q}' = \{(Q_i - S_r^j) \mid Q_i \in \mathcal{Q}(G)\}$, only to check the suitability for branching of a set $M \in \mathcal{M}(G - S_r^j)$, and since $M \cap (Q_i - S_r^j) = M \cap Q_i$ (for $M \cap S_r^j = \phi$) for any $Q_i \in \mathcal{Q}(G)$ we see that we can use $\mathcal{Q}(G)$ itself for the Intersection check at any stage, provided we have some means of knowing, at level r, exactly which cliques, Q_i, of G would generate the maximum cliques $(Q_i - S_r^j)$ of $(G - S_r^j)$.

The mechanism to provide the answer to the above question is remarkably simple, however. Recalling that for any completely connected set Q and any independent set M, we know that $|Q \cap M| \leq 1$, we merely define a vector Q SIZE of cardinality $|\mathcal{Q}(G)|$ which we initialize by setting Q SIZE (i) equal to the cardinality of the clique Q_i of G (at the time of determination of $\mathcal{Q}(G)$), and then reduce Q SIZE (i) by unity at any forward branching stage for which the appropriate MIS, say $M = (M_i - S_r^j)$, intersects the clique Q_i, whilst leaving Q SIZE (i) unchanged if M does not intersect Q_i. Similarly, one augments Q SIZE (i) by unity at any backtrack from a stage for which the appropriate MIS intersects Q_i; but leaves it unchanged otherwsie. By letting $Q = Q_1$, say, be that maximum clique of G to which we conceptually add the dummy vertices at the transition phases, we would at first have Q SIZE (1) = $\omega(G)$ so that during the first depth-first phase, recalling that we assume $\gamma(G) = \omega(G)$, we would have Q SIZE (1) decreasing by unity at every forward branching (in view of the I.T.); and correspondingly increasing by unity at every backtrack. Thus during the first transitional phase, Q SIZE (1) will represent, at any level r, the

cardinality of the maximum cliques of $(G - S^{k}_{r-1})$ $(S_o = \phi)$ so that exactly those cliques Q_i for which Q SIZE (i) = Q SIZE (1) at level r are used for the I.T. at that level, this being so for any $1 \leq r \leq \gamma(G)$.

Again, when a transition phase arises, all that is necessary is to increase Q SIZE (1) by unity to simulate the augmentation of Q_1 with a dummy vertex so that Q SIZE (1) = $\omega(G^i) + 1 = \omega(G^{i+1})$, and, exactly as before, at any level r, the cliques which are used for the intersection check all satisfy Q SIZE (i) = Q SIZE (1) at that stage. Recalling that during the depth-first search applied to the conceptional graph G^i there will be exactly i stages on every forward path from the root for which the I.T. will be relaxed, we can now formulate the algorithm.

4.5 Description of the Multiple Depth-First Algorithm for Optimal Graph Colouring

(Initial Calculations) 1. Calculate $\mathcal{M}(G) = \{M_1, M_2, \ldots, M_t\}$, the family of MIS of the graph $G = (X, \Gamma)$. Calculate $\mathcal{Q}(G) = \{Q_1, Q_2, \ldots, Q_s\}$; the family of cliques of G, and ensure (by an interchange if necessary) that

$$|Q_1| = \max_{1 \leq i \leq s} \{|Q_i|\}$$

For each i = 1,2,...s put Q SIZE (i) = $|Q_i|$

(Initialize first transition phase) 2. Put f = -1, QSIZE (1) = Q SIZE (1) - 1, r = 0 $S^1_o = \phi$, $\mathcal{S}_o = \{S^1_o\}, t_o = 1$

(Next Transition)

3. Put $f = f + 1$, $g = 0$, Q SIZE (1) = Q SIZE (1) + 2

4. Put $h_r = 0$

(Get Next Set of Level r)

5. Put $h_r = h_r + 1$, & put $j = h_r$

If $j > t$, put $r = r - 1$ and go to Step 15.

(Update Q SIZE)

6. For $i = 2, 3, \ldots, s$:

6.1 If $S_r^j \cap Q_i = 0$, then put

Q SIZE (i) = Q SIZE (i) - 1

6'. Iterate.

7. Put Q SIZE (1) = Q SIZE (1) - 1

(Find those MIS (if any) satisfying the Intersection Theorem without recourse to dummy vertex)

8. Determine $\hat{\mathcal{Q}} = \{Q_i \in \mathcal{Q} | Q \text{ SIZE } (i) = Q \text{ SIZE } (1)\}$

Calculate $\mathcal{M}(G - S_r^j)$; the family of MIS of $G - S_r^j$ and determine $\hat{\mathcal{M}}(G - S_r^j)$ as

$$\left\{ M_k^{r+1} \in \mathcal{M}(G - S_r^j) \middle| (M_k^{r+1} \cap Q_i) = 1 \quad \forall Q_i \in \hat{\mathcal{Q}} \right\}$$

If $\hat{\mathcal{M}}(G - S_r^j) \neq \phi$, go to Step 11, otherwise go to Step 9.

(If no such MIS exists and no dummy vertices remain, backtrack)

9. If $g = f$ go to Step 16, otherwise go to Step 10.

(Find the MIS, using a dummy vertex, which satisfies the I.T.)

10. Determine $\hat{\mathcal{M}}(G - S_r^j)$ as:

$$\left\{ M_k^{r+1} \in \mathcal{M}(G - S_r^j) \middle| M_k^{r+1} \cap Q_i) = 1 \; \forall Q_i \in (\hat{\mathcal{Q}} - Q_i) \right.$$

Update $g = g + 1$.

(Find next x_{r+1})

11. Find $x_{r+1} \in X - S_r^j$ belonging to the least number, t_{r+1}, of sets $\in \hat{\mathcal{M}}(G - S_r^j)$, and re-index these sets as $M_1^{r+1}, M_2^{r+1}, \ldots, M_{t_{r+1}}^{r+1}$

(Calculate $\mathcal{S}_{r+1}$)

12. For $i = 1,2,\ldots,t_{r+1}$:

12.1 Put $S^i_{r+1} = S^j_r \cup M^{r+1}_i$

12'. Iterate.

(If only one MIS is left we are finished)

13. If $t_{r+1} = 1$ go to Step 19; otherwise go to Step 14.

(Branch Forward)

14. Put $r = r + 1$ and go to Step 4.

(Backtrack)

15. If $r = 0$ go to Step 3;

otherwise go to Step 16.

16. Put $j = h_r$.

(Update Q SIZE)

17. For $i = 2,\ldots|2|$

17.1 If $M^r_j \cap Q_i \neq \phi$, put Q SIZE (i) = Q SIZE (i) + 1

17'. Iterate.

(Return dummy vertex if necessary)

18. Put Q SIZE (1) = Q SIZE (1) + 1.

If $M^r_j \cap Q_1 = \phi$, put $g = g - 1$. Go to Step 5.

(Terminate)

19. Stop with $r + 1$ as $\gamma(G)$, the corresponding colouring being given by $M^1_{h_1}, M^2_{h_2}, \ldots, M^{r+1}_1$

Example: We demonstrate the workings of the Multiple Algorithm, again by determining an optimal colouring for the graph G of Figure 18 (p 169).

Step 1. $\mathcal{M}(G) = \{\{1,3\}, \{1,4,7\}, \{2,4,7\}, \{2,5,7\}, \{2,5,8\}, \{3,5\}, \{3,6\}, \{6,8\}\}$

$\mathcal{Q}(G) = \{\{1,2,6\}, \{1,5,6\}, \{1,8\}, \{2,3\}, \{3,4,8\}, \{3,7,8\}, \{4,5,6\}, \{6,7\}\}$, $t = 8$, $s = 8$

Q SIZE (i) = 3 for i = 1,2,5,6,7; Q SIZE (i) = 2 for i = 3,4,8

2. $f = -1$, Q SIZE (1) = 2, $r = 0$, $S_o^1 = \phi$, $\mathcal{S}_o = \phi$, $t_o = 1$

3. $f = 0$, $g = 0$, Q SIZE (1) = 4

4. $h_o = 0$

5. $h_o = 1$, $j = 1$

6-6'. $S_o^1 = \phi$

7. Q SIZE (1) = 3

8. $\hat{\mathcal{Q}} = \{\{1,2,6\}, \{1,5,6\}, \{3,4,8\}, \{3,7,8\}, \{4,5,6\}\}$
$\mathcal{M}(G - \phi) = \{\{1,3\}, \{1,4,7\}, \{2,4,7\}, \{2,5,7\}, \{2,5,8\}, \{3,5\}, \{3,6\}, \{6,8\}\}$
$\hat{\mathcal{M}}(G - \phi) = \{\{1,4,7\}, \{2,5,8\}, \{3,6\}, \{6,8\}\}$

11. $x_1 = 1$, $t_1 = 4$

12=12' $S_1^1 = \{1,4,7\}$, $S_1^2 = \{2,5,8\}$, $S_1^3 = \{3,6\}$, $S_1^4 = \{6,8\}$

13. $t_1 \neq 1$

14. $r = 1$

4. $h_1 = 0$

5. $h_1 = 1$, $j = 1$, $(S_1^1 = \{1,4,7\})$

6-6'. Q SIZE (i) = 2 for i = 2,4,5,6,7; Q SIZE (1) = 3;
Q SIZE (3) = 1, Q SIZE (8) = 1

7. Q SIZE (1) = 2

8. $\hat{\mathcal{Q}} = \{\{1,2,6\}, \{1,5,6\}, \{2,3\}, \{3,4,8\}, \{3,7,8\}, \{4,5,6\}\}$
$(G - \{1,4,7\}) = \{\{2,5,8\}, \{3,5\}, \{3,6\}, \{6,8\}\}$
$(G - \{1,4,7\}) = \{\{2,5,8\}, \{3,6\}\}$

11. $x_2 = 2$, $t_2 = 2$

12-12'. $S_2^1 = \{1,4,7;2,5,8\}$, $S_2^2 = \{1,4,7;3,6\}$

13-14. $r = 2$

4-5. $h_2 = 1$, $j = 1$ ($S_2^1 = \{1,2,4,5,7,8\}$)

6-6'-7. Q SIZE (i) = 1 for i = 1,2,4,5,6,7,8; Q SIZE (3) = 0

8. $\hat{\mathcal{Q}} = \{\{1,2,6\}, \{1,5,6\}, \{2,3\}, \{3,4,8\}, \{3,7,8\}, \{4,5,6\}, \{6,7\}\}$

$\mathcal{M}(G - \{1,2,4,5,7,8\}) = \{\{3,6\}\}$

$\mathcal{M}(G - \{1,2,4,5,7,8\}) = \{\{3,6\}\}$

11. $x_3 = 3$, $t_3 = 1$

12-12'. $S_3^1 = \{1,4,7;2,5,8;3,6\}$

13. $t_3 = 1$

19. STOP with $\chi(G) = 3$, an optimal colouring being $\{1,4,7\}$, $\{2,5,8\}$, $\{3,6\}$

In the above example an optimal colouring was obtained very quickly, in fact with no backtracking being necessary. This occurrence is of course not general, the reason for the fine performance in the above case being due to the fact that G satisfies $\chi(G) = \omega(G)$, which we have seen to be the optimum situation for an algorithm based on the I.T.

We note here that removing the arc $(4,6)$ of the graph G to form the graph G' will still result in the graph G' satisfying $\chi(G') = \omega(G')$, and indeed the Multiple Algorithm will proceed in exactly the same manner to obtain an optimal colouring for G' in one pass. On the other hand, it is reported in [3] that using the ordinary depth-first search algorithm to colour G' would result in a search-tree which is "too long to be included" in the paper, thus illuminating further the power of the Multiple Algorithm.

Example As a simple demonstration of the Multiple Algorithm for the case where $\gamma(G) > \omega(G)$ we find an optimal colouring for the graph G of Figure 19.

Step 1. $\mathcal{M}(G) = \{\{1,3,5\}, \{1,3,6\}, \{1,4,6\}, \{2,4,6\}, \{2,4,7\}, \{2,5,7\}, \{3,5,7\}\}$

$\mathcal{Q}(G) = \{\{1,2\}, \{1,7\}, \{2,3\}, \{3,4\}, \{4,5\}, \{5,6\}, \{6,7\}\}$

Q SIZE (i) = 2, i = 1,2,3,4,5,6,7

2. f = 1, Q SIZE (1) = 1, r = o, $S_o^1 = \phi$, $\mathcal{S}_o = \phi$, $t_o = 1$.

3. f = 0, g = 0, Q SIZE (1) = 3.

4-5. $h_o = 1$, j = 1, $(S_o^1 = \phi)$.

6-7. Q SIZE (i) = 2 $\forall i$

8. $\hat{\mathcal{Q}} = \mathcal{Q}(G)$; $\hat{\mathcal{M}}(G - \phi) = \phi$

9. g = f

15. r = o

3. f = 1, g = 0, Q SIZE (1) = 4

4-5. $h_o = 1$, j = 1, $(S_o^1 = \phi)$

6-7 Q SIZE (1) = 3

8. $\hat{\mathcal{Q}} = \{1,2\}$; $\mathcal{M}(G - \phi) = \mathcal{M}(G)$

11. $x_1 = 1$, $t_1 = 3$

12-12'. $S_1^1 = \{1,3,5\}$, $S_1^2 = \{1,3,6\}$, $S_1^3 = \{1,4,6\}$

13-14. r = 1

4-5. $h_1 = 1$, j = 1, $(S_1^1 = \{1,3,5\})$

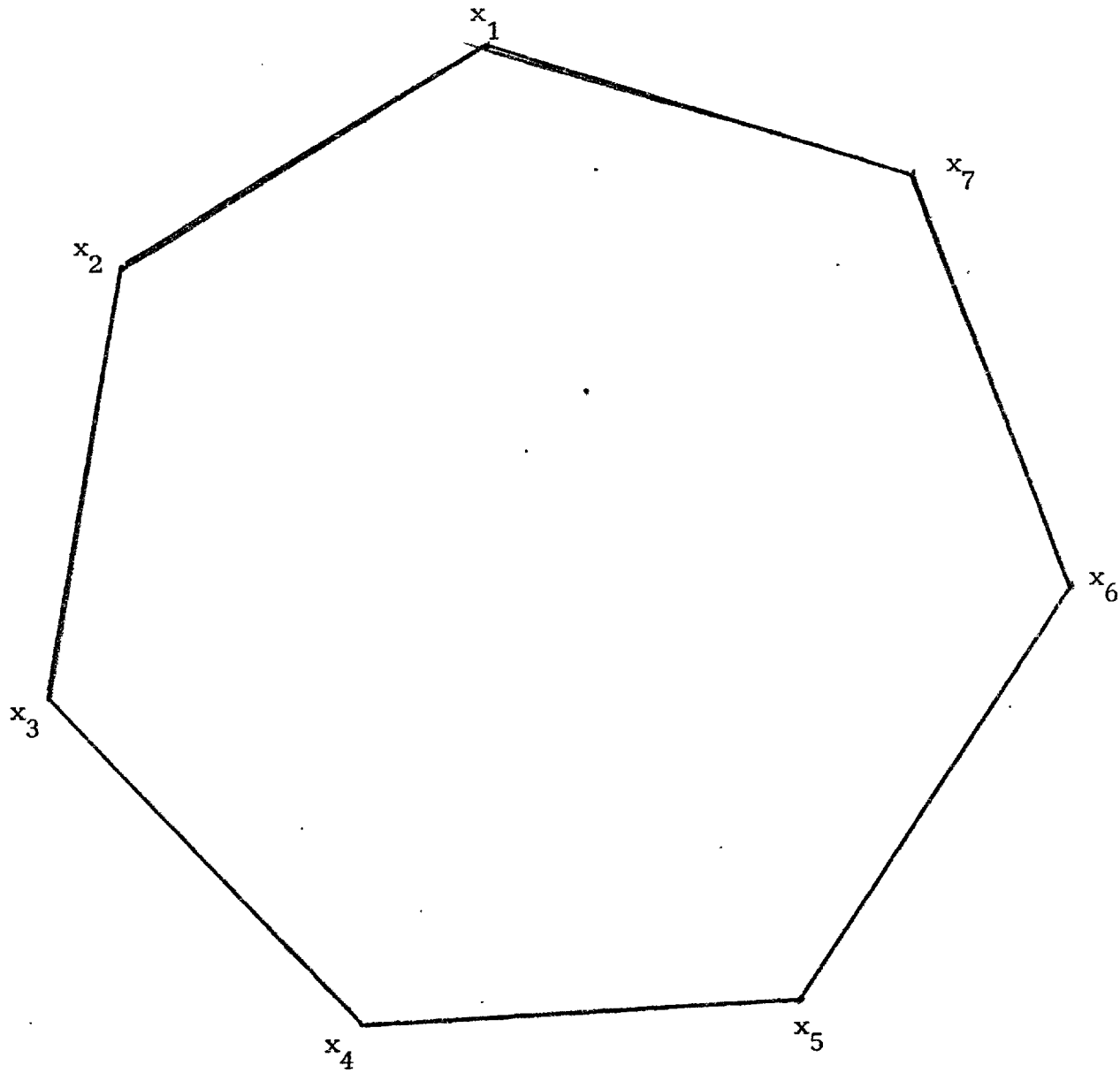


FIGURE 19

6-6'. Q SIZE (i) = 1, for i = 2,3,4,5,6. Q SIZE (1) = 3
Q SIZE (7) = 2.

7. Q SIZE (1) = 2.

8. $\hat{\mathcal{L}} = \{\{1,2\}, \{6,7\}\}$, $\hat{\mathcal{M}}(G - \{1,3,5\}) = \{\{2,4,6\}, \{2,4,7\}\}$
$\hat{\mathcal{M}}(G - \{1,3,5\}) = \{\{2,4,6\}, \{2,4,7\}\}$

11. $x_2 = 6$, $t_2 = 1$.

12-12'. $S_2^1 = \{1,2,3,4,5,6\}$;

13-14 $r = 2$

4-5. $h_2 = 2$, $j = 2$, $S_2^1 = \{1,2,3,4,5,6\}$

6-6'. Q SIZE (i) = 0 for i = 3,4,5,6. Q SIZE (1) = 2,
Q SIZE (2) = 1 = Q SIZE (7)

7. Q SIZE (1) = 1.

8. $\hat{\mathcal{L}} = \{\{1,2\}, \{1,7\}, \{6,7\}, \hat{\mathcal{M}}(G - \{1,2,3,4,5,6\}) = \{\{7\}\}$.
$\hat{\mathcal{M}}(G - \{1,2,3,4,5,6\}) = \phi$.

9. $g < f$

10. $\hat{\mathcal{M}}(G - \{1,2,3,4,5,6\}) = \{\{7\}\}$, $g = 1$

11. $x_3 = 7$

12. $S_3^1 = \{1,2,3,4,5,6,7\}$

13. $t_3 = 1$.

19. $\gamma(G) = 3$ with colouring $\{1,3,8\}$, $\{2,4,6\}$, $\{7\}$.

4.6 Computational Considerations of the Multiple Algorithm

We shall consider now in greater detail the choice of the specific vertex x_r (see section 4.4) at level r and the consequnces of such a choice. Recall that in the simple depth-first algorithm, one chose the vertex which belonged to the fewest MIS of the uncoloured subgraph G_u, with ties being broken arbitrarily. This strategy can also be employed in the Multiple Algorithm prior to the intersection check which would then be applied to establish the family of candidate MIS containing x_r; and we shall give the computational results of this strategy in a later section.

However, as we shall see in Section 5 , it is far more efficient to apply the intersection check to <u>all</u> the MIS of the uncoloured subgraph G_u^f, and only then to choose that vertex x_r which belongs to the least number of the resulting candidate MIS. Now if G_u^f should contain dummy vertices (i.e. if $(f - g) > 0$ in the algorithm) we know from Lemma 2 that all MIS of G_u^f must intersect all maximum cliques of G_u^f. Some of these MIS may intersect Q_1 as well as all other maximum cliques (if any), while the rest of the MIS will intersect only these other maximum cliques but not Q_1, thereby necessitating the (conceptional) 'colouring' of a dummy vertex (i.e. $g \leftarrow g + 1$).

Now by Corollary 3 of Theorem 4-3, we know that if one of the MIS currently being considered at level r should lead to a feasible (hence optimal) colouring of G then in the sequence $\omega(G_r^f) \leq \omega(G_{r+1}^f) \leq \ldots \leq \omega(G_{\gamma(G)}^f)$ there will be only equalities. It would then be advantageous to have the portion of the search from

this level as small (thin) as possible. A good heuristic for attaining this objective is then to select for the next forward branching some MIS which intersects all the maximum cliques of G_u^f without necessitating the use of a dummy vertex (assuming some remain). This is because of the fact - mentioned above - that any MIS containing a dummy vertex will always intersect all maximum cliques of the corresponding graph. Hence it is useful to reserve this 'free' intersection until such time as we are definitely forced to use it (as will certainly be the case eventually, else the dummy vertex would never have been added).

Similarly, in the situation where no MIS from this level (with the present partial colouring) will lead to an optimal colouring it is useful to backtrack from this level at the earliest opportunity. Thus, should there be a vertex x' which is in no MIS which intersects all maximum cliques of G_u^f (without recourse to a dummy vertex), so that using the dummy vertex would now in fact be forced, then selecting to branch on x' will lead to early backtracking, as desired. (Of course, the fact that the use of a dummy vertex is forced at any stage does not imply that no optimal colouring can be obtained from this branching, but rather that the chances are less than that for some other branching from this level which preserves its dummy vertices).

Therefore, in the implementation of the Multiple Algorithm, a tie-breaking criterion is used for choosing the specific vertex x_r as follows:

Find that vertex of $G - S_r^j$ which is contained in the fewest number of MIS of $G - S_r^j$ which intersect all maximum cliques of $G - S_r^j$. In the case of ties choose among these the (real) vertex which is contained in the fewest MIS which intersect all maximum cliques of $G - S_r^j$ but, which do not contain any real vertices of Q_1. Again,

by the reasoning above, once having chosen x_r , it is beneficial to order the MIS containing x_r so that those MIS which contain a 'real' vertex of Q_1 are considered before those which do not, and only then ordering each of these subfamilies according to non-increasing cardinality.

To underline the above ideas, we give in Table 6, (p208), the computational results obtained by adopting various heuristic strategies for the choices of the specific vertex x_r in the Multiple Algorithm; and these can then be compared with the results, shown in Table 7 (p 209) obtained by using the strategy discussed above.

This completes our detailed description of the Multiple Algorithm.

4.7 Variants of the Multiple Algorithm

We recall that the basis of the method is to assume $\gamma(G) = \omega(G)$, do a depth-first search based on this assumption and using the I.T., then add a dummy vertex to some specific maximum clique of G, and do a depth-first search based on the I.T. and the assumption $\gamma(G) = \omega(G) + 1$ etc., until a feasible colouring is attained which is, in fact, optimal. As is easily seen, a depth-first search, based on the I.T. for a graph G^f including f dummy vertices will terminate with a feasible colouring, using q colours, if and only if

$$q \leq \omega(G) + f \qquad (2)$$

Therefore, we can emply a 'dual' form of the Multiple Algorithm by using the inequality

$$\gamma(G) \leq n \qquad (3)$$

(where $n = |X|$, the cardinality of the vertex set of G) as follows :

1. Add $n - \omega(G)$ dummy vertices to G and, using the depth-first search, obtain (by (2) and (3)) a feasible colouring of G using q_o colours. (Note that q_o will in general be much less than n, and that only q_o nodes of the search-tree are used throughout this search).

2. Next, add $q_o - 1 - \omega(G)$ dummy vertices to G (since now we know $\gamma(G) \leq q_o$) and again, by depth-first search, obtain a feasible solution using q_1 colours (where now $q_1 \leq q_o - 1$).

3. Continue as above using successively $q_1 - 1 - \omega(G)$, $q_2 - 1 - \omega(G), \ldots$ dummy vertices until a state is reached where the addition of $q_t - \omega(G)$ dummy vertices does not lead to a feasible colouring of G.

4. We then have (using (2), together with the fact that $\gamma(G) \leq q_i$ where q_i is the number of colours in any feasible colouring),

$$\gamma(G) > q_t$$

$$\gamma(G) \leq q_{t-1}$$

which imply, since $q_t = q_{t-1} - 1$ that $\gamma(G) = q_{t-1}$.

This variant has also been programmed in a similar (although not exactly equivalent) manner to that of the original Multiple Algorithm, and the computational results are given in Section 5.

Yet another approach to the GCP is to use both variants in the same algorithm in a general strategy of bi-directional search.

Thus, one would proceed alternately forward from no dummy vertices, then backward from $n - \omega(G)$ additional dummy vertices etc., until either the forward (primal) variant or

the dual variant establishes an optimal colouring as mentioned before. The rationale behind this particular approach is that, since the size of the depth-first search tree involving f dummy vertices varies inversely as the absolute value of $((\gamma(G) - \omega(G)) - f)$, one would hope to substitute depth-first searches for large values of $|\gamma(G) - \omega(G) - f|$ for those for small values of $|\gamma(G) - \omega(G) - f|$, thereby saving some computation time.

In practice, however, since in the primal variant, f increases by exactly unity at each phase, while in the dual variant, f decreases by unity except (in general) for the first one or two phases, the bi-directional approach does not, in general, lead to significant gains in efficiency.

4.8 SCP and GCP

Finally, we mention that the Multiple Algorithm strategy (i.e. the use of the I.T. together with the concept of dummy vertices) can be embedded - in any of its variants-into the set-covering tree search algorithm of Chapter 2 (for solving the GCP). However, as will be seen in the following Section, where computational results for various colouring algorithms are detailed, the computational results for the SCP strategy (with the I.T.) reflect the loss of flexibility involved in predetermining, via the fixed block structure, the choice of the various vertices x_r which determine the forward branchings from the corresponding levels r.

Conversely, the strategy of keeping lists of previously branched-from sets at each level as introduced in the above mentioned SCP algorithm can be incorporated into the depth-first methods of this section to limit the search tree. This incomplete dynamic programming strategy was indeed implemented as an option in our computer program

of the Multiple Algorithmbut with not so significant a gain as would occur with the normal depth-first algorith, due to the search tree being already greatly reduced by the I.T.

Again, the bound, described in Section 3.3 of Chapter 2 for the SCP algorithm, is also valid - in its unicost form - for the GCP, and so it, too, may be incorporated into the Multiple Algorithm. Indeed, using the I.T. will improve the bound somewhat when used as part of a graph-colouring algorithm. For we recall that the bound was based upon a computation concerning all the MIS of the uncoloured subgraph at any stage; whereas the use of the I.T. may in fact eliminate some of these MIS from further consideration at this stage, thereby possibly tightening the bound.

It appears however, that the strength of the I.T. would, to a large extent dominate the bound (in its unicost form), and hence the bound was not incorporated into our computer program on which the computational results of the following section are based.

5. COMPUTATIONAL RESULTS

In this Section we give the tables already referres to in section 4.6. Table 6 compares various strategies for selecting the specific vertex on which to branch, while Table 7 compares various depth-first colouring algorithms. As these results have already been mentioned previously in the chapter, we do not go into further detail here, the table being basically self-explanatory, but merely confirm that the primal and dual versions of the Multiple Algorithm are computaionally similar, both being better than the SCP with I.T., with its fixed block structure.

TABLE 6 VARIOUS VERTEX SELECTION STRATEGIES

VERTICES	DENSITY	TOTAL NODES IN SEARCH-TREE				
		A	B	C	D	E
25	.1	3	3	3	53	3
	.3	7	5	5	200	5
	.5	10	13	10	228	9
	.7	14	9	9	115	9
	.9	14	14	14	95	14
30	.1	3	3	3	114	3
	.3	7	9	7	1115	5
	.5	44	20	12	775	15
	.7	39	16	14	442	14
	.9	17	16	15	125	16
35	.3	57	36	19		17
	.5	206	46	32		39
	.7	116	33	28	579	23
40	.5		175	146		154
	.7		349	254	1372	238

A Choose Specific Vertex x_r as belonging to the fewest MIS of G

B Choose x_r as belonging to the fewest MIS which intersects the real vertices of Q_1 as well as all other maximum cliques of G

C Choose x_r as belonging to fewest MIS which intersect all maximum cliques of G.

D Consider MIS which do not intersect the real vertices of Q_1 before any which do,

E Choose x_r as belonging to the fewest MIS which intersect all maximum cliques of G but do not intersect the real vertices of Q_1.

TABLE 7 DEPTH-FIRST TREE-SEARCH ALGORITHMS

VERTICES	DENSITY	TIMES (CDC 6400 secs.)				NODES IN TREE		
		A	B	C	D	A	B	C
25	.1	13.9	.6	.6	72.0	962	3	3
	.3	11.6	.3	.4	13.1	694	5	6
	.5	7.8	.3	.4	2.9	1180	9	12
	.7	2.0	.2	.2	2.4	438	9	9
	.9	.1	.3	.2	.3	14	14	14
30	.1		1.8	3.5			3	3
	.3		.9	1.5	11.2		5	8
	.5	155.6	1.0	1.4	9.2	14218	15	21
	.7	28.3	.6	1.0	7.2	5013	14	25
	.9	.8	.7	.7	.5	40	16	16
35	.3		18.8	21.7			17	23
	.5		4.9	5.6	31.5		39	47
	.7		1.7	2.4	21.4		23	34
	.9		1.2	1.8	1.0		19	22
40	.5		30.1				154	
	.7		26.2		135.5		238	
	.9				3.8			

A Depth-first method of Wang [3]

B The Multiple Algorithm

C The Dual Version of the Multiple Algorithm

D The SCP Tree Search Algorithm (Section 3 of Chapter 2) and incorporating the Intersection Theorem

6. REFERENCES

1. CHRISTOFIDES, N. "An Algorithm for the Chromatic Number of a Graph", Computer Journal 14 (1971), pp 38-39.

2. ROSCHKE, S.I., and FURTADO, A.L. "An Algorithm for Obtaining the Chromatic Number and an Optimal Colouring of a Graph", Information Processing Letters 2 (1973), pp 34-38.

3. WANG, C.C. "An Algorithm for the Chromatic Number of a Graph", Journal of A.C.M. 21 (1974), pp 385-391.

CHAPTER VI

VERTEX-BY-VERTEX GRAPH COLOURING

The previous chapter dealt with one of the two major strategies for the direct optimal colouring of graphs, by the assignment of (independent) sets of vertices to the various colours, thus partitioning the vertex set of G. In this chapter we discuss the other main (and more obvious) method of directly solving the GCP, by the sequential assignment of colours to vertices. This latter method is also the basic strategy used in many non-optimal heuristics for 'solving' the GCP. (e.g. [6,7,8])

1. THE BASIC ALGORITHM AND SOME VARIATIONS

1.1 A Simple Vertex-by-Vertex Colouring Algorithm

In what follows we shall assume that the vertices of X, where $G = (X, \Gamma)$ have an associated sequential ordering (at this stage arbitrary) $x_1, x_2, \ldots x_n$. An elementary and well-known algorithm (e.g. [1,2,4]) for finding an optimal colouring for G is then to colour the vertices of G in ascending order of their index. Thus, initially one assigns colour 1 to x_1, and then colours the remaining vertices in sequence by assigning to vertex x_i the colour $C(x_i)$ represented by the lowest integer which has not been assigned to any of the vertices adjacent to x_i and of index lower than i. That is, each vertex in order is assigned the lowest feasible colour, so that:

$$C(x_i) \leq \max_{j<i} \{C(x_j)\} + 1, \quad i = 1,2,\ldots,n \qquad (1)$$

When x_n has been coloured in this manner, a feasible colouring of G will have been attained using q_1, say, colours where

$q_1 = \max_{1 \le i \le n} \{ C(x_i) \}$. Though this colouring will rarely be optimal (after the first pass), q_1 can obviously serve as an upper bound for the next phase of the algorithm, whereby we attempt to generate a feasible colouring using $q_2 \le q_1 - 1$ colours. To accomplish this we would of course have to recolour (at least) those vertices to which we had assigned colour q_1, by backtracking as follows.

Suppose x_ℓ to be the vertex of lowest index which has been assigned colour q_1. Then, according to the rule used for colouring, it must necessarily be true that x_ℓ is adjacent to vertices of lower index coloured with each of colours $1,2,\ldots,q_1 - 1$. Let x_k be that vertex adjacent to x_ℓ of highest index less than ℓ; x_k is then to be the first vertex to be recoloured in the second phase. Again, by our colouring rule, x_k will have been coloured with colour $C(x_k)$ only because it is adjacent to vertices of lower index to which we have assigned colours $1,2,\ldots, C(x_k) - 1$, so that $C(x_k) + 1$ is the lowest alternative colour available to recolour x_k. If $C(x_k) = q_1 - 1$ we must backtrack again immediately (by recolouring some vertex x_h $(h<k)$ currently assigned some colour $C(x_h) \le q_1 - 2$) before branching forward again since we are now seeking to colour G by using at most $q_1 - 1$ colours.

On the other hand, if $C(x_k) + 1 < q_1$, we recolour x_k with colour $C(x_k) + 1$ and proceed forward again as before, colouring

x_{k+1} with its smallest feasible colour, etc. until such time as either x_n is coloured, or some vertex x_p is reached which requires colour q_1. In the former case a feasible colouring of $q_2 = q_1 - 1$ colours has been obtained and by replacing q_1 by q_2 in the above procedure, we backtrack again to try to find a colouring using $q_3 \leq q_2 - 1$ colours; while in the latter case we backtrack from x_p, and proceed forward as before. If we eventually ever need to backtrack from x_1, then, since all recolouring possibilities have been exhausted, the algorithm terminates with q as the chromatic number, where q is the (least) integer for which a feasible colouring has been obtained at the end of the previous phase, and with the actual colouring being given by the $C(x_i)$ in use at that time.

Example: Consider the graph G of Figure 2[illegible]. The sequential values of the $C(X_i)$ assigned to the vertices of G will then be as follows (where again we represent a vertex by its index) :

$$C(1) = 1,\ C(2) = 2,\ C(3) = 1,\ C(4) = 1,\ C(5) = 2,\ C(6) = 3,\ C(7) = 4,$$
$$C(5) = 3,$$
$$C(4) = 2,\ C(5) = 1,\ C(6) = 3,$$
$$C(5) = 3,$$
$$C(4) = 3,\ C(5) = 1,\ C(6) = 3,$$
$$C(5) = 2,\ C(6) = 3,$$
$$C(3) = 3,\ C(4) = 1,\ C(5) = 2,\ C(6) = 3,\ C(7) = 1.$$

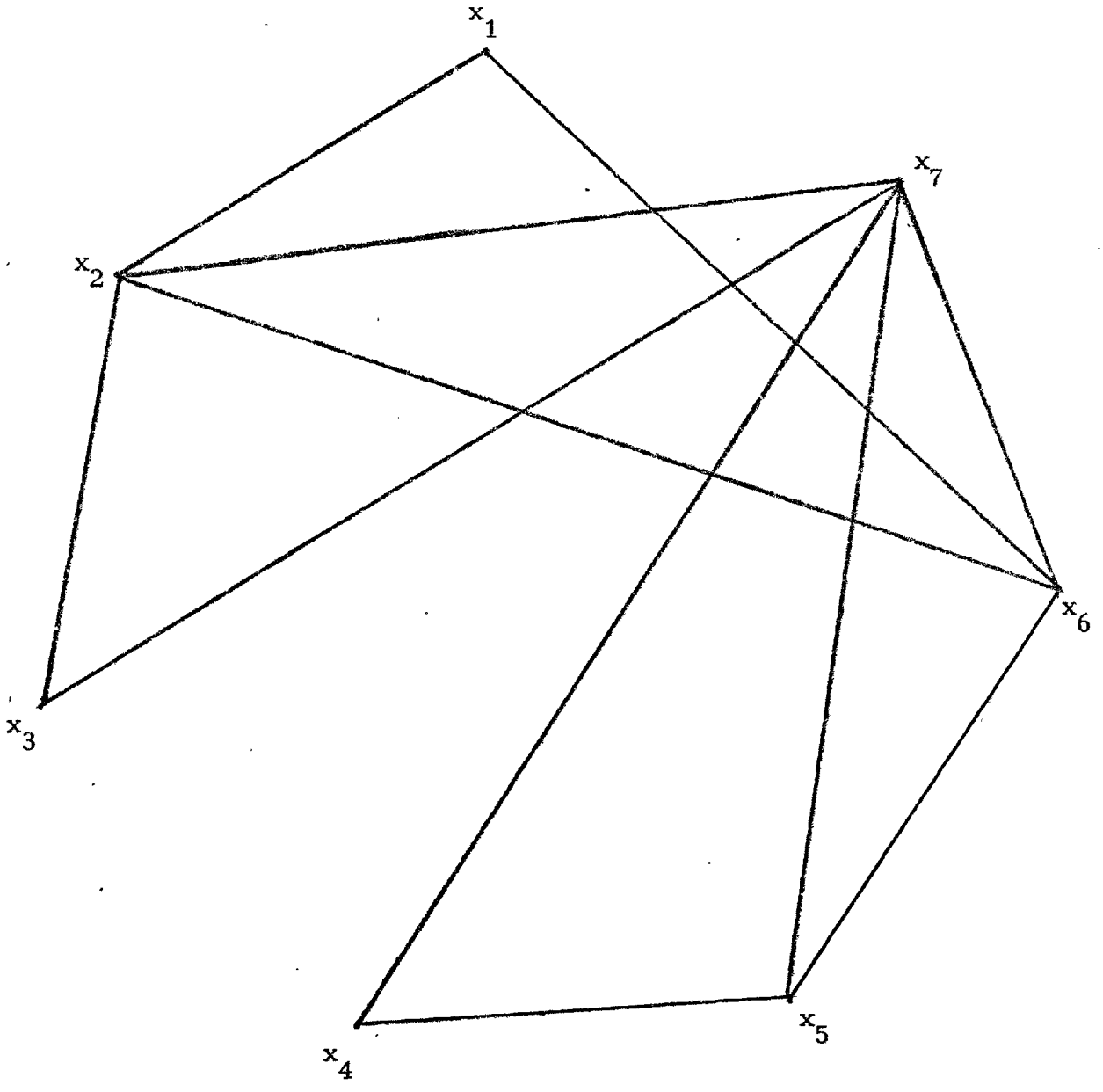


FIGURE 20

1.2 Improvements to the Simple Algorithm

In the presentation given above, we have implicitly made use of the fact that if a vertex x_i cannot be coloured with any of the colours used for the previous $(i-1)$ vertices, then one need only consider the assignment of colour $q + 1$ to x_i, where $q = \max_{j<i}\{C(x_j)\}$, as the sole alternative for forward branching at this stage. This result, which is also formally proved in [1], leads naturally to some limitation of the size of the search tree, - and indeed we have already used a variant of this result to similar effect in the previous chapter.

In particular, we know then that the only colour which will be assigned to vertex x_1 throughout the course of the algorithm will be colour 1. Similarly, the only alternatives which will be considered for vertex x_2 will be colours 1 and 2, depending on whether x_1 and x_2 are to have the same or different colours. Thus, if, x_2 is adjacent to x_1 (so that they cannot be coloured the same) then the only colour considered for x_2 will be colour 2. Repeating this argument, we see then that if the vertices $x_1, x_2, \ldots, x_p$ constitute a clique there will be but one colour assigned to each throughout the course of the algorithm, so that the algorithm can be terminated when all backtrackings from vertex x_{p+1} have been completed, thus again reducing the search.

Recalling that the ordering of the vertices prior to the algorithm was arbitrary, it is then obvious that arranging the vertices so that the first $\omega(G)$ vertices constitute a maximum clique of the graph G, [2], will improve the efficiency of the algorithm. A further (but lesser) enhancement is to order the remaining vertices so that, for any

i, vertex x_i is adjacent to more of the vertices $x_1, x_2, \ldots, x_{i-1}$ than are any of the vertices $x_{i+1}, \ldots, x_n$, (ties being broken by choosing the vertex of greater degree [1]). Indeed, as finding a maximum clique of a graph is itself a non-trivial problem and as the latter criterion above will in general lead to a near-maximum clique anyway, it is reasonable to use this criterion to order all the vertices, with x_1 then being a vertex of maximum degree in G.

Another improvement, [1], is to use a look-ahead procedure during the course of the algorithm in an effort to reduce the number of backtracking steps, as follows.

Whenever any vertex x_i is assigned a colour $C(x_i)$, we note down the set of those vertices x_j with $j>i$, for which $C(x_i)$ is no longer a feasible assignment with respect to the current colouring of $x_1, x_2, \ldots, x_i$, and we keep a record to the effect for each such vertex x_j and hence, when any vertex, x_k say, is to be coloured we know which colours, $\{c_p\}$ are available. For each of these feasible colours, the purpose of the lookahead procedure is to find out what effect assigning c_p to x_i (i.e. setting $C(x_i) = c_p$) would have on each x_j, where $j>i$. Thus it may occur that assigning c_p to x_i will leave some future vertex x_j with no feasible assignments less than or equal to q_h, the current upper bound (i.e. the best colouring attained to date) for the chromatic number; in which case we can backtrack immediately by not assigning c_p to x_i.

Also, this procedure can also be used as a method of directing the search. Thus, one determines, for each feasible colour c_p for

x_i the number of vertices x_j adjacent to x_i which in the future could possibly be assigned c_p, if c_p is not assigned to x_i (termed in [1] the number of preventions.) Then instead of considering the set of colours which are feasible for vertex x_i in their natural sequential order when branching from x_i, the set of feasible colours is now arranged in an order corresponding to the number of preventions.

Example With reference to the graph G of Figure 20, let us define a vector π_s as the number of preventions for each feasible colour c_p, at stage s. Then the workings of the look-ahead procedure can be demonstrated as follows (where the minimum of $\pi(c_p)$ is taken only over feasible possible colours bearing in mind the reasoning at the start of this section. Also ties are broken in favour of the lowest colour.)

Stage	Activity	Next Vertex	$\pi(c_1)$	$\pi(c_2)$	$\pi(c_3)$	$\pi(c_4)$
0	Initiallize $\pi(c_1) = 0$	x_1	2	-	-	-
1	C(1) = 1	x_2	-	3	-	-
2	C(2) = 2	x_3	1	-	-	-
3	C(3) = 1	x_4	1	1	-	-
4	C(4) = 1	x_5	-	1	-	-
5	C(5) = 2	x_6	-	-	1	-
6	C(6) = 3	x_7	-	-	-	0
7	C(7) = 4	x_5	-	-	0	-
8	The assignment C(5) = 3 is stopped by the bound	x_4	-	1	2	-

Stage	Activity	Next Vertex	$\pi(C_1)$	$\pi(C_2)$	$\pi(C_3)$	$\pi(C_4)$
9	C(4) = 2	x_5	0	-	0	-
10	C(5) = 1	x_6	-	-	0	-
11	The assignment C(6) = 3 is stopped by the bound	x_5	-	-	0	-
12	As stage 8	x_4	-	-	2	-
13	C(4) = 3	x_5	0	0	-	-
14	C(5) = 1	x_6	-	-	0	-
15	As stage 11	x_5	-	0	-	-
16	C(5) = 2	x_6	-	-	0	-
17	As stage 15	x_3	-	-	0	-
18	C(3) = 3	x_4	2	1	1	-
19	C(4) = 2	x_5	1	-	1	-
20	C(5) = 1	x_6	-	-	0	-
21	As stage 17	x_5	-	-	1	-
22	As stage 12	x_4	2	-	1	-
23	C(4) = 3	x_5	1	0	-	-
24	C(5) = 2	x_6	-	-	0	-
25	C(6) = 3	x_7	0	-	-	-
26	C(7) = 1	x_5	1	-	-	-
27	STOP (bound in operation from now on)					

Even though in this particular example the tree-search is essentially similar to that of the basic algorithm, in general there are savings to be made; and although an additional measure of 'book-keeping' is established in keeping track of the state of the search, it is reported in [1] that this procedure is computationally worthwhile. (See also Section 4 where our own computational results are given).

1.3 Further Modifications and a Lower Bound

It can be easily seen (cf Chapter 4, Section 3) that completing the colouring of an n-vertex graph $G = (X, \Gamma)$, having already fixed the colouring, - using p colours - of some subset $X' = \{x_1, \ldots x_t\}$ of its vertex set, is equivalent to colouring the $(n - t + p)$-vertex 'pseudograph' $\tilde{G} = (\tilde{Q} \cup (X-X'), \tilde{\Gamma})$, where $\tilde{G}$ is obtained by attaching a p-clique, $\tilde{Q}$, of pseudo-vertices to the uncoloured subgraph $(G-X')$. More explicitly, for each assigned colour c, the set X_c of all vertices in X' which are assigned colour c is coalesced into a pesudo-vertex $\tilde{x}_c$, which is then joined by an arc to each vertex of $X-X'$ from which there is (in G), an arc to any vertex of X_c; finally, all pseudo-vertices are joined by arcs to one another.*

It is, however, readily apparent that at any stage of the colouring algorithm of the previous sections, when we are about to colour vertex x_k, the colours assigned to vertices $x_1, x_2, \ldots, x_{k-1}$ will already have been fixed and indeed will remain so until we eventually backtrack to x_{k-1}. Thus, the pseudo-graph concept may be utilized in the following manner:

(i) We have previously shown (in Chapter 1) that if, for any two non-adjacent vertices x_r and x_s, we have $\Gamma x_r \subseteq \Gamma x_s$, then x_r is 'dominated'; and can always be assigned the same colour as x_s with no adverse effects (as far as optimal colouring is concerned). Hence x_r may be (temporarily) ignored. Since for any pseudo-vertex

---

* A pseudo-vertex $\tilde{x}_c$ corresponding to the colouring of $x_i, x_j, x_k, \ldots$ with colour c, say where $i < j < k$ can be simply visualized in the place of the actual vertex x_i with the removal from the graph of the other vertices $x_j, x_k, \ldots$

$\tilde{x}_c$, we have $\tilde{\Gamma}\tilde{x}_c \equiv \bigcup_{\substack{i<k \\ C(x_i)=c}} (\Gamma x_i \cap (X-X')) \cup (\tilde{Q}-\{\tilde{x}_c\})$, so that in particular $\Gamma x_i \subseteq \tilde{\Gamma}\tilde{x}_c$. Thus any condition of the form $\Gamma x_r \subseteq \Gamma \tilde{x}_c$ would tend to be more effective than the condition $\Gamma x_r \subseteq \Gamma x_s$, where $1 \leq s \leq k$ and $C(x_s) = c$. Therefore, the use of the pseudo-vertex at any stage k would serve to further eliminate fruitless forward branchings on x_k.

(ii) Again, as is easily seen from the definition, $\omega(G_i)$ the cardinality of the maximum clique of successive pseudo-graphs, each containing one fewer (uncoloured) vertex of G, is a non-decreasing function, whose lowest value $\omega(G_0)$ is the cardinality of the maximum clique of the graph G itself. Recalling that for any graph G, $\omega(G)$ is a lower bound for $\gamma(G)$, and that $\gamma(G_i) \geq \gamma(G)$ for any pseudo-graph G^i derived from G, we see then if we reach a stage for which the corresponding pseudo-graph $\tilde{G}'$ satisfies:

$$\omega(\tilde{G}') \geq q_h \qquad (2)$$

where, as before, q_h is the value of the best colouring obtained to date, a backtracking step may immediately be taken. That is, the cardinality of the maximum clique of $\tilde{G}'$, or indeed any known lower bound on $\gamma(\tilde{G}')$, can be used to limit the search for any optimal colouring of G which incorporates the current assignments $C(x_i)$. It is to be noted that, since $\omega(\tilde{G}') \geq \omega(G)$, inequality (2) represents a better bounding condition than does the simple condition $\omega(G) \geq q_h$. Moreover, in the case of a pseudo-graph $\tilde{G}'$ for which the clique of pseudo-vertices, $\tilde{Q}'$, is of cardinality $q_h - 1$, the above bound is equivalent to the backtracking criterion given in the previous section.

Little extra effort is required to establish the appropriate

pseudo-graph at any stage of the (vertex-by-vertex) algorithm and so the above concepts are easily incorproated into the main algorithm. We note, too, that these two modifications based on the concept of pseudo-graphs are quite general and can in fact be incorporated into any graph-colouring algorithm, should it be felt to be computationally worthwhile.

Example: Consider again the graph G of Figure 20. The pseudo-graph $\tilde{G}$ corresponding to the assignments $C(1) = 1$, $C(2) = 2$, $C(3) = 1$ is shown in Figure 20, where it is easily seen that $\omega(\tilde{G}) = 4$. Thus having established the initial colouring as before, no further forward branching will take place (because of bound (2)), until $C(3) = 3$, at which time the optimal colouring will be generated as before. Thus we see that much of the search-tree of the previous example has been eliminated.

1.4 An Improved Vertex-by-Vertex Colouring Algorithm

As we have mentioned above, the basic vertex-by-vertex colouring algorithm can be improved by considering the possible feasible colours which may be assigned to any vertex in some heuristic order other than the natural order, the vertices themselves being considered in the natural initial order, which was fixed, whether explicitly or implicitly, at the start of the procedure. However, as we shall verify in Section 4, the basic algorithm can be greatly improved by a dynamic re-ordering of the as-yet-uncoloured vertices as the algorithm progresses.

Thus, the basis of this vertex re-ordering strategy is to arrange, at any stage, that the vertex to be coloured next is that one, amongst

all uncoloured vertices, which has the smallest number of feasible possible colour assignments available to it. This heuristic strategy will then naturally result in a reduction of the size of the total search-tree for the problem by terminating those forward branchings which do not in fact lead to optimal solutions at as early a stage as possible.

The above procedure is indeed relatively easy to incorporate into the basic algorithm, and in fact, as we shall see in greater detail in Section 4, it gives rise to a marked reduction in computing times. Moreover, using the vertex re-ordering procedure at each stage, will serve a similar purpose as the backtracking criterion of the look ahead procedure of [1]. For, if at any stage of the algorithm there exists some vertex x_k which cannot be assigned any colour less than or equal to q_h, the number of colours in the best colouring to date, (i.e. x_k has no feasible possible assignments open to it) then x_k will be chosen as the next vertex to be coloured and a back-tracking step will immediately ensue.

Example: We consider yet again the graph G of Figure 20 (p.215) as an illustration for this algorithm. Let us define a 'Possibility Table' to give the number of feasible possibilities $P(x_i)$ open to each vertex x_i at each stage of the algorithm. The computation then proceeds as follows (where ties for minimisation are taken in favour of the least index):

Stage	Activity	P(1)	P(2)	P(3)	P(4)	P(5)	P(6)	P(7)
0	Initialize $P(i) = n\ \forall i$	7	7	7	7	7	7	7
1	$C(1) = 1$	-	6	7	7	7	6	7
2	$C(2) = 2$	-	-	6	7	7	5	6
3	$C(6) = 3$	-	-	6	7	6	-	5
4	$C(7) = 1$	-	-	5	6	5	-	-
5	$C(3) = 3$	-	-	-	6	5	-	-
6	$C(5) = 2$	-	-	-	5	-	-	-
7	$C(4) = 3$	-	-	-	-	-	-	-
8	STOP since x_2 cannot be coloured 1							

Thus we see that by dynamically re-ordering the vertices much backtracking can be saved relative to the original algorithm (although, of course, there will in general be some backtracking).

We note, finally, that the re-ordering of the vertices as detailed above is independent of the order in which feasible colours are considered for any vertex, so that we can also apply the colour-reordering modifications of [1] to the above modified algorithm.

1.5 A 'Dual' Vertex-by-Vertex Colouring Algorithm

In the previous sections, an algorithm was described which involves the finding of feasible colourings of the graph G which use successively fewer colours q_i, until the situation is obtained where no 'colouring' of G using less than q_h, say, colours is feasible. This situation then gives rise to the assertion that q_h is optimal (i.e. $\gamma(G) = q_h$), and the successful termination of the algorithm.

However, just as in the case of the Multiple Algorithm of the previous chapter, so too here it is possible to formulate a 'dual' algorithm whereby one finds colouring of G which use successively more colours q_j and which are 'optimal' but not feasible (so that each q_j is a lower bound on $\gamma(G)$ until such time as a colouring of G, using q_h, colours, is obtained which is feasible; which is then the optimal solution to the GCP.

A simple way to implement such an algorithm would be to first re-order the vertices, as before, so that $\{x_1, x_2, \ldots, x_{\omega(G)}\}$ represents a maximum clique of G, and then to branch forward as before, (and colouring vertices with colours $1, 2, \ldots \omega(G)$ respectively, etc) but with the restriction that no colours greater than q (initially $q = \omega(G)$) are to be used. If a stage is reached where it is impossible to continue without assigning colour $q + 1$, a backtracking step is take, following which a further attempt is made to branch forward, etc., as seen earlier. If all n vertices can be thus coloured, then $\gamma(G) = q$, and the algorithm is terminated with this optimal feasible colouring. If, however, it is found that G cannot be coloured with only q colours - (so that, as indicated in a previous section, we have backtracked fully on $x_{\omega(G)+1}$) - then q is increased by unity, and a further attempt is made to colour G with this new value of q colours. This process is then continually applied until such time as a value of q is obtained for which G can be coloured using only these q colours - in which case the algorithm terminates with $\gamma(G) = q$.

In fact, as was the case in the previous chapter, where another dual algorithm was established, the computation results, which are given in

Section 4 of this chapter, indicate that the orders of efficiency of the primal and dual versions of the vertex-by-vertex colouring algorithms are essentially equivalent to one another.

Again, the phase of the above dual algorithm which corresponds to any given q is similar to that of the original vertex-to-vertex colouring algorithm, and it is therefore immediately obvious that any of the modifications and improvements detailed in the earlier sections of this chapter (and, in particular, the dynamic vertex re-ordering procedure) can equivalently be incorporated into the dual algorithm.

1.6 Vertex-by-Vertex Colouring and Bi-directional Search

Since any particular phase of either of the primal or dual algorithms above is essentially independent of any other phase of the same algorithm, here too they can be combined into a bi-directional tree search strategy by running primal and dual phases alternatively. Thus initially we find a feasible colouring of p_1, (say) colours using the primal algorithm and put $p = p_1 - 1$. We then initiallize $q = \omega(G)$ for the dual algorithm. Then, returning to the primal, having previously stored the current state of its search (in a simple manner), we seek a colouring involving p colours which if found allows the updating of p as $p-1$. Reverting again to the dual, if a colouring is found involving q colours this is optimal and the algorithm terminates; on the other hand, if no such colouring can be found we update $q = q + 1$ and return to another phase of the primal method. We continue in the alternating manner until such time as either

(i) a dual phase terminates, as above, with an optimal colouring, or

(ii) a primal phase terminates in the standard way by not finding a

feasible colouring using only p_h colours, having previously established a feasible colouring involving $p_h + 1$ colours, (so that $\chi(G) = p_h + 1$), or

(iii) a situation is reached where a primal phase has established a feasible colouring with p_ℓ colours and the following dual phase with $q_k = p_\ell - 1$ does not lead to a feasible colouring; in which case $\chi(G) = p_\ell$.

We see then that should $\chi(G)$ be close to p_1, the bi-directional search would be better than the dual algorithm but worse than the primal, with the situation being reversed if $\chi(G)$ is close to $\omega(G)$; while if $\chi(G)$ is about $(p_1 + \omega(G)) / 2$, the bi-directional search would be less efficient than either of the other two. On average, therefore, one would expect the bi-directional search to be approximately as effective as either of the other two; and indeed, as we shall see in Section 4, the computational results detailed there concur with this reasoning.

2. A DICHOTOMOUS SEARCH ALGORITHM

Before completing this chapter with a further modification using the Intersection Theorem and a discussion of the computational results of the methods given earlier, we first discuss one other algorithm which is described in [3] as an extension of a classical method first mentioned in [9] and which can also be classed as a vertex-by-vertex approach.

2.1 Zykov's Original Method [9]

Let us consider any two non-adjacent vertices, x_1 and x_2, say, of the n-vertex graph $G = (X, \Gamma)$ which is to be coloured. Now any feasible colouring of G must fall into one of two distinct mutually exclusive classes: consisting respectively of those colourings wherein x_1 and x_2 are assigned the same colour (whichever it might be), and those wherein x_1 and x_2 are assigned different colours. As has already been seen, in the former case, G is colour-equivalent* to the $(n-1)$-vertex pseudo-graph, $\tilde{G}$, obtained by coalescing x_1 and x_2 into a single pseudo-vertex $\tilde{x}_1$ which is joined to all vertices of $\Gamma x_1 \cup \Gamma x_2$. In the latter case, as x_1 and x_2 are to be coloured differently, it is then immaterial whether x_1 is or is not adjacent to x_2, so that G is colour-equivalent to the n-vertex graph $G' = (X, \Gamma')$ obtained from G by adding the arc (x_1, x_2). Also, as the two classes above exhaust all the possibilities, we have that

$$\gamma(G) = \min \left(\gamma(\tilde{G}), \gamma(G') \right) \qquad (3)$$

We note here that both $\tilde{G}$ and G' (called the reduced graph of G) are nearer to being complete graphs than is G, since $\Gamma \tilde{x}_1 \supseteq \Gamma x_1$, and $\Gamma' x_1 \supset \Gamma x_1$, $\Gamma' x_2 \supset \Gamma x_2$, respectively.

Next, we consider, in turn, $\tilde{G}$ and G' and, taking any two non-adjacent vertices of that graph, we can re-apply the above procedure to each of G and G', thus obtaining (since minimization is a transitive operation) that

$$\gamma(G) = \min \left\{ \gamma(\tilde{\tilde{G}}), \quad \gamma(\tilde{G}') \quad \gamma(\tilde{G}') \quad \gamma(G'') \right\} \qquad (4)$$

---

* A graph G and a pseudo-graph $\tilde{G}$, derived from G are colour-equivalent if $\gamma(\tilde{G}) = \gamma(G)$.

We then apply the same procedure to each of these new (reduced) graphs, etc., the entire process continuing until there is no remaining reduced graph which contains any pair of non-adjacent vertices, in which case, $\gamma(G)$ will be the minimum chromatic number of any of these final graphs. But the above statement implies that all irreducible graphs are complete graphs, K_p, of p vertices for the various p, so that $\gamma(K_p) = \omega(K_p) = p$. Thus, as concluded in [9], the optimum colouring of G is that one associated with the complete graph of least vertices obtained by the above dichotomous method, the actual colouring being determined by colouring with the same colour those vertices which have been coalesced with one another during the course of the algorithm.*

2.2 Improvements

It is readily seen that the above algorithm is relatively cumbersome in both computing time and storage in that:

(a) many reduced graphs will be generated which will in the final reckoning have no bearing on the determination of the chromatic number; and moreover,

(b) there are likely to be a large number of reduced graphs which are isomorphic**, or nearly isomorphic, to one another thus leading to great replications when each of these graphs is then (individually) reduced.

With a view to minimising these redundancies, there is given, in [3], an improvement of the basic method using the principle of branch

---

* Thus it is seen that the final coalesced vertices basically correspond to the condensation vertices defind in Chapter 4.

** Two graphs are said to be <u>isomorphic</u> if, apart from labels on vertices, they represent the same graph.

and bound [5]. In accordance with this approach, the tree (of reduced graphs) is now generated in a depth-first fashion so that once a reduced graph G_1 is selected for further reduction (by establishing two non-adjacent vertices thereof) no other reduced graphs (except of course those stemming from G_1) will be considered until all graphs stemming from G_1 are complete graphs. After this is accomplished, one would then choose some other reduced graph G_2, etc. Then if at any stage of the algorithm, an irreducible graph is obtained which is a complete graph K_q containing q vertices, it is obvious that q is an upper bound for $\chi(G)$. Thus, if during further branching (i.e. graph-reducing), a graph H encountered which is known to contain a clique of cardinality q - so that none of its irreducible graphs will be complete graphs K_p with $p < q$ - then H cannot lead to a colouring of G requiring less than q colours (the best to date), and so it is not necessary to proceed further to reduce H and a backtracking step may be taken immediately.

In practice, as we have previously seen, it is a non-trivial problem to determine whether H contains a q-clique. Instead, the strategy is in fact to continue to reduce H, but in such a manner as to reduce it to irreducible graphs at as early a stage as possible. This same strategy is of course, useful during all phases of the algorithm in shortening the search, and is accomplished by a suitable selection of the two non-adjacent vertices by means of which the graph H is to be reduced. Thus, one first finds a 'q-cluster' of H; that is a subgraph H_o (say) of H which has q vertices and which contains a high density of arcs. Then two non-adjacent vertices x_1, x_2 are chosen from the vertices of H_o thus generating (from H) the two

reduced graphs H' and $\tilde{H}$; the former by adding the arc (x_1, x_2) and the latter by coalescing x_1 and x_2 into a single vertex. Then the same procedure is applied in depth-first fashion, first to H', for which the q-cluster of H corresponding to H_o again contains q vertices and a higher - by one arc - density of arcs, (and so is a q-cluster of H'), until the q-cluster becomes a q-clique whereupon this branch is terminated. Similarly, this procedure is applied to $\tilde{H}$, for which the subgraph corresponding to H_o (which now contains q-1 vertices), together with some other vertex of $\tilde{H}$ form a q-cluster of $\tilde{H}$.

If q is the actual chromatic number of G, the above procedure will always terminate any branchings by the building of a clique of cardinality at least q. If, on the other hand, q is not the chromatic number (that is, $\chi(G) < q$) then at some stage a reduced graph formed by a coalescence will contain only $q-1$ vertices, so that in such a case one would simply update $q = q - 1$ and continue as before.

This algorithm, which heavily involves recursion, was programmed in ALGOL by the authors of [3], and we shall give, in Section 4, a computational comparison of their results with those of the methods detailed in the previous sections of this chapter. Before doing so, however, we round off this chapter on the vertex-by-vertex colouring approach by showing how the Intersection Theorem of Chapter 4 can be applied to these methods as well.

3. VERTEX-BY-VERTEX COLOURING AND THE INTERSECTION THEOREM

3.1 Preliminaries

We have seen in the previous sections that each of the algorithms in this chapter can be divided into separate and distinct phases, each relating to a different value of the chromatic variable q, the sequences of such values being monotonic. Thus for the primal methods of Section 1 the first phase begins with $n = |X|$ as an upper bound in the chromatic number of $G = (X, \Gamma)$ (i.e. with the assumption that $\gamma(G) = n$) and ends with the establishment of a colouring of G using $q \leq n$ colours. The second, and each succeeding, phase then begins with the attempt to establish a (feasible) colouring of G using $q - 1$ colours* and ends either with such a colouring being successfully found, in which case one updates $q = q - 1$ and proceeds to the next phase; or (it ends) with showing (exhaustively) that no such colouring can be found in which case the algorithm terminates with $\gamma(G) = q$. Correspondingly, for the 'dual' method of section 1.5, the first phase begins by seeking to establish a colouring of G using $q = \omega(G)$ colours and ends either with success in which case the algorithm terminates with $\gamma(G) = q$ or, as before, by showing that no colouring using only q colours is feasible. In this case, the second and (if necessary) each succeeding phase begins by updating $q = q + 1$ and then again seeking to establish a colouring of G using q colours.

In any case, as in the previous chapter, the Intersection Theorem (I.T.) is applicable, at any particular phase, to the 'augmented' graph obtained by appending to G the requisite number of dummy

---

* More exactly, we are attempting to colour G using <u>at most</u> $q - 1$ colours, but as we have seen from Chapter 5, both in practice and as far as the Intersection Theorem is concerned, this is equivalent to the statement in the text.

vertices (as distinct from pseudo-vertices*) this number being $q - \omega(G)$ in the instance of the dual algorithm and $q - \omega(G) - 1$ otherwise. Henceforth in this section we shall discuss the use of the I.T. more fully with particular reference to the primal vertex-by-vertex algorithms of section 1.4 the modifications to the other algorithms being analagous and similar.

3.2 Use of the Intersection Theorem

The actual use made of the I.T. to limit the search when incorporated into the algorithms of this chapter will be somewhat different from that of the previous chapter, in order to accommodate the vertex-by-vertex aspects of the current strategies.

3.2.1 Necessary Conditions

It is apparent that the I.T. holds for graphs whose vertices are uncoloured. Hence it can be easily incorporated into any stage, r say, of a colouring algorithm of the colour-by-colour type. For, once having established (at that stage) the subset S_r^j of vertices that have been coloured in r colours, then extending this partial colouring to a colouring of all the vertices of G using p, say, colours is equivalent, as we have seen earlier, to colouring the uncoloured subgraph $G - S_r^j$ with $(p - r)$ colours, as the first r colours can not be used for any of the vertices of $G - S_r^j$. Thus the I.T. can be applied directly to the graph $G - S_r^j$. With a vertex-by-vertex colouring algorithm, on the other hand, at some stage where T_r, say, represents the set of coloured vertices (with r colours having been used), it is quite possible to use some of these colours again for the remaining uncoloured graph. That is, completing the colouring

---

* Recall that pseudo-vertices effectively replace sets of co-coloured vertices, whereas dummy vertices are extra vertices which are added to a graph specifically in order to increase the cardinality of its maximum clique.

(of T_r) so as to colour the whole of G using a total of p colours is <u>not</u> equivalent to colouring $G - T_r$ using only colours r+ 1, r + 2,...p. Hence, it is not valid to apply the I.T. at this stage to the graph $G - T_r$. Moreover, since part of G is now coloured (so that the I.T. can not be applied to G itself) it is necessary to take recourse to our previously defined concept of pseudo-graphs. The succession of pseudo-graphs corresponding to an optimal colouring, as we have already seen, are each colour-equivalent to the original graph G, with the colour assignments of the appropriate vertices being fixed, so that the effects of these assignments are taken into account. On the other hand, pseudo-vertices can basically be considered as uncoloured (although, of course, pseudo-vertex x_c will eventually be assigned colour c) so that, a pseudo-graph effectively contains no coloured vertices at all. Hence the I.T. can be directly applied to the pseudo-graph obtained at any stage, which may perhaps contain dummy vertices.

One further point must be indicated at this time, namely that the essence of using the I.T. at any phase of the algorithm resides in the fact that, as mentioned in the previous chapter, it is assumed during this phase that $\gamma(G^i) = \omega(G^i)$ for some augmented graph G^i, <u>with $\omega(G^i)$ fixed</u> throughout the phase. However, as we saw in section 1.3, the cardinality of the maximum clique of a pseudo-graph in a sequence may be greater than that of its preceding pseudo-graph, which we have just seen would nullify the I.T. Thus it is to be understood that when using the I.T. on a sequence of pseudo-graphs, the bound (2) is enforced at every stage and a backtracking step immediately taken when the cardinality of the maximum clique of

the pseudo-graph is about to increase (without the addition of a dummy vertex).

3.2.2 The Application of the I.T. at an Arbitrary Stage

We recall that the result of satisfying the I.T., at any stage of the search, is the establishment of an MIS consisting of as-yet-uncoloured vertices which intersects all maximum cliques of the remaining graph (which may perhaps contain dummy vertices). Thus the I.T. is ideally suited for an algorithm of the colour-by-colour type, whereby at each stage one is seeking to determine some MIS, all of whose vertices (and only those) will then be coloured with the same colour. With a vertex-by-vertex algorithm, on the other hand, the establishment of this set, M say, of vertices, all of which are to be coloured with the same colour, c say, necessitates some modification of the general algorithm to ensure that all those, and only those, uncoloured vertices which are in M are to be coloured c.

Once a set M is established, therefore, it is useful to utilize the vertex-reordering version of the basic algorithm. In this case, one merely arranged that the vertices of M are the next m vertices to be assigned a colour (where $m = |M|$), and then places markers to indicate that no other colours be feasible for these vertices until (at a backtracking step) the intersection condition leads to the choice of a new MIS, M', which also intersects all maximum cliques. This new choice M' then leads to a new ordering of the vertices and the algorithm again proceeds in the above way.

In fact, as is easily shown, the above arrangement of the vertices then immediately serves to satisfy the 'only those' part of the condition above. For, once the m vertices of M have been

assigned colour c, then by the terms of the vertex-to-vertex algorithm, any as-yet-uncoloured vertex x_k for which c is a feasible assignment would necessarily be non-adjacent to any vertex already assigned c. But that would imply that $M \cup \{x_k\}$ is an independent set (of the appropriate graph), contradicting the assumption that M is a maximal independent set of the same graph.

3.3 Practical Considerations

We have seen in the previous section that implementing the Intersection Theorem necessitates some alteration of the search strategy as well as the continual recording (be it implicit or explicit) of pseudo-graphs, together with the calculation of the cliques and MIS of these pseudo-graphs. Again, as was the case for ordinary graphs, the cliques and MIS of one pseudo-graph can be easily determined from that of a preceding pseudo-graph by having recourse to the appropriate results of Chapter 3; so that, recalling that the pseudo-graph corresponding to the first stage (of the first phase) is the graph G itself, there is no need for any further additional invoking of a clique / MIS procedure.

Thus, it is the alteration of the search strategy that is the determining factor for the application of the I.T.; indeed, if the I.T. is used to its full at every opportunity, the search strategy will degenerate to that of a colour-by-colour algorithm. It is suggested therefore that the I.T. be applied only when there are but few possibilities of satisfying its intersection conditions.

We describe now how the I.T. can be used, within the framework of the vertex-by-vertex algorithm, either as an efficient look-ahead procedure, or, as before, as a method of directing the search.

3.3.1 Use of the I.T. as a Look-ahead Bound

Let us consider a stage where $\tilde{G}'$ represents the augmented pseudo-graph obtained at that stage. Now, by the I.T., any set of vertices of $\tilde{G}'$ which are to be coloured the same must intersect all maximum cliques of $\tilde{G}'$. In particular any vertex which is to be coloured the same as a particular pseudo-vertex, $\tilde{x}_c$, of G' must be in some independent set which contains $\tilde{x}_c$ and which intersects all maximum cliques of $\tilde{G}'$. Therefore, letting $M_1, M_2, \ldots, M_s$ be all the MIS of $\tilde{G}'$ which contain $\tilde{x}_c$ and satisfy the intersection condition, any uncoloured vertex x_i for which $x_i \notin \bigcup_{i=1}^{s} M_i$ cannot be (optimally) co-coloured with $\tilde{x}_c$ in $\tilde{G}'$. In other words, assigning colour c to x_i cannot lead to the completion of the current partial colouring to obtain an optimal colouring during this phase of the search. Since the set of pseudo-vertices of $\tilde{G}'$ corresponds to the set of all colours used to date, by adopting a similar strategy for each pseudo-vertex, we can then effectively eliminate immediately some (feasible) colour assignments from forward branching at this stage, thereby reducing the search.

3.3.2 Use of the I.T. to Direct the Search

(i) Two Simple Situations

Consider then a stage where n_d vertices have been coloured and $\tilde{G}'$ represents the resultant augmented pseudo-graph. If there is some vertex of $\tilde{G}'$, say x_o, which is not contained in any MIS of $\tilde{G}'$ which satisfies the intersection condition, then, as was shown in the previous chapter, there is no possibility of

completing the current partial colouring to obtain a feasible colouring of G using at most the requisite number of colours, and therefore a backtracking step may be taken immediately. *

Or if there is some vertex, x_r, of $\tilde{G}'$ which is contained in exactly one MIS, M, of $\tilde{G}'$ which satisfies the intersection condition then, again as was shown earlier, by choosing x_r to be coloured next we know that the only possibility of obtaining a feasible completion of the current partial solution is to colour all the vertices of M, and only those, with the same colour. (If oneof these vertices is a pseudo-vertex then of course all its constituent vertices will be coloured with its colour). ** In this case, too, it is obviously worthwhile to make use of this fact in the manner indicated in the previous section, i.e. by colouring next the vertices of M, and recording the corresponding markers; the actual colour, c, which is assigned to these vertices then being either

(i) that of a pseudo-vertex, should M contain some such vertex, or

(ii) the lowest colour not yet used, if M contains no pseudo-vertex.

The stage is then obtained by coalescing the set M into the next pseudo-vertex $\tilde{x}_c$ so that we are about to colour the $(n_d + 1 + |M|)^{th}$ vertex of G. And, later on in the search, when backtracking from the stage representing $(n_d + |M| + 1)$ coloured vertices, we can immediately backtrack to the stage with n_d vertices coloured.

---

* Although theoretically x_o may be a pseudo-vertex, it is in fact unlikely, (due to the relatively high degree of a pseudo-vertex) that any pseudo-vertex will be the only such vertex. In any case, the analysis holds for pseudo-vertices as well.

** It is to be noted that, since all pseudo-vertices are, by definition, adjacent to each other, no two of them can be in any independent set, thus establishing the soundness of the statement.

(ii) The General Case

There remains but to consider the more general situtation where every vertex of $\tilde{G}'$ is contained in more than one MIS satisfying the I.T.; and let us assume the vertex belonging to the fewest such MIS, call it x_r, is contained in $M_1, \ldots, M_t$. Again, any possible feasible completion must involve the colouring of the vertices in some M_i, $1 \leq i \leq t$, but there is no way of predetermining which M_i (if any) leads to a feasible completion. Therefore, to correctly exhaust the branching possibilities, one would have to branch forward using some M_k; $1 \leq k \leq t$; then when the current stage is reached again after a backtracking step, one would have to select some other M_ℓ, $1 \leq \ell \leq t$ $\ell \neq k$, and branch forward, etc., until all the MIS M_i, $1 \leq i \leq t$, have been used. As mentioned above, such a strategy would tend to turn the algorithm into one of the set-colouring variety.

We suggest therefore that the full power of the I.T. (as given above) is used only where t is small e.g. $t \leq 3$ *. For $t > 3$, it is still possible to gain some advantage by using the I.T. in a slightly weaker manner, while at the same time avoiding some of of the disadvantages mentioned above, as follows.

By the way the M_i are defined, we know that $x_r \in M_i$ $i = 1,2,\ldots t$ so that $x_r \in \bigcap_{i=1}^{t} M_i$ Now, as was noted above, if x_r were selected as the next vertex to be coloured, it must be co-coloured with the other vertices of some M_i, $1 \leq i \leq t$. If in addition we have that

* The cases $t = 0$ and $t = 1$ are of course the simple cases described in the previous subsection.

$\left| \bigcap_{i=1}^{t} M_i \right| \geq 2$, then in each such continuation, x_r will certainly be co-coloured with the other vertices of $\bigcap_{i=1}^{t} M_i$

Thus by in fact selecting x_r as the next vertex, we can immediately assign the colour, c say, chosen for x_r to <u>all</u> the vertices of $\bigcap_{i=1}^{t} M_i$, so that again there need be no intermediate backtrackings between the last of these vertices and the vertex x_r. We note parenthetically here that in such a case, it is still possible that some other future vertex will also be assigned colour c.

Alternatively, it is possible to go one step better and in fact <u>select</u> x_r to be that uncoloured vertex for which $\left| \bigcap_{i=1}^{t} M_i \right|$ is maximal.

It will very often occur, however that $\bigcap_{i=1}^{t} M_i = \{x_r\}$, (i.e. $\bigcap_{j-1}^{t} M_i = 1$) so that the above strategies are ineffective. In such a case one would not invoke the I.T. to direct the search, and either use it only as a look-ahead or proceed as in the original algorithm until such time as one of the previous situations occur.

3.4 <u>The I.T. and a Lower Bound</u>

Finally, we mention that the improved bound based on the I.T., discussed in the previous chapter, (section 3.4) is equally applicable for the methods of this chapter. Thus it too can be incorporated into any of the vertex-by-vertex algorithms described earlier.

4. COMPUTATIONAL RESULTS

In this Section, we confirm - from Table 8 - that, as mentioned in earlier sections of this chapter, that the dynamic vertex re-ordering strategy is a major improvement with regard to the vertex-by-vertex colouring algorithm and also that the primal, dual and bidirectional versions are generally computationally equivalent. We also give the results of Corneil and Graham [3] for their version of the Zykov algorithm [9].

TABLE 8 VERTEX-BY-VERTEX COLOURING

VERTICES	DENSITY	TIMES (CDC 6400 seconds)					
		A	B	C	D	E	F
30	.1	.2	.1	.2	.2	.2	
	.3	1.2	.1	.2	.3	.6	9.7
	.5	2.9	.2	.3	.2	.6	
	.7	4.5	.2	.2	.4	1.4	8.5
	.9	.6	.2	.2	.3	.3	
35	.1	.4	.1	.4	.4	.5	
	.3	3.5	.5	.4	.4	1.0	127.8
	.5	10.4	1.3	1.6	1.1	3.1	
	.7	31.2	1.6	2.1	1.6	6.8	20.0
	.9	.4	.3	.3	.3	.6	
40	.1	.5	.4	.4	.3	.3	
	.3	27.5	.7	.7	.7	1.5	
	.5	50.8	2.4	2.9	2.3	14.3	
	.7	71.4	3.4	4.8	3.8	20.6	78.2
	.9	.7	.6	.7	.6	1.6	
45	.1	.9	.6	.4	.6	.5	
	.3	87.1	1.1	1.0	1.2	3.0	
	.5	120.5	2.6	3.1	2.7	37.8	
	.7	180.8	3.2	12.1	4.6	74.2	
	.9	.8	1.0	1.0	.9	7.9	
50	.1		.7	.7	.7	.7	
	.3		7.3	10.4	7.6	21.3	
	.5		59.0		62.0		
	.7		64.4		72.1		
	.9		2.5				

A Basic Vertex-by-Vertex Colouring Algorithm

B As A, but with Dynamic Vertex Re-ordering

C Dual version of B

D B and C combined in bidirectional search strategy

E Dynamic Vertex Re-ordering version of Brown's method [1]

F Corneil and Graham's version [3] of Zykov's Algorithm [9] Note that these latter terms are in IBM 370/165 seconds and that the algorithm was coded in ALGOL.

5. REFERENCES

1. BROWN, J.R. "Chromatic Scheduling and the Chromatic Number Problem", Management Science 19 (1972), pp 456, 463.

2. CHRISTOFIDES, N. Graph Theory, In Press.

3. CORNEIL, D.G. and GRAHAM, B. "An Algorithm for Determining the Chromatic Number of a Graph", SIAM Journal of Computing 2 (1973), pp 311-318.

4. GILLIAN, R.J. "The Chromatic Number of a Graph", M.Sc Thesis, Department of Management Science, Imperial College, London.

5. LITTLE, J.D.C., MURTY, K.G. SWEENEY, D.W. and KAREL, C., "An Algorithm for the Travelling Salesman Problem", Operations Research 11 (1963), pp 972-989.

6. MATULA, D.W. MARBLE, G. and ISAACSON, J.D. "Graph Colouring Algorithms" in Graph Theory and Computing (READ ed) Academic Press, N.Y., 1974.

7. WELSH, D.J.A. and POWELL, M.B. "An Upper Bound on the Chromatic Number of a Graph and its Applications to Time-tabling Problems", Computer Journal 10 (1967), pp 85-86.

8. WOOD, D.C. "A System for Computing University Examinations Timetables", Computer Journal 10 (1967), pp 139-141.

9. ZYKOV, A.A. "On Some Properties of Linear Complexes", Amer. Math Soc. Translation 79, (1952).

CHAPTER VII

CONCLUSIONS

1. GENERAL SUMMARY

In this thesis we have considered, in the main, the Graph Colouring Problem (GCP); and also with two problems which are related to the GCP, viz. the Set Covering Problem (SCP) and the Clique-Generating Problem. Except where otherwise implied, generally by quoting the appropriate reference, all the preceding work is, as far as we are aware, original to the author; as are all the computer programs on which the computational results are based (again with the exception of explicitly quoted results).

The main contributions of this thesis are in the development of new tree-search algorithms for the solution of the GCP. To this effect, we have established some new theorems related to optimal graph colouring, in particular, the Intersection Theorem (I.T.). Using the Intersection Theorem, we have then derived an original depth-first tree-search algorithm - the Multiple Algorithm - of the type which determines, at each stage, the set of vertices to which will be assigned the next colour. The Multiple Algorithm has the unusual, but very useful, property that the first feasible solution obtained is known to be optimal which, together with the fact that the I.T. provides a strong bounding condition throughout the search, ensures that this algorithm is far more efficient that any previously-known algorithm of the colour-sequential variety.

We have also seen that the Multiple Algorithm is most effective for graphs G whose chromatic number is near its clique number, a situation which is, indeed, often true for practical colouring problems, as noted by Williams [I-22] Again, we have shown that, for perfect graphs, a sub-class of the colour of graphs for which the chromatic number is equal to the clique number, the algorithm becomes a once-through affair. Lastly, we have discussed a dual variant of the Multiple Algorithm and

indicated a bidirectional search strategy incorporating both the primal dual variants.

Turning to the vertex-sequential tree-search colouring algorithm, we have introduced the idea of dynamic vertex-reordering during the course of the algorithm, and shown the computational benefit of such a strategy; and again we have indicated dual and bidirectional variants of this algorithm. Lastly, by defining the concept of pseudo-graphs we have shown how the I.T. could be incorporated into the vertex-sequential algorithm as a bounding (and branching)strategy.

We noted that the GCP can be formulated as an SCP and in addition to a computational comparison of several known algorithms, we have developed a tree-search algorithm for the SCP which includes an (incomplete) dynamic programming strategy to limit the search and which incorporates a new bound. Finally this algorithm was shown to compare favourably with the other methods coded.

We have also considered the problems of maximal independent-set-generation on which the colour-sequential algorithms are based and of clique-generation, on which the I.T. strategies are based, and and seen that these were equivalent. As well as comparing, computationally, several established clique-generating algorithms we have shown how cliques and maximal independent sets of pseudo-graphs can be obtained from cliques and maximal independent sets of related graphs, and vice versa, thus enabling the efficient use of the I.T. in the vertex-sequential tree searches mentioned above.

In summary, of all the algorithms for the solution of graph-colouring the vertex-sequential algorithm with dynamic vertex reordering gave the best computational results, while of the methods of the colour-sequential variety - which is the more graph-theoretical, and which can have

intrinsic advantages in some applications - , the Multiple Algorithm is the most efficient. Lastly, as is to be expected, (0-1) Integer Programming algorithms of a more general type are not so useful in solving graph-colouring problems.

Finally, we have seen earlier that as well as leading to graph-colouring algorithms, the theoretical results are of interest in their own right. In the following section, we shall briefly indicate some other potential consequences of the earlier theorems.

2. CONJECTURES OLD AND NEW

In this section we indicate how the results of Chapter 4 can possibly be used in relation to two famous conjectures of Graph Theory, viz. The Four Colour Conjecture - stating that every planar graph G is 4-colourable (see Chapter 1), - and the Strong Perfect Graph Conjecture - stating that a graph G is perfect if and only if neither it nor its complement contains an odd cycle as an induced subgraph.

2.1 The Four Colour Conjecture

It is an obvious corollary from a well-known theorem of Kuratowski that $\omega(G) < 5$ for any planar graph G. Moreover, since the clique, K_4, of cardinality 4 is easily seen to be planar it follows that the graph G' formed by 'hinging' K_4 to G via any 'outside' vertex, $\bar{x}$, of G is a planar graph. Furthermore, since $\{\bar{x}\}$ is an articulating set, the results of Section 1-4, indicate that the colouring of K_4 has no effect on the colouring of G so that

$$\gamma(G \cup K_4) = \max(\gamma(G), \gamma(K_4)) = \max(\gamma(G), 4).$$

Hence to prove that $\gamma(G) \leq 4$ it suffices to prove that $\gamma(G \cup K_4) \leq 4$, so that it is apparent that the following statement is equivalent to the Four Colour Conjecture: A planar graph G which contains a clique of cardinality 4 is 4-chromatic, i.e. $\gamma(G) = \omega(G)$.

However, as can be seen from Corollary 3 of Theorem 4-3, to show that $\gamma(G') = \omega(G')$ for any graph G', it suffices to determine $\omega(G')$ consecutive coloured sets S_j, where S_1 is an MIS of G' which intersects all maximum cliques of G', S_2 is an MIS of $G' - S_1$ which intersects all maximum cliques of $G' - S_1$ etc.

We thus obtain the following restatement of the Four Colour Conjecture - involving cliques and MIS of graphs - which may perhaps be more tractable to prove (or indeed disprove):

"For any planar graph G with $\omega(G) = 4$ one can find MIS S_j such that S_1 intersects all 4-cliques of G, S_2 intersects all 3-cliques of $G - S_1$, S_3 intersects all 2-cliques of $G - (S_1 \cup S_2)$, while $G - (S_1 \cup S_2 \cup S_3)$ is an independent set."

2.2 The Strong Perfect Graph Conjecture

It can be seen from Section 4-2 that an equivalent statement to the Strong Perfect Graph Theorem is that every p-critical subgraph is either an odd cycle or its complement. We also showed there that every p-critical graph $G = (X, \Gamma)$ contains at least n maximum MIS and at least n maximum cliques, where $n = |X|$.

Let us now write $N_\alpha(G)$ = number of maximum MIS of G

$N_\omega(G)$ = number of maximum cliques of G.

The following original conjecture - which in fact holds true for all the random graphs produced for the algorithmic comparisons of

this thesis - seems very likely to be true.

α-ω Conjecture For any graph $G = (X, \Gamma)$ we have

$$\min\ (N_\alpha(G), N_\omega(G)) \leq n, \quad \text{where } n = |X|$$

Again, from Corollaries 1 and 2 of Theorem 4-6 we know that for any p-critical graph $G_p = (X_p, \Gamma_p)$, $\min\ (N_\alpha(G_p), N_\omega(G_p)) \geq n$ which, in view of the conjecture would imply $N_\alpha(G_p) = N_\omega(G_p) = n$. Now it is readily seen that for a graph G which is an odd cycle, $N_\alpha(G) = n$, and also that $N_\omega(G) = N_\alpha(\bar{G}) = n$, and it seems natural to suppose that the only graphs for which $\min\ (N_\alpha(G), N_\omega(G)) = n$ are the odd cycles (and their complements).

Hence a proof of the above statement together with a proof of the α-ω Conjecture would settle affirmatively the Strong Perfect Graph Conjecture, so that we have transformed this conjecture as well into a form relating to the cliques and MIS of graphs.