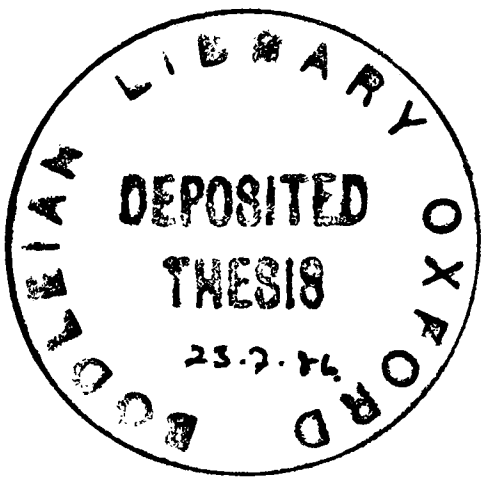
TOPICS IN COMPUTATIONAL COMPLEXITY

Graham Ernest Farr,
Jesus College, Oxford.



Thesis submitted for the degree of Doctor of Philosophy at the University of Oxford.

April, 1986.

TOPICS IN COMPUTATIONAL COMPLEXITY

Graham Farr,
Jesus College, Oxford.

Thesis submitted for the degree of Doctor of Philosophy,
Hilary Term, 1986.

ABSTRACT

This thesis concerns the computational complexity of several combinatorial problems.

Chapter 1 is notation and definitions. In Chapter 2 the subgraph homeomorphism problem, for fixed pattern graphs H, is considered. For the cases $H \cong C_6, C_7, W_4$ or W_5, we obtain exact characterizations of graphs with no subgraph homeomorphic from H, and derive efficient algorithms for recognizing such graphs. We then consider counting homeomorphs, and show that if $\mathcal{H}$ is any finite nontrivial family of graphs then the problem of determining the number of subgraphs of a graph which are homeomorphic from a member of $\mathcal{H}$ is $\#$P-complete.

In Chapter 3 we consider languages in NP whose certificate size is bounded by a fixed, slowly growing function (say $f(n)$) of the input size. The classes $f(n)$-NP are defined in order to classify such languages; this is related to work of Kintala and Fischer. We show that several natural problems, involving Boolean satisfiability, graph colouring and Hamiltonian circuits are complete for $f(n)$-NP, and obtain in passing some new languages which are logspace complete for P.

Multicolouring is a natural generalization of graph colouring. We prove in Chapter 4 that determining whether a graph is (r,s)-colourable is NP-complete whenever $s > 2r$. This extends a result of Irving concerning the case $s = 2r + 1$. We also consider the complexity of some graph homomorphism problems.

In Chapter 5 we first give a polynomial time algorithm for 3-colouring graphs G whose minimum degree is $> \frac{8}{15}|V(G)|$. We then consider separability of disjoint pairs of languages, and obtain NP-hardness results for certain promise problems involving colouring.

The final Chapter concerns a problem of partitioning graphs subject to certain restrictions. We prove that several subproblems are NP-complete.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dominic Welsh, for his guidance, patience and encouragement. My work has also benefited from discussions with a number of other people, particularly Ken Regan, Keith Edwards and Colin McDiarmid.

I am grateful for financial support in the form of a Commonwealth Scholarship, and thank the various bodies responsible for this: the Commonwealth Scholarship Commission, the Association of Commonwealth Universities and the British Council. After expiry of this Scholarship I received support for one term from Jesus College, Oxford, which I gratefully acknowledge.

The support I have received from my wife Ruth has been enormous, and I am deeply grateful for it. In addition, her very careful proofreading leaves me no excuse for errors.

Finally I thank Jane Cox for her excellent and rapid typing of this thesis.

CONTENTS

INTRODUCTION

This thesis is concerned with several distinct topics in computational complexity.

The basic definitions and notation are introduced in Chapter 1.

The subgraph homeomorphism problem (SHP) is the problem of deciding, for any graph G, whether G has a subgraph homeomorphic from a fixed graph H. Its complexity remained open [18, p.285; 37] until recently, when Robertson and Seymour (see [47]) proved that the problem has a polynomial time algorithm for each fixed H. Their algorithms are extremely general, and leave us with little understanding of the exact structure of graphs with a specific homeomorph excluded. In Chapter 2 we consider the cases $H \cong C_6, C_7, W_4$ and W_5 and in each case obtain an exact characterization of graphs with no H-homeomorph. The most important result here is that any 3-connected graph satisfying a simple edge connectivity condition has a W_5-homeomorph iff it has a vertex v of degree at least 5 and a circuit, disjoint from v, of length at least 5. Our characterizations enable us to obtain efficient algorithms for the above cases of the SHP. We then consider the corresponding enumeration problems, and prove that if H is any finite nontrivial family of graphs, then the problem of determining the number of subgraphs of a graph which are homeomorphic from a member of H is $\#$P-complete (see [15]). This is strong evidence that these enumeration problems are all intractable, in contrast with the polynomial time solvability of the decision problems.

(§2.4 is joint work with Colin McDiarmid.)

Chapter 3 concerns the complexity of problems in NP whose certificates are short in the sense that their size is bounded by a slowly growing function of the input size. For example, we may ask about the complexity of deciding whether a graph G on n vertices has a clique of size $f(n)$, where f is some function. We are interested in how the complexity of such problems depends on f. We define the class $f(n)$-NP, which is related to the class $P_{f(n)}$ of Kintala and Fischer [32, 33], and completeness for $f(n)$-NP. Several natural combinatorial problems are shown to be $f(n)$-NP-complete. These all involve deciding whether a structure has a partial function of a certain kind (e.g. a partial truth assignment, or a partial 3-colouring), whose domain may or may not be specified, which can be extended to the whole structure by a simple procedure known as "forcing". When the domain of the partial function is specified in the problem input, we use nontrivial modifications of standard transformations. When the domain is unspecified, a new technique is used. As special cases of these results we obtain some new logspace completeness results for P. It seems likely (subject to $P \neq NP$) that problems which are $f(n)$-NP-complete (for certain suitable f) are neither in P nor NP-complete, however this is likely to be extremely difficult to prove. We discuss these and other issues fully in Chapter 3.

Multicolouring is a natural generalization of graph colouring and was introduced by Hilton, Rado and Scott [26]. Its complexity was first considered by Irving [27] who showed

that determining whether a graph is (r,s)-colourable is NP-complete if $s = 2r + 1$. We extend this result considerably in Chapter 4 by showing that the problem is NP-complete whenever $s > 2r$. In the final section some related questions involving graph homomorphism are considered, and we obtain a simple sufficient condition on a graph H under which the problem of determining whether a graph is homomorphic to H is NP-complete.

Chapter 5 concerns some graph colouring promise problems. In §1 the promise is simply a lower bound on vertex degree, and we give a polynomial time algorithm for finding a 3-colouring, when one exists, of any graph G with minimum degree greater than $\frac{8}{15}\cdot|V(G)|$. §§2 and 3 concern promise problems whose promise is an NP-hard condition on the chromatic number. In §2 we study promise problems implicitly via separability of disjoint pairs of languages, which is a more convenient concept for our purposes. We establish a number of elementary results, and in §3 we use some of these concepts to show that certain colouring promise problems are NP-hard.

The final Chapter is a brief study of the complexity of a problem of partitioning graphs subject to certain restrictions. This is a practical problem: we discuss its origins in §1. We show that several very restricted subproblems are NP-complete.

(1.1)

CHAPTER 1

PRELIMINARIES

This chapter is concerned with some elementary definitions. We consider in turn the fields of graph theory and computational complexity.

Note that all logarithms in this thesis are to base 2 unless otherwise indicated.

§1. Graphs

We state some definitions which we will need, and establish some notation. Undefined terms follow standard usage (see e.g. [25]).

All graphs in this thesis are undirected with no loops or multiple edges, except where otherwise stated.

If G is a graph we denote its vertex set by $V(G)$ and its edge set by $E(G)$. We usually denote an edge just by juxtaposition of its vertices; thus an edge $\{v,w\}$ joining vertices v,w is writen vw. If vertices v and w are adjacent we write $v \simeq w$, and if not we write $v \not\simeq w$. The neighbourhood $N_G(v)$ of vertex v in G is the set of vertices which are adjacent to v in G. We say the graph $G' = (V',E')$ is a subgraph of G, and write $G' \leq G$, if $V' \subseteq V(G)$, $E' \subseteq E(G)$, and every edge in E' joins two vertices of V'. If $V' \subseteq V(G)$ then the subgraph of G induced by V' is (V',E') where $E' = \{vw \in E(G) \mid v,w \in V'\}$ and is denoted by $\langle V' \rangle$; an

induced subgraph is a subgraph so formed. If $W \subseteq V(G)$ we denote by $G - W$ the graph $\langle V(G) \setminus W \rangle$.

A walk in G is a sequence of vertices in which every two consecutive vertices are adjacent. The first and last vertices of a walk are its endpoints, and a walk is closed if its endpoints are identical. A path is a walk in which no vertex occurs more than once; a circuit is a path whose endpoints are identical.

A path (or circuit) in G defines a subgraph of G in the natural way: its vertices are the terms of the sequence constituting the path (or circuit), and its edges are all pairs of vertices which are consecutive in the sequence. We identify a path (considered as a sequence) with the graph so formed, and do the same for a circuit. The length of a path or circuit is the number of edges it has. For every $k \geq 0$, P_k denotes the (unique, up to isomorphism) path of length k and C_k the circuit of length k. If a path P has one endpoint in $V_1 \subseteq V(G)$ and the other endpoint in $V_2 \subseteq V(G)$, we say P is a $V_1 - V_2$ path (equivalently, a $V_2 - V_1$ path). If either of V_1 or V_2 is a singleton here, we drop its brackets: thus a $v - U$ path is a path from v to some member of U, and a $v - w$ path is a path with endpoints v and w. A vertex of a path P is internal if it is not an endpoint of P. If P is any path and $v,w \in V(P)$ then we use interval notation, as shown in the table of Figure 1, to describe various subpaths of P. The table gives full details. (The empty "path" with no vertices may result in some cases.) The distance between vertices v and w in G, denoted by $d_G(v,w)$, is the length of the shortest $v - w$ path in G.

(1.1)

Symbol.	Symbol denotes portion of P between v and w ...
P(v,w)	... excluding v and w.
P(v,w]	... excluding v, including w.
P[v,w)	... including v, excluding w.
P[v,w]	... including v and w.

Figure 1. Table giving the interval notation for subpaths.

We will be making use of certain special graphs. C_k^* is the multigraph consisting of two vertices joined by k parallel edges. W_k is the k-wheel, formed by taking a copy of C_k and adding a new vertex joined to every vertex of C_k. A graph is a null graph if it has no edges.

Let G_1 and G_2 be any graphs. The sum $G_1 + G_2$ is formed by taking disjoint copies of G_1 and G_2 and joining every vertex of G_1 to every vertex of G_2. The composition $G_1[G_2]$ is formed as follows. Take $|V(G_1)|$ disjoint copies of G_2, indexed by $V(G_1)$; let them be $G_2^{(v)}$, $v \in V(G_1)$. Then add additional edges as follows. For every $\{v,w\} \in E(G_1)$, add edges joining every vertex of $G_2^{(v)}$ to every vertex of $G_2^{(w)}$. For every $\{v,w\} \notin E(G_1)$, there are no edges joining any vertex of $G_2^{(v)}$ to any vertex of $G_2^{(w)}$.

We say that $W \subseteq V(G)$ is a separating set of G, and that W separates G, if $G - W$ is disconnected. If $\{w\}$ separates G then w is a cutvertex of G. If G has a separating set of size k then G is k-separable. Thus, G is k-separable iff G is not $(k+1)$-connected. If W separates G then the

(1.1)

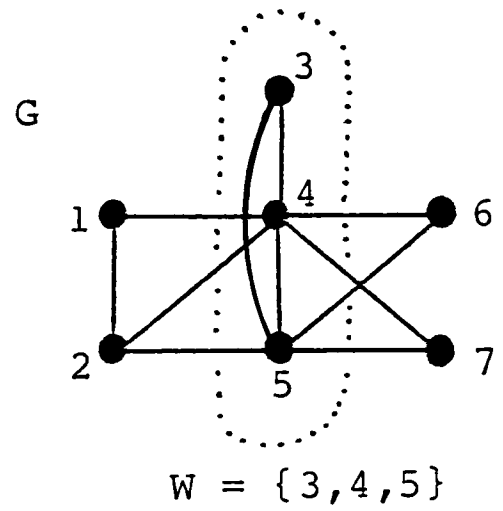
separation (of G by W), written G|W, is the set of all subsets U of G satisfying

(i) Any two vertices of U are joined by a path in G which does not meet W, except at those of its end-points, if any, which lie in W;

and (ii) U is maximal with respect to (i).

We will refer to the elements of G|W as its parts. A k-separation is a separation G|W in which $|W| \leq k$. It is easily seen that for any separation G|W of a graph G, every vertex of G lies in a part of G|W. Furthermore, the sets of the form $U \backslash W$, where U is a part of G|W, partition $V(G) \backslash W$.

An example illustrating the above definitions is given in Figure 2.



Parts of G|W:

{1,2,4,5};

{4,5,6};

{4,5,7};

{3,4,5}.

Figure 2. A graph G, and the parts of the 2-separation G|W.

The concept of clique-summing was introduced by Seymour [60] for binary matroids (although Wagner [67] had earlier defined k-sums, for $k \leq 3$, for graphs). For graph-theoretic purposes this may be defined as follows. Suppose G_1 and G_2 are graphs. A graph G is said to be a k-sum of G_1 and G_2 (where $k \in \mathbb{N}$) if G can be formed from G_1 and G_2 by taking a k-clique in each of G_1 and G_2, identifying these two k-cliques into a single k-clique K, and deleting any desired number (including

(1.1)

all or none) of the edges of the k-clique K. G is a clique-sum of G_1 and G_2 if it is a k-sum of G_1 and G_2 for some k. None of these clique-sums are to be confused with the sum $G_1 + G_2$ defined earlier.

It is clear that a graph has a k-separation iff it is a k'-sum for some $k' \leq k$.

Throughout this thesis, unless stated otherwise, G denotes a graph with vertex set V and edge set E.

(1.2)

§2. Computational complexity

We briefly review some of the basic concepts of complexity theory. For undefined terms we refer the reader to Garey and Johnson [18], whose notation and terminology we generally follow.

If Σ is an alphabet (that is, a finite set of symbols), then Σ^* denotes the set of all finite strings of members of Σ. If x and y are strings in Σ^* then xy denotes the concatenation of x and y, and if $\sigma \in \Sigma$ and $n \in \mathbb{N}$ then σ^n is the string consisting of n σ's. Throughout this thesis, Σ denotes an alphabet with at least two symbols. For convenience we assume $\{0,1\} \subseteq \Sigma$.

A <u>language</u> is a subset of Σ^*; in defining a language we will frequently regard it as the set of accepted instances of a decision problem and use the Input/Question format of [18]. If L is a language we write L^c for the <u>complement</u> $\Sigma^* \backslash L$ of L. We abbreviate "deterministic Turing machine" (respectively, "nondeterministic Turing machine", "oracle Turing machine", "nondeterministic oracle Turing machine") to DTM (NDTM, OTM, NOTM). A number of complexity classes will be encountered in this thesis. The main ones are as follows.

P is the class of languages recognized by polynomial time DTMs;
NP is the class of languages recognized by polynomial time NDTMs;
LOGSPACE (sometimes abbreviated to L) is the class of languages recognized by logspace DTMs;
NL is the class of languages recognized by logspace NDTMs;
PSPACE is the class of languages recognized by polynomial space DTMs;

If $S(n)$ is any function, then DSPACE $(S(n))$ (resp. NSPACE $(S(n))$) is the class of languages recognized by DTMs (NDTMs) whose space complexity is bounded above by $S(n)$;

If $A \subseteq \Sigma^*$ then P^A (resp. NP^A) is the class of languages which can be recognized by a polynomial time OTM (NOTM) with an oracle for A.

If M is a DTM or NDTM, then $L(M)$ denotes the language recognized by M.

If L_1 and L_2 are languages over alphabets Σ_1 and Σ_2 respectively, and $f : \Sigma_1^* \to \Sigma_2^*$ is computable in polynomial time, then f is a <u>polynomial transformation</u> from L_1 to L_2 if, for all $x \in \Sigma_1^*$, $x \in L_1 \iff f(x) \in L_2$. If there exists such a transformation we write $L_1 \propto L_2$. If f is logspace computable then we say f is a <u>logspace transformation</u> and write $L_1 \propto_{LOG} L_2$. L_1 and L_2 are <u>polynomial time equivalent</u>, written $L_1 \equiv_p L_2$, if $L_1 \propto L_2$ and $L_2 \propto L_1$. Further we say L_1 is <u>(polynomial time) Turing reducible</u> to L_2, written $L_1 \propto_T L_2$, if there is a polynomial time OTM with an oracle for L_2 which recognizes L_1. It is well known that

$$L_1 \propto_{LOG} L_2 \Rightarrow L_1 \propto L_2 \Rightarrow L_1 \propto_T L_2.$$

If $L_1 \propto_T L_2$ and $L_2 \propto_T L_1$ then L_1 and L_2 are <u>(polynomial time) Turing equivalent</u>, written $L_1 \equiv_T L_2$. L is <u>NP-complete</u> if $L \in NP$ and $L' \propto L$ for all $L' \in NP$; the class of NP-complete languages is denoted by NPC. Among the many known NP-complete languages are the following, which we will use later.

SATISFIABILITY (SAT)

Input: Boolean expression ϕ in conjunctive normal form (c.n.f.).

Question: Is ϕ satisfiable?

(1.2)

3SAT

Input: Boolean expression ϕ in c.n.f. with exactly 3 literals in each clause.

Question: Is ϕ satisfiable?

HAMILTONIAN CIRCUIT (HAM)

Input: Graph G.

Question: Does G have a Hamiltonian circuit?

k-COLOURABILITY (kCOL) (where k is a fixed integer and $k \geq 3$)

Input: Graph G.

Question: Is G k-colourable?

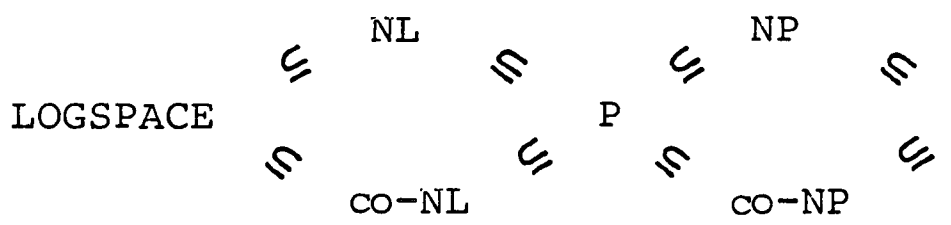
CLIQUE

Input: Graph G, integer k.

Question: Does G have a clique of size k?

L is <u>NP-hard</u> if $L' \propto_T L$ for some $L' \in NPC$. We define co-NP = $\{L | L^C \in NP\}$. The language L is <u>co-NP-complete</u> if L^C is NP-complete, and co-NPC = $\{L | L^C \in NPC\}$. Also co-NL = $\{L | L^C \in NL\}$. If $C \subseteq 2^{\Sigma^*}$, as is the case if C is any of the complexity classes we consider, then we say L is <u>logspace complete for C</u> (resp. <u>Turing complete for C</u>) if $L \in C$ and $L' \propto_{LOG} L$ ($L' \propto_T L$) for all $L' \in C$.

Among the containment relationships known to exist between the classes so far defined are the following.



All these inclusions are believed to be proper, but all that is known is that LOGSPACE $\neq$ PSPACE.

Having defined some classes of languages, we now define some classes of functions. When defining such functions we again frequently give them as problems, in Input/Output format.

FP denotes the class of all functions computable by polynomial time DTMs and FL the class of all functions computable by logspace DTMs. If M is an NDTM with input alphabet Σ, then we define a function $[M] : \Sigma^* \to \{0,1\}^*$ where $[M](x)$ equals the number (in binary) of accepting computations of M on input x. We regard $[M]$ as an enumeration problem. Indeed it seems fair to say that any problem which can sensibly be called an enumeration problem can be thought of as representable in the form $[M]$ for some NDTM M. The class of functions $\#P$, introduced by Valiant [65] to help classify enumeration problems according to their complexity, is defined by

$$\#P = \{[M] \mid M \text{ is a polynomial time NDTM}\}.$$

Thus $\#P$ essentially consists of all functions that can be regarded as counting the accepting computations of polynomial time NDTMs. The class $\#L$ is defined similarly:

$$\#L = \{[M] \mid M \text{ is a logspace NDTM}\}.$$

Cook [5] proved that $NL \subseteq P$. The following is a corresponding result for enumeration problems, and is implicit in the literature as we shall see.

Theorem 1.

$$\#L \subseteq FP.$$

Proof. Jones [29] proved that the following problem is logspace complete for NL.

DIGRAPH ACCESSIBILITY
Input: Directed graph G, specified vertices s, t ϵ V(G).
Question: Is there a (directed) walk from s to t in G?

This is proved by a generic transformation from each member of NL. It is easy to check that this transformation is parsimonious (see [18, pp. 168-9]). It follows that the corresponding enumeration problem, of counting the number of walks between two specified vertices of a directed graph, is complete for $\#L$ with respect to parsimonious logspace reducibility. This problem, however, is well known to be solvable in polynomial time (see [25, Theorem 16.8]). It follows that all problems in $\#L$ can be solved in polynomial time.

This result could also be proved directly, using a technique similar to that of Cook's proof [5] that NL $\subseteq$ P. □

If $f : \Sigma_1^* \to \Sigma_2^*$ and $g : \Sigma_1^* \to \Sigma_2^*$ then we say f is <u>(polynomial time) Turing reducible</u> to g, written $f \propto_T g$, if f can be computed by a polynomial time OTM with an oracle for g. f and g are <u>(polynomial time) Turing equivalent</u>, written $f \equiv_T g$, if $f \propto_T g$ and $g \propto_T f$. We say f is <u>$\#$P-complete</u> if $f \in \#P$ and $g \propto_T f$ for all $g \in \#P$, and f is <u>$\#$P-hard</u> if $g \propto_T f$ for some $\#$P-complete g.

A language can be identified with its characteristic function so the above definition of Turing reduction in fact includes the previous definition we made for languages. In this sense we can regard language classes as function classes (but generally not vice versa). Thus we can assert:

(1.2)

$$P \subset FP \quad \text{and} \quad L \subset FL.$$

The inclusions are proper for the technical reason that FP and FL contain functions which are not characteristic functions.

To summarise, the following inclusions hold among our complexity classes of functions:

$$\begin{array}{ccccccc} \text{LOGSPACE} & \subseteq & NL & \subseteq & P & \subseteq & NP \\ & & \cap & & \cap & & \\ FL & \subseteq & \#L & \subseteq & FP & \subseteq & \#P \end{array} .$$

We close with some basic definitions about functions on $\mathbb{N}$; these follow standard usage. Suppose f and g are functions. We say f <u>dominates</u> g if $f(n) \geq g(n)$ for all sufficiently large n. If $k \cdot g(n)$ dominates $f(n)$ for some constant k then we write $f(n) = O(g(n))$, and if $f(n)/g(n) \to 0$ as $n \to \infty$ then we write $f(n) = o(g(n))$. Finally, we write $f(n) = \Omega(g(n))$ if $g(n) = O(f(n))$, and $f(n) = \Theta(g(n))$ if both $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

(2.1)

CHAPTER 2

THE COMPLEXITY OF HOMEOMORPHISM PROBLEMS

The concept of homeomorphism has been of considerable importance in graph theory, ever since Kuratowski's celebrated characterization [35] of planar graphs as those graphs with no subgraph homeomorphic from K_5 or $K_{3,3}$. In this chapter we consider algorithmic questions involving homeomorphism. We begin in §1 with the necessary definitions and a brief survey of the field. In §§ 2 and 3 we consider the complexity of deciding whether a graph has a subgraph homeomorphic from another fixed graph, when this fixed graph is a small circuit or a small wheel, and obtain some results on the exact structure of graphs not containing such fixed graphs as homeomorphs. Finally in §4 we turn our attention to the corresponding enumeration problems and prove that for any finite nontrivial family of graphs H, the problem of finding the number of subgraphs of a graph which are homeomorphic from a member of H is $\#$P-complete. The main results of §4 have appeared in [15].

§1. The subgraph homeomorphism problem

In this section we briefly survey previous work on the complexity of the subgraph homeomorphism problem and related problems. We begin with some definitions.

Definitions: The operation of contracting an edge e of a graph G consists of the removal of e from G and the identification of its endpoints. If at least one endpoint of e

(2.1)

has degree 2 in G then such a contraction is called a series-contraction. Let G and H be graphs. We say G is contractible to H if an isomorphic copy of H can be obtained from G by a sequence of contractions. G is homeomorphic from H if an isomorphic copy of H can be obtained from G by a sequence of series-contractions. Here we also say G is a homeomorph of H, or an H-homeomorph. Further, G is homeomorphic with H if both G and H are homeomorphic from a third graph. If G has a subgraph contractible to H then H is said to be a minor of G.

The following trivial Lemma tells us that the properties of being homeomorphic from H and contractible to H are identical if H has no vertex of degree greater than or equal to 4.

Lemma 1: For every graph H of maximum degree at most 3 and every graph G,

(1) G is contractible to H iff G is homeomorphic from H,

and (2) G has a minor isomorphic to H iff G has a subgraph homeomorphic from H. □

These statements are false if H has a vertex of degree ≥ 4. For example, $K_{3,3}$ is contractible to W_4 (contract any edge) but is clearly not homeomorphic from W_4.

Although we make some mention of problems involving contractibility and minors because of their close connection with homeomorphism, we are concerned mainly with the latter. In particular we are interested in determining when a given graph has a subgraph homeomorphic from a second, fixed graph. This

(2.1)

problem may be stated formally as follows, where H is a fixed graph. (We will sometimes refer to H as the pattern graph.)

SUBGRAPH HOMEOMORPHISM (H) (abbreviated SHP(H)).

Input: Graph G.

Question: Does G contain a subgraph homeomorphic from H?

A related problem on minors is the following.

MINOR (H)

Input: Graph G.

Question: Does G have a minor isomorphic to H?

It is well known that if H is allowed to vary, as part of the input to the problem, then both these problems become NP-complete. This is probably most easily seen by letting H be a circuit of length $|V(G)|$ and observing that then SHP(H) = HAMILTONIAN CIRCUIT. One may then ask for each of the problems SHP(H) and MINOR(H) : for which graphs H is it in P and for which graphs is it NP-complete? (Both are clearly in NP for all H.) For SHP(H), this question was first asked by LaPaugh and Rivest [37] and is stated in Garey and Johnson's list [18] of open problems. At that stage very little was known. For several nontrivial pattern graphs H, exact characterizations had been found for those graphs not in SHP(H). We summarise these results in

Theorem 2: (1) $G \notin SHP(K_3)$ iff G is a forest.

(2) (Dirac [8], Duffin [9]) $G \notin SHP(K_4)$ iff G is a series-parallel graph.

(2.1)

(3) (Hall [24]) $G \notin SHP(K_{3,3})$ iff G can be built up from planar graphs and copies of K_5 by repeated 1- and 2-sums.

(4) (Menger [40]) $G \notin SHP(C_k^*)$ iff for each $v,w \in V(G)$ there is a separating set W, with $|W| < k$, such that no part of $G|W$ contains both v and w. □

Here (3) follows from a theorem of D.W. Hall [24] that if $G \notin SHP(K_{3,3})$ then either G is planar, or isomorphic to K_5, or has a 2-separation. (4) of course is a trivial corollary of Menger's Theorem. Observe that by Lemma 1, $SHP(H) = MINOR(H)$ if H has no vertex of degree at least 4, so these languages are certainly equal for $H \in \{K_3, K_4, K_{3,3}\}$. Thus (1), (2), (3) in Theorem 2 (above) and Corollary 3 (below) give us results for the equivalent languages $MINOR(H)$ as well. Using the characterizations of Theorem 2, polynomial time algorithms can be found:

Corollary 3:

(1) $SHP(K_3) \in P$.

(2) $SHP(K_4) \in P$.

(3) $SHP(K_{3,3}) \in P$.

(4) $SHP(C_k^*) \in P$ for all k. □

We mention in passing that the directed version of the subgraph homeomorphism problem was considered by Fortune, Hopcroft and Wylie [16] who completely classified all directed pattern graphs H according to whether the directed subgraph homeomorphism problem for H is in P or NPC.

No significant progress, on the complexity of $SHP(H)$ in general, was made until Robertson and Seymour [47] considered the complexity of the problem of detecting minors.

(2.1)

Theorem 4 [47]: For any graph H, MINOR(H) ϵ P. □

This very general result was obtained as a result of research into graph minors, tree-width and well-quasi-ordering reported in [44-56]; most of it has still to be published. Their polynomial time algorithms are very general and are not efficient in practical terms for all but the most trivial pattern graphs H. They do not yield elegant exact characterizations of the sort, for example, of the results in Theorem 2 (1), (2), (3) above. As suggested by P.D. Seymour (private communication) it is still of interest to obtain exact characterizations of graphs with a specific minor excluded and efficient algorithms for MINOR(H) for specific graphs H. It is with this motivation that we consider the languages $MINOR(C_k) = SHP(C_k)$ for certain small values of k in the next section. We note that characterizations have also been found for graphs with K_5 (Wagner [67]) or with $K_5 - e$ (Seymour, unpublished) excluded as a minor. ($K_5 - e$ is the graph obtained from K_5 by removing an edge.)

A related problem is the following.

DCP(k)

Input: Graph G and specified vertices $s_1,\ldots,s_k,t_1,\ldots,t_k$ in G.

Question: Do there exist k paths $P_1,\ldots,P_k$ which are vertex-disjoint, except possibly at their endpoints, such that P_i connects s_i to t_i for all i, $1 \leq i \leq k$?

If k is allowed to vary, as part of the input, then the problem is well known to be NP-complete [31]. Robertson and Seymour (see [47]) recently used their graph minor algorithms to obtain

a difficult and complicated polynomial time algorithm for any fixed k.

Theorem 5 [47]: For any fixed integer k, $DCP(k) \in P$. □

This shows that for any H, $SHP(H) \in P$, by the following well known result.

Lemma 6: For any H, $SHP(H) \propto_T DCP(|E(H)|)$. □

Combining Theorem 5 and Lemma 6 we have

Theorem 7 (Robertson and Seymour [47]):

$SHP(H) \in P$ for all H. □

Thus, in the sense of polynomial time solvability versus NP-completeness, the subgraph homeomorphism problem is now solved.

Once again the algorithm is very general and not practical. In fact this comment applies even more to Robertson and Seymour's algorithms for $SHP(H)$ than to their algorithms for $MINOR(H)$. The algorithm for $SHP(H)$ involves solving $DCP(|E(H)|)$ a fixed (but large if H is not reasonably trivial) number of times (Lemma 6), and the algorithm for $DCP(|E(H)|)$ in turn uses minor algorithms for fixed but large pattern graphs.

Thus the exact structure of graphs with no H-homeomorph is still undetermined for all but a few cases. In fact excluding homeomorphs seems in general to yield more complicated structures than excluding minors. For those pattern graphs H with no vertex of degree at least 4 then $SHP(H)$ equals $MINOR(H)$ and so is in fact no more difficult; circuits, which we investigate in the next section, come into this category of course. For pattern graphs without this property, though, the situation seems rather different. In §3 we obtain results for graphs with

W_4 and W_5 excluded as a homeomorph. These are natural pattern graphs to study as they are the simplest 3-connected pattern graphs H which have a vertex of degree at least 4 and therefore for which SHP(H) is not trivially equal to MINOR(H). It is to be hoped that the study of such graphs may shed some light on the structure of graphs obtained by excluding homeomorphs in general. However, this goal is still some way off; the case $H \cong K_5$ for example is still far from understood, in contrast to the situation where K_5 is excluded as a minor which was settled by Wagner [67] as far back as 1937.

§2. Excluding circuits

We consider the structure of graphs with no C_k-homeomorph, when $k \leq 7$, obtaining new results and algorithms for the cases $k = 6$ and $k = 7$.

The first two results stated have probably been known for some time and are easily proved. The earliest reference we can find in the literature is the paper of Seymour and Walton [62] who establish the result for general matroids.

Lemma 1 [62]: If a connected graph G has no circuit of length greater than or equal to 4, then either

(i) $|V(G)| \leq 3$,

or (ii) G has a cutvertex. □

Corollary 2: A graph G has no circuit of length ≥ 4 iff it can be constructed by 1-sums from graphs on 3 or fewer vertices. □

It follows immediately that $SHP(C_4) \in P$.

The technique for proving the above results is one that we will use repeatedly. If the pattern graph is C_k (where $k \leq 7$), we determine the structure of nonmembers of $SHP(C_k)$ without a cutvertex. Since a circuit in a graph can be contained in at most one block, this completely determines the structure of nonmembers of $SHP(C_k)$ (as in, for example, Corollary 2). A polynomial time algorithm for $SHP(C_k)$ can then be found, based on finding, for an input graph G, a cutvertex w and then examining the parts of $G|\{w\}$. Similar techniques are used in §3 (when the pattern graphs are wheels), where we use 2-separations instead of cutvertices.

(2.2)

Exclusion of C_5 has also been considered by Seymour and Walton for general matroids. Restricting attention to graphs, they proved

Theorem 3 [62]: If a connected graph G has no circuit of length ≥ 5 then either

(i) $|V(G)| \leq 4$,

or (ii) $G \leq K_{1,1,n-2}$ $(n = |V(G)|)$,

or (iii) G has a cutvertex. □

Now, as remarked earlier, every circuit of a graph G lies entirely within some block of G. By considering the blocks of G, then, we obtain

Corollary 4: A graph G has no circuit of length ≥ 5 iff it can be constructed by 1-sums from graphs on 4 or fewer vertices and subgraphs of $K_{1,1,n}$. □

Corollary 5: $SHP(C_5) \in P$.

For the purpose of the next Theorem, we define the following graphs. A_k is the graph obtained from K_4 by distinguishing two of its vertices u and v and then joining $k - 4$ new vertices to precisely the vertices u and v (see Fig. 1(a)). $B_{k,\ell}$ is the graph obtained from a triangle, with vertices v_1, v_2, v_3 say, by taking $k + \ell$ new vertices, joining k of them to both v_1 and v_2 and joining ℓ of them to both v_1 and v_3 (see Figure 1(b)).

(2.2)

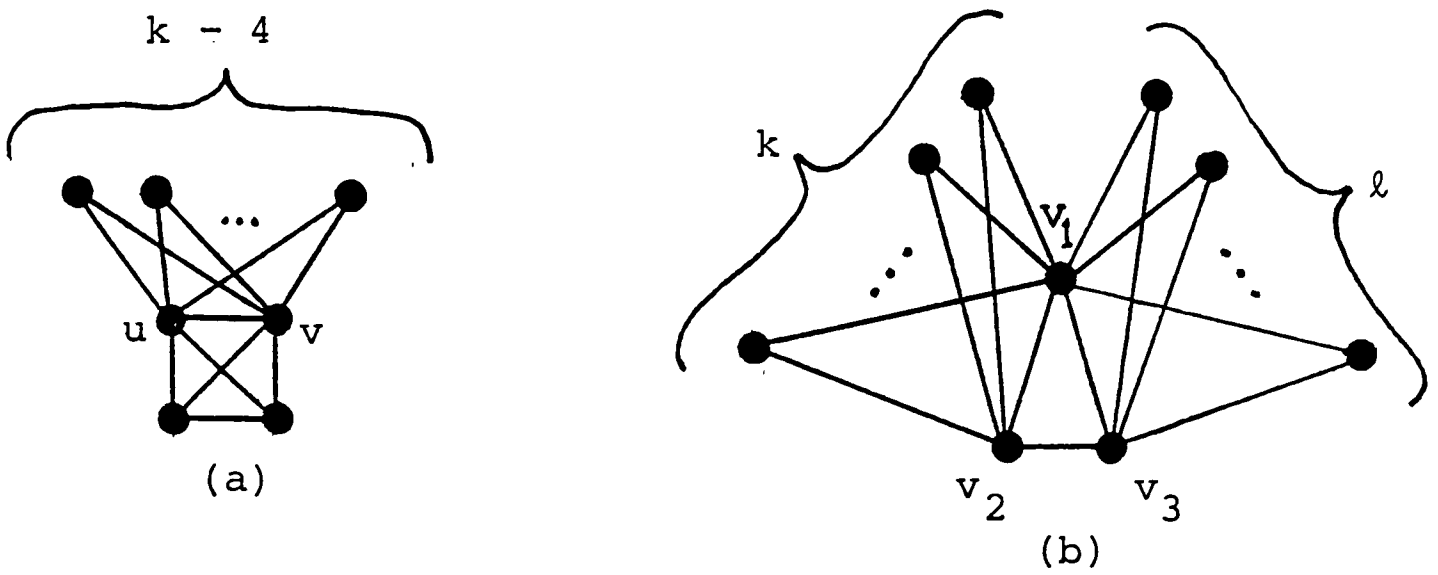


Figure 1. (a) The graph A_k. (b) The graph $B_{k,\ell}$.

Theorem 6: If a connected graph G has no circuit of length at least 6, then at least one of the following is true:

(i) $|V(G)| \leq 5$;

(ii) $G \leq A_k$ for some k;

(iii) $G \leq B_{k,\ell}$ for some k,ℓ;

(iv) G has a cutvertex.

Proof: Let $G = (V,E)$ be a connected graph with no circuit of length ≥ 6. Put $n = |V|$. Let m be the size of a longest circuit of G, and suppose G satisfies neither (i) nor (iv) above. We show G satisfies (ii) or (iii).

If $m \leq 4$, then Theorem 3 and the fact that $n \geq 6$ imply that $G \leq K_{1,1,n-2}$. But $K_{1,1,n-2} \leq A_{n+2}$, so we are done.

Suppose then that $m = 5$ and let C be a circuit of G of size 5 with vertex set $V_C = \{v_1,\ldots,v_5\}$ and edge set $E_C = \{v_1v_2, v_2v_3, v_3v_4, v_4v_5, v_5v_1\}$.

Let U be any part of the separation $G|V_C$ such that U contains some vertex not in V_C (some such U exists since $n \geq 6$). If $|U \cap V_C| \leq 1$ then G is not 2-connected,

(2.2)

so assume $|U \cap V_C| \geq 2$. If $U \cap V_C$ contains two vertices adjacent in C then a circuit of length ≥ 6 can easily be constructed. But this must happen if $|U \cap V_C| \geq 3$, so $U \cap V_C$ consists of precisely two members of V_C. As we have just seen, they are not adjacent; suppose w.l.o.g. $U \cap V_C = \{v_1, v_3\}$.

Now every $v_1 - v_3$ path in $\langle U \rangle$ has length ≤ 2: a path of length ≥ 3 when joined with the path v_3, v_4, v_5, v_1 gives a circuit of size ≥ 6. Also every $v \in U \backslash V_C$ is joined in $\langle U \rangle$ to $\{v_1, v_3\}$ by two paths which are vertex-disjoint except at v, else G has a cutvertex. Thus in fact each $v \in U \backslash V_C$ is adjacent to v_1 and v_3.

Now suppose by way of contradiction that $|U \backslash V_C| \geq 2$. Let $v, w \in U \backslash V_C$, $v \neq w$. By definition of U there is a $v - w$ path P_{vw} in $\langle U \rangle$ which does not meet V_C. Then

$$v_1, v, P_{vw}(v,w), w, v_3, v_4, v_5, v_1$$

is a circuit of length ≥ 6 in G. Hence $|U \backslash V_C| = 1$, and we have shown that each part U of $G|V_C$ with $U \backslash V_C \neq \emptyset$ induces a path of length 2 between some two vertices of V_C which are not adjacent in C, together with possibly an edge between these two vertices (see Figure 2).

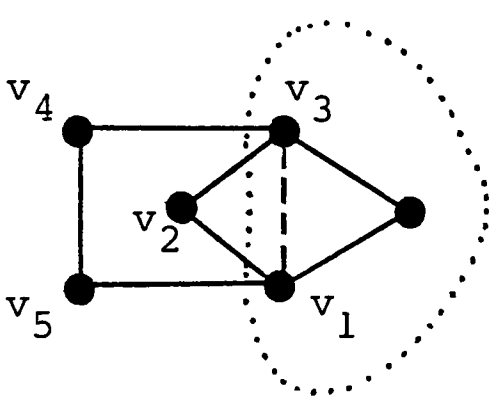


Figure 2: The subgraph enclosed in the dotted line is a typical part of $G|V_C$ in the proof of Theorem 6. The dashed edge may or may not exist.

Define R to be the set of all pairs $\{v,w\}$ of vertices $v,w \in V_C$ such that $v,w \in U$ for some part U of $G|V_C$ satisfying $U\backslash V_C \neq \emptyset$. Then (V_C,R) is a graph, which we denote by $S_G(C)$. We know that $vw \in E_C \Rightarrow vw \notin R$, so $R \cap E_C = \emptyset$. Suppose R has two disjoint edges, which we may suppose w.l.o.g. to be v_1v_3 and v_2v_4. Let U_{13} and U_{24} be parts of $G|V_C$ containing $\{v_1,v_3\}$ and $\{v_2,v_4\}$ respectively and satisfying $U_{13}\backslash V_C \neq \emptyset$, $U_{24}\backslash V_C \neq \emptyset$. Then the paths of length 2 in U_{13} and U_{24}, joining pairs $\{v_1,v_3\}$ and $\{v_2,v_4\}$ respectively, together with edges v_1v_2 and v_3v_4, give a circuit of length ≥ 6 (see Figure 3).

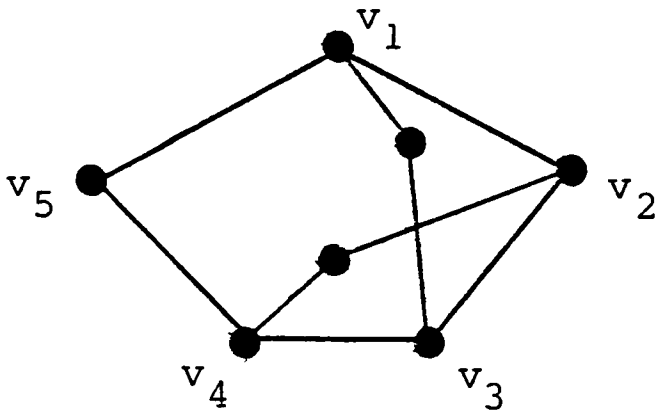
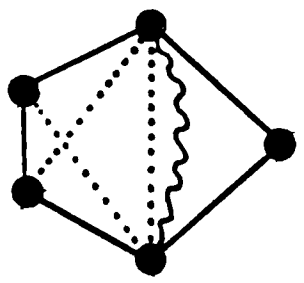


Figure 3

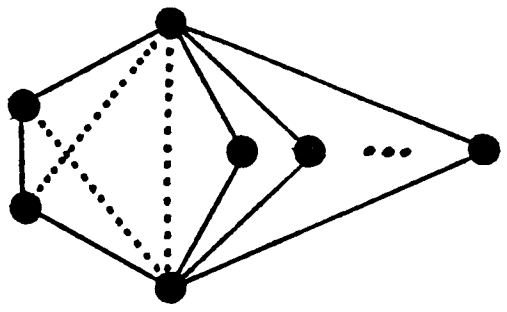
So R has no pairs of disjoint edges. Then $S_G(C)$ is either a triangle, or a star $K_{1,k}$ for some k. If $S_G(C)$ is a triangle, or $K_{1,k}$ with $k \geq 3$, then some edge of C is in R, a contradiction. So $S_G(C) \cong K_2$ or $K_{1,2}$. At this point observe that if $\{v,w\} \in R$, and u is a common neighbour of v,w in C, then u is adjacent to no other vertex of C (else a circuit of size ≥ 6 can be constructed). Now if $S_G(C) \cong K_2$, so that $|R| = 1$, then this observation allows us to conclude immediately that $G \leq A_k$ for some k (see Figure 4; the wavy lines in (a) indicate edges of R, and dotted lines indicate edges with whose presence or absence we

(2.2)

are not concerned).



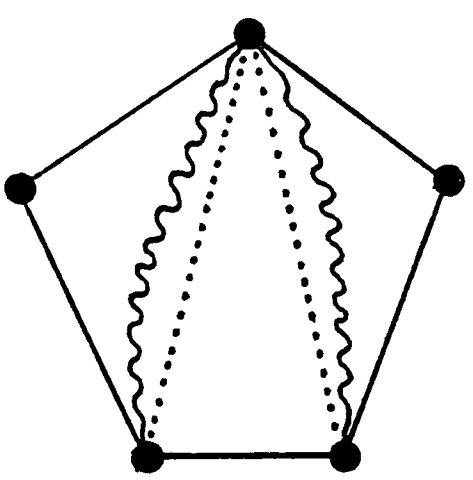
(a)



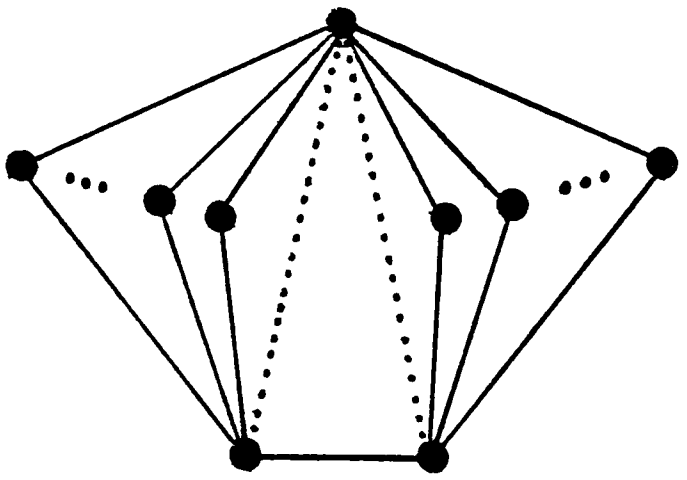
(b)

Figure 4: The case $S_G(C) \cong K_2$. Structure of G as shown in (b) deduced from that of $S_G(C)$ (the wavy edges in (a)).

Finally, if $S_G(C) \cong K_{1,2}$ our above observation applied to both edges in R allows the conclusion $G \leq B_{k,\ell}$ for some k,ℓ (see Figure 5; same conventions as for Figure 4). The proof is now complete. □



(a)



(b)

Figure 5: The case $S_G(C) \cong K_{1,2}$. Structure of G as shown in (b) deduced from that of $S_G(C)$ (a).

Corollary 7: A graph G has no circuit of length ≥ 6 iff it can be constructed by 1-sums from graphs on 5 or fewer vertices and subgraphs of A_k and $B_{k,\ell}$ for some k,ℓ. □

Corollary 8: $SHP(C_6) \in P$. □

A number of graphs must be defined for the next Theorem. See Figure 6. $L(k)$ is obtained from K_5 by selecting two vertices of it and joining k new vertices to precisely these two vertices. $M(k) \cong K_{1,1,1,k}$. $N(k_1,k_2,k_3)$ is obtained from a copy of K_4, with vertices v_0,v_1,v_2,v_3 say, by including $k_1 + k_2 + k_3$ new vertices, of which k_i are joined precisely to v_0 and v_i for each $i \in \{1,2,3\}$. $P(k_1,k_2,k_3)$ is obtained by taking k_1 copies of K_4 each with a distinguished edge, and k_2 copies of $K_{1,1,k_3+1}$ each with a distinguished edge which has one endpoint of degree 2, and identifying all the distinguished edges. $Q(k_1,k_2)$ is obtained from a copy of K_4 on vertices v_i, $i = 1$ to 4, by introducing $k_1 + k_2$ new vertices and joining k_i of them to precisely the vertices v_{2i-1} and v_{2i}, for $i = 1,2$.

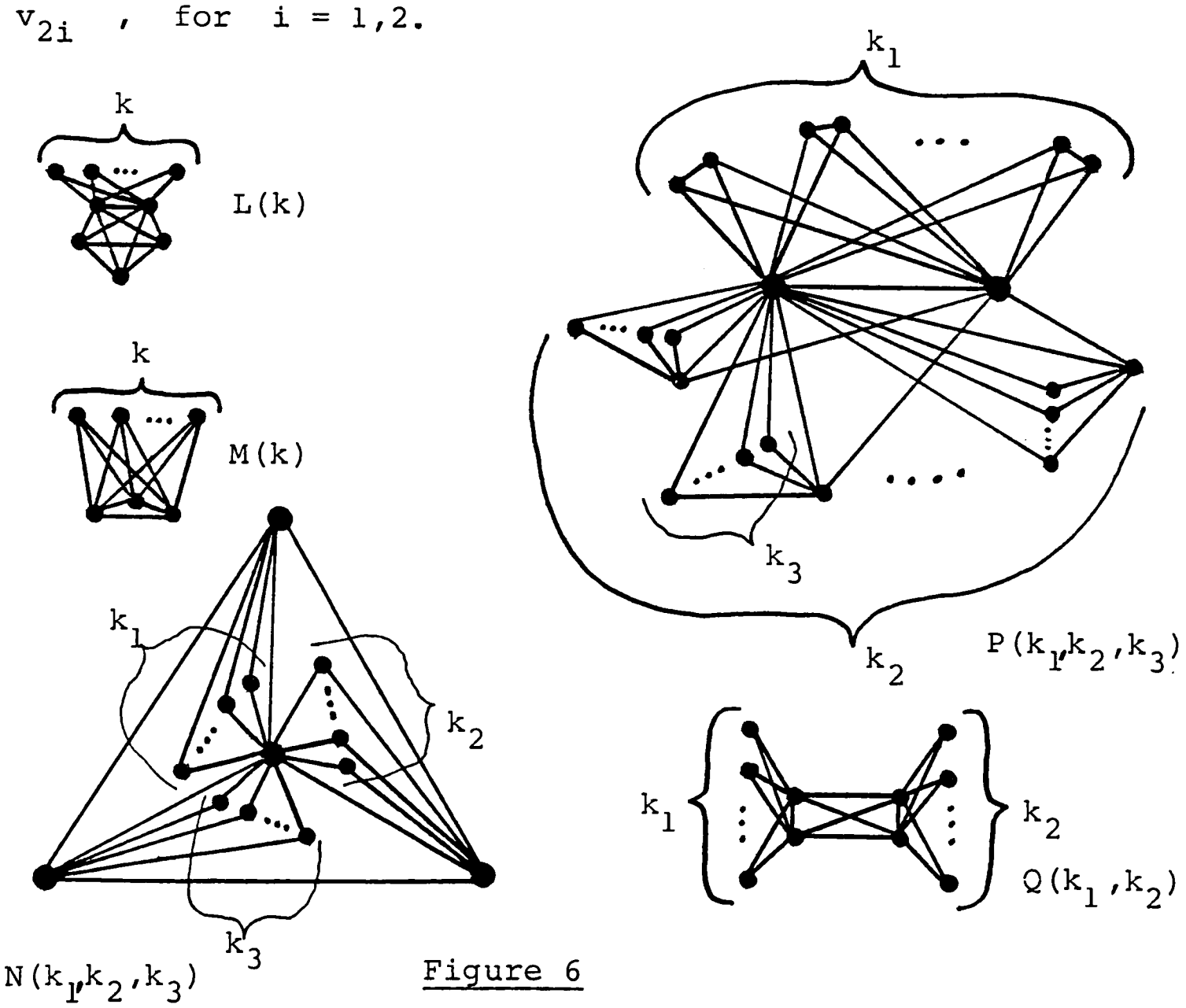


Figure 6

We will give only an outline of the proof of this Theorem; we apologize to the reader for omitting the details, but the proof is long and laborious, and contains nothing new in technique beyond what was used in proving Theorem 6.

Theorem 9: If a connected graph G has no circuit of length ≥ 7, then at least one of the following statements holds:

(i) $|V(G)| \leq 6$;

(ii) $G \leq L(k)$ for some k;

(iii) $G \leq M(k)$ for some k;

(iv) $G \leq N(k_1,k_2,k_3)$ for some k_1, k_2, k_3;

(v) $G \leq P(k_1,k_2,k_3)$ for some k_1, k_2, k_3;

(vi) $G \leq Q(k_1,k_2)$ for some k_1, k_2;

(vii) G has a cutvertex.

Proof (sketch): Let $G = (V,E)$ be a connected graph with no circuit of size ≥ 7. Let m be the size of the longest circuit in G, and suppose G satisfies neither (i) nor (vii) above. We show G satisfies one of (ii) - (vi).

If $m \leq 5$, the result follows easily from Theorem 6. So suppose $m = 6$ and let C be a circuit of G of size 6 with vertex set $V_C = \{v_1,\ldots,v_6\}$ and edge set $E_C = \{v_1v_2, v_2v_3, \ldots, v_5v_6, v_6v_1\}$. As in the proof of Theorem 6, consider a part U of $G|V_C$ such that $U \backslash V_C \neq \emptyset$ and observe that $|U \cap V_C| \geq 2$. Furthermore U cannot contain two vertices adjacent in C and it follows that $|U \cap V_C| \leq 3$. We determine the structure of U in the following three cases, which (without loss of generality) cover all possibilities.

Case 1: $U \cap V_C = \{v_1, v_3\}$.

$\langle U \rangle$ consists just of a single path of length 2 between the two vertices of $U \cap V_C$, together with possibly an edge between them.

Case 2: $|U \cap V_C| = 3$.

Show that each $v \in U \backslash V_C$ is adjacent to each member of $U \cap V_C$, and that $|U \backslash V_C| = 1$. Thus $U \backslash V_C$ consists of a single vertex adjacent to each vertex in $U \cap V_C$.

Case 3: $U \cap V_C = \{v_1, v_4\}$.

Here it can be shown, with more effort than in the previous two cases, that $\langle U \rangle$ is one of the two graphs shown in Figure 7, where dashed lines indicate edges whose presence or absence does not matter. The second graph consists (apart from dashed edges) of a copy of $K_{2,k}$, for some k, with one edge removed.

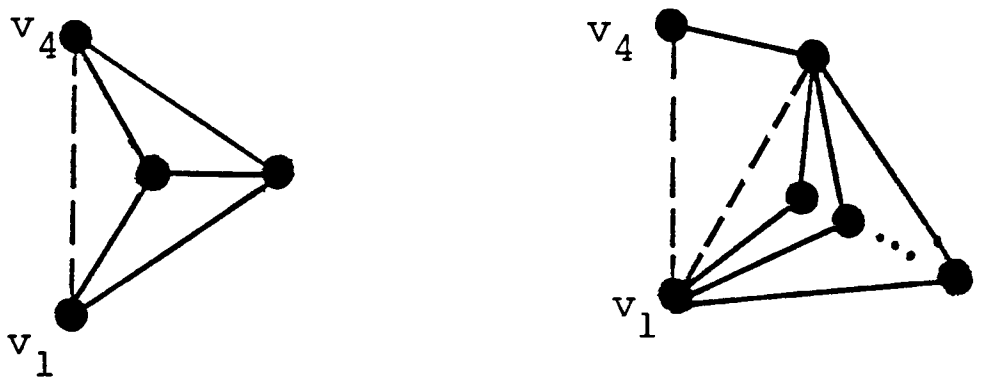


Figure 7

We now determine the structure of G by considering the ways in which the parts of $G|V_C$ can meet V_C.

Let U_1 and U_2 be any parts of $G|V_C$ satisfying $U_1 \backslash V_C \neq \emptyset$, $U_2 \backslash V_C \neq \emptyset$ and $U_1 \neq U_2$. U_1 and U_2 are said to cross if there exists $u_1, w_1 \in U_1 \cap V_C$, with $u_1 \neq w_1$ (and necessarily $u_1 \not\sim w_1$), such that the two $u_1 - w_1$ paths

in C each have a member of $U_2 \cap V_C$ as one of their internal vertices. (This definition is clearly symmetric in U_1 and U_2.) Show that it is in fact impossible to have such crossing parts of $G|V_C$. This considerably restricts the structure of G. Consider two cases:

Case 1: Some part U of $G|V_C$ with $U\backslash V_C \neq \emptyset$ also satisfies $|U \cap V_C| = 3$.

It can be shown here, using our "no-crossing" condition, that $G \leq K_{1,1,n-3}$, so we are done.

Case 2: Every part U of $G|V_C$ with $U\backslash V_C \neq \emptyset$ also satisfies $|U \cap V_C| = 2$.

Define the graph $S_G(C) = (V_C,R)$ exactly as in the proof of Theorem 6. The "no-crossing" condition described above forces $|R| \leq 3$. The rest of the proof involves considering, in turn, the possibilities $|R| = 3$, $|R| = 2$, and $|R| = 1$, and tediously deducing, for each case and each possible structure of $S_G(C)$, that G satisfies one of (ii) - (vi). The arguments are entirely routine, and the proof is then complete. □

Corollary 10: A graph G has no circuit of length ≥ 7 iff it is a 1-sum of graphs with at most 6 vertices and subgraphs of $L(k)$, $M(k)$, $N(k_1,k_2,k_3)$, $P(k_1,k_2,k_3)$ and $Q(k_1,k_2)$ for some k, k_1, k_2, k_3. □

Corollary 11: $SHP(C_7) \in P$. □

Excluding C_8 would give an even more complicated and less elegant result, and the situation would appear to get worse when excluding larger circuits.

(2.3)

§3. Excluding wheels

We now consider the structure of graphs with, in turn, W_4 and W_5 excluded as homeomorphs. Most of this section is taken up with Theorem 4 on W_5.

The first result characterizes 3-connected members of $SHP(W_4)$. It is reminiscent of Tutte's characterization of 3-connected graphs (see [25, p.46]) but we cannot see a more direct connection.

Theorem 1: Let G be a 3-connected graph. Then G contains a W_4-homeomorph iff G has a vertex of degree greater than or equal to 4. □

We will prove a technical strengthening of this result which we will need for Theorem 4.

Definition: If $k \geq 3$ and H is a subgraph of G which is a W_k-homeomorph, then we say H is centred on a vertex $v \in V(G)$ if v has maximum degree in H.

Thus a W_3-homeomorph is centred on each of its vertices of degree 3, while if $k \geq 4$ a W_k-homeomorph is centred only on its single vertex of degree k.

Lemma 2: If G is a 3-connected graph and v is any vertex of degree ≥ 4 in G, then G contains a W_4-homeomorph centred on v.

Proof: Suppose G is 3-connected and that $v_0 \in V(G)$ with $\deg v_0 \geq 4$. Let v_1, v_2, v_3, v_4 be four neighbours of v_0. By the 3-connectivity of G there exist paths P_1 and P_2 from $\{v_1, v_3\}$ to $\{v_2, v_4\}$ in G which are vertex-

disjoint and avoid v_0. Now consider the vertex sets of these paths. Again by the 3-connectivity of G there exist two vertex-disjoint paths Q_1 and Q_2 from $V(P_1)$ to $V(P_2)$ each of which meets each of $V(P_1)$ and $V(P_2)$ only once and avoids v_0. These paths P_1, P_2, Q_1, Q_2 together with v_0 and the edges $v_0v_i, 1 \le i \le 4$, constitute the required W_4-homeomorph. (Writing w_{ij} for the sole member of $V(P_i) \cap V(Q_j)$ for each $i,j \in \{1,2\}$, the circuit not containing v_0 in our W_4-homeomorph is made up of the paths $P_1[w_{11},w_{12}]$, $P_2[w_{21},w_{22}]$, Q_1 and Q_2.) Figure 1 shows (w.ℓ.o.g.) the result. □

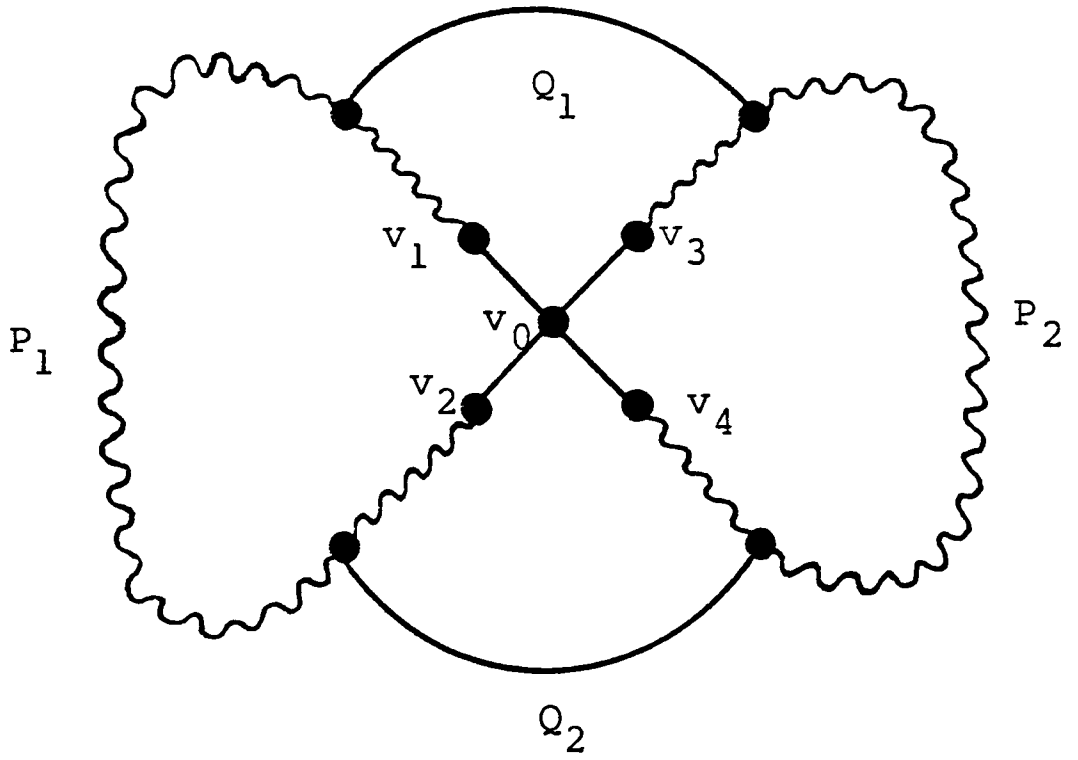


Figure 1: The W_4-homeomorph found in the proof of Lemma 2.

Theorem 1 forms the basis of the polynomial time algorithm for $SHP(W_4)$ given in the next Theorem. The Theorem itself, of course, follows also from the general results of Robertson and Seymour (see Theorem 1.7).

(2.3)

Theorem 3: $SHP(W_4) \in P$.

Proof: Consider the following algorithm.

1. Input : Graph G.
2. If $|V(G)| \leq 4$, reject G.
3. If G is 3-connected, go to 4; otherwise:
   - 3.1. Find a separating set V_0 for G of at most two vertices, with V_0 minimal.
   - 3.2. If $|V_0| = 2$, form G' by adding an edge between the two members of V_0 if none exists already. If $|V_0| = 1$, G' := G.
   - 3.3. Find the parts $U_1,\ldots,U_k$ of the separation $G'|V_0$.
   - 3.4. Apply the algorithm recursively to each $\langle U_i \rangle$, $1 \leq i \leq k$. If any $\langle U_i \rangle$ is accepted, accept G. Otherwise, reject G.
4. If G has a vertex of degree at least 4, accept G; otherwise, reject G.

This algorithm accepts a graph G iff one of the following holds:

(i) G is 3-connected and has a vertex of degree at least 4.

(ii) G has a separating set V_0 with $|V_0| \leq 2$ such that some subgraph induced by one of the parts of $G'|V_0$ is accepted when the algorithm is recursively applied to it.

If a graph G has a separating set V_0 of size at most 2 then G has a W_4-homeomorph iff some subgraph of G induced by a part of $G'|V_0$ has a W_4-homeomorph. This follows from the 3-connectivity of W_4. In effect, the only

way a W_4-homeomorph can "straddle" V_0 is in the case $|V_0| = 2$, by a single path which passes through V_0 twice, returning to the part of $G'|V_0$ that it came from; this is reflected in the addition of an edge between the two members of V_0 in Step 3.2.

This observation, together with Theorem 1, implies that one of (i), (ii) holds iff G has a W_4-homeomorph. It is clear that the algorithm runs in polynomial time (it is a standard divide-and-conquer method), so the proof is complete. □

We now attack the wheel W_5. A technical definition is necessary.

Definition: A proper 3-edge-cutset in a graph G is a set E' of at most 3 edges of G such that $G - E'$ is disconnected with each component having more than one vertex.

Theorem 4: Let G be 3-connected, with no proper 3-edge-cutset. Then G has a W_5-homeomorph iff G has a vertex v of degree at least 5 and a circuit of size at least 5 which does not contain v.

Proof: Suppose G is 3-connected with no proper 3-edge-cutset. The forward implication is trivial. Suppose then that G has a vertex v_0 of degree at least 5 and a circuit of size at least 5 which does not contain v_0.

By Lemma 2 G contains a W_4-homeomorph centred on v_0. If $H = (V_H, E_H)$ is any W_4-homeomorph centred on v_0, we write: C_H for the circuit of H which doesn't meet v_0; v_i^H, $1 \le i \le 4$, for the four vertices of degree 3 in H; and P_i^H for the path from v_0 to v_i^H in H (which meets C_H only at v_i^H), for each i, $1 \le i \le 4$. We assume (w.ℓ.o.g.) that the v_i^H

are arranged on C_H so that it is possible to move around C_H encountering v_1^H, v_2^H, v_3^H, v_4^H in exactly this order. Clearly $N_G(v_0)\backslash N_H(v_0) \neq \emptyset$, since deg $v_0 \geq 5$. If $u \in N_G(v_0)\backslash N_H(v_0)$, define the vertex set $U_H(u)$ as follows: if $u \notin V_H$, $U_H(u)$ is the part of $G|V_H$ which contains u; if $u \in V_H$, $U_H(u) = \{v_0,u\}$.

There are three cases to consider.

Case 1: G has a W_4-homeomorph H centred on v_0 and a vertex $u \in N_G(v_0)\backslash N_H(v_0)$ such that $U_H(u)$ contains a vertex u_1 of $C_H\backslash\{v_1^H,v_2^H,v_3^H,v_4^H\}$.

There is then a $v_0 - u_1$ path in $\langle U_H(u)\rangle$ which does not meet H except at v_0 and u_1, and this path together with H gives a W_5-homeomorph in G. (See Figure 2; the wavy line is the $v_0 - u_1$ path in $\langle U_H(u)\rangle$.)

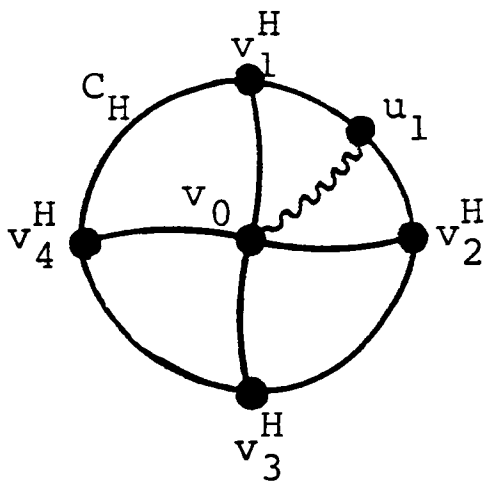


Figure 2: Diagram for Case 1.

Case 2: G has a W_4-homeomorph H centred on v_0 and a vertex $u \in N_G(v_0)\backslash N_H(v_0)$ such that $U_H(u)$ contains two vertices, other than v_0, on two separate P_i^H. (Note that in this case $u \notin V_H$.)

Let these two vertices be u_1,u_2. Since G is 3-connected there are two paths Q_1 and Q_2 in $\langle U_H(u)\rangle$ from u to u_1

and u_2 respectively which meet H only at u_1 and u_2 and which are vertex-disjoint except at u. Assume w.ℓ.o.g. that u_1 is on P_1^H and u_2 is on P_2^H or P_3^H.

Subcase 2a: $\{u_1,u_2\} \neq \{v_1^H,v_3^H\}$.

Assume w.ℓ.o.g. that $u_2 \neq v_3^H$. G contains a W_5-homeomorph formed from H as follows: add the vertex u, the edge uv_0 and paths Q_1 and Q_2, and remove the internal vertices of the $v_1^H - v_2^H$ path in C_H which avoids v_3^H. Figures 3(a) and (b) illustrate this when u_2 is on P_2^H or P_3^H respectively, and in each case the starred portion (*) is the path to be removed as just described. The reader may easily verify the presence of a W_5-homeomorph in each instance.

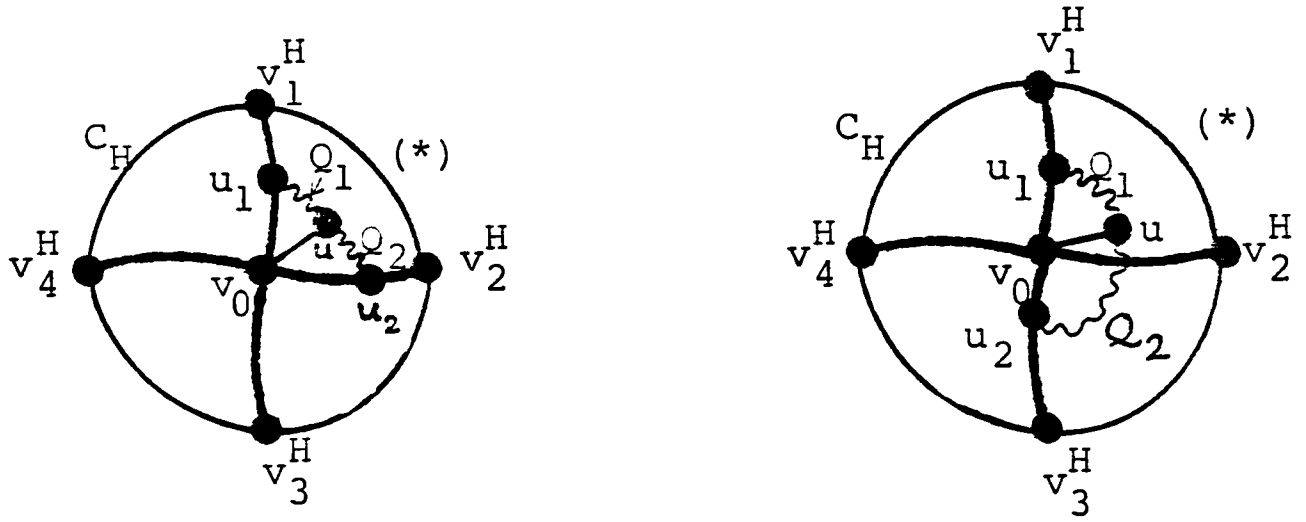


Figure 3: Diagrams for Subcase 2a in the proof of Theorem 4.

Subcase 2b: $u_1 = v_1^H$, $u_2 = v_3^H$.

Put $W = \{v_0,v_1^H,v_3^H\}$. We may suppose that $U_H(u) \cap V_H = W$, since if $(U_H(u) \cap V_H)\backslash W \neq \emptyset$ then either Case 1 or Subcase 2a applies and we know then that G has a W_5-homeomorph.

Suppose there is a path Q in G from an internal vertex w_1 of P_1^H or P_3^H to a vertex w_2 of H not on

P_1^H or P_3^H where Q does not meet H except at its endpoints. Suppose w.ℓ.o.g. that w_1 is on P_1^H. Then G contains a W_5-homeomorph: Figure 4 shows essentially all the configurations of w_1, w_2 and Q relative to H, and in each case removal of the starred portion leaves a W_5-homeomorph. (Note that Q avoids $U_H(u)$ by definition of the latter.) From now on then suppose there is no such path Q. Thus, if $i \in \{1,3\}$, then P_i^H is either a single edge or is not in the same part of $G|W$ as any vertex in $H - P_1^H - P_3^H$.

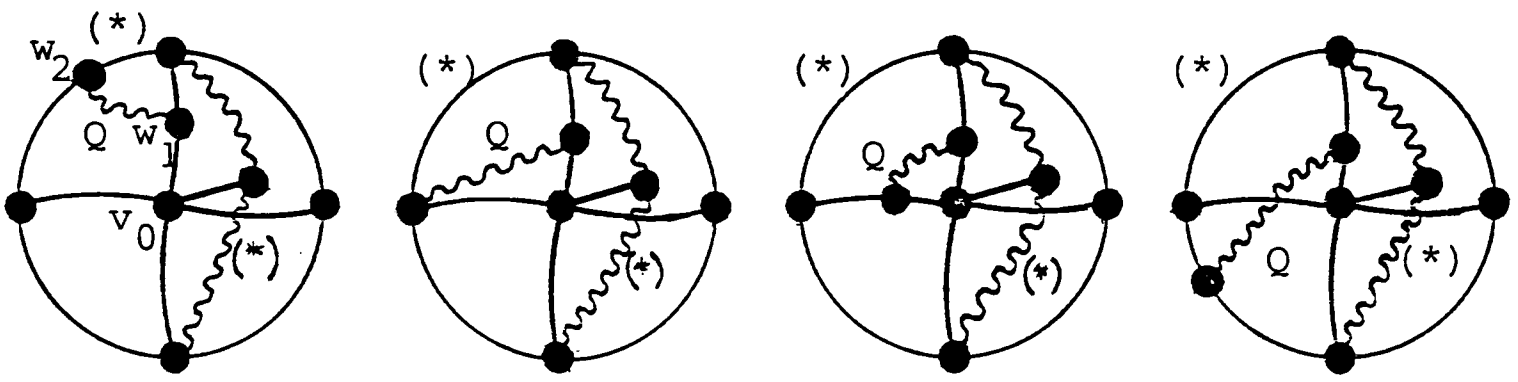


Figure 4: Diagrams illustrating the existence of W_5-homeomorphs for various configurations of H and Q in G, in Subcase 2b.

Now let H_2 and H_4 be the components of $H - P_1^H - P_3^H$ which contain v_2^H and v_4^H respectively. Thus H_2 (resp. H_4) consists of the path $P_2^H[v_2^H, v_0)$ $(P_4^H[v_4^H, v_0))$ and the path obtained by removing the endpoints of the v_1^H-v_3^H path in C which meets $v_2^H(v_4^H)$. Suppose there exists a path Q in G from a vertex w_1 of H_2 to a vertex w_2 of H_4 which avoids H except at its endpoints. Then once again it can be seen that G contains a W_5-homeomorph: Figure 5 shows the possible configurations, and in each case removal of the

starred portion leaves a W_5-homeomorph. (Note that Q avoids $U_H(u)$.) From now on then suppose there is no such path Q. Thus H_2, H_4 are in different parts of $G|W$, and we denote these parts by U_2 and U_4 respectively.

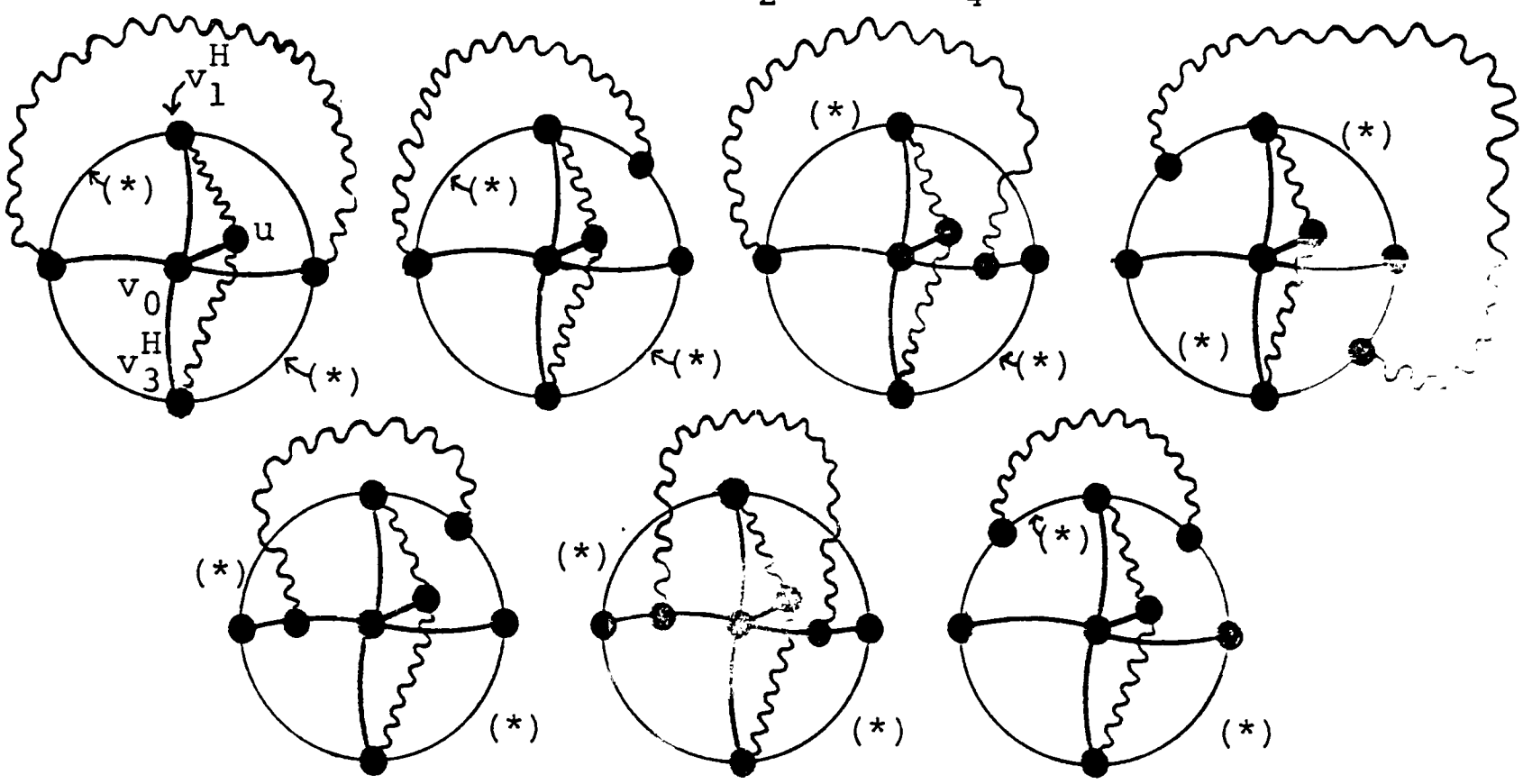


Figure 5: Diagrams illustrating the existence of a W_5-homeomorph when there is an H_2 - H_4 path, in Subcase 2b.

Now W is contained in a part of $G|W$ (namely any of U_2, U_4, $U_H(u)$); it follows that each part of $G|W$ contains some vertex of G not in W, and further that each part of $G|W$ contains W else 3-connectivity is violated. Now $G - v_0$ has a circuit D of size at least 5. If V(D) is contained in some part of $G|W$ then that part trivially contains at least two vertices not in W. On the other hand, if V(D) is not contained in any part of $G|W$, then D must meet v_1^H and v_3^H and the two parts of $D|\{v_1^H, v_3^H\}$ are each contained in a part of $G|W$. But since D has length at least 5, some part of $D|\{v_1^H, v_3^H\}$ has at least two vertices

other than v_1^H and v_3^H. Thus there is a part U^* of $G|W$ which contains at least two vertices not in W. Consider the set of edges

$$E' = \{xy | x \in W, \quad y \in U^* \backslash W\}.$$

Clearly E' is a cutset of G. Since each part of $G|W$ contains W, we know $|E'| \geq 3$. Now one component of $G - E'$ contains $U^*\backslash W$, which we know has at least two vertices, and the other component of $G - E'$ contains W; thus each component of $G - E'$ contains at least two vertices. This implies that if $|E'| = 3$ then E' is a proper 3-edge-cutset, contradictory to our assumptions. Hence $|E'| \geq 4$. Thus some member x of W has at least two neighbours, say w_1 and w_2, in $U^*\backslash W$. By 3-connectivity there exist two vertex-disjoint paths Q_1, Q_2 in $\langle U^* \rangle$, avoiding v_0, from w_1, w_2 respectively to the two members of $W\backslash\{x\}$. Furthermore there is a path R in $\langle U^* \rangle$ from some vertex w_1' on Q_1 to some vertex w_2' on Q_2 which avoids W and which meets Q_1, Q_2 only at its endpoints. It is clear that neither P_1^H nor P_3^H meets Q_1, Q_2 or R except at v_I^H and v_3^H; this is immediate for either of P_1^H, P_3^H which is a single edge, and otherwise follows from the first fact proved in this subcase. Now there are certainly two parts $U^{(1)}$, $U^{(2)}$ of $G|W$ other than U^*, since $G|W$ contains (at least) the three parts U_2, U_4 and $U_H(u)$. For each $i \in \{1,2\}$ we also know that $\langle U^{(i)} \rangle$ contains a vertex $u^{(i)}$ not in W and three paths $R_0^{(i)}$, $R_1^{(i)}$, $R_2^{(i)}$ from $u^{(i)}$ to v_0, v_1^H, v_3^H respectively which are vertex disjoint except at $u^{(i)}$. Now if $x = v_0$ then G contains the W_5-homeomorph centred on v_0 which consists of the paths

(2.3)

Q_1, Q_2, R, P_1^H, P_3^H, $R_0^{(1)}$, $R_1^{(1)}$ and $R_2^{(1)}$ and the edges v_0w_1, v_0w_2, as shown in Figure 6(a). If $x \neq v_0$, suppose w.ℓ.o.g. $x = v_1^H$ and observe that then G contains the W_5-homeomorph centred on v_0 consisting of the paths Q_1, Q_2, R P_1^H, $R_0^{(1)}$, $R_1^{(1)}$, $R_2^{(1)}$, $R_1^{(2)}$ and $R_2^{(2)}$ and the edges $v_1^Hw_1$, $v_1^Hw_2$, as shown in Figure 6(b). We remark that this W_5-homeomorph is not centred on v_0, but rather on v_1^H.

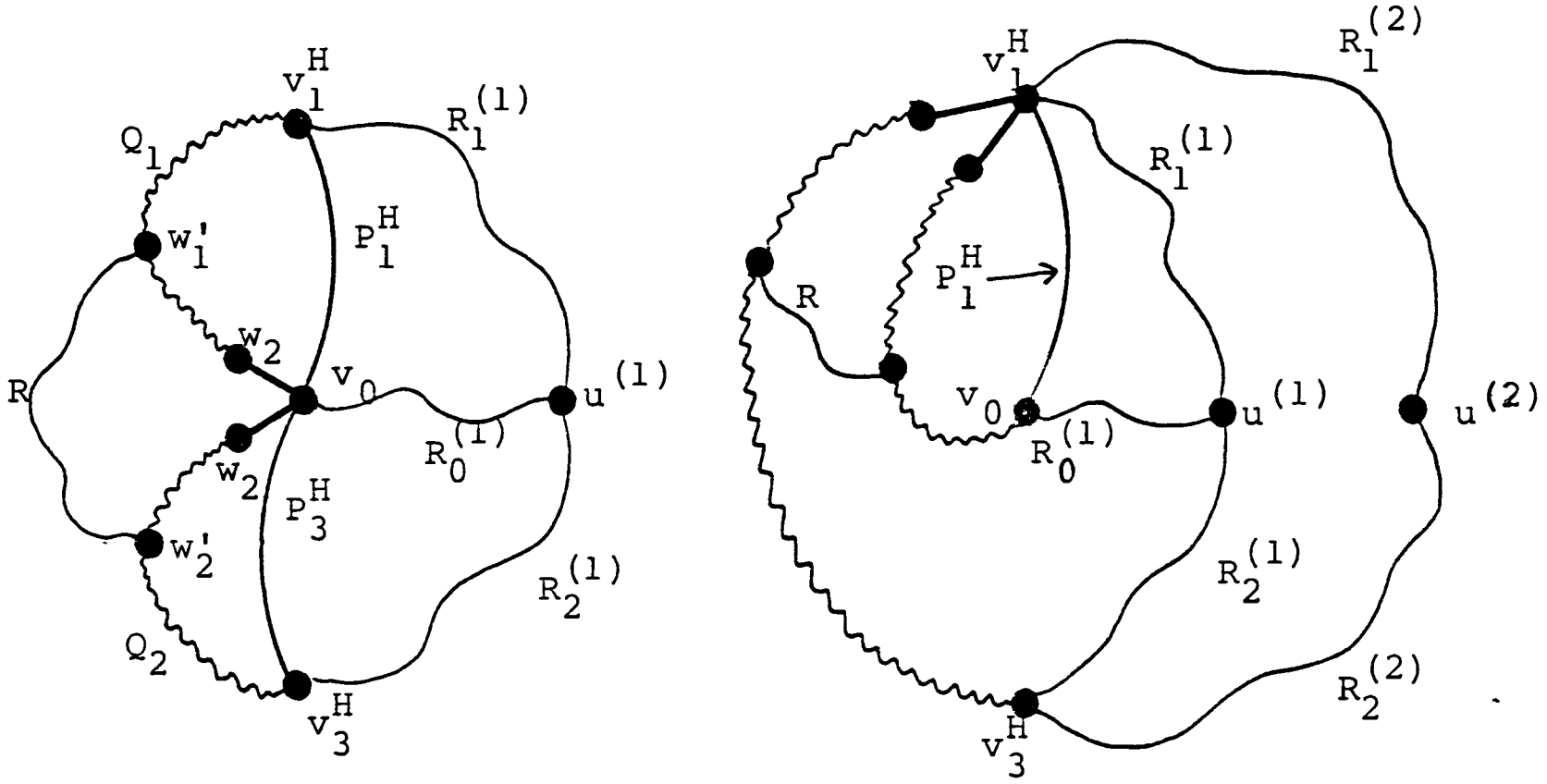


Figure 6: W_5-homeomorphs found in the final cases dealt with in the proof of Subcase 2b.

Case 3: G has a W_4-homeomorph H centred on v_0 and a vertex $u \in N_G(v_0) \setminus N_H(v_0)$ such that

$$U_H(u) \cap V_H \subseteq P_i^H \quad \text{for some} \quad i \in \{1,2,3,4\}.$$

Assume w.ℓ.o.g. that $U_H(u) \cap V_H \subseteq P_1^H$. Let $u_1(H,u)$ be the last vertex of $U_H(u) \cap V_H$ encountered in traversing P_1^H from v_0 to v_1^H. Assume that H, u are chosen to minimize

(2.3)

$$d_{P_1^H}(u_1(H,u), v_1^H) \ .$$

Let Q_0 be a path in $\langle U_H(u) \rangle$ from v_0 to $u_1(H,u)$ which is disjoint from H other than at its endpoints, and put $Q_1 = P_1^H[v_0, u_1(H,u)]$. (See Figure 7.)

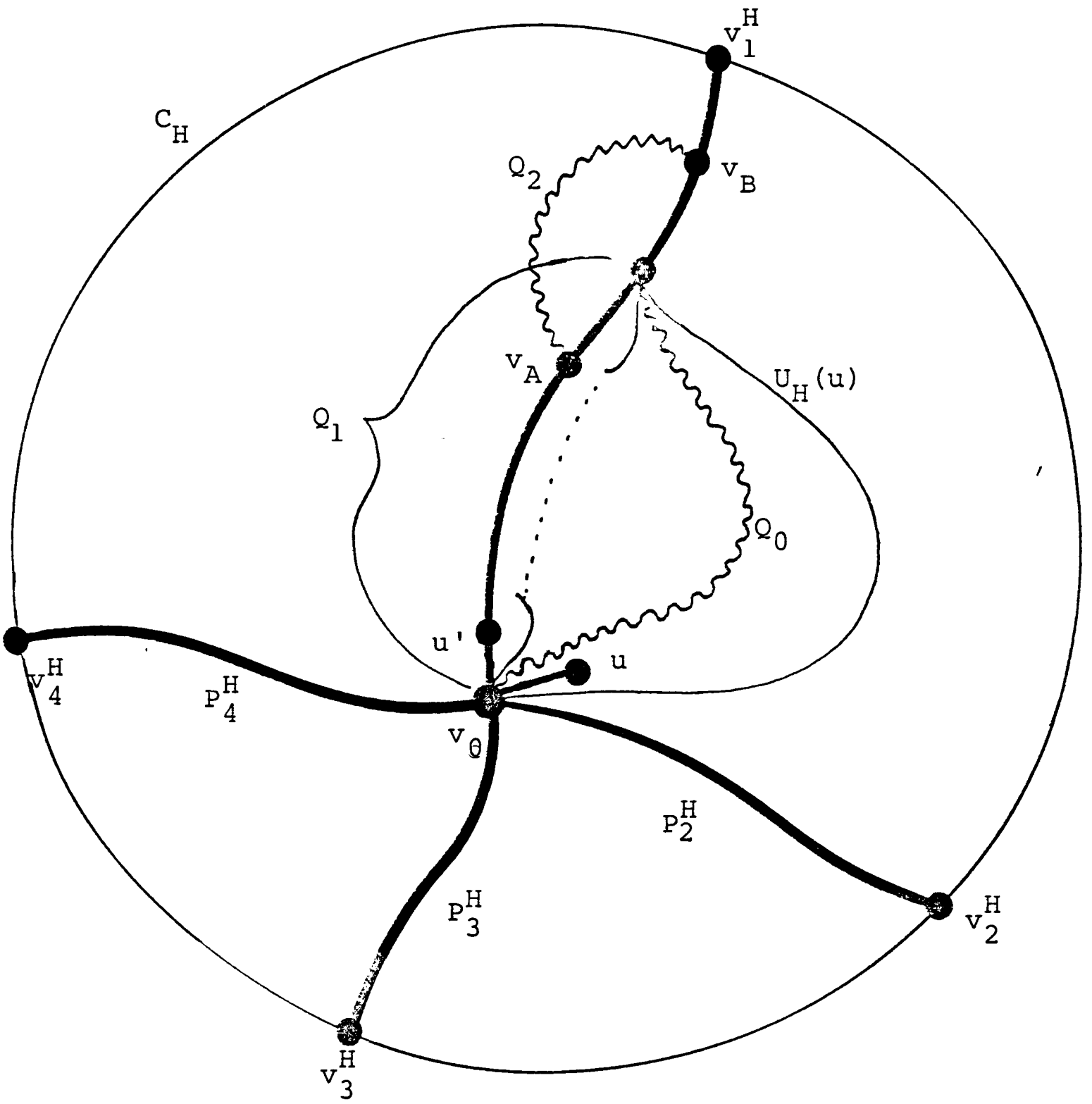


Figure 7: Diagram for Case 3 in the proof of Theorem 4.

Set

$$A = (U_H(u) \cup V(Q_1)) \setminus \{v_0, u_1(H,u)\},$$

$$B = V(H - Q_1) .$$

Observe that: $A \cap B = \emptyset$; $A \neq \emptyset$, else Q_0 and Q_1 are parallel edges; and $B \neq \emptyset$. Now removal of $\{v_0, u_1(H,u)\}$ cannot disconnect A from B, since G is 3-connected. Hence there is a path Q_2 from A to B which avoids $\{v_0, u_1(H,u)\}$ and which, it may be assumed, meets A and B each only once, in vertices v_A and v_B respectively. Suppose $v_A \in U_H(u) \setminus V(Q_1)$. Then Q_2 is contained in $\langle U_H(u) \rangle$ which implies $v_B \in U_H(u)$, a contradiction. Thus

$$v_A \in V(Q_1) \setminus \{v_0, u_1(H,u)\}.$$

(See Figure 7.) It follows in fact that the distance from v_0 to $u_1(H,u)$ along Q_1 is at least 2.

Let $L = (V_L, E_L)$ be the W_4-homeomorph obtained from H by replacing P_1^H with the $v_0 - v_1^H$ path, which we call P_1^L, consisting of Q_0 together with $P_1^H[u_1(H,u), v_1^H]$. We set $P_2^L = P_2^H$, $P_3^L = P_3^H$, $P_4^L = P_4^H$, $C_L = C_H$, so by definition $v_i^L = v_i^H$ for all $i \in \{1,2,3,4\}$. Define u' to be the vertex adjacent to v_0 on Q_1; it is not in L, by the last sentence of the previous paragraph, so $u' \in N_G(v_0) \setminus N_L(v_0)$. Clearly $v_B \in U_L(u')$. If $v_B \in C_L \setminus \{v_1^L, v_2^L, v_3^L, v_4^L\}$ then L, u' satisfy the hypotheses of Case 1 and so G has a W_5-homeomorph. If $v_B \in V(P_2^L) \cup V(P_3^L) \cup V(P_4^L)$ then L, u' satisfy the hypotheses of Case 2 and so G has a W_5-homeomorph. Thus we suppose that $v_B \in V(P_1^L)$. Since v_B is in B, it is on $P_1^L(u_1(H,u), v_1^L] = P_1^H(u_1(H,u), v_1^H]$. Thus $u_1(L,u')$, the

(2.3)

last vertex on P_1^L (going away from v_0) in $U_L(u') \cap V_L$, is on $P_1^L(u_1(H,u), v_1^L]$. Thus

$$d_{P_1^L}(u_1(L,u'), v_1^L) < d_{P_1^L}(u_1(H,u), v_1^L) = d_{P_1^H}(u_1(H,u), v_1^H),$$

contradicting the minimality of $d_{P_1^H}(u_1(H,u), v_1^H)$ (w.r.t. H,u).

It is not difficult to see that the three cases we have considered are exhaustive: in fact, for every H and every $u \in N_G(v_0) \setminus N_H(v_0)$, $U_H(u)$ must be of one of the three types described. We have shown that in each case G must contain a W_5-homeomorph, so the proof is complete. □

This Theorem forms the basis of a simple polynomial time algorithm for $SHP(W_5)$.

Theorem 5: $SHP(W_5)$ is solvable in polynomial time.

Proof: Consider the following polynomial time algorithm.

1. Input: Graph G.
2. If $|V(G)| \leq 5$, reject G.
3. If G is 3-connected, go to 4; otherwise, do steps 3.1 to 3.4 which we do not write out as they are exactly the same as the corresponding steps in the algorithm of Theorem 3 (although in 3.4 "the algorithm" must now be taken to refer to our algorithm here).
4. If G has no proper 3-edge-cutset, go to 5; otherwise:
   4.1. Find a proper 3-edge-cutset E' of G. Suppose $E' = \{e_1, e_2, e_3\}$.
   4.2. Let G_1, G_2 be the components of G - E'. For each j, $1 \leq j \leq 3$, let the endpoints of e_j in G_1, G_2 be u_j, v_j respectively. Form G_1'

(2.3)

from G_1 (resp. G_2' from G_2) by adding a single new vertex w_1 (w_2) adjacent to each of u_1, u_2, u_3 (v_1, v_2, v_3).

4.3. Apply the algorithm recursively to G_1' and G_2'. If either is accepted, accept G; otherwise, reject G.

5. If G has no vertex of degree at least 5, reject G; otherwise, continue.

6. For each vertex v of G of degree at least 5, apply the algorithm of Corollary 2.5 to determine whether $G - v$ has a circuit of length at least 5. If some such $G - v$ has such a circuit, accept G; otherwise, reject G.

It is straightforward to use Theorem 4 to verify that this algorithm recognizes $SHP(W_5)$. The argument is very similar to that of Theorem 3, except that the case in which G has a proper 3-edge-cutset has to be dealt with as well. This is not very different, however: a W_5-homeomorph H in G can only contain edges of such a cutset E' if either (i) some path of H leaves and returns to one component of $G - E'$ via E', or (ii) H contains one vertex of degree 3 in a different component of $G - E'$ from its other vertices of degree at least 3 (see Figure 8). The way we form G_1' and G_2' fully takes account of these possibilities, and so it is not difficult to see that G has a W_5-homeomorph iff at least one of G_1', G_2' does. □

(2.3)

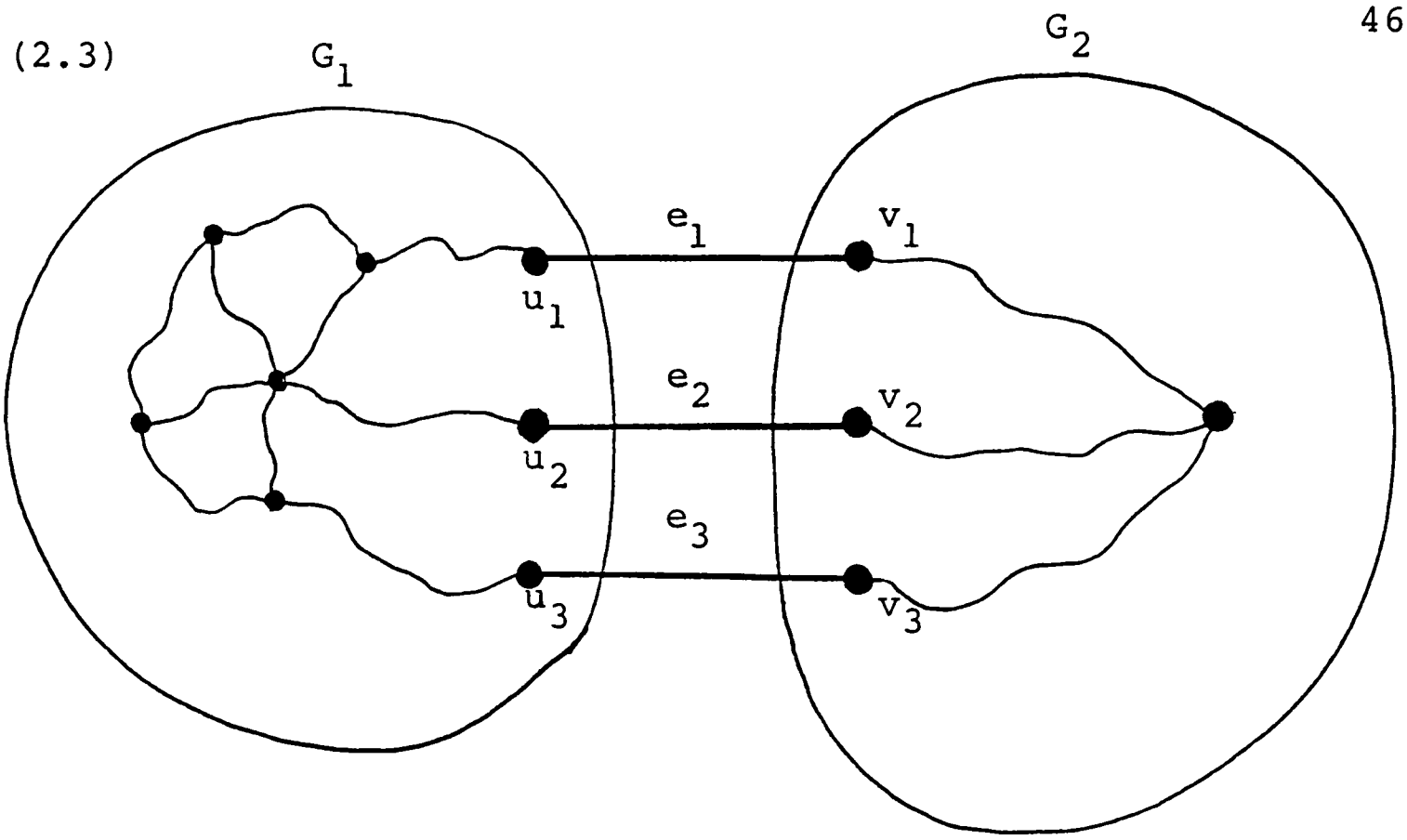


Figure 8

In going from the proof of Lemma 2, concerning W_4-homeomorphs, to the proof of Theorem 4, concerning W_5-homeomorphs, a considerable jump in difficulty occurs. As can possibly be appreciated, dealing with exclusion of W_6 as a homeomorph (at least by our sort of approach) becomes even more complicated again. There are a number of reasons for this which we now discuss. Firstly, an approach like ours would exploit the existence under suitable conditions of a W_5-homeomorph H and would investigate how the parts of $G|V(H)$ interact with H. We would expect to have rather more cases to deal with than the four cases we had using this kind of approach when H was a W_4-homeomorph in the proof of Theorem 4. Secondly, in proving Theorem 4 we made great use of the fact that G not only had a W_4-homeomorph, but a W_4-homeomorph centred on v_0, our chosen vertex of degree at least 5. In proving a similar theorem for W_6 we would expect to exploit the presence in a graph satisfying suitable conditions of a W_5-homeomorph. However, we may not be able to expect to find a W_5-homeomorph centred exactly where we please. Certainly a graph G satisfying the

conditions of Theorem 4 need not have a W_5-homeomorph centred on a given vertex v_0 of degree at least 5. An example illustrating this possibility is given in Figure 9; here there is no W_5-homeomorph centred on v_0.

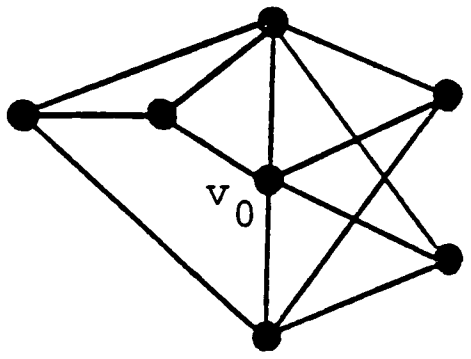


Figure 9

Finally, we expect there to be a need for an edge-connectivity condition of some sort, as was the case for Theorem 4. The form this may take however is unclear. This is though a minor problem. It is the two factors described above that would guarantee a prohibitive amount of case analysis in our approach. It would be of interest to find a similar result for W_6, however, and to find a better approach to dealing with such excluded homeomorphs for this and other pattern graphs.

§2.4. The complexity of counting homeomorphs

(This section is joint work with Colin McDiarmid. Most of it has been published in [15].)

In the previous three sections we have considered the problem of deciding for any graph whether it contains H as a homeomorph. This is now known to be in P for all H. We considered some particular graphs H and found elegant characterizations of graphs with no H-homeomorph, which led to simple polynomial time algorithms. We turn now to the associated enumeration problems and find the situation is very different. Theorem 1 tells us, essentially, that any problem of counting homeomorphs is $\#P$-complete. This result can be regarded as evidence of intrinsic difficulty [65, 66].

Everything in this section works if graphs are allowed to have loops or multiple edges, though this is not necessary.

If $\mathcal{H}$ is a set of graphs, then G is a homeomorph from $\mathcal{H}$ if G is homeomorphic from some graph H in $\mathcal{H}$. We are interested in the number of homeomorphs from $\mathcal{H}$ in a graph G, that is, the number of subgraphs of G that are homeomorphic from some member of $\mathcal{H}$. Our principal result concerns the following problem.

HOMEOMORPHS FROM $\mathcal{H}$

Input: Graph G.

Output: The number of homeomorphs from $\mathcal{H}$ in G.

A set $\mathcal{H}$ of graphs is nontrivial if there is a graph in $\mathcal{H}$ with at least one edge.

Theorem 1: For any finite nontrivial set H of graphs, HOMEOMORPHS FROM H is $\#$P-complete.

Proof: Let H be a finite nontrivial set of graphs.

Membership of $\#$P is straightforward to prove. Let M be the polynomial time NDTM which, on input G, guesses a subgraph of G homeomorphic from a member of H and then checks it. Then [M] is just HOMEOMORPHS FROM H, so the latter is in $\#$P.

One problem which has been shown to be $\#$P-complete [66] is the following.

S-T PATHS
Input: Graph G, specified distinct vertices s, t of G.
Output: The number of s - t paths in G.

We show

S-T PATHS $\propto_T$ HOMEOMORPHS FROM H .

In proving this Turing reduction, some intermediate problems will be used. For each $k = 0,1,2,...$ define a problem HOM(k) as follows.

HOM(k)
Input: Graph G, set A of edges and isolated vertices of G, with $|A| = k$.
Output: The number HOM(G,A) of subgraphs of G which are homeomorphs from H and include all elements of A.

Thus HOM(0) is essentially the problem HOMEOMORPHS FROM H.

Claim: There exists $m \geq 0$ such that S-T PATHS $\propto_T$ HOM(m).

Let us establish the Claim. For any graph G, let $n(G)$ be the number of vertices of degree not equal to 2 (where loops count twice). If G is homeomorphic from H, then of course $n(G) = n(H)$. Choose a graph H in $H \setminus \{\text{null graphs}\}$ with maximum value of $n(H)$ (say $n(H) = n$), and then maximum number of components.

Given a graph G with specified distinct vertices s and t, form a graph $\hat{G}$ as follows. Take a copy of H disjoint from G. Let u and v be endpoints of some edge e in H (where possibly $u = v$ if we allow loops). Delete e from H and add edges $\{u,s\}$ and $\{t,v\}$, thus forming a graph H'. Let A be the set of edges of H' together with any isolated vertices, and put $m = |A|$. Now add the vertices of G (other than s and t) and the edges of G to obtain $\hat{G}$ (see Figure 1).

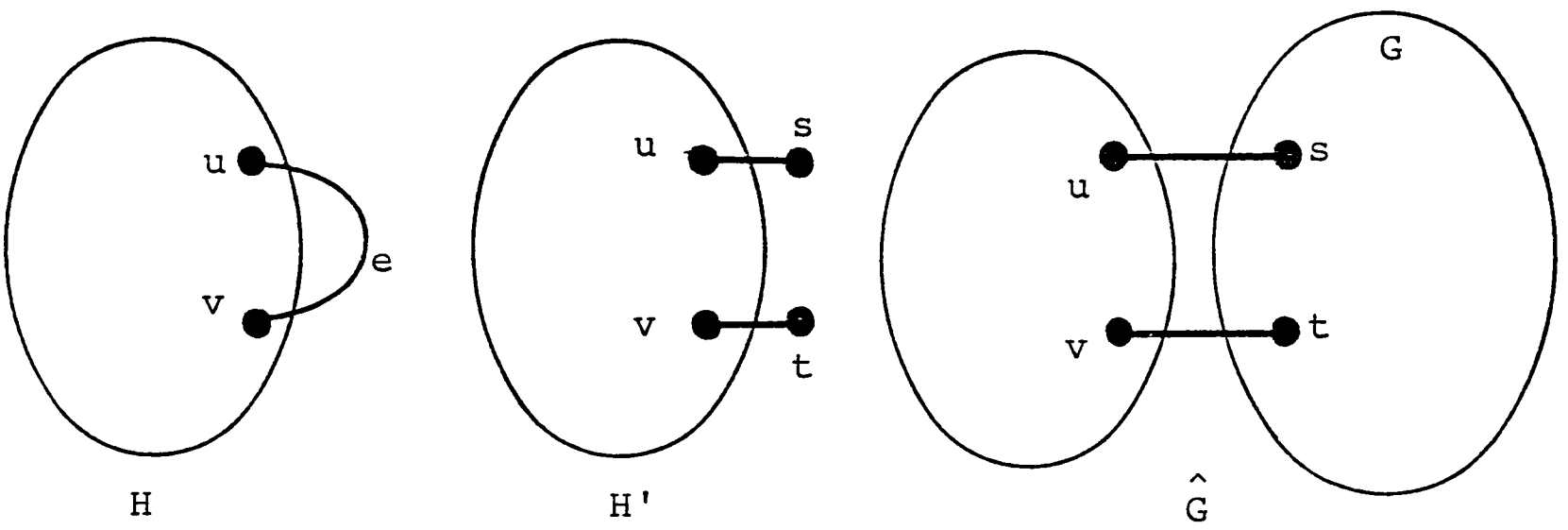


Figure 1

Let $\hat{H}$ be any subgraph of $\hat{G}$ which is a homeomorph from H and which includes all elements of A. Write $\hat{H} \cap G$ for the subgraph of $\hat{H}$ contained in G. Since $n(\hat{H}) \leq n$ and $n(H') = n + 2$, each vertex of $\hat{H} \cap G$ must have degree 2 in $\hat{H}$. Thus $\hat{H} \cap G$ has one component an

s - t path in G, and further it cannot have any other components by our choice of H. Hence, HOM($\hat{G}$,A) equals the number of s - t paths in G. The construction of $\hat{G}$ and A from G, s and t is clearly polynomial time. The Claim is therefore established.

The rest of the proof of the Theorem is straightforward. For if $x \in A$ then

$$HOM(G,A) = HOM(G,A\backslash\{x\}) - HOM(G-x,A\backslash\{x\}),$$

since every homeomorph from H in G which includes all of $A\backslash\{x\}$ (the total number of which is $HOM(G,A\backslash\{x\})$) either includes x, in which case it is counted by HOM(G,A), or does not include x, in which case it is counted by $HOM(G-x,A\backslash\{x\})$. Hence, for any $k \geq 1$,

$$HOM(k) \propto_T HOM(k-1).$$

Therefore,

$$HOM(k) \propto_T HOM(0)$$

and so, by the claim,

$$\text{S-T PATHS} \propto_T \text{HOMEOMORPHS FROM } H .$$

The Theorem is now proved. □

Some examples of applications of this Theorem to graph-theoretic counting problems follow.

Corollary 2: The problems of counting each of the following types of subgraph of a graph are all #P-complete:

(1) paths,

(2) circuits,

(3) minimal nonplanar subgraphs,

(4) minimal nonouterplanar subgraphs,

(5) minimal non-series-parallel subgraphs.

Proof: Apply Theorem 1, with (1) $H = \{K_2\}$, (2) $H = \{K_3\}$, (3) $H = \{K_5, K_{3,3}\}$, (4) $H = \{K_4, K_{2,3}\}$, (5) $H = \{K_4\}$. □

In the light of these results it is natural to ask about the complexity of counting planar subgraphs, or series-parallel subgraphs, of a graph. These are open problems.

Valiant [66] noted that counting Hamiltonian subgraphs of a graph is #P-hard for directed graphs. As a consequence of Theorem 1 we can prove #P-hardness for the undirected version. Firstly we name it.

HAMILTONIAN SUBGRAPHS

Input: Graph G.

Output: The number of Hamiltonian subgraphs of G.

Corollary 3: HAMILTONIAN SUBGRAPHS is #P-hard.

Proof: Given a graph G, form G' by subdividing each edge, by placing a new vertex on it. Then the circuits of G correspond precisely to the Hamiltonian subgraphs of G'. This shows that HOMEOMORPHS FROM $\{K_3\}$ $\propto_T$ HAMILTONIAN SUBGRAPHS. □

Neither the directed nor undirected Hamiltonian subgraphs problem is known to be in #P.

Theorem 1 also enables us to deal with a closely related family of counting problems involving homeomorphism. Again, let H be a set of graphs.

HOMEOMORPHS WITH H .

Input: Graph G.

Output: The number of subgraphs of G which are homeomorphic with a member of H .

Corollary 4: For any finite nontrivial set H of graphs, HOMEOMORPHS WITH H is $\neq$P-complete.

Proof: Let H be a finite nontrivial set of graphs.

It is clear that HOMEOMORPHS WITH $H \in \neq$P.

Now given any graph H there is a graph H^0 such that for any graph G, G is homeomorphic with H iff G is homeomorphic from H^0. Form the finite nontrivial set of graphs H^0 from H as follows:

$$H^0 = \{H^0 \mid H \in H\}.$$

By Theorem 1 HOMEOMORPHS FROM H^0 is $\neq$P-complete, and it is clear that HOMEOMORPHS FROM $H^0 \propto_T$ HOMEOMORPHS WITH H . □

We remark that all the results in this section apply to directed graphs with the natural notion of homeomorphism in that context.

A loose application to counting minors has been withdrawn.

(3.1)

CHAPTER 3

THE COMPLEXITY OF PROBLEMS WITH SHORT CERTIFICATES

§1. Introduction

Consider the problem of determining, for any graph G and integer k, whether G has a circuit of size at least k. It is well known that with no restriction at all on k, the problem is NP-complete, while if k is fixed then it is solvable in polynomial time. It is natural to ask what happens when k is a function of $|V(G)|$, intermediate between these extreme possibilities. For any function $f : \mathbb{N} \to \mathbb{N}$, we make the following definition.

$f(n)$-CIRCUIT

Input: Graph G.

Question: Does G have a circuit of size $\geq f(|V(G)|)$?

In practice we will want the function f to be nondecreasing and such that $f(n)$ is computable in time bounded by a polynomial in n (these requirements ensure that $f(n)$-CIRCUIT $\in$ NP); we call such a function good.

How does the complexity of $f(n)$-CIRCUIT depend on f, where f is good? It is in P if f is bounded above by a constant, and in fact a bit more can be said: Monien [41] has developed an algorithm which solves $f(n)$-CIRCUIT in polynomial time when $f(n) = O\left(\frac{\log n}{\log \log n}\right)$. At the other extreme, it is not difficult to show that if f is good, $f(n) \leq n$, and there exists k such that $f(n) \geq n^{1/k}$ for all n, then

f(n)-CIRCUIT is NP-complete. (The proof is by transformation from HAMILTONIAN PATH.)

See Figure 1.

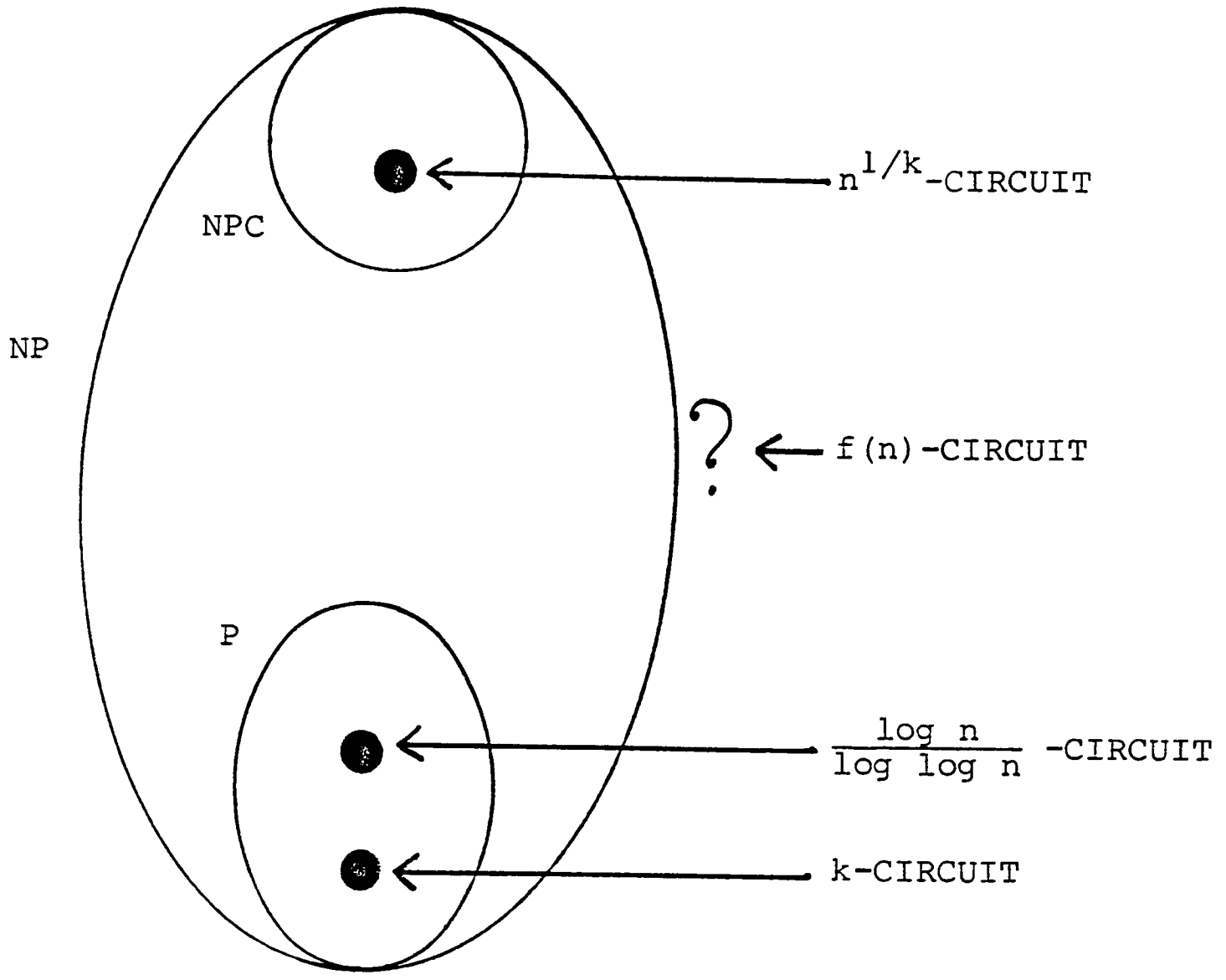


Figure 1: The dependence of the complexity of f(n)-CIRCUIT on the function f.

Suppose, however, that f is intermediate between these two extreme behaviours (example: $f(n) = (\log n)^k$, $k \geq 1$). We ask: for which f is f(n)-CIRCUIT in P, and for which f is it NP-complete? Indeed, must f(n)-CIRCUIT be in either of these classes? Ladner [36] has shown that if $P \neq NP$ then there exist languages in NP which are neither in P nor NP-complete. It seems plausible that some good function f might give rise to a language

f(n)-CIRCUIT which is in NP\(P ∪ NPC) if P ≠ NP (see Figure 1). Such a result would be of considerable interest, in that it would provide the first "natural" problem known to belong to NP\(P ∪ NPC) (if P ≠ NP). Ladner's languages are constructed by complicated diagonalization methods and could not really be termed natural.

In fact if we drop the requirement that f be non-decreasing, then on the assumption P ≠ NP it is possible to have f(n)-CIRCUIT in NP\(P ∪ NPC) with f(n) still computable in time polynomial in n. This is a consequence of

Theorem 1 [36,58]: If P ≠ NP then ∃ B ∈ P such that the membership or otherwise in B of a string x ∈ Σ* depends only on |x| and such that

$$(HAM \cap B) \cup B^c \in NP\setminus(P \cup NPC).$$ □

Suppose B is a language as in the Theorem. Define $f : \mathbb{N} \to \mathbb{N}$ by

$$f(n) = \begin{cases} n, & 1^n \in B; \\ 0, & 1^n \notin B. \end{cases}$$

Then f(n) is computable in time polynomial in n and clearly

$$f(n)\text{-CIRCUIT} = (HAM \cap B) \cup B^c$$

and so f(n)-CIRCUIT is in NP\(P ∪ NPC) (if P ≠ NP). However, all the languages B constructed by the proof of Theorem 1, and consequently the functions f defined as above, are again very unnatural, being constructed by the complicated diagonalization arguments of Ladner and others [36, 58].

Thus the goal remains of finding a natural good function f such that $f(n)$-CIRCUIT is neither in P nor is NP-complete, and of determining exactly how the complexity of $f(n)$-CIRCUIT depends on f.

We have focussed on $f(n)$-CIRCUIT here only because it is a convenient example for illustrating the questions that interest us. Similar questions could be asked for many other combinatorial problems, such as: "Does a graph G have a clique of size $f(|V(G)|)$?" We shall encounter a number of other such problems in this Chapter. Many such problems, including the examples we have mentioned, have short certificates: they can be solved by polynomial time NDTMs in which the number of nondeterministic steps is bounded by a slowly growing function of the input size. It is reasonable to hope that complexity classes defined in terms of such machines may be helpful in classifying the sorts of problems we consider. To this end we introduce the class $f(n)$-NP in the next section and completeness therein in §3. We then apply these concepts to a number of problems in §§3-5. As special cases of our results, we obtain in §5 some new log-space completeness results for P.

§2. The class f(n)-NP

Complexity classes defined in terms of restricted nondeterminism were introduced by Kintala and Fischer. They defined [32, 33] a class $P_{f(n)}$ as follows.

A language L is in $P_{f(n)}$ iff there exists a polynomial time NDTM which recognizes L and, for each input x, scans $\leq f(|x|)$ squares of the guess tape.

Kintala and Fischer observe that

(1) $$P = P_{\log n} \subseteq P_{(\log n)^2} \subseteq \cdots \subseteq P_{(\log n)^k} \subseteq \cdots$$

$$\cdots \subseteq P_n \subseteq P_{n^2} \subseteq \cdots \subseteq P_{n^k} \subseteq \cdots \subseteq NP,$$

and ask whether these inclusions are proper. Their main results concern relativized versions of these classes. They prove:

Theorem 1:

(a) For all $k \geq 0$ there is an oracle H depending on k such that

$$P^H \subsetneq P^H_n \subsetneq P^H_{n^2} \subsetneq \cdots \subsetneq P^H_{n^k} = P^H_{n^{k+1}} = P^H_{n^{k+2}} = \cdots = NP^H,$$

and there is another oracle H which makes all these inclusions proper.

(b) For all $k \geq 1$ there is an oracle H depending on k such that

$$P^H \subsetneq P^H_{(\log n)^2} \subsetneq \cdots \subsetneq P^H_{(\log n)^k} = P^H_{(\log n)^{k+1}} = \cdots = NP^H.$$

(c) For all $k \geq 1$, and for all oracles X, the class $P^X_{(\log n)^k}$ has a complete member, namely the language

$A_k(X) = \{(M,x,1^t) \mid M$ is a polynomial time NOTM with oracle X which, for all $y \in \Sigma^*$, scans at most the first $(\log|y|)^k$ squares on the guess tape, and some computation of M accepts x in at most t steps$\}$. □

The results (a) and (b) suggest that the question of whether or not the inclusions of (1) are proper is probably a very hard one. In (c) Kintala and Fischer exploit a standard technique for constructing complete members for many complexity classes to show that $P^X_{(\log n)^k}$ has a complete member. The result holds of course for the empty oracle, and easily generalizes to yield complete languages for $P_{f(n)}$ for any good f.

For several reasons which will shortly be explained, it is more natural and convenient to work with the class $f(n)$-NP, which we now define, rather than $P_{f(n)}$.

Definition: the class $f(n)$-NP. A language L is in the class $f(n)$-NP iff there exists a polynomial time NDTM M with binary guess tape alphabet and a polynomial q such that M recognizes L and, for each input x, M scans $\leq f(q(|x|))$ squares of the guess tape.

Thus,

$$f(n) - NP = \bigcup_{k \in \mathbb{N}} P_{f(n^k)} .$$

We now give three reasons to justify making this new definition rather than just working with $P_{f(n)}$.

Firstly, issues of how problems are coded into languages are avoided when working with $f(n)$-NP. For example, consider

the language HAM of graphs with a Hamiltonian circuit. If graphs are encoded as adjacency matrices (strings of length $|V(G)|^2$) then HAM $\in P_{\sqrt{n}.\log n}$. However, if graphs are encoded as adjacency lists (see, e.g., [1, p.51]) then this is no longer known to hold: one can say HAM $\in P_n$ but, with the present state of knowledge, not much more. (The adjacency list of G has size $\Theta(|E(G)|.\log|V(G)|)$, and a certificate of size $\sqrt{|E(G)|.\log|V(G)|}$ appears to be insufficient if, for example, $|E(G)| = |V(G)|^{3/2}$.) It is desirable that the complexity classification of a problem should not depend on how it is encoded (as long as a "reasonable" encoding scheme is used: see [18, pp. 19-23] for a discussion of this). This is indeed the case for f(n)-NP. In the case of graphs, for example, any sensible way of coding graphs which yields strings whose lengths are polynomially related to the number of vertices will do.

The second reason is as follows. The number of non-deterministic steps of a NDTM has so far been regarded as a function of the actual input length. For the purposes of determining whether a language L is in f(n)-NP, it is sufficient to use, instead of actual input length, any "length" function polynomially related to the length of the input string (see [18, p.20]). Thus, when dealing with graphs, the number of vertices (denoted by n) will frequently be used as the input length parameter.

Our third reason can be regarded as just a specialization of each of the above two reasons. Given $L \in P_{f(n^k)}$, we can define the language

$$L' = \{1^{|x|^k - |x|} x \mid x \in L\}.$$

Now $L' \in P_{f(n)}$ but it does not appear to be the case that in general $L \in P_{f(n)}$. Thus the complexity classifications of L and L' into the classes $P_{f(n)}$ are expected to differ. However, the languages themselves differ only trivially, L' being obtained from L by "padding" (to a polynomially bounded extent) the strings of L. It is certainly desirable that languages which are so similar should belong to the same complexity classes. This is indeed the case for the classes $f(n)$-NP.

We now make some elementary observations concerning $f(n)$-NP.

(2) $P = 0 - NP = k - NP = k.\log n - NP$ for all constants k.

(3) $NP = n - NP = n^{1/k} - NP = n^k - NP$ for all constants k.

For any f,

$$P \subseteq f(n) - NP \subseteq NP$$

(see Figure 1) and if g dominates f then

$$f(n) - NP \subseteq g(n) - NP.$$

The question of whether this inclusion is proper becomes non-trivial if g is large enough.

Open Question: Suppose $g(n)$ dominates $f(n^s)$ for all $s \geq 1$. Is it true that

$$f(n) - NP \subsetneq g(n) - NP \ ?$$

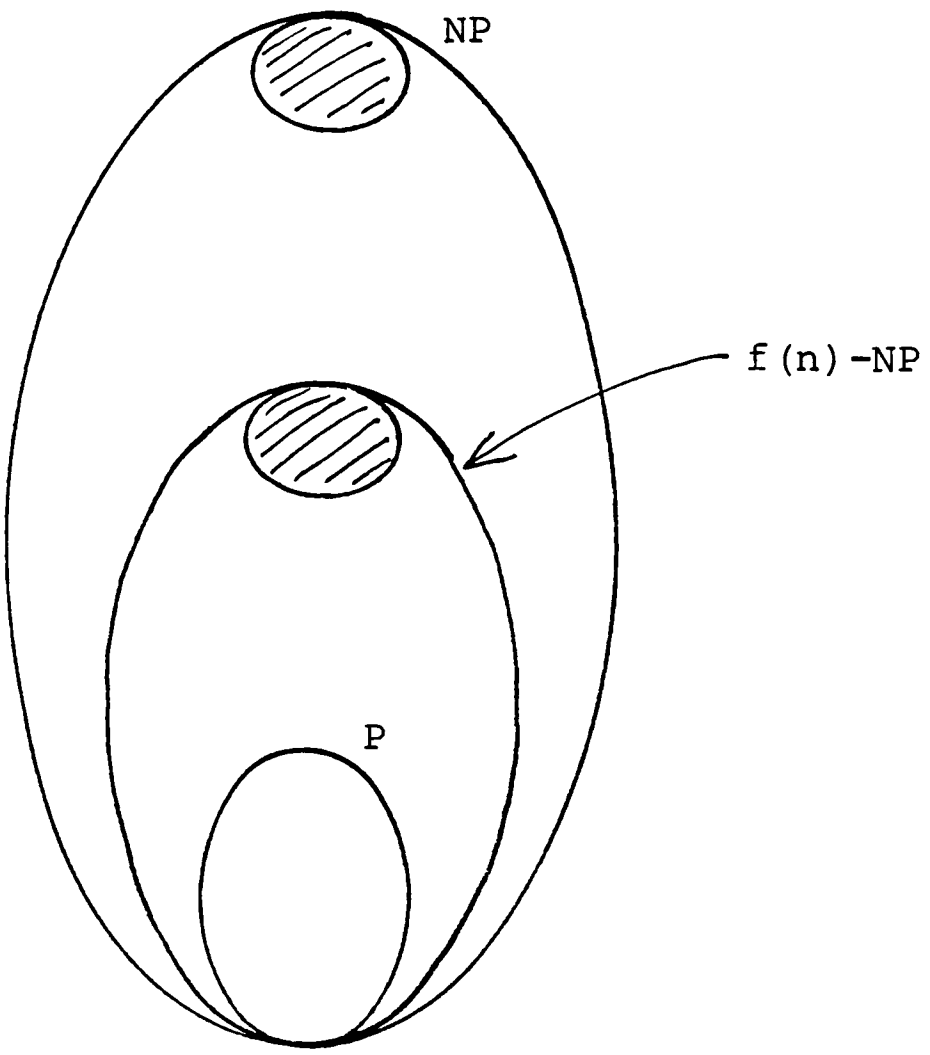


Figure 1: Diagram illustrating the postulated containment relationships between P, f(n)-NP and NP. Shaded regions indicate complete members of the associated class (see §3).

It appears very likely that the answer to this question is yes, so that significantly increasing the amount of non-determinism available will enable significantly more languages to be recognized. But an affirmative answer, for any particular f and g, would imply $P \neq NP$, so we will not tackle it.

One crucial mathematical difference between the classes f(n) - NP and the classes $P_{f(n)}$ is the following important elementary property of f(n)-NP, which is not known to hold for all the classes $P_{f(n)}$.

Lemma 2: Let f be good. Then if $L \in f(n) - NP$ and $L' \propto L$ then $L' \in f(n) - NP$.

Proof: Suppose $L \in f(n) - NP$. Then there is a polynomial time NDTM (with binary guess tape alphabet) M which recognizes L and a polynomial q such that on any input $x \in \Sigma^*$, M only scans $f(q(|x|))$ squares of the guess tape.

Let $\pi : \Sigma^* \to \Sigma^*$ be a polynomial transformation from L' to L. Then there exists a polynomial p' such that $|\pi(x)| \leq p'(|x|)$ for all x. Consider the following non-deterministic algorithm for L'.

1. Input: $x \in \Sigma^*$.
   Guess: $y \in \{0,1\}^*$.
2. Compute $\pi(x)$.
3. Run machine M on input $\pi(x)$ and guess y.
4. Accept x iff M halts in an accepting state.

This clearly gives a polynomial time NDTM M', and M' recognizes L', for

$$\begin{aligned} x \in L' &\iff \pi(x) \in L \\ &\iff \text{there is a string } y \text{ s.t. on input } \pi(x) \text{ and guess } y, M \text{ halts in an accepting state} \\ &\iff x \in L(M'). \end{aligned}$$

Furthermore, the number of squares of the guess tape read by M on input $\pi(x)$ is bounded above by $f(q(|\pi(x)|)) \leq f(q(p'(|x|)))$, so the algorithm M' demonstrates that $L' \in f(n) - NP$. □

This concludes our introductory remarks on the class $f(n) - NP$. In the next section we introduce the concept of $f(n)$-NP-completeness, with which the main results of this Chapter are concerned.

§3. Completeness in f(n) - NP

Completeness can be defined for f(n) - NP in the same way as for other complexity classes.

Definition: A language L is complete for f(n) - NP if $L \in f(n)$ - NP and, for all $L' \in f(n)$ - NP, $L' \propto L$. We will occasionally denote this by $L \in f(n)$ - NPC.

Completeness for f(n) - NP is really the most precise way we have of locating a language in the classes f(n) - NP, in the absence of any definite knowledge about when f(n) - NP is properly contained in g(n) - NP.

If f is good then the technique of Theorem 2.1(c) can be used to construct a complete language for f(n) - NP. Languages so constructed however are not very good "springboards" for obtaining completeness results for other more natural languages. We will follow a different route, using a generic transformation in the manner of Cook's Theorem [4] to get our first completeness result for f(n) - NP, and this will give a more useful springboard for further results. The main results of this Chapter are in fact f(n)-NP-completeness results. The question of whether f(n)-NP-complete languages are in $NP \setminus (P \cup NPC)$ remains (in general) open, and is likely to be difficult in view of the relativization results of Kintala and Fischer (see Theorem 2.1(b)). A proof of f(n)-NP-completeness could nevertheless be regarded as evidence of membership of $NP \setminus (P \cup NPC)$ (provided $P \neq NP$, and that, for all k, $n^{1/k}$ dominates f(n) and f(n) dominates k.log n).

For many of our f(n)-NP-completeness results, additional conditions on f are needed. We therefore make the following definitions.

f has <u>property P1</u> if either $f(n)=\Omega(n)$ or there exists $c<1$ such that $f(n)\leqslant cn$ for all sufficiently large n.

f has <u>property P2</u> if either $f(n)=\Omega(\log n)$ or $f(n)=O(\log n)$.

Although properties P1 and P2 are somewhat technical, most natural functions (including all considered by Kintala and Fischer) have both properties P1 and P2 in addition to being good: examples include $n^{1/k}$, $(\log n)^k$, $\underbrace{\log \log \ldots \log}_{k} n$, $\log n.\log \log n$ and so on. Thus, requiring that a good function f have property P1 or P2 (or both) is not a severe restriction, from the point of view of finding natural f(n)-NP-complete problems (and natural candidates for membership of $NP\setminus(P\cup NPC)$).

In passing we note that the existence of f(n)-NP-complete languages when f is good allows us to deduce the following trivial consequence of Lemma 2.2.

Lemma 1: If f is good then a language L is in f(n) - NP iff $L \propto L'$ for some f(n)-NP-complete language L'. □

To describe our first f(n)-NP-complete language we need a number of definitions.

Throughout this chapter ϕ denotes a Boolean expression in c.n.f., and $Var(\phi)$ is the set of variables of ϕ. A partial truth assignment for a Boolean expression ϕ is a function τ assigning truth values to some (possibly all; possibly none, when we may write $\tau = \emptyset$) of the variables of ϕ. A partial truth assignment τ is total if $\mathrm{dom}\ \tau = Var(\phi)$.

We now define the concept of forcing for partial truth assignments of a Boolean expression. This concept is not new. It was used as part of the Davis-Putnam (exponential time) Procedure [7] for solving SATISFIABILITY, and it also occurs in the definition of the language UNIT RESOLUTION (see the end of §7) which Jones and Laaser [30] have shown to be logspace complete for P.

Suppose $Var(\phi) = \{u_1,\ldots,u_n\}$ and that the clauses of ϕ are $C_1,\ldots,C_m$. Suppose τ is a partial truth assignment for ϕ. A forcing clause is a clause C of ϕ in which one literal, say $u_i' \in \{u_i,\bar{u}_i\}$, receives no truth value under τ and all other literals of C are False under τ; C is then also said to be a forcing clause for u_i' (and for $\bar{u}_i'$), since any extension of τ which satisfies ϕ must give u_i' the value T.

Here we can form a new partial truth assignment extending τ by simply adding the assignment of T to u_i' to the truth assignments made by τ. We can do this for every forcing clause of ϕ, and keep repeating the process until we are compelled to stop, when there are no more forcing clauses. The resulting partial truth assignment is denoted by τ^*.

We now define this repeated forcing procedure more formally, and take care of the possibility of contradictions arising which we overlooked in the informal definition of the preceding paragraph.

Suppose $C_{j_1}, C_{j_2}, \ldots, C_{j_\ell}$ are the forcing clauses of τ, and that for all h with $1 \le h \le \ell$, $u_{i_h}' \in \{u_{i_h}, \bar{u}_{i_h}\}$ is the unassigned literal in clause C_{j_h}. A partial truth assignment τ_1 is said to be <u>1-forced by τ</u> if

(1) $\text{dom } \tau_1 = \text{dom } \tau \cup \{u_{i_h} \mid 1 \le h \le \ell\}$,

(2) $\tau_1 \big|_{\text{dom } \tau} = \tau$,

and (3) $\tau_1(u_{i_h}) = \begin{cases} T, & u_{i_h}' = u_{i_h}, \\ F, & u_{i_h}' = \bar{u}_{i_h}. \end{cases}$

<u>k-forcing</u> is defined inductively: a partial truth assignment τ' is said to be <u>k-forced by τ</u> if there is a partial truth assignment τ'' such that τ' is 1-forced by τ'' and τ'' is $(k-1)$-forced by τ. τ' is said to be <u>forced by τ</u> if τ' is k-forced by τ for some k. τ is <u>contradictory</u> if there exists τ', forced by τ, such that some clause of ϕ has all literals False under τ'. If τ is contradictory then repeated 1-forcing may eventually yield a "function" which is not even well-defined: condition (3) of

(3.3)

the definition of 1-forcing may result in both values T and F being assigned to a single variable. However, if τ is noncontradictory, then for all k the partial truth assignment τ_k which is k-forced by τ is well-defined and unique. (Note though that τ being noncontradictory, does <u>not</u> imply that τ has an extension that satisfies ϕ.) Clearly

$$\tau \subseteq \tau_1 \subseteq \tau_2 \subseteq \cdots \subseteq \tau_k \subseteq \tau_{k+1} \subseteq \cdots$$

and, for some k, $\tau_k = \tau_{k'}$ for all $k' > k$, since ϕ is finite. This τ_k is the maximal partial truth assignment for ϕ forced by τ and we denote this by τ^*. If τ^* is total and satisfies ϕ we say τ <u>forces</u> a satisfying truth assignment for ϕ. (In this case we do know that τ has an extension that is total and satisfies ϕ.)

Our complete language can now be defined.

SF-SAT(f(n))

Input: Boolean expression ϕ in c.n.f. with m clauses, set U of $\leq$ f(m) specified variables of ϕ.

Question: Is there a partial truth assignment to the variables in U which forces a satisfying truth assignment for ϕ?

A comment on the name SF-SAT(f(n)). S indicates that the domain of the required partial truth assignment is <u>s</u>pecified; we will later encounter a similar problem in which this domain is <u>f</u>ree (denoted F), that is, not specified in the problem input. The second letter in the problem name, F, indicates that the problem involves <u>f</u>orcing.

Our result on this problem is the following.

Theorem 2: For any good function f, SF-SAT(f(n)) is f(n)-NP-complete. □

The proof rests on a close analysis of the proof of Cook's Theorem. It is rather long and detailed, and so for the sake of continuity we have postponed it to an appendix to this Chapter, §7.

Using Theorem 2, an f(n)-NP-completeness result concerning an analogue of 3SAT is now readily obtained. Let SF-3SAT(f(n)) consist of all members of SF-SAT(f(n)) in which the Boolean expression has exactly 3 literals in each clause.

Theorem 3: Apart from the cases where $f(n) \leq 1$ for all n, SF-3SAT(f(n)) is f(n)-NP-complete whenever f is good and has property P2. □

We have not given the proof of this Theorem; it uses a polynomial transformation from SF-SAT(f(n)) to SF-3SAT(f(n)) which is essentially identical to the standard transformation from SAT to 3SAT (see [18, pp.48-49]), although there are some routine technical details (such as the addition of "dummy" clauses, as happened at the end of the proof of Theorem 2, in §7).

The rest of this section, and all of the next two sections, are devoted to f(n)-NP-completeness proofs for a number of combinatorial problems. The transformations used, together with the two transformations already considered (Theorems 2 and 3), are shown diagrammatically in Figure 1 below. In this diagram, an arrow $A \rightarrow B$ indicates $A \propto B$, and the labels on the arrows refer to where the results are proved. Problem definitions may be found immediately preceding the

Theorems concerning their f(n)-NP-completeness. (The prefix FF is explained below.)

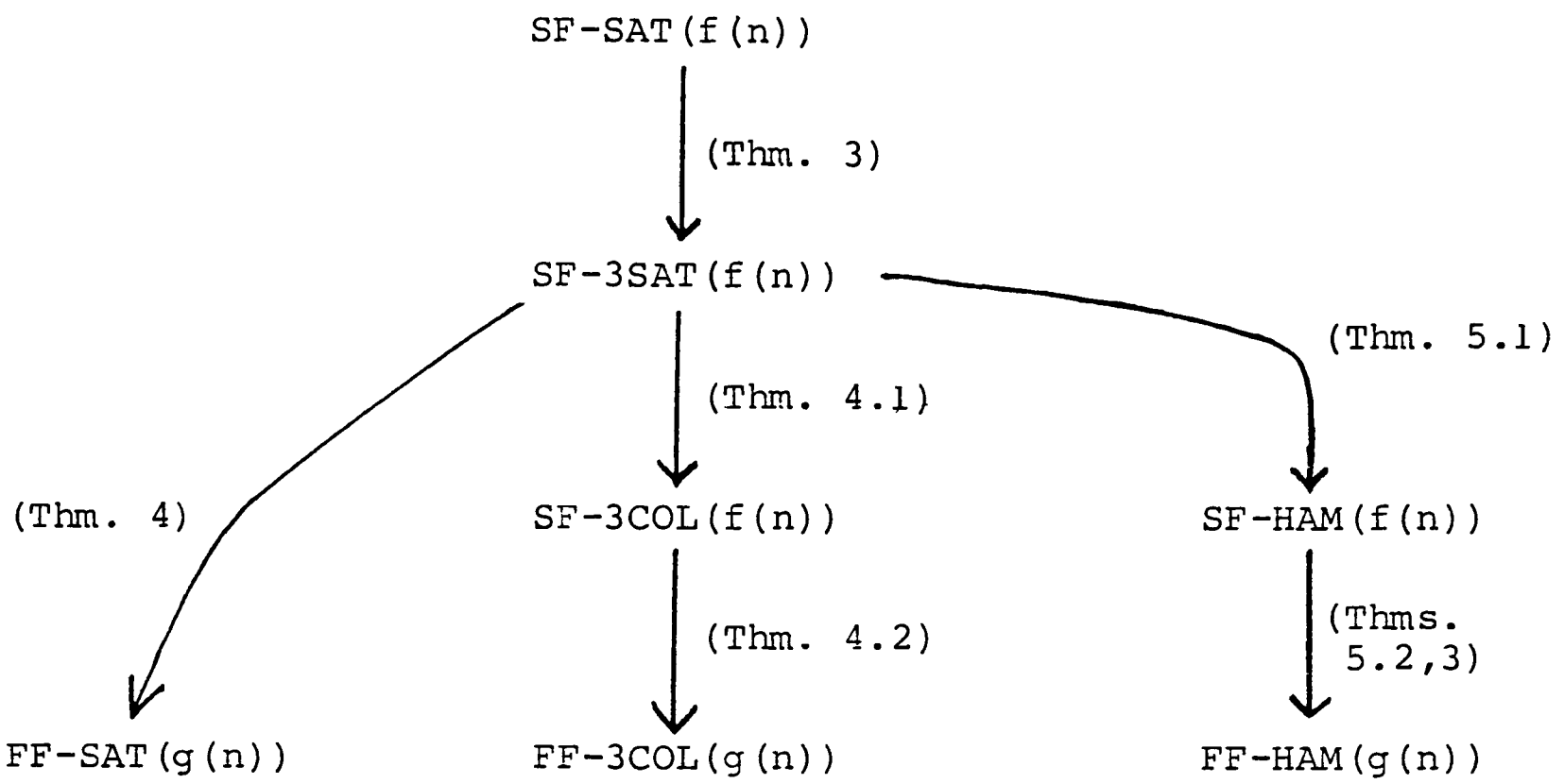


Figure 1: Transformations used for f(n)-NP-completeness results, where $f(n) = g(n)\log n$.

In the two problems considered so far, the domain of the required partial truth assignment has been specified in the input. Of greater interest, perhaps, are problems where the domain is not specified (i.e. is "free"). Such problems are rather more like the problems that originally motivated us, where we wanted to find a circuit (or clique, etc.) of a certain size and it did not matter where it was in the graph. We indicate that the domain is unspecified by an initial letter F (for free), rather than S.

Let us remove the specified variables from the input for SF-SAT(f(n)) and define

FF-SAT(g(n))

Input: Boolean expression ϕ in c.n.f. with m clauses.

Question: Is there a partial truth assignment to some g(m) variables of ϕ which forces a satisfying truth assignment for ϕ?

In the next Theorem we see that this problem is g(n)logn-NP-complete (if g is good). This result is not immediate, and the polynomial transformation given uses new techniques. Unfortunately there is also a lot of technical detail. This Theorem and its proof take up the rest of this section.

Theorem 4: If g is good and has property P1 then FF-SAT(g(n)) is g(n)logn-NP-complete.

Proof: Define, for all $n \in \mathbb{N}$,

$$L(n) = \lceil \log(n+1) \rceil .$$

Thus L(n) is the number of digits in the binary representation of n. Put $f(n) = g(n).L(n)$. Clearly

$$f(n) - NP = g(n)\log n - NP.$$

It will be convenient to speak of f(n) - NP rather than g(n)log n - NP.

We begin by proving that FF-SAT(g(n)) belongs to f(n) - NP. Let ϕ be a Boolean expression in c.n.f. with m clauses, and denote by $|\phi|$ the length of ϕ considered as a string in Σ^*. Our certificate for ϕ, to demonstrate membership of FF-SAT(g(n)), is a set of g(m) variables of ϕ and a truth value for each. ϕ has at most $|\phi|$ variables,

so each variable can be represented by a binary number with $\leq L(|\phi|)$ digits, and the truth value of each variable can be conveyed by just one digit. Thus our certificate for ϕ has length at most

$$g(m)(L(|\phi|) + 1) \leq g(2.|\phi|)\, L(2.|\phi|) = f(2.|\phi|).$$

Given such a partial truth assignment (call it t), we can form t* and check that t* is a satisfying truth assignment for ϕ, all in polynomial time.

In order to prove completeness, we consider the two cases arising from the definition of property P1. The first is easy to deal with, as follows.

Case 1: $g(n) = \Omega(n)$.

Here, f(n)-NP = NP so we are in fact proving NP-completeness.

Let c be a positive constant such that g(n) dominates cn.

We show

$$3SAT \propto FF\text{-}SAT(g(n)).$$

Let ϕ be a Boolean expression in c.n.f. with m clauses each of size exactly 3, and N variables. Let $k = \left\lceil \frac{N}{c} - m \right\rceil$. Form ϕ' by adding to ϕ the clauses $C_i = \{w_i\}$, $1 \leq i \leq k$, where the w_i are new variables not in ϕ.

If $\phi \in 3SAT$ then there exists a satisfying truth assignment t for ϕ. But t is then also a partial truth assignment for ϕ' such that t* is total and satisfies ϕ', and $|\text{dom } t| = N \leq g(m+k)$ (for sufficiently large m) by definition of k. Thus $\phi' \in FF - SAT(g(n))$.

Conversely, if $\phi' \in$ FF-SAT(g(n)) then there is a partial truth assignment t such that t* is total and satisfies ϕ'; but then t* restricted to Var(ϕ) demonstrates that $\phi \in$ 3SAT.

Hence $\phi \in$ 3SAT iff $\phi' \in$ FF-SAT(g(n)). The transformation is polynomial time since the number k of new singleton clauses added in forming ϕ' from ϕ is less than $\left\lceil \frac{3m}{c} - m \right\rceil$ (since $N \leq 3m$).

Case 2: There exists a constant $c<1$ such that $g(n) \leqslant cn$ for all sufficiently large n.

This case takes up the rest of the proof. We show

[1] SF-3SAT(f(n)) $\propto$ FF-SAT(g(n)).

Let ϕ be a Boolean expression in c.n.f. with m clauses each of size 3, and let $U \subseteq$ Var(ϕ) with $|U| \leq f(m)$. We suppose there exists an integer μ such that $|U| = \mu.L(m)$. If this is not actually the case, we can introduce $< L(m)$ dummy variables (not already occurring in ϕ), each appearing only in its own new singleton clause; the new expression so formed is equivalent, for our purposes, to the old one. Suppose w.ℓ.o.g. that

$$\text{Var}(\phi) = \{v_1, \ldots, v_N\}$$

and $$U = \{v_1, \ldots, v_{\mu L(m)}\},$$

with $N \geq \mu.L(m)$. Note that by assumption $\mu \leq g(m)$.

We show how to construct a Boolean expression ϕ' such that

$$(\phi, U) \in \text{SF-3SAT}(f(n)) \quad \text{iff} \quad \phi' \in \text{FF-SAT}(g(n)).$$

---

[1] Observe that if g is good then f has property P2 so Theorem 3 tells us that SF-3SAT(f(n)) is f(n)-NP-complete.

The idea is to introduce new variables and clauses such that the names of the new variables describe possible truth assignments to the variables in U. The construction proceeds as follows.

Introduce $\mu\hat{m}$ new variables $u_0, u_1, \ldots, u_{\mu\hat{m}-1}$, where $\hat{m} = 2^{L(m)}$. For each new variable u_i, do the following:

1. Find q, j such that $i = q\hat{m} + j$, $0 \le j < \hat{m}$, $0 \le q \le \mu - 1$. (Here, as shall be seen, the set of binary expansions of all j, $0 \le j < \hat{m}$, corresponds to the set of all possible truth assignments to the variables $v_{qL(m)+1}, \ldots, v_{(q+1)L(m)}$. In this way the u_i can be thought of as describing truth assignments to U.)

2. Express j in binary:

$$j = \delta_{L(m)}\delta_{L(m)-1} \cdots \delta_3\delta_2\delta_1 ,$$

where each $\delta_{j'} \in \{0,1\}$. So

$$j = \sum_{j'=1}^{L(m)} \delta_{j'} \cdot 2^{j'-1} .$$

3. Add to ϕ clauses with the meaning

$$u_i \rightarrow v^{(j)}_{qL(m)+1} \wedge v^{(j)}_{qL(m)+2} \wedge \cdots \wedge v^{(j)}_{(q+1)L(m)}$$

where, for each j', $1 \le j' \le L(m)$, the literal $v^{(j)}_{qL(m)+j'}$ is defined by

$$v^{(j)}_{qL(m)+j'} = \begin{cases} v_{qL(m)+j'}, & \delta_{j'} = 1; \\ \bar{v}_{qL(m)+j'}, & \delta_{j'} = 0 . \end{cases}$$

(See Figures 2 and 3.) This is achieved by adding to ϕ the clauses

$$\{\bar{u}_i, v^{(j)}_{qL(m)+j'}\}, \qquad 1 \le j' \le L(m).$$

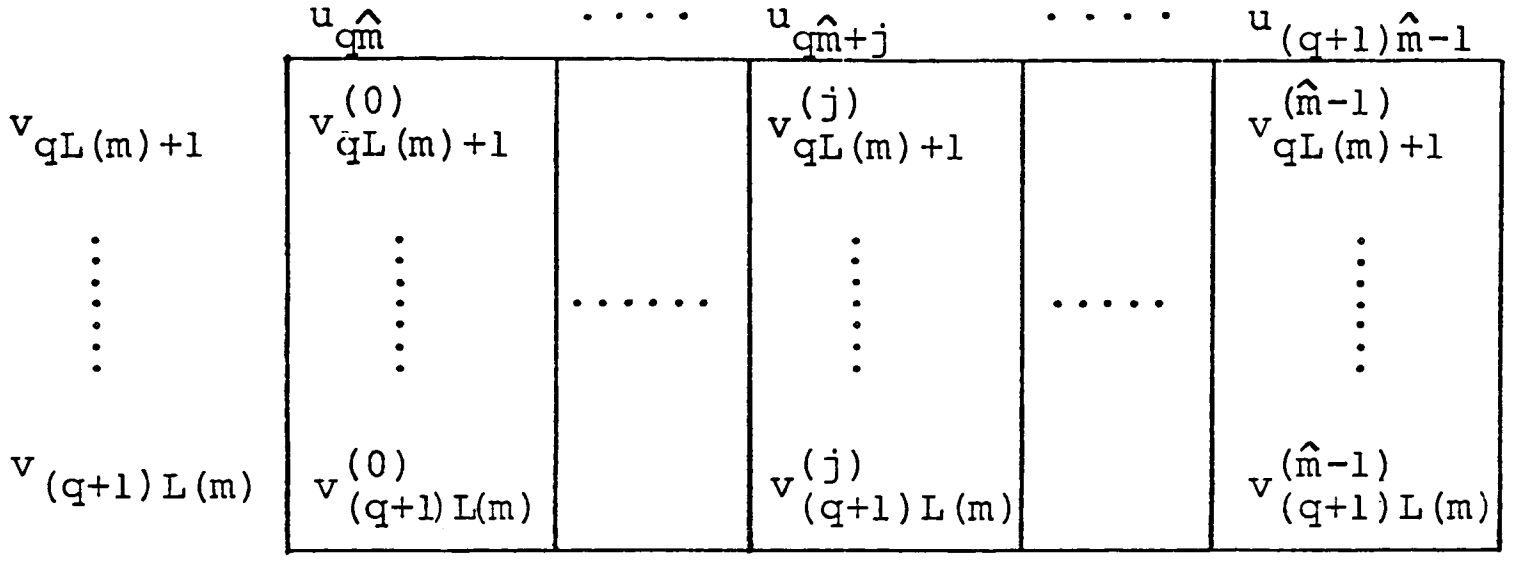


Figure 2: A table of variables and literals constructed in steps 1 to 3 (imagine one such table for each q, $0 \le q \le \mu - 1$). The column under $u_{q\hat{m}+j}$, consisting of the literals $v^{(j)}_{qL(m)+j'}$, encodes the binary expansion of j, in the manner described in Step 3.

	u_{8q}	u_{8q+1}	u_{8q+2}	u_{8q+3}	u_{8q+4}	u_{8q+5}	u_{8q+6}	u_{8q+7}
v_{3q+1}	$\bar{v}_{3q+1}$	v_{3q+1}	$\bar{v}_{3q+1}$	v_{3q+1}	$\bar{v}_{3q+1}$	v_{3q+1}	$\bar{v}_{3q+1}$	v_{3q+1}
v_{3q+2}	$\bar{v}_{3q+2}$	$\bar{v}_{3q+2}$	v_{3q+2}	v_{3q+2}	$\bar{v}_{3q+2}$	$\bar{v}_{3q+2}$	v_{3q+2}	v_{3q+2}
v_{3q+3}	$\bar{v}_{3q+3}$	$\bar{v}_{3q+3}$	$\bar{v}_{3q+3}$	$\bar{v}_{3q+3}$	v_{3q+3}	v_{3q+3}	v_{3q+3}	v_{3q+3}

Figure 3: A table of the variables and literals of the construction in the case $\hat{m} = 8$. The entry in the u_{8q+j} column and $v_{3q+j'}$ row gives the literal $v^{(j)}_{3q+j'}$. The u_{8q+j} column encodes the binary expansion of j : the $2^{j'-1}$ digit is 1 or 0 according as the literal $v^{(j)}_{3q+j'}$ is $v_{3q+j'}$ or $\bar{v}_{3q+j'}$.

4.[1] Add to ϕ clauses with the meaning -

"At most one of the $u_{q\hat{m}+j}$, $0 \le j \le \hat{m} - 1$, is true".

This is achieved by adding to ϕ the clauses

$$\{\bar{u}_{q\hat{m}+j_1}, \bar{u}_{q\hat{m}+j_2}\}, \quad 0 \le j_1 < j_2 \le \hat{m} - 1.$$

5. (Padding). Let m_1 be the number of clauses in the expression so far constructed. Let k be the least non-negative integer satisfying $g(m_1+k) = \mu + k$. That k exists and is bounded by a polynomial in m can be seen from the integrality of g, the nondecreasing nature of g, and the assumption $g(n) \leqslant cn$, and it is here that we make crucial use of the latter. For all i, $1 \le i \le k$, add the new variables u_i', u_i'' and the clause $\{u_i', u_i''\}$. Let $W = \{u_i', u_i'' \mid 1 \le i \le k\}$.

Denote by ϕ' the Boolean expression we have now constructed. (Figure 4 may be helpful in keeping track of the variables of ϕ'.) Let m' be the number of clauses of ϕ'; clearly $m' = m_1 + k$.

This completes the description of the mapping $(\phi, U) \mapsto \phi'$. It remains to show that this is indeed a polynomial transformation from SF-3SAT(f(n)) to FF-SAT(g(n)).

Denote by (*) the following assertion.

(*) t is a partial truth assignment for ϕ' such that $U \subseteq \operatorname{dom} t^*$.

Suppose t satisfies (*). We consider what truth assignments to the u_i are forced by t.

---

[1] This step is in fact unnecessary.

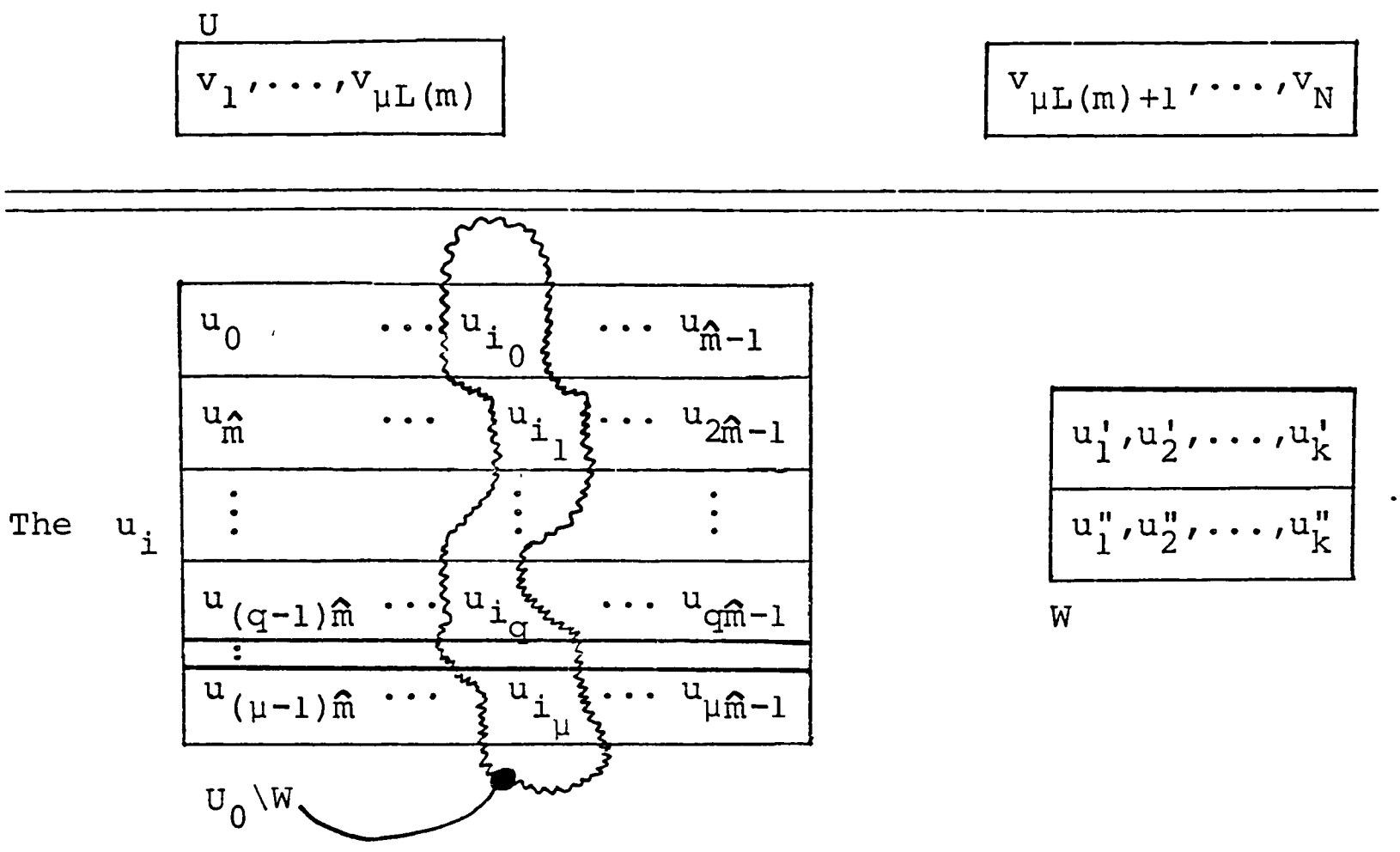


Figure 4: Table of variables for the polynomial transformation of the proof of Theorem 4.

Suppose $0 \le q \le \mu - 1$, and consider the truth values under t^* of the variables $v_{qL(m)+j'}$, $1 \le j' \le L(m)$. Let j be the number whose expression in binary is

(A) $\quad j = \delta_{L(m)}\ \delta_{L(m)-1} \cdots \delta_2\delta_1$ where

(B) $$\delta_{j'} = \begin{cases} 1, & t^*(v_{qL(m)+j'}) = T; \\ 0, & t^*(v_{qL(m)+j'}) = F\ . \end{cases}$$

(Thus the digits of j describe precisely the truth values under t^* of the variables $v_{qL(m)+j'}$, $1 \le j' \le L(m)$.) Then

(C) $$t^*\left(v^{(j)}_{qL(m)+j'}\right) = T \quad \forall j',\ 1 \le j' \le L(m).$$

Let $i = qm + j$. Now u_i is contained in the clauses $\{\bar{u}_i, v^{(j)}_{qL(m)+j'}\}$, none of which, by (C), can have been a forcing clause for u_i during the construction of t^* from t. The only other clauses of ϕ' containing u_i are those constructed in Step 4 above, namely $\{\bar{u}_i, \bar{u}_{i_1}\}$ for all $i_1 = q\hat{m} + j_1$, $0 \le j_1 < \hat{m}$, $i_1 \neq i$. Consider any such i_1. Since $j_1 \neq j$, the binary expansions of j and j_1 differ in some digit, say the j'-th digit from the right. Hence

$$t^*\left(v^{(j_1)}_{qL(m)+j'}\right) \neq t^*\left(v^{(j)}_{qL(m)+j'}\right) = T,$$

so $$t^*\left(v^{(j_1)}_{qL(m)+j'}\right) = F.$$

Thus the clause $\{\bar{u}_{i_1}, v^{(j_1)}_{qL(m)+j'}\}$ is forcing for u_{i_1}, so $t^*(u_{i_1}) = F$, so $\{\bar{u}_i, \bar{u}_{i_1}\}$ is not a forcing clause for u_i.

Thus no clause containing u_i is a forcing clause for u_i (at any stage of the construction of t^* from t). The following result follows immediately.

<u>Claim 1</u>: Suppose t satisfies (*), and for each $q \in \{0,1,\ldots,\mu-1\}$ let j be defined by (A) and (B). Write $i = q\hat{m} + j$.

Then: if t^* is total then $u_i \in \text{dom } t$.

Thus for any $q \in \{0,\ldots,\mu-1\}$, there exists $i_q = q\hat{m} + j_q$, $0 \le j_q < \hat{m}$, such that <u>if</u> t^* is total then $u_{i_q} \in \text{dom } t$, and such that (as shown above) in any case the following holds:

$$(D)\quad \begin{cases} t^*(v^{(j_q)}_{qL(m)+j'}) = T, & 1 \le j' \le L(m) \\ \\ t^*(u_i) = F, & i \ne i_q, \quad q\hat{m} \le i < (q+1)\hat{m}. \end{cases}$$

These truth values (D), taken for all q, tell us the truth values under t^* of every variable of ϕ' except the u_{i_q}, the u'_i and u''_i, and $Var(\phi)\backslash U$.

We are now in a position to prove that

$(\phi, U) \in$ SF-3SAT$(f(n))$ iff $\phi' \in$ FF-SAT$(g(n))$.

($\Longrightarrow$) Suppose t is any partial truth assignment to U such that t^* is total and satisfying for ϕ. t satisfies (*) so we apply the above argument to get the u_{i_q} as described and to deduce the values of t^* given in (D). Define

$$U_0 = \{u_{i_q} \mid 0 \le q \le \mu - 1\} \cup \{u'_i \mid 1 \le i \le k\}$$

and define t_0: $U_0 \to \{T,F\}$ by

$$t_0(u_{i_q}) = T \quad \forall q,$$

$$t_0(u'_i) = F \quad \forall i.$$

Considering t as a partial truth assignment for ϕ', we will prove that

$$\text{dom } t_0^* = \text{dom } t^* \cup U_0 \cup W$$

and that t_0^* and t^* agree outside $U_0 \cup W$.

Note firstly that $t_0^*(u''_i) = T \ \forall\, i$. Now take any q, $0 \le q \le \mu - 1$. Define j_q by $i_q = q\hat{m} + j_q$. Then the clauses $\{\bar{u}_{i_q}, v^{(j_q)}_{qL(m)+j'}\}$ force

$$t_0^*\left(v^{(j_q)}_{qL(m)+j'}\right) = T, \quad 1 \le j' \le L(m).$$

Now if $q' \neq q$, let j_1' be the number of digits to the right at which the binary expansions of j_q and $j_{q'}$ first differ. Then

$$t_0^* \left(v_{qL(m)+j_1'}^{(j_{q'},)} \right) = F$$

and so the clause $\{\bar{u}_{q\hat{m}+j_{q'}}, v_{qL(m)+j_1'}^{(j_{q'},)}\}$ forces $t_0^*(u_{q\hat{m}+j_{q'}}) = F$. Thus, for all $i'_q \neq i_q$, with $q\hat{m} \leq i_q' < (q+1)\hat{m}$, we have $t_0^*(u_{i_q'}) = F$. Thus

$$\text{dom } t_0^* = \text{dom } t^* \cup U_0 \cup W$$

and (referring to (D)) t_0^* and t^* agree outside $U_0 \cup W$.

Since t^* is total and satisfying for ϕ, so is t_0^*, and we have just seen that t_0^* is total and satisfying for the clauses of ϕ' not in ϕ. Thus t_0^* is total and satisfying for ϕ'. Observe that $|\text{dom } t_0| = \mu + k = g(m')$ (see Step 5). We conclude that $\phi' \in \text{FF-SAT}(g(n))$.

(<=) Now suppose t is a partial truth assignment for ϕ' such that t^* is total and satisfies ϕ' and $|\text{dom } t| \leq g(m')$. Then t certainly satisfies (*) so apply the argument following (*) to get the u_{i_q} as described and deduce the values of t^* given in (D). From Claim 1 applied to each i_q it follows that

$$\{u_{i_q} \mid 0 \leq q \leq \mu - 1\} \subseteq \text{dom } t.$$

Also one member at least of each of the clauses $\{u_i', u_i''\}$, $1 \leq i \leq k$, must be in dom t, so assume w.l.o.g. that

$$\{u_i' \mid 1 \leq i \leq k\} \subseteq \text{dom } t.$$

Thus $|\text{dom } t| \geq \mu + k = g(m')$. But $|\text{dom } t| \leq g(m')$ so in fact

$$\text{dom } t = \{u_{i_q} \mid 0 \leq q \leq \mu - 1\} \cup \{u'_i \mid 1 \leq i \leq k\},$$

and certainly $(\text{dom } t) \cap \text{Var}(\phi) = \emptyset$. Put $t_1 = t^*|_U$. The fact that t^* is total and satisfying for ϕ then implies that t_1^* is total and satisfying for ϕ. Obviously $|\text{dom } t_1| = |U| \leq g(m)$. Hence $(\phi,U) \in \text{SF-3SAT}(f(n))$.

In conclusion, we have exhibited a mapping $(\phi,U) \mapsto \phi'$ and shown that

$$(\phi,U) \in \text{SF-3SAT}(f(n)) \quad \text{iff} \quad \phi' \in \text{FF-SAT}(g(n)).$$

The mapping is polynomial time computable (see in particular the remarks on Step 5), and so is the required polynomial transformation. □

§4. Some colouring problems complete for f(n) - NP

We now turn to graph colouring and show that two graph colouring problems, both involving a natural concept of forcing, are f(n)-NP-complete. We concentrate on 3-colouring, although the results easily generalize to a greater number of colours.

In the proofs of Theorems 3.2 and 3.4 we have attempted to include all details necessary for full rigour. However, many such details are routine and vary little from one problem to another. For example, in constructing the image formulae in the polynomial transformations of Theorems 3.2 and 3.4, the number of clauses is increased, and since this is the parameter of f (or g) care must be taken. Usually it is necessary to pad the formula with dummy clauses (see the construction of ϕ' from ϕ towards the end of the proof of Theorem 3.2 in §7, and Step 5 in the construction of the proof of Theorem 3.4). Having considered the details in full in the previous section, we will usually pass over them from now on. We will also omit proofs of membership of f(n)-NP where such membership is clear.

Informally, our forcing procedure for 3-colouring is just to find an uncoloured vertex with two differently coloured neighbours, give it the colour not used by its neighbours, and repeat this operation for as long as possible. We will use this concept of forcing again in §5.1.

Formally, let G be a graph. A <u>partial 3-colouring</u> of G is a 3-colouring of an induced subgraph of G. (We stress that, in a 3-colouring of a graph G, every vertex receives a colour and adjacent vertices receive different colours.)

Let c be any partial 3-colouring of G, using colours 1, 2 and 3 say, and let $U \subseteq V(G)$ be the domain of c. The basic forcing step is to find a vertex $v \notin U$ adjacent to some two vertices $v_1, v_2 \in U$ such that $c(v_1) \neq c(v_2)$, and then to colour v by the single element of $\{1,2,3\} \setminus \{c(v_1), c(v_2)\}$. Thus we form a new partial 3-colouring of G which extends c. If by repeating this basic forcing step we can obtain a partial 3-colouring of G in which a vertex u receives colour k, then we say c *forces* u (*to receive colour* k). If a partial 3-colouring c' of G extends c, then we say c *forces* c' if for every $v \notin U$, v is forced to receive colour $c'(v)$ by c. If c forces a partial 3-colouring c' in which some uncoloured vertex of G is adjacent to three differently coloured vertices then c is said to be *contradictory*. The following remarks are clear.

(a) If c forces c' then any 3-colouring of G which extends c must extend c' as well.

(b) If c has an extension which is a 3-colouring of G then c is not contradictory.

(c) The converse to (b) does not necessarily hold.

Finally we write c^* for the maximal partial 3-colouring of G forced by c, and observe:

(d) If c is not contradictory then c^* is well-defined and unique; the converse is not necessarily true.

With these definitions the next f(n)-NP-complete problem and its proof of completeness may be given.

SF-3COL(f(n))

Input: Graph G on n vertices, set $U \subseteq V(G)$ where $|U| \leq f(n)$.

Question: Does there exist a partial 3-colouring c of G with domain U such that c forces a 3-colouring of G?

Theorem 1: Apart from the cases where $f(n) \leq 1$ for all n, SF-3COL(f(n)) is f(n)-NP-complete whenever f is good and has property P2.

Proof: Membership of f(n)-NP is clear.

We show

$$\text{SF-3SAT}(f'(n)) \propto \text{SF-3COL}(f(n))$$

where f' is defined by

$f'(n) = 0$ for all n, if $f(n) = O(\log n)$;

and $f'(n) = f(n)$ for all n, otherwise.

Firstly we describe some special graphs which will be used in this polynomial transformation. From now on we will always work with the three colours R, B and G.

These graphs will contain 3 special vertices labelled R, B, G. (These will be shown as squares in diagrams.) The colours they receive will be given by their labels.

The graph $G(w_1,w_2,w_3)$, where w_1,w_2,w_3 are vertices, is shown in Figure 1.[1]

We now establish some crucial properties of $G(w_1,w_2,w_3)$. The vertex set of $G(w_1,w_2,w_3)$ is denoted by V_G. Throughout, c is a partial 3-colouring of this graph with $c(R) = R$, $c(B) = B$ and $c(G) = G$. For any such 3-colouring c, define

[1] $\{w_1,w_2,w_3\}$ will in fact be a separating set for (those graphs we construct which contain) $G(w_1,w_2,w_3)$.

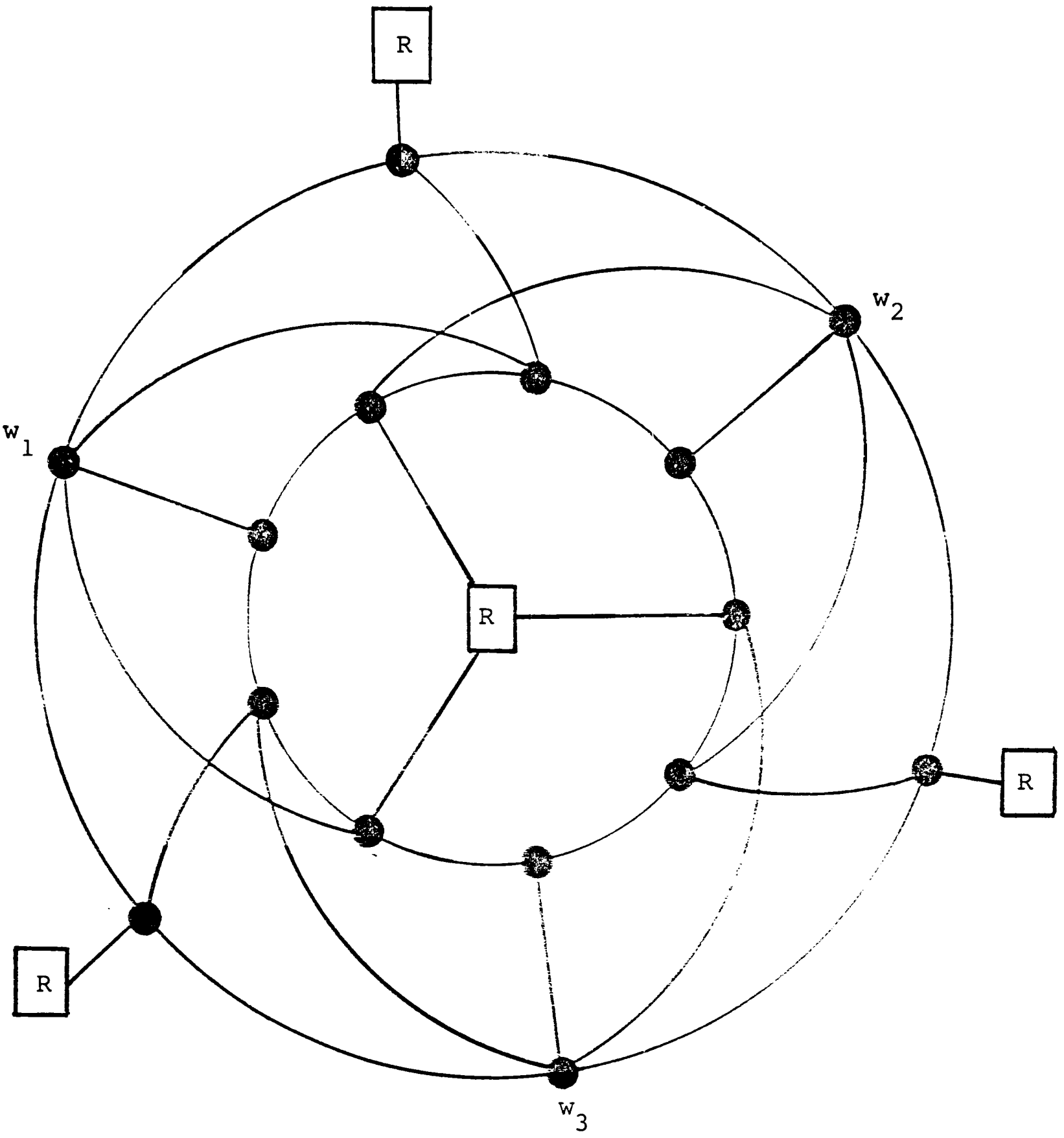


Figure 1: The graph $G(w_1,w_2,w_3)$.

$$\mathrm{dom}_0 c = (\mathrm{dom}\ c)\backslash\{R,B,G\}.$$

(1) Let $i_3 \in \{1,2,3\}$ and let $\{i_1,i_2\} = \{1,2,3\}\backslash\{i_3\}$. Suppose

$$(\mathrm{dom}_0 c) \cap V_G = \{w_{i_1}, w_{i_2}\}$$

and $c(w_{i_1}) = c(w_{i_2}) = G$. Then $V_G \subseteq \mathrm{dom}\ c^*$ and $c^*(w_3) = R$.

Proof: Suppose without loss of generality that $(i_1,i_2,i_3) = (1,2,3)$. Let c be such a colouring of w_1 and w_2. The result is easily seen by consulting Figure 2: it shows, in order, the vertex colourings which are forced by previously assigned colours, beginning with c. The numbering of the vertices indicates the order in which they are coloured, and it can be seen by inspection that the colour given to vertex labelled i is forced by c and the colouring of vertices j where $j < i$.

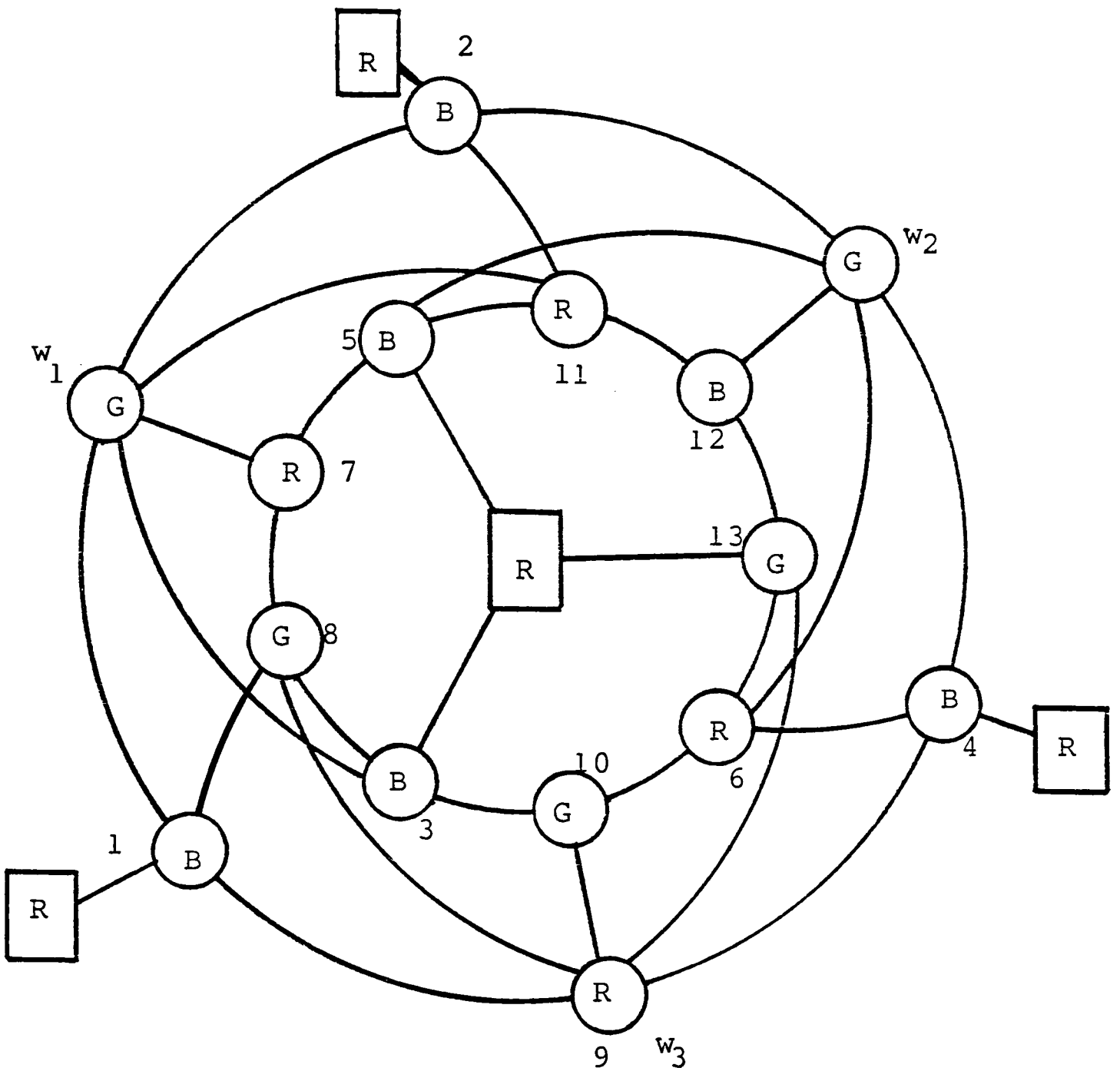


Figure 2: Colouring of $G(w_1,w_2,w_3)$ illustrating the proof of (1).

(2) If $dom_0 c \cap V_G = \emptyset$ then $dom_0 c^* \cap V_G = \emptyset$.

Proof: It is clear that in $G(w_1,w_2,w_3)$ with no colours assigned other than vertex R receiving colour R, no other vertex has a forced colouring.

(3) If $dom_0 c \cap V_G = \{w_i\}$ where $i \in \{1,2,3\}$ and $c(w_i) = G$ then

$$dom_0 c^* \cap \{w_1,w_2,w_3\} = \{w_i\}.$$

Proof: Easily seen.

$H(v_1,v_2,v_3)$ is the graph obtained from $G(w_1,w_2,w_3)$ by the construction given in Figure 3. Its vertex set is denoted by V_H.

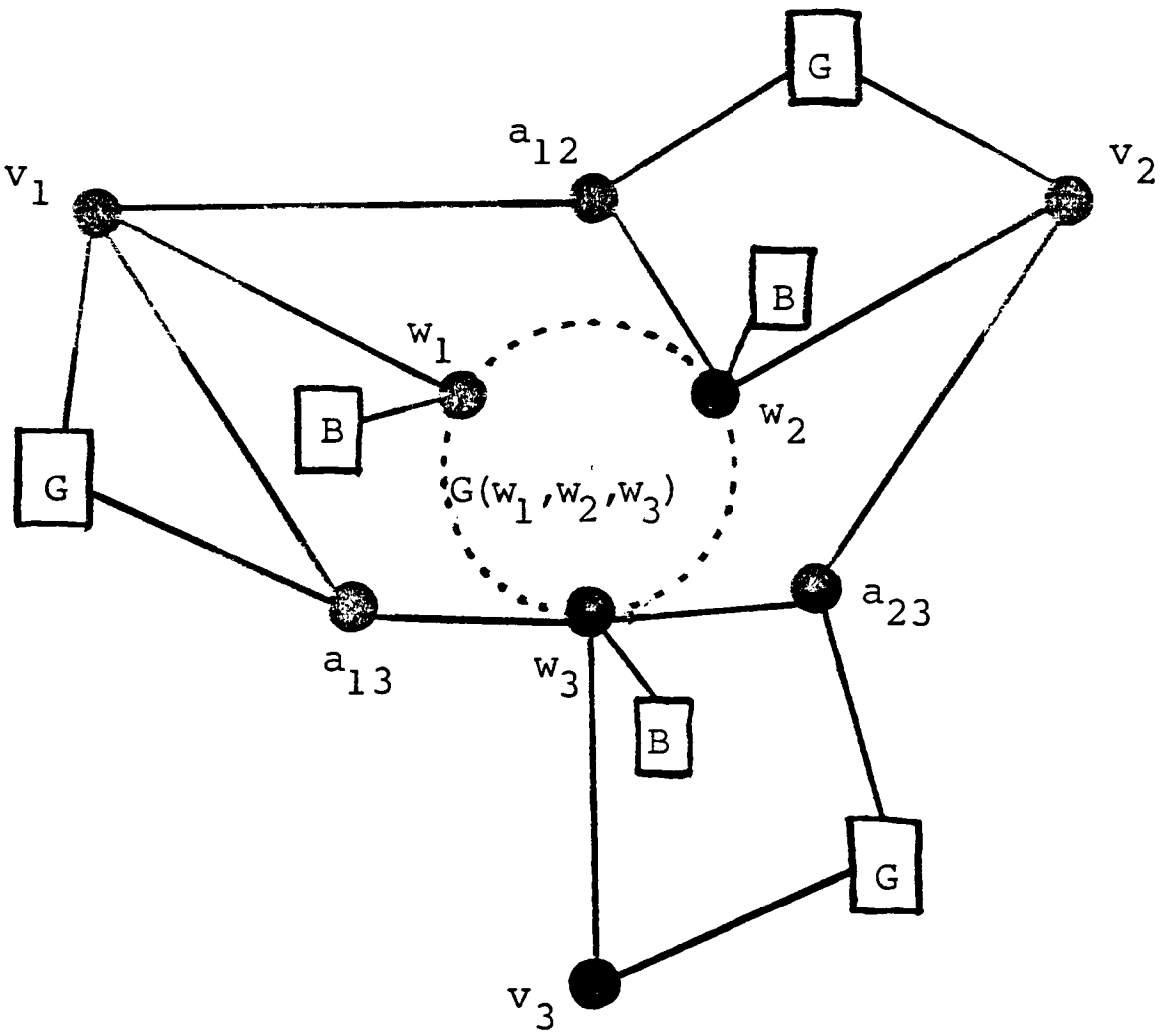


Figure 3: Construction of $H(v_1,v_2,v_3)$ from $G(w_1,w_2,w_3)$.

(4) If two of v_1, v_2, v_3 receive colour R under c, then $V_H \subseteq$ dom c*, and the third of the v_i receives colour B under c*.

Proof: $c(v_1) = R, c(v_2) = R$ forces $c^*(w_1) = G$, $c^*(w_2) = G$ respectively. By (1), $V_G \subseteq$ dom c*, and $c^*(w_3) = R$ which forces $c^*(v_3) = B$. $c(v_1) = R$ forces $c^*(a_{12}) = c^*(a_{13}) = B$, and $c(v_2) = R$ forces $c^*(a_{23}) = B$.

Arguments of similar style (but not identical, due to the asymmetry of H) take care of the cases $c(v_1) = c(v_3) = R$ and $c(v_2) = c(v_3) = R$.

(5) If $c(v_1) = c(v_2) = c(v_3) = R$, then c is contradictory.

Proof: This follows immediately from (4).

(6) If $c(v_1) = c(v_2) = c(v_3) = B$, then $V_H \subseteq$ dom c*.

Proof: Routine checking, and use (1).

(7) If, under c, precisely one of v_1, v_2, v_3 receives colour R and the other two receive colour B, then $V_H \subseteq$ dom c*.

Proof: Again, routine checking using (1).

(8) If $\text{dom}_0 c \cap V_H \subseteq \{v_1, v_2, v_3\}$, and c satisfies

(i) some v_i is not in dom c

and (ii) at most one of the v_i receives colour R under c,

then $V_H \not\subseteq$ dom c*.

Proof: Observe that the only way w_i can receive a colour under c* is if $c^*(v_i) = R$ forces $c^*(w_i) = G$, or

$c^*(w_j) = G$ for each $j \neq i$ forces $c^*(w_i) = R$. (This uses (1), (2), (3).)

The rest of the proof is once again a matter of routinely checking the claim for the relevant colourings of $\{v_1, v_2, v_3\}$, making use of (1), (2), (3).

We have so far proved, from (4) to (8) above:

(9) Suppose $dom_0 c \cap V_H \subseteq \{v_1, v_2, v_3\}$. Then

(i) c^* is contradictory (in $H(v_1, v_2, v_3)$) iff $c(v_1) = c(v_2) = c(v_3) = R$.

(ii) c^* colours all vertices in V_H iff:

(a) c colours all the v_i with $c(v_j) = B$ for some j,

or (b) c colours two of the v_i with colour R and does not colour the remaining one.

(iii) c^* leaves some vertex of $H(v_1, v_2, v_3)$ uncoloured iff c leaves some v_i uncoloured and the other two do not both receive colour R.

The polynomial transformation may now be given.

Let ϕ be a Boolean expression in c.n.f. with m clauses each of size 3, and let U be a subset of $Var(\phi)$ with $|U| \leq f'(m)$.

Construct the graph $G(\phi)$ as follows. Firstly form a triangle T from three vertices labelled R, B, G (Figure 4). Secondly, for each variable u of ϕ, form the triangle whose vertices are the literals u, $\bar{u}$ and the vertex G of T. (These triangles are the truth-setting components; see Figure 5. Colours B, R correspond to truth values T, F respectively, on the vertices u, $\bar{u}$.) Thirdly, for each

clause C_j, consisting say of the literals x, y and z, form the graph H(x,y,z) where all the vertices R, B, G of this graph are identified with the appropriate vertex of T. Thus, the graph

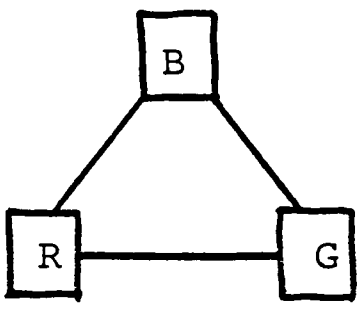


Figure 4

The triangle T.

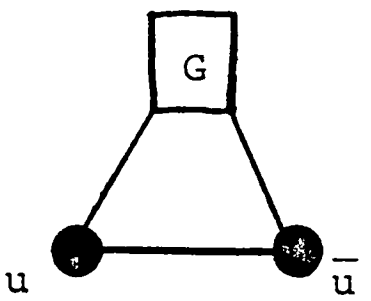


Figure 5

The truth-setting component for variable u.

H(x,y,z) meets the rest of $G(\phi)$ only in the vertices R, B, G and x, y, z. If $f(n) = O(\log n)$, this completes the construction of $G(\phi)$. Otherwise, we do some padding as follows. Let n_0 be the number of vertices in the graph constructed so far, and suppose k is fixed and satisfies $f(k) \geq 2$. Add $(k - 1)n_0$ new vertices to the graph, each of which is adjacent only to vertices R and B. The construction of $G(\phi)$ is then complete.

Set

$$V_0 = \{u \in V(G(\phi)) \mid u \in U\} \cup \{R,B\},$$

so V_0 consists of the vertices of $G(\phi)$ labelled by variables in U, plus {R,B}. Put $n_\phi = |V(G(\phi))|$.

If $f(n) = O(\log n)$, $f'(n) \equiv 0$ so $U = \emptyset$ and $V_0 = \{R,B\}$. Hence

$$|V_0| = 2 \leq f(n_0) = f(n_\phi)$$

(if n_0 is sufficiently large).

On the other hand, if $f(n) = \Omega(\log n)$ then (recalling the padding above) $n_\phi = kn_0$, so

$$f(n_\phi) = f(kn_0) \geq f(k) + f(n_0) \quad \text{since} \quad f(n) = \Omega(\log n)$$

$$\geq f(n_0) + 2$$

$$\geq f(m) + 2 \quad \text{as} \quad m \leq n_0$$

$$\geq |U| + 2$$

$$= |V_0|.$$

Thus, in any case,

(A) $|V_0| \leq f(n_\phi)$.

The mapping $(\phi, U) \mapsto (G(\phi), V_0)$ is our polynomial transformation. It is clear that it is polynomial time computable, so it remains to show that $(\phi, U) \in$ SF-3SAT$(f(n))$ iff $(G(\phi), V_0) \in$ SF-3COL$(f(n))$.

Claim 1: Suppose that t is a partial truth assignment to variables $W \subseteq \mathrm{Var}(\phi)$, and that c is a partial 3-colouring of $G(\phi)$ with domain W, such that for all $w \in W$,

(B) $$\begin{cases} c(w) = B \text{ iff } t(w) = T, \\ \\ c(w) = R \text{ iff } t(w) = F. \end{cases}$$

Then t is contradictory iff c is contradictory, and $t^* = c^*|_W$.

Proof of Claim 1: Suppose $n = |\mathrm{Var}(\phi)|$ and $N = |W|$. We prove the result by induction on $n - N$. If $n = N$, then t and c are total on W (regarding W as the domain of both t and c), so $t^* = t$ and $c^*|_W = c$, so by assumption

$t^* = c^*|_W$. Also t is contradictory iff c is contradictory, since the subgraphs $H(-,-,-)$ (defined above; see Figure 3) of $G(\phi)$ simulate the clauses of ϕ. (We use (9) (i) here.)

Now suppose the hypotheses of the Claim are satisfied with $n \neq N$. If there are no forcing clauses of ϕ for t then the result is easy, since $t^* = t$ and $c^*|_W = c$ and, by the argument above, t is contradictory iff c is contradictory. Suppose then that ϕ has a forcing clause for t, say the clause $\{x',y',z'\}$ containing literals $x' \in \{x,\bar{x}\}$, $y' \in \{y,\bar{y}\}$, $z' \in \{z,\bar{z}\}$, where $x,y,z \in Var(\phi)$. Suppose w.l.o.g. that x' and y' are F under t and $z \notin W$. Then t forces the partial truth assignment t_1, with domain $W \cup \{z\}$, defined by

$$t_1(u) = t(u) \qquad \forall\, u \in W;$$

$$t_1(z) = \begin{cases} T, & z' = z, \\ F, & z' = \bar{z}\ . \end{cases}$$

t is contradictory iff t_1 is contradictory, and $t^* = t_1^*$. Now define the partial 3-colouring c_1, with domain $W \cup \{z\}$, by

$$c_1(u) = c(u) \qquad \forall\, u \in W;$$

$$c_1(z) = \begin{cases} B, & z' = z, \\ R, & z' = \bar{z}\ . \end{cases}$$

Since $G(\phi)$ contains the subgraph $H(x',y',z')$, with x', y' coloured R under c, it follows that c forces c_1, and hence that c is contradictory iff c_1 is contradictory,

and $c^* = c_1^*$. Now t_1 and c_1 satisfy (B) for all $w \in W \cup \{z\}$, so by the inductive hypotheses (applicable since $n - |W \cup \{z\}| < n - N$), Claim 1 tells us that t_1 is contradictory iff c_1 is contradictory, and $t_1^* = c_1^*|_{W \cup \{z\}}$. Hence t is contradictory iff c is contradictory, and $t^* = c^*|_W$, and the Claim is proved.

The proof of the Theorem may now be completed. Suppose $(\phi,U) \in$ SF-3SAT(f(n)). Then there is a partial truth assignment t to U such that t^* is total and satisfies ϕ. Define the partial 3-colouring c on V_0 by

$$c(u) = \left\{\begin{array}{ll} B, & t(u) = T \\ R, & t(u) = F \end{array}\right\} \qquad \forall\, u \in U;$$

$$c(R) = R, \quad c(B) = B.$$

By Claim 1, c^* is total on $Var(\phi)$, and for every three literals x', y', z' such that $H(x',y',z')$ is a subgraph of $G(\phi)$, one of these literals is coloured B under c^*. By (9)(ii) applied to each $H(-,-,-)$, c^* is total on $G(\phi)$ and so is a 3-colouring of $G(\phi)$. We have seen (A) that $|V_0| \leq f(|V(G(\phi))|)$, so

$$(G(\phi),V_0) \in \text{SF-3COL}(f(n)).$$

Conversely, suppose $(G(\phi),V_0)$ belongs to SF-3COL(f(n)). Then there is a partial 3-colouring c with domain V_0 such that c^* is a 3-colouring of G. We assume w.ℓ.o.g. that $c(R) = R, c(B) = B$, so that $c^*(G) = G$. Define $t: U \to \{T,F\}$ by

$$t(u) = \left\{\begin{array}{ll} T, & c(u) = B; \\ F, & c(u) = R\ . \end{array}\right.$$

By Claim 1, t is noncontradictory since c is, and $t^* = c^*|_U$. Thus t^* is total since c^* is. By (9)(i), every clause of ϕ has some literal which receives value T under t, so t^* satisfies ϕ. Thus $(\phi,U) \in$ SF-3SAT(f(n)), and the Theorem is proved. □

We remark that the forcing properties of the graph $G(-,-,-)$ of Figure 1 - see (1) and its corollary (4) - are crucial to this proof, and it is the need for these properties that makes use of this graph necessary rather than, say, the standard satisfaction testing component in the standard reduction 3SAT $\propto$ 3COL [19] (see Figure 6).

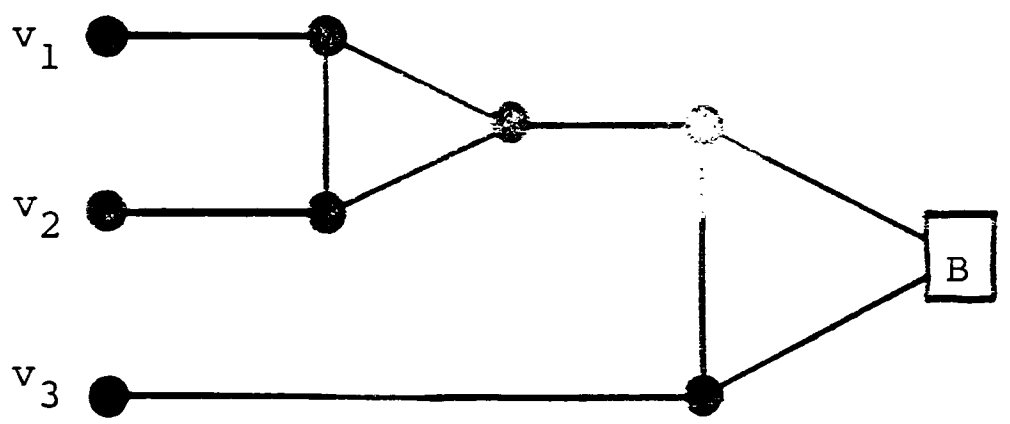


Figure 6

Now consider the result of not specifying, in the input, the domain of the partial 3-colouring. Let us define

FF-3COL(g(n))

Input: Graph G on n vertices.

Question: Does there exist a partial 3-colouring c of G with $|\text{dom } c| \leq g(n)$ such that c forces a 3-colouring of G?

Theorem 2: Apart from the cases where $g(n) \leq 1$ for all n, FF-3COL(g(n)) is g(n)logn-NP-complete whenever g is good and has property P1 .

Proof: Define, for all $n \in \mathbb{N}$,

$$L_3(n) = \lceil \log_3(n+1) \rceil .$$

Thus $L_3(n)$ is the number of digits in the ternary representation of n. Put $f(n) = g(n)L_3(n)$. Clearly $f(n)$ - NP = $g(n)\log n$ - NP. It will be convenient to speak of $f(n)$ - NP rather than $g(n)\log n$ - NP.

Many of the arguments used to establish this result are essentially identical to arguments used in the proof of Theorem 3.4, and the underlying technique is similar. We will not, therefore, give every detail in our proof: the interested reader will be able to fill such details in by analogy with the proof of Theorem 3.4.

Firstly, membership of $f(n)$ - NP is easily shown by virtually the same argument that was used in the corresponding part of the proof of Theorem 3.4.

We now show completeness.

If $g(n) = \Omega(n)$, then $f(n)$ - NP = NP so we prove NP-completeness. This is achieved by showing that

$$3COL \propto FF - 3COL(g(n))$$

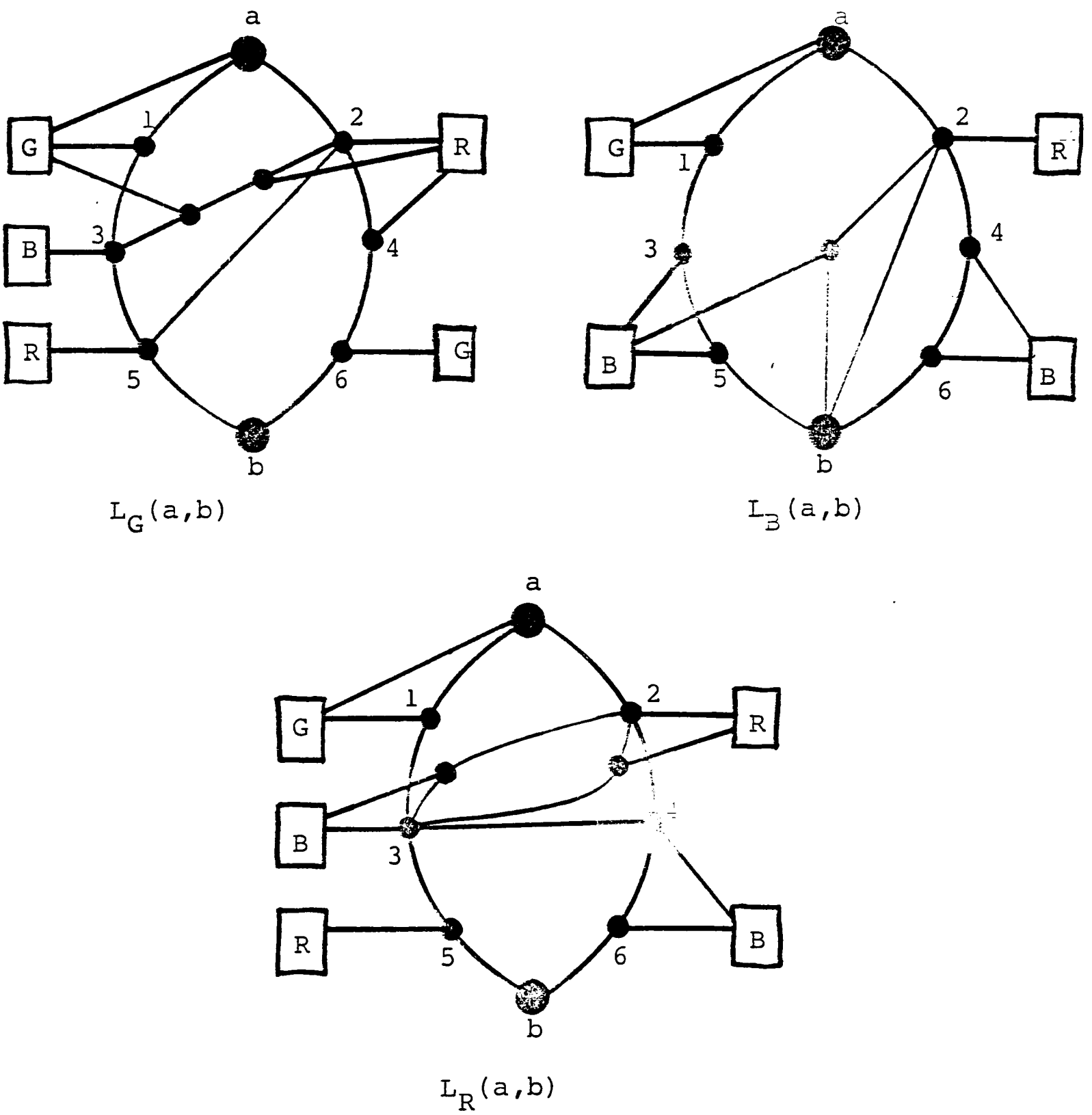
using substantially the same argument as was used in proving Case 1 in the proof of Theorem 3.4.

Suppose then that $g(n) \leqslant cn$ where $c<1$. We show

$$SF - 3COL(f(n)) \propto FF - 3COL(g(n)).$$

Our reduction employs essentially the same technique as Theorem 3.4. Before describing it, some special graphs must be described which will be used in the reduction.

The graphs $L_G(a,b)$, $L_B(a,b)$, $L_R(a,b)$ are shown in Figure 7, where a, b are the vertices indicated. (Once again we use the colour set {G,B,R}, and the vertices labelled G, B, R in our special graphs will be coloured according to their label.)



$L_G(a,b)$ $L_B(a,b)$

$L_R(a,b)$

Figure 7: The graphs $L_G(a,b)$, $L_B(a,b)$, and $L_R(a,b)$.

Their construction is rather ad hoc, but each $L_X = L_X(a,b)$ where $X \in \{R,B,G\}$ has the following properties with respect to any partial 3-colouring c such that $c(R) = R$, $c(B) = B$, $c(G) = G$.

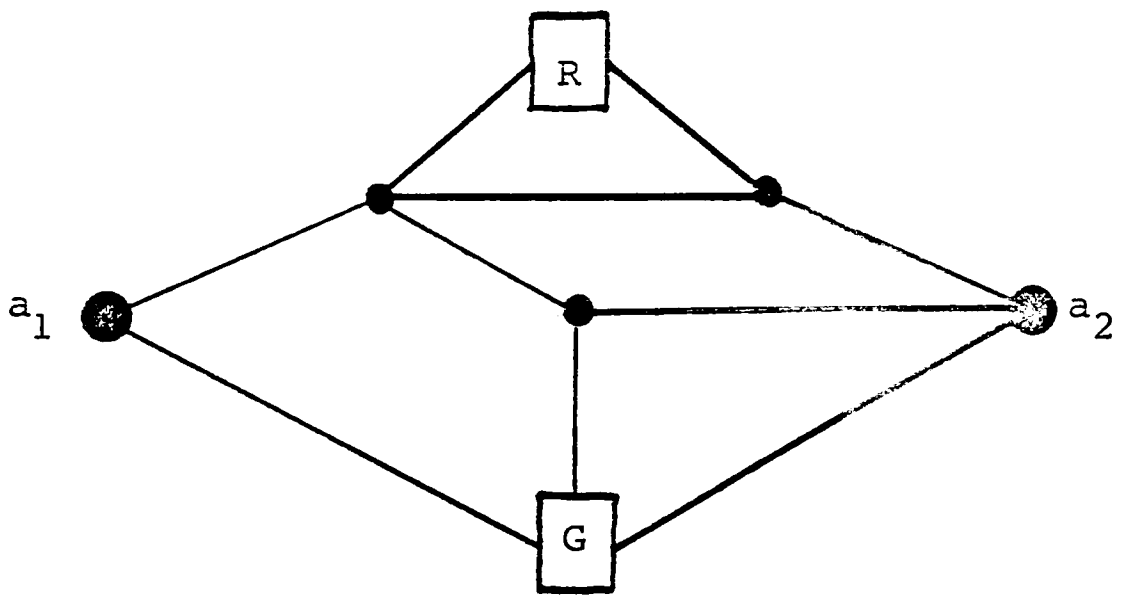
(L1) $c(a) = B$ forces all of L_X, including $c^*(b) = X$.

(L2) $c(a) = R$ alone (i.e. with no other vertex of $V(L_X)\setminus\{G,B,R\}$ belonging to dom c) does not force any colouring of b or either of its neighbours in L_X.

(L3) If $c(a) = R$ and $b \in$ dom c, then c forces all of L_X.

(L4) If $c(b)$ is defined but $c(b) \neq X$, then c forces all of L_X including $c^*(a) = R$.

(L5) $c(b) = X$, alone, forces all vertices of L_X <u>except</u> $\{a,1\}$. $c^*(2) = c^*(3) = G$, so that if a is also in dom c, then all of L_X is forced (whatever $c(a)$ is).



<u>Figure 8</u>: The graph $N(a_1,a_2)$.

The graph $N = N(a_1,a_2)$ of Figure 8 has the following properties (where again $c(R) = R$, $c(B) = B$, $c(G) = G$):

(N1) $c(a_i) = B$ forces all of N including $c^*(a_{3-i}) = R$, for each $i \in \{1,2\}$.

(N2) $c(a_1) = c(a_2) = R$ forces all the rest of N.

The polynomial transformation may now be described. The similarity with that of Theorem 3.4 will be evident and should explain the role of the graphs constructed above.

Suppose G is a graph on n vertices $\{v_1,\ldots,v_n\}$ and $U \subseteq V(G)$ with $|U| \leq f(n)$. Suppose w.ℓ.o.g. that $U = \{v_1,\ldots,v_{\mu L_3(n)}\}$ where $\mu \leq g(n)$.

Introduce $\mu\hat{n}$ new vertices $u_0,\ldots,u_{\mu\hat{n}-1}$, where $\hat{n} = 3^{L_3(n)}$. For each new vertex u_i, do the following.

1. Find q, j such that $i = q\hat{n} + j$, $0 \leq j < \hat{n}$, $0 \leq q \leq g(n) - 1$.

2. Express j in ternary:

$$j = \delta_{L_3(n)}\ \delta_{L_3(n)-1}\ \cdots\ \delta_3\delta_2\delta_1\ ,$$

where each $\delta_{j'} \in \{0,1,2\}$. So

$$j = \sum_{j'=1}^{L_3(n)} \delta_{j'}.3^{j'-1}\ .$$

3. Set $X_{j'} = \begin{cases} G, & \delta_{j'} = 0; \\ B, & \delta_{j'} = 1; \\ R, & \delta_{j'} = 2. \end{cases}$

For each $j' = 1,2,\ldots,L_3(n)$, add the graph

$$L_{X_{j'}}(u_i, v_{qL_3(n)+j'})\ .$$

4. Add the graph $N(u_{q\hat{n}+j_1}, u_{q\hat{n}+j_2})$ for all j_1, j_2, $0 \leq j_1 < j_2 \leq \hat{n} - 1$.

5. Padding: this step is very similar to Step 5 in the construction in the proof of Theorem 3.4; the details are routine, and we omit them. (As in that proof, it is in this step that we need $g(n) \leqslant cn$.)

Let G' be the graph so constructed. Some remarks on the above steps before proceeding further: in Step 3, $c(u_i) = B$ means that the ternary expansion of j does in fact describe the colouring of vertices $v_{qL_3(n)+j'}$, with digit j' (from the right) of j giving the colour assigned to $v_{qL_3(n)+j'}$. These colourings of the $v_{qL_3(n)+j'}$ are forced by $c(u_i) = B$ via the $L_{X_{j'}}(u_i, v_{qL_3(n)+j'})$ (see (L1) above). $c(u_i) = R$ means no such forcing occurs (see (L2)), and in fact if any of the $v_{qL_3(n)+j'}$ do <u>not</u> receive colour $X_{j'}$, then u_i is forced to be R (see (L4)). Properties (L3) and (L5) ensure that c^* is total or not total on the $L_X(-,-)$ as necessary. The use of the graphs $N(-,-)$ in Step 4 ensures that for all j_1, j_2, u_{qn+j_1} and u_{qn+j_2} are not both coloured B. If they were, a contradiction would result as two different colourings of the $v_{qL_3(n)+j'}$ would be implied.

The construction is clearly polynomial time, so it remains only to show

$$(G,U) \in \text{SF-3COL}(f(n)) \quad \text{iff} \quad G' \in \text{FF-3COL}(g(n)).$$

The proof of this is routine and very similar to the corresponding part of the proof of Theorem 3.4 and so we omit it. □

§5. f(n)-NP-complete problems involving Hamiltonian circuits

This is the last section devoted to proofs of f(n)-NP-completeness results. We show that two forcing problems involving Hamiltonian circuits are f(n)-NP-complete. We conclude by deducing, as corollaries of the results of this section and the previous two sections, some new logspace completeness results for P. This section continues the trend of paying progressively less attention to routine details of completeness proofs.

We begin by defining forcing in this new context.

Let G be a graph. A partial Hamiltonian circuit (p.h.c.) in G is a set of edges of G forming either a Hamiltonian circuit of G or a set of vertex-disjoint paths of G. Note that any subgraph of a Hamiltonian circuit of G is a p.h.c. of G but that the converse does not necessarily hold (where, for the purposes of this statement, a subgraph is identified with its set of edges). We allow the null p.h.c., $\emptyset$. If $E' \subseteq E(G)$ and $h : E' \to \{0,1\}$ then we say h is a partial Hamiltonian function (p.h.f) of G if the set

$$\{e \in E' \mid h(e) = 1\},$$

denoted by C(h), is a p.h.c. of G. We think of the edges e with $h(e) = 0$ as being forbidden from occurring in any p.h.c. containing C(h) (in particular, we proscribe them from occurring in any Hamiltonian circuit containing C(h)).

Let h be a p.h.f. of G with domain $\hat{E}$. We can form a function h' extending h by first setting $h'|_{\hat{E}} = h$ and then applying any of the three basic forcing steps defined below.

(i) If $e \in E(G)\backslash\hat{E}$ is incident with $v \in V(G)$, and e is one of at most two edges incident with v which do not receive value 0 under h, then set $h'(e) = 1$. (See Figure 1(i).)

(ii) If $e \in E(G)\backslash\hat{E}$ is incident with $v \in V(G)$, and there exist edges e_1, e_2 incident with v such that $e \notin \{e_1, e_2\}$ and $h(e_1) = h(e_2) = 1$, then set $h'(e) = 0$. (See Figure 1(ii).)

(iii) If some component of $C(h)$ is a path P of G such that $|P| \leq |V(G)| - 2$, and an edge $e \in E(G)\backslash\hat{E}$ is incident with both endpoints of G, then set $h'(e) = 0$. (See Figure 1(iii).)

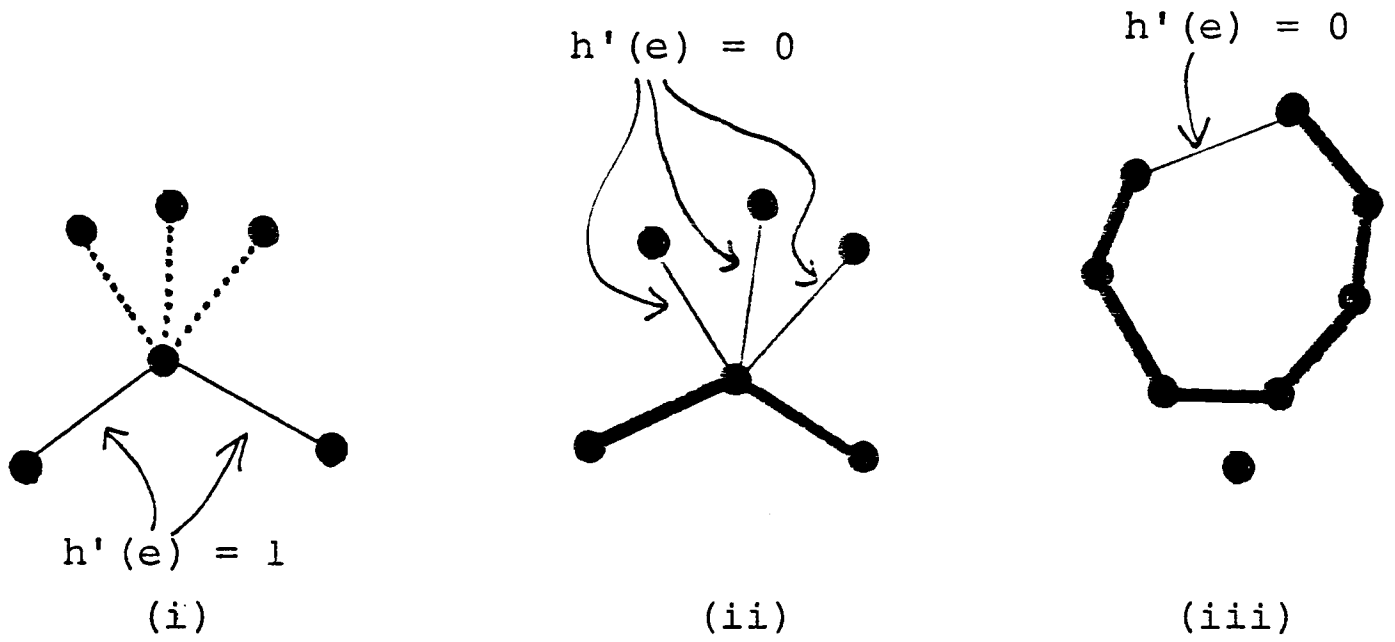


Figure 1: The three basic forcing steps. Heavy edges indicate $h(e) = 1$; dotted edges, $h(e) = 0$; other edges are not in $\hat{E}$.

If h' can be obtained from h by repeated application of basic forcing steps, then we say h forces h'. Also if C is a p.h.c. of G then we say h forces C if $C \subseteq C(h')$ for some h' forced by h. We say h is contradictory if there exist h_1 and h_2 forced by h and an

edge e such that $h_1(e) \neq h_2(e)$. We write h* for the maximal function forced by h. It can be seen that if h is noncontradictory then h* is well-defined, unique, and is a p.h.f. of G. We say h <u>forces a Hamiltonian circuit of G</u> if C(h*) is a Hamiltonian circuit of G, and that G is <u>forcibly Hamiltonian</u> if the null p.h.f. $\emptyset$ forces a Hamiltonian circuit of G (see the problem FORCED HAM, defined just before Theorem 4).

The next problem shown to be f(n)-NP-complete is the following.

SF-HAM(f(n))
Input: Graph G on n vertices, set of edges $D \subseteq E(G)$ with $|D| \leq f(n)$.
Question: Is there a p.h.f. h with domain D such that h forces a Hamiltonian circuit of G?

<u>Theorem 1</u>: For any good function f with property P2, SF-HAM(f(n)) is f(n)-NP-complete.

<u>Proof</u>: Membership of f(n)-NP is clear.

We show

$$\text{SF-3SAT}(f'(n)) \propto \text{SF-HAM}(f(n))$$

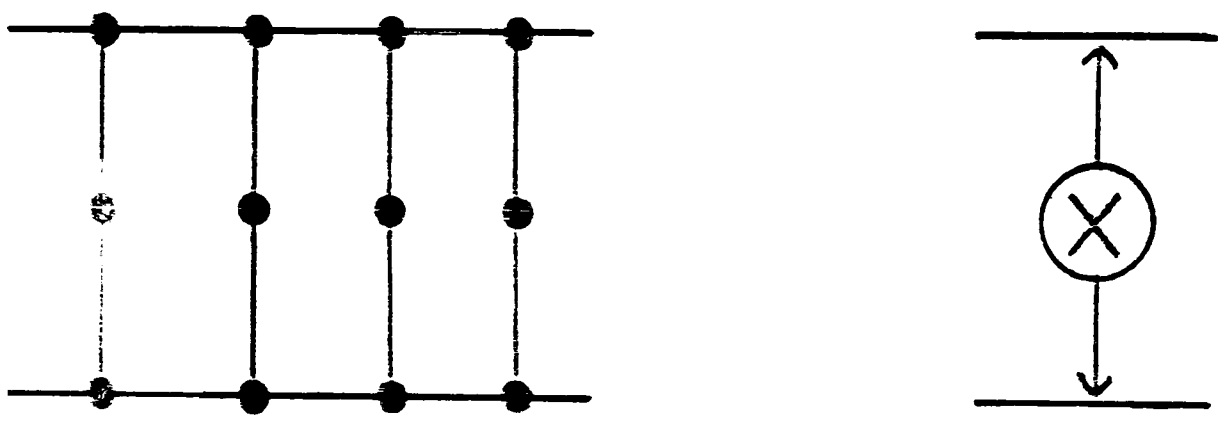
where f' is defined by

$f'(n) = 0$ for all n, if $f(n) = O(\log n)$;

and $f'(n) = f(n)$ for all n, otherwise.

The transformation is a slight modification of the transformation from 3SAT to HAM given in [42, Theorem 15.6, pp. 366-370].

We use some special graphs. The first is from [42]; it is shown in Figure 2(a) together with its symbolic representation in Figure 2(b). We call it the exclusive-or (after [20]). It behaves like two edges, exactly one of which must belong to any Hamiltonian circuit (in any graph containing the exclusive-or as an induced subgraph). An important forcing property of the exclusive-or is that if it is an induced subgraph[1] of a graph G, and h is a p.h.f. of G which is defined on any of the "horizontal" edges (see Figure 2(a)) of the exclusive-or, then h* is defined on the whole exclusive-or.



(a) (b)

Figure 2: The exclusive-or.

The second special graph is called the 2-input-or (after [20]) and is shown in Figure 3(a); it is used in the standard transformation from VERTEX COVER to HAM (see [18, Theorem 3.4]). Its symbolic representation is given in Figure 3(b). It behaves like two edges such that at least one of them belongs to any Hamiltonian circuit (of any graph containing the 2-input-or as an induced subgraph[1]). It has the following important forcing properties. If $h(e) = 0$ for any $e \in \{e_1, e_2, e_3, e_4\}$, then h* is defined on the whole 2-input-or; and if $h(e) = h(f) = 1$ for any e, f such that $\{e,f\} \neq \{e_1,e_2\}$ or $\{e_3,e_4\}$, then h* is again defined on the whole 2-input-or.

[1] with no other edges incident with the vertices of the exclusive-or/2-input-or.

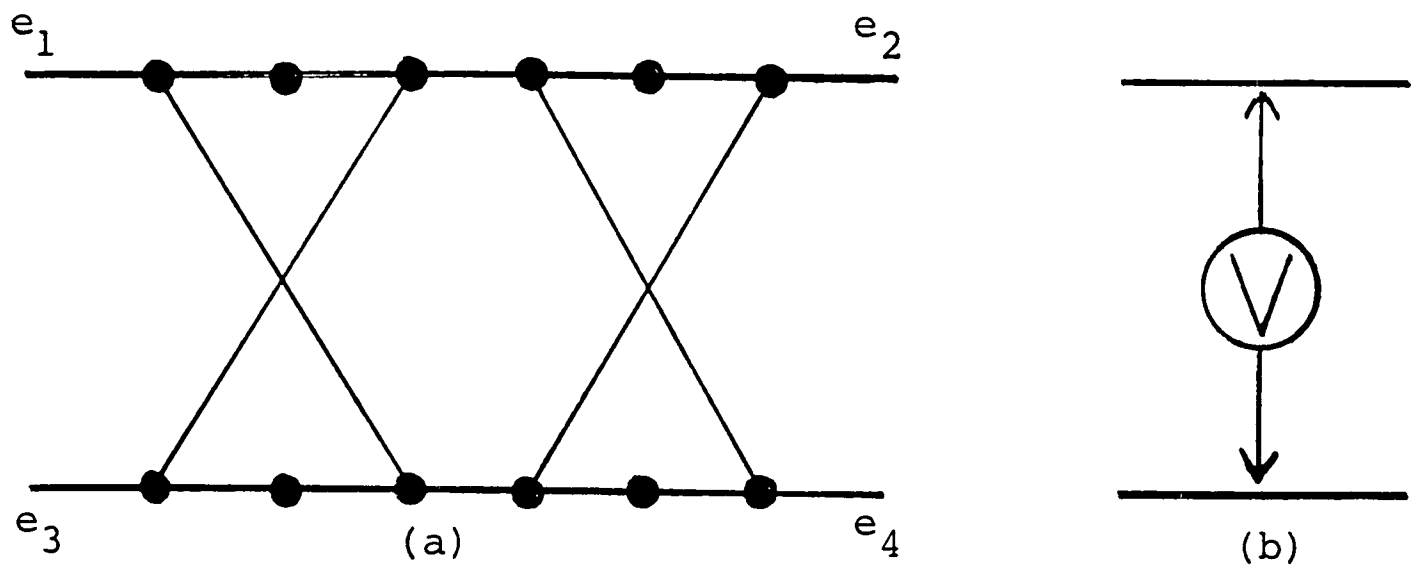


Figure 3: The 2-input-or.

The required transformation is as follows. Let ϕ be a Boolean expression in c.n.f. with m clauses each of size 3, and let $U \subseteq Var(\phi)$ with $|U| \leq f'(m)$. Construct from ϕ the graph G exactly as in [42], except that instead of using the clause simulator of [42] (shown in Figure 4) we use the clause simulator shown in Figure 5. In each case, the starred edges are connected by exclusive-ors to the truth-setting components for the 3 literals appearing in the clause (we refer the reader to [42] for the details of this construction). The circled vertices are used to join all the clause simulators in series, so that any Hamiltonian circuit must enter each clause simulator through one of its circled vertices and leave it through the other. The new clause simulator is a modification of the old one and we use it because it has the required forcing properties whereas the old one does not.

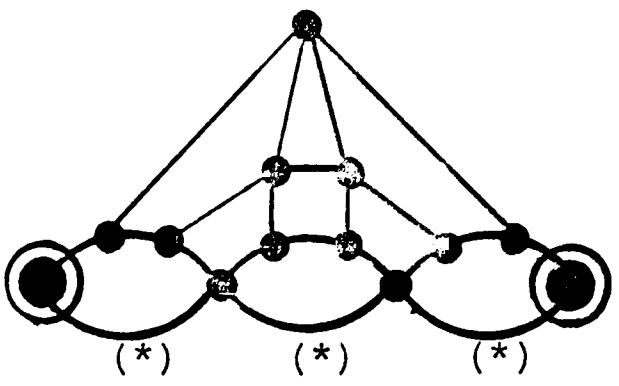


Figure 4

The old clause simulator.

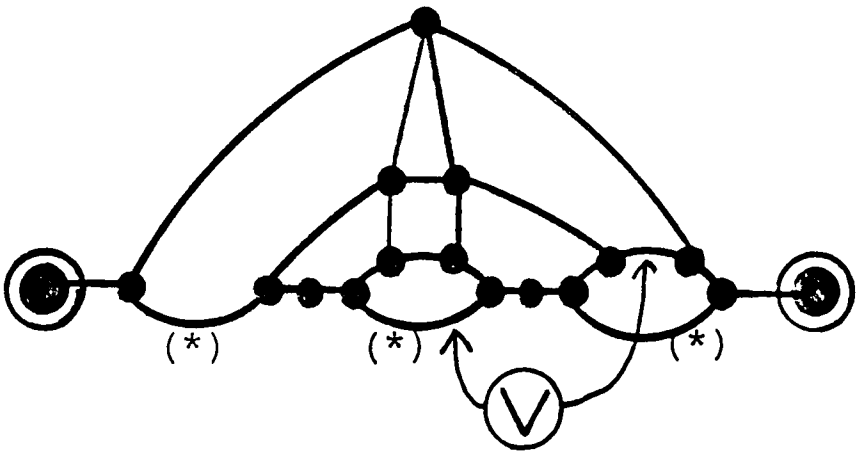


Figure 5

The new clause simulator.

Having constructed G from ϕ, we just form the edge set E' consisting of the edges of G corresponding to variables in U (see [42] for details of the variable-edge correspondence). Some padding may also be necessary but this is a trivial matter so we do not discuss it further. The mapping $(\phi,U) \mapsto (G,E')$ is our polynomial transformation; the reader may easily verify that it is polynomial time computable and that

$$(\phi,U) \in \text{SF-SAT}(f'(n)) \text{ iff } (G,E') \in \text{SF-HAM}(f(n)). \quad \square$$

It is natural to consider what happens when the domain of the p.h.f. is not specified in the input.

F_1F-HAM(g(n))
Input: Graph G on n vertices.
Question: Does there exist a p.h.f. h of G with $|\text{dom } h| \leq g(n)$ such that h forces a Hamiltonian circuit of G?

(We use the subscript 1 here as we will shortly consider another, very similar problem which we will indicate by subscript 2.)

Theorem 2: If g is good and has property P1 then F_1F-HAM(g(n)) is g(n)logn-NP-complete.

Proof (Sketch): Put $f(n) = g(n)L(n)$, where $L(n)$ is defined as at the start of the proof of Theorem 3.4.

Membership of f(n)-NP is clear.

The technique for showing completeness is once again based on that for showing the f(n)-NP-completeness of

FF-SAT(g(n)) (Theorem 3.4). In particular, if $g(n) \leq cn$ then we closely follow Case 2 of the proof of Theorem 3.4. We sketch this proof. The reduction is from SF-HAM(f(n)) restricted to graphs containing a vertex of degree 2, which we know is f(n)-NP-complete since all graphs constructed by the polynomial transformation of Theorem 1 have a vertex of degree 2.

Let G be a graph on n vertices and suppose $E' \subseteq E(G)$ where $|E'| \leq f(n)$. Suppose $E' = \{e_1, \ldots, e_{\mu L(n)}\}$ where $\mu \leq g(n)$. Let w be a vertex of degree 2 in G and suppose e_0 is an edge of G incident with w. We may assume $e_0 \notin E'$ since e_0 is forced by any p.h.f. (including $\emptyset$) of G. Take $\mu n + \mu L(n)$ parallel pairs of new edges:

$\{v_i, \bar{v}_i\}$, $1 \leq i \leq \mu L(n)$;

$\{u_i, \bar{u}_i\}$, $0 \leq i \leq \mu n - 1$.

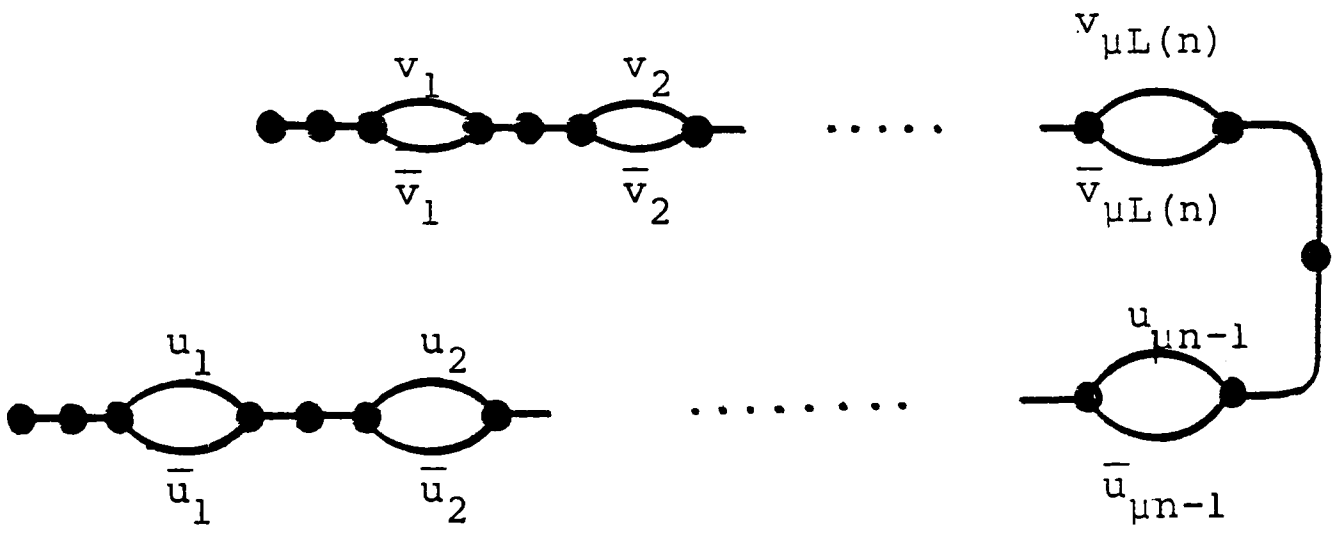


Figure 6: The graph Q.

String them together as shown in Figure 6, and denote by Q the graph so formed. Q has two vertices of degree 1: identify these vertices with the endpoints of e_0, and delete e_0 (see Figure 7).

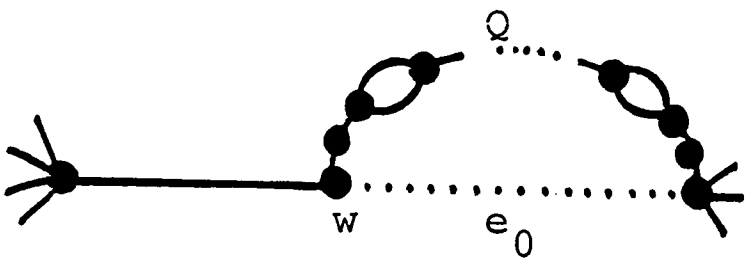


Figure 7

Firstly, for each i, $1 \le i \le \mu L(n)$, connect edges e_i and $\bar{v}_i$ by an exclusive-or.

The rest of the construction consists of five steps which are essentially the same as Steps 1 to 5 of the construction of Theorem 3.4. In fact, if in Steps 1 to 4 of that construction we replace m by n, "literal" by "edge", and add 2-input-ors connecting pairs of edges instead of clauses containing pairs of literals, then we have exactly the construction we need here. (The padding step is easily taken care of.) Let H be the graph we have now constructed.

It can now be shown, by arguments very similar to those of the proof of Theorem 3.4, that $(G,E') \in$ SF-HAM$(f(n))$ iff $H \in F_1$F-HAM$(g(n))$. The map $(G,E') \mapsto H$ is polynomial time computable, so the result follows. □

It is perhaps even more natural to look for p.h.c.'s rather than p.h.f.'s. To this end we make some definitions. If C is a p.h.c. of a graph G then h_C is the p.h.f. defined by

$$h_C(e) = 1 \quad \forall\, e \in C,$$

$h_C(e)$ being undefined otherwise. If C_1 and C_2 are p.h.c.'s with $C_1 \subseteq C_2$, then we say C_1 <u>forces</u> C_2 if there is a p.h.c. C_3 such that h_C forces C_3 and $C_2 \subseteq C_3$. We then define

F_2F-HAM(g(n))

Input: Graph G on n vertices.

Question: Does there exist a p.h.c. C of G with $|C| \leq g(n)$ such that C forces a Hamiltonian circuit of G?

Theorem 3: If g is good and has property P1 then F_2F-HAM(g(n)) is g(n)logn-NP-complete.

Proof (Sketch): Membership of g(n)log n-NP is clear.

To prove completeness, we reduce from F_1F-HAM(g(n)) restricted to graphs with a vertex of degree 2, which we know is g(n)log n-NP-complete by Theorem 2 since all graphs constructed by the transformation there have a vertex of degree 2.

Let G be a graph on n vertices and suppose w is a vertex of degree 2 in G. Let e_0 be an edge of G incident with w. Suppose $E(G)\backslash\{w\} = \{e_1,\ldots,e_m\}$. Take m parallel pairs of new edges $\{x_i,\bar{x}_i\}$, $1 \leq i \leq m$, and string them together to form the graph Q as shown in Figure 8 (compare Figure 6). Q has two vertices of degree 1: identify these vertices with the endpoints of e_0, and delete e_0. Finally, for each i, $1 \leq i \leq m$, connect e_i to $\bar{x}_i$ by an exclusive-or. Thus, the choice of which of $x_i,\bar{x}_i$ belongs to some p.h.c. C determines the value of h_C^* on e_i.

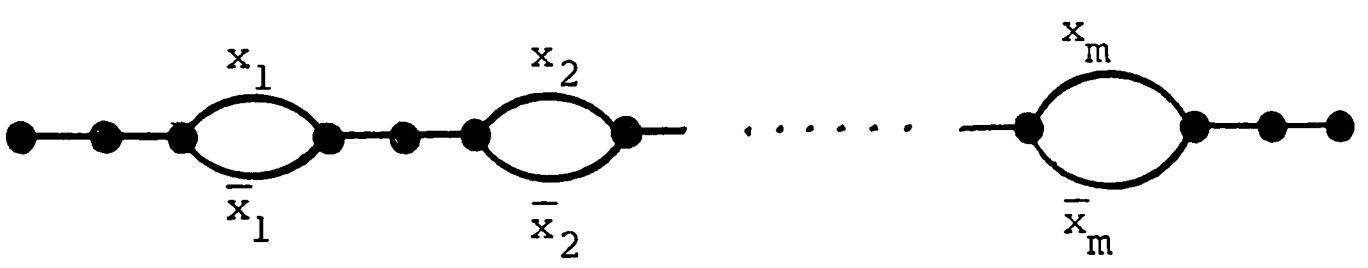


Figure 8: The graph Q.

We leave to the reader the verification that this construction is in fact a polynomial transformation with the required properties. □

We conclude this section by considering the complexity of the following problems, which are the restrictions of SF-SAT(f(n)), FF-3COL(g(n)), and SF-HAM(f(n)) to the cases $f(n) \equiv 0$, $g(n) \equiv 3$ and $f(n) \equiv 0$ respectively.

FORCED SAT

Input: Boolean expression ϕ in c.n.f.

Question: Does the empty partial truth assignment force a satisfying truth assignment for ϕ?

FORCED 3COL

Input: Graph G.

Question: Is there a colouring of some triangle of G which forces a 3-colouring of G?

FORCED HAM

Input: Graph G.

Question: Is G forcibly Hamiltonian?

Theorem 4: FORCED SAT, FORCED 3COL and FORCED HAM are logspace complete for P.

Proof: All the polynomial transformations given in this Chapter are in fact logspace computable. Thus all our completeness results apply also to logspace reducibility. The result then follows from Theorems 3.2, 4.2 and 5.1 in the cases $f(n) \equiv 0$ and $g(n) \equiv 3$. □

Note that the logspace completeness for P of FORCED SAT also follows from Jones and Laaser's proof [30] that UNIT RESOLUTION (see end of §7) is logspace complete for P.

§6. Open problems

The work of this Chapter suggests several natural questions which remain open.

We have obtained a number of nontrivial f(n)-NP-completeness results, all concerning problems derived from known NP-complete problems and all involving forcing of some kind. Our results can be regarded as just a sample of what is possible: it would appear that most known NP-complete problems have more or less natural "forcing" versions which could be shown (probably using similar techniques to ours) to be f(n)-NP-complete. Is there a general theorem behind this? As with many other properties that appear to hold for all known NP-complete problems (e.g. #P-completeness of the corresponding enumeration problem), such a general result would probably be difficult to obtain.

Although we have successfully used f(n)-NP-completeness to classify the complexity of a number of combinatorial problems (see entry (1) of the Table at the end of this section), we have not been able to apply it to the problems which originally motivated this work (see §1, and entries (2) and (3) in the Table). Consider for example:

f(n)-CLIQUE
Input: Graph G.
Question: Does G have a clique of size $f(|V(G)|)$?

We know f(n)-CLIQUE belongs to f(n)log n-NP, and in fact more can be said about its complexity.

Lemma 1: For all f, f(n)-CLIQUE belongs to DSPACE(f(n)log n).

Proof: Given a graph G, lexicographically go through all sets of f(n) vertices of G, and check for each such set whether it induces a clique in G. The space bound is then easily verified. □

This can be regarded as evidence that f(n)-CLIQUE is not f(n)log n-NP-complete since it does not appear to be the case that f(n)log n-NP $\subseteq$ DSPACE(f(n)log n). Evidence against f(n)log n-NP-completeness can also be found for f(n)-CIRCUIT (see entry (3) in the Table) and many other problems of similar type (see e.g. entry (5) in the Table, a curious problem). We have not been able to obtain any more precise information on the complexity of these problems.

We have focussed on the amount of nondeterminism as a problem parameter, and investigated how varying this parameter affects the complexity of the problem. Other problems, of course, have parameters which are not, apparently, related to restricted nondeterminism. Often, the complexity of such problems when the parameter is either fixed or arbitrary is not unlike that of the problems studied so far: see for example entry (4) in the Table. Again, we ask what happens when the parameter is intermediate between these extremes. Indeed such questions are not confined to the P vs. NP situation: see (6) and (7) in the Table.

With such problems in mind we consider the following very general situation.

Suppose $(L_k : k \in \mathbb{N})$ is a sequence of languages. Define the language

$$L_\omega = \{(x,k) \mid x \in L_k\}.$$

(There is an analogy here with the ω-jump of recursive function theory [57, p.257].) For example, if L_k = k-CLIQUE then L_ω = CLIQUE. It is intended that k is a problem parameter and that the L_k are the languages obtained by fixing k, so that L_ω is the language obtained by removing all restriction on k. We are interested in what happens when k is neither fixed nor arbitrary. For any f, define

$$L(f) = \{x | x \in L_{f(|x|)}\}.$$

Question: Let C be a class such that $L_k \in C$ for all k and $L_\omega \notin C$. What can be said about the complexity of $L(f)$?

(Thus, if L_k = k-CLIQUE and C = LOGSPACE, then (assuming LOGSPACE $\neq$ NP) we are asking about the complexity of f(n)-CLIQUE.)

We have attacked this question, with some success, in the case of certain combinatorial problems involving forcing ((1) in Table). It would be interesting to have some results for other more commonly encountered problems, such as the others in the Table. More interesting would be results giving some kind of complexity classification of $L(f)$ under moderately general conditions, although a full answer to the question is probably a hopeless aim.

PROBLEM	COMPLEXITY		
	If k is constant	If k is arbitrary	If k = f(n) where f is good
(1) SF-SAT(k) or FF-SAT(k) or SF-3COL(k) ... (etc.)	P	NPC	f(n)-NPC
(2) CLIQUE Input: G, k. Question: Does G have a clique of size k?	LOGSPACE	NPC	Open. Belongs to f(n)log n-NP and DSPACE(f(n)log n).
(3) CIRCUIT Input: G, k. Question: Does G have a circuit of size ≥ k?	NL	NPC	Open. Belongs to f(n)log n-NP and NSPACE(f(n)log n).
(4) CLIQUE MINOR Input: G, k. Question: Does G have a K_k-minor?	P [47]	NPC [61]	Open. Does not appear to be in f(n)log n-NP.
(5) RAMSEY Input: G, k. Question: Does G have a clique or an independent set of size k?	Trivial. [22]	NPC	Open. Belongs to f(n)log n-NP and DSPACE(f(n)log n); ∃ constant c s.t. the problem is trivial if f(n) ≤ c.log n [22, pp. 3-6].
(6) EDGES Input: G, k. Question: Does G have ≥ k edges?	DSPACE(0)	LOGSPACE	Belongs to DSPACE(log(f(n))).
(7) k-QBF Input: Boolean formula ϕ with ≤ k quantifiers. Question: Is ϕ true?	k-th level of poly. hierarchy (see [18,§7.2])	PSPACE-complete	Open. Belongs to PSPACE.

§7. Appendix: the proof of Theorem 3.2

We restate the result and then prove it in detail.

Theorem 3.2: For any good function f, SF-SAT($f(n)$) is $f(n)$-NP-complete.

Proof: Firstly, SF-SAT($f(n)$) $\in$ $f(n)$-NP: given (ϕ,U) as input, guess a partial truth assignment t for ϕ with domain U. Checking that t^* is total and satisfies ϕ is easily done, and t as a certificate for (ϕ,U) is essentially just a string of $|U|$ binary digits and so has size at most $f(m) \leq f(|(\phi,U)|)$, as required.

It remains to show that $L \propto$ SF-SAT($f(n)$) for all $L \in f(n)$-NP. As remarked earlier, this proof is based on a close analysis of the proof of Cook's Theorem (see [18], from which we get much of our notation).

For convenience we will use here, as our model of non-deterministic computation, a single tape TM with a two-way infinite tape; initially the input is written on the positively numbered squares and the guess is written in binary on the negatively numbered squares (we call these negatively numbered squares the guess tape, and when we speak of the first k squares of the guess tape we mean squares -1 to $-k$). Once input and guess are on the tape, computation proceeds deterministically.

Suppose $L \in f(n)$-NP. Then there is a polynomial time NDTM M (with binary guess tape alphabet) recognizing L, and a polynomial q such that for any input $x \in \Sigma^*$, M only scans (at most) the first $f(q(|x|))$ squares of the guess tape. Let M have input alphabet Σ, tape alphabet $\Gamma = \{s_0, s_1, \ldots, s_\nu\}$ where $s_0 = b$ is the blank symbol,

guess tape alphabet $\{s_1, s_2\}$ and set of states $Q = \{q_0, \ldots, q_r\}$ where $q_1 = Y$ and $q_2 = N$ are the accepting and rejecting states respectively. Let p be a polynomial bounding the time complexity of M.

We describe a polynomial time computable function $g_L : \Sigma^* \to \Sigma^*$ such that

$x \in L \iff g_L(x) \in$ SF-SAT($f(n)$).

Suppose then that $x \in \Sigma^*$ is an input for M. As in the proof of Cook's Theorem, a Boolean expression ϕ in c.n.f. is constructed, which has a satisfying truth assignment iff M has an accepting computation for some guess of length at most $f(q(|x|))$ on input x, which in turn is true iff $x \in L$. The construction of ϕ is exactly as in [18], §2.6, Theorem 2.1, pp. 39-44, with the difference that whenever j ranges over the interval $[-p(|x|), p(|x|) + 1]$ in their construction, we have it ranging over the interval $[-f(q(|x|)), p(|x|) + 1]$. This is because j indicates the tape square being scanned by M, with $j < 0$ indicating a square of the guess tape, and here M never looks beyond the $f(q(|x|))$-th square of the guess tape. We also add the clauses $\{S[0,j,1], S[0,j,2]\}$ for all j, $-f(q(|x|)) \leqslant j \leqslant -1$ (meaning that initially the guess is binary).

We adopt directly from [18] the following notation for the variables of ϕ: $S[i,j,\ell]$, $0 \le i \le p(|x|)$, $-f(q(|x|)) \le j \le p(|x|) + 1$, $0 \le \ell \le |\Gamma| - 1$, is the variable of ϕ with the meaning, "at time i, tape square j contains symbol s_ℓ"; $Q[i,k]$, $0 \le i \le p(|x|)$, $0 \le k \le |Q| - 1$, has the meaning "at time i, M is in state q_k"; $H[i,j]$, $0 \le i \le p(|x|)$, $-f(q(|x|)) \le j \le p(|x|) + 1$, means "at time i, the read-write head is at square j".

Define $V(i)$ to be the set of all variables of ϕ pertaining to the configuration of M at time i; that is,

$$V(i) := \{S[i,j,\ell], Q[i,k], H[i,j] \mid$$

$$j \in [-f(q(|x|)), p(|x|) + 1],$$

$$\ell \in [0, |\Gamma| - 1],$$

$$k \in [0, |Q| - 1]\} .$$

Define V to be the set of all variables of ϕ; thus $V = \bigcup_i V(i)$. Let

$$U = \{S[0,j,1] \mid -f(q(|x|)) \leqslant j < 0\},$$

where by definition $U = \emptyset$ if $f(q(n)) \equiv 0$.

Claim 1: If t is a noncontradictory partial truth assignment for ϕ whose domain is U then dom $t^* = V$.

Proof of Claim 1: We show by induction on i that $V(i) \subseteq$ dom t^* for all i.

Consider the case $i = 0$. $S[0,j,\ell] \in$ dom t^* $\forall j<0$, $\forall \ell$ (by U and clauses $\{S[0,j,1], S[0,j,2]\}, \{\overline{S[0,j,\ell]}, \overline{S[0,j,\ell']}\}$). Since ϕ has the clauses $\{S[0,0,0]\}$, $\{S[0,1,k_1]\}$, $\{S[0,2,k_2]\}, \ldots,$ $\{S[0,|x|,k_{|x|}]\}$, $\{S[0,|x| + 1,0]\}, \ldots,$ $\{S[0,p(|x|) + 1,0]\}$

(where $x = s_1 s_2 \ldots s_{|x|}$), the single variables in these clauses are all in dom t*. Now for all $j \geq 0$, ϕ contains the clauses $\{\overline{S[0,j,\ell]}, \overline{S[0,j,\ell']}\}, \ell' \neq \ell$. Thus, since $t^*(S[0,0,0]) = T, t^*(S[0,0,\ell']) = F$ for all $\ell' \neq 0$; and since $t^*(S[0,1,k_1]) = T$, $t^*(S[0,1,\ell']) = F$ for all $\ell' \neq k_1$; and so on. Thus $S[0,j,k] \in$ dom t* for all j and k. Also ϕ has the clause $\{Q[0,0]\}$, so $t^*(Q[0,0]) = T$; and for all $j \neq 0$, ϕ has the clause $\{\overline{Q[0,0]}, \overline{Q[0,j]}\}$ which implies $t^*(Q[0,j]) = F$ and $Q[0,j] \in$ dom t*. Similarly $H[0,j] \in$ dom t* for all j. Hence $V(0) \subseteq$ dom t*.

Now suppose $V(i-1) \subseteq$ dom t*. At time i - 1, suppose the machine is in state q_k, scanning square j which contains symbol s_ℓ. Then under t* the variables $S[i-1,j,\ell]$, $Q[i-1,k]$, $H[i-1,j]$ receive value T. Now M moves from this configuration, according to its transition function, to a new square $j + \Delta$ where $\Delta = -1$ or 1 (or 0 if $q_k \in \{q_Y, q_N\}$), changes to a new state $q_{k'}$, and writes a new symbol $s_{\ell'}$ on square j. This is described by the following clauses of ϕ (from clause group G_6 in [18]):

$\{H[i-1,j], Q[i-1,k], S[i-1,j,\ell], H[i,j+\Delta]\}$,

$\{\quad " \quad , \quad " \quad , \quad " \quad , Q[i,k']\}$,

$\{\quad " \quad , \quad " \quad , \quad " \quad , S[i,j,\ell']\}$.

In each of these clauses the first three literals have value F under t*, so the last literal must have value T under t*:

$$t^*(H[i,j+\Delta]) = t^*(Q[i,k']) = t^*(S[i,j,\ell']) = T.$$

We now consider the squares j' for all $j' \neq j$. The symbols on these squares do not change, and accordingly ϕ contains the clause

$$\{\overline{S[i-1,j',\ell_{ij'},]}, H[i-1,j'], S[i,j',\ell_{ij'},]\},$$

where $S_{\ell_{ij'}}$ is the symbol contained by square j' at time $i - 1$. The first two literals of this clause are F under t^*, so $t^*(S[i,j',\ell_{ij'},]) = T$.

Thus one of the variables $H[i,\cdot]$ has t^*-value T, one of the $Q[i,\cdot]$ has t^*-value T, and for all j one of the $S[i,j,\cdot]$ has t^*-value T. This was also true in the case $i = 0$ above, and the same arguments used there can be used here to show that these truth assignments alone force truth assignments to all the other variables in $V(i)$. As all the details are spelt out there, we omit them here. Thus $V(i) \subseteq \text{dom } t^*$, so we have proved that $V(i) \subseteq \text{dom } t^*$ for all i, and therefore that $V \subseteq \text{dom } t^*$, thus establishing the claim.

Claim 2: ϕ is satisfiable iff there exists a partial truth assignment t for ϕ such that $\text{dom } t \subseteq U$, $\text{dom } t^* = V$, and t^* satisfies ϕ.

Proof of Claim 2: Suppose ϕ is satisfiable. Let τ be a satisfying truth assignment for ϕ. Define $t = \tau|_U$. Then t is a noncontradictory partial truth assignment whose domain is U, so by Claim 1, $\text{dom } t^* = V$. The reverse implication is trivial so Claim 2 is proved.

Having constructed ϕ, we form ϕ' by adding $c = \max\{0, q(|x|) - m\}$ [1] new clauses to ϕ, each consisting of a single new variable not occurring in ϕ. Obviously ϕ' is satisfiable iff ϕ is. Furthermore, Claim 2 also holds with ϕ' instead of ϕ. If m' is the number of clauses of ϕ' then clearly

$$m' = \max\{m, q(|x|)\} .$$

Now

$$|U| = f(q(|x|)) \leq f(m').$$

So ϕ is satisfiable iff $\phi' \in$ SF-SAT(f(n)).

These facts, together with Claim 2, tell us that the function $g_L : \Sigma^* \to \Sigma^*$ defined by $g_L(x) = (\phi', U)$ satisfies

$$x \in L \iff (\phi', U) \in \text{SF-SAT}(f(n)).$$

It can be seen that, in any case, g_L is polynomial time computable. Therefore it is a polynomial transformation

---

[1] where m is the number of clauses of ϕ.

from L to SF-SAT(f(n)). Since this holds for all $L \in NP$, we have shown that SF-SAT(f(n)) is f(n)-NP-complete. □

When this proof is specialized to the case $f(n) \equiv 0$, we obtain Jones and Laaser's proof [30] that the following language is logspace complete for P.

UNIT RESOLUTION = $\{\phi \mid \emptyset$ is a contradictory partial truth assignment for $\phi\}$.

CHAPTER 4

THE COMPLEXITY OF MULTICOLOURING.

§1. Multicolourings of graphs

Graph multicolouring is an interesting and natural generalisation of graph colouring which was introduced by Hilton, Rado and Scott [26]. Further work in the subject includes [3, 63]; we note in particular Garey and Johnson's use of multicolouring, in [17], which we emulate in Theorem 5.3.2. In this Chapter we are concerned with the complexity of multicolouring, which was first considered by Irving [27]. We obtain significant extensions of Irving's results by proving that a large class of multicolouring problems are NP-complete.

In this Section we give basic definitions and state some elementary results that we will need.

A graph G is <u>(r,s)-colourable</u> if to each of its vertices we can assign r colours, from an available set of s colours, such that adjacent vertices receive disjoint colour sets. Such an assignment is then called an <u>(r,s)-colouring</u> of G, and a <u>multicolouring</u> of G is an (r,s)-colouring for some r and s. It is clear that if $r = 1$ we have ordinary graph colouring, and it is not difficult to see that if $s = 2r$ then we are considering 2-colourability. Figure 1 shows a $(3,7)$-colouring of C_7.

The <u>r-chromatic number</u> of a graph G is given by

$$\chi_r(G) = \min\{s \mid G \text{ is } (r,s)\text{-colourable}\}.$$

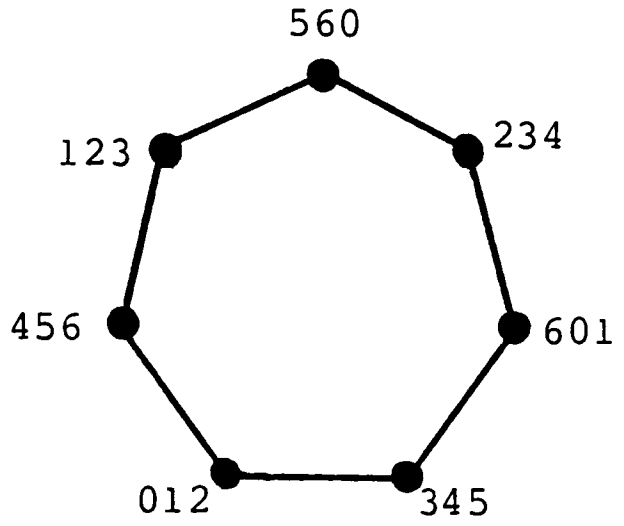


Figure 1: A (3,7)-colouring of C_7, with colours $\{0,1,\ldots,6\}$.

For example, we see from Figure 1 that $\chi_3(C_7) \leq 7$. (In fact $\chi_3(C_7) = 7$; if $\chi_3(C_7)$ were 6 or less then C_7 would be 2-colourable, a contradiction.)

The r-chromatic number can in fact be regarded as the ordinary chromatic number of a suitably constructed graph. For it is easy to show the following.

Lemma 1 [21, 63]: For any graph G,

$$\chi_r(G) = \chi(G[K_r]).$$ □

In fact any r-chromatic graph would do here instead of K_r :

Lemma 2 [21, 63]: For any graphs G and H,

$$\chi(G[H]) = \chi_{\chi(H)}(G).$$ □

The multichromatic number, denoted by χ^*, is given by

$$\chi^*(G) = \inf \left\{ \frac{\chi_r(G)}{r} \,\middle|\, r \in \mathbf{N} \right\}.$$

For example, it is easily seen that $\chi^*(K_n) = n$, and it is not hard to show [63] that

$$\chi^*(C_{2n+1}) = 2 + \frac{1}{n}.$$

(Trivially, $\chi^*(G) = 2$ if G is bipartite.) χ^* was first introduced by Hilton, Rado and Scott [26] who proved (before the Four Colour Theorem) that if G is planar then $\chi^*(G) < 5$. Subsequently Clarke and Jamison [3] discovered an interesting application of elementary game theory to χ^*, and showed that the infimum defining χ^* is actually attained.

For any graph G, denote by $\alpha(G)$ the maximum size of an independent set of G. We give a direct generalization of a standard lower bound on $\chi(G)$ in terms of $\alpha(G)$.

Lemma 3: If G is (r,s)-colourable then

$$r.|V(G)| \leq s.\alpha(G).$$

Proof: Observe that each colour class is an independent set, and count, in two different ways, the number of times a vertex receives a colour. □

§2. The complexity of (r,2r+1)-colouring

We are interested in the complexity of deciding whether or not a graph is (r,s)-colourable. Let us define

(r,s)-COLOURABILITY

Input: Graph G.

Question: Is G (r,s)-colourable?

Clearly (r,s)-COLOURABILITY belongs to NP for all r,s.

The complexity of these languages was first considered by Irving [27], who showed that for all $r \geq 1$, (r,2r+1)-COLOURABILITY is NP-complete. The reduction used was one of component design, from 3SAT. We give an easier proof, reducing from $\binom{2r+1}{r}$-COLOURABILITY by local replacement.

The next Lemma establishes some important facts about multicolourings of paths. Some notation: if $a,b \in \mathbb{Z}_s$, then $[a,b]$ denotes the subset $\{a,a+1,\ldots,b\}$ of $\mathbb{Z}_s$. Clearly $[a,b]$ and $[b,a]$ are both always defined, and are unequal unless $a = b$. We will write $|a - b| \geq k$ to mean $b \notin [a-k+1,a+k-1]$ (which obviously holds iff $a \notin [b-k+1,b+k-1]$).

Lemma 1: Let P_λ be a path of length λ with vertices $v_0,v_1,\ldots,v_\lambda$, and suppose we have a set of s available colours, which we can designate by $\mathbb{Z}_s$. If v_0 receives the colour set $[0,r-1]$, and if any colour set of the form $[k,k+r-1]$, where $k \in [\lambda r,-\lambda r]$, is assigned to v_λ, then the resulting (r,s)-colouring of v_0 and v_λ can be extended to an (r,s)-colouring of P_λ.

Proof: We prove this by induction on λ. It is obvious for $\lambda = 1$.

Given P_λ as described, consider the subpath $P_{\lambda-1}$ of length $\lambda - 1$ on the vertices $v_0,\ldots,v_{\lambda-1}$. Suppose v_0 is to receive colour set $[0,r-1]$. By our inductive hypothesis $v_{\lambda-1}$ can receive any colour set $[k,k+r-1]$ where $k \in [(\lambda-1)r, -(\lambda-1)r]$, and the resulting (r,s)-colouring is still extendable to an (r,s)-colouring of the whole of $P_{\lambda-1}$. But it can be seen that every colour set $[k,k+r-1]$, where $k \in [\lambda r,-\lambda r]$, is disjoint from some $[k,k+r-1]$ where $k \in [(\lambda-1)r, -(\lambda-1)r]$. Hence the result holds for P_λ, and so by induction for all λ. □

Theorem 2 (Irving [27]): For all integers $r \geq 1$, $(r,2r+1)$-COLOURABILITY is NP-complete.

Proof: Having observed above that it is in NP, we describe the polynomial transformation from $\binom{2r+1}{r}$-COLOURABILITY. Given any graph G, we form the graph G' by simply replacing each edge $uv \in E(G)$ by a copy of the path P_{2r-1} with endpoints u and v. This is illustrated in Figure 1.

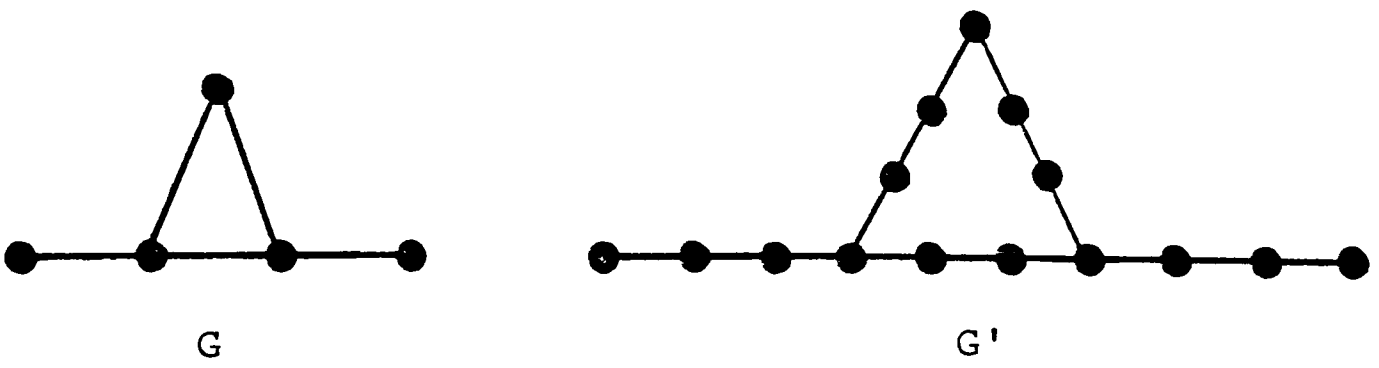


Figure 1: Example of the construction of G' from G, in the case $r = 2$.

To see that this works, consider $(r,2r+1)$-colourings of the path P_{2r-1} with endpoints u, v.

If u, v receive identical colour sets, then the rest of P_{2r-1} cannot be $(r,2r+1)$-coloured. For such an $(r,2r+1)$-colouring of P_{2r-1} yields an $(r,2r+1)$-colouring of C_{2r-1}. But $\alpha(C_{2r-1}) = r - 1$, so applying Lemma 1.3 gives a contradiction.

Now let u receive colour set $[0,r-1]$, where the set of available colours here is $\mathbb{Z}_{2r+1}$. Let v receive colour set $[k,k+r-1]$ where $k \not\equiv 0 \pmod{2r+1}$. The length of the path is $2r - 1$, and it is easy to see that

$$(2r-1)r \equiv 1 \pmod{2r+1}$$

and so

$$-(2r-1)r \equiv 2r \pmod{2r+1} .$$

Thus

$$k \not\equiv 0 \pmod{2r+1}$$

$$\Longleftrightarrow \quad k \in [(2r-1)r, -(2r-1)r].$$

It follows from Lemma 1 that the $(r,2r+1)$-colouring of u and v can be extended to an $(r,2r+1)$-colouring of the entire path P_{2r-1}. Since we have proved this for any $k \not\equiv 0 \pmod{2r+1}$, it follows immediately (by renaming colours if necessary) that any colouring of u and v by distinct sets of r colours can be extended to an $(r,2r+1)$-colouring of all of P_{2r-1}.

Thus

$$G \in \binom{2r+1}{r}\text{-COLOURABILITY}$$

$\Leftrightarrow$ G can be coloured by the r-subsets of $\{1,\ldots,2r+1\}$ such that adjacent vertices receive distinct colour sets

$\Leftrightarrow$ G' can be $(r,2r+1)$-coloured,

so

$$\binom{2r+1}{r}\text{-COLOURABILITY} \propto (r,2r+1)\text{-COLOURABILITY}. \qquad \square$$

This result enables us to prove the following Theorem on the complexity of computing χ_r.

Theorem 3: For all $r \in \mathbb{N}$, the problem of determining the r-chromatic number of a graph is NP-hard.

Proof: Let G be any graph. By definition of χ_r,

$$G \in (r,2r+1)\text{-COLOURABILITY} \Leftrightarrow \chi_r(G) \leq 2r + 1.$$

Thus an oracle for χ_r could be used to solve $(r,2r+1)$-COLOURABILITY. The result follows, from Theorem 2. $\square$

§3. The general multicolouring problem

We now consider the complexity of (r,s)-colourability in general. In view of Irving's result it is natural to expect that it is NP-complete whenever $s > 2r$. In this Section we prove that this is indeed the case.

Theorem 1: If r and s are positive integers with $s > 2r$ then (r,s)-COLOURABILITY is NP-complete.

Proof: Suppose r and s satisfy the hypotheses of the Theorem. The result is of course known for $r = 1$ so we assume $r \geq 2$.

To prove the Theorem, we will show

$$\binom{s}{r}\text{-COLOURABILITY} \propto (r,s)\text{-COLOURABILITY}.$$

The polynomial transformation is one of local replacement. The idea is once again to replace each edge uv of a given graph G by a graph $G_{r,s}$ with distinguished vertices u, v (see Figure 1) such that

(1) $G_{r,s}$ cannot be (r,s)-coloured so that u,v receive identical colour sets,

and (2) any assignment of distinct sets of r colours to u and v can be extended to an (r,s)-colouring of $G_{r,s}$.

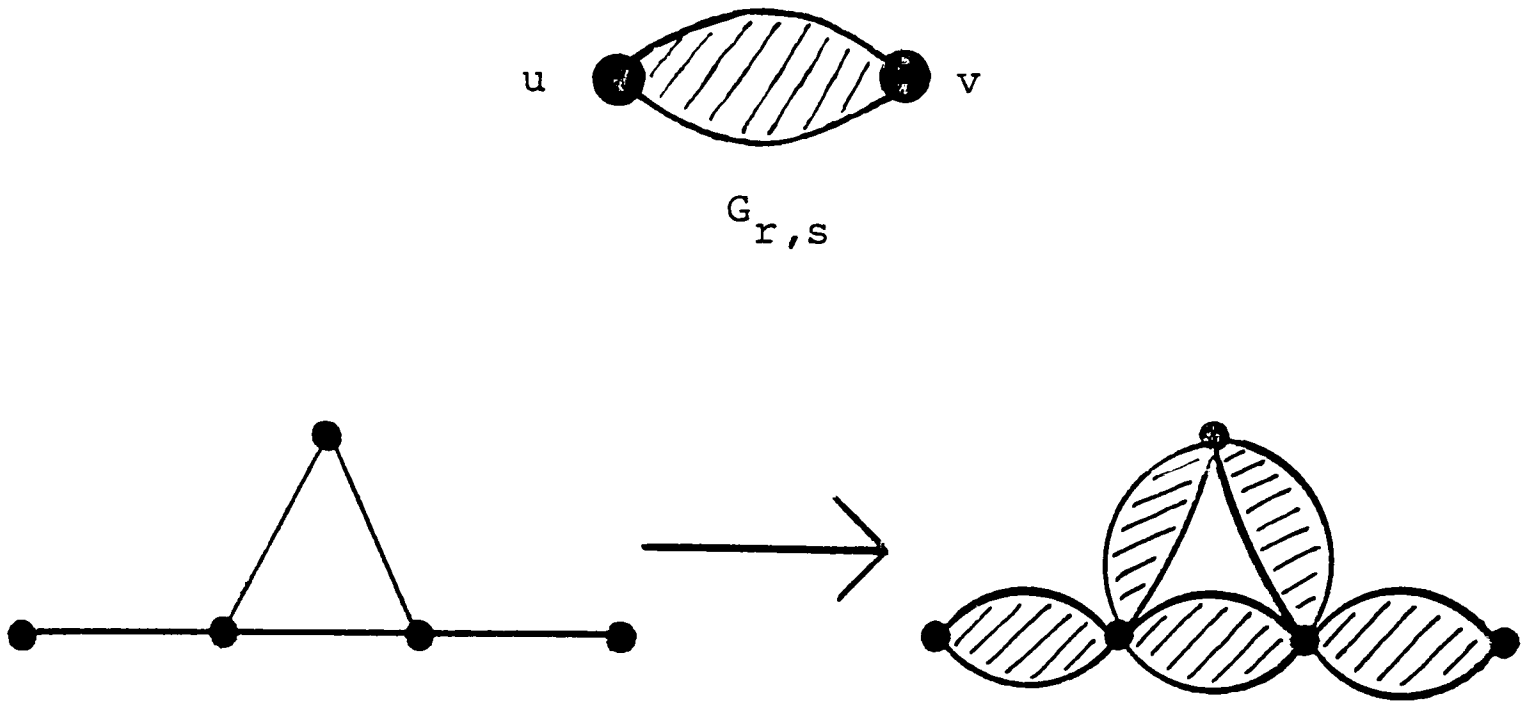


Figure 1: Example of the polynomial transformation.

We will refer back to these desired properties of $G_{r,s}$ on occasion, calling them "property (1)" and "property (2)".

In constructing $G_{r,s}$ we use some special graphs. Define the graph K_r^s as follows. It has $\binom{s}{r}$ vertices, which correspond to those subsets of $\{1,\ldots,s\}$ containing exactly r elements; two vertices are adjacent if they correspond to disjoint r-subsets of $\{1,\ldots,s\}$. These are the Kneser graphs introduced by Kneser [34] who conjectured that

$$\chi(K_r^s) = s - 2r + 2 \quad \forall\ r,s.$$

This was subsequently proved by Lovász [38]. These results, however, do not concern us.

K_r^s has the obvious canonical (r,s)-colouring, in which each vertex is coloured by the r-subset of $\{1,\ldots,s\}$ to which it corresponds. We denote this canonical (r,s)-colouring of K_r^s by c_0, so that if $v \in V(K_r^s)$ then $c_0(v)$ is the r-subset of $\{1,\ldots,s\}$ to which v

corresponds. Other (r,s)-colourings are usually denoted by c, and we always use $\{1,\ldots,s\}$ as the set of available colours.

The following fact is easily proved by an extension of Irving's proof [27] for the case $s = 2r + 1$.

Lemma 2: Up to a permutation of colours, K_r^s is uniquely (r,s)-colourable. □

The graph $G_{r,s}$ may now be described.

Case 1: r is odd.

Take two distinct copies A_r^s, B_r^s of K_r^s. Let u (resp. v) be a vertex of A_r^s (B_r^s). Choose $u_1 \in V(A_r^s)$ such that

$$|c_0(u) \cap c_0(u_1)| = \frac{r+1}{2} ,$$

and similarly choose $v_1 \in V(B_r^s)$ such that

$$|c_0(v) \cap c_0(v_1)| = \frac{r+1}{2} .$$

Finally, add an edge joining u_1 and v_1. (See Figure 2).

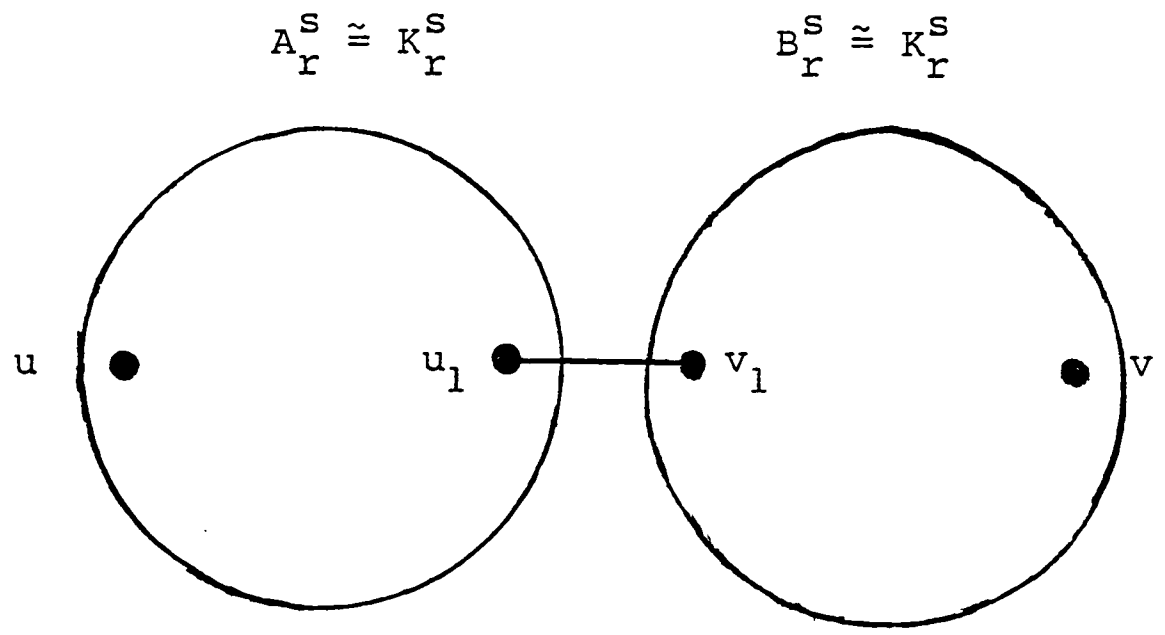


Figure 2: The graph $G_{r,s}$.

Case 2: r is even.

Again, take two disjoint copies A_r^s, B_r^s of K_r^s and let

u, v be vertices of A_r^s and B_r^s respectively. Choose $u_1 \in V(A_r^s)$ such that

$$|c_0(u) \cap c_0(u_1)| = \frac{r}{2},$$

and choose $v_1 \in V(B_r^s)$ such that

$$|c_0(v) \cap c_0(v_1)| = \frac{r}{2} + 1.$$

Finally, add an edge joining u_1 and v_1. (See Figure 2.)

The following is an immediate consequence of Lemma 2.

Lemma 3: Let $G_{r,s}$ be constructed as above.

(a) If r is odd, then an (r,s)-colouring c of u, u_1, v and v_1 can be extended to an (r,s)-colouring of the whole graph $G_{r,s}$ iff

$$|c(u) \cap c(u_1)| = \frac{r+1}{2}$$

and $$|c(v) \cap c(v_1)| = \frac{r+1}{2}.$$

(b) If r is even, then an (r,s)-colouring c of u, u_1, v and v_1 can be extended to an (r,s)-colouring of the whole graph $G_{r,s}$ iff

$$|c(u) \cap c(u_1)| = \frac{r}{2}$$

and $$|c(v) \cap c(v_1)| = \frac{r}{2} + 1.$$ □

We first demonstrate that $G_{r,s}$ has property (1).

Lemma 4: No (r,s)-colouring of $G_{r,s}$ gives the same colour set to u and v.

Proof: Let c be an (r,s)-colouring of $G_{r,s}$ and suppose $c(u) = c(v)$.

Now $u_1v_1 \in E(G_{r,s})$, so

$$c(u_1) \cap c(v_1) = \emptyset .$$

On the other hand,

$$|c(u_1) \cap c(v_1)| \geq |(c(u) \cap c(u_1)) \cap (c(v) \cap c(v_1))|$$

$$= |c(u) \cap c(u_1)| + |c(v) \cap c(v_1)|$$

$$- |(c(u) \cap c(u_1)) \cup (c(v) \cap c(v_1))|.$$

Now

$$|c(u) \cap c(u_1)| + |c(v) \cap c(v_1)| = r + 1$$

by Lemma 3, and

$$|(c(u) \cap c(u_1)) \cup (c(v) \cap c(v_1))| = |c(u) \cap (c(u_1) \cup c(v_1))|$$

as $c(u) = c(v)$,

$$\leq |c(u)| = r .$$

Hence

$$|c(u_1) \cap c(v_1)| \geq r + 1 - r = 1,$$

which is a contradiction. □

It remains to show that $G_{r,s}$ has property (2).

Lemma 5: Suppose $2 \leq k \leq r + 1$, and let vertices u and v of $G_{r,s}$ receive the colour sets $[1,r]$ and $[k,k + r - 1]$ respectively. Then we can extend this (r,s)-colouring of u and v to an (r,s)-colouring of $G_{r,s}$.

Proof: Let vertices u, v of $G_{r,s}$ receive the colour sets $[1,r]$, $[k,k + r - 1]$ respectively, where $2 \leq k \leq r + 1$.

We (r,s)-colour $G_{r,s}$ as follows.

Case 1: r odd.

If $2 \le k \le \frac{r+3}{2}$, put

$c(u_1) = [1, \frac{r+1}{2}] \cup [r+2, \frac{3r+1}{2}]$;

$c(v_1) = [\frac{r+3}{2}, r+1] \cup [\frac{3r+3}{2}, 2r]$.

If $\frac{r+5}{2} \le k \le r+1$, put

$c(u_1) = [\frac{r+1}{2}, r] \cup [\frac{3r+3}{2}, 2r]$;

$c(v_1) = [1, \frac{r-1}{2}] \cup [r+1, \frac{3r+1}{2}]$.

Case 2: r even.

If $2 \le k \le \frac{r}{2} + 1$, put

$c(u_1) = [1, \frac{r}{2}] \cup [r+2, \frac{3r}{2}+1]$;

$c(v_1) = [\frac{r}{2}+1, r+1] \cup [\frac{3r}{2}+2, 2r]$.

If $\frac{r}{2} + 2 \le k \le r+1$, put

$c(u_1) = [\frac{r}{2}+1, r] \cup [\frac{3r}{2}+2, 2r+1]$;

$c(v_1) = [1, \frac{r}{2}-1] \cup [r+1, \frac{3r}{2}+1]$.

It is routine to check that in each case

$$|c(u_1)| = |c(v_1)| = r, \quad c(u_1) \cap c(v_1) = \emptyset ,$$

and that if r is odd then

$$|c(u) \cap c(u_1)| = |c(v) \cap c(v_1)| = \frac{r+1}{2} ,$$

while if r is even then

$$|c(u) \cap c(u_1)| = \frac{r}{2}$$

and $|c(v) \cap c(v_1)| = \frac{r}{2} + 1.$

It follows that the (r,s)-colourings described of u, u_1, v, v_1 can always be extended to (r,s)-colourings of A_r^s and B_r^s, and hence to the whole graph $G_{r,s}$. □

Corollary 6: Any assignment of distinct sets of r colours to the vertices u, v of $G_{r,s}$ can be extended to an (r,s)-colouring of $G_{r,s}$.

Proof: This is immediate from Lemma 4 (just rename colours as necessary). □

Theorem 1 is now immediate.

Proof of Theorem 1 concluded.

Lemma 4 shows that $G_{r,s}$ has property (1), and Corollary 6 establishes property (2). Hence the function which for any input graph G replaces each edge uv by a copy of $G_{r,s}$ is a polynomial transformation demonstrating that

$$\binom{s}{r}\text{-COLOURABILITY} \propto (r,s)\text{-COLOURABILITY}.$$ □

§4. Some related graph homomorphism problems

A graph G is said to be homomorphic to a graph H if there exists a map $\phi : V(G) \to V(H)$ such that $\{u,v\} \in E(G) \Rightarrow \{\phi(u),\phi(v)\} \in E(H)$; such a map is a homomorphism. Equivalently, we require that G contains pairwise disjoint independent sets $S_1,\ldots,S_k$ such that if, for each i, all the vertices in S_i are identified, the resulting graph is isomorphic to a subgraph of H.

Clearly a graph is k-colourable iff it is homomorphic to K_k. We observe that a graph is (r,s)-colourable iff it is homomorphic to K^s_r, where K^s_r is a Kneser graph (defined in §3).

Consider now the language defined for any fixed graph H by

$$H\text{-HOMOMORPHISM} = \{G \mid G \text{ is homomorphic to } H\}.$$

Clearly this is in NP for all (finite) H. It is an unsolved problem to determine precisely for which H this language is NP-complete, and for which H it is in P (one would expect every H to fall into one of these two categories). The belief (see [39]) is that H-HOMOMORPHISM $\in$ P iff H

is 2-colourable, and that it is NP-complete otherwise. It is clear that if H is 2-colourable then

H-HOMOMORPHISM = 2-COLOURABILITY,

and so is in P; still open is the conjecture that if H is not 2-colourable then H-HOMOMORPHISM is NP-complete.

We know this conjecture is true for the graphs K_k where $k \geq 3$ and Theorem 3.1 implies that it is true for the graphs K_r^s for all r, s such that $s > 2r$. It has also been shown to be true for $H \cong C_{2k+1}$ by Maurer, Sudborough and Welzl [39], who also establish several properties of the languages H-HOMOMORPHISM which do seem to suggest that the above conjecture is true.

The following Theorem gives a simple technique for proving NP-completeness of some homomorphism problems, and enables us to give a shorter proof of Maurer, Sudborough and Welzl's result for odd circuits (Corollary 2).

Theorem 1: Let H be a graph. Suppose there exists $k \in \mathbb{N}$ such that

(i) for every two vertices u, v of H there exists a walk from u to v in H of length $2k + 1$, and

(ii) there is no odd circuit in H of length less than or equal to $2k + 1$.

Then H-HOMOMORPHISM is NP-complete.

Proof: We describe a polynomial transformation

$|V(H)|$-COLOURABILITY $\propto$ H-HOMOMORPHISM.

Let G be a graph. Replace each edge uv by a path of length $2k + 1$ with endpoints u, v to yield a new graph G'. It can be seen that no homomorphism ϕ from G' to H maps two endpoints of such a path P_{2k+1} in G' to a single vertex, because then H would contradict (ii) above. Also any mapping of $u,v \in V(G)$ to two distinct vertices of H can be extended to a homomorphism of the entire path joining them, in G', to H, by property (i) above. Thus the replacement of an edge uv in G by a path P_{2k+1} in G' has the effect of forbidding vertices u, v from being assigned, under any homomorphism to H, to the same vertex in H, but allowing them to be assigned to any two different vertices in H. Hence G is $|V(H)|$-colourable iff G' is homomorphic to H. □

Corollary 2[39]: C_{2m+1}-HOMOMORPHISM is NP-complete for all $m \in \mathbb{N}$.

Proof: Observe that with $k = m - 1$ the hypotheses of Theorem 1 are satisfied. □

Theorem 1 can be used to show that a number of other graphs yield NP-complete homomorphism problems: for example, the Grötsch graph (Figure 1) (see also [39]).

Generalizations of Theorem 1 could probably be proved, depending on the graphs $\hat{G}_{r,s}$ used in the proof of Theorem 3.1 rather than the paths P_{2k+1}. However the situation becomes substantially more complicated, more restricted and less interesting, and we have not pursued it.

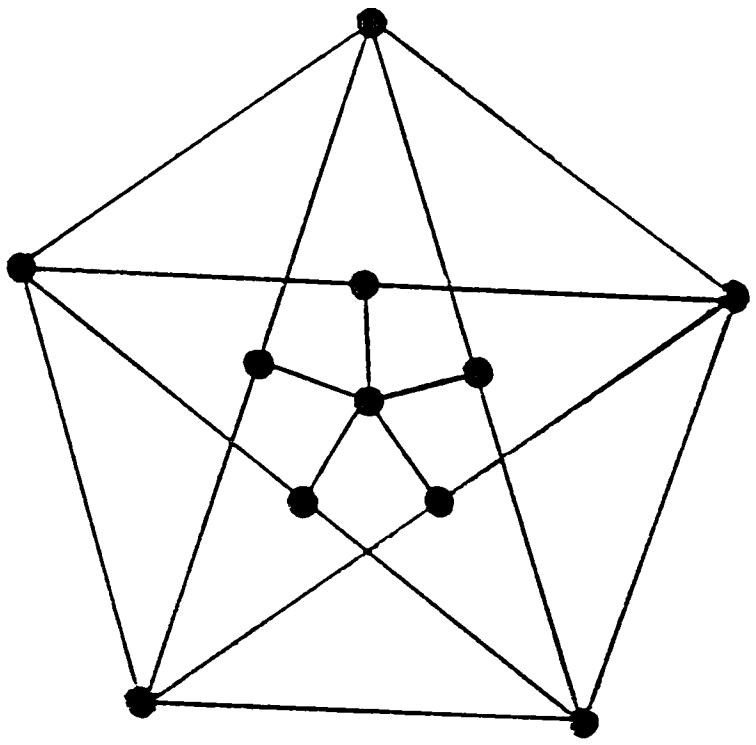


Figure 1: The Grötsch graph.

We close by noting that our proof of Theorem 2.2 showed, essentially, that the graphs $H \cong K_r^{2r+1}$ satisfy the hypotheses of Theorem 1 (with $k = r - 1$).

CHAPTER 5

COLOURING AND PROMISE PROBLEMS

In this chapter we consider promise problems related to graph colouring. In the first section we consider 3-colourings of graphs with high vertex degrees and obtain a polynomial time algorithm for finding a 3-colouring of any 3-colourable graph G whose minimum degree is greater than $\frac{8}{15}|V(G)|$. The second section concerns the general problem of separability of disjoint pairs of languages, and we develop a framework for classifying such language pairs. Finally in §3 we apply the concepts of §2 to some promise problems involving graph colourability, where the promise itself concerns the chromatic number, and obtain some NP-hardness results.

§1. 3-colouring graphs with high vertex degrees

It is well known that the problem of deciding whether a graph G is k-colourable is NP-complete for all $k \geq 3$. Various restricted classes of graphs have been studied from the point of view of the complexity of colouring their members. In this section we look at the complexity of 3-colouring graphs whose vertex degrees are bounded below by some fixed fraction of the number of vertices.

We are motivated by the work of Welsh and Petford [68] on their randomised 3-colouring algorithm. The success of this algorithm in quickly finding 3-colourings, when such colourings exist, for graphs whose vertex degrees are all high prompted their conjecture that, for all $\alpha > 0$, graphs G with minimum degree at least $\alpha|V(G)|$ could be 3-coloured in polynomial time. We will prove this conjecture for all $\alpha > \frac{8}{15}$.

Consider the following algorithm which 3-colours certain of the vertices of some graphs.

Algorithm A.

1. Input: Graph G, and specified triangle T in G.
2. Colour the vertices of T (arbitrarily) using three colours, say c_1, c_2, c_3.
3. If there is no uncoloured vertex of G adjacent to two differently coloured vertices,

   output the partial 3-colouring of G so far found.

   Otherwise, continue.
4. Choose some $v \in V(G)$ such that v has at least two differently coloured neighbours. If in fact v has

three differently coloured neighbours then G is not 3-colourable; output a message to this effect.
Otherwise, colour v by that colour in $\{c_1,c_2,c_3\}$ not used by any of its neighbours.

5. Go to Step 3.

We make some basic observations about the behaviour of this algorithm.

Lemma 1:

(a) The time complexity of A is $O(|V|^2)$.

(b) For any input graph G with a triangle T, A will either output the conclusion that G is not 3-colourable, or find a partial 3-colouring of G (in which case G may or may not be 3-colourable).

(c) If G is 3-colourable then there exists a 3-colouring of G which extends the partial 3-colouring found by A. □

The proof of Lemma 1 is very straightforward and we omit it.

It is clear that A just finds the partial 3-colouring of G forced by a 3-colouring of the triangle T in G (see §3.4). Thus A can be used to solve FORCED 3COL (see p. 110) as follows. Let G be a graph. For all triangles T of G, apply A to (G,T). $G \in$ FORCED 3COL iff for some triangle T of G, the algorithm A applied to (G,T) finds a 3-colouring of G.

For any $\alpha \geq 0$ define G_α to be the set of all graphs G with minimum degree greater than or equal to $\alpha \cdot |V(G)|$. We now examine the performance of A on graphs in G_α.

Lemma 2: Let G be a graph in G_α with a triangle T. If G is 3-colourable then A applied to (G,T) will 3-colour at least $(3\alpha-1)\cdot|V(G)|$ of the vertices of G.

Proof: Let G be a 3-colourable graph in G_α with a triangle T. Put $n = |V(G)|$. Apply A to (G,T). For each $i \in \{1,2,3\}$, let $V_i \subseteq V(G)$ be the set of vertices which have colour c_i once A has stopped, and choose any $v_i \in V_i$. Define, for each i,

$$N_i = N_G(v_i) \setminus (V_1 \cup V_2 \cup V_3) .$$

Now v_i has no neighbours in V_i and $|N_G(v_i)| \geq \alpha n$ by definition of G_α, so

(1) $|N_i| \geq \alpha n - |V_1 \cup V_2 \cup V_3| + |V_i|$.

By the 3-colourability of G and our assumption that A has stopped, the N_i are disjoint. Hence

$$|V(G) \setminus (V_1 \cup V_2 \cup V_3)| \geq \sum_{i=1}^{3} |N_i| .$$

Combined with (1) this gives

$$|V_1 \cup V_2 \cup V_3| \geq (3\alpha - 1)n,$$

which is the desired lower bound on the number of vertices of G coloured by A. □

We remark before proceeding further that if $\alpha > \frac{2}{3}$ then any $G \in G_\alpha$ contains a subgraph isomorphic to K_4 (by Turán's Theorem; see, e.g., [25, p.18]) and so is not 3-colourable.

Our main algorithms A_1 and A_2, which use A, may now be described. Both these algorithms will find 3-colourings

(whenever they exist) for graphs in G_α when α is sufficiently high, as we show below.

Algorithm A_1.

1. Input: Graph G.
2. Find a triangle T in G. (If none exists, output a message to that effect).
3. Apply A to (G,T).

Algorithm A_2.

1. Input: Graph G.
2. Apply A_1 to G.
3. If A_1 finds that G is not 3-colourable, or finds a 3-colouring of G, then stop.
   Otherwise, let U be the set of uncoloured vertices of G. Determine whether <U> is 2-colourable. If so, find a 2-colouring of <U> using colours not used for any coloured neighbour of any vertex in U, and output the resulting colouring of G. Otherwise, output the message that G is not 3-colourable.

Apart from the initial step of finding a triangle, these algorithms run in time $O(|V|^2)$. Thus, if a triangle is sought by the elementary method of checking all vertex triples, then A_1 and A_2 have time complexity $O(|V|^3)$. However it is not difficult to show that linear time is sufficient for finding triangles in members of G_α for $\alpha > \frac{1}{2}$. Hence, for this class of graphs, A_1 and A_2 can be made to run in time $O(|V|^2)$.

Theorem 3:

(a) If $\alpha > \frac{4}{7}$ then A_1 will find a 3-colouring of any 3-colourable $G \in G_\alpha$.

(b) If $\alpha > \frac{8}{15}$ then A_2 will find a 3-colouring of any 3-colourable $G \in G_\alpha$. These results are best possible (for A_1 and A_2) with respect to the bounds for α.

Proof: Suppose $\alpha > \frac{1}{2}$ and let G be a 3-colourable member of G_α. Put $n = |V(G)|$. Let T be any triangle in G; there must be at least one, by Turán's Theorem (see [25, p. 17]) Now A_2 begins by applying A_1, and A_1 begins by applying A, so we consider first the result of applying A to (G,T). Suppose the colours used are c_1, c_2 and c_3, and suppose that when A stops the set of vertices with colour c_i is V_i, for each $i \in \{1,2,3\}$. Write $V_0 = V_1 \cup V_2 \cup V_3$ and $U = V(G) \setminus V_0$. If $U = \emptyset$ we are done, so assume $U \neq \emptyset$. By Lemma 2,

$$(2) \qquad |V_0| \geq (3\alpha - 1)n ,$$

and it follows that $|U| < \frac{1}{2}n$. Now any member of U has at least $\lceil \alpha n \rceil \geq \lceil \frac{1}{2}n \rceil$ neighbours and so must be adjacent to a member of V_0. Furthermore, since A has halted with V_0 as the set of coloured vertices, no member of U can be adjacent to two differently coloured members of V_0. Also G is 3-colourable so no member of U can have three differently coloured neighbours in V_0, by Lemma 1(c). Hence, if we put

$$N_i = \{u \in U \mid u \sim v \text{ for some } v \in V_i\}$$

for each $i \in \{1,2,3\}$, then U is the disjoint union of the N_i.

Now for each $i \in \{1,2,3\}$, any vertex in V_i has all its neighbours in $N_i \cup V_{j_1} \cup V_{j_2}$, where $\{j_1,j_2\} = \{1,2,3\} \setminus \{i\}$. Hence we obtain the following inequalities.

$$(3) \quad \alpha n \leq |N_1| + |V_2| + |V_3|,$$

$$(4) \quad \alpha n \leq |V_1| + |N_2| + |V_3|,$$

$$(5) \quad \alpha n \leq |V_1| + |V_2| + |N_3|.$$

We will use these inequalities shortly.

Since G is 3-colourable, by Lemma 1(c) there exists a 3-colouring c of G which extends the partial 3-colouring found by A. For each ordered pair $i,j \in \{1,2,3\}$ where $i \neq j$, denote by N_{ij} the subset of N_i consisting of vertices which receive colour c_j under c. Then for each i, $N_i = N_{ij_1} \cup N_{ij_2}$ where $\{j_1,j_2\} = \{1,2,3\} \setminus \{i\}$, and these unions are disjoint. Each N_{ij} is by definition an independent set in G.

We consider the following mutually exclusive cases.

(i) Some N_i is a nonempty independent set in G.

(ii) All the N_{ij} are nonempty.

(iii) There is exactly one nonempty N_i, say N_{i_1}, and $N_{i_1j_1}$ and $N_{i_1j_2}$ are also nonempty (where $\{j_1,j_2\} = \{1,2,3\} \setminus \{i_1\}$).

(iv) There are exactly two nonempty N_i, say N_{i_1} and N_{i_2}, and $N_{i_1j_1}$, $N_{i_1j_2}$, $N_{i_2j_3}$, and $N_{i_2j_4}$ are also nonempty (where $\{j_1,j_2\} = \{1,2,3\} \setminus \{i_1\}$ and $\{j_3,j_4\} = \{1,2,3\} \setminus \{i_2\}$).

Case (i): Suppose, w.ℓ.o.g., that $N_3 \neq \emptyset$ and that N_3 is independent in G. Suppose $v \in N_3$. Then

$$N_G(v) \subseteq N_1 \cup N_2 \cup V_3 .$$

Hence

$$\alpha n \leq |N_1| + |N_2| + |V_3|.$$

Adding this to (5) above, we find $\alpha \leq \frac{1}{2}$, a contradiction.

Case (ii): Considering the neighbourhoods and degrees of a typical member of each N_{ij}, we obtain the following inequalities.

$$\alpha n \leq |V_1| + |N_{13}| + |N_{21}| + |N_{23}| + |N_{31}| ,$$

$$\alpha n \leq |V_1| + |N_{12}| + |N_{31}| + |N_{32}| + |N_{21}| ,$$

$$\alpha n \leq |V_2| + |N_{21}| + |N_{32}| + |N_{31}| + |N_{12}| ,$$

$$\alpha n \leq |V_2| + |N_{23}| + |N_{12}| + |N_{13}| + |N_{32}| ,$$

$$\alpha n \leq |V_3| + |N_{32}| + |N_{13}| + |N_{12}| + |N_{23}| ,$$

$$\alpha n \leq |V_3| + |N_{31}| + |N_{23}| + |N_{21}| + |N_{13}| ,$$

Adding these inequalities and dividing by 2 we obtain

$$3\alpha n \leq 2n - |V_0|.$$

But we know (2) that $|V_0| \geq (3\alpha - 1)n$. It follows that $\alpha \leq \frac{1}{2}$, a contradiction.

Case (iii): Suppose w.ℓ.o.g. that $i_1 = 1$. Thus, $N_{12} \neq \emptyset$, $N_{13} \neq \emptyset$, and $N_2 = N_3 = \emptyset$. Suppose w.ℓ.o.g. that $|N_{12}| \geq |N_{13}|$. Take any $v \in N_{12}$. Then

$$N(v) \subseteq N_{13} \cup V_1$$

and so

$$\alpha n \leq |N_{13}| + |V_1|$$

$$\leq \tfrac{1}{2}|N_1| + |V_1| .$$

Adding this to (3) gives

$$2\alpha n \leq \frac{3}{2}|N_1| + |V_0| .$$

Since $U = N_1$, we have

$$|V_0| \leq (3 - 4\alpha)n .$$

By (2) it follows that $\alpha \leq \frac{4}{7}$.

Case (iv): By considering, in turn, the neighbourhoods of typical members of N_{12}, N_{13}, N_{21} and N_{23}, we obtain the following inequalities.

(6) $\alpha n \leq |V_1| + |N_{13}| + |N_{23}| + |N_{21}|$,

(7) $\alpha n \leq |V_1| + |N_{12}| + |N_{21}|$,

(8) $\alpha n \leq |V_2| + |N_{13}| + |N_{12}| + |N_{23}|$,

(9) $\alpha n \leq |V_2| + |N_{12}| + |N_{21}|$.

Adding (6) and (8), and then adding $|N_{12}| + |N_{21}|$ to each side, gives

(10) $|N_{12}| + |N_{21}| \leq |V_1| + |V_2| + 2(n - |V_0|) - 2\alpha n.$

Adding (7) and (9) gives

(11) $2\alpha n - |V_1| - |V_2| \leq 2(|N_{12}| + |N_{21}|).$

(10) and (11) now yield (eliminating $|N_{12}| + |N_{21}|$)

(12) $\quad 3|V_3| \leq (4 - 6\alpha)n - |V_0|$.

Now adding (3) and (4), remembering that $N_3 = \emptyset$, gives

(13) $\quad (2\alpha - 1)n \leq |V_3|$.

From (12) and (13) we obtain (eliminating $|V_3|$)

$$|V_0| \leq (7 - 12\alpha)n \ .$$

Together with (2) this implies $\alpha \leq \frac{8}{15}$.

This concludes our treatment of the four cases. We have seen that, with $\alpha > \frac{1}{2}$, cases (i) and (ii) cannot arise. Furthermore, if $\alpha > \frac{4}{7}$, none of the four cases can arise and so A applied to (G,T) must correctly 3-colour the whole of G. This bound is sharp: if $\alpha = \frac{4}{7}$, the graph G of Figure 1 is in G_α and yet if T is any triangle in G then A applied to (G,T) will not 3-colour all of G.

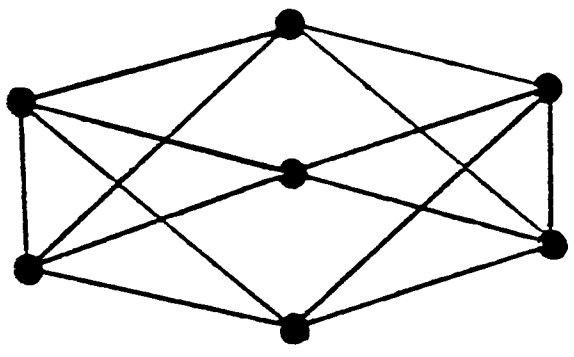


Figure 1

Finally, if $\frac{8}{15} < \alpha \leq \frac{4}{7}$, then the only way A (when applied to (G,T)) can fail to colour all of G is if Case (iii) arises. But then the uncoloured vertices of G induce the 2-colourable subgraph $\langle N_{i_1} \rangle$ of G. Now all coloured neighbours of vertices in N_{i_1} are coloured c_{i_1} by A. It

follows that Step 3 of A_2 will take care of colouring N_{i_1}, and that A_2 applied to the graph G with triangle T will correctly 3-colour the whole of G. Again, our lower bound on α is sharp, as illustrated by the graph $G \in G_{8/15}$ shown in Figure 2. There, a circle

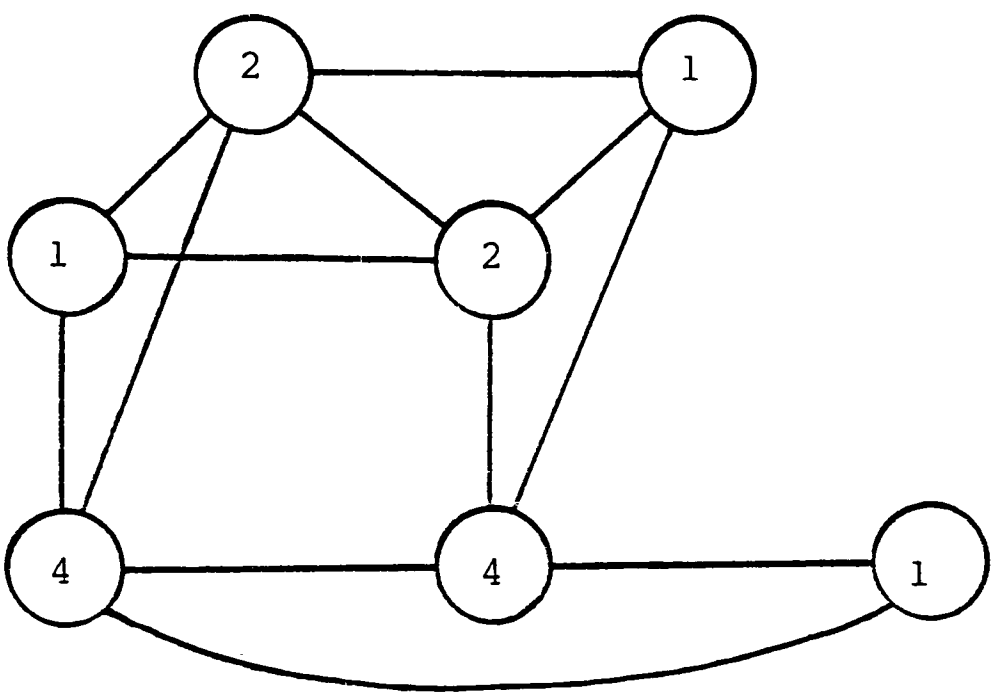
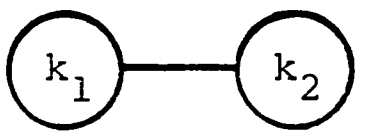


Figure 2

with a number k inside indicates an independent set of k vertices; a line joining two circles indicates that the corresponding independent sets together induce a complete bipartite subgraph of G. Thus,  indicates two disjoint independent sets of sizes k_1 and k_2 whose union induces a subgraph of G isomorphic to K_{k_1,k_2}. The triangle T may be chosen to be any triangle formed by taking one vertex from each of the three independent sets at the bottom of the diagram. □

We found it convenient, in the above proof, to prove (a) while proving (b). (a) alone could have been proved more quickly, and is related to the result of Bollobás [2] that 3-colourable graphs in G_α where $\alpha > \frac{4}{7}$ are uniquely 3-colourable.

Theorem 3 has subsequently been extended by K.J. Edwards [10], who completely resolved the aforementioned conjecture of Welsh and Petford:

Theorem 4 [10]: For any fixed $k \geq 3$, the problem of deciding whether an arbitrary graph in G_α is k-colourable is

(i) in P if $\alpha > \frac{k-3}{k-2}$;

and (ii) NP-complete if $0 \leq \alpha \leq \frac{k-3}{k-2}$. □

Thus, 3-COLOURABILITY restricted to graphs in G_α is in P for all $\alpha > 0$. Edwards proves (i) by giving a polynomial time algorithm for finding a k-colouring when one exists. When $k = 3$ the algorithm has time complexity $O(n^d)$ where $d \geq 2 - \frac{\log 3}{\log(1-\alpha)}$. Thus our algorithms are still faster if $\alpha > \frac{8}{15}$.

§2. Promise problems and language separability

The work of this section and the next is motivated by the following "promise problem", posed by D.J.A. Welsh: given that a graph G is 3-colourable, find a 4-colouring of G. Thus we seek a polynomial time algorithm which correctly 4-colours all 3-colourable graphs (and can be expected to 4-colour some 4-chromatic graphs as well). If A is such an algorithm, then the language L_A of graphs which are correctly 4-coloured by the algorithm is in P, and it <u>separates</u> $\{G|\chi(G) \leq 3\}$ and $\{G|\chi(G) \geq 5\}$ in the sense that the former is contained in L_A and the latter is contained in L_A^c. With such situations in mind, we study the separation of disjoint pairs of languages, particularly (as is the case in the above example) pairs consisting of one language belonging to NP and another belonging to co-NP.

We begin with some formal definitions for promise problems and then develop a framework within which to study the separation of disjoint pairs of languages. In §3 we return to the specific examples which motivate us, obtaining some NP-hardness results. Unfortunately most of the really important questions raised appear to be very difficult and we have not been able to resolve them.

<u>Definitions: Promise problems [11, 12, 59]</u>: A <u>promise problem</u> is a pair (Q,R) where $Q,R \subseteq \Sigma^*$ are languages. Q is called the <u>promise</u>. A DTM M <u>solves</u> a promise problem (Q,R) if M halts on all inputs and, for all $x \in Q$, M accepts x if $x \in R$ and rejects x if $x \notin R$. (The behaviour of M for $x \notin Q$ is irrelevant.) A promise

problem is solvable in polynomial time if it is solvable by a polynomial time DTM. A language L is a solution to (Q,R) if L = L(M) for some DTM M which solves (Q,R). Thus, a solution to (Q,R) is just a recursive language L such that

$$Q \cap R \subseteq L \quad \text{and} \quad Q \setminus R \subseteq L^c .$$

(Q,R) is NP-hard if every solution to (Q,R) is NP-hard.

Note that if (Q,R) is a promise problem and $Q = \Sigma^*$, then we are essentially just talking about the language R, and that if $Q = \emptyset$ then the concept is meaningless in the sense that any DTM solves (Q,R) (provided it always halts). We will describe promise problems in the Input/Promise/Question format of [12].

Also note that the problems of §1 are promise problems, in which the promises are the sets G_α (for α fixed).

The problems that motivate us here were originally formulated as promise problems, and we will make some use of the above definitions. However, we will generally find it more convenient to speak of pairs of languages and separation thereof, which we define now.

Definitions: Separability: A language pair is simply a pair of disjoint languages and is written using square brackets (to avoid confusion with promise problems). Thus the language pair consisting of L_1 and L_2 is written $[L_1, L_2]$. The language L separates $[L_1, L_2]$ if $L_1 \subseteq L$ and $L_2 \subseteq L^c$. The language pair $[L_1, L_2]$ is polynomially separable [23] (p-separable for short) if there exists $L \in P$ such that L separates $[L_1, L_2]$. The connection with

promise problems is direct: if L is recursive, then L separates $[L_1,L_2]$ iff L is a solution to the promise problem $(L_1 \cup L_2, L_1)$. We say $[L_1,L_2]$ is NP-hard if every recursive language separating $[L_1,L_2]$ is NP-hard. Finally, $[L_1,L_2]$ is an NP-co-NP pair if $L_1 \in$ NP and $L_2 \in$ co-NP.

It is not usually easy to tell whether two languages are polynomially separable. It is nevertheless an important area with applications to public-key cryptography (see e.g. [11,23]). We present some elementary results of Selman and Yacobi.

Theorem 1 [11, 59]: If the pair $[L_1,L_2]$ is p-separable then both L_1 and L_2 are polynomial time Turing reducible to $\Sigma^* \setminus (L_1 \cup L_2)$. □

Theorem 2 [11, 59]: If $L_1 \cup L_2 \in P$ and L_1 (and hence L_2) is NP-hard, then $[L_1,L_2]$ is NP-hard. □

Little else is known about when, in general, two languages are p-separable. The area seems to be beset with deep difficulties of the sort that make so many basic questions in complexity theory somewhat beyond the range of known mathematical techniques. An important open problem is the following, of interest in itself but originally arising from the study of the complexity of public-key cryptosystems.

Conjecture 1 [59]: There is no NP-hard language pair $[L_1,L_2]$ such that

(i) $L_1 \cup L_2 \in$ NP,

and (ii) $[L_1,L_2]$ is separable by a language in NP and by a language in co-NP.

The truth of this Conjecture would imply NP ≠ co-NP, and so proving the Conjecture is likely to be very difficult. It would also imply [11] that NP ≠ U, where U is the class of languages recognized by polynomial time NDTMs which have at most one accepting computation for each input [64]. The question of whether NP ≠ U is likely to be extremely hard in view of the relativization results of Rackoff [43].

Of interest also are the following related questions, which we have been unable to resolve.

Questions:

(1) Is there an NP-hard language pair $[L_1,L_2]$ such that $L_1,L_2 \in$ NP (a special case of Conjecture 1)?

If P ≠ NP then an affirmative answer here would give an affirmative answer to the next question.

(2) Is there a pair $[L_1,L_2]$, with $L_1,L_2 \in$ NP, which is not p-separable?

The answer is trivially "yes" if P ≠ NP ∩ co-NP. Grollman and Selman [23] have shown that the answer is yes if P ≠ U, and that in any event a positive answer would answer the next question positively.

(3) Is there a pair $[L_1,L_2]$, with L_1 and L_2 both NP-complete, which is not p-separable?

Remark: It has also been shown [11] that if we replace (i) in Conjecture 1 by the condition $L_1 \cup L_2 \in$ co-NP, then the resulting assertion is true iff NP = co-NP. There is however no reason why this should affect our opinion of Conjecture 1.

We do not consider these open questions further.

We now define two reductions for language pairs, analogous to the polynomial transformations of ordinary complexity theory.

Definitions: Let $L_1, L_2 \subseteq \Sigma_1^*$ and $L_1', L_2' \subseteq \Sigma_2^*$ be languages, and suppose $f : \Sigma_1^* \to \Sigma_2^*$. f is said to be a strong p-transformation from $[L_1, L_2]$ to $[L_1', L_2']$ if f is polynomial time computable and for all $x \in \Sigma_1^*$

$$x \in L_1 \iff f(x) \in L_1'$$

and $$x \in L_2 \iff f(x) \in L_2' .$$

(Thus, f is simultaneously a polynomial transformation from L_1 to L_1' and from L_2 to L_2'.) If such a transformation exists we write

$$[L_1, L_2] \propto_s [L_1', L_2'].$$

If $g : \Sigma_1^* \to \Sigma_2^*$ is polynomial time computable and, for all $x \in \Sigma_1^*$,

$$x \in L_1 \Rightarrow g(x) \in L_1'$$

and $$x \in L_2 \Rightarrow g(x) \in L_2' ,$$

then g is a weak p-transformation from $[L_1, L_2]$ to $[L_1', L_2']$. If such a transformation exists we write

$$[L_1, L_2] \propto_w [L_1', L_2'] .$$

The following fact is clear.

Lemma 3: Both strong and weak p-transformations are transitive, and if $[L_1, L_2] \propto_s [L_1', L_2']$ then $[L_1, L_2] \propto_w [L_1', L_2']$. □

Remark: Weak p-transformability does not in general imply strong p-transformability.

To see this, put

$$L_1' = \{G \mid G \text{ is connected}\}$$

and $L_2' = (L_1')^c$,

and let L_1, L_2 be any subsets of L_1', L_2' respectively known not to belong to P. (Such L_1 and L_2 certainly exist: in fact any subset of Σ^* can be encoded as a subset of the connected, or disconnected, graphs.) Clearly

$$[L_1, L_2] \propto_w [L_1', L_2']$$

via the identity function. However no strong p-transformation from the first pair to the second pair exists: firstly, such a function must map $\Sigma^* \setminus (L_1 \cup L_2)$ to the empty set which is a contradiction, and secondly, such a function would show that $L_1 \propto L_1'$ and hence that $L_1 \in P$, again a contradiction.

The notions of weak and strong p-transformability do coincide, however, if $L_2 = L_1^c$.

The next two results, whose proofs are trivial, establish some simple properties of strong and weak p-transformability.

Lemma 4: If $[L_1, L_2]$ is NP-hard and $[L_1, L_2] \propto_w [L_1', L_2']$ then $[L_1', L_2']$ is NP-hard. □

Lemma 5: If $[L_1, L_2] \propto_s [L_1', L_2']$ where $[L_1', L_2']$ is an NP-co-NP pair then $[L_1, L_2]$ is an NP-co-NP pair. □

The class of NP-co-NP pairs is, in many respects, a natural analogue for language pairs of the class NP; we will return to this later. Lemma 5 can then be regarded as the language pair companion to the elementary result that if $L \propto L'$ and $L' \in NP$ then $L \in NP$. It does not hold with $\propto_w$ instead of $\propto_s$. The example given above to show that weak p-transformability does not imply strong p-transformability suffices to establish this point also (provided L_1 and L_2 are known not to be in NP $\cup$ co-NP).

Theorem 6 below gives some elementary equivalent conditions for p-separability. These conditions show that questions of p-separability of language pairs can be restated as questions about p-transformations to "target" pairs already known to be p-separable. The Theorem corresponds, in a sense, to the trivial fact of ordinary complexity theory that if L and L' are languages with $L' \in P$, then

$$L \in P \text{ iff } L \propto L'.$$

It also gives some conditions under which weak p-transformability implies strong p-transformability.

A technical definition[1] is useful here: a <u>polynomial embedding</u> of L_1 into L_2 is a polynomial time computable function $f : \Sigma^* \to \Sigma^*$ such that $f(L_1) \subseteq L_2$.

<u>Theorem 6</u>: Let $[L_1, L_2]$ be any language pair, and let $[L_1', L_2']$ be any p-separable language pair such that there exist polynomial embeddings of L_1 into L_1' and L_2 into L_2'. Then the following are equivalent:

(i) $[L_1, L_2]$ is p-separable;

(ii) $[L_1, L_2] \propto_w [L_1', L_2']$.

---

[1] This definition is not needed.

If in addition $L_1 \propto L_1'$, $L_2 \propto L_2'$, and the following assumption holds,

(A) $\quad L_1 \cup L_2 \neq \Sigma^* \Rightarrow L_1' \cup L_2' \neq \Sigma^*$,

then (i) and (ii) are also equivalent to

(iii) $[L_1, L_2] \propto_s [L_1', L_2']$.

Proof:

(i) $\Rightarrow$ (ii): Suppose $L \in P$ and L separates $[L_1, L_2]$. Let f_1 and f_2 be polynomial embeddings of L_1 into L_1' and L_2 into L_2' respectively. Define $g : \Sigma^* \to \Sigma^*$ by

$$g(x) = \begin{cases} f_1(x), & x \in L, \\ f_2(x), & x \notin L . \end{cases}$$

Then $x \in L_1 \Rightarrow g(x) = f_1(x) \in L_1'$ and similarly $x \in L_2 \Rightarrow g(x) \in L_2'$. Clearly g is polynomial time computable, so g is a weak p-transformation from $[L_1, L_2]$ to $[L_1', L_2']$.

(ii) $\Rightarrow$ (i): Let $f : \Sigma^* \to \Sigma^*$ be a weak p-transformation from $[L_1, L_2]$ to $[L_1', L_2']$. Suppose $L' \in P$ and L' separates $[L_1', L_2']$. Define

$$L = \{x \in \Sigma^* \mid f(x) \in L'\}.$$

Then $L \in P$, since given $x \in \Sigma^*$ we can compute $f(x)$ and test whether $f(x) \in L'$ in polynomial time. Now

$$x \in L_1 \Rightarrow f(x) \in L_1' \subseteq L' \Rightarrow x \in L,$$

so $L_1 \subseteq L$, and similarly $L_2 \subseteq L^c$. Hence $[L_1, L_2]$ is p-separable.

(iii) $\Rightarrow$ (ii): Clear.

(i) $\Rightarrow$ (iii) (assuming $L_1 \propto L_1'$, $L_2 \propto L_2'$, and (A)):

Suppose $L \in P$ and L separates $[L_1, L_2]$. Let f_1, f_2 be polynomial transformations from L_1 to L_1' and from L_2 to L_2' respectively, and suppose $L' \in P$ and L' separates $[L_1', L_2']$.

Define $g : \Sigma^* \to \Sigma^*$ as follows, where c is a fixed member of $\Sigma^* \setminus (L_1' \cup L_2')$ if the latter is nonempty (which is certainly the case, by (A), if $L_1 \cup L_2 \neq \Sigma^*$):

$$g(x) = \begin{cases} f_1(x), & x \in L,\ f_1(x) \in L' ; \\ f_2(x), & x \notin L,\ f_2(x) \notin L' ; \\ c, & \text{otherwise.} \end{cases}$$

If $x \in L$ and $f_1(x) \in L'$, then (by definition of f_1) $x \in L_1$ iff $g(x) = f_1(x) \in L_1'$. Similarly if $x \notin L$ and $f_2(x) \notin L'$ then $x \in L_2$ iff $g(x) \in L_2'$.

In the final case of the definition of g, $x \notin L_1 \cup L_2$. Hence (by (A)) c exists and $g(x) = c \notin L_1' \cup L_2'$.

It is clear that g is polynomial time computable, so g is a strong p-transformation from $[L_1, L_2]$ to $[L_1', L_2']$. □

We now concentrate on the class of NP-co-NP pairs.

A further, very easy equivalent condition for p-separability can be given for NP-co-NP pairs.

Lemma 7: An NP-co-NP pair $[L_1, L_2]$ is polynomially separable iff there exist languages $L \in NP$, $Q \in P$ such that

$$L \cap Q = L_1 \quad \text{and} \quad L^c \cap Q^c = L_2 .$$

Proof: Suppose $Q \in P$ separates $[L_1, L_2]$. Set $L = (Q^c \setminus L_2) \cup L_1$. Then $L \cap Q = L_1$ and $L^c \cap Q^c = L_2$,

as required. The reverse implication is even more trivial since Q separates $[L_1, L_2]$ already. □

For convenience, we denote the class of NP-co-NP pairs by NΠ. The class of p-separable NP-co-NP pairs will be called Π. Now the class of all pairs $[L_1, L_1^c]$ where $L_1 \in NP$ is essentially just NP, and in this sense NΠ properly contains NP. In the same way it can be said that Π properly contains P. It is easily seen that

$$N\Pi = \Pi \iff NP = P.$$

Completeness for NΠ is closely related to NP-completeness. A pair $[L_1, L_2]$ is said to be <u>NΠ-complete</u> if $[L_1, L_2] \in N\Pi$ and $[L_1', L_2'] \propto_w [L_1, L_2]$ for all $[L_1', L_2'] \in N\Pi$. It is natural to use weak p-transformability here (see Lemma 4).

That NΠ-complete pairs exist follows from the next remark.

<u>Lemma 8</u>: If L is NP-complete then $[L, L^c]$ is NΠ-complete.

<u>Proof</u>: Suppose L is NP-complete and that $[L_1, L_2] \in N\Pi$. Then $L_1 \in NP$ so there is a polynomial transformation from L_1 to L, say f. But then f is also a weak p-transformation from $[L_1, L_2]$ to $[L, L^c]$. □

To prove that a pair $[L_1, L_2]$ is NΠ-complete, it suffices to show that $[L_1, L_2] \in N\Pi$ and that $[L_1', L_2'] \propto_w [L_1, L_2]$ for some NΠ-complete pair $[L_1', L_2']$. The similarity with NP-completeness is obvious.

It is clear that if a language pair is NΠ-complete then it is NP-hard, and so is not p-separable unless P = NP.

In the next section we will apply some of the concepts introduced in this section, particularly weak p-transformability and NΠ-completeness, to the specific problems that originally motivated this work.

§3. Separability and graph colouring

The problem mentioned at the beginning of §2, of finding a 4-colouring of a 3-colourable graph, has the following obvious generalization. For any fixed integers $k_1 > 2$ and $k_2 > 3$, define

$COL(k_1,k_2)$
Input: Graph G.
Promise: G is k_1-colourable.
Output: A $(k_2 - 1)$-colouring of G.

It is convenient to speak of (k_2-1)-colourings here, rather than k_2-colourings, as comparison with $COL_0(k_1,k_2)$ (defined below) is then more direct.

We suppose $k_1 < k_2 - 1$, since otherwise $COL(k_1,k_2)$ is easily seen to be just the problem of finding a (k_2-1)-colouring, if one exists, in a graph.

We conjecture that $COL(k_1,k_2)$ is NP-hard also for all k_1 and k_2 with $2 < k_1 < k_2 - 1$, but have not been able to prove this in general.

Closely related to $COL(k_1,k_2)$ is the following promise problem.

$COL_0(k_1,k_2)$
Input: Graph G.
Promise: $\chi(G) \leq k_1$ or $\chi(G) \geq k_2$.
Question: Is G k_1-colourable?

If M is a DTM which solves $COL(k_1,k_2)$, then the language L_M, defined by

$$L_M = \{G \mid M \text{ finds a } (k_2-1)\text{-colouring of } G\},$$

is a solution to $COL_0(k_1,k_2)$. Hence, if $COL(k_1,k_2)$ is polynomial time solvable then $COL_0(k_1,k_2)$ has a solution in P. I have been unable to determine whether or not the converse holds.

As stated earlier (in §2), solutions to promise problems are precisely recursive languages separating appropriate language pairs. In order to translate the above promise problem into a language pair, define, for all $k \in \mathbb{N}$,

$$\chi[\leq k] = \{G | \chi(G) \leq k\},$$

$$\chi[\geq k] = \{G | \chi(G) \geq k\}.$$

Then the recursive languages separating $[\chi[\leq k_1], \chi[\geq k_2]]$ are precisely the solutions to $COL_0(k_1,k_2)$, and we are asking if such language pairs are p-separable.

I believe but cannot prove:

Conjecture 1: $[\chi[\leq k_1], \chi[\geq k_2]]$ is NP-hard for all k_1 and k_2 such that $2 < k_1 < k_2 - 1$.

As we show below, this Conjecture is closely related to the following Conjecture of Garey and Johnson.

Conjecture 2 [17]: If $P \neq NP$ then for every rational k there is no polynomial time algorithm which determines the chromatic number of all graphs to within a factor of k. (That is, there is no polynomial time computable function f such that $\chi(G) \leqslant f(G) \leqslant k.\chi(G)$ for all graphs G.)

Theorem 1: If Conjecture 1 is true then Conjecture 2 is true.

Proof: Suppose Conjecture 2 is false. Let $k \in \mathbb{Q}$ be fixed and let $f \in FP$ be such that $f(G) \leq k.\chi(G)$ for all G. Define

$$L_f = \{G \mid f(G) \leq 3k\}.$$

Then $L_f \in P$ since $f \in FP$, and L_f separates $[\chi[\leq 3], \chi[\geq 3k+1]]$. If $P \neq NP$, this contradicts Conjecture 1. □

The converse of Theorem 1 does not seem to follow easily and it may well be untrue. It appears conceivable that $[\chi[\leq k_1], \chi[\geq k_2]]$ may be p-separable for certain fixed k_1 and k_2 even if Conjecture 2 is true.

Although Conjecture 2 is open for general k, it has been shown in [17] to be true for $k < 2$. We can adapt the proof technique of this result (see also [18, pp. 142-145]) to obtain NΠ-completeness results for $[\chi[\leq k_1], \chi[\geq k_2]]$ for certain k_1, k_2.

Theorem 2: Suppose k_1 and k_2 are integers satisfying one of the following:

$$\text{(i)} \quad 7 \leq k_1 + 1 < k_2 \leq \tfrac{4}{3} k_1 \ ;$$

$$\text{(ii)} \quad 8 \leq k_1 + 1 < k_2 \leq 2k_1 - 4 \ .$$

Then $[\chi[\leq k_1], \chi[\geq k_2]]$ is NΠ-complete, and hence $COL_0(k_1,k_2)$ and $COL(k_1,k_2)$ are NP-hard.

Proof: First suppose k_1 and k_2 satisfy (i). Put $k_1 = 3q + r$ where $q \geq 2$ and $0 \leq r \leq 2$. Let G be any graph. Form

$$G' = K_q[G] + K_r \ .$$

Then

$$\chi(G') = \chi(K_q[G]) + \chi(K_r)$$

$$= \chi_{\chi(G)}(K_q) + r \qquad \text{(Lemma 4.1.2)}$$

$$= q.\chi(G) + r.$$

Thus if $\chi(G) \leq 3$ then $\chi(G') \leq k_1$, and if $\chi(G) \geq 4$ then

$$\chi(G') \geq 4q + r = \frac{4}{3} k_1 - \frac{r}{3}$$

$$\geq k_2 - \frac{r}{3} .$$

Hence $\chi(G') \geq k_2$ as k_2 is an integer. It follows that

$$[\chi[\leq 3], \chi[\geq 4]] \propto_s [\chi[\leq k_1], \chi[\geq k_2]]$$

and so $[\chi[\leq k_1], \chi[\geq k_2]]$ is NΠ-complete. (Membership of NΠ is obvious, and $[\chi[\leq 3], \chi[\geq 4]]$ is NΠ-complete by Lemma 2.8.)

Secondly, suppose k_1 and k_2 satisfy (ii). Let G be any graph. Form the composition $K_3^{k_1}[G]$ where $K_3^{k_1}$ is a Kneser graph (see §4.3). Then

$$\chi(K_3^{k_1}[G]) = \chi_{\chi(G)}(K_3^{k_1}) \qquad \text{(Lemma 4.1.2).}$$

Thus if $\chi(G) \leq 3$ then

$$\chi(K_3^{k_1}[G]) \leq \chi_3(K_3^{k_1}) = k_1 \qquad \text{as shown in [17],}$$

and if $\chi(G) \geq 4$ then

$$\chi(K_3^{k_1}[G]) \geq \chi_4(K_3^{k_1}) = 2k_1 - 4 \qquad \text{as shown in [17]}$$

$$\geq k_2 .$$

It follows that $[\chi[\leq k_1], \chi[\geq k_2]]$ is NΠ-complete. □

Of course if k_1 and k_2 are sufficiently large then (ii) $\Rightarrow$ (i). It would be of interest to prove an NP-hardness result for $[\chi[\leq k_1], \chi[\geq k_2]]$ for some k_1, k_2 satisfying $2 < k_1 \leq \frac{k_2}{2}$. It would also be of some interest to determine the complexity of separating $[\chi[\leq k_1], \chi[\geq k_2]]$ for certain small values of k_1 and k_2 not covered by (i) or (ii) above, for example $(k_1,k_2) = (3,5)$ (our original problem), $(4,6)$, $(4,7)$, $(5,7)$ and others.

If we allow k_1 and k_2 to be functions of the number n of vertices of the graph, then if $k_2 - k_1$ is sufficiently large we can obtain a result of p-separability as a corollary of Johnson's approximate colouring algorithm [28].

Theorem 3: $[\chi[\leq 3], \chi[\geq \frac{9n}{\log n} + 1]]$ is p-separable.

Proof: This follows directly from the polynomial time algorithm of Johnson [28] which determines the chromatic number of a graph on n vertices to within a factor of $\frac{3n}{\log n}$ for all $n > N$, where N is a fixed constant. One language in P which separates the above pair is $L_1 \cup L_2$ where

$$L_1 = \chi[\leq 3] \cap \{G \mid |V(G)| \leq N\}$$

and L_2 is the set of all graphs for which Johnson's algorithm outputs a value less than or equal to $\frac{9n}{\log n}$. □

There is thus a large gap between the functions $k_2 - k_1$ (when considered as a function of n) for which $[\chi[\leq k_1], \chi[\geq k_2]]$ is known to be NP-complete (small constant functions - see Theorem 2) and those functions $k_2 - k_1$ for which it is known to be in P (Theorem 3). One may wonder what happens for functions $k_2 - k_1$ which are intermediate, between these two extremes. This question is reminiscent

of some of the questions we asked in Chapter 3 and, as with the questions raised there, may be very difficult to resolve conclusively.

Many other examples of p-separable NP-co-NP pairs may be constructed using Lemma 2.7. As we have seen, there are examples of NP-co-NP pairs which are nontrivially NΠ-complete. It is natural to ask how, in general, an arbitrary NP-co-NP pair can be most efficiently separated. Formally:

Question: For any NP-co-NP pair $[L_1, L_2]$, what is the minimal (with respect to $\propto$) language in NP which separates $[L_1, L_2]$?

It is not clear that the answer to this question must be unique (up to polynomial time equivalence), although I have not been able to find a counterexample. Thus we ask:

Question: Does there exist, for each NP-co-NP pair $[L_1, L_2]$, a unique (up to polynomial time equivalence) $\propto$-minimal member of

$$\{L \in NP \mid L \text{ separates } [L_1, L_2]\}?$$

An affirmative answer would tell us that in a sense NΠ really is nothing more than just NP; a negative answer would indicate that the structure of NΠ is significantly more complex than that of NP. The above questions appear to be quite deep and I have made no progress in resolving them.

CHAPTER 6

THE COMPLEXITY OF A GRAPH PARTITIONING PROBLEM

§1. Introduction

In this Chapter the theory of computational complexity is applied to a problem which was put to us by F. Albrow and B.D. Bramson of the Royal Signals Research Establishment, Malvern, and which has also been studied in [6, 69].

Let G be a graph and suppose F is a set of unordered pairs of vertices of G. (We call the members of F forbidden pairs. Clearly F can be regarded as a graph, and we will frequently do so.) An F-partition of G is a partition of $V(G)$ such that for every pair $\{u,v\} \in F$ the vertices u and v are in different parts of the partition. An edge $e \in E(G)$ is a strong edge for an F-partition Π of G if the endpoints of e are in different parts of Π. If w is a positive weight function on $E(G)$ then the weight of the F-partition Π is the sum of the weights of the strong edges of Π.

The problem is the following.

MINIMUM WEIGHT GRAPH PARTITION (abbreviated MWGP)

Input: Triple (G,w,F) where G is a graph, $w : E(G) \to \mathbb{N}$ is a weight function and F is a set of unordered pairs of vertices of G; integer K.

Question: Is there an F-partition of G with weight (under w) less than or equal to K?

The vertices of G may represent the divisions of some organisation, and we may think of the parts of a partition of

V(G) as being locations among which the divisions are to be distributed. (The exact nature and position of locations is irrelevant for all our purposes, and we assume there is an infinite number of available locations, so the number of parts of the partition is not limited.) There are restrictions, given by F: certain divisions may be forbidden from being at the same location (for example, it may be desired that divisions performing similar functions be at different locations, to reduce vulnerability). The edges of G indicate which divisions must communicate. The weights give communication costs, which only apply if the divisions concerned are at different locations. The problem, then, is to arrange the divisions among the locations, obeying the restrictions given by F, such that the total communication cost is minimized.

If G and F are as above, then an <u>F-cut</u> of G is a set of edges $E' \subseteq E(G)$ such that for every $\{u,v\} \in F$, every $u - v$ path in G contains an edge in E'. If w is a weight function then the <u>weight</u> of an F-cut E' is the sum of the weights of its edges. In [69], MWGP is defined as follows:

MWGP (equivalent formulation)
Input: As above.
Question: Is there an F-cut of G with weight (under w) less than or equal to K?

The equivalence of the two formulations of MWGP can be seen from the following elementary fact.

Lemma 1: Let the triple (G,w,F) be as in the input to MWGP. If $S \subseteq E(G)$ then S is the set of strong edges of a minimum weight F-partition of G iff S is a minimum weight F-cut of G. □

The main results of [69] for MWGP concern the directed version, which we do not consider. For the undirected problem they make the following observations.

If F is a fixed graph, let MWGP(F) be the restriction of MWGP to instances in which the forbidden pair graph F is isomorphic to F. Denote by mK_2 the graph consisting of m disjoint copies of K_2.

Lemma 2 [69]: If $|E(F)| = m$ then MWGP(F) $\propto$ MWGP (mK_2). □

Lemma 3 [69]: If F is a complete bipartite graph, or $F \cong 2K_2$, then MWGP(F) is solvable in polynomial time by network flow methods. □

Lemma 4 [69]: For all F, the restriction of MWGP(F) to instances in which G is a tree is in P. □

The following results are due to Dahlhaus et al. [6].

Theorem 5 [6]: The following restrictions of MWGP are both NP-complete:

(a) $F \cong K_3$, all edge weights are 1;

(b) F is a complete graph and G is planar with maximum degree 4. □

Theorem 6 [6]: For each fixed k, the restriction of MWGP(K_k) to instances in which G is planar is in P. □

Dahlhaus has also obtained the following unpublished result.

Theorem 7 (Dahlhaus)

The following restrictions of MWGP are both NP-complete:

(a) G is a tree, F is a complete graph and all edge weights are 1;

(b) G is a binary tree, all edge weights 1.

In the next section we obtain more NP-completeness results for various restrictions of MWGP. We consider mainly restrictions on G, in contrast to the foregoing results which are principally concerned with restrictions on F.

§2. NP-completeness results

We consider here the complexity of a number of subproblems of MWGP. These subproblems mostly involve structural constraints on G rather than F.

The first two results give the most severe restrictions yet on G under which the problem is known to be NP-complete.

A star graph is a graph which is isomorphic to $K_{1,k}$ for some k.

Theorem 1: MWGP is NP-complete when restricted to triples (G,w,F) such that G is a star graph and all edge weights are 1.

Proof: We reduce VERTEX COVER to this restriction of MWGP.

Let (G,k) be input to VERTEX COVER [18, p.46].

Suppose $V(G) = \{v_1,\ldots,v_n\}$.

Define the graph H as follows.

$$V(H) = V(G) \cup \{v_0\},$$

$$E(H) = \{v_0v \mid v \in V(G)\}.$$

Thus H is a star (see Figure 1).

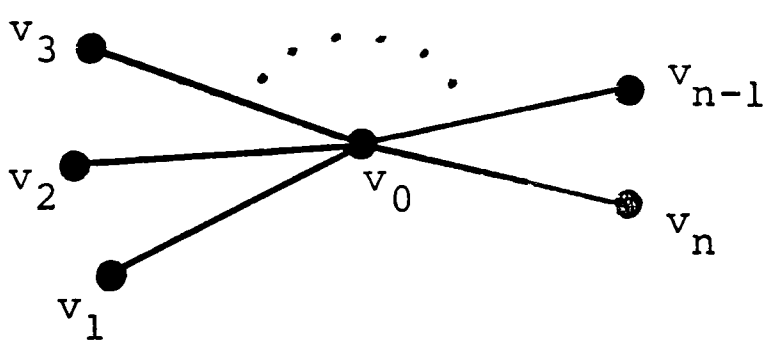


Figure 1: The graph H.

Define the weight function w by

$$w(e) = 1 \quad \text{for all } e \in E(H),$$

and define a set $\mathcal{F}$ of forbidden pairs of vertices of H by

$$\mathcal{F} = E(G).$$

Thus, as a graph, $\mathcal{F}$ is isomorphic to the graph obtained from G by removing isolated vertices.

Now,

H has an $\mathcal{F}$-partition of weight (under w) $\leq k$

$\Leftrightarrow$ H has a set S of $\leq k$ edges such that, for all $\{v_i,v_j\} \in \mathcal{F}$, the (necessarily unique) $v_i - v_j$ path in G contains an edge of S (Lemma 1.1)

$\Leftrightarrow$ H has a set S of $\leq k$ edges such that, for all $\{v_i,v_j\} \in E(G)$, at least one of v_0v_i, v_0v_j is in S

$\Leftrightarrow$ There is a set $S \subseteq \{v_0v_1,v_0v_2,\ldots,v_0v_n\}$, $|S| \leq k$, such that for all $\{v_i,v_j\} \in E(G)$, at least one of v_0v_i, v_0v_j is in S

$\Leftrightarrow$ G has a vertex cover of size $\leq k$. □

Theorem 2: MWGP is NP-complete when restricted to triples $(G,w,\mathcal{F})$ satisfying the following:

(i) G is a tree whose longest path has length 4;

(ii) $w(e) = 1 \quad \forall\, e \in E(G)$;

(iii) All pairs of $\mathcal{F}$ are disjoint.

Proof: We reduce VERTEX COVER to this problem.

Let (G,k) be input to VERTEX COVER and suppose $V(G) = \{v_1,\ldots,v_n\}$.

Form the graph H' as follows.

$$V(H') = \{v_0\} \cup V(G) \cup \{(v_i,v_j) | v_iv_j \in E(G)\}$$

$$E(H') = \{v_0v_i | v_i \in V(G)\}$$

$$\cup \{\{v_i,(v_i,v_j)\} | v_i \in V(G), v_iv_j \in E(G)\}.$$

Define the weight function w' by

$$w'(e) = 1 \quad \text{for all} \quad e \in E(H),$$

and the forbidden pair set F' by

$$F' = \{(v_i,v_j),(v_j,v_i) | v_iv_j \in E(G)\}.$$

This construction of (H',w',F') from G is directly related to the construction of (H,w,F) from G in the proof of the previous Theorem. We illustrate this by an example in Figure 2 (see also Lemma 1.2). The forbidden pairs are indicated by dotted lines; all edges of H and H' have weight 1.

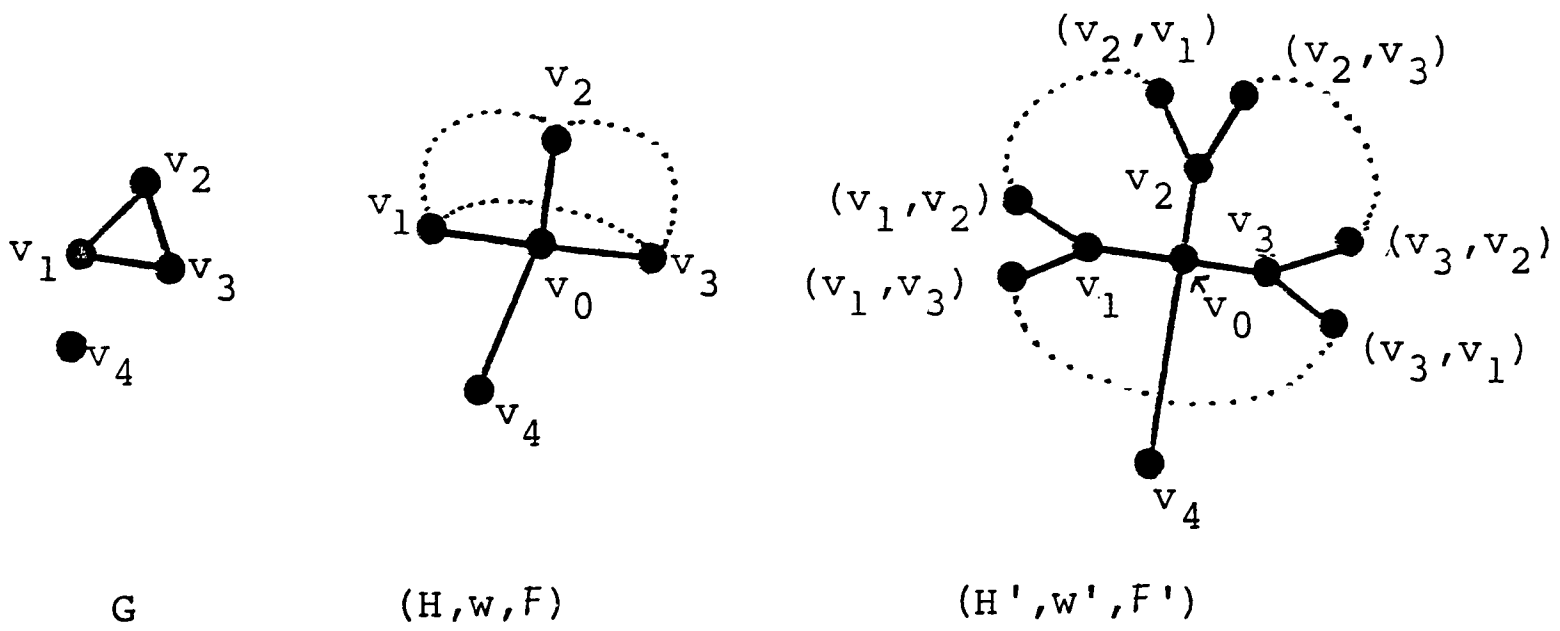


Figure 2

It is now straightforward to verify (using a slightly more detailed version of the argument used at the end of the proof of the previous Theorem) that our map $G \mapsto (H',w',F')$ is the required polynomial transformation. □

We now mention a result of K.J. Edwards which was obtained in a different way to those above.

Theorem 3 (Edwards): The restriction of MWGP to instances in which $G \cong K_{3,n-3}$ and all edge weights are 1 is NP-complete.

Proof: Reduction from 3-COLOURABILITY. □

Recalling the practical motivation for MWGP as discussed in §1, it is of interest to consider the case when all divisions must communicate and all communication costs are the same (the "uniform cost" case). Thus we restrict the problem to instances in which $G \cong K_n$ and all edge weights are (w.ℓ.o.g.) 1. A minimum weight F-partition of G is then just a colouring (in the usual sense) of F with minimum cost, where the cost of a colouring of F is the number of differently coloured vertex pairs of F. (A vertex pair is differently coloured if its two members receive different colours.)

We write H instead of F for the time being, and put $n = |V(H)|$.

Suppose all our colours are members of $\mathbb{N}$. If c is a colouring of H then we denote the number of vertices of H which receive colour i under c by $n_i(c)$. Where no ambiguity arises we may put $n_i = n_i(c)$. Clearly

$$\sum_i n_i(c) = n.$$

Our problem is to find a colouring c of H which minimizes

$$\text{cost}(c) = \sum_{i<j} n_i(c).n_j(c) .$$

This is achieved by precisely those colourings c of H which maximize

$$\sigma(c) = \sum_i n_i(c)^2 ,$$

since

$$2.\text{cost}(c) + \sigma(c) = n^2 .$$

Let us then rewrite the problem, and call it p_2. The reason for the subscript 2 will be explained later.

p_2

Input: Graph H, integer M.

Question: Is there a colouring c of H such that

$$\sigma(c) = \sum_i n_i(c)^2 \geq M \ ?$$

Remarks:

(1) A p_2-optimal colouring of H need not necessarily be a $\chi(H)$-colouring.

For example, let H be as shown in Figure 3, with the colouring indicated.

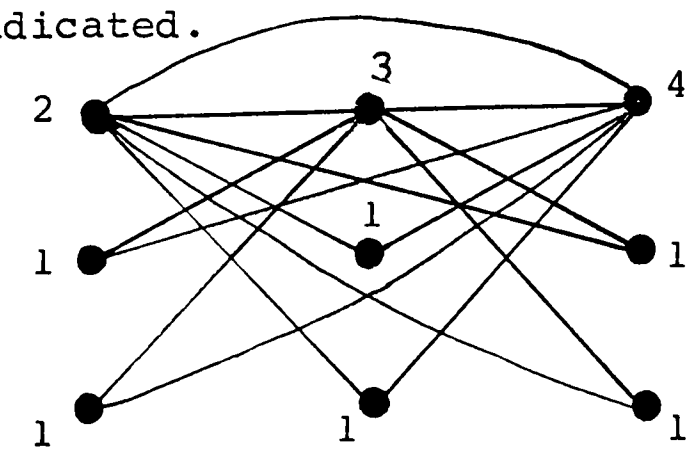


Figure 3

Here, $\chi(H) = 3$. With the essentially unique 3-colouring, $\sigma = 27$. However, the 4-colouring indicated gives $\sigma = 39$.

It appears that colourings with large colour classes tend to be better than colourings with a minimum number of colours. However this is not always the case as we now show.

(2) Colourings with a maximum independent set as one of their colour classes also need not be p_2-optimal.

For example, let H be the graph in Figure 4. Here V_i is independent for all $i = 1,\ldots,4$, and $|V_1| = |V_4| = 5$, $|V_2| = |V_3| = 3$. The edges are all possible edges joining a vertex of V_i to a vertex of V_{i+1}, for each $i = 1,2,3$.

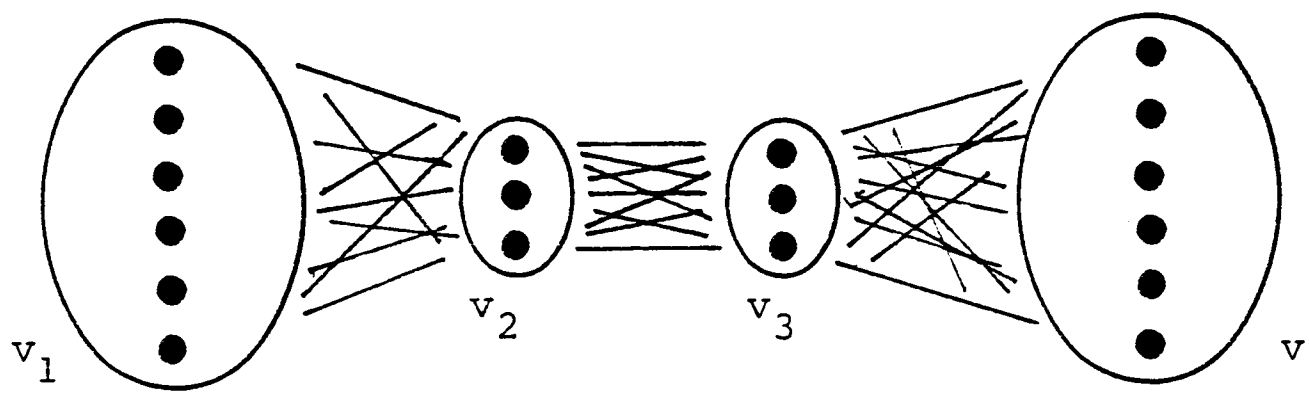


Figure 4

With the sole maximum independent set $V_1 \cup V_4$ as a colour class we can attain $\sigma = 118$. However the colouring with classes $V_1 \cup V_3$, $V_2 \cup V_4$ gives $\sigma = 128$.

(3) The graph H of Figure 5 illustrates both the above remarks simultaneously.

The numbers on the vertices give a 3-colouring, so $\chi(H) = 3$. The best possible σ with a 3-colouring is 114.

The squared vertices are the sole maximum independent set of H. The best σ obtainable by a colouring with this set as one colour class is 124.

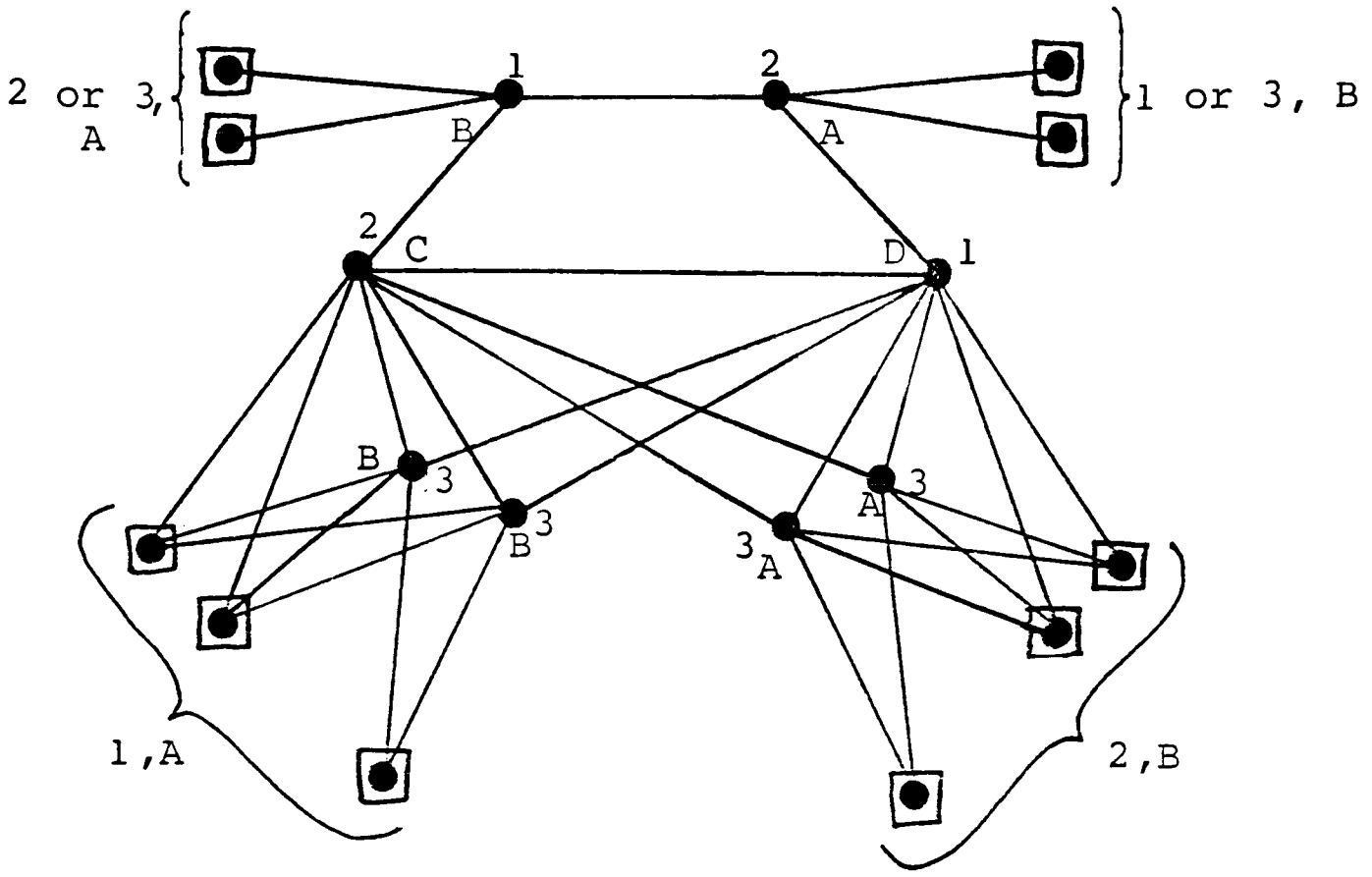


Figure 5

However the maximum σ is attained by the colouring indicated by the letters A, B, C, D and has value 130.

p_2 is nevertheless related to the maximum independent set problem, as the proof of the following result will reveal.

Theorem 4: p_2 is NP-complete.

Proof: Membership of NP is clear.

We show

INDEPENDENT SET $\propto p_2$.

Let (G,k) be input to INDEPENDENT SET. Put $n = |V(G)|$. Take $N = \left\lceil \frac{n^2+1}{2} \right\rceil$ and put $M = (k + N)^2$. Add N isolated vertices to G, forming the graph $H = G \cup \bar{K}_N$.

Consider any colouring c of H, and denote by c_0 the restriction of c to G. Suppose w.ℓ.o.g. that the colours used are $\{1,\ldots,r\}$ and that

$$n_1(c_0) \geq n_2(c_0) \geq \ldots \geq n_r(c_0).$$

Since we are concerned with maximizing $\sigma(c)$, we may restrict attention to colourings c where all vertices in the copy of $\overline{K}_N$ receive colour 1. Thus $n_i(c) = n_i(c_0) = n_i$ (say) for $i \geq 2$, and $n_1(c) = n_1(c_0) + N$, so

$$\sigma(c) = (n_1(c_0) + N)^2 + \sum_{i=2}^{r} n_i^2 .$$

We show $\sigma(c) < M$ iff $k > n_1(c_0)$.

$(\Rightarrow)$ $\sigma(c) < M \Leftrightarrow \sigma < (k + N)^2$

$$\Leftrightarrow \sum_{i=2}^{r} n_i^2 < (k + N)^2 - (n_1(c_0) + N)^2$$

$$\Rightarrow 0 < (k - n_1(c_0))(k + n_1(c_0) + 2N)$$

$$\Rightarrow k > n_1(c_0) .$$

$(\Leftarrow)$ $k > n_1(c_0) \Rightarrow (k + N)^2 - (n_1(c_0) + N)^2$

$$\geq (k + N)^2 - (k - 1 + N)^2$$

$$= 2k + 2N - 1$$

$$\geq n^2 \quad \text{by definition of } N$$

$$> \sum_{i=2}^{r} n_i^2$$

$$\Leftrightarrow \sigma(c) < M.$$

Now:

$H \notin p_2 \Rightarrow$ for every colouring c of H, $\sigma(c) < M$

$\Rightarrow$ in particular, for every colouring c of H in which the first colour class is a maximum independent set (so $n_1(c_0) = \alpha(G)$), we have $\sigma(c) < M$

$\Rightarrow$ for every such colouring c, $k > n_1(c_0)$ (by the above result)

$\Rightarrow$ G has no independent set of size k.

Conversely,

G has no independent set of size k

$\Rightarrow k > n_1(c_0)$ for every colouring c of H (since $n_1(c_0) \leq \alpha(G) < k$)

$\Rightarrow \sigma(c) < M$ for every colouring c of H

$\Rightarrow H \notin p_2$.

Thus,

$(G,k) \in$ INDEPENDENT SET $\Leftrightarrow (H,M) \in p_2$.

The map $(G,k) \mapsto (H,M)$ is easily seen to be polynomial time computable, and the result follows. □

We briefly comment on our name for p_2 and thereby on some kindred problems. In p_2 we ask for the colouring c which minimizes the Euclidean norm $\|\cdot\|_2$ of the vector $(n_i(c))_{i \geq 1}$. We could instead use some other norm $\|\cdot\|_\alpha$ and thus define a new problem, p_α. In particular,

$$\|(n_i)_{i\geq 1}\|_\infty = \max_{i\geq 1} (n_i) \ ,$$

so p_∞ is just INDEPENDENT SET. Using techniques very similar to those of the proof of Theorem 4, it can be shown that p_α is NP-complete for all $\alpha > 1$. The problem p_1 is trivial, since then the value of $\|(n_i)_{i\geq 1}\|_1$ is always just n.

Having obtained a number of NP-completeness results for MWGP, we close by briefly mentioning a subproblem that can be solved in polynomial time.

The restriction of MWGP to instances in which G is a path can be simply transformed into the weighted dominating set problem for strongly chordal graphs (for this definition see [13, 14]). This problem has a polynomial time algorithm [14], with the following consequence. (This result has been proved independently by K.J. Edwards.)

Theorem 5: The restriction of MWGP to instances in which G is a path can be solved in polynomial time. □

REFERENCES

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA, 1974.

[2] B. Bollobás, Uniquely colourable graphs, J. Combinatorial Theory (Ser. B) 25 (1978), 54-61.

[3] F.H. Clarke and R.E. Jamison, Multicolourings, measures and games on graphs, Discrete Math. 14 (1976), 241-245.

[4] S.A. Cook, The complexity of theorem proving procedures, Proc. 3rd Ann. ACM Symp. on Theory of Comput., Assoc. Comput. Mach., New York, 1971, pp. 151-158.

[5] S.A. Cook, Characterizations of pushdown machines in terms of time bounded computers, J. Assoc. Comput. Mach. 18 (1971), 4-18.

[6] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour and M. Yannakakis, Unpublished work (1985).

[7] M. Davis and H. Putnam, A computing procedure for quantificational theory, J. Assoc. Comput. Mach. 7 (1960), 201-215.

[8] G.A. Dirac, A property of 4-chromatic graphs and remarks on critical graphs, J. London Math. Soc. 27 (1952), 69-81.

[9] R.J. Duffin, Topology of series-parallel networks, J. Math. Anal. Appl. 10 (1965), 303-318.

[10] K.J. Edwards, The complexity of colouring problems on dense graphs, Theoret. Comput. Sci., to appear.

[11] S. Even, A.L. Selman and Y. Yacobi, The complexity of promise problems with applications to public-key cryptography, Inform. and Control 61 (1984), 159-173.

[12] S. Even and Y. Yacobi, Cryptocomplexity and NP-completeness, in: Automata, Languages and Programming, Seventh Colloquium (ICALP), Lecture Notes in Computer Science 85, Springer, Berlin, 1980, pp. 195-207.

[13] M. Farber, Characterizations of strongly chordal graphs, Discrete Math. 43 (1983), 173-189.

[14] M. Farber, Domination, independent domination and duality in strongly chordal graphs, Discrete Appl. Math. 7 (1984), 115-130.

[15] G. Farr and C. McDiarmid, The complexity of counting homeomorphs, Theoret. Comput. Sci. 36 (1985), 345-348.

[16] J. Fortune, J.E. Hopcroft and J. Wyllie, The directed subgraph homeomorphism problem, Theoret. Comput. Sci. 10 (1980), 111-121.

[17] M.R. Garey and D.S. Johnson, The complexity of near-optimal graph colouring, J. Assoc. Comput. Mach. 23 (1976), 43-49.

[18] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Co., San Francisco, 1979.

[19] M.R. Garey, D.S. Johnson and L. Stockmeyer, Some simplified NP-complete graph problems, Theoret. Comput. Sci. 1 (1976), 237-267.

[20] M.R. Garey, D.S. Johnson and R.E. Tarjan, The planar Hamiltonian circuit problem is NP-complete, SIAM J. Comput. 5 (1976), 704-714.

[21] D. Geller and S. Stahl, The chromatic number and other functions of the lexicographic product, J. Combinatorial Theory (Ser. B) 19 (1975), 87-95.

[22] R.L. Graham, B.L. Rothschild and J.H. Spencer, Ramsey Theory, Wiley, New York, 1980.

[23] J. Grollmann and A.L. Selman, Complexity measures for public-key cryptosystems, 25th Ann. Symp. on Foundations of Comput. Sci., IEEE Computer Society Press, Washington, 1984, pp. 495-503.

[24] D.W. Hall, A note on primitive skew curves, Bull. Amer. Math. Soc. 49 (1943), 935-937.

[25] F. Harary, Graph Theory, Addison-Wesley, Reading, MA, 1969.

[26] A.J.W. Hilton, R. Rado and S.H. Scott, A(<5)-colour theorem for planar graphs, Bull. London Math. Soc. 5 (1973), 302-306.

[27] R.W. Irving, NP-completeness of a family of graph-colouring problems, Discrete Appl. Math. 5 (1983), 111-117.

[28] D.S. Johnson, Worst case behaviour of graph colouring algorithms, Proc. 5th South-East Conf. on Combinatorics, Graph Theory and Comput., Congress. Numer. X, Utilitas Mathematica, Winnipeg, 1974, pp. 513-527.

[29] N.D. Jones, Space-bounded reducibility among combinatorial problems, J. Comput. System Sci. 11 (1975), 68-85.

[30] N.D. Jones and W.T. Laaser, Complete problems for deterministic polynomial time, Theoret. Comput. Sci. 3 (1977), 105-117.

[31] R.M. Karp, On the complexity of combinatorial problems, Networks 5 (1975), 45-68.

[32] C.M.R. Kintala and P.C. Fischer, Computations with a restricted number of nondeterministic steps, Proc. 9th Ann. Symp. on Theory of Comput., Assoc. for Comput. Mach., New York, 1977, pp. 178-185.

[33] C.M.R. Kintala and P.C. Fischer, Refining nondeterminism in relativized polynomial-time bounded computations, SIAM J. Comput. 9 (1980), 46-53.

[34] M. Kneser, Aufgabe 300, Jahresber. Deutsch. Math.-Verein. 58 (1955).

[35] K. Kuratowski, Sur le problème des courbes gauches en topologie, Fund. Math. 15 (1930), 271-283.

[36] R.E. Ladner, On the structure of polynomial time reducibility, J. Assoc. Comput. Mach. 22 (1975), 155-171.

[37] A.S. LaPaugh and R.L. Rivest, The subgraph homeomorphism problem, Proc. 10th Ann. ACM Symp. on Theory of Comput., Assoc. for Comput. Mach., New York, 1978, pp. 40-50.

[38] L. Lovász, Kneser's conjecture, chromatic number and homotopy, J. Combinatorial Theory (Ser. A) 25 (1978), 319-324.

[39] H.A. Maurer, J.H. Sudborough and E. Welzl, On the complexity of the general colouring problem, Inform. and Control 51 (1981), 128-145.

[40] K. Menger, Zur allgemeinen Kurventheorie, Fund. Math. 10 (1927), 96-115.

[41] B. Monien, How to find long paths efficiently, Ann. Discrete Math. 25 (1985), 239-254.

[42] C.H. Papadimitriou and K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[43] C. Rackoff, Relativized questions involving probabilistic algorithms, J. Assoc. Comput. Mach. 29 (1982), 261-268.

[44] N. Robertson and P.D. Seymour, Graph minors. I. Excluding a forest, J. Combinatorial Theory (Ser. B) 35 (1983), 39-61.

[45] N. Robertson and P.D. Seymour, Graph minors. II. Algorithmic aspects of tree-width, J. Algorithms, to appear.

[46] N. Robertson and P.D. Seymour, Graph minors. III. Planar tree-width, J. Combinatorial Theory (Ser. B) 36 (1984), 49-64.

[47] N. Robertson and P.D. Seymour, Graph minors - a survey, in: Ian Anderson (ed.), Surveys in Combinatorics 1985, London Mathematical Society Lecture Note Series 103, Cambridge University Press, Cambridge, 1985, pp. 153-171.

[48] N. Robertson and P.D. Seymour, Graph minors. IV. Tree-width and well-quasi-ordering, to appear.

[49] N. Robertson and P.D. Seymour, Graph minors. V. Excluding a planar graph, J. Combinatorial Theory (Ser. B), to appear.

[50] N. Robertson and P.D. Seymour, Graph minors. VI. Disjoint paths across a disc., J. Combinatorial Theory (Ser. B), to appear.

[51] N. Robertson and P.D. Seymour, Graph minors. VII. Disjoint paths on a surface, J. Combinatorial Theory (Ser. B), to appear.

[52] N. Robertson and P.D. Seymour, Graph minors. VIII. A Kuratowski theorem for general surfaces, submitted.

[53] N. Robertson and P.D. Seymour, Graph minors. IX. Disjoint crossed paths, submitted.

[54] N. Robertson and P.D. Seymour, Graph minors. X. Tilts, submitted.

[55] N. Robertson and P.D. Seymour, Graph minors. XI. Distance on a surface, submitted.

[56] N. Robertson and P.D. Seymour, Unpublished work.

[57] H. Rogers, Theory of Recursive Functions and Effective Computability, McGraw-Hill, New York, 1967.

[58] U. Schöning, A uniform approach to obtain diagonal sets in complexity classes, Theoret. Comput. Sci. 18 (1982), 95-103.

[59] A.L. Selman and Y. Yacobi, The complexity of promise problems, in: Automata, Languages and Programming, Ninth Colloquium (ICALP), Lecture Notes in Computer Science 140, Springer, Berlin, 1982, pp. 502-509.

[60] P.D. Seymour, Decomposition of regular matroids, J. Combinatorial Theory (Ser. B) 28 (1980), 305-359.

[61] P.D. Seymour, private communication (1985).

[62] P.D. Seymour and P.N. Walton, Detecting matroid minors, J. London Math. Soc. (2) 23 (1981), 193-203.

[63] S. Stahl, n-tuple colourings and associated graphs, J. Combinatorial Theory (Ser. B) 20 (1976), 185-203.

[64] L.G. Valiant, Relative complexity of checking and evaluating, Inform. Process. Lett. 5 (1976), 20-23.

[65] L.G. Valiant, The complexity of computing the permanent, Theoret. Comput. Sci. 8 (1979), 189-201.

[66] L.G. Valiant, The complexity of enumeration and reliability problems, SIAM J. Comput. 8 (1979), 410-421.

[67] K. Wagner, Über eine Eigenschaft der ebenen Komplexe, Math. Ann. 114 (1937), 570-590.

[68] D.J.A. Welsh and D.M. Petford, A randomised attack on an NP-complete problem, unpublished manuscript (1985).

[69] M. Yannakakis, P.C. Kanellakis, S.C. Cosmadakis and C.H. Papadimitriou, Cutting and partitioning a graph after a fixed pattern, in: Automata, Languages and Programming, Tenth Colloquium (ICALP), Lecture Notes in Computer Science 154, Springer, Berlin, 1983, pp. 712-722.