The NP-completeness of some edge-partitioning problems

by

Sebastian M. Cioabă

A thesis submitted to the Department of Mathematics and Statistics in conformity with the requirements for the degree of Master of Science

Queen's University

Kingston, Ontario, Canada

May, 2002

Contents

CHAPTER 1

Introduction

We use $|A|$ to denote the number of elements of the set A.

A *simple graph* G with n vertices and m edges consists of a *vertex set* $V(G) = \{v_1, v_2, \ldots, v_n\}$ and an *edge set* $E(G) = \{e_1, e_2, \ldots, e_m\}$, where each edge is an unordered pair of vertices. We write uv for the edge $\{u, v\}$. If $uv \in E(G)$, then u and v are adjacent. The vertices contained in an edge e are its *endpoints*.

We denote $\nu(G) = n = |V(G)|$ and $\epsilon(G) = m = |E(G)|$.

A subgraph of a graph G is a graph H such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$; we write this as $H \subseteq G$ and we say G contains H.

A *complete graph* or *clique* is a simple graph in which every pair of vertices forms an edge. We denote by K_n the complete graph with n vertices. The complete graph with 3 vertices will be called *triangle*. The *trianglegraph* of a graph G is the graph T whose vertices are the triangles of G, with two distinct vertices of T deemed adjacent if, as triangles in G they share a common edge. We denote T as $T(G)$. If G is triangle-free, then we define $T(G) = \emptyset$.

A *vertex cover* in a graph G is a set S of vertices such that S contains at least one endpoint of each edge of G. A vertex cover S of G is called *minimal* if for any vertex cover S' of G we have $|S| \leq |S'|$.

The cardinality of a minimal vertex cover of a graph G is denoted $\alpha(G)$.

An *independent set* in a graph G is a vertex subset $S \subseteq V(G)$ such that any two vertices in S are not adjacent. An independent set of vertices S in a graph G is called *maximal* if for any independent set of vertices S' of G we have $|S| \geq |S'|$.

The cardinality of a maximal independent set of a graph G is denoted by $\beta(G)$.

The *chromatic number* of a graph G, denoted by $\chi(G)$ is the minimum number of independent sets that partition the vertex set of G.

A graph is *bipartite* if its vertices can be partitioned into (at most) two independent sets.

A *complete bipartite graph* or a *biclique* is a bipartite graph in which the edge set consists of all pairs having a vertex from each of the two independent sets in the vertex partition.

A *clique covering of* G is a family $\mathcal{C}$ of cliques of G such that every edge of G is in at least one member of $\mathcal{C}$.

If the members of $\mathcal{C}$ are pairwise edge-disjoint then C is said to be a *clique partition* of G.

Similarly, we can define the *biclique covering* and *biclique partition* of a graph G.

A clique covering $\mathcal{C}$ is said to be *minimal* if $|\mathcal{C}'| \geq |\mathcal{C}|$ for all clique coverings $\mathcal{C}'$.

The *clique-covering number* of G, $cc(G)$ is the cardinality of a minimal clique covering.

The *clique-partition number* of G, $cp(G)$ is the cardinality of a minimal clique partition.

The *biclique-covering number* of G, $bc(G)$ is the cardinality of a minimal biclique covering.

The *biclique-partition number* of G, $bp(G)$ is the cardinality of a minimal biclique partition.

Fundamental questions posed by Boole in 1868 on the theory of sets have been translated to problems in graph theory. The motivation for defining $cp(G)$ and $cc(G)$ stem from the following observation in set theory.

Given a family of sets $S_1, S_2, \ldots, S_n$ it is possible to associate with it a multigraph G with vertices $x_1, x_2, \ldots, x_n$ such that the number of edges between x_i and x_j is equal to $|S_i \cap S_j|$.

Conversely, [**Szpilrajn-Marczewski 45**] proved the following theorem:

Theorem 1.0.1. *Let G be a multigraph with vertices $x_1, x_2, \ldots, x_n$. Then there exists a set S and a family of subsets $S_1, S_2, \ldots, S_n$ of S such that $|S_i \cap S_j|$ is equal to the number of edges in G joining x_i to x_j.*

A problem posed by [**Erdos,Goodman,Posa 66**] is the following: what is the minimum number of elements in a set S that satisfies Theorem 1.0.1. An element of S induces a clique in G. The minimum number of elements of S is $cp(G)$.

Similarly, the following theorem is true:

Theorem 1.0.2. *Let G be a graph with vertices $x_1, x_2, \ldots, x_n$. Then there exists a set S and a family of subsets $S_1, S_2, \ldots, S_n$ of S such that $|S_i \cap S_j| \geq 1$ iff x_i and x_j are adjacent.*

The minimum number of elements of a set S satisfying the previous theorem is $cc(G)$.

CHAPTER 2

The NP-completeness of some graph problems

2.1. Introduction to the theory of NP-completeness

The problems of covering and partitioning the edge-set of a graph with a minimum number of cliques (complete subgraphs) have been studied by a number of writers over the years, as have the related problems of covering and partitioning by bicliques (complete bipartite subgraphs).

We are interested in studying the complexity of these problems.

One of the main goals of this introductory part is to make explicit the connection between the formal terminology and the more intuitive, informal shorthand that is commonly used in its place. Once we have this connection well in hand, it will be possible for us to pursue our discussion primarily at the informal level.

As a matter of convenience, the theory of NP-completeness is designed to be applied only to decision problems. Such problems have only two possible solutions, either the answer “yes” or the answer “no”. Abstractly, a decision problem Π consists simply of a set D_Π of instances and a subset Y_Π of yes-instances. The reason for restriction to decision problems is that they have a very natural, formal counterpart, which is suitable to study in a mathematically precise theory of computation.

An important observation is that a decision problem can be formulated from an optimization problem.

If the optimization problem asks for a structure of a certain type that has a minimum "cost" among all such structures, we can associate with that problem the decision problem that includes a numerical bound B as an additional parameter and asks whether there exists a structure of the required type having cost no more than B. Decision problems can be derived from maximization problems in an analogous way, simply by replacing "no more than" by "at least". The key point to observe about this correspondence is that, so long as the cost function is relatively easy to evaluate, the decision problem can be no harder than the corresponding optimization problem.

Since we are interested in problems of finding the minimum number of cliques or bicliques which partition or cover the edge-set of a graph, it is easy to observe how we can derive from these problems decision problems which are equivalent to the initial optimization problem.

For example, from the problem of finding the minimum number of cliques which partition the edges of a graph we can derive the following decision problem:

CP(G,k)

INSTANCE: A graph $G = (V, E)$ and an integer k

QUESTION: Is there a partition of E into no more than k cliques ?

Now if we have a "good" algorithm for finding $cp(G)$ for any graph G, then we could use this algorithm to solve the decision problem.

Conversely, if we have a "good" algorithm for solving the decision problem **CP(G,k)** for any k, then :

$cp(G)$=min$\{k$:the answer to **CP(G,k)** is yes $\}$

The foundations for the theory of NP-completeness were laid in a paper of Stephen Cook, presented in 1971, entitled “The Complexity of Theorem Proving Procedures”. In his paper, Cook did several important things.

First, he emphasized the significance of “polynomial time reducibility”, that is, reductions for which the required transformation can be executed by a polynomial time algorithm. The principal technique used for demonstrating that two problems are related is that of “reducing” one to the other, by giving a constructive transformation that maps any instance of the first problem into an equivalent instance of the second. If we have a polynomial time reduction for one problem to another, this ensures that any polynomial time algorithm for the second problem can be converted into a corresponding polynomial time algorithm for the first problem.

Second, he focused attention on the class NP of decision problems that can be solved in polynomial time by a nondeterministic computer model, which has the ability to pursue an unbounded number of independent computational sequences in parallel. Informally, there is a polynomial time algorithm for checking any guess at a solution.

Third, he proved that one particular problem in NP, called the satisfiability problem, has the property that every other problem in NP can be polynomially reduced to it. If the satisfiability problem can be solved with a polynomial time algorithm by a deterministic computer model, then so can every problem in NP. Thus, in a sense the satisfiability problem is the hardest problem in NP.

Finally, Cook suggested that other problems in NP might share with the satisfiability problem this property of being one of the “hardest” members of NP.

Subsequently, Richard Karp presented a collection of results proving that indeed the decision problem versions of many well known combinatorial problems are just as "hard" as the satisfiability problem.

Since then a wide variety of other problems have been proved equivalent in difficulty to these problems, and this equivalence class, consisting of the "hardest" problems in NP, has been given a name: the class of NP-complete problems.

Formally, the definition of the class NP of decision problems involves notions such as nondeterministic Turing machines. We will not be interested in the formal definition of this class. The reader is referred to [**Garey,Johnson 79**] for details concerning this definition.

Informally, we can define NP in terms of what we shall call a "nondeterministic algorithm" [**Garey,Johnson 79**]. We view such an algorithm as being composed of two separate stages, the first being a guessing stage and the second a checking stage.

A nondeterministic algorithm "solves" a decision problem Π if the following two properties hold for all instances I $\in D_\Pi$:

1) If I $\in Y_\Pi$, then there is some structure S that , when guessed for input I will lead the checking stage to respond "yes" for I and S.

2) If I $\notin Y_\Pi$, then there exists no structure S that, when guessed for input I will lead the checking stage to respond "yes" for I and S.

A nondeterministic algorithm that solves a decision problem Π is said to operate in "polynomial time" if there exists a polynomial p such that for every instance I $\in Y_\Pi$, there is some guess S that leads the deterministic checking stage to respond "yes" for I and S within time p(length(I)).

The class NP is defined informally to be the class of all decision problems Π that can be solved by polynomial time nondeterministic algorithms. The use of the term "solve" in these informal definitions should, of course, be taken with a grain of salt. It should be evident that a polynomial time nondeterministic algorithm is basically a definitional device for capturing the notion of polynomial time verifiability, rather than a realistic method for solving decision problems.

The class P is defined using the notion of deterministic Turing machines. Informally, P is the class of decision problems which can be solved using deterministic polynomial time algorithms.

If there is a polynomial transformation from problem L_1 to problem L_2, we write $L_1 \propto L_2$.

The significance of the polynomial transformation comes from the following lemmas:

LEMMA 2.1.1. *If $L_1 \propto L_2$, then $L_2 \in P$ implies $L_1 \in P$ (and equivalently, $L_1 \notin P$ implies $L_2 \notin P$).*

LEMMA 2.1.2. *If $L_1 \propto L_2$ and $L_2 \propto L_3$, then $L_1 \propto L_3$.*

The proofs of these lemmas can be found in [**Garey,Johnson 79**].

We can now define the class of NP-complete problems.

A decision problem Π is NP-complete if $\Pi \in$ NP and for all other decision problems $\Pi' \in$ NP, $\Pi' \propto \Pi$.

Lemma 1 leads us to our identification of the NP-complete problems as "the hardest problems in NP". If any single NP-complete problem can be solved in polynomial time, then all the problems in NP can be solved by polynomial time algorithms. Most researchers believe that the NP-complete problems do not have polynomial time solution algorithms. This is the famous "P $\neq$ NP" problem.

The following lemma, which is an immediate consequence of our definitions and the transitivity of $\propto$, shows that matters would be simplified considerably if we possessed just one problem that we knew to be NP-complete.

LEMMA 2.1.3. *If L_1 and L_2 belong to NP, L_1 is NP-complete and $L_1 \propto L_2$, then L_2 is NP-complete.*

This lemma gives us a straightforward approach for proving new problems NP-complete.

To prove that Π is NP-complete, we merely show that:

1) $\Pi \in$ NP

2) some known NP-complete problem Π' transforms in polynomial time to Π.

Many graph theory problems have been shown to be NP-complete and so are believed not to have polynomial time algorithm.

[**Garey,Johnson 79**] give a list of then-known NP-complete problems and a bibliography on the subject. The list of NP-complete problems has grown significantly since then.

The question whether or not the NP-complete problems are intractable (i.e. so hard that no polynomial time algorithm can possibly solve them) is now considered to be one of the foremost open questions of contemporary mathematics and computer science. Despite the willingness of most researchers to conjecture that the NP-complete problems are all intractable, little progress has yet been made toward establishing either a proof or a disproof of this far-reaching conjecture. However, even without a proof that NP-completeness implies intractability, the knowledge that a problem is NP-complete suggests, at very least, that a major breaktrough will be needed to solve it with a polynomial time algorithm.

2.2. Graph-partition problems

We are concerned about the problems of partitioning or covering the edge-set of a graph with cliques or bicliques.

It was [**Orlin 77**] who first proved the next theorem :

THEOREM 2.2.1. *The following five problems are all NP-complete.*

P0) Determine the fewest number $ncc(G)$ of cliques which include all of the vertices of graph G.

P1) Determine the fewest number of bicliques of G which will cover a specified subset H of edges of G (where G is bipartite).

P2) Determine $bc(G)$ for a bipartite graph G.

P3) Determine $cc(G)$ for a graph G.

P4) Determine the minimum number of cliques that cover a specified subset of edges of G.

Proof :

Solving P0) is exactly the same as determining the chromatic number of the complement of G. P0) is proven to be NP-complete in [**Karp 72**].

For the rest of the proof it will be shown that P0) $\propto$ P1) $\propto$ P2) $\propto$ P3) $\propto$ P4) $\propto$ P0).

P0) $\propto$ P1)

Let G be a graph with vertices $v_1, v_2, \ldots, v_n$. Let G' be a graph with vertices $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_n$. In G' x_i is adjacent to y_i, for $i = 1$ to n. Also $x_i y_j$ is an edge in G' if $v_i v_j$ is an edge in G. The specified set H' of edges in G' are the edges $x_i y_i$ for all $i = 1$ to n.

Any clique C in G which includes v_i induces a complete bipartite subgraph in G' which includes the edge $x_i y_i$. Conversely, let C' be a biclique in G' that includes the edges $x_{i_1} y_{i_1}, x_{i_2} y_{i_2}, \ldots, x_{i_k} y_{i_k}$. Then in G it is true that $v_{i_1}, v_{i_2}, \ldots, v_{i_k}$ is a clique.

Thus the minimum number of cliques that cover the vertices of G is equal to the minimum number of bicliques in G' that cover the edges of H'.

P1) $\propto$ P2)

Let G be a bipartite graph with vertices $x_1, \ldots, x_s, y_1, \ldots, y_t$ and a specified subset H of edges. Let G' be another bipartite graph such that G is a subgraph of G'. In addition, for every edge $x_i y_j$ of G that is not in H there are two extra vertices in G' labeled x_{ij} and y_{ij}. In G' x_{ij} is adjacent to y_{ij}, x_{ij} is adjacent to y_j and x_i is adjacent to y_{ij}. Thus in G', $x_{ij} y_{ij}$ is an element of a unique maximal biclique that includes edge $x_i y_j$. We may assume that the biclique occurs in every biclique cover of G'. Since the construction has been carried out for every edge $x_i y_j \notin H$, it follows that all edges not in H have been covered by bicliques.

Thus the minimum number of bicliques of G that cover the edges of H is equal to $bc(G') - card(H^c)$, where H^c denotes the set of edges of G that are not in H.

P2) $\propto$ P3)

Let G be a bipartite graph with vertices $x_1, \ldots, x_s, y_1, \ldots, y_t$. We define a graph G' such that $G \subset G'$. In addition for every i and j such that $i \neq j$ we have in G' that x_i is adjacent to x_j and y_i is adjacent to y_j.

Finally, two extra vertices are in G'. Vertex z_1 is adjacent to every x_i. Vertex z_2 is adjacent to every y_j. In G' edges z_1x_1 and z_2y_1 belong to unique maximal cliques $z_1, x_1, \ldots, x_s$ and $z_2, y_1, \ldots, y_t$. The remaining edges in G' correspond to edges in G.

Thus $bc(G) = cc(G') - 2$.

P3) $\propto$ P4)

Choose H to be all edges in G.

P4) $\propto$ P0)

Let G be a graph with vertices $v_1, \ldots, v_n$. Let the specified subset of edges of G be $H = \{e_1, \ldots, e_m\}$. Let G' be another graph with vertices $w_1, \ldots, w_m$ where for $i \neq j$ w_i and w_j are adjacent in G' iff e_i and e_j belong to a common clique in G (i.e. if the 3 or 4 endpoints of edges e_i and e_j are all adjacent to each other). A clique $w_{j_1}, \ldots, w_{j_k}$ in G' corresponds to a clique in G which includes edges $e_{j_1}, \ldots, e_{j_k}$.

Thus, the minimum number of cliques of G covering the edges of H is equal to the minimum number of cliques of G' covering all the vertices.

This concludes the proof of Theorem 2.2.1.

Independently in [**Kou,Stockmeyer,Wong 78**], the problem P3) was studied in connection with a keyword conflict problem raised in [**Kellerman 78**]. There was made the connection between the keyword conflict problem, the intersection graphs and the covering of graphs by cliques.

They provided a proof that finding the minimum number of cliques that cover a graph G(i.e. $cc(G)$) is NP-complete problem.

Let first

SET_NCC = $\{ (G, k) : G$ graph, k integer, $ncc(G) \leq k\}$

SET_ECC = { (G, k) : G graph, k integer, $cc(G) \leq k$}

Theorem 2.2.2. **SET_ECC** *is NP-complete.*

Proof :

It is a trivial verification that **SET_ECC** belongs to NP.

Since **SET_NCC** is known to be NP-complete [**Karp 72**], we need only to show that **SET_NCC** is transformable to **SET_ECC** by a function computable in polynomial time.

Let (G, k) be a graph-integer pair. Let e be the number of edges of G. Consider $e+1$ additional vertices $U = \{u_1, \ldots, u_{e+1}\}$ and join each u_i to all vertices of G. Let the new graph be G'.

We first claim that $cc(G') \leq (e+1) \cdot ncc(G) + e$.

To prove this statement, let $\{C_1, C_2, \ldots, C_{ncc(G)}\}$ be a node-clique-cover for G. By construction, $\{u_i\} \cup C_j$ is a clique in G' and the set of all such cliques $\{u_i\} \cup C_j$ for all $1 \leq i \leq e+1$ and $1 \leq j \leq ncc(G)$ covers all edges from U to G. In order to cover all edges in G, we need at most e more cliques.

Thus the total is at most $(e+1) \cdot ncc(G) + e$.

We are going to prove now that $(G, k) \in$ **SET_NCC** iff $(G', k \cdot (e+1) + e) \in$ **SET_ECC**.

It is obvious from the previous relation that $(G, k) \in$ **SET_NCC**(i.e. $ncc(G) \leq k$) implies that $(G', k \cdot (e+1) + e) \in$ **SET_ECC**(i.e. $cc(G') \leq k \cdot (e+1) + e$).

Now suppose that $(G', k \cdot (e+1) + e) \in$ **SET_ECC**. Let S be a set of cliques of cardinality less or equal to $k \cdot (e+1) + e$ that covers the edges of G'. For $1 \leq i \leq e+1$, let $S_i \subseteq S$ be the set of cliques which contain the node u_i and let k_i be the cardinality of S_i. Note that $S_i \cap S_j = \emptyset$ for $i \neq j$ because u_i and u_j are not connected. For any i, the cliques in S_i with the node u_i deleted can clearly be used as a node-clique-cover for G.

Thus within polynomial time we can extract from S a node-clique-cover for G with cardinality equal to

$\min_{1 \leq i \leq e+1} k_i \leq \frac{\sum_{i=1}^{e+1} k_i}{e+1} \leq \frac{|S|}{e+1} \leq k + \frac{e}{e+1}$

Since the left side is an integer number, it follows that the inequality holds with k in the right side.

Thus within polynomial time we can find a node-clique-cover of G with fewer than k cliques which implies that $(G, k) \in$ **SET_NCC**.

Hence, finding $cc(G)$ for a graph G is a NP-complete problem. This completes the proof of Theorem 2.2.2.

Next we consider a similar problem of partitioning the edges of a graph into cliques.

We define this edge-partition problem EP_n as follows.

Given a graph $G = (V, E)$, the problem is to determine whether for a given n, the edge-set E can be partitioned into subsets $E_1, E_2, \ldots$ in such a way that each E_i is a subgraph of G isomorphic to the complete graph K_n with n vertices.

We shall present a proof due to [**Holyer 81**] that the problem EP_n is NP-complete for each $n \geq 3$. From this we deduce that a number of other edge-partition problems are also NP-complete.

In order to show that EP_n is NP-complete, we will exhibit a polynomial reduction from the known NP-complete problem **3SAT** which is defined as follows. A set of clauses $C = \{C_1, C_2, \ldots, C_r\}$ in variables $u_1, u_2, \ldots, u_s$ is given, each clause C_i

consisting of three literals $l_{i,1}, l_{i,2}, l_{i,3}$, where a literal $l_{i,j}$ is either a variable u_k or its negation $\overline{u_k}$.

The problem is to determine whether C is satisfiable; that is, whether there is a truth assignment to the variables which simultaneously satisfies all the clauses in C. A clause is satisfied if one or more of its literals has value "true".

Our first task is to find a graph which can be edge-partitioned into K_n's in exactly two distinct ways. Such a graph can be used as a "switch" to represent the two possible values "true" and "false" of a variable in an instance of **3SAT**.

For each $n \geq 3$ and $p \geq 3$ we define a graph $H_{n,p} = (V_{n,p}, E_{n,p})$ by

$V_{n,p} = \{x = (x_1, x_2, \ldots, x_n) \in \mathbb{Z}^n{}_p : \sum_{i=1}^{n} x_i \equiv 0(\text{modp})\}$

$E_{n,p}$=\{ xy : there exists i and j such that $x_k \equiv y_k(\text{modp})$ for $k \neq i, j$ and $y_i \equiv x_i + 1(\text{modp}), y_j \equiv x_j - 1(\text{modp})\}$

The properties of $H_{n,p}$ are given in the following lemma.

Lemma 2.2.3. *The graph $H_{n,p}$ has the following properties:*

(i) The degree of each vertex is $2\binom{n}{2}$.

(ii) The largest complete subgraph is K_n and any K_3 is contained in a unique K_n.

(iii) The number of K_n's containing a particular vertex is $2n$.

(iv) Each edge occurs in exactly $2K_n$'s.

(v) Each two distinct K_n's are either edge-disjoint or have just one edge in common.

(vi) There are just two distinct edge-partitions of $H_{n,p}$ into K_n's.

Proof :

(i) By translational symmetry we need only to consider $\mathbf{0}$=$(0, 0, \dots, 0)$. This is adjacent to $(1, -1, 0, \dots, 0)$ and the distinct nodes obtained from it by permuting its coordinates ($0, 1, -1$ are distinct modulo p since $p \geq 3$). There are clearly $2\binom{n}{2}$ of these.

(ii) By translation and coordinate permutations we may assume that a largest complete subgraph contains the vertices $\mathbf{0}$=$(0, 0, \dots, 0)$, $(1, -1, 0, \dots, 0)$ and $(1, 0, -1, \dots, 0)$. It is then forced to be the standard K_n, which we call K and whose vertices are:

$(0, 0, 0, \dots, 0)$

$(1, -1, 0, \dots, 0)$

$(1, 0, -1, \dots, 0)$

$\dots$

$(1, 0, 0, \dots, -1)$

(iii) The K_n's containing $\mathbf{0}$ are obtained from K and its inverse $-K$ by cyclic permutation of the coordinates. Thus there are $2n$ of them.

(iv) We need only to consider a particular edge containing the vertex $\mathbf{0}$ and check that is contained in just two of the K_n's given in (iii).

(v) If the two K_n's are not disjoint, we may assume that they have vertex $\mathbf{0}$ in common. We may then use (iii) to check that they have just one more vertex in common.

(vi) The edges containing $\mathbf{0}$ can be partitioned in at most two ways and these extend to the whole of $H_{n,p}$. All the K_n's are obtained from K or $-K$ by translation. One edge-partition consists of the translates of K and the other consists of the translates of $-K$.

This completes the proof of Lemma 2.2.3.

We now can make the following definitions. The $T-partition\ of\ H_{n,p}$ (corresponding to the logical value "true") consists of the translates of K and the $F-partition\ of\ H_{n,p}$ (corresponding to the logical value "false") consists of the translates of $-K$.

Two K_n's in $H_{n,p}$ are called *neighbors* if they have common edge.

A *patch* is a subgraph of $H_{n,p}$ consisting of the vertices and edges of a particular K_n and of its neighbors. It is a $T-patch$ if the central K_n belongs to the T-partition and it is an $F-patch$ otherwise.

Two patches P_1, P_2 in $H_{n,p}$ are called $noninterfering$ if the distance $d(x,y)$ in $H_{n,p}$ between vertices $x \in V(P_1)$ and $y \in V(P_2)$ is always at least 10, say.

THEOREM 2.2.4. *The edge-partition problem EP_n is NP-complete.*

Proof :

The problem EP_n is clearly in NP. Suppose we have an instance $C = \{C_1, C_2, \ldots, C_r\}$ of **3SAT** in s variables $u_1, u_2, \ldots, u_s$ where each C_i consists of the literals $l_{i,1}, l_{i,2}, l_{i,3}$. We reduce this instance of **3SAT** to an instance $G_n = (V_n, E_n)$ of EP_n as follows.

Choose p sufficiently large such that up to $3r$ noninterfering patches can be chosen in $H_{n,p}$, say $p = 100r$. Take a copy U_i of $H_{n,p}$ to represent each variable u_i and copies $C_{i,1}, C_{i,2}, C_{i,3}$ of $H_{n,p}$ to represent each clause C_i.

Join these copies of $H_{n,p}$ together as follows. If literal $l_{i,j}$ is u_k, then identify an F-patch of $C_{i,j}$ with an F-patch of U_k. If $l_{i,j}$ is $\overline{u_k}$, then identify an F-patch of $C_{i,j}$ with a T-patch of U_k.

Also join $C_{i,1}, C_{i,2}, C_{i,3}$ for each i by identifying one F-patch from each and then removing the edges of the central K_n.

Choose all those patches who occur in a single copy of $H_{n,p}$ to be noninterfering.

Denote by $G_n = (V_n, E_n)$ the graph obtained in this way. We now show that there is an edge-partition of G_n into K_n's if and only if the instance C of **3SAT** is satisfiable.

Suppose there is an edge-partition of G_n into a set S of K_n's and consider a particular copy H of $H_{n,p}$ involved in the construction of G_n. Take a K_n in S, say A, which is in H, but not near any join. Using the properties in the lemma, we see that the neighbors of A do not belong to S, the neighbors of the neighbors of A do belong to S and so on. Continuing in this way, we deduce that all the edges of H except perhaps those involved in joins are T-partitioned or all F-partitioned. Thus we may say that H is either T-partitioned or F-partitioned.

Now suppose $l_{i,j}$ is u_k and consider the join between $C_{i,j}$ and U_k. We claim that the edges in the vicinity of this join can be edge-partitioned into K_n's if and only if at least one of $C_{i,j}, U_k$ is T-partitioned. If $C_{i,j}$ is T-partitioned, this accounts for all the edges of $C_{i,j}$ near the join except for for those of the patch itself. The patch can then be regarded as belonging to U_k which can then be locally partitioned in either way. If on the other hand both $C_{i,j}$ and U_k are F-partitioned, the argument of the previous paragraph shows that the edges of the patch not belonging to the central K_n are forced to belong to the F-partitions of both $C_{i,j}$ and U_k, which is a contradiction.

Similarly if $l_{i,j}$ is $\overline{u_k}$, then either $C_{i,j}$ is F-partitioned or U_k is T-partitioned.

Now consider the join between $C_{i,1}, C_{i,2}$ and $C_{i,3}$. We claim that the edges in the vicinity of this join can be edge-partitioned into K_n's if and only if exactly one of $C_{i,1}, C_{i,2}, C_{i,3}$ is F-partitioned. The argument is the same as above, except that now, as the the central K_n is missing, the remaining edges of the patch must be claimed by an F-partition in exactly one of $C_{i,1}, C_{i,2}, C_{i,3}$.

Thus if G_n can be edge-partitioned into K_n's, then there is a truth assignment to $u_1, \ldots, u_s$ which satisfies C, namely u_k has value "true" if and only if U_k is T-partitioned.

If C is satisfiable, we partition G_n by partitioning U_k according to the truth of u_k in a satisfying assignment, choosing one "true" literal $l_{i,j}$ for each i and F-partitioning the corresponding $C_{i,j}$.

It should be clear that the above reduction from **3SAT** to EP_n can be carried out using a polynomial time algorithm and so the proof of the theorem is complete.

Now using Theorem 2.2.4 we are able to prove the next theorem.

THEOREM 2.2.5. *The following problems are NP-complete:*

(i) Find the maximum number of edge-disjoint K_n's in a graph ($n \geq 3$).

(ii) Find the maximum number of edge-disjoint maximal cliques in a graph.

(iii) Edge-partition a graph into the minimum number of complete subgraphs(i.e. finding $cp(G)$ for a graph G).

(iv) Edge-partition a graph into maximal cliques.

(v) Edge-partition a graph into cycles C_m of length m.

Proof :

For (i) we use the same construction as for EP_n. For (ii), (iii) and (iv) we use the same construction as for EP_3. Note that G_3 contains no K_4's and every edge K_2 is in a K_3, so the maximal cliques coincide with the K_3's.

For (v) we alter the construction for EP_3 in the following way. Note that the edges in $H_{3,p}$ occur in three distinct directions, say **a**, **b** and **c** and that the joins in the construction of G_3 are made so that edges which are identified have the same direction. In G_3, replace each edge with direction $\mathbf{a}$(say) by a path of $m-2$ edges.

This concludes the proof of Theorem 2.2.5.

So far, we have seen that determining $cc(G)$ and $cp(G)$ for a graph G are NP-complete problems.

We now consider the problem of partitioning the edge-set of a graph with bicliques.

Using an argument provided in [**Kratzke,Reznick,West 88**] we can prove the following theorem.

THEOREM 2.2.6. *Determining $bp(G)$ for a graph G is a NP-complete problem.*

Proof :

We make the reduction from the **VERTEX COVER** problem :

INSTANCE : A graph $G = (V, E)$ and a positive integer $k \leq |V|$.

QUESTION: Is there a vertex cover of size k or less for G, that is, a subset $V' \subseteq V$ such that $|V'| \leq k$ and for each edge $uv \in E$, at least one of u and v belongs to V' ?

The **VERTEX COVER** problem has been shown to be NP-complete by making a transformation from the **3SAT** problem in [**Karp 72**].

Let (G, k) be an instance of the **VERTEX COVER** problem. We construct the graph G' by replacing each edge of G by a path of 3 edges. It is obvious that G' contains no 4-cycles. Therefore the only complete bipartite subgraphs of G' are the stars. It follows that $bp(G') = \beta(G')$.

Since $\beta(G') = \beta(G) + |E|$, it follows that $bp(G') = \beta(G) + |E|$.

Hence, $\beta(G) \leq k$ if and only if $bp(G') \leq k + |E|$.

Thus determining $bp(G)$ for a graph G is a NP-complete problem.

Since $bc(G') = bp(G')$, the previous argument also proves the following theorem.

THEOREM 2.2.7. *Determining $bc(G)$ for a graph G is a NP-complete problem.*

Hence, determining $cp(G), cc(G), bp(G), bc(G)$ are all NP-complete problems.

In the next section we will study the complexity of these problems for graphs with a given maximum degree.

2.3. Some simplified graph-partition problems

In the previous section, we discussed the complexity of the following problems:

CC (clique covering of edges)

INSTANCE : A graph G and a positive integer k.

QUESTION : Is $cc(G) \leq k$?

CP (clique partition of edges)

INSTANCE : A graph G and a positive integer k.

QUESTION : Is $cp(G) \leq k$?

BC (biclique covering of edges)

INSTANCE : A graph G and a positive integer k.

QUESTION : Is $bc(G) \leq k$?

BP (biclique partition of edges)

INSTANCE : A graph G and a positive integer k.

QUESTION : Is $bp(G) \leq k$?

In this section we are interested in the following refinements of these problems.

For n a fixed positive integer, let **CC**(n) denote **CC** with the restriction that $\Delta(G) \leq n$. Define **CP**(n), **BC**(n), **BP**(n) similarly. The smaller n is, the easier(at least intuitively) **CC**(n), **CP**(n), **BC**(n), **BP**(n) will be.

Our aim is to find the smallest values of n for which the problems are NP-complete.

The following two theorems were proven in [**Hoover 92**].

THEOREM 2.3.1. **CP***(5) is NP-complete.*

Proof :

We reduce **IS**(3) to **CP**(5). The problem **IS**(3) which has been shown to be NP-complete in [**Garey,Johnson,Stockmeyer 76**] is the following :

IS(3) (independent set for graphs with degree at most 3)

INSTANCE : A graph G with $\Delta(G) \leq 3$ and a positive integer k.

QUESTION : Is $\alpha(G) \leq k$?

Let (G, k) be an instance of **IS**(3). Form the graph G' by replacing each edge ab of G by a path of 5 edges. Observe that $\alpha(G') = \alpha(G) + 2 \cdot |E|$ (*).

Now form G'' by replacing each vertex v of G by one of the complexes in Figure 1, according to the degree of v.

The edges (1),(2) and (3) correspond to edges e_1, e_2, e_3 of G which are incident to v. If v and w are joined by an edge e_1, then identify edge (1) in the v complex with the edge (1) of the w complex and so forth.

Further, $T(G'')$, the triangle graph of G'', is isomorphic to the graph G'. Hence by (*),

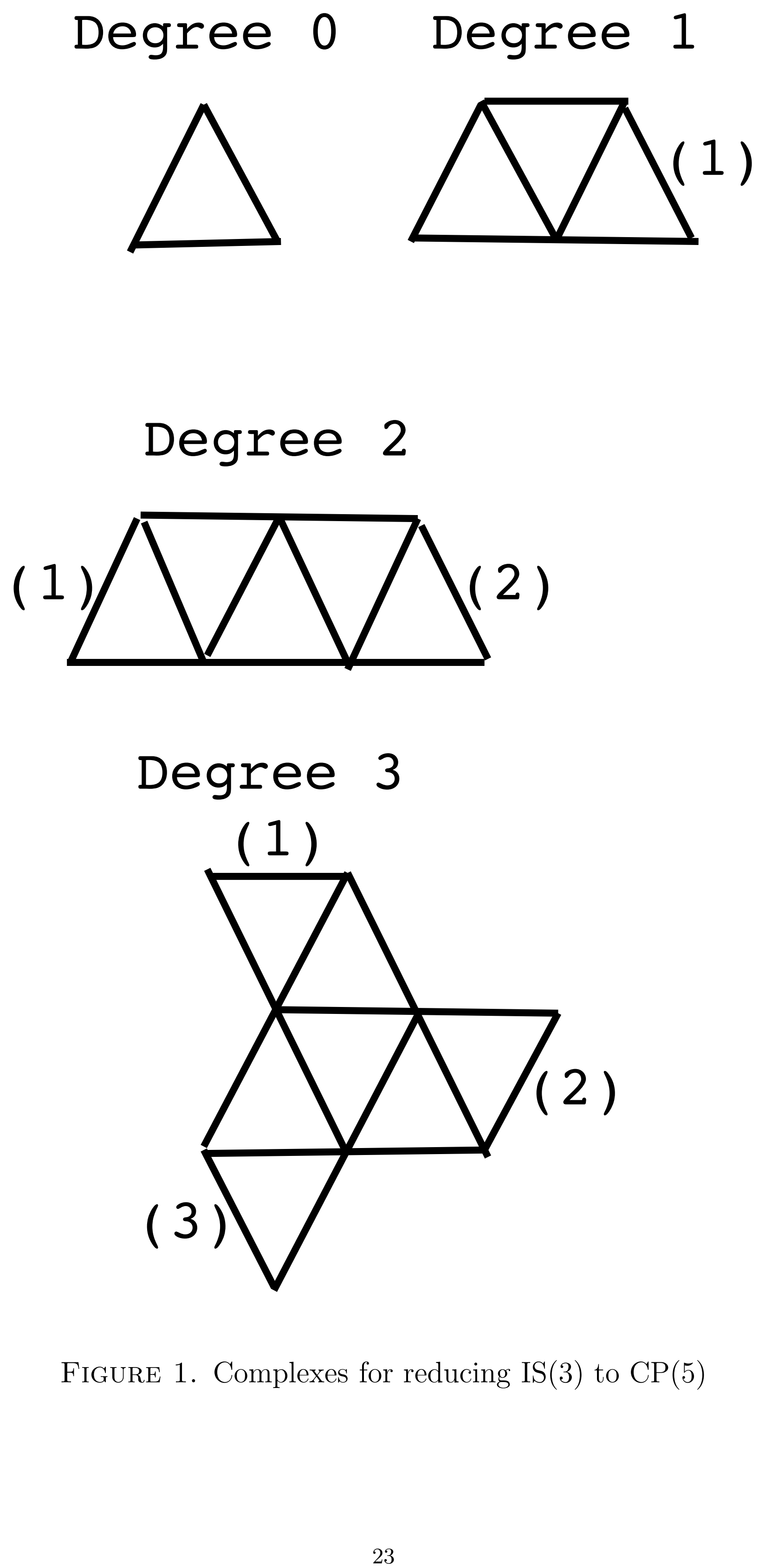


Figure 1. Complexes for reducing IS(3) to CP(5)

$cp(G'') = |E(G'')| - 2 \cdot (\alpha(G) + 2 \cdot |E(G)|)$.

Theorem 2.3.1 follows.

THEOREM 2.3.2. **CC***(6) is NP-complete.*

Proof :

We make the reduction from **VC**(3) to **CC**(6). The problem **VC**(3) which has been shown to be NP-complete in [**Garey,Johnson,Stockmeyer 76**] is the following :

VC(3) (vertex cover for graphs with maximum degree 3)

INSTANCE : Graph G with $\Delta(G) \leq 3$ and a positive integer k.

QUESTION : Is $\beta(G) \leq k$?

Let (G, k) be an instance of **VC**(3). Assume that G has no isolated vertices, since such a vertex can be omitted from any vertex cover. Form a graph G' by replacing each vertex v of G by one of the complexes $C(v)$ in Figure 2, according to the degree of v. As in the construction for **CP**(5), (1), (2) and (3) are edges of G incident to v. If v and w are joined by an edge e_1 in G, then the (1) edge of $C(v)$ is identified with the (1) edge of $C(w)$ and so forth. To ensure $\Delta(G') \leq 6$, this must be done so that the arrow on (1) in $C(v)$ has the opposite orientation to the arrow (1) in $C(w)$. Then the resulting graph G' has degree at most 6. Let $v_i(G)$ indicate the number of vertices of G which have degree i.

We have

$cc(G') = \beta(G) + 2 \cdot v_1(G) + v_2(G) + 6 \cdot v_3(G)$.

Theorem 2.3.2 follows immediately.

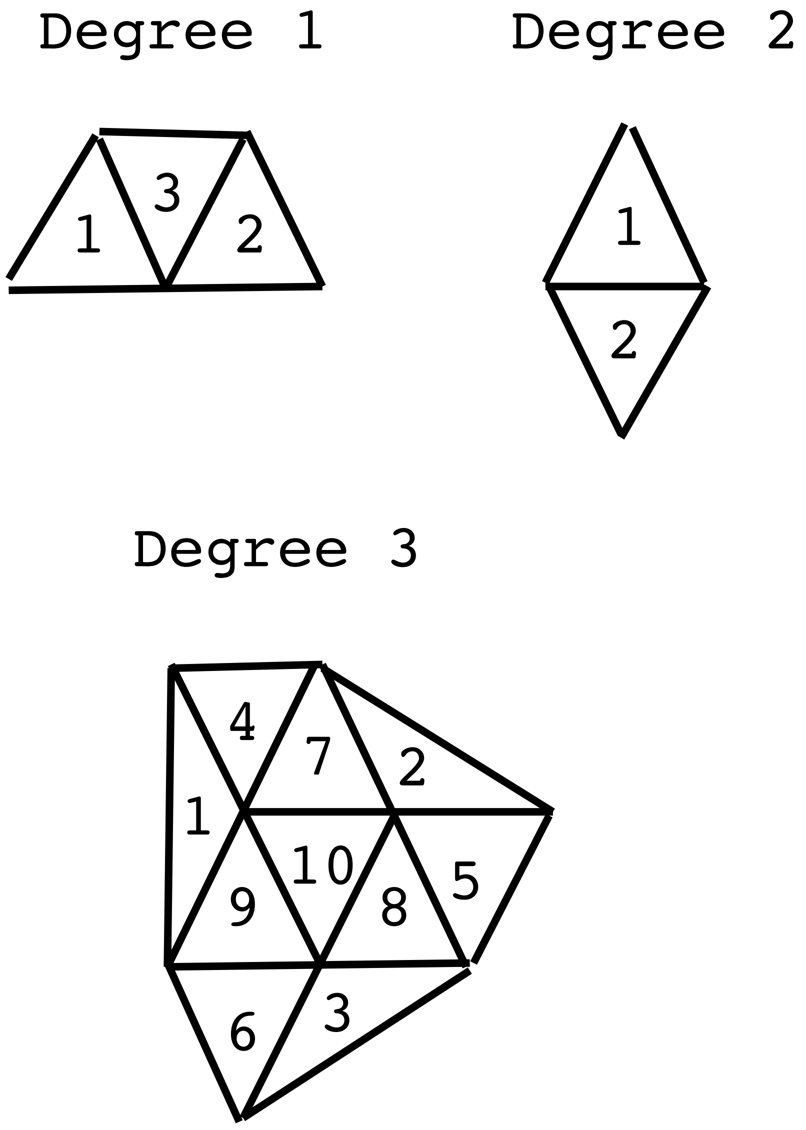


FIGURE 2. Complexes for reduction of VC(3) to CC(6)

In the case of bicliques, the corresponding problems seem a lot simpler. We can show the following :

THEOREM 2.3.3. **BP***(3) is NP-complete.*

Proof :

We make the reduction from the problem $\mathbf{VC}(3)$ which has been described in the proof of Theorem 2.3.2.

Let (G, k) be an instance of $\mathbf{VC}(3)$. We construct the graph G' by replacing each edge of G with a path of length 3. It is obvious that G' does not contain 4-cycles and $\Delta(G') \leq 3$ because $\Delta(G) \leq 3$. It follows that the only bicliques in G' are the stars. Hence $bp(G') = \beta(G')$.

Since $\beta(G') = \beta(G) + |E(G)|$, we deduce that $bp(G') = \beta(G) + |E(G)|$.

This implies that determining $bp(G)$ for graphs G with $\Delta(G) \leq 3$ is NP-complete.

One can see that this argument can be used for proving the next theorem.

Theorem 2.3.4. **BC***(3) is NP-complete.*

The existence of polynomial time algorithms for solving $\mathbf{CP}(4)$ [**Pullman 84**] and $\mathbf{CC}(5)$ [**Hoover 92**] as well as the fact that the graphs with maximum degree less than 3 are very easy to characterise (union of cycles and paths) implies that the values found in the previous theorems are the best possible.

We will present these polynomial algorithms as well as an heuristic for determining $cc(G)$ for a graph G in the next chapter.

CHAPTER 3

Clique covering of graphs. Algorithms

3.1. Heuristic for covering graphs with cliques

It is well known that the **NCC** problem is merely a restatement of the graph coloring problem(by considering the complementary graph).

[**Garey,Johnson 76**] have bounded the performance of polynomial-time graph coloring algorithms, under the assumption that $\mathbf{P} \neq \mathbf{NP}$. Translating their result to the **NCC** problem, it states that if $\mathbf{P} \neq \mathbf{NP}$ and if there are constants c and d and a polynomial-time algorithm A' for the **NCC** problem such that $A'(G) \leq c \cdot ncc(G) + d$ for all graphs G, then $c \geq 2$. A similar result for the **CC** problem will follow from the next theorem, which establishes a relationship between heuristic algorithms for the **NCC** and **CC** problems.

The following theorem was proven in [**Kou,Stockmeyer,Wong 78**].

Theorem 3.1.1. *Let c be a nonnegative constant. There is a polynomial time algorithm A for the* **CC** *problem such that A(G) ≤ c·cc(G) + d for all G if and only if there is a polynomial time algorithm A′ for the* **NCC** *problem such that A′(G) ≤ c·ncc(G) + d' for all G.*

Proof:

If.

Given an algorithm A' for the **NCC** problem we describe an algorithm A for the **CC** problem. Let G be a given graph for the **CC** problem. Say that G has edges

$e_1, e_2, \ldots, e_m$. Form a graph G' with nodes $n_1, n_2, \ldots, n_m$ and edges as follows. For $1 \leq i \leq j \leq m$, let I_{ij} denote the set of nodes in G upon which e_i and e_j are incident (so that I_{ij} contains either three or four nodes, depending on whether or not e_i and e_j share an endpoint). Join n_i and n_j by an edge in G' iff I_{ij} contains the nodes of a complete subgraph of G. It is easy to verify that $cc(G) = ncc(G')$. The algorithm A constructs G' and applies A' to G'.

Only if.

Let A be a given algorithm for the **CC** problem. Let G be an input for the **NCC** problem and say that G has e edges. Consider t additional nodes $U = \{u_1, u_2, \ldots, u_t\}$, where $t = [c \cdot e + d + 1]$. Join each u_i to all nodes in G. Let the new graph be G'.

We first claim that $cc(G') \leq t \cdot ncc(G) + e$.

To prove this statement, let $C_1, C_2, \ldots, C_{ncc(G)}$ be a node clique cover for G. By construction, $u_i \cup C_j$ is a clique in G' and the set of all such cliques $u_i \cup C_j$ for $1 \leq i \leq t$ and $1 \leq j \leq ncc(G)$ covers all the edges from U to G. To cover the edges of G, we need at most e more cliques. Thus the total is at most $t \cdot ncc(G) + e$. The polynomial time algorithm A applied to G' produces a set S of cliques which cover the edges of G'. By hypothesis, the cardinality of S is $A(G) \leq c \cdot cc(G') + d \leq c \cdot (t \cdot ncc(G) + e) + d$.

For $1 \leq i \leq t$, let $S_i \subseteq S$ be the set of cliques which contain the node u_i and let k_i be the cardinality of S_i. Note that $S_i \cup S_j = \emptyset$ for $i \neq j$ because u_i and u_j are not connected. For any i, the cliques in S_i with the node u_i deleted can clearly be used as a node clique cover for G. Thus within polynomial time we can extract from S a node clique cover for G with cardinality equal to

$\min_{1 \leq i \leq t} k_i \leq [\frac{\sum_{i=1}^{t} k_i}{t}] \leq [\frac{A(G')}{t}] \leq$ c$\cdot ncc(G) + 1$.

The following corollary is immediate from Theorem 3.1.1 (only if) and the result of Garey and Johnson mentioned above.

Corollary 3.1.2. *If* $\mathbf{P} \neq \mathbf{NP}$ *and if there are nonnegative constants c and d and a polynomial time algorithm A for the* **CC** *problem such that $A(G) \leq c \cdot cc(G) + d$ for all graphs G, then $c \geq 2$.*

In view of this result, it seems unlikely that there is a polynomial time algorithm A for the **CC** problem with a ratio $\frac{A(G)}{cc(G)}$ less than 2 and there is even cause to believe that there is no such algorithm with bounded ratio. The literature contains a number of heuristic algorithms for the **NCC** problem (in the guise of graph coloring algorithms); each such **NCC** algorithm yields an **CC** algorithm by Theorem 3.1.1 (if).

However, [**Johnson 2,74**] has analyzed the behaviour of several coloring algorithms and has shown the ratio $\frac{A(G)}{ncc(G)}$ to be unbounded for all of those which he considered. In light of Theorem 3.1.1 (only if), this adds weight to the suspicion that there is no polynomial time **CC** algorithm with $\frac{A(G)}{cc(G)}$ bounded.

[**Kellerman 78**] has proposed a polynomial time heuristic for the keyword conflict problem.

[**Kou,Stockmeyer,Wong 78**] constructed an improved version of the heuristic.

When translated to the **CC** problem, the heuristic in [**Kellerman 78**] can be stated as follows. Let G be a given graph input without isolated nodes. We assume that the nodes of G have been labelled $1, 2, \ldots, n$. The algorithm forms cliques $C_1, C_2, \ldots$ by examining the nodes one by one. When a node i is examined, only edges connecting i to nodes with smaller label are considered. In the following description of the algorithm, i is the node currently being examined and k is the number of cliques which have been created so far. The set W contains those nodes $j \leq i$ such that an edge connecting $\{i, j\}$ has yet to be covered.

Algorithm

1. Initialize $k \leftarrow 0$ and $i \leftarrow 0$.

2. Set $i \leftarrow i + 1$ and terminate the algorithm if $i \leq n$.

Set $W \leftarrow \{j | j \leq i$ and $\{i, j\}$ are connected in G$\}$.

3. If $W = \emptyset$, then set $k \leftarrow k + 1$, create a new clique $C_k = \{i\}$ and go to 2.

4. Try to insert i into existing cliques :

Set $m \leftarrow 1$ and $V \leftarrow \emptyset$.

While $m \leq k$ and $V \neq W$ do

If $C_m \subseteq W$, then set $C_m \leftarrow C_m \cup \{i\}$ and $V \leftarrow V \cup C_m$.

$m \leftarrow m + 1$

5. Update W to account for those edges which were covered in step 4 : $W \leftarrow W - V$.

6. If $W = \emptyset$, then go to 2. Otherwise new cliques must be added. Find an $m, 1 \leq m \leq k$, such that the cardinality of $C_m \cap W$ is maximal; break ties by choosing the smallest such m. Set $k \leftarrow k + 1, C_m \leftarrow (C_m \cap W) \cup \{i\}, W \leftarrow W - C_m$ and go to 6.

Note that the step 3 cannot produce useless singleton cliques because we have assumed that G contains no isolated nodes. The cliques initialized in step 3 are subsequently grown in step 6.

The improvement proposed in [**Kou,Stockmeyer,Wong 78**] requires one additional pass after the original algorithm terminates in step 2 :

7. Suppose $C_1, C_2, \ldots, C_k$ are the cliques produced by the previous heuristic. Examine the cliques one by one to see if the edges covered by a clique are a subset of the union of the edges covered by the remaining edges. If a clique is subsumed by the union of the remaining cliques, then eliminate it.

This version is still polynomial time and it certainly will never produce more cliques than the original heuristic.

It should pointed out that, although a straightforward implementation of the clique elimination step 7 runs in polynomial time, this step might consume more time than the original heuristic. This increase could be, in the worst case, a factor proportional to the number of cliques found by the original heuristic.

The family of graphs $\{G_{m,t}\}$ constructed in [**Kou,Stockmeyer,Wong 78**] illustrates that even the improved heuristic can produce a number of cliques which is arbitrarily large multiple of the optimal number. Let $\{a_1, a_2, \ldots, a_m\}$ be the nodes of a complete graph A and let $\{b_1, b_2, \ldots, b_m\}$ be the nodes of a complete graph B. Join $\{a_i, b_i\}$ for $i = 1, 2, \ldots, m$. Let $\{u_1, u_2, \ldots, u_t\}$ be a set if disjoint nodes. Join each u_i to all nodes in A and B. The graph thus constructed is the required $G_{m,t}$.

We label the nodes of $G_{m,t}$ in the order $u_1, u_2, \ldots, u_t, a_1, b_1, a_2, b_2, \ldots, a_m, b_m$. Then steps 1-6 of the heuristic produce the following cliques:

$C_i = \{a_1, b_1, u_i\} i = 1, 2, \ldots, t,$

$C_{t+1} = \{a_1, a_2, \ldots, a_m, u_1\},$

$C_{t+i} = \{a_2, b_2, u_i\} i = 2, 3, \ldots, t,$

$C_{2t+1} = \{b_1, b_2, \ldots, b_m, u_1\},$

$C_{(t-1)(j-1)+i+2} = \{a_j, b_j, u_i\} 2 \leq i \leq t, 3 \leq j \leq m.$

For example, when b_2 is under consideration, $C_1, C_2, \ldots, C_{2t}$ are already in existence. b_2 is connected to $u_1, u_2, \ldots, u_t, a_2$ and b_1, so that b_2 can be inserted into the cliques $C_{t+2}, \ldots, C_{2t}$. This leaves only the edges $\{b_1, b_2\}$ and $\{u_1, b_2\}$ uncovered; so a new clique C_{2t+1} is created. Thereafter, when examining a_j for $j \geq 3$, step 4 of the algorithm inserts a_j into C_{t+1}. No other insertions are possible because all other existing cliques contain some node in B which is not connected to a_j. Therefore step 6 creates $t-1$ new cliques to cover the edges from a_j to $u_2, u_3, \ldots, u_t$. Then b_j is inserted into C_{2t+1} and into the $t-1$ cliques which were just created for a_j. Note that the number of cliques is $m \cdot (t-1) + 3$.

Consider now the effect of the clique elimination step 7 when applied to this edge clique cover. C_1 can be eliminated, but each of the other cliques C_j for $j \geq 2$ covers an edge which is covered by no other clique. For example, C_j is the only clique which covers the edge connecting u_j to a_1, for $j = 2, 3, \ldots, t$. Letting H denote the heuristic algorithm consisting of steps 1-7 above, we therefore have

$H(G_{m,t}) = m \cdot (t-1) + 2.$

On the other hand, if we form cliques in the following manner:

$C'_i = \{a_1, \ldots, a_m, u_i\}, i = 1, 2, \ldots, t,$

$C'_{t+i} = \{b_1, \ldots, b_m, u_i\}, i = 1, 2, \ldots, t,$

$C'_{2t+j} = \{a_j, b_j\}, j = 1, 2, \ldots, m,$

then $\{C'_1, \ldots, C'_{2t+m}\}$ is an edge clique cover for $G_{m,t}$.

Therefore,

$cc(G_{m,t}) \leq 2t + m.$

Choosing $m = t$, we have

$\frac{H(G_{m,t})}{cc(G_{m,t})} \geq \frac{m\cdot(t-1)+2}{2t+m} = \frac{t^2-t+2}{3t} \to \infty$ as $t \to \infty$.

In the next two sections, we will present polynomial algorithms for solving **CC**(4) [**Pullman 84**] and **CC**(5) [**Hoover 92**]. The idea of the algorithms is that in graphs of the appropriate low degree, configurations which can make it difficult to find a minimum clique covering can only occur in small clique component of G or of a graph derived from G. The problem can then be solved component by component, on large components by an easy algorithm and on small components by brute force.

3.2. Polynomial algorithms for CP(4) and CC(4)

In this section, we will present polynomial time algorithms for determining minimal clique coverings and partitions and clique covering and partition numbers of graphs with maximal degree at most 4. These algorithms were constructed in [**Pullman 84**]

A brute force aprroach to the calculation of these entities for graphs on n vertices would involve an exponential function of n operations (exponential time) even if the maximal degree is 4. The algorithms proposed in [**Pullman 84**] accomplish their work in $O(n)$ operations (linear time).

Since the general problems (arbitrary maximal degree) **CP** and **CC** were shown to be NP-complete in the previous section, presenting linear algorithms for moderately large subclass of graphs may be of interest.

We use $\overline{H}(G)$ to denote the subgraph of G obtained by deleting the edges, but not the vertices of H from G. This subgraph $\overline{H}(G)$ is called the *complement of H in G*. We write $\overline{H}$ instead of $\overline{H}(G)$ when it will cause no confusion.

We say that a graph H *separates the cliques of* G, if, for every clique K of G, either every edge of K lies in H or every edge of K lies in $\overline{H}$.

The proofs omitted from the following two lemmas are obvious.

Lemma 3.2.1. *The following statements are equivalent:*

(i) H separates the cliques of G

(ii) $\overline{H}$ separates the cliques of G

(iii) If K is a triangle in G then all its edges lie in H or all its edges lie in $\overline{H}$.

Lemma 3.2.2. *If H separates the cliques of G then $cp(G) = cp(H) + cp(\overline{H})$ and $cc(G) = cc(H) + cc(\overline{H})$.*

Lemma 3.2.3. *If $\mathcal{H}$ is a family of subgraphs each separating the cliques of G then the union of the subgraphs in $\mathcal{H}$ separates the cliques of G.*

Proof:

Let K be a clique in G and L be the union of all members of $\mathcal{H}$. If $K \not\subset L$, then $K \nsubseteq H$ for all $H \in \mathcal{H}$; therefore $K \subseteq \overline{H}$ for all $H \in \mathcal{H}$, consequently $K \subseteq \cap\{\overline{H} : H \in \mathcal{H}\} = \overline{L}$. Therefore L separates the cliques of G.

Suppose H is a subgraph of G. Define the *neighborhood* of H, $N(H)$, to consist of H and every vertex and edge of G adjacent to vertices of H. It has been proven in [**deCaen,Pullman 81**] that:

Lemma 3.2.4. *If $\Delta(G) = k$ and K is a k-clique, then $N(K)$ separates the cliques of G.*

If v is a vertex of G adjacent to, but not a vertex of, a subgraph H, we say v is *externally adjacent* to H. If vx is an edge of G, v is externally adjacent to H and x is a vertex of H, then we say that vx is *externally adjacent to H (at x)*. Let $\nu'(H)$ and $\epsilon'(H)$ denote the number of vertices and edges of G externally adjacent to H.

LEMMA 3.2.5. *If $\Delta(G) = k$ and K is a k-clique of G then $\nu'(K) \leq k$, $\epsilon'(K) \leq k$ and, unless $N(K)$ is a $(k+1)$-clique, cc($N(K)$) = 1 + $\nu'(K)$ and cp($N(K)$) = 1 + $\epsilon'(K)$.*

Proof:

There is at most one vertex and one edge externally adjacent to K, per vertex of K, because K is a $\Delta(G)$-clique . Therefore $\nu'(K) \leq k$ and $\epsilon'(K) \leq k$. For each vertex v externally adjacent to K, let $C(v)$ be the clique whose vertices consist of v and and those vertices of K adjacent to v. If v' is another vertex externally adjacent to K, then $C(v')$ has no vertices in common with $C(v)$, as there is at most one edge externally adjacent to K per vertex of K. Therefore $C(v)$ and $C(v')$ are in different cliques of any covering $\mathcal{C}$ of $N(K)$. Moreover, K and $C(v)$ are in different members of $\mathcal{C}$ because $N(K)$ is not a $(k+1)$-clique. Therefore $\mid \mathcal{C} \mid \geq 1 + \nu'(K)$, but K and the family of all $C(v)$ with v externally adjacent to K form a clique covering of $N(K)$ having $1 + \nu'(K)$ members. Therefore $\text{cc}(N(K)) = 1 + \nu'(K)$.

It was shown in [**deCaen,Pullman 81**] that $\text{cp}(N(K)) = 1 + \epsilon'(K)$ when $\epsilon'(K) = k$. Suppose $\epsilon'(K) \leq k$, then K and its $\epsilon'(K)$ externally adjacent adjacent edges form a clique partition of $N(K)$. Therefore $\text{cp}(N(K)) \leq 1 + \epsilon'(K)$. By adjoining $k - \epsilon'(K)$ edges, one to each of the $k - \epsilon'(K)$ vertices of K of degree $k - 1$ and augmenting a minimal clique partition $\mathcal{C}$ of $N(K)$ by these edges , we can form a clique partition $\mathcal{C}'$ of a new neighborhood $N'(K)$ of K, with k edges externally adjacent to K . Therefore, $\mid \mathcal{C}' \mid = \mid \mathcal{C} \mid + k - \epsilon'(K)$. If $\mid \mathcal{C} \mid \leq 1 + \epsilon'(K)$, then we would have $\mid \mathcal{C}' \mid \leq 1 + k$, which is a contradiction with the result proved in [**deCaen,Pullman 81**].

If H is a subgraph containing some, but not all of the edges and vertices of G, then we say that H is a *proper* subgraph. If a proper nonempty subgraph separates the cliques of G, then we say that G is *clique-separable*. Note that the empty graphs

(those without any edges) are clique-inseparable. If a subgraph B of G separates the cliques of G, but no proper nonempty subgraph of B does so, then B is called a *clique-block* of G. Note that then B is clique-inseparable in itself.(If L were a proper subgraph of B separating the cliques of B, then L would also separate the cliques of G because $\overline{L}(B) \subseteq \overline{L}(G)$.) Therefore a subgraph B is a clique block of G if and only if B is a clique inseparable graph and B is not a subgraph of any other clique-inseparable graph contained in G. The definitions directly imply the following lemma.

Lemma 3.2.6. *(a) Edges contained in no triangles of G and (b) triangles sharing edges with no other triangles of G are clique blocks in G.*

Lemma 3.2.7. *The family of all clique blocks in G partitions the edge set of G.*

Proof:

Let B and H be clique blocks in G. Suppose the edge sets $E(B)$ and $E(H)$ of B and H are not disjoint. Let t be any triangle in G. If t contains an edge of $B \cap H$ then $t \subseteq B$ and $t \subseteq H$, because B and H separate the cliques of G. Therefore $t \subseteq B \cap H$ if (and only if) t contains an edge of $B \cap H$. If t does not contain an edge of $B \cap H$, then all three edges of t are in B and none are in H or all are in H and none are in B or none are in $B \cup H$ (as B and H separate cliques). In any case, $t \subseteq \overline{B \cap H}$. Thus any triangle is either wholly in $B \cap H$ or wholly in $t \subseteq \overline{B \cap H}$. Therefore, by Lemma 3.2.1, $B \cap H$ separates the cliques of G and hence $B = H$.

A subgraph D of G is *deletable* if $\mathrm{cp}(G) = \mathrm{cp}(D) + \mathrm{cp}(\overline{D})$. For example, by Lemma 3.2.2, subgraphs that separate the cliques of G are deletable. Applying Lemma 3.2.4 (and Lemma 3.2.5 for part (d)) we have :

Lemma 3.2.8. *The following are deletable subgraphs of G:*

(a) isolated vertices,

(b) neighborhoods of $\Delta(G)$-cliques(in particular, $(\Delta(G)+1)$-cliques),

(c) triangles sharing no edges with other triangles of G,

(d) $\Delta(G)$-cliques.

Moreover (e) if H' is a deletable subgraph of H and H is a deletable subgraph of G, then H' is a deletable subgraph of G.

We recall that if G has some triangles, the graph T whose vertices are the triangles of G, with two distinct vertices of T deemed adjacent if, as triangles in G they share a common edge, is called the *triangle graph* of G. We denote T as $T(G)$. If G is triangle-free, then we define $T(G) = \emptyset$.

LEMMA 3.2.9. *If every edge of G lies in some triangle of G, then the triangle graph of G is connected if and only if G is clique inseparable.*

Proof:

Suppose that the triangle graph T of G has more than one connected component and T_1 is one of them. There is some subgraph H of G for which $T(H) = T_1$. Every triangle of G lies in H or $\overline{H}$, because every vertex of T lies in T_1 or some other connected component of G. Therefore by Lemma 3.2.1(c), H separates the cliques of G. The subgraph H must be proper, because $T \neq T_1$. Therefore G is clique separable. Conversely, if G is clique separable, then it has more than one clique block. Let H_1 be one of them and H_2 another. If an edge of $T(G)$ had one end in $T(H_1)$ and the other in $T(H_2)$, then a triangle in H_1 would share an edge with a triangle in H_2. This contradicts Lemma 3.2.7 which implies that the edge sets of H_1 and H_2 are disjoint. Therefore the triangle graph of G is disconnected.

Corollary 3.2.10. *The connected components of $T(G)$ are the images under T of those clique blocks of G which are not edges.*

Lemma 3.2.11. *Suppose G contains no 4-cliques and $\mathcal{T}(\mathcal{C})$ denotes the set of triangles in a clique partition $\mathcal{C}$ of G. If $\mathcal{T}$ is a maximum independent set of vertices of $T(G)$, then there exists a minimal clique partition $\mathcal{C}$ of G such that $\mathcal{T} = \mathcal{T}(\mathcal{C})$. Conversely, if $\mathcal{C}$ is a minimal clique partition of G, then $\mathcal{T}(\mathcal{C})$ is a maximum independent set of vertices of $T(G)$.*

Proof:

Let $\mathcal{C}_0$ be a fixed minimal clique partition of G and $\mathcal{C}$ be an arbitrary clique partition. We have $|\mathcal{T}(\mathcal{C})| \leq |\mathcal{T}(\mathcal{C}_0)|$ with equality if and only if $\mathcal{C}$ is a minimal clique partition because G is 4-clique free. Moreover $\mathcal{T}(\mathcal{C})$ is independent because $\mathcal{C}$ partitions the edges of G. If $\mathcal{T}_0$ is a maximum independent set of vertices of $T(G)$, then in particular, $|\mathcal{T}_0| \leq |\mathcal{T}(\mathcal{C}_0)|$. Augment $\mathcal{T}_0$ by those edges of G which are in none of the triangles in $\mathcal{T}_0$ to form a clique partition $\mathcal{C}_1$ of G with $\mathcal{T}(\mathcal{C}_1) = \mathcal{T}_0$. Thus $|\mathcal{T}(\mathcal{C}_1)| = |\mathcal{T}(\mathcal{C}_0)|$ and hence $\mathcal{C}_1$ is minimal and, as $|\mathcal{T}(\mathcal{C}_1)| = |\mathcal{T}_0|$, $\mathcal{T}(\mathcal{C}_1)$ is a maximum independent set. If $\mathcal{C}$ is a minimal clique partition of G, then $|\mathcal{T}(\mathcal{C})| = |\mathcal{T}(\mathcal{C}_0)|$ and hence $\mathcal{T}(\mathcal{C})$ is maximum.

Corollary 3.2.12. *If G is 4-clique free, then:*

$cp(G) = \epsilon(G) - 2\alpha(T(G))$.

Corollary 3.2.13. *If P_j, P'_j and C_j are defined as in Fig 1., then (a) $cp(P_j) = cp(P'_j) = j + \frac{1+(-1)^j}{2}$ and (b) $cp(C_j) = j + \frac{1-(-1)^j}{2}$.*

Proof:

(a) The graph $T(P_j)$ is a simple path of length $j-1$,

$T(P_j') = T(P_j), \epsilon(P_j') = \epsilon(P_j) = 2j+1, \alpha(T(P_j)) = \frac{j+\frac{1-(-1)^j}{2}}{2}$.

(b) The graph $T(C_j)$ is simple cycle of length j,

$\epsilon(C_j) = 2j, \alpha(T(C_j)) = \frac{j-\frac{1-(-1)^j}{2}}{2}$.

The graphs of Fig 3.1, Fig 3.2 and Fig 3.3 are all clique inseparable by Lemma 3.2.9 as their triangle graphs are simple paths (in the case of P_j and P_j') , simple cycles (in the case of C_j) and other connected graphs T_j.

The following theorem establishes that apart from K_2, K_5 and $\overline{K}_n$, these are the only clique inseparable graphs of maximum degree at most 4.

Theorem 3.2.14. *If G is clique inseparable and $\Delta(G) \leq 4$, then G is $\overline{K}_n, K_2, K_5$, one of the ten graphs G_i, one of the paths P_j, P_j' or C_j.*

Proof:

If G is triangle-free, then G has 0 or 1 edges by Lemma 3.2.6(a). Therefore G is edgeless or $G = K_2$.

Therefore, we may suppose that $T(G) \neq \emptyset$. According to Lemma 3.2.9, $T(G)$ is connected.

Suppose first that G contains a 4-clique, H. By Lemma 3.2.4, $N(H)$ separates the cliques of the clique inseparable graph G. Therefore $G = N(H)$. According to Lemma 3.2.5, H has ϵ' externally adjacent edges (at most one per vertex of H). Inseparability implies that $\epsilon' \neq 1$. Therefore the number of edges of $G, \epsilon(G) = 6, 8, 9, 10$. If $\epsilon(G) =$ 6, 8 or 9, then $G = G_6, G_7$ or G_9 respectively. If $\epsilon(G) = 10$, then $G = G_8$ or K_5 accordingly as G has 6 or 5 vertices respectively.

We may assume now that G is 4-clique free. Suppose that $T(G)$ contains a triangle. Call its vertices x, y and z. These are triangles in G. Consider the intersection of their

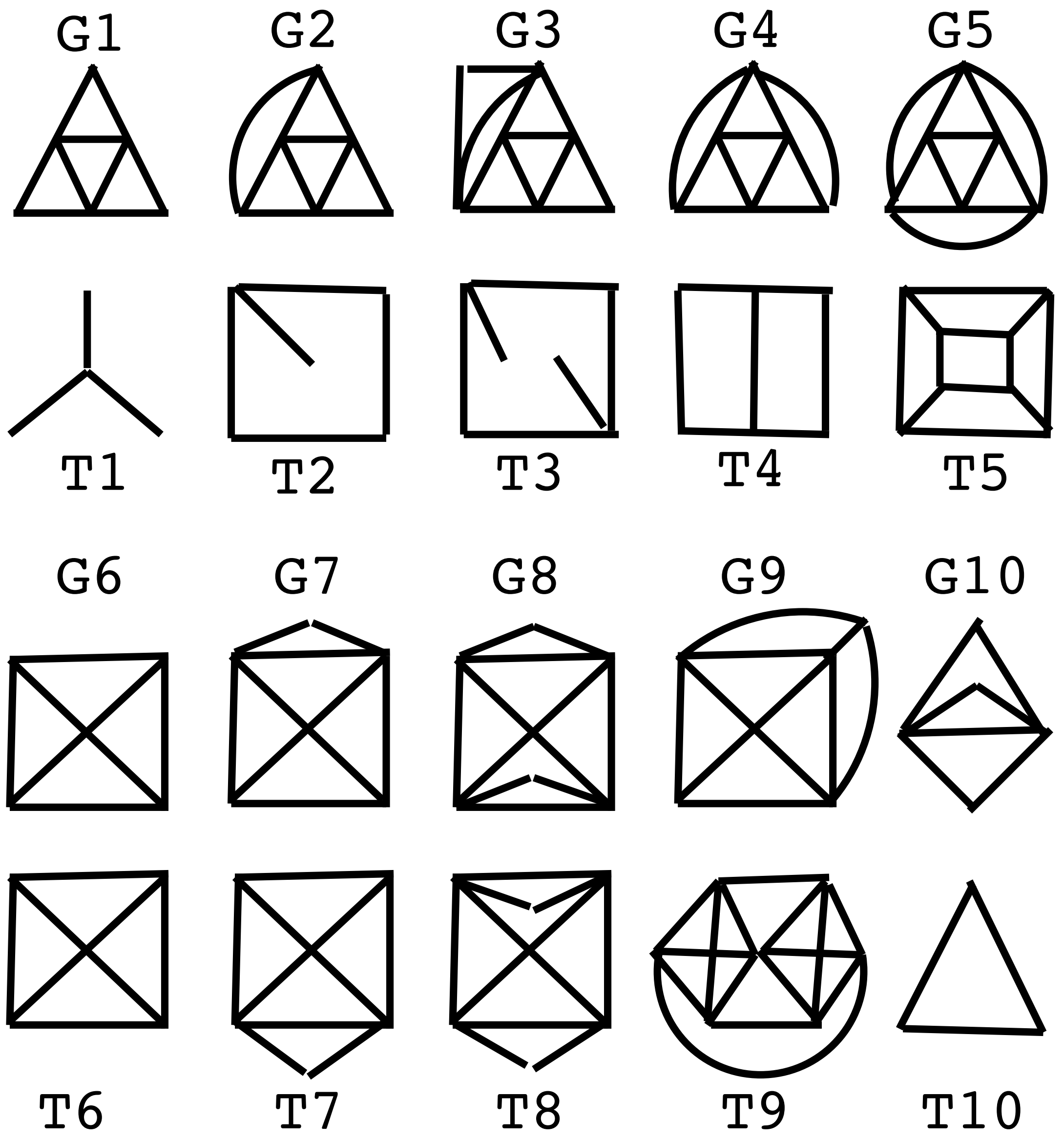


FIGURE 1. The graphs G_i with $i = 1, 2, \ldots, 10$

edge sets in G. If it were empty, then as each of their pairwise edge intersections is nonempty, their union would form a 4-clique. Therefore the triangles x, y and z share an edge and hence G contains G_{10}. Since G is 4-clique free and clique inseparable it follows that $G = G_{10}$.

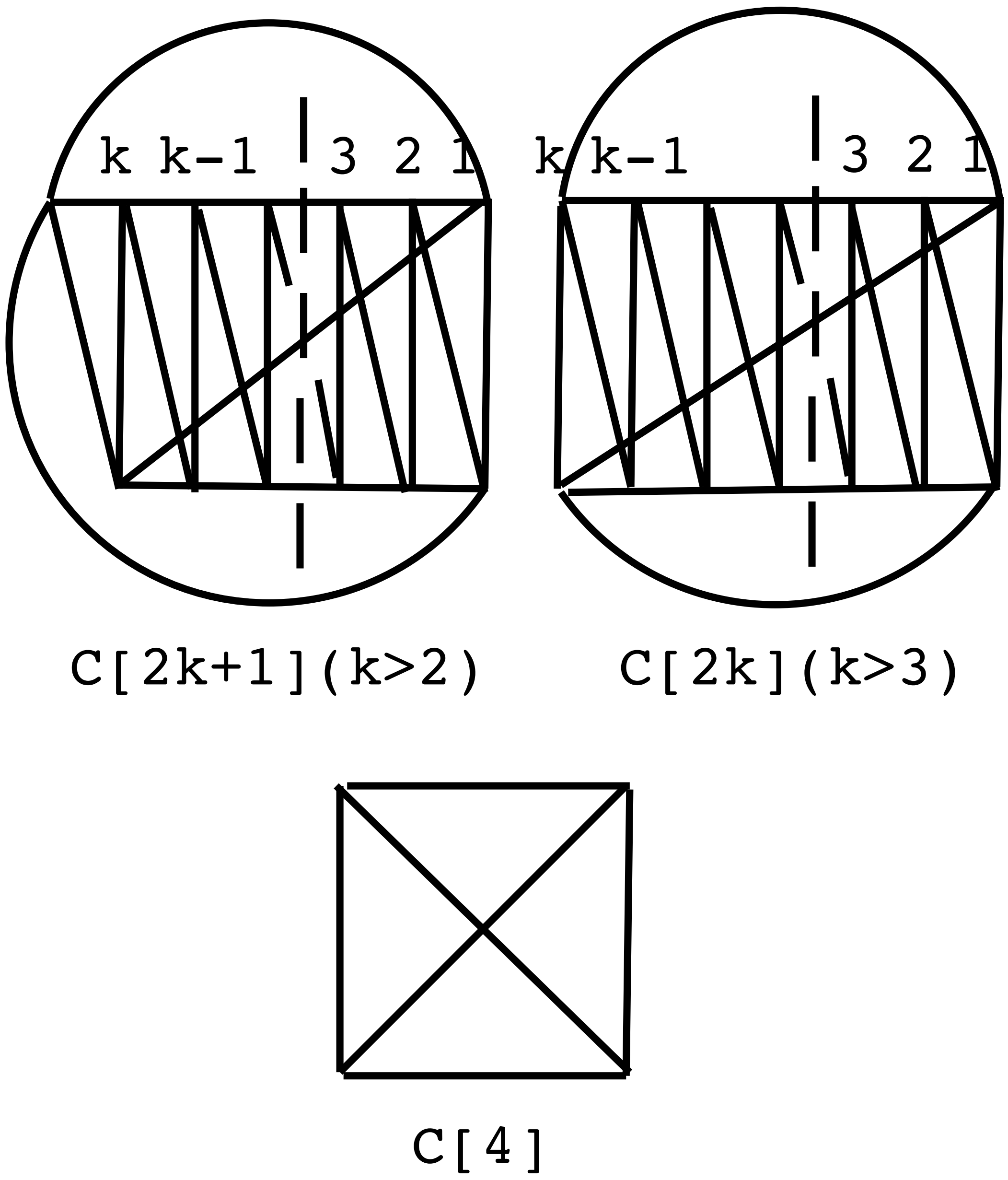
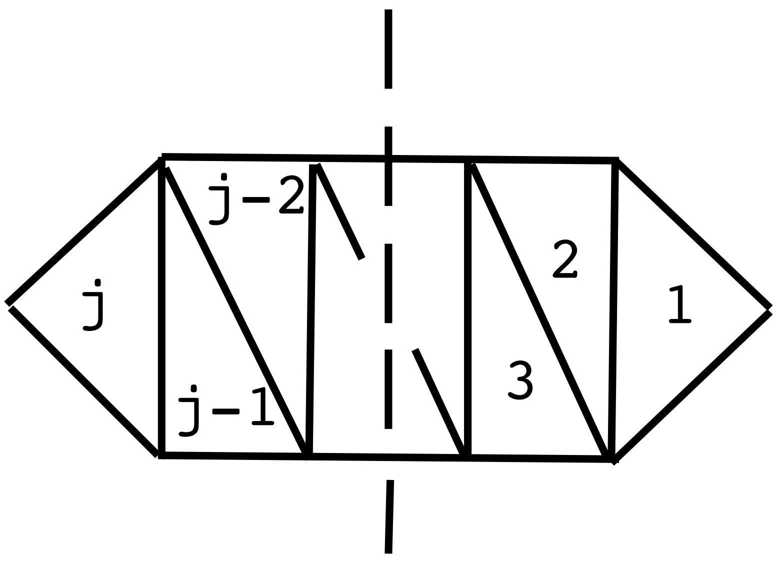


FIGURE 2. The graphs C_i with $i \geq 3$

Now we may assume that $T(G)$ is triangle-free, as well as 4-clique free. Therefore $\Delta(T(G)) \leq 3$, because if some triangle in G shared its edges with four other triangles, then one of its edges would be shared with two other triangles; G would contain G_{10} and hence $T(G)$ would contain $T(G_{10}) = K_3$.



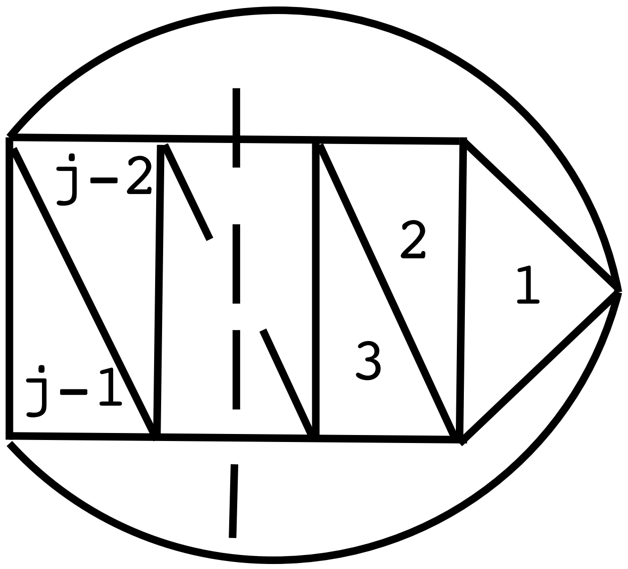


Figure 3. The graphs P_j and P'_j

We distinguish four cases corresponding to the possible values of $\Delta(T(G))$.

Case 1. $\Delta(T(G)) = 0$

In this case, $T(G) = K_1$. Therefore G contains exactly one triangle. By Lemma 3.2.6(b), it separates the cliques of G. The clique inseparable graph G must then be that triangle. Therefore $G = P_1$.

Case 2. $\Delta(T(G)) = 1$

In this case $T(G) = K_2$ as $T(G)$ is connected. Therefore G contains exactly two triangles and these share an edge. Therefore G contains P_2 , but G is clique inseparable and contains no triangles other than the two comprising P_2. Consequently $G = P_2$.

Case 3. $\Delta(T(G)) = 2$

Either $T(G)$ is a simple path or simple cycle in this case, because $T(G)$ is connected. It can be shown, by a careful examination of cases when $j \leq 5$ and by induction for $j \leq 5$ that if $T(G)$ is a simple path of length $j - 1$, then $G = P_j$ if $1 \leq j \leq 5$ and $G = P_j$ or P'_j if $j \leq 5$. Similarly, by a careful examination of cases when $j \leq 6$ and by induction when $j \leq 6$, it can be shown that if $T(G)$ is a simple cycle of length j , then $G = G_{10}$ if $j = 3$ and $G = C_j$ for $j = 4$ and $j \leq 6$. There is no G for which $T(G)$ is a 5-cycle or a 6-cycle.

Case 4. $\Delta(T(G)) = 3$

In this case, $T(G)$ contains the subgraph T_1 and G contains the subgraph G_1 of Fig. 2. We distinguish four subcases corresponding to the number of edges determined by the vertices A, B, C of degree 2 of G_1.

Subcase 4.0.

None of AB, BC, CA are edges of G. In this case, suppose a triangle XYZ of G had an edge XY in G_1. One of X, Y has degree 4 in G_1, therefore Z is in G_1 (

because $\Delta(G) \leq 4$). Therefore the arbitrary triangle XYZ of G is in G_1 and hence $G = G_1$ by clique inseparability of G. We also have $T(G) = T_1$.

Subcase 4.1.

Exactly one pair of the vertices $\{A, B, C\}$ of degree 2 of G_1 are adjacent in G. Suppose we label this pair AB. In this case, G_2 is a subgraph of G. If G_2 separates G's cliques then $G = G_2$ and $T(G) = T_2$. Otherwise some other triangle XYZ in G has an edge XY in G_2 and an edge XZ not in G_2. Both X and Y must be in $\{A, B, C\}$ as $\Delta(G) \leq 4$. Therefore $XY = AB$ and hence G_3 is a subgraph of G. This subgraph separates the cliques of G. To show that, let UVW be any triangle in G with edge UV in G_3. At least one of the vertices U, V has degree 4 in G_3. Therefore each edge of UVW is in G_3 and hence G_3 separates the cliques of G. By inseparability, $G_3 = G$ so $T(G) = T_3$.

Subcase 4.2.

Exactly two pairs of the vertices $\{A, B, C\}$ of G_1 are adjacent in G. Suppose these are AB and BC. In this case G_4 is a subgraph of G. If UVW is any triangle in G with an edge UV in G_4 then at least one end of UV has degree 4 in G_4 and hence $UVW \subseteq G_4$. Therefore G_4 separates the cliques of G, but G is clique inseparable so $G = G_4$ and $T(G) = T_4$.

Subcase 4.3.

The vertices of degree 2 in G_1 are in a triangle in G. In this case, G_5 is a 4-regular subgraph and hence separates the cliques of the clique inseparable graph G. Therefore $G = G_5$ and $T(G) = T_5$. This concludes the proof of Theorem 3.2.14.

Corollary 3.2.15. *If $\Delta(G) \leq 4$, then the only connected triangle graphs $T(G)$ are simple paths of all lengths $l \leq 0$, simple cycles of lengths $l \leq 3$, except $l = 5$ and $l = 6$, the graphs $T(G_j)$ for $1 \leq j \leq 9$ and $T(K_5)$.*

Initially, a subtotal for $cp(G)$ and $cc(G)$ is begun as $\epsilon(G)$, the number of edges in G. We set $Y = T(G)$. A procedure to determine the triangle graph of G is presented in [Pullman 1984] and it has complexity $O(n)$.

In each pass of the algorithm:

(a) a connected component X of Y is computed,

(b) the block B for which $T(B) = X$ is identified sufficiently to determine $cp(G), cc(G), \epsilon(G)$,

(c) $cp(G), cc(G)$ are added to the totals and $\epsilon(G)$ is subtracted from them both,

(d) Y is replaced by $Y - X$, the graph obtained by deleting all vertices and edges of X from Y.

This sequence is repeated until Y has no more vertices left. Thus a total of ω passes is performed, where ω is the number of connected components of $T(G)$ ($\omega = 0$ if $T(G) = \emptyset$). At the ith pass, a clique-block B_i is determined which contains a triangle of G. At the ωth pass the subtotal for $cp(G)$ is $\epsilon(G) + \sum_{i=1}^{\omega}(cp(B_i) - \epsilon(B_i))$ but $\epsilon(G) - \sum_{i=1}^{\omega} \epsilon(B_i)$ is the number of clique-blocks which are 2-cliques of G and hence contribute 1 each to $cp(G)$. Therefore the subtotal on $cp(G)$ showing at the last pass is $cp(G)$. Similarly for $cc(G)$.

There is an algorithm [**Aho,Hopcroft,Ullman 74**] which computes one by one, all the connected components of a graph on k vertices and m edges in $O(max(m, k))$ operations. Since the number of vertices of $T(G)$ is at most $\frac{4n}{3}$ and $T(G)$ has at most $3n$ edges (because $\Delta(T(G)) \leq 6$) it follows that the connected components of

$T(G)$ can be computed in linear time. At every pass, we interrupt the algorithm from [**Aho,Hopcroft,Ullman 74**] by a constant number of operations after it computes one connected component of a certain graph. It then resumes work on a smaller graph. It follows that the algorithm works in linear time.

Algorithm 1(computes $cp(G)$ and $cc(G)$ for graphs G with $\Delta(G) \leq 4$)

Step 1

Let $CP = CC = \epsilon(G)$ and $T = T(G)$.

Step 2

If $T = \emptyset$, then $cp(G) = CP, cc(G) = CC$, stop. Otherwise, let Y be a connected component of T.

Step 3

Let ν be the number of vertices in Y.

If $1 \leq \nu \leq 8$ recognize Y by using the properties of the graphs G_j, C_j, P_j and P'_j.

If $\nu = 10$ and $\epsilon = 30$, then subtract 29 from each CC and CP.

For all other ν

if $\epsilon(Y)$ is odd, subtract $\nu + 1$ from CC and $\nu + \frac{1-(-1)^\nu}{2}$ from CP

if $\epsilon(Y)$ is even, subtract ν from CC and $\nu - \frac{1-(-1)^\nu}{2}$ from CP

Step 4

Replace T by $T - Y$ (deleting the edges and the vertices of Y). Go to *Step* 2.

The corollary of the next lemma leads to a very simple procedure for determining $cp(G)$ and a minimal clique partition for G with $\Delta(G) \leq$ and 4-clique free.

LEMMA 3.2.16. *Suppose $\Delta(G) \leq 4$, t is a triangle in G, $N(t)$ is the family of triangles of G sharing edges with t and $H(t)$ is the subgraph of G generated by $N(t)$. Let $d(t)$ be the degree of t as a vertex of $T(G)$.*

(a) If $d(t) \leq 1$, then t is deletable.

(b) If $d(t) = 2$ and $d(t') = 3$ for some t' in $N(t)$, then t is deletable.

(c) If $d(t) = 2$ and $d(t') = 2$ for all t' in $N(t)$, then the clique-block of G containing $N(t)$ is G_{10}, G_2, P_j, P'_j or C_j for some j.

(d) If $d(t) = 3$, then $H(t)$ is deletable. Suppose $N(t)=\{t_1, t_2, t_3\}$. If t_1 shares an edge of t_2, then $cp(H(t)) = 1$. Otherwise $cp(H(t)) = 3$.

(e) If $d(t) = 4$, then $H(t)$ is deletable and $cp(H(t)) = 3$.

(f) If $d(t) = 6$, let $t' \in N(t)-\{t\}$. If $d(t') = 6$, then $H(t) \cup H(t')$ is deletable and its clique partition number is 1. Otherwise $H(t)$ is deletable and $cp(H(t)) = 4$.

Proof:

(a) If $d(t) = 0$, then t is a clique-block by Lemma 3.2.6. If $d(t) = 1$, let L denote the clique-block of G containing t.

By Figs. 1, 2 and 3, $L = P_j$ or P'_j for some $j \leq 1$ or $L = G_j$ for some $1 \leq j \leq 3$. In each case, t is a part of a maximal independent set of $T(L)$. Therefore, by Lemmas 3.2.8 and 3.2.11, t is deletable from G.

(b) According to Fig 3 $H(t) = G_i$ for $i = 2, 3$ or 4. In each case, t is a part of a maximal independent subset of $T(H(t))$. Therefore, by Lemmas 3.2.8 and 3.2.11, t is deletable.

(c) The connected component of t in $T(G)$ is T_2 or a path or cycle by Figs. 1, 2 and 3.

(d) According to Figure 3, $H(t) \subseteq G_i$ for some $1 \leq i \leq 7$.

If $H(t) \subseteq G_i$, then $\{t_1, t_2, t_3\}$ is independent in $T(G)$ if $1 \leq i \leq 5$ and $N(t)$ is a 4-clique in $T(G)$ if $i = 6$ or 7.

If t_1 is adjacent to t_2 in $T(G)$, then $H(t) = G_6$ and $cp(G_6) = 1$. Otherwise $\{t_1, t_2, t_3\}$ is part of a clique partition of $G_i (i \leq 5)$ by Lemma 3.2.11, therefore $t_1 \cup t_2 \cup t_3 = H(t)$ is deletable from G_1 and hence from G by Lemma 3.2.8.

Moreover $H(t) = G_1$ so $cp(H(t)) = 3$.

(e) According to Figure 3.3, $H(t) = G_7$ because neither T_8 nor T_9 contain vertices of degree 4 adjacent to all other vertices of degree 4. We have $cp(H(t)) = cp(G_7) = 7$.

(f) First note that every triangle in K_5 shares an edge with six other triangles.

Therefore the triangle graph of K_5 is 6-regular. Theorem 3.2.14 and the definitions of P_j, P'_j and C_j imply that K_5 is the only clique inseparable graph whose triangle graph is 6-regular. By Figs 3.1, 3.2 and 3.3, G_9 is the only clique inseparable graph whose triangle graph has exactly one vertex of degree 6.

Both graphs are deletable by Lemma 3.2.8.

Corollary 3.2.17. *If $\Delta(G) \leq 4$ and G contains no 4-cliques, then every triangle of G of minimal degree in the triangle graph of G is deletable.*

Algorithm 1'(computes $cp(G)$ when $\Delta(G) \leq 4$ and G has no 4-cliques)

Step 1

Let $T = T(G)$ and $C = \epsilon(G)$.

Step 2

If $T = \emptyset$, then $cp(G) = C$. Stop.

Step 3

Let t be a vertex of T of minimal degree. t is deletable from G, by the previous corollary.

Substract 2 from C because t contributes 1 to $cp(G)$ and 3 to $\epsilon(G)$.

Replace T by $T - N(t)$. The graph $T - N(t)$, obtained by deleting the vertices of the neighborhood in T of t (along with every edge incident with them) from T, is $T(\overline{t(G)}$. Go to Step 2.

3.3. Polynomial algorithm for CC(5)

In this section, we will present a polynomial time algorithm for solving **CC**(5) which was constructed in [**Hoover 92**].

We want to construct a minimum clique cover successively adding cliques to a partial cover.

Let H be a subgraph of G.

A *G-clique cover of H* is a collection of cliques of G which together contain all edges of H. $cc_G(H)$ is the minimum cardinality of a G-clique cover of H.

An *H-eligible clique of G* is either

(a) a maximal clique of G which contains at least two edges of H or

(b) an edge of H which is contained in no clique of the kind described in (a).

An edge of H is *H-exposed* if it belongs to only one H-eligible clique. A clique of G is *H-exposed* if it is H-eligible and contains an H-exposed edge. Υ_H is the class of all H-exposed edges.

The set of interior edges of H, $int(H)$, consists of those edges of H which belong to no H-exposed cliques.

Of course, a clique cover of G is just a G-clique cover of G, hence $cc(G) = cc_G(G)$.

The following lemma is clear.

Lemma 3.3.1. *If C is a minimum G-clique cover of int(H), then $C \cup \Upsilon_H$ is a minimum G-clique cover of H.*

The Lemma 3.3.1 shows how to begin an algorithm to find a minimum clique cover C of G.

```
$C \leftarrow \emptyset; H \leftarrow G$
Repeat
$C \leftarrow C \cup \Upsilon_H;$
$H \leftarrow int(H);$
Until $(H = int(H))$
```

Since the loop is executed at most $\epsilon(G)$ times and each execution takes polynomial time, this part of the algorithm takes polynomial time. After this, we are left with a subgraph H of G which has no H-exposed edges. Thus each edge of H belongs to at least two H-eligible cliques, but not to any G-exposed clique since $H \subseteq int(G)$. As any H-eligible clique which consists of a single edge would be exposed, every H-eligible clique must be a maximal clique of G which contains at least two edges of H.

By Lemma 3.3.1, we can finish the algorithm by finding a minimum G-clique cover for H. This is done in the following way.

Let $elig(H)$ be the graph whose vertices are H-eligible cliques, two cliques being adjacent if they share an edge of H. Since a minimum G-clique cover of H corresponds to a collection of minimum G-clique covers of the $elig(H)$-components of H, a minimum cover may be found $elig(H)$-component by component. Hence we may assume that H is $elig(H)$-connected.

We will show that H has no more than some fixed number of edges or else

(1) every edge of H is contained in exactly two H-eligible cliques

and

(2) no H-eligible clique is adjacent to more than two other H-eligible cliques.

By (1) a minimum cover of H by eligible cliques is the same as a minimum vertex cover of $elig(H)$. By (2), except in small components of $elig(H)$, no vertex of $elig(H)$ has degree more than 2. Hence one can find a minimum vertex cover in polynomial time.

The following sequence of lemmas will establish (1) and (2), thereby completing the polynomial algorithm for **CC**(5).

LEMMA 3.3.2. *Any K_4 of G which contains an edge of int(G) is contained in a clique component of G which has at most 17 edges.*

Proof:

Since $\Delta(G) \leq 5$, it is easy to see that any K_5 of G either has an exposed edge or else is contained in a K_6, in which case all edges are exposed. Hence any K_5 of G contains no edge of $int(G)$.

On the other hand it follows from looking at cases that any K_4 *abcd* which is a maximal clique of G and contains an edge of $int(G)$ must be contained in a clique component isomorphic to the one in Figure 4 which has 17 edges.

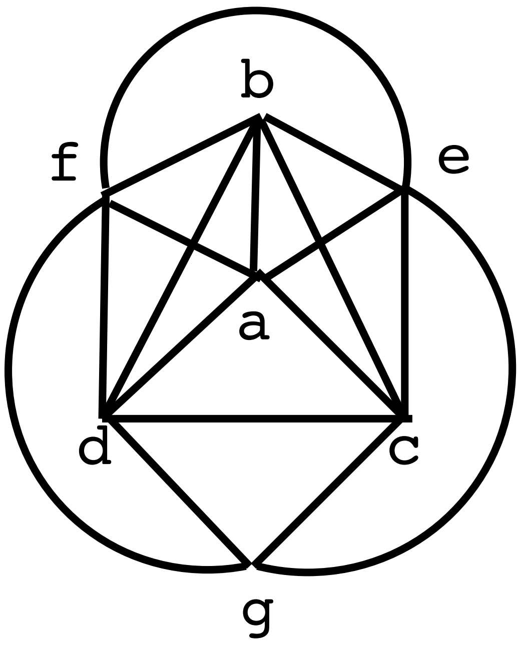


FIGURE 4. A maximal K_4 in G

Lemma 3.3.2 implies that except for small clique components of G, all H-eligible cliques are triangles.

LEMMA 3.3.3. *Suppose ab is an edge of int(G) which is contained in at least three maximal cliques of G. Then ab belongs to a clique component of G which has at most twenty edges.*

Proof:

By the preceding lemma, we may take the maximal cliques containing ab to be triangles abc, abd and abe. Since $ab \in H$, the triangles are not G-exposed. As we already have four edges incident at a and b, we must have the situation in Figure 5 $f \neq g$, since otherwise abc would not be a maximal clique.

Now every vertex in the diagram has current degree at least 4, so any triangle in the clique component of G of ab contains one of the edges in the diagram. As the

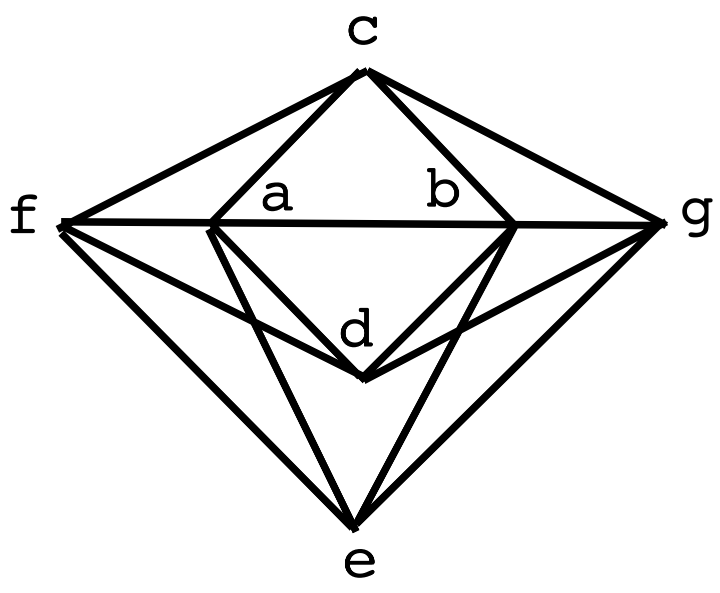


FIGURE 5. An edge of H contained in three triangles of G

diagram has 15 edges and 5 vertices which may each be incident to one more edge(a and b cannot be incident to any more edges), there can be at most 20 edges in the clique component.

Since H has no exposed edges, Lemma 3.3.3 implies that except on small $elig(H)$-components of H, each edge of H belongs to exactly two H-eligible cliques. That is, (1) holds.

Lemma 3.3.2 and 3.3.3 also imply that, except on small components, any H-eligible clique of which shares an edge with each of three other H-eligible cliques must be a triangle of H.

Thus the following lemma establishes (2).

LEMMA 3.3.4. *Suppose that no K_4 of G contains an edge of H and no edge of H is contained in more than two cliques of G. Let abc be a triangle of H which is adjacent* *to three H-eligible triangles. Then each vertex of H which is connected to abc has H-distance at most 4 from the nearest vertex of abc.*

Proof:

We begin by investigating the structure of G near abc.

First, each edge of abc is contained in exactly one other clique of G since abc is not exposed. These other cliques must be adjacent H-eligible triangles. Since these triangles are in fact adjacent(i.e. share an edge of H with abc), each edge ab, bc, ca must be one of those edges and hence must be in H.

Let the adjacent triangles be abe, acd and bcf. d, e, f are distinct since otherwise abc would be contained in a K_4.

As these triangles contain edges of H, none of these edges can be exposed.

It follows that G contains one of the configurations in Figure 6. We assume that we have the first configuration. For the others the proof is similar.

Since the vertices d through i are the only vertices of G adjacent to abc, hence the only vertices which may be adjacent in H, it suffices to prove that any vertex which is connected by an H-path to one of the vertices d, e, f, g, h, i is connected by an H-path of length at most two.

Suppose, then, that this is not the case. Up to isomorphism, we may assume that we have a vertex l which has H-distance exactly three from either g or e.

The following simple observations will be used several times.

1) No H-eligible clique has a G-exposed clique.

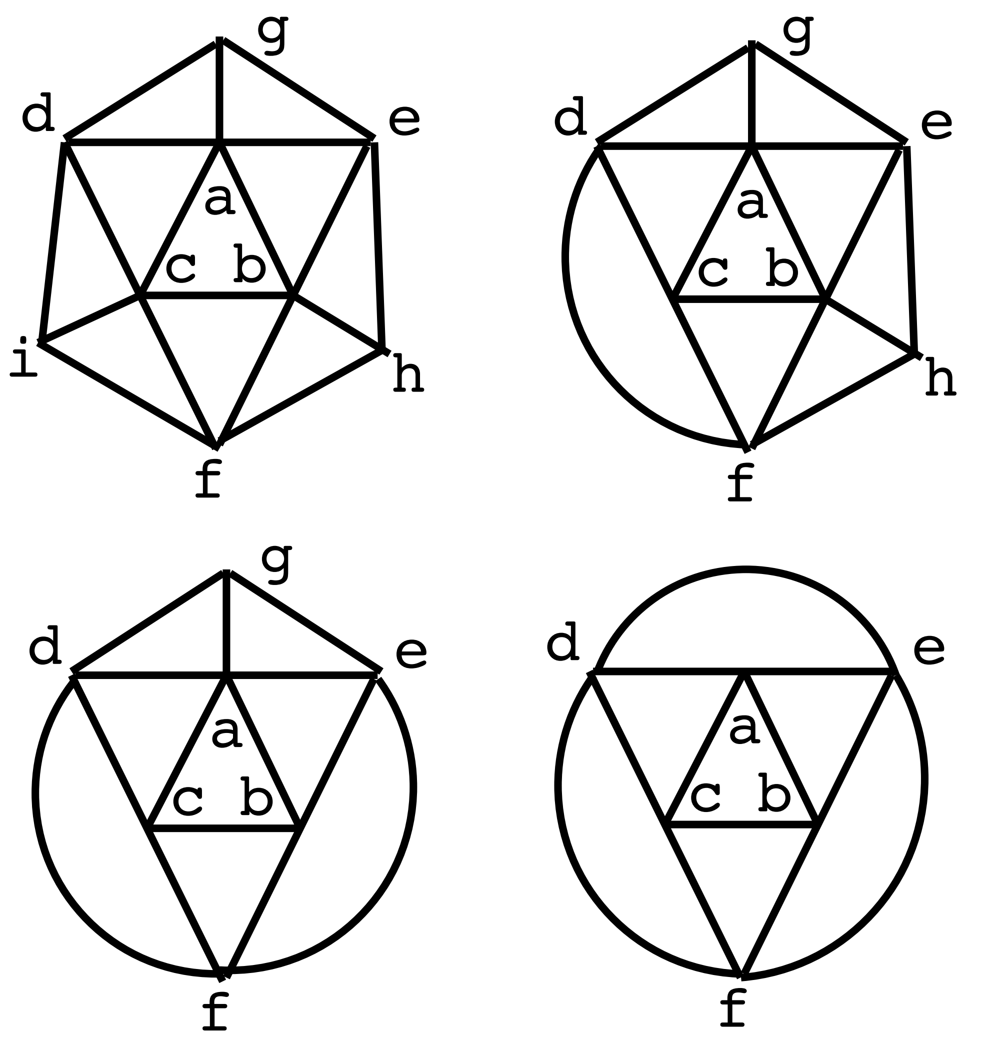


Figure 6. Neighborhood of a triangle of H adjacent to three eligible triangles

2) No H-eligible clique containing l contains any of the vertices d, e, f, g, h, i since there would be an H-path of length at most two from l to that vertex.

Case 1: The shortest H-path from l to a vertex d, e, f, g, h, i is $P = (g, j, k, l)$.

As the current degree of g is 4, one of the two H-eligible cliques containing gj must be gjd or gje, say gjd. Now jd must not be G-exposed, hence there must be an edge ji. Now the current degree of j is 4, so one of the two H-eligible cliques containing jk must be either jki or jkg.

Suppose it is jki. As ki must not be G-exposed, there must be a triangle kfi. Now the current degree of k is 4, so the two eligible cliques containing kl must be jkl and klm. m is a new vertex by observation 2). But k and all vertices adjacent to it but l and m are full, so km must be exposed, impossible since klm contains an edge of H.

On the other hand, if one of the H-eligible cliques containing jk is gjk, then as gk is not G-exposed, k must be adjacent to e. Again, the two H-eligible cliques containing kl must be klm and jkl and again km must be exposed, contradiction.

The subcase where one of the triangles containing jk is gjk, is handled similarly, as in Case 2, where the shortest H-path from l to a vertex $d, \ldots, i$ is $P = (e, j, k, l)$.

Bibliography

[Aho,Hopcroft,Ullman 74] A.V. Aho, J.E. Hopcroft, J.D. Ullman *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass. 1974

[Bondy,Murty 77] J.A.Bondy, U.S.R.Murty *Graph theory with applications* MacMillan, London, 1977

[deCaen,Pullman 81] D. deCaen, N.J.Pullman *Clique covering of graphs I: clique-partitions of regular graphs* Utilitas Math., 19(1981), pp.177-206

[Erdos,Goodman,Posa 66] P.Erdos, A.W.Goodman, L.Posa *The representation of a graph by set intersections*, Canadian J.Math., 18(1966), pp.106-112

[Garey,Johnson 79] M.R.Garey, D.S.Johnson *Computers and Intractability* 1979, W.H.Freeman, San Francisco

[Garey,Johnson 76] M.R.Garey, D.S.Johnson *The complexity of near-optimal graph coloring* J.ACM 23, 1(Jan.1976), pp.43-49

[Garey,Johnson,Stockmeyer 76] M.R.Garey, D.S.Johnson, L.J.Stockmeyer *Some simplified NP-complete graph problems* Theoretical Computer Science, Volume 1(1976), pp.237-267

[Harary 68] F.Harary *Graph theory* Addison-Wesley, Reading, MA, 1968

[Holyer 81] I.Holyer *The NP-completeness of some edge-partition problems* SIAM J. Comput., Volume 10(1981), Number 4, pp.713-717

[Hoover 92] D.Hoover *Complexity of graph covering problems for graphs of low degree* The Journal of Combinatorial Mathematics and Combinatorial Computing, Volume 11(1992), pp.187-208

[Johnson 1,74] D.S.Johnson *Approximation algorithms for combinatorial problems* J.Comptr.Syst.Sci. 9(1974), pp.256-278

[Johnson 2,74] D.S.Johnson *Worst case behavior of graph coloring algorithms* Proc. Fifth Southeastern Conf. of Combinatorics, Graph Theory and Computing, Utilitas Mathematica Pub., Winnipeg, Canada, 1974, pp.513-528

[Karp 75] R.M.Karp *On the computational complexity of combinatorial problems* Networks 5(1975), pp.45-68

[Karp 72] R.M.Karp *Reducibility among combinatorial problems* Complexity of Computer Computations 1972, Plenum Press, New York, pp.85-103

[Kellerman 78] E.Kellerman *Determination of keyword conflict* IBM Tech. Disclosure Bull. 16(1978), pp.544-546

[Kou,Stockmeyer,Wong 78] L.T.Kou,L.J.Stockmeyer,C.K.Wong *Covering edges by cliques with regard to keyword conflicts and intersection graphs* Communications of the ACM, Volume 21(1978), Number 2, pp.135-139

[Kratzke,Reznick,West 88] T.Kratzke,B.Reznick,D.West *Eigensharp graphs: Decomposition into complete bipartite subgraphs* Trans. of Amer. Math. Society, Volume 308(1988), Number 2, pp.637-653

[Orlin 77] J.Orlin *Contentment in graph theory: Covering graphs with cliques* Indagationes Math. 39(1977), pp.406-424

[Pullman 84] N.J.Pullman *Clique covering of graphs IV. Algorithms* SIAM J. Comput., Volume 13(1984), No. 1, pp.57-75

[Szpilrajn-Marczewski 45] E.Szpilrajn-Marczewski *Sur deux properietes des classes d'ensembles* Fund.Math.33, pp.303-307(1945)