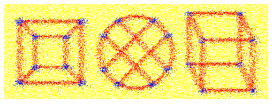
Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 10, no. 2, pp. 159–190 (2006)

A Survey of the Algorithmic Properties of Simplicial, Upper Bound and Middle Graphs

Grant A. Cheston *Tjoen Seng Jap*

Department of Computer Science
University of Saskatchewan
http://www.cs.usask.ca/
cheston@cs.usask.ca

Abstract

Three classes of graphs, simplicial, upper bound, and middle graphs, have been known for some time, but many of their algorithmic properties have not been published. The definitions for these graph classes are reviewed, and their relationships with other common graph classes (especially line and perfect graphs) are presented. Efficient algorithms are referenced or outlined to recognize each class of graphs. Finally, for each class of graphs and most of the common parameters of graphs, either an algorithm or an NP-complete result is presented, or it is referenced in the literature.

Article Type	Communicated by	Submitted	Revised
survey paper	A. Gibbons	May 2005	April 2006

1 Introduction

Many interesting properties of graphs are NP-hard to compute for general graphs [17]. As a result, many special classes of graphs have been studied with the objective of finding classes that appear in actual applications, but are sufficiently specialized that various properties can be efficiently computed. Many of these classes are subclasses of the class of perfect graphs [7]. In this paper, we consider some classes of graphs where many of the graph properties can be efficiently computed, but they are not subclasses of the class of perfect graphs. These classes of graphs have been known for some time, especially middle graphs [22]. However, many of the relationships amongst them and many of their algorithmic properties are first presented in this paper. Table 1 summarizes the algorithmic status of each of the properties. In the table, NP-C stands for NP-complete, ISO-C stands for isomorphism-complete, ?? indicates that the complexity of the problem is unknown, and a function of n, e, s and/or p expresses the worst case time complexity of the problem. Note that for the two most general classes, simplicial graphs and line graphs, they each have polynomial algorithms for only 6 of the 20 problems. Of these, three problems are polynomial for both classes. On the other hand, for the most restricted graph class, middle graphs, 14 problems are known to be polynomial. The rest of the paper defines the graph classes, describes the relationships between the classes, and justifies the entries in the table.

2 Classes of Graphs

Unless otherwise specified, the graphs considered will be undirected and have no self-loops or multiple edges, i.e., they are simple graphs. If self-loops and/or multiple edges are permitted, the graph will be called a pseudograph. Let $V(G)$ denote the vertex set of graph G, $E(G)$ the edge set, $e = |E(G)|$, and $n = |V(G)|$. Also for a vertex u and a subset S of the set of vertices, the closed neighbourhood sets are given by

$$N[v] = \{v\} \cup \{u \mid u \text{ is adjacent to } v\},$$

$$N[S] = \cup_{v \in S} N[v],$$

and $<S>$ is the subgraph of G induced by the set S. For our algorithms, we assume that the graph representation includes for each vertex a linked list of adjacent vertices (its neighbours).

A graph is said to be complete if every pair of distinct vertices is joined by an edge. Any maximal complete subgraph of a graph is called a clique. A simplicial (unicliqual) vertex is a vertex that appears in exactly one clique. A clique containing one or more simplicial vertices is called a simplex. Note that if u is a simplicial vertex, then $<N[u]>$ is the unique clique (simplex) containing u. There are at most n simplices in a graph as each simplex contains a (simplicial) vertex not in any other simplex. Of course, a general graph might not have any simplicial vertices.

Table 1: Algorithmic status of each property for each graph class

	simplicial	upper bound	rep. of hereditary hypergraph	middle	line
membership	$n + s * e$	$n + s * e$	$n + s * e$	$n + e$	$n + e$
clique cover	NP-C	$n + e$	$n + e$	$n + e$	$n + e$
vertex clique cover	$n + e$	$n + e$	$n + e$	$n + e$	NP-C
independent set	$n + e$	$n + e$	$n + e$	$n + e$	$p^{0.5} * n + e$
vertex cover	$n + e$	$n + e$	$n + e$	$n + e$	$p^{0.5} * n + e$
maximum clique	NP-C	NP-C	??	$n + e$	$n + e$
chromatic number	NP-C	NP-C	??	n^2	NP-C
chromatic index	??	??	??	??	NP-C
dominating set	NP-C	NP-C	NP-C	$n^{1.5} + e$	NP-C
edge dominating set	$n^{0.5} * e$	$n^{0.5} * e$	$n^{0.5} * e$	$n^{0.5} * e$	NP-C
maximum minimal dominating set	$n + e$	$n + e$	$n + e$	$n + e$	NP-C
irredundant set	NP-C	$n + e$	$n + e$	$n + e$	NP-C
minimum dominating independent set	NP-C	NP-C	NP-C	$n^{1.5} + e$	NP-C
minimum maximal irredundant set	NP-C	NP-C	NP-C	$n^{1.5} + e$	NP-C
hamiltonian path	NP-C	NP-C	NP-C	NP-C	NP-C
induced path	NP-C	NP-C	NP-C	NP-C	NP-C
graph isomorphism	ISO-C	ISO-C	ISO-C	ISO-C	ISO-C
subgraph isomorphism	NP-C	NP-C	NP-C	NP-C	NP-C
steiner tree	NP-C	NP-C	NP-C	NP-C	NP-C
simple maximum cut	NP-C	??	??	$n + e$	$n + e$

n = number of vertices in the graph e = number of edges in the graph
s = number of simplices in the graph p = number of vertices in the root graph
NP-C stands for NP-complete ISO-C stands for isomorphism-complete

A graph is called simplicial [11] if every vertex is contained in a simplex, i.e., every vertex is either simplicial or adjacent to a simplicial vertex. Similarly, a graph is called edge simplicial if every edge is contained in a simplex. It turns out that the edge simplicial class is the same as the class of upper bound graphs. If $(P, \leq)$ is a partially ordered set (poset), the upper bound graph [31] of P, $G = (V, E)$, is defined as follows: $V = P$, and $(x, y) \in E$ if and only if $x \neq y$ and there exists a $z \in P$ such that $x \leq z$ and $y \leq z$. The proof of equivalence of these two classes [11] is based on the simplicial vertex of a simplex corresponding to the greatest upper bound of the same items in the poset.

If instead of a partial order on P, there is a quasi-order (an irreflexive and transitive relation) on P, then instead of obtaining an upper bound graph, a strict upper bound graph [31] is obtained. For the quasi-order that corresponds to a given partial order, the edges of the strict upper bound graph are a strict subset of those of the upper bound graph. The strict upper bound class is a subclass of the class of competition graphs [41] that has been used in food web models of ecosystems. However, the strict upper bound and competition graph classes are incomparable with the class of upper bound graphs as each has members not in the upper bound class and vise versa.

Of course, the simplicial class includes all upper bound (edge simplicial) graphs. We also consider classes of graphs that are contained in the upper bound class. We begin by recalling that a hypergraph [3], $H = (V, E)$, consists of a set of vertices V and a collection of edges E, where each edge is an arbitrary non-empty subset of V. If the sets in E are distinct, then H is called a simple hypergraph. From each hypergraph H, we can obtain a graph, $\Omega(H)$, called the representative graph of H. $\Omega(H)$ is the graph where its vertices are the edges of H, and two vertices are adjacent in $\Omega(H)$ if they have a non-empty intersection (as sets of V). Hence, $\Omega(H)$ is the intersection graph of the edge set of H. Not only is $\Omega(H)$ a graph, but every (simple) graph is the representative graph of some hypergraph. However, we are interested in a subclass of the upper bound class. To get such a subclass, we need to consider hereditary hypergraphs [14]. A hypergraph $H = (V, E)$ is called hereditary if $e_1 \in E$, $e_2 \subseteq e_1$, and $e_2 \neq \emptyset$, then $e_2 \in E$.

Lemma 1
The representative graph of any hereditary hypergraph is an upper bound graph.

Proof:
Let HH be an arbitrary hereditary hypergraph, and $G = \Omega(HH)$.
Let $e = (x, y)$ be an arbitrary edge in G.
Then x and y are edges in HH with $x \cap y \neq \emptyset$. Let $u \in x \cap y$.
Since HH is hereditary, $\{u\}$ is an edge in HH, and hence a vertex in G.
In fact, $\{u\}$ is a simplicial vertex in G, and e belongs to $\{u\}$'s simplex.
Therefore, G is edge simplicial, and thus upper bound. □

To see that representative graphs of hereditary hypergraphs are a proper subclass of upper bound graphs, note that $K_{1,3}$ is an upper bound graph but not the representative graph of any hereditary hypergraph.

Of course, treating an edge as a set containing a pair of vertices, every graph $G = (V, E)$ is a hypergraph so that hypergraphs are a generalization of graphs. Considering G as a hypergraph, the graph $\Omega(G)$ is defined. $\Omega(G)$ is commonly known as the line (or edge) graph of G and denoted by $L(G)$. Also, G is called the root graph of $L(G)$. Although a simple graph is not a hereditary hypergraph, there is a natural hereditary hypergraph $HH(G)$ associated with every graph when every vertex is added to the edge collection as a singleton set. Thus for every graph G, we can obtain the graph $\Omega(HH(G))$. It turns out [14] that $\Omega(HH(G)) \approx M(G)$, where $M(G)$ is the middle graph of G, and $\approx$ is graph isomorphism (see section 4.12). For a graph $G = (V, E)$, the middle graph of G [22] is given by $M(G) = (V \cup E, E')$ where two vertices of $M(G)$ are adjacent if they are incident in G, i.e., either they are two edges of G that share a vertex, or one is an edge of G and the other is a vertex that is one end of the edge. From $\Omega(HH(G)) \approx M(G)$, it follows that the class of middle graphs is a subclass of the representative graphs of hereditary hypergraphs. Let G^+ be the graph obtained by adding to G n new vertices and n new edges where each new edge joins a distinct new vertex to a distinct vertex of G. Then it can be shown [22] that $M(G) \approx L(G^+)$. Thus, the class of middle graphs is a subset of the class of line graphs.

There are other characterizations of middle graphs. Let $S(G)$ be the subdivision graph of G, i.e., the graph obtained by inserting a vertex into every edge of G. Now the middle graph of G can be obtained by adding the edges of $L(G)$ into $S(G)$ [14]. Alternately, the middle graph of G can be obtained by removing the edges of G from the total graph of G [2, 23]. A final characterization of middle graphs is given by the following result:

Lemma 2 *[43]*
A graph G is a middle graph if and only if there exists simplices $S_1, S_2, ..., S_s$ of G satisfying the following conditions:
(i) $|S_i \cap S_j| \leq 1$, for $i \neq j$
(ii) $S_1, S_2, ..., S_s$ is a partition of $E(G)$, and
(iii) every non-simplicial vertex lies in exactly two simplices.

By analyzing $M(G)$, it is clear that the vertices of a simplex of $M(G)$ correspond to a vertex of G and its incident edges. In particular, if we associate a specific simplicial vertex of a simplex S of $M(G)$ with vertex v of G, then the non-simplicial vertices of S must be edges incident to v in G and other simplicial vertices of S must be self-loops about v in G. Thus, if self-loops are not allowed in the root graph, in this lemma there should be a fourth condition:

(iv) every simplex contains only one simplicial vertex.

Henceforth, we will assume that this condition is included in this characterization of middle graphs. If self-loops and multiple edges are permitted in the root graph, the class will be called pseudomiddle. Similarly, a pseudoline graph is the line graph of a pseudograph.

As a point of comparison, an alternate characterization of line graphs [28] is that a set of complete subgraphs can be found which partitions the edges and no vertex belongs to more than two of the subgraphs. For a middle graph, each complete subgraph must in addition be simplicial with exactly one simplicial vertex.

In addition to middle graphs, there is another known subclass of representatives of hereditary hypergraphs. Given a graph $G = (V, E)$, let I be the set of all independent sets in graph G. Then $H = (V, I)$ is an hereditary hypergraph whose representative graph $I(G)$ is called the independence graph of G. The set of all such graphs [13] is called the class of independence graphs.

The middle graph of the hypergraph $H = (V, E)$ is defined in [6] to be the intersection graph $\Omega(F)$, where

$$F = V' \cup E, V = \{v_1, v_2, ..., v_n\}, \text{ and } V' = \{\{v_1\}, \{v_2\}, ..., \{v_n\}\}$$

Naturally, if the hypergraph corresponds to a simple graph, i.e., it has no duplicate edges and each edge has size 2, this results in the standard middle graph of the graph. In the more general case, the class of middle graphs of hypergraphs is the same as the class of upper bound graphs. This follows from a theorem in [6] that G is the middle graph of a hypergraph iff G has an independent set of vertices such that the collection of closed neighbourhoods of these vertices forms a set of complete subgraphs which covers the edges of G. It is easy to see that the closed neighbourhoods are the simplices of the upper bound graph, and the independent set consists of one simplicial vertex per simplex.

In order to put the graph classes being studied in a context, some further subclasses of simplicial graphs will be mentioned. The class of graphs where each vertex belongs to exactly one simplex, call them vertex unisimplicial graphs, have been studied in the literature [38]. (Note that in the community studying well-covered graphs, these graphs are called simplicial.) Obviously, vertex unisimplicial graphs are simplicial, but they are neither upper bound nor split unless they are disconnected or complete. It might seem that this class of graphs would have a very restricted structure, but for any graph G, G^+ is vertex unisimplicial. The edge unisimplicial graph class, where each edge belongs to exactly one simplex, can also be defined. These graphs are upper bound and contain middle graphs. However, they do not contain pseudomiddle graphs as multiple edges lead to edges in two simplices.

A class of graphs that has many polynomial algorithms is the class of perfect graphs [3, 7]. The classes considered here are only weakly related to perfect graphs. The line graph of a bipartite graph is both a perfect graph [7, 27] and a line graph. Also, the subclass of perfect graphs called split graphs [21] is a subclass of simplicial graphs. To see this, recall that a graph is called split if its vertex set can be partitioned into two sets so that one set induces a complete graph and the other set is an independent set. As a result, each vertex of the independent set is a simplicial vertex. Also each vertex of the complete set is either incident to a vertex of the independent set (a simplicial vertex), or else its closed neighbourhood is the complete set of the partition so that it is itself a simplicial vertex.

Another class of perfect graphs related to the classes studied here is the class of hereditary upper bound graphs. A graph is a hereditary upper bound graph if every induced subgraph is an upper bound graph [34, 30]. An equivalent specification for this class is (cograph ∩ chordal) or (P_4, C_k)-free for $k \geq 4$ [34, 20]. Thus, the class is chordal [21] and perfect.

Figure 1 contains a diagram of the containment relationships amongst these classes of graphs. This paper will emphasize the algorithmic properties of middle, representatives of hereditary hypergraphs, upper bound, simplicial, and line graphs (in bold in the diagram).

3 Testing for membership

The most fundamental algorithm for a class of graphs is an algorithm to test whether a graph belongs to the class. In this section, we review the recognition algorithms for simplicial, upper bound, and middle graphs, and outline the algorithm for representative graphs of hereditary hypergraphs.

Recognition of simplicial graphs depends on the observation that within a simplex, every vertex of minimum degree is a simplicial vertex [11]. Thus, if the vertices of a graph are scanned in non-decreasing order by degree, the first vertex encountered of a simplex will be a simplicial vertex of the simplex. The removal of the simplex of a simplicial vertex results in another simplicial graph, so the process can be repeated until the graph is empty. The most time-consuming part is checking that the neighbourhood of each proposed simplicial vertex is complete. As a result, the time bound is $O(n + s * e)$, where s is the number of simplices.

Algorithms to recognize middle graphs and upper bound graphs (middle graphs of hypergraphs) are given in [44]. They also work by repeatedly removing the simplex whose simplicial vertex has the least degree, and checking the properties of the simplices. The version for middle graphs executes in linear time as each vertex is in at most two simplices.

To determine whether a graph belongs to the class of representative graphs of hereditary hypergraphs has the complication that the hereditary property needs to be tested. Suppose that $G = (V, E)$ is the graph to be tested, and our objective is to find an hereditary hypergraph HH such that $G = \Omega(HH)$. Before looking at the algorithm, we examine the relationships between G and HH. A set containing one element in HH becomes simplicial in G because all sets that contain the element must have a non-empty intersection. In addition, every simplicial vertex corresponds to a singleton set since non-singleton sets will be adjacent to singleton sets (which are not adjacent). Finally, because HH is hereditary, every element of every set must appear as a singleton set, and hence as a simplicial vertex in G. Thus finding the simplicial vertices of G finds the element set of HH. Moreover, the vertices of G adjacent to a simplicial vertex are exactly the sets of HH that contain the element associated with the simplicial vertex.

The algorithm first checks that the graph G is upper bound. While doing this, the vertices of HH are found (the simplicial vertices of G), and the edges

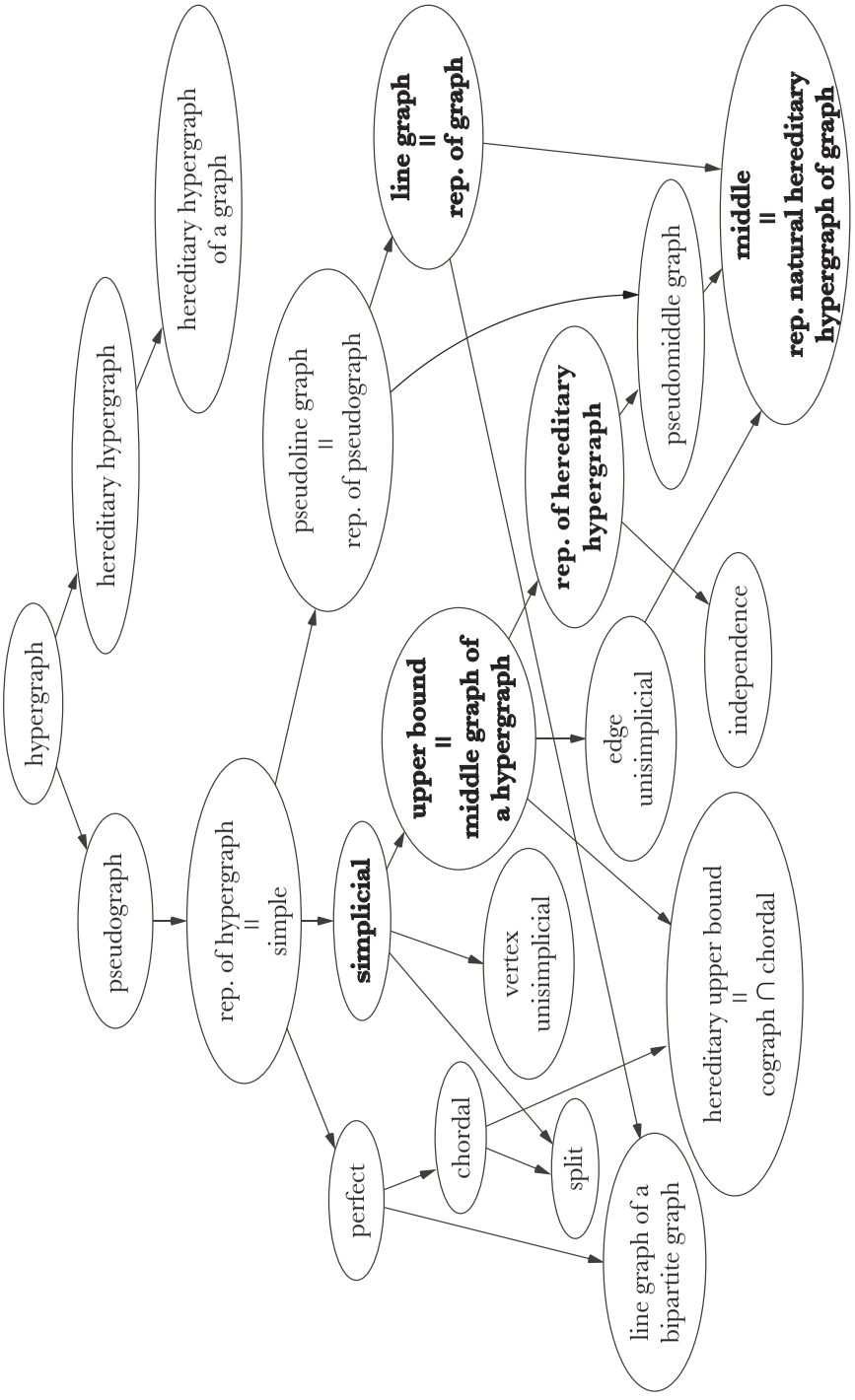

Figure 1: Containments of the graph classes

of HH are constructed (each edge of HH (a vertex of G) consists of the set of simplicial vertices of G incident to it (as a vertex of G)). Thus, G is the representative graph of the hypergraph HH, so it only remains to check that HH is hereditary. This is done by finding each maximal set of HH and then checking that all its non-empty subsets belong to HH. To do the latter task, a bit vector is set up so that it has one bit for each subset of the maximal set. Now in G, each vertex adjacent to the vertex corresponding to the maximal set is analyzed. If the set of HH corresponding to the adjacent vertex is a subset of the maximal set, its subset bit is marked. After analyzing all the adjacent vertices, it is necessary to check that all bits are marked. (Notes on algorithm notation: indentation indicates nesting, and { } encloses a comment.)

Algorithm: Test for representative graph of an hereditary hypergraph
Input: $G = (V, E)$
Output: A Boolean variable that indicates whether G is the representative of a hereditary hypergraph or not, and if it is, the representative hypergraph HH for G.

1. {check that G is upper bound graph and find the sequence of vertices for each edge set of HH}
   label each vertex of G with an empty sequence
   run the algorithm to test for an upper bound graph
      as each simplicial vertex is found
         label it with the next natural number i
         for every vertex in its closed neighbourhood
            append i to the end of the sequence for the vertex
2. {find the maximal edge sets of HH and check the hereditary property}
   initially all vertices of G are unmarked
   repeat as long as there is an unmarked vertex and no failure yet
      find an unmarked vertex u of G with the longest sequence
      mark u
      $r \leftarrow$ the length of the sequence for u
      if degree(u) $< 2^r - 2$
      then failure {u must be incident to a vertex for each subset of u's sequence (except $\emptyset$ and u's own sequence)}
      else set up a bit vector of 0's with length $2^r - 1$
         set the bit for u's own sequence to 1
         for each vertex v adjacent to u while not failure
            $k \leftarrow$ compute_bit(sequence of u, sequence of v)
            if k is 0
            then {the sequence of v is not a subsequence of u's}
            else set kth bit to 1, and mark the vertex v
         for each position k in the bit vector
            if kth bit is 0
            then failure {hereditary property fails as the vertex corresponding to this bit is not present in G, so its subset is missing from HH}

For the call compute_bit(A, B), A is the sequence corresponding to a maximal set, and B is another sequence. If B is not a subsequence (in the sense of an ordered set) of A, then 0 is returned. If B is a subsequence of A, then the algorithm returns the position of B's subsequence when the subsequences of A are listed in lexicographic order (if $A = <1, 2, 3>$, then the lexicographic ordering of its subsequences is $\{1\}, \{1, 2\}, \{1, 2, 3\}, \{1, 3\}, \{2\}, \{2, 3\}, \{3\}$).

Algorithm: Compute_bit
Input: sequences A, B stored in ascending order in arrays
Output: the position of the B sequence in the lexical ordering of the subsequences of A

1. bit_pos $\leftarrow$ 1, found $\leftarrow$ false, A_pos $\leftarrow$ 1, and B_pos $\leftarrow$ 1
2. while not found and bit_pos $\neq$ 0
   if A[A_pos] $\neq$ B[B_pos]
   then bit_pos $\leftarrow$ bit_pos + $2^{|A|-A\_pos}$
   else if B_pos = $|B|$
   then found $\leftarrow$ true
   else bit_pos $\leftarrow$ bit_pos+1, B_pos $\leftarrow$ B_pos+1
   A_pos $\leftarrow$ A_pos +1
   if A_pos > $|A|$
   then bit_pos $\leftarrow$ 0 {B[B_pos] does not exist in A}
3. return bit_pos

The time complexity for compute_bit is $O(|A|)$, i.e., $O(r)$. The first step of the test algorithm requires time $O(n+s*e)$ as we have already noted. The outer loop of Step 2 is done once for every maximal sequence, i.e., at most n times. Note that a bucket sort can be used to sort the vertices by sequence length in order $O(n)$ time before the start of the loop. Thus, the total time for all times through the loop to find the next largest unmarked vertex is $O(n)$. $O(2^r)$ time is needed to initialize the bit vector and to check it after the adjacency scan. But degree$(u) \geq 2^r - 1$, so degree u can be used to bound these two operations. The time for the scan of the adjacency list of u is degree u times $O(r)$ (for the call of compute_bit). But r is bounded by s, the number of simplices. Thus $O(n + s * e)$ is also a bound for step 2 and the whole algorithm.

To conclude this section, we note that there are well known linear algorithms [29, 42] to test for a line graph and determine its root graph.

4 Graph Properties

In this section, we discuss a number of graph properties. For each of these properties and for each of the graph classes, we justify the complexity for obtaining the property for graphs of the class.

4.1 Clique Covers

A clique cover of a graph is a set of cliques whose union is the whole graph, and a vertex clique cover (also called a partition into cliques) is a set of vertex-disjoint complete subgraphs such that the union of their vertex sets is the vertex set of the graph. The minimum-sized clique cover and minimum-sized vertex clique cover for graph G are denoted by $\Theta_e(G)$ and $\Theta_v(G)$ respectively. The key results related to these are the following:

Theorem 1 *[31]*
Upper bound graphs are exactly the graphs for which $\Theta_e(G) = \Theta_v(G)$.

Lemma 3 *[8]*
If $\Theta_e(G) = \Theta_v(G)$*, then the set of simplices forms the unique minimal collection of cliques which include all the vertices and edges of* G.

Thus for the class of upper bound graphs and all subclasses of it, $\Theta_e(G) = \Theta_v(G) = s$, the number of simplices, and this value is easily calculated via the recognition algorithm.

For simplicial graphs, the vertex clique cover problem is also easy. To cover all vertices by complete subgraphs, a subgraph is needed for every simplex (to cover its simplicial vertex). The set of simplices is clearly sufficient to cover the vertices, hence $\Theta_v(G) = s$. A vertex clique cover can be obtained from the recognition algorithm by selecting the subgraphs to be vertex-disjoint subgraphs of the simplices.

The clique cover problem for simplicial graphs is much more difficult – the decision version of the problem is NP-complete. To see this, as before let G^+ be the graph obtained by adding to each vertex of G an edge to a new degree 1 vertex. Then G^+ is a simplicial graph as each vertex is in the simplex formed by the new edge incident to it. Also it is easy to see that $\Theta_e(G^+) = \Theta_e(G) + n$ (where $n = |V(G)|$), since an additional clique is necessary and sufficient for each of the additional edges of G^+. Thus, it is as difficult to determine the clique cover for simplicial graphs as for graphs in general. Since the decision version of the clique cover problem is NP-complete for general graphs, it is also NP-complete for simplicial graphs.

The situation for clique covering of line graphs is opposite to that for simplicial graphs. The vertex clique problem is known to be NP-complete for line graphs [17]. On the other hand, once the root graph is known, we now show it is easy to determine the clique cover for a line graph. The graph $K_{1,i}$ is called a star and the center of it is the vertex of degree more than 1 (if $i > 1$). Clearly, the line graph of a star is a complete graph. If v is a vertex in a graph H, let S_v denote the complete subgraph in $L(H)$ obtained from the edges incident to v in H. Also three mutually adjacent vertices form a triangle, and the line graph of a triangle is complete. When u, v and w are the vertices of a triangle in H, let S_{uvw} denote the corresponding complete subgraph of $L(H)$. These two types subgraphs are interesting to us since if $G = L(H)$, then the complete subgraphs of G are either S_{uvw} where u, v and w are the vertices of a triangle in H, or S_v for v a vertex of H (not all these are maximal).

Lemma 4 *The minimum clique cover for line graph $G = L(H)$ consists of the following set of subgraphs:*

(i) S_v, for each vertex v of H provided that either
(a) degree(v) $\neq$ 2,
or (b) degree(v) = 2
and (its adjacent vertices are not adjacent
or they both have degree three or more),
or (c) degree(v) = 1
and its neighbour u has degree 1 and S_u isn't included
and (ii) S_{vuw}, for u, v and w forming a triangle in H with at least two of these vertices having degree two.

Proof: First the set of all S_v clearly covers the graph G. The only S_v not included in the set of the lemma are of two types. One type is a vertex v of degree 1. When its neighbour u also has degree 1, $S_u = S_v$ and exactly one of them is included in the cover. When its neighbour u has degree greater than 1, then S_v is contained in S_u, and S_v is not needed in a cover. The other type is for v having degree two when the two vertices adjacent to it (u and w) are adjacent, and at least one of them has degree two. In this case, the three vertices form a triangle that has at least two vertices of degree two, and hence the edges, that would be covered by S_v and either (or both of) S_u and S_w, are covered by S_{uvw}. Thus, the set of subgraphs covers G.

We now show that the set has a minimum size.

First note that an edge (a, b) of G corresponds to two incident edges $a = (u, v)$ and $b = (v, w)$ of H. The only cliques of G that cover (a, b) are S_v and, if u is adjacent to w, S_{uvw}. Hence, if v is a non-simplicial vertex of H, there is a pair of edges incident to v that do not form a triangle so S_v must be in any cover of G to cover the incidence of these two edges.

Now assume that v is a simplicial vertex of H. If v has degree four or more, then it would take far too many triangles to cover the line graph of the edges incident to v. Hence a minimum cover of G contains S_v. If v has degree three, a minimum cover of the line graph of $N[v]$ can be either four S_i's or four S_{ijk}'s. Thus, we can use the S_i's in our minimum cover. Suppose degree(v)=1 with u adjacent to v. If degree(u)>1, then S_v is contained in S_u and S_v is not needed in a cover. If degree(u)=1, then S_u=S_v and only one of them is needed in a cover. The case of v having degree zero will not occur, as the line graph of an isolated vertex is the empty graph.

The final case is for v a simplicial vertex of degree two in H. Suppose that u and w are the neighbours of v. If at least one of the vertices u and w has degree 2, the use of S_{uvw} in covering $N[v]$ saves at least one subgraph over using S_v,

S_u and S_w. If v is the only one with degree 2, then S_v, S_u and S_w can be used in a minimum cover as otherwise S_{uvw}, S_u and S_w would be needed.

Thus, the specified set is a minimum cover of G. □

Therefore if G is a line graph with a known root graph, then by analyzing the degrees of the vertices in the root graph, a minimum clique cover for G can be found in time $O(p+n)$ where p and n are respectively the number of vertices and edges in the root graph. Hence, the time is dominated by the time to find the root graph.

4.2 Independent Set and Vertex Cover

A subset of the vertices is called an independent (stable) set if no pair of the vertices is adjacent. The size of the maximum independent set will be denoted by $\beta_0(G)$. This value is related to the clique covering numbers of graphs.

Theorem 2 *[12]*
The following are equivalent
(i) $\Theta_e(G) = \Theta_v(G)$
(ii) $\beta_0(G) = \Theta_e(G)$.

This implies that for upper bound graphs $\beta_0(G) = \Theta_e(G) = \Theta_v(G) = s$. For simplicial graphs, no two independent vertices can belong to the same simplex, and simplicial vertices from distinct simplices are independent. Hence for simplicial graphs, $\beta_0(G) = \Theta_v(G) = s$, and this value is easily calculated.

For line graphs, an independent set corresponds in the root graph to a set of edges such that no pair of them are incident, i.e., a matching. The maximum matching of the root graph can be found in time $O(p^{0.5}n)$ [33, 36], where p is the number of vertices and n is the number of edges in the root graph. Hence, a maximum independent set can be obtained by finding the root graph and then a maximum matching for it.

A vertex cover is a set of vertices such that each edge of the graph is incident to at least one of the vertices. This is closely related to independence as a set S is a vertex cover iff $V - S$ is an independent set. Thus, the time bounds for maximum independent set apply to minimum vertex cover.

4.3 Maximum Clique

The maximum clique problem is to determine a largest clique in a graph. First we consider this problem for upper bound graphs. For an arbitrary graph G, construct graph $\hat{\mathrm{G}}$ by adding to G as follows: for each edge of G, add a new vertex which is only adjacent to the two ends of the edge. Then $\hat{\mathrm{G}}$ is an upper bound graph, and unless G contains no triangles, a maximum clique of $\hat{\mathrm{G}}$ has the same size as a maximum clique of G. Thus, if the size of a maximum clique could be efficiently determined for upper bound graphs, then it could be determined for arbitrary graphs. Since it is NP-hard for general graphs, it is

NP-hard to determine the size of a maximum clique in upper bound graphs, and by implication in simplicial graphs.

The situation for line and middle graphs is quite different. We have already seen that if G is a line graph with root graph H, then the complete subgraphs of G are either S_{uvw} where u, v, and w are the vertices of a triangle in H, or S_v for v a vertex of H (not all these are maximal). For a graph line, we first find the largest star (largest degree) in the root graph. Only if this star has size less than three is it necessary to test for triangles. But in this case, the triangle test is quick as no vertex has degree as much as three. Thus, a maximum clique can be found in time proportional to the number of vertices and edges of its root graph.

In a middle graph, a triangle in the root graph only generates a clique as large as 3. However, a triangle contains P_2 (a star), and a P_2 by itself generates a K_3. Hence, a triangle never generates a clique larger than the largest clique generated by a star, so the maximum clique corresponds to the largest star of the root graph.

The complexity of this problem for the representative graphs of hereditary hypergraphs is presently unknown.

4.4 Chromatic Number and Chromatic Index

The chromatic number problem involves colouring the vertices of an undirected graph so that no two adjacent vertices have the same colour. $\chi(G)$ is used to represent the smallest number of colours needed for colouring the vertices of G. The chromatic index, $\chi'(G)$, is the smallest number of colours needed to colour the edges of a graph so that no two incident edges have the same colour.

Since edge colouring is NP-complete for general graphs, vertex colouring is NP-complete for line graphs (since $\chi(L(G)) = \chi'(G)$). Vertex colouring is also NP-complete for upper bound graphs. To see this, for any graph G, consider the upper bound graph $\hat{\mathrm{G}}$. For $k \geq 3$, now it is easy to see that the vertices of G can be coloured with k colours iff the vertices of $\hat{\mathrm{G}}$ can be coloured with k colours. From this the NP-complete result follows for upper bound graphs.

Now consider middle graphs.

Theorem 3
$\chi(M(G)) = \chi'(G^+) = \Delta(G) + 1$, *where* $\Delta(G)$ *is the maximum degree in graph* G.

Proof: As we previously noted, $M(G) = L(G^+)$.
Hence $\chi(M(G)) = \chi(L(G^+)) = \chi'(G^+)$.
By Vizing's fundamental result [45], $\chi'(G) = \Delta(G)$ or $\Delta(G) + 1$.

case (i) $\chi'(G) = \Delta(G)$
Then $\chi'(G^+) = \Delta(G) + 1$ as all the added edges can be given the new colour.
But $\Delta(G^+) = \Delta(G) + 1$ so $\chi'(G^+) = \Delta(G) + 1$.

case (ii) $\chi'(G) = \Delta(G) + 1$

But each vertex of G has at most $\Delta(G)$ incident edges so there is at least one colour remaining. This colour can be used for the new edge incident to the vertex.

Therefore $\chi'(G^+) = \Delta(G) + 1$.

□

Now to find a vertex colouring for $M(G)$, we first find G and G^+, and then need a $\Delta(G)+1$ edge colouring of G^+. Implicit in the proof of Vizing's Theorem by Fiorini and Wilson [16] is an $O(e^2)$ algorithm to edge colour any graph G with $\Delta(G) + 1$ colours. Thus middle graphs can be vertex coloured in $O(n^2)$ time, where n is the number of vertices in the middle graph. Again, the time complexity of this problem is unknown for representative graphs of hereditary hypergraphs.

The problem of computing the chromatic index of a graph is a very difficult to classify. For many classes of graphs, it is an open question as to whether there is a polynomial algorithm [25]. However, the problem has been shown [9] to be NP-complete for line graphs. Also, all NP-complete results to date have been for regular graphs. None of simplicial, upper bound, representatives of hereditary hypergraphs and middle graphs can ever be regular unless they are complete. Thus, perhaps there exist polynomial algorithms for these classes. Middle graphs in particular seem to be simple enough that there might be an efficient algorithm to edge colour them. Nevertheless, the problem remains open for all these classes.

4.5 Dominating Set

A set of vertices is called dominating if every vertex is either in the set or adjacent to a vertex in the set. The domination problem is to find a minimum dominating set. The size of such a set is denoted by $\gamma(G)$. To analyze the complexity of the domination problem for upper bound graphs, we again consider the graph $\hat{\mathrm{G}}$. It is easy to see that when G has no isolated vertices, a vertex cover of G is a dominating set for $\hat{\mathrm{G}}$, and a minimal dominating set for $\hat{\mathrm{G}}$ can be converted into a vertex cover for G of the same size. Thus, an efficient algorithm for the dominating set of upper bound graphs would imply an efficient algorithm for the vertex cover of arbitrary graphs. Since the latter problem is NP-complete, domination for upper bound and simplicial graphs is NP-complete.

In fact, the dominating set problem is NP-complete for representative graphs of hereditary hypergraphs. It is known [14] that for $H = (V, E)$ a hereditary hypergraph, $\gamma(\Omega(H)) = \pi(H)$, where $\pi(H)$ is the smallest order of a partition of V by the sets of E. However, the maximal edges of H dominate the most vertices of V, and all subsets of a maximal edge exist in E. Thus, domination in $\Omega(H)$ reduces to minimum set cover of the vertices of the root hypergraph by its maximal edges. But minimum set cover is NP-complete [17] so domination for the representative graphs of hereditary hypergraphs is NP-complete.

Middle graphs are a sufficiently specialized class of hereditary hypergraphs that domination can be efficiently solved. It is known [14] that for a connected graph G, $\gamma(M(G)) = \pi(M(G)) = \alpha_1(G)$, where $\alpha_1(G)$ is the size of a minimum edge cover, i.e., a minimum set of edges so that every vertex is incident to at least one edge of the set. The edge cover problem can be solved by using maximum matching to obtain a maximum set of independent edges (covering a maximum number of vertices) and then adding a minimal set of edges to cover vertices not already covered [35]. Thus, given the root graph G with p vertices and q edges, the dominating set problem for $M(G)$ can be solved in the time to do maximum matching in G, i.e., $O(p^{0.5}q)$. As the number of vertices in $M(G)$, n, dominates both p and q, the bound is $O(n^{1.5})$ plus the time to find G.

The last dominating set problem that we consider is for line graphs. Now a dominating set of a line graph corresponds in the root graph to a set of edges such that every edge of the root graph is incident to at least one edge of the set, i.e., the edge dominating set problem. As the edge dominating set problem is NP-complete [46], it follows that the dominating set problem is NP-complete for line graphs.

4.6 Edge Dominating Set

As we have just seen, a set $S \subseteq E$ is called an edge dominating set for graph $G = (V, E)$ if every edge either belongs to S or is incident to an edge of S. The edge dominating set problem for a graph is to determine the smallest edge dominating set for the graph. Let γ' denote the size of such a set. For a set $S \subseteq E$, a vertex or edge is called covered if it is incident to an edge of S.

Consider a minimal edge dominating set. Suppose it has two incident edges (x, y) and (y, z). Since (y, z) covers all the edges of y, there is an edge (w, x) which is only covered by (x, y). Thus the set $(S - \{(x, y)\}) \cup \{(w, x)\}$ is minimal edge dominating with one fewer pair of incident edges. Therefore, there exists a minimal edge dominating set of the same size as S with no incident edges. A set of edges with no pair of them incident is called a matching. But a maximal matching has every edge covered so it is a edge dominating set. Thus finding γ' is the same as finding the size of a minimum maximal matching. In turn, this is the same as finding a maximal matching which leaves the most vertices uncovered.

It is known [24] that the minimum edge dominating set problem is NP-complete for line graphs. However, it is easily solved for a complete graph as an edge dominating set can leave at most one vertex uncovered ($\gamma'(K_{2n}) = n$ and $\gamma'(K_{2n+1}) = n$). In the case of covering edges of a simplex, we can assume that the uncovered vertex (if it exists) is an arbitrary simplicial vertex. Also if only simplicial vertices are uncovered, then all edges between simplices are covered. Therefore, in a simplicial graph, a maximal matching which maximizes the number of simplices with an uncovered simplicial vertex is a minimum maximal matching and a minimum edge dominating set. This approach will be used to find γ' in a simplicial graph, i.e., find a maximal matching such that a maximum number of simplices have an uncovered simplicial vertex.

Algorithm: Finding γ' in a simplicial graph
Input: Graph $G = (V, E)$
Output: An edge dominating set of size γ'
1. find the simplices and simplicial vertices of G.
2. delete a simplicial vertex from each simplex to obtain graph G'.
3. find M, a maximum matching of G'.
4. for each simplicial vertex removed in step 2
   return it to the graph
   if there is another uncovered vertex in its simplex
   then add to M the edge that joins them.

The result is obviously a maximal matching of the overall graph. It maximizes the number of edges in the matching of G', which minimizes the number of uncovered vertices in G', which minimizes the number of edges added to the matching in step 4, which maximizes the number of simplices in G with an uncovered vertex, which minimizes the size of the edge dominating set. Thus, the algorithm finds a minimum edge dominating set.

The most time consuming part of the algorithm is finding a maximum matching in G'. Thus the time complexity is $n^{0.5}e$, as that is the time to find a maximum matching. However, in practice the time will normally be much better as a large initial matching can easily be found by starting with the empty graph and adding each of the simplices (without the deleted simplicial vertex) one at a time. As each is added, an initially empty matching is extended so that it is a maximum size in the simplex.

4.7 Maximum Minimal Dominating Set

For any undirected graph G, there are six parameters that are related as follows [14]:

$$ir(G) \leq \gamma(G) \leq i(G) \leq \beta_0(G) \leq \Gamma(G) \leq IR(G).$$

$\gamma(G)$ and $\beta_0(G)$ are the domination and independence numbers that we have already considered. $i(G)$ is the size of a minimum (smallest) maximal independent set (a minimum dominating independent set), and $\Gamma(G)$ is the size of a maximum (largest) minimal dominating set. For a set I and vertex $v \in I$, v is said to have itself as a private closed neighbour if v is not adjacent to any vertex of I; and v is said to have u as its private neighbour, if v is adjacent u, $u \notin I$ and u is not adjacent to any other member of I. A set of vertices I is called irredundant if each member of I has a private (closed) neighbour, i.e., using the notation of Section 2, for all $v \in I$, $N[I] - N[I - \{v\}] \neq \emptyset$. $IR(G)$ is the size of a largest maximal (i.e., maximum) irredundant set, and $ir(G)$ is the size of a minimum (smallest) maximal irredundant set. In this section, we consider $\Gamma(G)$, and consider $i(G)$, $IR(G)$ and $ir(G)$ in subsequent sections.

The paper [10] showed that for upper bound graphs $\beta_0(G)) = \Gamma(G) = IR(G) = s$, and for simplicial graphs $\beta_0(G)) = \Gamma(G) = s$, with $IR(G)$ not necessarily equal to the others. Thus, since it is easy to calculate the number

of simplices s, $\Gamma(G)$ is easily calculated for the class of simplicial graphs and descendants of this class.

On the other hand, this is not true for line graphs as it has been shown [32] that the calculation of $\Gamma(L(G))$ is NP-complete.

4.8 Irredundant Set

The previous section has the definition for $IR(G)$, and the fact that $IR(G) = s$ for upper bound graphs. Hence for upper bound graphs $IR(G)$ is easy calculated.

We will now show that calculation of $IR(G)$ is NP-complete for line and simplicial graphs. The same reduction, from Simple Max Cut, is used for both problems. Recall the Simple Max Cut problem is to determine whether there is a partition of the vertices of G such that there are at least k edges with one end in one set of the partition and the other end in the other set of the partition.

For a graph G, the subdivision graph of G, $S(G)$, is obtained by adding a new vertex w for each edge (u, v) of G and replacing (u, v) by (u, w) and (w, v) [23]. Also, if F is a set of edges with vertex set the same as graph G, then $G + F$ is the graph (possibly a multigraph) obtained by adding the edges of F into G. Thus for $G = (V, E)$, $G + E$ is the multigraph obtained by doubling each edge of G. The following theorem is the key to the reduction for NP-completeness of IR for simplicial and line graphs.

Theorem 4
For $G = (V, E)$ an undirected graph, G has a vertex partition with at least k edges between the sets iff the line graph of the subdivision graph of the multigraph $G + E$, $L(S(G + E))$, has an irredundant set of size at least $2 * k$.

Proof: Note that for each vertex a of degree d in G, there is a corresponding clique $C(a)$, called the vertex clique of a, of size $2*d$ in $L(S(G+E))$. The set of vertex cliques forms a partition of the vertices of $L(S(G+E))$. Also, each edge in G is first doubled and then each resulting edge is subdivided. This results in four vertices in $L(S(G + E))$. A doubling pair is defined to be two vertices from $L(S(G + E))$ that are obtained from the same edge of G and belong to the same vertex clique. The two vertices of $L(S(G + E))$ that result from the subdivision of an edge of $G+E$ are called a subdivision pair, and an edge which joins a subdivision pair is called a subdivision edge. A vertex in $L(S(G + E))$ is only adjacent to one vertex outside the vertex clique that contains it. The one vertex is its subdivision partner and the edge joining them is a subdivision edge.

Only If:

Suppose G has a vertex partition where V is partitioned into V_1 and V_2 such that there are at least k edges with one end in V_1 and the other end in V_2. Let D be the set of these edges.

Let CV_1 be the set of vertex cliques corresponding to the vertices of V_1, and CV_2 be the set of vertex cliques corresponding to the vertices of V_2.

For each $e \in D$, $e = (a, b)$, where $a \in V_1$ and $b \in V_2$.

Then $C(a) \in CV_1$ and $C(b) \in CV_2$.
Suppose u_1 and u_2 are the doubling pair from $C(a)$ that result from edge e.
Add u_1 and u_2 to I.

Then $|I| = 2 * k$, and $I \subseteq \cup\{C \mid C \in CV_1\}$.
Also, each vertex $u \in I$ is a member of a doubling pair for some edge $e \in D$.
Note that u has as its private neighbour its subdivision partner (whose only other adjacencies are in some $C(b) \in CV_2$).
Therefore I is an irredundant set of size at least $2 * k$ in $L(S(G + E))$.
If:
Suppose $L(S(G + E))$ has an irredundant set I of size p.
We will show that G has a vertex partition with at least $\lceil p/2 \rceil$ edges between the sets of the partition.
The general approach is to modify the set I so that it is still irredundant and has an even size of at least p, and then show that from it we can find a vertex partition of G with at least $|I|/2$ edges between the sets.
Consider an arbitrary $u \in I$.
Suppose $u \in C(a)$ where u originated from edge $(a, b) \in E$.
Let v be u's subdivision partner, u' be u's doubling partner, and v' be v's doubling partner (and u''s subdivision partner).
Hence, $u, u' \in C(a)$ and $v, v' \in C(b)$.

If $u' \notin I$ and v is a private neighbour of u,
then $I \cap C(b) = \emptyset$
add u' to I as it has private neighbour v'.
If $u' \notin I$ and v is not a private neighbour of u,
then $\exists w \in I \cap C(b)$
$I \cap C(a) = \{u\}$ as u's private neighbour must be in $C(a)$
Note: if $\exists z \in I \cap C(b), z \neq w$,
z must have a private neighbour outside $C(b)$, as both $w, z \in I$
remove w and u from I, and add v and v' to I
Thus, v and v' have u and u' as private neighbours.

Therefore, after this step either both u and u' belong to I, or both v and v' belong to I.
Repeat this step for every vertex $u \in I$.
Hence, the resulting set I is irredundant, has size at least p, and if a vertex u exists in I then its doubling partner also exists in I.
Now define $V_1 = \{a \in V \mid C(a) \cap I \neq \emptyset\}$, and $V_2 = \{b \in V \mid C(b) \cap I = \emptyset\}$.
Then V_1 and V_2 form a partition of V.
For each doubling pair from I, say u and u' from $C(a)$, with private neighbours v and v' (their subdivision partners) from some $C(b)$, there is an edge in G from $a \in V_1$ to $b \in V_2$.
Hence there are at least $|I|/2$ edges from V_1 to V_2 so it is the required partition.
□

Corollary 1
Let $G = (V, E)$ be an arbitrary undirected graph.
Form the graph G' by adding to each vertex clique of $L(S(G + E))$ a vertex adjacent to every vertex in the vertex clique and no other vertices.
Then G' is simplicial, and G has a vertex partition with at least k edges between the sets iff G' has an irredundant set of size $2 * k$.

Proof: G' is simplicial as the vertex cliques cover the vertices of G' and for each vertex clique, the new vertex added to it is simplicial.
It is easy to see that the Only If proof of the theorem still holds when a simplicial vertex is added to each vertex clique.
For the If proof of the theorem, it is necessary to specify how to handle the case when an added simplicial vertex is a member of I. Suppose $s \in I$, s is the simplicial vertex of vertex clique $C(a)$, and $u \in C(a)$ is a private neighbour of s.
Then u's subdivision partner v does not belong to I. Suppose $v \in C(b)$.

If $I \cap C(b) = \emptyset$
    remove s from I and add u and its doubling partner to I
    v and its doubling partner are private neighbours of u and its doubling partner
If $I \cap C(b) = \{w\}$
    remove s and w from I and add v and its doubling partner to I
    u and its doubling partner are private neighbours of v and its doubling partner
If $|I \cap C(b)| \geq 2$
    remove s from I and add v and its doubling partner, if it isn't already present, to I
    u and its doubling partner are private neighbours of v and its doubling partner

Now the rest of the If proof holds.
So, the theorem holds for G and G', and the corollary follows. □

Corollary 2
IR is NP-complete for line graphs and simplicial graphs.

Proof: It is obvious that the IR problem is in NP for it is possible to guess a set I of size k and then verify in polynomial time that it is irredundant.
IR for line graphs and simplicial graphs will now be reduced to the Simple Max Cut problem.
Let G be an arbitrary undirected graph.
By the theorem, a max cut of size at least k exists iff $L(S(G + E))$ has an irredundant set of size at least $2k$.
Note that given G, $L(S(G+E))$ can be obtained in polynomial time. Also given an irredundant set I for $L(S(G+E))$, a partition of G with at least $|I|/2$ edges can be obtained in polynomial time by the constructive approach used in the

theorem.
Therefore, Simple Max Cut reduces to IR for line graphs.
Similarly, it reduces to IR for simplicial graphs using the previous corollary.
Hence, since Simple Max Cut is NP-complete [18], IR is NP-complete for line graphs and simplicial graphs. □

4.9 Minimum Dominating Independent Set and Minimum Maximal Irredundant Set

These two parameters were introduced in section 4.7. The key to their complexity is the following result:

Theorem 5 *[14]*
If $H = (V, E)$ is a hereditary hypergraph with no isolated vertices, then
$ir(\Omega(H)) = \gamma(\Omega(H)) = i(\Omega(H))) = \pi(H)$
where $\pi(H)$ is the smallest order of a partition of V into elements of E.

In section 4.5, we have already seen that calculation of $\pi(H)$ is NP-complete for H a hereditary hypergraph. Thus, γ, ir and i are NP-complete for representatives of hereditary hypergraphs, upper bound graphs, and simplicial graphs. Again in section 4.5, we recalled that for G a connected graph, $\gamma(M(G)) = \pi(M(G)) = \alpha_1(G)$, and $\alpha_1(G)$ can be calculated in time $O(n^{1.5} + e)$, where $M(G)$ has n vertices and e edges. Thus, ir and i can be calculated in this same time.

Now consider calculating $i(L(G))$, i.e. finding the size of a minimum sized maximal set of vertices of $L(G)$ that is independent. In G, this will be a minimum sized maximal set of edges that are not incident, i.e., a minimum maximal matching. However, as we have already seen, the minimum maximal matching problem is NP-complete, so i is NP-complete for line graphs.

McRae [32] shows that calculating $ir(L(G))$ is also NP-complete.

Note that a graph is called well-covered [37] if all maximal independent sets have the same size, i.e., $i(G) = \beta_0(G)$. It turns out that the class of simplicial graphs that are also well-covered is the same as the class of vertex unisimplicial graphs [39]. Also, for the natural definition of a graph being fractionally well-covered, a graph is vertex unisimplicial iff it is fractionally well-covered [15]. Vertex unisimplicial graphs have even more properties equal.

Theorem 6 *[40]*
If G is a vertex unisimplicial graph, then
$\gamma(G) = i(G) = \beta_0(G) = \Gamma(G) = \Theta_v(G) = s$
where s is the number of simplices in the graph.

This cannot be extended to the other related parameters as it is quite easy to find vertex unisimplicial graphs with $ir(G) < \gamma(G)$ or with $\Gamma(G) < IR(G)$.

4.10 Hamiltonian Cycle/Path

A Hamiltonian path in a graph is a path that passes through every vertex exactly once. If it returns to the starting vertex, it is called a Hamiltonian circuit. It is known that the Hamiltonian path problem is NP-complete for line graphs [4]. In fact, Bertossi's proof constructs a line graph that is actually a middle graph. Hence the proof shows that the Hamiltonian path problem is NP-complete for middle graphs. In addition, the same proof also works for the Hamiltonian circuit problem, so both problems are NP-complete for all the classes of graphs being considered here.

4.11 Induced Path

The Induced Path problem has as input an undirected graph $G = (V, E)$ and an integer k, and the objective is to determine whether the graph has a subset V' of vertices such that the subgraph of G induced by V' is a simple path. The fact that the Induced Path problem is NP-complete for middle graphs, and hence the other graph classes, follows directly the following theorem:

Theorem 7
For a graph G with n vertices, G has a Hamiltonian path iff $M(G)$ has an induced path of length at least n.

Proof: Recall that the set of simplices of a middle graph forms a partition of the edges. Also in an induced path, there can be at most one edge from any simplex (otherwise a triangle would be induced). Hence the longest possible induced path in $M(G)$ has length n, when G has n vertices.
Only If:
Suppose G has a Hamiltonian path $P = v_1 v_2 ... v_n$.
Define path P' in $M(G)$ by $v_1 v_{1,2} v_{2,3} ... v_{n-1,n} v_n$, where $v_{a,b}$ is the vertex in $M(G)$ arising from edge (a, b) of G.
This path has length n. Also there is exactly one edge from each simplex so it is the desired induced path.
If:
If an induced path has length n in $M(G)$, then it must contain an edge from each simplex. Also each interior vertex of the induced path corresponds to an edge of G (since the other vertices of $M(G)$ are simplicial which would imply at least 2 edges from the simplex). Since consecutive interior vertices of the induced path are adjacent, the corresponding edges of G are incident and form a path of length $n - 1$ in G. The vertices of the path in G must be distinct or else there would be more than one edge of the path in some simplex of $M(G)$. Hence the path in G is a Hamiltonian path. □

4.12 Graph Isomorphism

The graph isomorphism problem asks the question of whether two graphs have the same structure. In particular, given graphs $G_1 = (V_1, E_1)$ and $G_2 =$

(V_2, E_2), does there exists a one-to-one and onto function $f : V_1 \to V_2$ such that $(u, v) \in E_1$ iff $(f(u), f(v)) \in E_2$. The computational complexity of this problem is particularly interesting as, although it is easily shown to be in NP, it has not been proved to be in either P or NP-complete. However if NP $\neq$ P, then it can be proved that problems exist that are in NP but not in either P or NP-complete. The graph isomorphism problem is one of the leading candidates to fit this criterion. This has led to the development of the Isomorphism Complete class of problems, i.e., problems whose complexity class is the same as graph isomorphism. It has been known for some time that isomorphism of middle graphs is the same as general graphs [14]. This follows from the following relationships:

$$G_1 \approx G_2 \text{ iff } G_1^+ \approx G_2^+ \text{ iff } L(G_1^+) \approx L(G_2^+) \text{ iff } M(G_1) \approx M(G_2).$$

Thus, graph isomorphism is Isomorphism complete for all the graph classes under consideration. It is fairly easy to see that isomorphism for independence graphs is also Isomorphism complete.

4.13 Subgraph Isomorphism

Given graphs G and H, the subgraph isomorphism problem asks whether H is isomorphic to a subgraph of G. It is known that this problem is NP-complete when G is restricted to one of several classes of graphs. We will now show that the problem is NP-complete when G is a middle graph by reducing the Hamiltonian cycle problem for cubic graphs to the subgraph isomorphism problem for middle graphs. Recall that a graph is called cubic if every vertex has degree three.

Theorem 8
For a cubic graph G with n vertices, G has a Hamiltonian circuit iff $M(G)$ has a subgraph isomorphic to $M(C_n)$, where C_n is the graph that consists of a cycle on n vertices.

Proof: Only If:
If G has a Hamiltonian circuit, then G has a subgraph H, $H \approx C_n$, where $n = |V|$. This obviously implies that $M(G)$ has a subgraph isomorphic to $M(C_n)$.
If:
Suppose $G = (V, E)$ is a cubic graph such that $M(G)$ has a subgraph isomorphic to $M(C_n)$. It is easy to see that $M(C_n) \approx \hat{C}_n$; i.e., it is a sequence if n triangles connected so that each triangle shares one vertex with the next triangle, this shared vertex is distinct from the one shared with the previous triangle, and the set of edges between the shared vertices induces a cycle of length n. Two triangles that share a vertex will be called incident.

The proof will be easily completed after we show that $M(C_n)$ is isomorphic to a subgraph of $M(G)$ such that for each triangle of $M(C_n)$, all of its edges are mapped by the isomorphism into the same simplex of $M(G)$. This will be

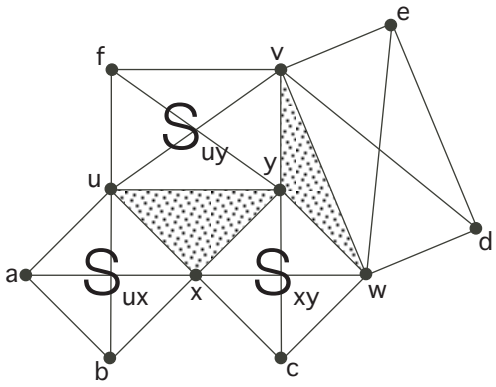

Figure 2: Incident triangles neither in a simplex

shown by considering a triangle of $M(C_n)$ that does not map to a simplex, and showing that the map can be changed to a map where the triangle does map to a simplex.

Suppose that a triangle of $M(C_n)$ is mapped to $\{u, x, y\}$, where the degree 2 vertex of the triangle is mapped to u. Recall that for middle graphs, each non-simplicial vertex lies in exactly two simplices, and the simplices partition the edges. Thus if $\{u, x, y\}$ do not all belong to the same simplex, then each edge of the triangle must belong in a different simplex, call them S_{ux}, S_{xy}, and S_{uy}. Because of the isomorphism from $M(C_n)$, the $\{u, x, y\}$ triangle has two incident triangles. We now claim that at least one of these two incident triangles must be interior to a simplex. Figure 2 shows the only way (subject to symmetry) in which two incident triangles can appear in $M(G)$ when neither of the triangles is interior to a simplex. Another triangle incident to $\{u, x, y\}$ must be incident at x, since a degree 2 vertex is mapped to u. As the new triangle is not interior to S_{ux} or S_{xy}, it must consist of $\{x, b, c\}$ and (b, c) is an edge of $M(G)$. But this is impossible as S_{xy} is a simplex with c as its simplicial vertex, so (b, c) is not an edge of $M(G)$. Hence, the second triangle incident to $\{u, x, y\}$ must be interior to S_{ux}, i.e., it is $\{a, x, b\}$ as shown in Figure 3(a). In particular, there cannot be a sequence of three incident triangles where none of them is interior to a simplex.

This implies that the triangle incident to $\{v, y, w\}$ must be interior to S_{vw}. Suppose that it is $\{v, e, d\}$ (see Figure 3(a)). The map from $M(C_n)$ can be changed so that instead of mapping to the triangles of Figure 3(a)), it maps to those of 3(b)). In particular, triangle $\{u, x, y\}$ can be replaced by $\{c, x, y\}$, and triangle $\{w, y, v\}$ replaced by $\{f, y, v\}$ to yield another isomorphism mapping - this one with all four of the triangles interior to simplices. Note that c and f must be simplicial vertices, and hence are not part of any triangles in the original mapping. In a second situation, if the triangle incident to $\{v, y, w\}$ was $\{w, e, d\}$, then during the replacements $\{w, e, d\}$ must be replaced by $\{v, e, d\}$

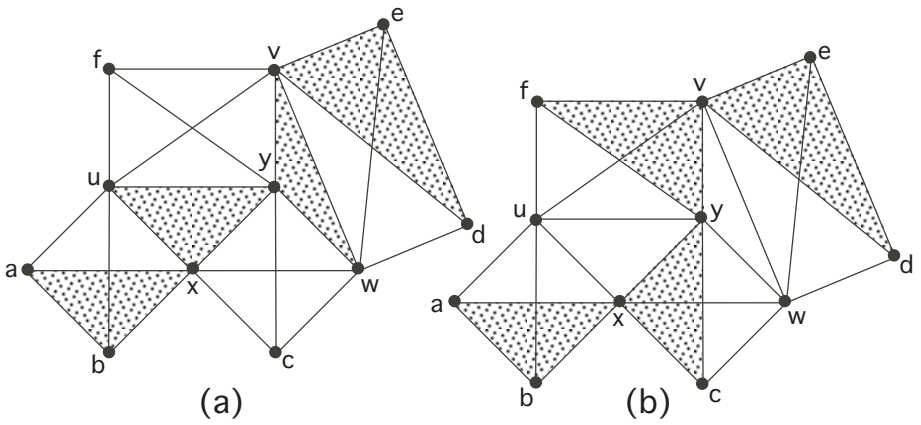

Figure 3: Handling two incident triangles neither in a simplex

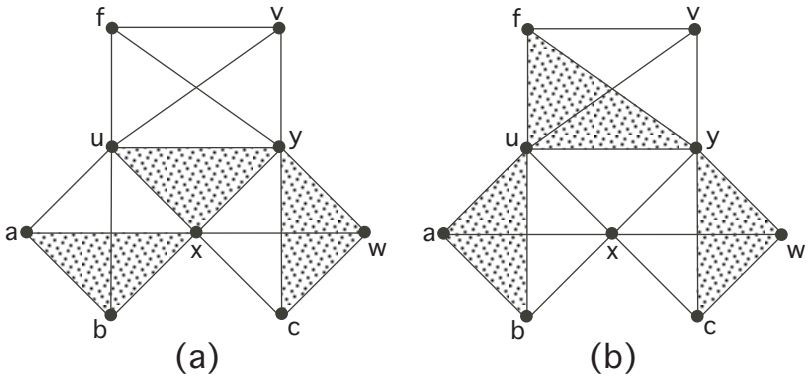

Figure 4: Handling a single triangle not in a simplex

to again yield Figure 3(b). Therefore, any incident pair of triangles that are not interior to simplices can be replaced by ones interior to simplices.

Now consider a triangle that is not interior to a simplex, but whose incident triangles are both interior to simplices (see Figure 4(a)). This time $\{x, a, b\}$ can be replaced by $\{u, a, b\}$, and $\{x, u, y\}$ replaced by $\{f, u, y\}$ to yield all interior triangles as shown in Figure 4(b). Note that if in the original mapping, the triangle incident to $\{u, y, x\}$ was $\{f, y, v\}$, then the only replacement is $\{u, x, y\}$ by $\{c, x, y\}$. Thus, any triangle that is not interior to a simplex can be replaced by one interior.

Therefore, in every case we can reduce the number of triangles that do not map to a simplex. By repeated application of this technique, an isomorphism can be obtained that maps all triangles into simplices with one triangle per simplex. Thus the cycle of incident triangles in $M(C_n)$ becomes a cycle of

simplices in $M(G)$ where consecutive simplices share a non-simplicial vertex. But each simplex of $M(G)$ corresponds to a vertex in G, and in G the cycle is a sequence of vertices with consecutive vertices connected by an edge (the non-simplicial vertex in $M(G)$). Thus, sequence is a cycle of length n and the vertices are distinct, so it is a Hamiltonian circuit. □

Since testing an arbitrary cubic graph G for a Hamiltonian circuit is NP-complete [19], testing $M(G)$ for a subgraph isomorphic to $M(C_n)$ is NP-complete, and the Subgraph Isomorphism problem is NP-complete for middle graphs, line graphs, upper bound graphs and simplicial graphs.

4.14 Steiner Tree

For this problem, the input is an undirected graph $G = (V, E)$, a subset of the vertices V', and an integer k. The objective is to determine whether there is a subtree T of G which spans V' and has at most k edges. As with the last problem, we will show this problem is NP-complete for all the graph classes being considered here. The proof will be very similar to the one of Karp [26] to show the Steiner Tree problem is NP-complete for general undirected graphs. In particular, the Exact 3-Cover Problem will be reduced to the Steiner Tree Problem for middle graphs.

The Exact 3-Cover Problem has as input a set $S = \{a_1, a_2, ..., a_n\}$ and a collection $C = \{C_1, C_2, ...C_p\}$ of 3 element subsets of S. The objective is to determine if C has a subcollection C' such that the sets of C' are disjoint and cover S (i.e., C' forms a partition of S). Note that each $C_i \in C$ is a 3 element set, but it will be convenient to identify one element as the first element, another as the second element, and the remaining element as the third element, i.e., $C_i = \{b_{i,1}, b_{i,2}, b_{i,3}\}$.

From an instance of the Exact 3-Cover Problem, define the following graph $G = (V, E)$

$$V = \{h\} \cup \{C_i | 1 \leq i \leq p\} \cup \{a_j | 1 \leq j \leq n\} \cup \{b_{i,k} | 1 \leq i \leq p, 1 \leq k \leq 3\}$$

$$E = \{(h, C_i) | 1 \leq i \leq p\} \cup \{(C_i, b_{i,k}) | 1 \leq i \leq p, 1 \leq k \leq 3\}$$
$$\cup \{(b_{i,k}, a_j) | a_j \text{ is the } k\text{th element of } C_i, 1 \leq j \leq n, 1 \leq i \leq p, 1 \leq k \leq 3\}$$

Note that the vertex h will be used to connect the $n/3$ C_i's which form C'. A path from an a_j to C_i represents that if C_i is added to C', it will cover a_j. The vertices $b_{i,k}$ are used to give such paths a length of 2 so that only one of these can be used for each a_j (or the tree will need too many edges).

We can now define a Steiner Tree problem for a middle graph by considering the graph $M(G)$ where G has just been described, $V' = \{h\} \cup \{a_j | 1 \leq j \leq n\}$, and $k = 3n + n/3$. This construction can clearly be done in polynomial time. Figure 5 shows the graph obtained for a specific Exact 3-Cover problem (which doesn't have an exact 3-cover so the graph doesn't have a Steiner Tree covering $S \cup \{h\}$ with 20 edges). Also, label each simplicial vertex by the vertex of G to which it corresponds, and label each non-simplicial vertex by the edge to which it corresponds.

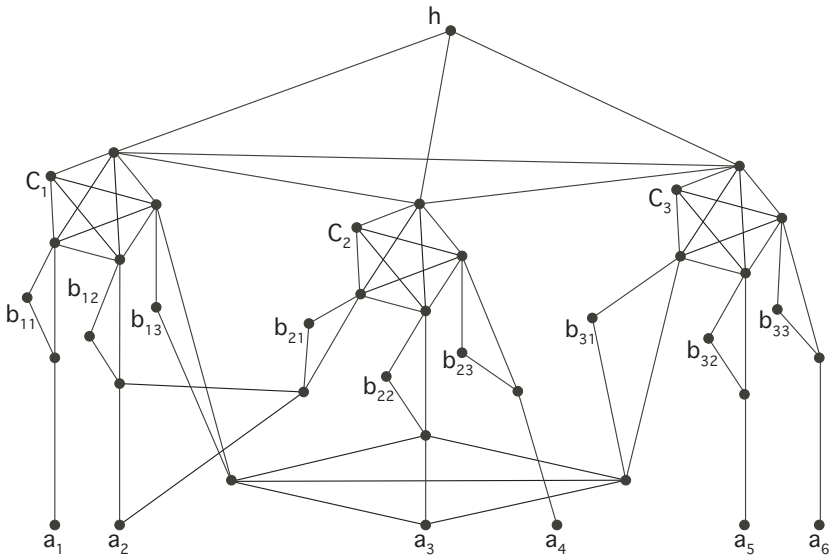

Figure 5: $M(G)$ for 3-Cover Problem $C_1 = \{a_1, a_2, a_3\}, C_2 = \{a_2, a_3, a_4\}, C_3 = \{a_3, a_5, a_6\}$

Lemma 5
S has an exact 3-cover iff $M(G)$ has a Steiner tree that covers V' using $3n+n/3$ or fewer edges.

Proof: Only If:
Suppose that S has an exact 3-cover C'.
For each $a_j \in S$, there is a unique $C_i \in C'$ with $a_j \in C_i$, say in the kth position. In the graph $M(G)$, form the following path P_j (defined by its sequence of vertices)

$$P_j = a_j, (a_j, b_{i,k}), (b_{i,k}, C_i), (C_i, h), h.$$

Define $T = \cup_{a_j \in S} P_j$.
T is a subtree of $M(G)$ since it is connected and has no cycles. Also T covers $\{h\} \cup S$. Finally T has $3n + n/3$ edges since the n paths of length 4 are edge distinct except for the $((C_i, h), h)$ edges, and each of these edges is used by three paths. Thus, T is the desired Steiner tree for $M(G)$.
If:
Suppose $M(G)$ has a subtree which covers $\{h\} \cup S$, and has $3n + n/3$ or fewer edges. Consider a smallest such subtree. If the subtree has an edge of the form $((C_i, h), (C_j, h))$, then its removal will disconnect the subtree at one of the endpoints, say (C_i, h), from being connected to h. Now the edge (C_i, h) can be added to the reconnected the subtree. The result is another subtree with the same number of edges and still covering $\{h\} \cup S$.

Also if edge $((C_i, h), h)$ belongs to the subtree, and there is a path inside the C_i simplex to vertex (C_i, h) of length greater than 1, then remove the edges of the path from the tree and add the edges which join each vertex on the path to vertex (C_i, h).

Note both operations result in the same number of edges as before, and the result is still a subtree covering $\{h\} \cup S$. Assume that both of these operations have been done as many times as possible, and call the resulting tree T.

Let x equal the number of edges in T of the form $((C_i, h), h)$.

Let y be the number of paths in T, called direct paths, of the form

$$a_j, (a_j, b_{i,k}), (b_{i,k}, C_i), (C_i, h), h, \text{ for appropriate } j, i, \text{ and } k.$$

Then $y \leq 3 * x$ since each C_i has only three elements with direct paths.
Now we count the edges of T.

x counts the edges of the form $((C_i, h), h)$.

$3y$ counts the edges on direct paths when the $((C_i, h), h)$ edges are not counted.

When y elements of S are connected by direct paths, there are $n-y$ elements of S that are connected to h by paths which are not direct. Each of these must either use at least four distinct edges of T that have not been counted yet, or else use three additional edges and connect to a non-direct path that uses at least five additional edges. Thus, at least four additional edges can be counted for each non-direct path, so

$$3n + n/3 \geq \text{number of edges in } T \geq x + 3y + 4(n - y) \geq y/3 + 4n - y,$$

since $y \leq 3x$.

Hence, $y - y/3 \geq n - n/3$, and so $y \geq n$
But y can be at most n, so $y = n$.
This implies $x = n/3$ since $y \leq 3x$ and T has at most $3n + n/3$ edges.
Thus the tree consists of $n/3$ distinct edges $((C_i, h), h)$, each of which has 3 distinct a_j's associated with it. Hence, the $n/3$ C_i's form an Exact 3-Cover of S. □

Since the Exact 3-Cover problem is NP-complete, the Steiner Tree problem must be NP-complete for middle graphs, and hence also for the other graph classes being considered here.

4.15 Simple Maximum Cut

For a graph $G = (V, E)$, the simple maximum cut problem is to partition the vertex set into two sets so as to maximize the number of edges of G with one end in each set. For some time this problem has been known to be NP-complete for general graphs [18], but for line graphs time O(n+e) is sufficient [1]. Just recently the problem has been shown to be NP-complete for split graphs [5], and hence for simplicial graphs. The time complexities of this problem for upper bound graphs and for representative graphs of hereditary hypergraphs remain open problems.

Acknowledgments

The first author would like to thank Steve Hedetniemi for first describing simplicial graphs to him, and to the Clemson Algorithms group for initial discussions of the simplicial and upper bound classes of graphs.

References

[1] C. Arbib. A polynomial characterization of some graph partitioning problems. *Inform. Process. Lett.*, 26:223–230, 1987/88.

[2] M. Behzad and G. Chartrand. Total graphs and traversability. *Proc. Edinburgh Math. Soc.*, 15:117–120, 1966.

[3] C. Berge. *Graphs and Hypergraphs.* American Elsevier Inc., New York, 1973.

[4] A. A. Bertossi. The edge Hamiltonian path problem is NP-complete. *Inform. Process. Lett.*, pages 157–159, 1981.

[5] H. L. Bodlaender and K. Jansen. On the complexity of the maximum cut problem. In *11th Annual Symposium on Theoretical Comp. Science*, volume 775 of *Springer-Verlag Lecture Notes in Computer Science*, pages 769–780, 1994.

[6] M. Borowiecki. A characterization of middle graphs and a matroid associated with middle graphs of hypergraphs. *J. Discuss. Math.*, 6:37–40, 1983.

[7] A. Brandstadt, V. Le, and J. Spinrad. *Graph Classes - A Survey.* Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, 1999.

[8] R. C. Brigham and R. D. Dutton. On clique covers and independence numbers of graphs. *Discrete Mathematics*, 44:134–138, 1983.

[9] L. Cai and J. A. Ellis. NP-completeness of edge-colouring some restricted graphs. *Discrete Applied Math.*, 30:15–27, 1991.

[10] G. A. Cheston and G. Fricke. Classes of graphs for which upper fractional domination equals independence, upper domination, and upper irredundance. *Discrete Applied Math.*, 55:241–258, 1994.

[11] G. A. Cheston, E. O. Hare, S. T. Hedetniemi, and R. C. Laskar. Simplicial graphs. *Congressus Numerantium*, 67:105–113, 1988.

[12] S. A. Choudom, K. R. Parthasarathy, and G. Ravindra. Line-clique cover number of a graph. *Proc. Indian National Science Academy*, 41:289–293, 1975.

[13] E. J. Cockayne and S. T. Hedetniemi. Independence graphs. In *Proc. 5th S. E. Conf. on Combinatorics, Graph Theory and Computing*, pages 471–491, 1974.

[14] E. J. Cockayne, S. T. Hedetniemi, and D. J. Miller. Properties of hereditary hypergraphs and middle graphs. *Canad. Math. Bull.*, 21:461–468, 1978.

[15] J. Currie and R. Nowakowski. A characterization of fractionally well-covered graphs. *Ars Combinatoria*, 31:93–96, 1991.

[16] S. Fiorini and R. J. Wilson. Edge-colorings of graphs. In L. W. Beineke and R. J. Wilson, editors, *Selected Topics in Graph Theory*, pages 103–126. Academic Press, New York, 1978.

[17] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

[18] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.

[19] M. R. Garey, D. S. Johnson, and R. E. Tarjan. The planar Hamiltonian circuit problem is NP-complete. *SIAM J. Comput.*, 5:704–714, 1976.

[20] M. C. Golumbic. Trivially perfect graphs. *Discrete Math.*, 24:105–107, 1978.

[21] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.

[22] T. Hamada and I. Yoshimura. Traversability and connectivity of the middle graph of a graph. *Discrete Math.*, 14:247–255, 1976.

[23] F. Harary. *Graph Theory*. Addison-Wesley, Reading, MA., 1969.

[24] J. D. Horton and K. Kilakos. Minimum edge dominating sets. *SIAM J. Disc. Math.*, 6:375–387, 1993.

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

[26] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–104. Plenum Press, New York, 1972.

[27] D. König. Über graphen und ihre anwendungen auf determinantentheorie und mengenlehre. *Math. Annalen*, 77:453–465, 1916.

[28] J. Krausz. Démonstration nouvelle d'une théorème de Whitney sur les réseaux. *Mat. Fiz. Lapok*, 50:75–89, 1943.

[29] P. G. H. Lehot. An optimal algorithm to detect a line graph and output its root graph. *J. ACM*, 21:569–575, 1974.

[30] T. A. McKee and F. McMorris. *Topics in Intersection Graph Theory*. SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, 1999.

[31] F. R. McMorris and T. Zaslavsky. Bound graphs of a partially ordered set. *J. Comb. Inf. Syst. Sci.*, 7:134–138, 1982.

[32] A. A. McRae. *Generalizing NP-Completeness Proofs for Bipartite Graphs and Chordal Graphs*. PhD thesis, Dept. of Computer Science, Clemson Univ., 1994.

[33] S. Micali and V. Vazirani. An $O(|V|^{1/2}|E|)$ algorithm for finding maximum matchings in general graphs. In *Proc. 21st Symp. Foundations of Computer Science*, pages 17–27. IEEE, 1980.

[34] G. T. Myers. *Upper Bound Graphs of Partially Ordered Sets*. PhD thesis, Bowling Green State University, Bowling Green, OH, 1982.

[35] R. Z. Norman and M. O. Rabin. An algorithm for a minimum cover of a graph. *Proc. Amer. Math. Soc.*, 10:315–319, 1959.

[36] P. A. Peterson and M. C. Loui. The general maximum matching algorithm of Micali and Vazirani. *Algorithmica*, 3:511–534, 1988.

[37] M. D. Plummer. Some covering concepts in graphs. *J. Comb. Theory*, 8:91–98, 1970.

[38] M. D. Plummer. Well-covered graphs: a survey. *Quaestiones Mathematicae*, 16:253–287, 1993.

[39] E. Prisner, J. Topp, and P. D. Vestergaard. Well covered simplicial, chordal, and circular-arc graphs. *J. of Graph Theory*, 21:113–119, 1996.

[40] B. Randerath and L. Volkmann. Simplicial graphs and relationships to different graph invariants. *Ars Combinatoria*, 46:211–217, 1997.

[41] F. S. Roberts. Food webs, competition graphs, and the boxicity of ecological phase space. In Y. Alavi and D. Lick, editors, *Theory and Applications of Graphs*, pages 477–490, New York, 1978. Springer Verlag.

[42] N. D. Roussopoulos. A max{m,n} algorithm for determining the graph H from its line graph G. *Inform. Process. Lett.*, 2:108–112, 1973.

[43] E. Sampathkumar and S. B. Chikkodimath. Semi-total graphs of a graph - I. *Karnatak Univ. Journal: Science XVIII*, pages 274–280, 1973.

[44] M. Skowrońsk and M. Sysło. An algorithm to recognize a middle graph. *Discrete Applied Math.*, 7:201–208, 1984.

[45] V. G. Vizing. On an estimate of the chromatic class of a p-graph (Russian). *Diskret. Analiz*, 3:25–30, 1964.

[46] M. Yannakakis and F. Gavril. Edge dominating sets in graphs. *SIAM J. Appl. Math.*, 38:364–372, 1980.