Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 11, no. 1, pp. 165–193 (2007)

Approximation Algorithms for the Maximum Induced Planar and Outerplanar Subgraph Problems

Kerri Morgan *Graham Farr*

Clayton School of Information Technology
Monash University
Victoria, 3800
Australia
http://www.infotech.monash.edu.au/
Kerri.Morgan@infotech.monash.edu.au Graham.Farr@infotech.monash.edu.au

Abstract

The task of finding the largest subset of vertices of a graph that induces a planar subgraph is known as the Maximum Induced Planar Subgraph problem (MIPS). In this paper, some new approximation algorithms for MIPS are introduced. The results of an extensive study of the performance of these and existing MIPS approximation algorithms on randomly generated graphs are presented. Efficient algorithms for finding large induced outerplanar graphs are also given. One of these algorithms is shown to find an induced outerplanar subgraph with at least $3n/(d + 5/3)$ vertices for graphs of n vertices with maximum degree at most d. The results presented in this paper indicate that most existing algorithms perform substantially better than the existing lower bounds indicate.

Article Type	Communicated by	Submitted	Revised
Regular paper	P. Eades	August 2006	April 2007

1 Introduction

The Maximum Induced Planar Subgraph problem (MIPS) is the task of finding the size of the largest subset of vertices in a graph that induces a planar subgraph. This problem is known to be NP-hard [19]. A similar problem is the Maximum Planar Subgraph problem, which is the task of finding the largest subset of edges in a graph that forms a planar subgraph. A comprehensive survey of the latter problem, with brief discussion of MIPS, is provided by Liebers [14].

Planarity of graphs has applications in areas such as graph drawing, circuit design and facility layout [4, 13, 12]. Many graph drawing applications find a large planar subgraph as the initial step in determining a layout of a graph. Jünger and Mutzel [13] discuss the use of MPS in the use of applications for graph layout. MIPS was little studied until Edwards and Farr [6, 8, 9] showed that the MIPS is useful in determining the fragmentability of classes of graphs. Fragmentability provides a measure of how susceptible graphs are to being broken into components of bounded size by the removal of a small proportion of vertices. A planar graph can be partitioned into such components by removing a small proportion of the vertices in linear time (Lipton and Tarjan [15]). Edwards and Farr show that a bound on their measure of the fragmentability of non-planar classes of graphs can be obtained from bounds on the proportion of vertices that need to be removed in order to produce a maximum induced planar subgraph [6].

In this paper, several new approximation algorithms for MIPS are presented. These include two new algorithms for finding large induced outerplanar subgraphs. An experimental analysis of both the new algorithms and the existing algorithms is provided. Algorithms for other subclasses of planar graphs are also considered for comparative purposes. These include maximal independent set and maximal induced forest.

Both new and existing algorithms were implemented. Their behaviour in terms of performance and running time was observed on over 12,000 randomly generated graphs of up to 10,000 vertices and on GDT-test-suite-CU [1], a graph set of about 10,000 graphs extracted from the ALF graph base used in [5]. Experimental results indicated that most algorithms performed substantially better than the best known lower bounds. In the experiments undertaken, the approximation algorithms for finding large induced outerplanar subgraphs produced larger subgraphs than those produced by many of the existing algorithms for MIPS. A mathematical analysis of one of the algorithms for finding an induced outerplanar subgraph shows that it finds one with at least $3n/(d + 5/3)$ vertices for graphs of n vertices with maximum degree at most d, which is close to the best known lower bound of $3n/(d + 1)$ for MIPS.

All graphs are simple, containing no multiple edges or loops. In this paper n denotes the number of vertices and m denotes the number of edges in a graph. We write P for the vertex set of an induced planar subgraph $\langle P \rangle$ of a graph G

[1]Downloaded from the GDTToolKit homepage (www.graphdrawing.org) on 4 January, 2007

and $R = V(G) \setminus P$. The neighbours of a vertex v that lie in a set of vertices X is denoted by $N_X(v)$ and the number of neighbours of v in X is denoted by $d_X(v)$.

2 Existing Algorithms for MIPS

There are few existing approximation algorithms for MIPS. In this section three of these algorithms are discussed. Algorithms for finding a maximal independent set and maximal induced forest are also considered. The subgraphs produced by these algorithms are planar (although not maximally so). These algorithms are examined in order to provide some comparison between these simpler approaches and more sophisticated methods.

2.1 The Halldórsson-Lau Algorithm

Halldórsson and Lau [11] give a linear time algorithm for graphs of maximum degree at most d with a performance ratio of $1/\lceil(d+1)/3\rceil$. They use a result of Lovász [16], who showed that for any graph of maximum degree d there exists a partition of the vertex set of the graph into α parts such that the parts induce subgraphs $G_1, G_2, ..., G_\alpha$ with each subgraph G_i having maximum degree at most d_i, and $\Sigma_{i=1}^{\alpha} d_i = d - \alpha + 1$. Hence the number of parts required to ensure each subgraph has at most degree 2 is $\lceil(d+1)/3\rceil$. Halldórsson and Lau's algorithm creates such a partition. A largest part is selected as the planar subgraph. As the subgraph produced has maximum degree at most two, this subgraph is unlikely to be a *maximal* planar subgraph.

2.2 The Vertex Addition Algorithm

Another algorithm for finding induced planar subgraphs for graphs of maximum degree at most d was presented by Edwards and Farr [7] based on a proof in [6]. In this paper, this algorithm will be referred to as the Vertex Addition algorithm. This algorithm has time complexity $O(mn)$ and finds an induced planar subgraph of at least $3n/(d+1)$ vertices, which implies a performance ratio of at least $3/(d+1)$. This performance ratio improves on that of the Halldórsson-Lau algorithm for cases where $d \not\equiv 2 \pmod 3$. Furthermore, the subgraphs produced by Edwards and Farr are not constrained to having maximum degree at most two. This algorithm partitions $V = V(G)$ into two sets, P (the vertices in the induced planar subgraph) and $R = V \setminus P$. Initially P is empty and $R = V$. Vertices are incrementally added to P (and removed from R) whilst maintaining the planarity of $\langle P \rangle$. In some cases a vertex in P is interchanged with a vertex in R. The restrictions on selection of vertices for inclusion into P are stricter than required to maintain planarity, but allow certain properties in the graph to be maintained which enable the performance of the algorithm to be analysed. See [6, 7] for further information.

The subgraphs produced by this algorithm are not necessarily maximal. The authors note that in some cases after the algorithm has stopped, it is possible to add some additional vertex to P whilst maintaining planarity [7]. Edwards and Farr comment that although it is clear that their lower bound is not tight, the difference between it and the actual value is unknown.

2.3 The Vertex Removal Algorithm

Edwards and Farr [8, 9] also provided another algorithm for finding large induced planar subgraphs for a wider class of graphs, namely those of *average* degree at most $\bar{d}$. This algorithm will be referred to as the Vertex Removal algorithm. It finds an induced subgraph of at least $3n/(\bar{d}+1)$ vertices for a graph with average degree at most $\bar{d} \geq 4$ or a graph that is connected and has average degree at most $\bar{d} \geq 2$. This algorithm also partitions the vertices into two sets, P and R, but in this case initially $P = V$ and $R = \emptyset$.

The graph is reduced by either removing any vertex of degree at most one, or by removing any vertex v of degree two and its incident edges, uv and vw, and inserting a single edge uw if such an edge is not already in the graph. These operations are performed iteratively until neither operation can be applied to the graph. This process is said to produce a *reduced graph.* At each iteration (a) a vertex of highest degree in the reduced graph is removed and (b) all possible reductions are performed on the graph $\langle P \rangle$. To avoid the cost of planarity testing at each iteration, Edwards and Farr use a loop condition that a vertex is to be removed from the reduced graph while the number of removed vertices is less than a value ρ which they calculate to be an upper bound on the size of the smallest set X of vertices of G such that $G - X$ is planar.

Faria, Figueiredo, Gravier, Mendonça and Stolfi [10] have presented an algorithm similar to the Vertex Removal algorithm. However, this algorithm is limited to graphs of maximum degree (and thus average degree) at most 3. The performance ratio of this algorithm is 3/4 which is the same as that achieved by the Vertex Removal algorithm for graphs of average degree at most 3.

2.3.1 The Vertex Subset Removal Algorithm

A modification of the Vertex Removal algorithm called the Vertex Subset Removal algorithm was also implemented. This algorithm was based on private communication from Edwards and Farr. Their algorithm was modified to select the vertex $v \in P$ with the largest number of neighbours of degree lower than the degree of v in the reduced graph. This is the first MIPS algorithm to consider ordering of selection of vertices. Liebers [14] comments that there is no known investigation of the impact of different vertex orderings on the size of subgraph produced. The comparison of the behaviour of the Vertex Removal and the Vertex Subset Removal algorithms provides some indication of the desirability of such a strategy.

2.4 Other Algorithms

With NP-hard problems there is usually a trade-off between quality of solution and time taken to produce the solution. Sometimes naive algorithms can provide a reasonable solution in less time than a more sophisticated algorithm. In the experiments undertaken two such algorithms were implemented: one to find a maximal independent set and the other to find a maximal induced forest. A comparison of the time taken and the size of subgraph produced by these algorithms with those produced by the approximation algorithms for MIPS provides an indication of what is gained by using more sophisticated methods and at what cost in terms of time.

By Turán's theorem [2, p.81], a maximal independent set has size at least $n/(d+1)$, which is 1/3 of the best known lower bound on the size of an induced planar subgraph. In our experiments we use a standard sequential algorithm that finds a maximal independent set. Initially, the vertices are sorted into ascending order of degree. Using bucket sort with at most $n-1$ buckets this can be done in $O(n)$ time. Each vertex is examined and added to P if it is not adjacent to any vertex in P. Thus, a maximal independent set can be found in time $O(n)$.

Alon, Mubayi and Thomas [1] showed that a graph of average degree $\bar{d}$ has a maximal induced forest of size at least $2n/(\bar{d}+1)$, which is 2/3 of the best known lower bound for MIPS. The algorithm implemented to find a maximal induced forest selects vertices in increasing order of degree. Initially, the vertices are sorted into ascending order of degree, which can be done in $O(n)$ time. Each vertex is examined and added to P if it has at most one neighbour in any component in $\langle P\rangle$. If d is the maximum degree of the graph, this takes time $O(nd)$. Thus the total time taken to determine which vertices can be added to P and update the components of $\langle P\rangle$ is at most $O(nd)$. Thus, a maximal induced forest can be found in $O(nd)$ time.

3 New Algorithms

In this section some new algorithms for finding a large induced planar subgraph are introduced, including algorithms for finding a large induced outerplanar subgraph. The performance of one of these algorithms is analysed mathematically, and a lower bound of $3n/(d+5/3)$ on its performance ratio is established. This is close to the bound of $3n/(d+1)$ for the existing algorithms [7, 8, 9] for MIPS. Furthermore, at least one additional vertex can be added to an induced outerplanar subgraph whilst maintaining planarity (unless the original graph is outerplanar). In most cases the size of a maximum induced planar subgraph is larger than the size of a maximum induced outerplanar subgraph. However, in the experiments performed the size of the induced outerplanar subgraph found was close to the largest induced subgraph found by any of the MIPS algorithms.

Our first two algorithms form an outerplanar subgraph from a maximal induced forest. They both incrementally add vertices whilst maintaining outerplanarity. In the second algorithm a vertex in the planar set is sometimes interchanged with a vertex in the non-planar set. This algorithm allows the size of the outerplanar subgraph produced to be analysed more easily.

A third algorithm is presented that produces induced planar subgraphs with a more restricted structure which we call a palm tree.

None of the existing algorithms, or the algorithms discussed so far in this section, add vertices that introduce K_4 minors into the induced subgraphs. Furthermore these algorithms usually only add vertices with few neighbours in P. An algorithm for an operation that identifies some of the vertices whose addition may produce an induced planar subgraph containing a K_4 minor was also designed (see Section 3.3). In our experiments, the effects of combining this operation with some of the existing MIPS algorithms were observed.

3.1 Algorithms for Finding Large Induced Outerplanar Subgraphs

In this section two algorithms for finding a large induced outerplanar subgraph are introduced. The second of these algorithms can be shown to find an induced outerplanar subgraph containing at least $3n/(d + 5/3)$ vertices. An analysis of the time complexities of these algorithms is included. This analysis assumes the use of the most efficient data structures and sorting/searching algorithms. These may not be those used in the actual implementation of the algorithms.

In Section 4.2 the performance of these algorithms is compared to that of the previous algorithms. The comparison of size of subgraph produced by these algorithms gives some indication of the improvement in size of subgraph achieved by using an algorithm to find a planar subgraph in comparison to an algorithm that finds an outerplanar subgraph.

3.1.1 Outerplanar Algorithm 1 (OP1)

The OP1 algorithm (see Algorithm 1) initially finds a maximal induced forest. It then identifies certain vertices with at most two neighbours in any component in $\langle P \rangle$ that can be added to P whilst maintaining outerplanarity.

A *block* is a maximal connected subgraph containing no cutvertex. Proposition 2 uses the following fact about outerplanar graphs:

Fact 1 *A graph G is outerplanar if and only if all blocks of G are outerplanar.*

Proposition 2 *Algorithm OP1 finds an induced outerplanar subgraph.*

Proof: Initially, the algorithm finds a maximal induced forest, which is an outerplanar subgraph. At each loop, Algorithm OP1 adds a vertex v which has at most two neighbours in any single component to P. If vertex v has at most one neighbour in any component of an outerplanar graph $\langle P \rangle$, then it is plain that $\langle P \cup \{v\} \rangle$ is outerplanar. Thus it suffices to show that if $\langle X \rangle$ is an outerplanar component of $\langle P \rangle$, and v has two neighbours in X, then $\langle X \cup \{v\} \rangle$ is outerplanar.

```
**Input**: A graph $G = (V, E)$ of maximum degree at most $d$
$P := \emptyset$,
$R := V$.
Find a maximal induced forest $F$: $P := V(F)$,    $R := V \setminus P$.
**while** $\exists v \in R$ *satisfying the following criteria:*
• *Vertex $v$ has at most two neighbours in any single component in $\langle P \rangle$*
• *In each component $\langle X \rangle$ of $P$ in which $v$ has two neighbours, $w_1$ and
$w_2$, there exists a $w_1$–$w_2$ path $T$, such that at most one edge of $T$ belongs
to a cycle of $\langle P \rangle$, and such an edge (if it exists) belongs to only one cycle
of $\langle P \rangle$*
**do**
    Select the vertex $v \in R$ having the lowest $d_V(v)$ satisfying the above
    criteria;
    $P := P \cup \{v\}$;
    $R := R \setminus \{v\}$;
**end**
**Output**: $P$
```

Algorithm 1: Outerplanar Algorithm 1 (OP1)

Let w_1 and w_2 be the neighbours of v in X, and let T be a w_1–w_2 path in $\langle X \rangle$.

Now $\langle X \rangle$ is an outerplanar graph, so its blocks are outerplanar (see Fact 1). Consider the blocks of $\langle X \cup \{v\} \rangle$.

If T contains no edge that belongs to any cycle in $\langle X \rangle$, then the blocks in $\langle X \cup \{v\} \rangle$ are the block $B = (T \cup \{v\}) + vw_1 + vw_2$ and all the blocks of $\langle X \rangle$ that are not just edges of T. The block B is outerplanar as it is a cycle, and all the blocks of $\langle X \rangle$ are outerplanar, so $\langle X \cup \{v\} \rangle$ is outerplanar.

Alternatively, suppose T contains an edge e that belongs to a cycle C. Since e belongs to no other cycle of $\langle X \rangle$, C is in fact a block of $\langle X \rangle$. Then the blocks in $\langle X \cup \{v\} \rangle$ are the block $B + C$, formed from B and C, and all the blocks of $\langle X \rangle$ that are not just edges of T. The block $B + C$ is a *theta graph*, that is, a graph consisting of three internally disjoint paths with the same two distinct endpoints. One of these three paths is a single edge in the block $B + C$, so it is outerplanar.

As all blocks in $\langle X \cup \{v\} \rangle$ are outerplanar, $\langle X \cup \{v\} \rangle$ is outerplanar. □

Proposition 3 *Algorithm OP1 has time complexity $O(mn)$.*

Proof: Let d be the maximum degree of the input graph.

Algorithm OP1 consists of two phases. In the first phase a maximal induced forest is found in time $O(nd)$ (see Section 2.4).

The loop in Algorithm OP1 then adds vertices to P whilst maintaining outerplanarity. There are at most $n(d-1)/(d+1)$ vertices in R at the commencement of the loop. At each iteration the algorithm checks if a path satisfying the criteria in the algorithm exists. The path is found by a modified depth first search technique. A valid path contains at most one edge belonging to a cycle in $\langle P \rangle$. If in any branch of the search tree a second edge belonging to a cycle is encountered, no further edges in this branch are examined. Thus, no edge can be examined more than once. Thus the search takes $O(m)$ time and the loop takes $O(mn)$ time.

As $m > d$, the total time taken by the algorithm is $O(mn)$. $\square$

3.1.2 Outerplanar Algorithm 2 (OP2)

Algorithm OP2 (see Algorithm 2) also initially finds a maximal induced forest. It then identifies additional vertices that can then be added whilst maintaining outerplanarity. However, Algorithm OP2 also interchanges some vertices in P and R in order to ensure that when the algorithm stops, all vertices in R have at least three neighbours in components of size at least three in $\langle P \rangle$. This enables a lower bound on the size of subgraph produced by this algorithm to be determined, but incurs an additional running cost of identifying such vertices for removal.

Proposition 4 *Algorithm OP2 finds an induced outerplanar subgraph.*

Proof: Initially P is the vertex set of a maximal induced forest, so $\langle P \rangle$ is outerplanar. In the first loop of Algorithm OP2, a vertex v is added if, in each component in $\langle P \rangle$, the neighbours of v lie on a unique path. It is sufficient to show that such a vertex can be added to a single component in $\langle P \rangle$ whilst maintaining outerplanarity.

As a vertex v with at most one neighbour in any component can easily be added to P and maintain outerplanarity, we will consider the case where v has at least two neighbours in component $\langle X \rangle$ where $X \subseteq P$.

Let T be the path containing the neighbours $w_1, w_2, \dots w_k$ of v in $\langle X \rangle$.

Now $\langle X \rangle$ is an outerplanar graph. As no two vertices in T are connected by any other path, T contains no edge that forms part of a cycle in $\langle X \rangle$. Thus the blocks in $\langle X \cup \{v\} \rangle$ are the block $B = (T \cup \{v\}) + vw_1 + vw_2 + \dots + vw_k$ and all the blocks of $\langle X \rangle$ that are not just edges of T. It is clear that B is outerplanar. Thus all the blocks of $\langle X \cup \{v\} \rangle$ are outerplanar.

Thus, when the first loop of Algorithm OP2 is completed, $\langle P \rangle$ is outerplanar.

The second loop in Algorithm OP2 deals with vertices in R that have at most two neighbours in Q. It is clear that a vertex with at most one neighbour in any non-tree component and at most two neighbours in any tree component can be added to P whilst maintaining outerplanarity. We need only show that a vertex v with two neighbours, say w_1 and w_2, in some non-tree component $\langle X \rangle$ can be added to the component whilst maintaining outerplanarity.

We showed in the proof of Proposition 2 that if there exists a unique w_1–w_2 path in $\langle X \rangle$, then v can be added to this component whilst maintaining outerplanarity. It remains to prove that v can be added to X when the w_1–w_2 path S is not unique.

Suppose for one of the neighbours of v in X, say w_1, there exists a path $w_1, \dots, x$ that is disjoint from S except for the endpoints. If w_1 is removed from

```
Input: A graph $G = (V, E)$
$P := \emptyset$,
$R := V$.
Find a maximal induced forest $F$: $P := V(F)$,    $R := V \setminus P$.
while $\exists v \in R$ such that for each component $\langle X \rangle$, where $X \subseteq P$, either $v$
has no neighbours in $\langle X \rangle$, or the neighbours of $v$ in $\langle X \rangle$ lie along a
unique path $T$ in $\langle X \rangle$ (i.e., there is no other path connecting any two
vertices in this path) do
    $P := P \cup \{v\}$;    $R := R \setminus \{v\}$;
end
Let $Q \subseteq P$ denote the set of vertices belonging to components of size at
least three in $\langle P \rangle$.
while $\exists v \in R$ such that $v$ has at most two neighbours in $\langle Q \rangle$ do
    if vertex $v$ has at most one neighbour in $\langle Q \rangle$ then
        $P := P \cup \{v\}$;    $R := R \setminus \{v\}$;
    else if $v$ has two neighbours, $w_1, w_2$, in some non-tree component
    $\langle X \rangle$ of $\langle Q \rangle$ (and the other neighbours of $v$ in $P$ are in components of
    size at most 2) then
        Let $S$ be a $w_1$–$w_2$ path in $\langle X \rangle$.
        if $S$ is the unique $w_1$–$w_2$ path in $\langle P \rangle$ then
            $P := P \cup \{v\}$;    $R := R \setminus \{v\}$;
        else
            Then there exists a $w$–$x$ path in $\langle P \rangle$ that is disjoint from $S$
            except for the endpoints $w$ and $x$.
            if such a path exists with $w_1$ as an endpoint then
                $w = w_1$
            else if such a path exists with $w_2$ as an endpoint then
                $w = w_2$
            else
                Let $w$ be the first vertex on $S$ for which such a path exists
            end
            $P := (P \cup \{v\}) \setminus \{w\}$;
            $R := (R \cup \{w\}) \setminus \{v\}$;
        end
    else if $v$ has at most one neighbour in any non-tree component and at
    most two neighbours in the tree components in $\langle Q \rangle$ then
        $P := P \cup \{v\}$;    $R := R \setminus \{v\}$;
    end
end
Output: $P$
```

Algorithm 2: Outerplanar Algorithm 2 (OP2)

X, $\langle X - \{w_1\}\rangle$ is outerplanar. Furthermore, vertex v has only one neighbour in $X \setminus \{w_1\}$ and so $\langle X \setminus \{w_1\} \cup \{v\}\rangle$ is outerplanar.

Alternatively suppose that w is the first vertex on S such that a w–x path exists in $\langle X\rangle$ which is disjoint from S except for the endpoints. Let $w_1, \ldots, w_i, w, \ldots, w_2$ be this w_1–w_2 path. As no path exists from any vertex in $\{w_1, \ldots, w_i\}$ to any vertex in $\{w, \ldots, w_2\}$, $\langle X \setminus \{w\}\rangle$ has two components. As vertex v has one neighbour in each of these components, v can be added to $X \setminus \{w\}$ whilst maintaining outerplanarity.

Thus if the two neighbours, w_1 and w_2, of v in component $\langle X\rangle$ lie on a non-unique w_1–w_2 path, there exists a vertex $w \in X$ such that $\langle X \setminus \{w\} \cup \{v\}\rangle$ is outerplanar.

Thus Algorithm OP2 finds an induced outerplanar subgraph. □

Proposition 5 *Algorithm OP2 has time complexity $O(m^2 n)$.*

Proof: Let d be the maximum degree of the input graph.

Algorithm OP2 initially finds a maximal induced forest, which can be found in time $O(nd)$ (see Section 2.4).

In the first loop each of the vertices in R are examined and those that satisfy the criteria are added to P. There are at most $n(d-1)/(d+1)$ vertices in R. A path containing the neighbours of $v \in R$ in a component $\langle X\rangle$ is found by initially having a path of length one consisting of a single neighbour of v. The path is then extended (at either end) including additional neighbours of v when possible until all the neighbours are in the path. This can be done in time $O(m)$. Suppose the path T cannot be extended to include all the neighbours of v, but there exists a path S that does contain all the neighbours of v. As T and S differ in at least one subpath, there exists at least two vertices in S that are connected by more than one path. Thus the neighbours of v do not lie on a unique path (in the sense required by the loop condition). Thus if the path cannot be extended to include all the neighbours of v then either no such path exists, or the path containing the neighbours of v is not unique.

If a path containing all the neighbours of v is found it may not be unique. In order to show that the path T is not unique (in the sense demanded by the first loop), another path that is not a subpath of T must be found that connects two vertices belonging to the path. For each edge $w_i w_{i+1}$ on the path, the algorithm attempts to find an alternative path in $\langle X\rangle \setminus \{w_i w_{i+1}\}$ from w_i to a vertex that occurs further along the path T. If such a path exists, then T is not unique. As there are at most m edges in T and for each edge an alternative path, if one exists, can be found in time $O(m)$, the total time to check if an alternative path exists is $O(m^2)$. For each of the at most n vertices in R the first loop finds a path in $O(m)$ time, then searches for an alternative path in $O(m^2)$ time. Thus the first loop takes time $O(m^2 n)$.

Let $Q \subseteq P$ be the set of vertices belonging to components of size at least three in $\langle P\rangle$. At each iteration of the second loop a vertex v with at most two neighbours in Q is added to P, and in some cases a vertex w with at least three neighbours in Q is removed from P. Thus, each iteration decreases the number of vertices in R with at most two neighbours in Q.

A path S connecting the two neighbours of v in Q, w_1 and w_2, must be found at each iteration. In the worst case, this involves checking at most $|E(\langle P\rangle)| < m$ edges. The algorithm then determines if another w_1–w_2 path can be found. For each e of the m edges on the path S the algorithm finds, if one exists, an alternative w_1–w_2 path in $\langle X\rangle \setminus \{e\}$ in time $O(m)$. Thus for each of the at most n vertices in R the algorithm finds a path S in time $O(m)$ and an alternative path in time $O(m^2)$. Thus, the second loop takes time $O(m^2 n)$.

Thus the total time taken by Algorithm OP2 is $O(m^2 n)$. □

Theorem 6 *For any graph of n vertices and maximum degree at most $d \geq 2$, Algorithm OP2 finds an induced outerplanar subgraph of $\geq 3n/(d+5/3)$ vertices.*

Proof: Let P be the set of vertices that induce the subgraph produced by the OP2 algorithm. When the algorithm is completed, every $v \in R$ has at least three neighbours in some component of size at least three, otherwise it would have been added to P.

Let $Q \subseteq P$ where Q consists of all vertices belonging to a component of size at least three in $\langle P\rangle$. Let $n_1 = |P|$ and let K_Q be the set of components of $\langle Q\rangle$.

For all $v \in R$, we have $d_Q(v) \geq 3$, so

$$|E(Q,R)| \geq 3|R| = 3(n-n_1) \tag{1}$$

Each component of size k in $\langle Q\rangle$ contains at least $k-1$ edges, so

$$|E(\langle Q\rangle)| \geq (\sum_{H_i \in K_Q} (|V(H_i)| - 1) = |Q| - |K_Q|. \tag{2}$$

As all components in Q have size at least three, $|K_Q| \leq |Q|/3$. So

$$|E(\langle Q\rangle)| \geq |Q| - |Q|/3 = 2|Q|/3. \tag{3}$$

Now,

$$\begin{aligned} |E(Q,R)| &= \sum_{v\in Q} d_{Q\cup R}(v) - 2|E(\langle Q\rangle)| \\ &\leq d|Q| - 2|E(\langle Q\rangle)| \\ &\leq (d-4/3)|Q| \quad \text{by (3)} \\ &\leq (d-4/3)|P|. \end{aligned} \tag{4}$$

From (1) and (4),

$$(d-4/3)n_1 \geq 3(n-n_1).$$

Therefore

$$(d+5/3)n_1 \geq 3n.$$

So

$$n_1 \geq 3n/(d+5/3).$$

□

3.2 Palm Trees

In the previous section algorithms for finding large induced outerplanar subgraphs were considered. In this section an algorithm that produces a tree-like structure is considered.

A *palm tree* is a connected graph which is either a cycle or can be constructed from the union of a tree T with an independent set X by joining each vertex in $v \in X$ to two adjacent vertices in T (these two vertices need not be the same for all $v \in X$). A *palm forest* is a graph all of whose components are palm trees.

If $\langle P \rangle$ is a maximal forest in G, then every vertex in R has at least two neighbours in some forest component in $\langle P \rangle$, otherwise P is not maximal. Thus, for all $v \in R$, $d_P(v) \geq 3$ or $d_P(v) = 2$ and the neighbours in P are in a single component of $\langle P \rangle$. All of the existing MIPS algorithms discussed in Section 2, except for the Halldórsson-Lau algorithm, guarantee that for all $v \in R$, $d_P(v) \geq 3$ when the algorithm ceases. On termination the Palm Tree algorithm also guarantees this condition, but the subgraph produced has a similar structure to a tree.

The Palm Tree Algorithm (see Algorithm 3) initially finds a maximal induced forest $\langle P \rangle$. It then considers all vertices $v \in R$ with $d_P(v) = 2$ and neighbours $w_1, w_2 \in P$. If w_1 and w_2 are adjacent, then the subgraph induced by $\{v, w_1, w_2\}$ is a cycle of length three. Any such vertex can always be added to P whilst maintaining planarity. If the neighbours of v are not adjacent then there exists a w_1–w_2 path S connecting these vertices in $\langle P \rangle$. (If no such path exists, vertex v has degree 2 and is adjacent to two components, which contradicts the maximality of the induced forest.) Now, if the component in $\langle P \rangle$ containing the neighbours of v consists solely of the path S, then vertex v can be added to P and the resulting component is a cycle. If this is not the case, $(S \cup \{v\}) + vw_1 + vw_2$ is a cycle of length at least four containing at least one vertex with at least three neighbours in $P \cup \{v\}$.

If one of the neighbours of v, say w_1, has at least two neighbours in $\langle P \rangle$, then w_1 will have at least three neighbours in $(P \cup \{v\}) \setminus \{w_1\}$. Alternatively, if both the neighbours of v have only one neighbour in $\langle P \rangle$ and the component does not consist solely of a path, then there exists some vertex along the path S that has at least three neighbours in P. Thus, a vertex exists in the component that can be interchanged with v whilst maintaining the planarity of $\langle P \rangle$. Interchanging vertex v with such a vertex decreases the number of vertices with $d_P(v) \leq 2$ in R. Thus when the Palm Tree algorithm terminates, all vertices in R have at least three neighbours in components of size at least three in $\langle P \rangle$.

Proposition 7 *The Palm Tree Algorithm has time complexity $O(n^2)$.*

Proof: The Palm Tree Algorithm initially finds a maximal induced forest in time $O(nd)$ (see Section 2.4). Each vertex $v \in R$ with two neighbours in P is then added to P. In some cases a vertex is removed from P when v is added. The removed vertex w has at least three neighbours in $P \cup \{v\} \setminus \{w\}$. Thus each iteration reduces the number of vertices in R with degree at most two.

```
Input: A graph $G = (V, E)$
$P := \emptyset$,
$R := V$.
Find a maximal induced forest $F$: $P := V(F)$,    $R := V \setminus P$.
while $\exists v \in R$ such that $d_P(v) = 2$ do
    while $\exists v \in R$ such that $d_P(v) = 2$, and the addition of $v$ to $P$ creates
    a cycle of length three do
        $P := P \cup \{v\}$;    $R := R \backslash \{v\}$;
    end
    while $\exists v \in R$ such that $d_P(v) = 2$ and the addition of $v$ to $P$
    creates a component that is a cycle do
        $P := P \cup \{v\}$;    $R := R \backslash \{v\}$;
    end
    while $\exists v \in R$ such that $d_P(v) = 2$ and the addition of $v$ to $P$ creates
    a cycle of length at least four that contains at least one vertex of
    degree at least three in $\langle P \cup \{v\} \rangle$ do
        Let $w_1$ and $w_2$ be the two vertices adjacent to $v$ in $P$.
        if $d_P(w_1) \geq 2$ then
            $P := (P \backslash \{w_1\}) \cup \{v\}$;
            $R := (R \backslash \{v\}) \cup \{w_1\}$;
        else if $d_P(w_2) \geq 2$ then
            $P := (P \backslash \{w_2\}) \cup \{v\}$;
            $R := (R \backslash \{v\}) \cup \{w_2\}$;
        else
            Find the first vertex $w$ with $d_P(w) \geq 3$ on the path $w_1, ..., w_2$
            in $\langle P \rangle$.
            $P := (P \backslash \{w\}) \cup \{v\}$;
            $R := (R \backslash \{v\}) \cup \{w\}$;
        end
        while $\exists v_1 \in R$ with at most one neighbour in each component of
        size at least three in $\langle P \rangle$ do
            $P := P \cup \{v_1\}$;    $R := R \backslash \{v_1\}$;
        end
    end
    while $\exists v \in R$ such that $v$ is adjacent only to components of size at
    most two do
        $P := P \cup \{v\}$;    $R := R \backslash \{v\}$;
    end
end
Output: $P$
```

Algorithm 3: Palm Tree Algorithm (PT)

In the worst case, all vertices in R with two neighbours in P require a vertex to be removed from P to enable them to be added. This requires the path connecting the neighbours of v in P to be examined to find a candidate vertex for removal. There are at most $|E(\langle P\rangle)| \leq m \leq 3|P| \leq 3n$ edges in such a path. Thus the main (outer) loop in the Palm Tree algorithm takes time $O(n^2)$.

When this loop is completed, any vertex adjacent to only components of size at most two is added to P. This loop takes $O(|R|)$ steps. Thus the time complexity of the Palm Tree Algorithm is $O(n^2)$. □

3.3 Finding more complex planar subgraphs

Each of the above algorithms finds an induced planar subgraph with no K_4 minor. A simple example is the induced planar subgraphs of the graph K_5 found by the various algorithms. Both algorithms by Edwards and Farr and the algorithm by Halldórsson and Lau find an induced planar subgraph of size three (the same subgraph found by the algorithms for finding an induced outerplanar subgraph), rather than the maximum induced planar subgraph K_4.

```
Input: A graph $G = (V, E)$
Let $G' = (V', E')$ where: $V' = \{V_1, V_2, \ldots, V_n\}$; $V_i = \{v_i\}$ for $1 \leq i \leq n$;
$E' = E$.
while $G'$ can be reduced do
    if there exist a vertex $V_i \in V'$ with one neighbour $V_j$ in $G'$ then
    $V_j = V_i \cup V_j$;
    $V' = V' \setminus V_i$;
    $E' = E' \setminus \{V_iV_j\}$;
    else if there exist a vertex $V_i \in V'$ with exactly two neighbours, $V_j$
    and $V_k$, in $G'$ and $V_jV_k \notin E'$ then
        $V_j = V_i \cup V_j$;
        $V_k = V_i \cup V_k$;
        $V' = V' \setminus V_i$;
        $E' = (E' \cup \{V_jV_k\}) \setminus \{V_jV_i, V_iV_k\}$
    end
end
Output: $G' = (V', E')$
```
Algorithm 4: Reduce Graph

We designed an algorithm that added some vertices, in some cases, to the induced planar subgraph. If in each component X containing neighbours of a vertex v, the neighbours of v lie on some face of some embedding of X, then $\langle P \cup \{v\}\rangle$ is planar. Our algorithm, which we call the *Enlarge Planar Subgraph* (EPS) operation (see Algorithm 6), detects some (but not all) situations of this kind, and adds v to P.

Algorithm 4 reduces the induced subgraph, $\langle P\rangle$, using a subset of the reduction operations discussed in Section 2.3. The operation that we omit here

```
Input: A graph $G = (V, E)$, a set of vertices $V' \subseteq V$ to be included in
       the path, and a set of edges $E' \subseteq E$ to be included in the path.
$T := \emptyset$;
if $V' = \emptyset$ then
    Output: $T$
else
    Let $w \in V'$;
    $T = \{w\}$ ;      $V' = V' \setminus \{w\}$;
end
while $V' \neq \emptyset$ do
    Let $w_1$ be the first vertex in $T$ and let $w_2$ be the last vertex in $T$;
    if $\exists uw_1 \in E'$ and $u \notin T$ then
        Prepend $u$ to $T$;      $V' = V' \setminus \{u\}$;
    else if $\exists uw_2 \in E'$ and $u \notin T$ then
        Append $u$ to $T$;      $V' = V' \setminus \{u\}$;
    else if $\exists u \in V'$ adjacent to $w_1$ then
        Prepend $u$ to $T$;      $V' = V' \setminus \{u\}$;
    else if $\exists u \in V'$ adjacent to $w_2$ then
        Append $u$ to $T$;      $V' = V' \setminus \{u\}$;
    else if $\exists u \in V, u \notin T$ adjacent to $w_1$ then
        Prepend $u$ to $T$;
    else if $\exists u \in V, u \notin T$ adjacent to $w_2$ then
        Append $u$ to $T$;
    else
        Output: $\emptyset$
    end
end
Output: $T$
```

Algorithm 5: Find Path

```
Input: A graph $G = (V, E)$ and $P \subseteq V$ such that $\langle P \rangle$ is planar.
$R := V \setminus P$.
$\langle P' \rangle$ = Reduce Graph($\langle P \rangle$);
foreach $v \in R$ do
    foreach Component $\langle X \rangle$ in $\langle P' \rangle$ do
        Let $N_X(v)$ be the neighbours of $v$ in $X$;
        Let $M$ be the set of mandatory edges. $M = \{V_iV_j : \exists w \in N_P(v)$
        such that $w \in V_i \bigcap V_j\}$;
        $T$ = Find Path ($\langle X \rangle$, $N_X(v)$, $M$);
        if $T \neq \emptyset$ then
            if T has length at most one then
                /* $\langle X \cup \{v\} \rangle$ is planar                    */
                continue ;
            else if T has length two and the internal vertex has at most
            three neighbours in X then
                /* $\langle X \cup \{v\} \rangle$ is planar                    */
                continue ;
            else
                Let $I = \{1, \ldots, |V(T)|\}$ and let $S$ be the sequence
                $(s_i : i \in I)$ where $s_i = d_X(t_i)$;
                i=2;
                while $i \leq |V(T) - 1|$ do
                    if $s_i = 2$ then
                        Remove $s_i$ from $S$;
                        Decrement the values $s_{i-1}$ and $s_{i+1}$ by 1;
                    end
                    i=i+1;
                end
                if the sequence S (excluding its first and last elements)
                contains at most one 3 and all other values are 2 then
                    /* $\langle X \cup \{v\} \rangle$ is planar                */
                    continue;
                else
                    reject $v$ (so it is not included in $P$);
                    break;
                end
            end
        else
            reject $v$ (so it is not included in $P$);
            break;
        end
        $P := P \cup \{v\}$;
        $R := R \setminus \{v\}$;
        $\langle P' \rangle$ = Reduce Graph($\langle P \rangle$);
    end
end
Output: $P$
```

Algorithm 6: EPS Algorithm

is the one that removes a vertex v of degree two and its incident edges, uv and vw, when the edge uw is already in the graph. Note that the reduction operations we retain here do not change the number of cycles in the graph, while the omitted operation reduces the number of cycles. Suppose $\langle P' \rangle$ is formed from $\langle P \rangle$ by applying reduction operations from this subset. Then $\langle P' \rangle$ preserves the cycle structure of $\langle P \rangle$ in the sense that every cycle in $\langle P \rangle$ is a subdivision of a cycle in $\langle P' \rangle$. Each vertex V_i in the reduced graph $\langle P' \rangle$ consists of a set of vertices in the original graph $\langle P \rangle$ that have been combined by the reductions.

At each iteration Algorithm 6 reduces the induced subgraph $\langle P \rangle$ to $\langle P' \rangle$ using Algorithm 4. The neighbourhood $N_{P'}(v)$ in P' of a vertex $v \in R$ is the set of $V_i \in P'$ that contain the neighbours of v in P. For each component $\langle X \rangle$ in $\langle P' \rangle$, $N_X(v) \subseteq N_{P'}(v)$ contains the neighbours of v in component X. At each iteration (i.e. for each $v \in R$), for each component $\langle X \rangle$ of $\langle P' \rangle$ the algorithm tries to find a path T containing these neighbours of v with the added constraint that if any neighbour of v in P is included in two vertex sets in P', say V_i and V_j, then T must contain the edge V_iV_j (which we call *mandatory*). In order to minimize the cost of searching for such a path, the EPS operation uses a simple local search strategy (see 5). At each iteration of the Find Path algorithm, a vertex in $N_X(v)$ is added to the path if possible, with preference given to any vertex whose addition adds one of the mandatory edges to T. If no vertex in $N_X(v)$ can be added to either end of the path, then if possible a vertex in P' that is not a neighbour of v can be added to an end of T. Note that this process might not find a path T containing all $N_X(v)$, even if one exists. If a path containing all $N_X(v)$ is not formed by this process, we continue with the next vertex in R.

As $\langle P' \rangle$ preserves the cycle structure of $\langle P \rangle$, $\langle P \cup \{v\} \rangle$ is planar if and only if $\langle P' \cup \{v\} \rangle$ is planar. Suppose each edge of T belongs to at most one triangle of $\langle P' \rangle$, and that every such triangle contains a vertex of degree 2. Suppose further that at most one edge outside these triangles meets an internal vertex of T, and (if such an edge exists) meets only one such vertex. Then $\langle P' \cup \{v\} \rangle$ is planar, and so is $\langle P \cup \{v\} \rangle$. If a path T is found that satisfies these conditions for each component X, the EPS operation adds the vertex v to P.

The EPS operation was combined with the PT, OP1 and Vertex Subset Removal algorithms, by running it on the induced planar subgraphs found by those algorithms, in order to draw some conclusions about the subgraphs found by those algorithms.

4 Experimental Results

4.1 Method

The behaviour of the algorithms was observed on d-regular graphs and graphs of expected average degree $\bar{d}$. Graphs of n = 20, 40, 60, 80, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000 and 10000 vertices were generated. For each such n, random graphs of expected average degree 3, 4, 5, 6, 7, 8 and 9 and random regular graphs of these same degrees were produced. Experiments were performed on 50 graphs of each type for $n \leq 1000$, and on 20 graphs of each type for $n > 1000$.

The classical method as described in [2, p.143] was used to produce random graphs of expected average degree $\bar{d}$ (so edge probability $p = \bar{d}/(n-1)$). The random d-regular graphs were produced by both the Bollobás [3] and Steger-Wormald [17] algorithms. While Bollobás' method generates graphs randomly with a uniform probability distribution, the Steger-Wormald method has a distribution that is approximately uniform and tends to uniform as $n \to \infty$. As a single iteration of the Bollobás method produces a d-regular simple graph with probability $e^{(1-d^2)/4}$ (see [18]), it was not feasible to use this method to produce large numbers of graphs. Thus in the experiments undertaken, the Bollobás method was used to produce graphs only for smaller values of n. In contrast the Steger-Wormald method generates d-regular graphs quickly. A comparison of the results on regular graphs of the same size and degree but generated by these different methods produced extremely close results. The closeness of these two results gives support to the validity of using the Steger-Wormald generated graphs for the larger sized graphs and those of higher degree, and using the results to draw conclusions about the behaviour of induced planar subgraph algorithms on random regular graphs.

Each induced planar subgraph algorithm was run on a series of graphs. For each combination of size, degree and graph generation method, the average size of subgraph produced and the average time taken was recorded. The algorithms were also run on the graphs from the GDT-test-suite-CU. As this suite contains sets of graphs of varying maximum degree of order 10 to 100, the average proportion of vertices in the planar subgraph for each n was recorded, and the average proportion of vertices in the planar subgraph for a given average degree within an interval of 0.1. Details about the computer on which all tests were run are listed in Table 1. The Halldórsson-Lau and Vertex Addition algorithms are designed to be used on graphs of maximum degree at most d. However, the experimental results also contain observations of their behaviour on graphs of expected average degree d. When considering these results, it should be noted that such graphs are likely to have maximum degree greater than d.

Computer:	Dell Inspiron 9300 Laptop
Processor:	Intel Pentium M processor 2.00 GHz
Speed:	1.95 GHz
Memory (RAM):	1024 MB
Operating System:	Linux Mandrake 9.2
Compiler:	gcc 3.3
Random Number Generator	drand48 (stdlib)

Table 1: Details of computer used for running algorithms in tests

4.2 Discussion of Results

A complete set of all results is available at `http://www.csse.monash.edu.au/~kmorgan/MIPS.html`. Tables containing the average proportion of vertices in the induced planar subgraph, the average size of induced planar subgraph and the average elapsed time taken are provided. The standard deviation is included in parentheses for each average size (or time) given. Table 2 provides a list of abbreviations used for MIPS algorithms.

Abbreviation	Algorithm
IS	Maximal Independent Set
T	Maximal Induced Forest
HL	Halldórsson-Lau Algorithm [11]
OP1	**Outerplanar Algorithm 1**
OP2	**Outerplanar Algorithm 2**
PT	**Palm Trees Algorithm**
VA	Vertex Addition Algorithm [7]
VR	Vertex Removal Algorithm [8, 9]
VSR	*Vertex Subset Removal Algorithm*
OP1+EPS	**Outerplanar Algorithm 1 combined with EPS operation**
PT+EPS	**Palm Tree Algorithm combined with EPS operation**
VSR+EPS	*Vertex Subset Removal Algorithm combined with EPS operation*

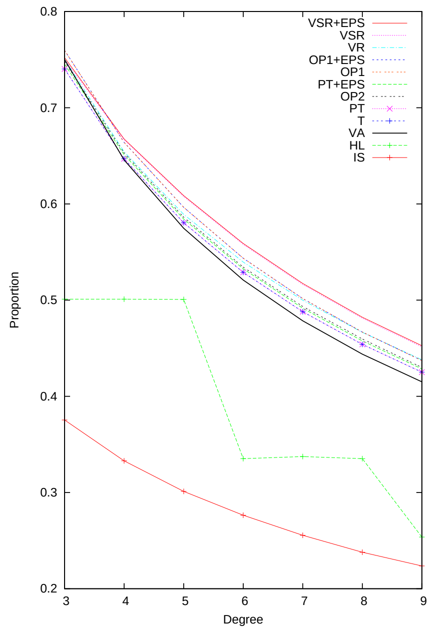
Rows in **bold** type indicate algorithms designed by the authors. Rows in *italics* indicate algorithms based on modifications of (or combinations that include) existing algorithms.

Table 2: Key to abbreviations for algorithm names

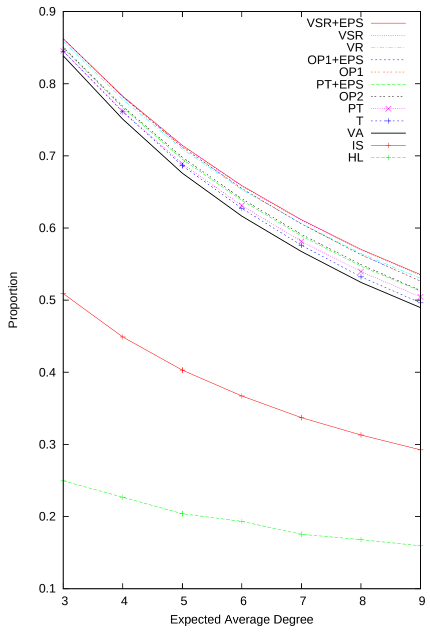
4.2.1 Size of Subgraph

In this section the behaviour of the algorithms in terms of the size of induced planar subgraph produced will be considered.

Standard Deviation The standard deviation of the size of subgraph found for random d-regular graphs is remarkably small, usually less than 1%. The standard deviation was higher for subgraphs of graphs produced by the classical method, but still well under 1% for large graphs except for the results of the HL algorithm where the standard deviation was as high as 8%. The results of the tests on the graphs of average degree from the GDT-test-suite-CU also exhibited a small standard deviation, usually less than 4%.



(a)



(b)

Figure 1: Proportion of vertices in induced planar subgraphs of random graphs of 10,000 vertices produced by (a) the Steger-Wormald method (d-regular graphs) and (b) the classical method (graphs with expected average degree $\bar{d}$)

The Performance of the MIPS Algorithms Figure 1(a) displays the performance of the algorithms as d varies when run on d-regular graphs of 10,000 vertices. The HL algorithm produced subgraphs with average size close to this algorithm's worst case lower bound from [11], namely $n/2$ for $3 \leq d \leq 5$, $n/3$ for $6 \leq d \leq 8$ and $n/4$ for $d = 9$. The only algorithm producing a smaller average size subgraph was the IS algorithm.

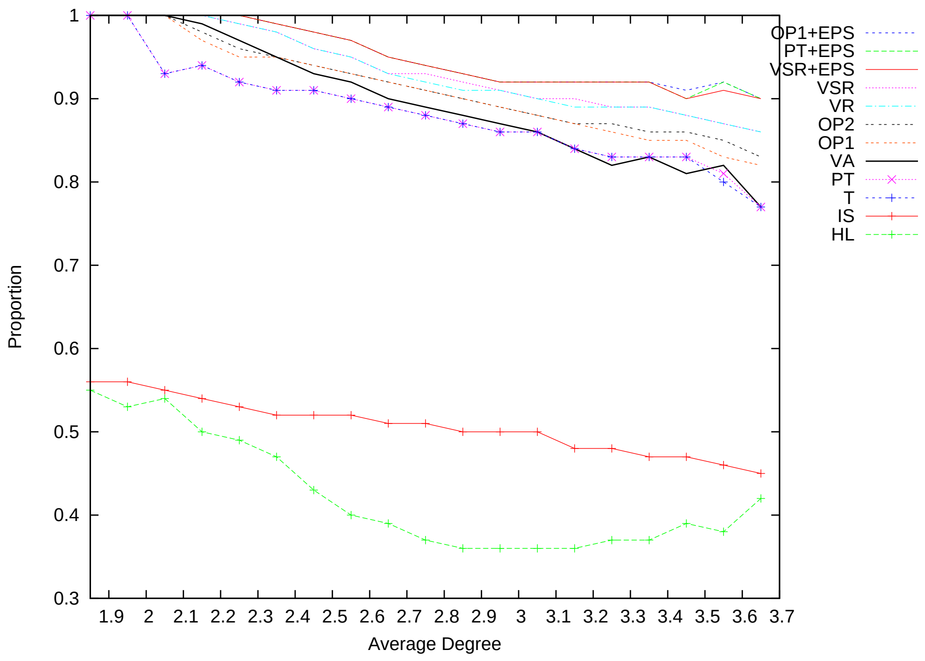


Figure 2: Proportion of vertices in induced planar subgraphs found in graphs from GDT-test-suite-CU. The average degree is given for intervals $[1.75, 1.85), [1.85, 1.95), \ldots, [3.65, 3.75)$.

The behaviour of the other algorithms as d increases in terms of the reduction in proportion of vertices in the induced subgraph is remarkably similar as can be seen in Figures 1 and 2. In Table 3 the proportion of vertices in the subgraphs found for graphs of 10,000 vertices are displayed. As n increases, the proportions of vertices in the subgraphs found by the various MIPS algorithms quickly converge to these proportions. This is typical of the behaviour of the algorithms on both d-regular graphs and graphs of expected average degree $\bar{d}$. For a fixed d, as n increases, the values rapidly converge to within 5% of the proportion found for n =10,000. Each of the algorithms converges, in this sense, on or before $n = 200$, except for the HL algorithm on graphs of expected average degree $\bar{d}$. The proportions of vertices in the subgraphs found by each of these algorithms differ by only a few percent in smaller graphs, but as n becomes large only by a fraction of a percent (see Table 3). The fact that the average sizes of subgraph produced by these algorithms are so similar even though the classes of induced subgraphs produced are not all the same may suggest that there may be some fundamental limit on the performance of algorithms for finding induced subgraphs with many hereditary properties.

Algorithm	6-regular graphs	Graphs with expected average degree 6
IS	0.2764 *(0.0014)*	0.3671 *(0.0021)*
T	0.5289 *(0.0012)*	0.6272 *(0.0022)*
HL	0.3353 *(0.0014)*	0.1933 *(0.0074)*
OP1	0.5436 *(0.0009)*	0.6547 *(0.0024)*
OP2	0.5343 *(0.0028)*	0.6405 *(0.0021)*
PT	0.5289 *(0.0011)*	0.6311 *(0.0023)*
VA	0.5209 *(0.0010)*	0.6164 *(0.0020)*
VR	0.5397 *(0.0007)*	0.6532 *(0.0020)*
VSR	0.5580 *(0.0006)*	0.6579 *(0.0020)*
OP1+EPS	0.5436 *(0.0009)*	0.6547 *(0.0024)*
PT+EPS	0.5327 *(0.0012)*	0.6382 *(0.0023)*
VSR+EPS	0.5589 *(0.0006)*	0.6587 *(0.0020)*

Table 3: Average proportion of vertices in induced planar subgraph found in randomly generated graphs of 10,000 vertices. (Standard deviations in parentheses.)

The average proportion of vertices in a subgraph found in a graph of given n and degree was larger for the graphs of expected average degree $\bar{d}$. These graphs are likely to contain some vertices with degree less than 3, which contribute to the size of subgraph produced, but do not present any obstacle to planarity.

The VSR algorithm usually produced the largest subgraphs of all MIPS algorithms (excluding algorithms combined with the EPS operation). However, the VSR+EPS algorithm usually found a larger subgraph.

In Figure 3 the results in terms of average size subgraph for a fixed number of vertices is shown for the tests on graphs from GDT-test-suite-CU. In this test suite there is no fixed maximum (or average) degree for any given n. These results also show that the algorithms combined with the EPS operation usually find a larger induced subgraph than the other algorithms.

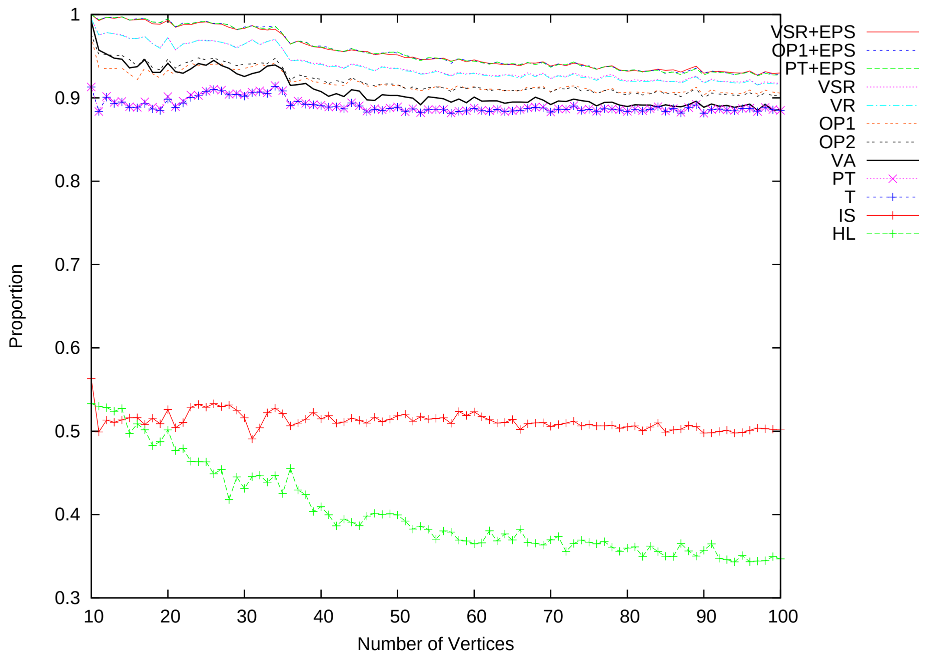


Figure 3: Proportion of vertices in induced planar subgraphs found in graphs from GDT-test-suite-CU.

Comparison between Algorithm T and other MIPS Algorithms Although the lower bound for the size of induced planar subgraph produced by the algorithm T for finding a maximal induced forest is only 2/3 of the best known lower bounds for MIPS, in practice the subgraphs produced by T and the other algorithms (VSR, VR, VA, PT, OP1 and OP2) are very close in size. The HL algorithm [11] produces an induced subgraph consisting of disjoint cycles and paths. Such a subgraph could be regarded as not much more than a linear forest: it may just have some additional vertices (and their incident edges) added to create cycle components. Yet the lower bound on the size of subgraph produced by this algorithm is considerably better than the lower bound of T (although in practice the subgraphs produced by the HL algorithm are small in comparison to the size of a maximal induced forest).

Most of the subgraphs found by the VA algorithm (see Figure 1) were smaller than the subgraphs found by the T algorithm. This may suggest the simple approach of finding a maximal induced forest provides a reasonable solution to

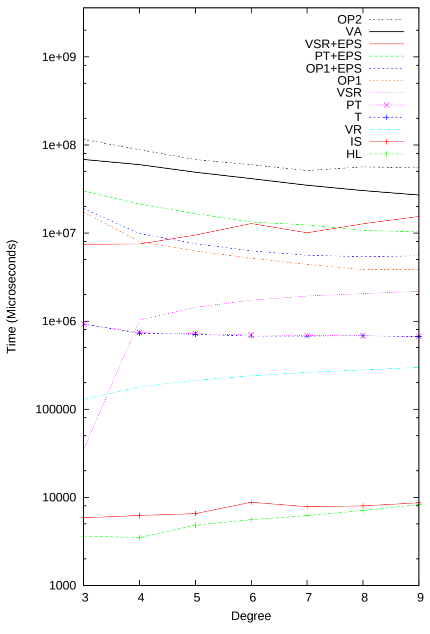
MIPS for the types of random graphs we considered. On the other hand it may indicate that the current algorithms are finding little more than a forest with a few additional vertices. This latter suggestion is supported by the similarity between the size of subgraph found by the MIPS algorithms and those found by algorithms for finding simpler structures such as outerplanar subgraphs and palm trees. The fact that the EPS operation is usually able to find additional vertices to add to subgraphs produced by these algorithms indicates that the subgraphs found are usually not maximal.

Conjectured Average Case Lower Bounds - Graphs of Maximum Degree d The maximum proportion of vertices in the induced subgraphs produced by the algorithms for d-regular graphs as d increases appears to be close to the sequence: 3/4, 6/9, 9/15, 12/22, 15/30, 18/39, 21/49, ... From this sequence it appears that the proportion of vertices in the subgraph produced for a given d by these algorithms is at least $3(d-2)/((d+1)(d+2)/2-6) = 6/(d+5)$. Now $3/(d+1) \leq 6/(d+5)$ when $d \geq 3$, so this proposed bound would exceed the existing lower bound of $3n/(d+1)$. Note that this proposed bound cannot apply to all d-regular graphs, since (for example) K_{31} contains an induced planar subgraph of size at least six. However, $6n/(d+5)$ appears to provide a reasonable guide to the proportion of vertices included in the subgraph on average, for random d-regular graphs. We conjecture that the actual lower bound for all d-regular graphs is close to $(3+((d-3)/d))/(d+1)$. This proposed lower bound suggests a subgraph of at least four vertices should be obtained from a graph of degree at most 30, so it works for K_{31}.

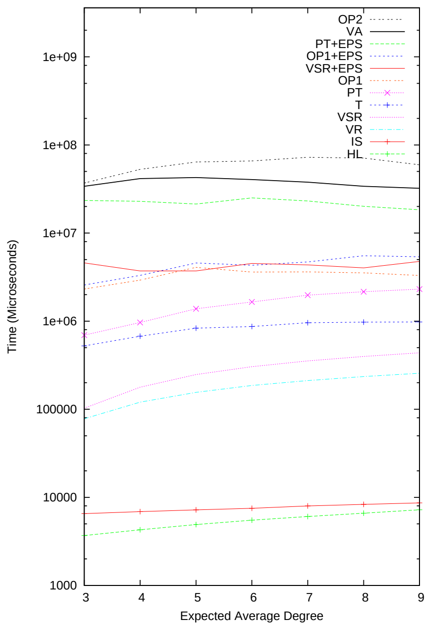
Conjectured Average Case Lower Bounds - Graphs of Average Degree $\bar{d}$ The maximum proportion of vertices in the induced planar subgraphs produced by the algorithms on graphs of expected average degree $\bar{d}$ as $\bar{d}$ increases appears to be close to the following sequence: $\sqrt{3/4}$, $\sqrt{3/5}$, $\sqrt{3/6}$, $\sqrt{3/7}$, $\sqrt{3/8}$, $\sqrt{3/9}$, $\sqrt{3/10}$, ... From this sequence it appears that the proportion of vertices in the subgraph produced for a given $\bar{d}$ by these algorithms is at least $(3/(\bar{d}+1))^{1/2}$. (A slightly closer estimated lower bound may be $(3/(\bar{d}+1+(\bar{d}-3)/\bar{d}))^{1/2}$.) Although this lower bound does not hold for d-regular graphs, graphs of average (but not maximum) degree $\bar{d}$ contain some vertices of lower degree which are more likely to be included in the induced planar subgraph. Thus, it is not surprising that the size of the induced planar subgraph is larger than that found in a d-regular graph.

4.2.2 Running Time of Algorithms

Algorithms such as the VA and OP2 performed worst in terms of running time on graphs of maximum degree d (see Figures 4(a) and 5). This reflects the cost of determining if a vertex should be removed when a vertex is added to the planar subset. The running time required by many of the algorithms for d-regular graphs with some fixed number n of vertices decreases as d increases.



(a)



(b)

Figure 4: Average Elapsed Time (microseconds) taken by algorithms on random graphs of 10,000 vertices produced by (a) the Steger-Wormald method (d-regular graphs) and (b) the classical method (graphs with expected average degree $\bar{d}$)

In some cases this may be due to there being fewer vertices that can be added when the degree is high. The VA algorithm shows this behaviour. By contrast the VSR algorithm shows a slight increase in running time as d increases. As d increases for a fixed size graph, the graph will become denser and thus is less likely to reduce as quickly as a less dense graph. Thus the location of a vertex for removal requires searching a larger reduced graph which may increase the processing time.

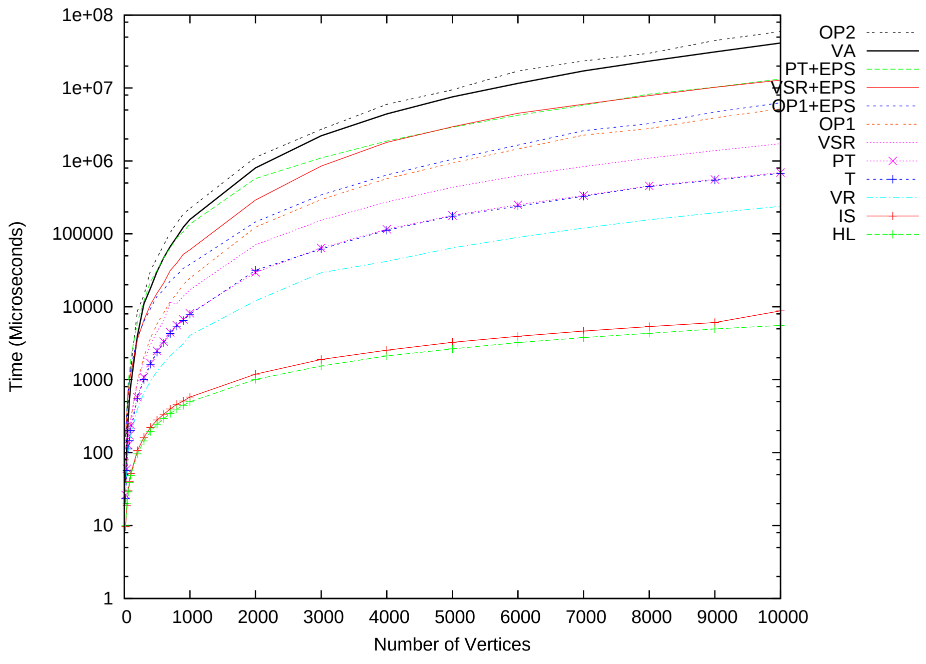


Figure 5: Average time taken by algorithms on random 6-regular graphs produced by Steger-Wormald method (elapsed time (microseconds)).

The algorithms behaved quite differently on graphs of expected average degree $\bar{d}$ (see Figure 4(b)). While the running times tended to decrease as d increased in the experiments on d-regular graphs, the running times remained more constant in the case of the graphs of expected average degree $\bar{d}$ (except in the case of the OP2 algorithm where the running times increased for $3 \leq d \leq 8$ and the VA algorithm for $d < 6$). It should be noted that the actual running times were lower than those for d-regular graphs.

The simpler algorithms such as IS and HL exhibit linear average time. Algorithms such as those using the EPS operation and the VA algorithm that include some searching for paths exhibit approximately quadratic complexity.

5 Conclusion and Further Work

Several new algorithms for the Maximum Induced Planar Subgraph and Maximum Induced Outerplanar Subgraph problems were designed. New and existing algorithms were implemented and an extensive experimental study of the behaviour of these algorithms on randomly generated graphs was undertaken. As most of the existing algorithms had not previously been implemented, the implementation of these algorithms provided new insights into how they perform in practice.

The connections between these problems and two related problems, namely Maximal Induced Forest and Maximal Independent Set, were also investigated. It was found that in practice the size of subgraph produced by existing algorithms on random graphs was similar to that produced by algorithms for finding a maximal induced forest.

One of our algorithms for finding large induced outerplanar subgraphs was analysed and shown to produce subgraphs of size at least $3n/(d+5/3)$ for graphs of maximum degree at most d. Although this lower bound is slightly less than the existing lower bound of $3n/(d+1)$ for induced planar subgraphs, experiments showed that for large n the size of the induced outerplanar subgraphs found were often larger than those produced by the existing algorithms for MIPS.

Although the observed results were close to the lower bound of $3n/(d+1)$ for $d = 3$, as d increased the lower bound was increasingly loose. The results for average proportion of vertices in the induced subgraph suggests that the lower bound on size of induced subgraph may be closer to $(3 + (d - 3)/d)/(d + 1)$.

Acknowledgments

We thank the referees for their helpful comments.

References

[1] N. Alon, D. Mubayi, and R. Thomas. Large induced forests in sparse graphs. *J. Graph Theory*, 38:113–123, 2001.

[2] N. Alon and J. Spencer. *The Probabilistic Method.* Wiley, New York, 1992.

[3] B. Bollobás. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European J. Combin*, 1:311–316, 1980.

[4] G. Di Battista, P. Eades, R. Tamassia, and I. Tollis. Algorithms for drawing graphs: an annotated bibliography. *Comput. Geom.*, 4:235–282, 1994.

[5] G. Di Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu. An experimental comparison of four graph drawing algorithms. *Comput. Geom.*, 7:303–325, 1997.

[6] K. Edwards and G. Farr. Fragmentability of graphs. *J. Combin. Theory Ser. B*, 82:30–37, 2001.

[7] K. Edwards and G. Farr. An algorithm for finding large induced planar subgraphs. In P. Mutzel, M. Jünger, and S. Leipert, editors, *Graph Drawing: 9th International Symposium, GD 2001*, Lecture Notes in Computer Science 2265, pages 75–83. Springer-Verlag, Berlin, 2002.

[8] K. Edwards and G. Farr. Planarization and fragmentability of some classes of graphs. Technical Report 2003/144, School of Computer Science and Software Engineering, Monash University, 2003. Available at http://www.csse.monash.edu.au/∼gfarr/research/publications.html.

[9] K. Edwards and G. Farr. Planarization and fragmentability of some classes of graphs. *Discrete Math.*, (to appear).

[10] L. Faria, C. de Figueirdo, S. Gravier, C. Mendonça, and J. Stolfi. Nonplanar vertex deletion: maximum degree thresholds for NP/max SNP-hardness and a 3/4-approximation for finding maximum planar induced subgraphs. *Electron. Notes Discrete Math.*, 18:121–126, 2004.

[11] M. M. Halldórsson and H. Lau. Low-degree graph partitioning via local search with applications to constraint satisfaction, max cut, and colouring. *J. Graph Algorithms Appl.*, 1:1–13, 1997.

[12] M. Hassan and G. Hogg. A review of graph theory application to the facilities layout problem. *Omega*, 15:291–300, 1987.

[13] M. Jünger and P. Mutzel. Maximum planar subgraphs and nice embeddings: practical layout tools. *Algorithmica*, 16:33–59, 1996.

[14] A. Liebers. Planarizing graphs — a survey and annotated bibliography. *J. Graph Algorithms Appl.*, 5:1–74, 2001.

[15] R. Lipton and R. Tarjan. A separator theorem for planar graphs. *SIAM J. on Appl. Math.*, 36:177–189, 1979.

[16] L. Lovász. On decomposition of graphs. *Studia Sci. Math. Hungar.*, 1:237–238, 1966.

[17] A. Steger and N. Wormald. Generating random regular graphs quickly. *Combin. Probab. Comput.*, 8:377–396, 1999.

[18] N. Wormald. Models of random regular graphs. In J. Lamb, editor, *Surveys in Combinatorics, 1999*, London Mathematical Society Lecture Note Series, 267, pages 239–298. Cambridge University Press, New York, 1999.

[19] M. Yannakakis. Node- and edge-deletion NP-complete problems. In *STOC '78: Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, pages 253–264. ACM Press, New York, 1978.