Defending Planar Graphs agains Star-Cutsets

Nicolas Sonnerat* and Adrian Vetta†

March 15, 2009

Abstract. In this paper we consider the problem of protecting edges in a network so that the network remains connected after the removal of any set of vertices that form a star consisting of unprotected edges.

We show that the problem of finding a minimal set of edges to protect is NP-complete in general graphs, and present an $O(\log n)$-approximation algorithm, where the $O(\log n)$ factor is essentially best possible.

For the special case of planar graphs, we analyse in detail the structure of minimal star-cutsets, and exploit it to construct a polynomial time approximation scheme.

1 Introduction

We consider the following simple model for network failures. A failure arises at a vertex and this causes a (possibly strict) subset of its neighbouring nodes to fail also. The initial failure could occur naturally, e.g. a breakdown in an electricity network or a virus transmitted between individuals, or as a result of a malicious attack, for instance a Denial of Service attack or the intentional infection of a computer with a virus. We are interested in the virus-like "spreading behaviour" shared by these failures, namely that neighbours of the initially affected vertex also cease to function.

Here, our measure of functionality will simply be connectivity, i.e. we will assume that a network is functioning or virus-resistant if it remains connected after a virus attack. This model of network failure is then a simple generalisation of that assumed in the ubiquitous vertex connectivity problem.

Formally, the type of network failure just described corresponds to a star in the graph. A set of vertices $S = \{v, x_1, x_2, \ldots, x_k\}$ is a *star* if (v, x_i) is an edge for each $1 \leq i \leq k$. We say that v is the *centre* of the star and that the edges (v, x_i) are the *rays* of the star. We call S a *star-cutset* if $G - S$ is disconnected (that is, if the graph induced by $V - S$ is disconnected). Our problem is then:

> `The Network Protection Problem.` Given a graph $G = (V, E)$ and a set of edges $F \subseteq E$, find the smallest set of edges $A \subseteq E$ such that $A \cup F$ intersects the set of rays of every star-cutset.

Clearly, the set $A \cup F$ can be thought of as a set of *protected* edges; if we protect $A \cup F$ then the graph is resistant to malicious attacks or network failures in the form of star-cutsets. The set F is a set of initially protected edges, which we can think of as already having been paid for, and A corresponds to edges we need to protect in addition to F to make the network secure. This formulation of the problem is useful given the recursive nature of the algorithm we propose for solving it. Typically, at the time of the first call we will have $F = \emptyset$, i.e. no initially protected edges, but later calls will pass on partial solutions and find extensions to them.

We call a star-cutset S *vulnerable* (with respect to F) if $R(S) \cap F = \emptyset$, where $R(S)$ is the set of rays of S. Consequently, we wish to solve the hitting set problem where the elements are edges and the sets we need to hit are vulnerable star-cutsets

1.1 Our Results

As we show in Section 4, the Network Protection Augmentation Problem is NP-hard for general graphs. We also exhibit an $O(\log n)$-approximation algorithm, where n is the number of vertices in the graph (Section

*Department of Mathematics and Statistics, McGill University. Email: `sonnerat@math.mcgill.ca`

†Department of Mathematics and Statistics, and School of Computer Science, McGill University. Supported in part by NSERC grant 28833. Email: `vetta@math.mcgill.ca`

4.2). Since the hardness reduction is from Set Cover and is such that approximation guarantees are preserved, this $O(\log n)$ factor is essentially best possible.

In planar graphs however, star-cutsets are structured in a very interesting way. We explore this structure in Section 2, and in Section 3 we exploit it to obtain a PTAS for the Network Protection Augmentation Problem.

We also show that even in planar graphs, the vertex version of the Network Protection problem is NP-hard.

1.2 Related Work

A natural generalisation of a star-cutset is a cutset which is spanned by a tree of depth r for some integer r (modelling an attack where a virus can spread further than just to the immediate neighbours of a vertex). In [10], the authors consider the problem of finding a small spanning subgraph without such cutsets in a given graph. A major issue is that the property of not containing such cutsets (i.e. of being virus-resistant) is non-monotonic, in that adding edges can create new cutsets. The simplest example of this phenomenon is a cycle to which a chord is added.

A second natural generalisation is to allow more than one simultaneous virus attack. This corresponds to asking whether a network has a cutset that can be written as the union of k stars. In [11], the authors show that it is NP-hard to answer this question for a given graph, if k is part of the input. However, there is an efficient algorithm for the case of a single attack, even if the radius r of the cutset is not restricted to equal 1 and is part of the input.

Star-cutsets are a special case of skew partitions, which have received a great deal of attention because of their relation to the structure of perfect graphs. For example, Chátal's Star-cutset Lemma asserts that no minimal imperfect graph contains a star-cutset ([3]). For a survey of results on skew partitions and perfect graphs, see [9].

The virus-like spreading behaviour we model using star-cutsets or trees of depth r also occurs in the Firefighter Problem, where the goal is to protect as many vertices as possible from a fire that propagates through the network (cf. Finbow et al. [5]). In other instances, however, it may be desirable to maximize the propagation, rather than protect the network against it. This is true for example in the case of viral marketing. Kempe et al. investigate how to maximize the spread of information of influence through a social network in [8].

2 On the Structure of Star-Cutsets in Planar Graphs

The goal of this section is a structural result on planar graphs that will be exploited by our polynomial time approximation scheme in Section 3. Before presenting this structural result, though, we examine how large *minimal* star-cutsets can be in a planar graph. We may restrict our attention to minimal star-cutsets for the simple reason that a set of edges contains a ray from every star-cutset if and only if it contains a ray from every minimal star-cutset. (A star-cutset $S = \{v, x_1, \ldots, x_k\}$ centred at v is *minimal* if $v \cup X$ is not a star-cutset for any proper subset $X \subset \{x_1, \ldots, x_k\}$.)

2.1 The Cardinality of Minimal Star-Cutsets

Here we will show that we may focus our attention on the case in which every minimal star-cutset contains exactly three vertices. We begin with some simple observations. Suppose that G contains a cut-vertex v. Thus, there is a star-cutset centered at v containing no rays; consequently, protecting edges cannot defend this star-cutset. In other words, a cut-vertex in the graph will always remain a cut-vertex, no matter how many edges we protect. Thus, we may make the following assumption.

Assumption 2.1. *G is 2-connected.*

Next, we call an edge (u, v) a *separating-edge* if the removal of its two end-vertices u and v disconnects the graph. So $\{u, v\}$ is a star-cutset with a single ray (centered at either u or v), and therefore (u, v) must be protected in any solution. Hence, we lose no generality in assuming that F already contains each separating-edge. The algorithm presented in Section 3 contains a preprocessing step that adds all these edges to F.

Assumption 2.2. *Every (vulnerable) star-cutset contains at least* 3 *vertices.*

In fact, we may focus solely on cutsets that contain exactly three vertices.

Lemma 2.3. *Every minimal star-cutset consists of exactly* 3 *vertices.*

Proof. Suppose $S = \{v, x_1, \ldots, x_k\}$ is a minimal star-cutset centred at v. So $G - S$ has at least two components, say C and D. Since S is minimal, there is an edge from C to each of $x_1, \ldots, x_k$, and an edge from D to each of $x_1, \ldots, x_k$. Since there is also an edge from v to each of $x_1, \ldots, x_k$, contracting C and D gives a $K_{3,k}$ minor in G. By planarity, this implies that $k \leq 2$. So by Assumption 2.2, we have $k = 2$. $\square$

We remark that Lemma 2.3 implies that there are only a polynomial number of minimal star-cutsets. Moreover, from an algorithmic perspective, we can find also find them all efficiently by exhaustive search[1].

2.2 The Structural Theorem

To state the structural theorem we need a couple of definitions. First, let $S = \{v, x, y\}$ be a minimal star-cutset centred at v. Then we say that a star-cutset S' *crosses* S if S' is centred at v, x or y and contains vertices from two distinct components of $G - S$. Second, the *wheel graph* W_n consists of a cycle on n vertices, together with an additional *hub vertex* adjacent to (some of) the vertices in the cycle. The edges from the hub vertex to vertices in the cycle are called *spokes*. Then we obtain:

Theorem 2.4. *Let $G = (V, E)$ be a* 2*-connected planar graph, and let $F \subseteq E$. If G has a vulnerable star-cutset, then one of the following must be true:*

(a) For every vulnerable star-cutset S, at most one of the components of $G - S$ is non-trivial [2].

(b) G contains an uncrossed vulnerable star-cutset S such that at least two components of $G - S$ are non-trivial.

(c) G is a wheel graph.

For ease of exposition, when we now talk about star-cutsets it shall be understood that they are vulnerable, unless specified otherwise. Now to prove Theorem 2.4 we need to delve deeper into how minimal star-cutsets can relate to each other. First we show that star-cutsets can cross only in restricted circumstances.

Theorem 2.5. *Let $S = \{v, x, y\}$ be a minimal star-cutset centred at v, and suppose S' is a star-cutset that crosses S. Then all of the following hold:*

(i) S' is centred at v.

(ii) S crosses S'.

(iii) $G - S$ has exactly 2 *components.*

[1] If desired, a faster algorithm to do this may be obtained by exploiting the decomposition theorem of this section.

[2] A *trivial* component consists of a single vertex.

Proof. There are two non-symmetric cases to deal with. Either S' is centred at v or it is not.

In the latter case we may assume that $S' = \{x, a, b\}$ where a and b lie in components C and D of $G - S$, respectively. Take $u \in G - S'$. We claim that there is a path from u to y in $G - S'$. If u is v or y, this is trivial. If u lies in some component of $G - S$ other than C or D, then clearly there is still a path from u to y in $G - S'$. So, without loss of generality, assume that u lies in C. If $S' = \{x, a, b\}$ hits every path from u to y, then the set $\{x, a\}$ must also hit every $u - y$ path, as there is no path from u to $\{v, y\}$ in $G - S'$. But there is an edge between x and a and so, since S' is a vulnerable star-cutset, (x, a) is a (vulnerable) separating-edge, contradicting Assumption 2.2. Hence, every vertex $u \in G - S'$ is connected to y. But this contradicts the assumption that S' is a cutset. Hence S' must also be centred at v, and (i) holds.

For the former case, let $S' = \{v, a, b\}$ with $a \in C, b \in D$, and again take $u \in G - S'$. We claim that there is either a path from u to x or a path from u to y in $G - S'$. The proof is similar to the previous argument. If u equals x or y, there is nothing to prove. If u lies in some component of $G - S$ other than C or D, then clearly there is still a path from u to x and from u to y. So assume, without loss of generality, that $u \in C$. If $S' = \{v, a, b\}$ hits every path from u to x and from u to y, then the set $\{v, a\}$ must also hit every $u - x$ path and every $u - y$ path, since S separates u from b. But then $\{v, a\}$ would be a vulnerable separating-edge. Hence, every vertex is connected to either x or y in $G - S'$. It follows that x and y must be in distinct components of $G - S'$, otherwise $G - S'$ would be connected. However, if x and y are in distinct components of $G - S'$, then S crosses S'. Thus, (ii) is true. Finally, (iii) follows from the fact that if $G - S$ had at least 3 components, then there would still be a path from x to y in $G - S'$, for any $S' = \{v, a, b\}$. □

Observe that (ii) implies that a star-cutset $S = \{v, x, y\}$ centred at v cannot be crossed if $xy \in E(G)$, since x and y would remain in the same component in $G - S'$ for any star-cutset S' crossing S.

It will be useful to partition the components of a minimal star-cutset $S = \{v, x, y\}$ into two types. We say that an component C of $G - S$ is *strongly attached* to S if each of v, x, y has an edge to some vertex of C. Otherwise we say that C is *weakly attached* to S.

Observation 2.6. *If C is weakly attached then there must be an edge from x into C, and an edge from y into C.*

Proof. If there were no edge from x into C, then vy would be a separating edge. □

A star-cutset cannot induce many strongly attached components:

Lemma 2.7. *Let $S = \{v, x, y\}$ be a minimal star-cutset in G. Then $G - S$ has at most 2 components that are strongly attached to S.*

Proof. Suppose there were three components C_1, C_2, C_3 of $G - S$ that are strongly attached to S. Then contracting these three components into single vertices would give a $K_{3,3}$-minor in G, contradicting planarity. □

Lemma 2.8. *If $S = \{v, x, y\}$ is a vulnerable star-cutset centred at v such that $e = xy \in E - F$, then $G - S$ has no components that are weakly attached to S.*

Proof. By definition, a weakly attached component W has edges only to x and y. But if $e = xy \in E - F$, then e is a vulnerable separating-edge, contradicting Assumption 2.2. □

Lemma 2.9. *If $S = \{v, x, y\}$ is a star-cutset centred at v and W is a component of $G - S$ that is weakly attached to S, then G has no star-cutset $S' = \{x, v, b\}$ or $S' = \{y, v, b\}$ with $b \in W$.*

Proof. Suppose, without loss of generality, that $S' = \{x, v, b\}$. Clearly every vertex not in W still has a path to y in $G - S'$. If there is a vertex $u \in W$ such that S' hits all the $u - y$ paths, then $\{x, b\}$ must also hit all the $u - y$ paths, since there is no edge from v into the weakly attached component W. But this contradicts Assumption 2.2. □

We remark that in Corollary 2.9, S' does not cross S, since it shares 2 vertices with S.

The next corollary will be applied repeatedly in the proof of our main structural result, Theorem 2.4.

Corollary 2.10. *Suppose $S = \{v, x, y\}$ is a star-cutset and $S' = \{v, a, b\}$ is a star-cutset crossing S, with a and b in distinct components, respectively C and D, of $G - S$. Then a must intersect every $x - y$ path whose internal vertices lie in C, and b must intersect every $x - y$ path whose internal vertices lie in D.*

Proof. If there were a path P from x to y all of whose internal vertices lie in C and not containing a, then x and y would be in the same component of $G - S'$, that is, S would not cross S'. □

Let's now return to Theorem 2.4. It must be the case that there exists a star-cutset S such that there are at least two non-trivial components in $G - S$, otherwise 2.4(a) holds. If S is not crossed by any other star-cutset, (b) holds. So suppose that every S with the property that $G - S$ has at least two non-trivial components is crossed by at least one other star-cutset. Our goal for the remainder of the section will be to show that in that case, G must by the wheel graph.

2.3 Discovering the Wheel

Take a crossed, non-trivial star-cutset $S = \{v, x, y\}$ with centre v, and consider the collection $\mathcal{S}$ of all the star-cutsets that cross S. Observe that if S is crossed, then Theorem 2.5 implies that $G - S$ has exactly two components. So let C and D be the components of $G - S$. In addition, let $a_1, \dots, a_r$ be the vertices in C that occur in some star-cutset belonging to $\mathcal{S}$, and let $b_1, \dots, b_t$ be the vertices in D that occur in some star-cutset belonging to $\mathcal{S}$.

Given an embedding of G in the plane, label the vertices $a_1, a_2, \dots, a_r$ and $b_1, b_2, \dots, b_t$ in clockwise order around v, as shown in Figure 1. For notational convenience, we also define $a_0 = b_{t+1} := x$ and $a_{r+1} = b_0 := y$.

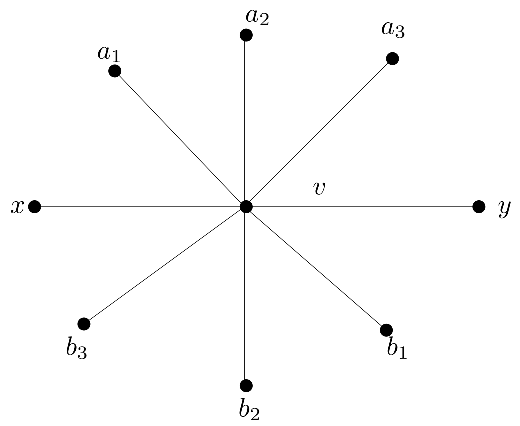


Figure 1: The wheel.

In the example of Figure 1 we have drawn all the a_i on one "side" of S, and all the b_j on the other "side". The following claim establishes formally that this is justified.

Claim 2.11. *There are no indices i, k, j with $i < k$ such that b_j lies between a_i and a_k in the cyclic ordering of the vertices $a_0, \dots, a_{r+1}, b_0, \dots, b_{t+1}$ around v.*

Proof. Suppose there are indices i, k and j as in the statement of the claim. Let P, Q be paths from x to y whose internal vertices lie in C respectively D. ¿From Corollary 2.10, we know that P must hit both a_i and a_k, so let P' be the subpath of P between a_i and a_k. But now the cycle T consisting of P' together with the edges va_i and va_k separates b_j from x and y (see Figure 2 (a)). Since b_j must be contained in Q, it follows that Q must intersect T. G is planar, so Q cannot cross any edges of T. If Q crosses T at a vertex, this contradicts the assumption that the internal vertices of P and Q lie in different components of $G - S$.

[Note that for this proof it doesn't matter whether P visits a_i or a_k first, although we will show in a moment that it must actually visit a_i first.] □

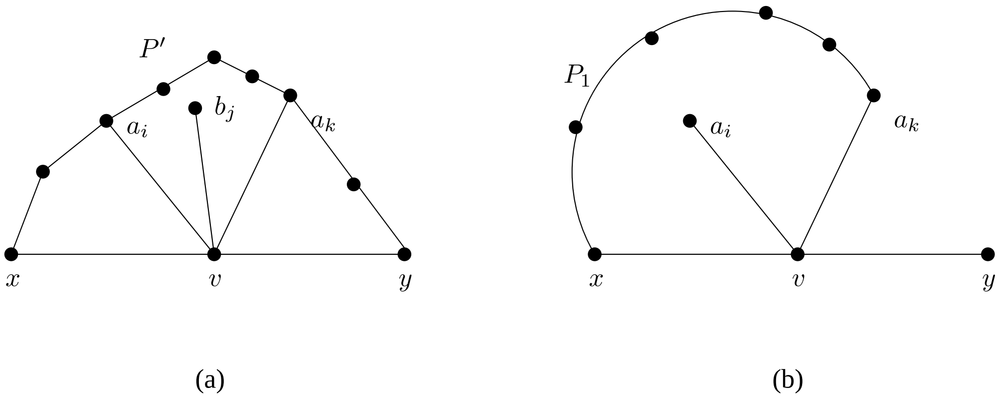


Figure 2: The cyclic ordering.

Lemma 2.12. *For each pair i, j with $1 \le i \le r$ and $1 \le j \le t$, the set $S' = \{v, a_i, b_j\}$ is a star-cutset.*

Proof. This follows immediately from Corollary 2.10. Every a_i hits all the $x - y$ paths through C, and every b_j hits all the $x - y$ paths through D, so there cannot be an $x - y$ path in $G - \{v, a_i, b_j\}$. □

Claim 2.13. *Any $x - y$ path P all of whose internal vertices lie in C visits the a_i in the order induced by the embedding of G in the plane, i.e. if $i < j$ then P visits a_i before a_j. Similarly, any $y - x$ path Q all of whose internal vertices lie in D visits the b_i in the order induced by the embedding of G in the plane.*

Proof. Suppose that P is a path from x to y with internal vertices in C, and suppose that visits a_j before a_i for some pair $i < j$. Note that we assume that P is a path, not a walk, so any vertex on P is only visited once. Let P_1 be the subpath of P from x to a_j, and let P_2 be the subpath from a_i to y. Consider the cycle T consisting of P_1 together with the edges $a_j v$ and vx. This cycle separates a_i and y, that is, if we remove T from the plane, a_i and y end up in different components. So P_2 must cross T somewhere. It cannot cross any of the edges of T, since G is planar, and it cannot cross T at x or v, since all of P's interior vertices lie in C. Finally, P_2 cannot cross T at a vertex belonging to P_1, since then P would visit a vertex twice. This gives us the desired contradiction. Figure 2 (b) illustrates the proof. □

Lemma 2.14. *For every i and k with $i < k$, $\{v, a_i, a_k\}$ is a star-cutset unless $a_i a_k$ is an edge which is used in every path from x to y through C. Similarly, $\{v, b_i, b_k\}$ is a star-cutset unless $b_i b_k$ is an edge which is used in every path from y to x through D.*

Proof. If $i = 0$ and $k = r + 1$, there is nothing to prove. Consider a path P from x to y all of whose internal vertices are in C. By Claim 2.13, P hits a_i before it hits a_k. Let A_P be the set of vertices of P between a_i to a_k. If A_P is non-empty, then $\{v, a_i, a_k\}$ is a star-cutset separating A_P from x and from y, since otherwise either a_i or a_k would miss an $x - y$ path through C. If A_P is empty, then $e = a_i a_k$ is an edge. It follows that $\{v, a_i, a_k\}$ is a star-cutset unless A_P is empty, in which case $a_i a_k$ is an edge on every $x - y$ path P through C. An analogous proof works for the b_j. □

With these results in hand, we can now prove Theorem 2.4.

Proof of Theorem 2.4. Recall that we may assume: G contains at least one non-trivial star-cutset; every non-trivial star-cutset is crossed. Again we take such a star-cutset S, the set $\mathcal{S}$ of star-cutsets crossing S, and define the a_i and b_j vertices as before. We will now show that G must be a wheel.

For each i from 0 to $r - 1$, consider the star $S_i = \{v, a_i, a_{i+1}\}$ centred at v. As in the proof of Lemma 2.14, let A_i consist of the vertices of $x - y$ paths lying between a_i and a_{i+1}. If $A_i = \emptyset$, then $a_i a_{i+1}$ must be an edge of G. If $A_i = \{a\}$ for some vertex a, then a must also hit all the $x - y$ paths through C and thus

$va \notin E(G)$, for otherwise a would be missing in the list $a_1, \dots, a_r$. A similar consideration applies for the stars $\{v, b_j, b_{j+1}\}$. Now, if for all i between 0 and $r-1$, and all j between 0 and $t-1$, the set of vertices between a_i and a_{i+1} (respectively, b_j and b_{j+1}) is either empty or a singleton vertex, then G must be a wheel graph, possibly with some of the spokes missing. In other words, (c) must hold.

Finally, if for some i, the set A_i between a_i and a_{i+1} (or the set B between b_j and b_{j+1} for some j) contains at least 2 vertices, then we claim that (b) holds for $S' = \{v, a_i, a_{i+1}\}$, a contradiction. This follows from the fact that a_i and a_{i+1} hit all the $x-y$ paths through C, so A_i is disconnected from x and y in $G - S'$. Moreover, there can be no star-cutset S^* crossing S', since the vertex a^* from S^* contained in A_i would have to hit all the $a_i - a_{i+1}$ paths through A_i, so it would also hit all the $x-y$ paths through C. Consequently, for any j, we have that $\{v, a^*, b_j\}$ is a star-cutset that crosses S, implying that a^* must be in the list of the a_is. $\square$

This concludes the proof of Theorem 2.4, which is at the core of the recursive algorithm presented in Section 3

3 A Polynomial Time Approximation Scheme

In this section we present our polynomial time approximation scheme for the network protection augmentation problem in planar graphs. The algorithm is an application of our structural result, Theorem 2.4, and details and quantitative performance bounds are provided in Sections 3.2 to 3.3. The algorithm is recursive and so, to complete the result, we show in Section 3.4 that the recursion can be applied in polynomial time.

To begin, recall that we are given a planar graph $G = (V, E)$ and a set of edges $F \subseteq E$, and we wish to obtain a minimum cardinality set $A \subseteq E$ such that $A \cup F$ is a solution to the network protection problem. We call any feasible solution to this problem an *extension* of F. So on input (G, F), our goal is to output an extension A of F (if one exists) such that $|A| \leq \alpha |J|$, where J is an optimal extension of F, and α is a constant tending to 1 as $n = |V(G)|$ goes to ∞.

Again, unless otherwise specified a star-cutset will always mean a star-cutset that is vulnerable with respect to some set of edges $F \subseteq E$.

The first step of the algorithm is to test whether G is 2-vertex connected. If it isn't, then it returns that there is no extension of F and stops. So we may assume that G is 2-connected. If G has no star-cutsets that are vulnerable with respect to F, then obviously $A = \emptyset$ is an optimal extension of F. So we may assume that there exists at least one vulnerable star-cutset. Then Theorem 2.4 applies, and gives us three cases to deal with. If 2.4(b) holds, the algorithm will find an extension by making recursive calls on certain subgraphs of G. If either 2.4(a) or 2.4(c) holds, the algorithm finds an extension directly, without recursing. We begin by discussing the two cases where no recursion is required.

3.1 Exploiting the wheel

First, let's consider the simplest case in the structural theorem; namely, Case 2.4(c) holds and so G is a wheel graph. Recall that a wheel consists of a cycle together with a hub vertex adjacent to some of the vertices in the cycle. We call the edges from the hub vertex to the cycle vertices *spokes*, and two spokes are said to be *adjacent* if the cycle vertices they contain are adjacent on the cycle.

Theorem 3.1. *Let G be a wheel graph with hub vertex v. Then a set of edges A is an optimal solution to the network protection problem if and only if it contains all but two adjacent spokes.*

Proof. First observe that a wheel graph never has any separating edges, so all minimal star-cutsets consist of 3 vertices.

The necessity in the statement of the theorem is obvious, since if $e = vx$ and $f = vy$ are unprotected spokes that are not adjacent, then $S = \{v, x, y\}$ is a star-cutset. For the sufficiency, observe that removing a star $\{v, x, y\}$ where vx and vy are adjacent spokes does not disconnect the graph. $\square$

It is easy to adapt the above argument to the case where we have a wheel graph G together with a set of already protected edges F. Simply select two adjacent unprotected spokes and let A be the set of remaining spokes. Then clearly $A - F$ is an optimal extension of F.

Finally, we remark that it is easy to implement the above to run in polynomial time.

3.2 Dealing with the trivial components

Next, suppose that Case 2.4(a) holds, i.e. that for every vulnerable star-cutset S, at most one of the components of $G - S$ is non-trivial. We will now show how the network protection augmentation problem can be approximated directly via a reduction to the vertex cover problem in a planar graph. So take our planar graph G and consider the following graph L^*. The vertices of L^* correspond to edges of G that are rays of minimal star-cutsets. We join $e = vx, f = vy \in V(L^*)$ by an edge if and only if $S = \{v, x, y\}$ is a minimal star-cutset in G. Observe that L^* is simply a subgraph of the line graph, $L(G)$, of G. To avoid confusion, we refer to the elements of $V(L^*)$ as nodes and to the elements of $E(L^*)$ as arcs.

Clearly this gives a one-to-one correspondence between solutions to the network protection problem in G and node covers in L^*. At first glance, this does not seem too helpful as the best known approximation guarantee for the node cover problem is a factor 2, and the line graph of a planar graph can be highly non-planar. However, if 2.4(a) holds, then the graph L^* turns out to be planar. This will be useful as there is a PTAS for node cover in planar graphs ([1]).

So let's see that L^* is indeed planar.

Theorem 3.2. *If every star-cutset S produces at most one non-trivial component in $G - S$ then L^* is a planar graph.*

Proof. Take a planar drawing of G. The idea of the proof is to show that we can associate a certain region of the plane with every star-cutset in such a way that for two distinct star-cutsets, their associated regions are disjoint. We can then use this to obtain a planar drawing of L^* from the drawing of G.

So let $S = \{v, x, y\}$ be a star-cutset centred at v. The following lemma takes care of a special case that complicates the proof of the theorem.

Lemma 3.3. *Suppose that $S' = \{w, v, b\}$ is a star-cutset crossing S. Then G is either a W_4 or a W_5.*

Proof. By assumption, $G - S$ has at most one non-trivial component. So, without loss of generality, we may assume that $\{w\}$ is a trivial component of $G - S$. Now we also know that $G - S'$ has at most one non-trivial component.

Since S' crosses S, Theorem 2.5 implies that S crosses S', and that both $G - S$ and $G - S'$ have exactly two components. But if S crosses S', then x and y must be in different components of $G - S'$. So $G - S'$ has two components, one of which contains x and the other containing y. By assumption, at least one of x and y must be a singleton component.

If both x and y are singleton components of $G - S'$ then G is a W_4, because $G - S'$ has exactly two components, and thus $V(G) = \{v, x, y, w, b\}$.

So suppose, wlog, that x is a singleton component, and that the component C'_y of $G - S'$ containing y contains at least one other vertex. By Corollary 2.10, b intersects every $x - y$ path going through the component C_b of $G - S$ containing b. So if C_y contained at least 2 vertices other than y, the set $\{v, y, b\}$ would be a star-cutset yielding *two* non-trivial components, contradicting our assumption on G. Hence, C'_y contains at most two vertices (including y), and the component C'_x of $G - S'$ containing x equals $\{x\}$, hence $V(G) = \{v, x, y, w, b, c\}$ where c is the unique other vertex in C'_y. Finally, this vertex c must be adjacent to y, because it is in C'_y, and it must also be adjacent to b, for otherwise (v, y) would be a separating-edge. Hence G must be a W_5 (the spoke (c, v) might be missing). □

Hence from now on we will assume that there is no star-cutset crossing S. We split our analysis into three cases.

`Case 1:` $xy \in E(G) - F$. Then by Lemma 2.8, there can be no weakly attached components. Consequently, by Lemma 2.7, $G - S$ has exactly two strongly attached components. By our assumption on G, one of the components is a single vertex, say $\{w\}$. Consider the three triangles $\Delta_1 = wxv, \Delta_2 = wyv$ and $\Delta_3 = wxy$ (as shown in Figure 3(b)). We claim that no vertices of G can lie in the interior of any of these triangles. To see this, observe that w has no neighbours other than v, x and y. So a vertex z in the interior of $\Delta_1 = wxv$ would be separated from y by the edge xv, contradicting Assumption 2.1. For analogous reasons, there can be no vertices of G in the interior of Δ_2 or Δ_3.

Now note that none of the edges wx, wy and wv can occur in star-cutsets: The star-cutset $\{v, x, y\}$ cannot be crossed (regardless on which vertex it is centred), and neither $\{w, x, y\}$ nor $\{w, x, v\}$ nor $\{w, v, y\}$ is a star-cutset, because both components of $G - S$ are strongly attached to S.

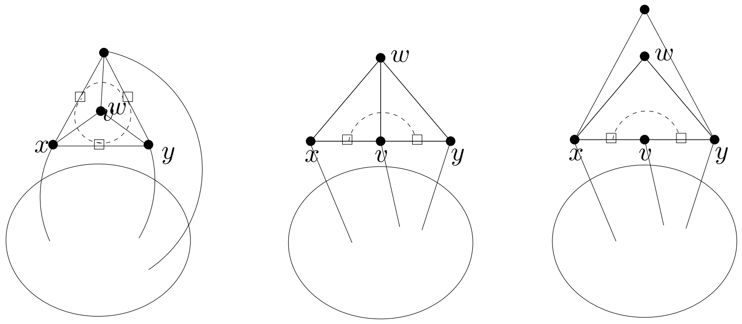
This implies that the triangles Δ_i are exclusive to S in the sense that they do not arise when we consider a different star-cutset $S' \neq S$.

`Case 2:` $xy \notin E(G) - F$ and there exists a trivial strongly attached component $\{w\}$, as shown in Figure 3(a). Consider the two triangles $\Delta_1 = wxv$ and $\Delta_2 = wyv$. By the argument given in Case 1, there can be no vertices of G in the interior of Δ_1 or Δ_2. Also, S is not crossed, and the sets $\{v, w, x\}$ and $\{v, w, y\}$ are clearly not star-cutsets, so none of the edges wv, wx and wy can appear in any star-cutsets.

As in Case 1, this implies that the triangles Δ_i are exclusive to S.

`Case 3:` $xy \notin E(G) - F$ and there does not exist a trivial strongly attached component. In this case, let $\{w\}$ be a trivial weakly attached component that lies on a face with the vertices x, y, v, as shown in Figure 3(c).

In this case, the interior of that face (which is the 4-cycle $w - x - v - y$) cannot contain any vertices of G, and again neither the edge wx nor the edge wy can belong a star-cutset. So the interior of the 4-cycle is exclusive to S.



(a) (b) (c)

Figure 3: Obtaining a planar drawing of L^*.

In all three cases discussed above, we can now obtain a planar drawing of L^* from the planar drawing of G: Place the nodes corresponding to $e = vx$ and $f = vy$ (and $g = xy$) at the midpoint of e and f (and g) in the drawing of G. The arc joining e and f can be drawn through the interior of the triangles wxv and wyv, crossing the edge wv once, as shown in Figure 3(b). For the case $g = xy \in E(G) - F$, we obtain a triangle efg crossing each of the edges wx, wy and wv once as in Figure 3(a). In the case shown in Figure 3(c), we draw the arc of L^* joining e and f through the interior of the 4-cycle $w - x - v - y$.

We showed above that in all three possible cases, the regions through which we draw the arcs of L^* are exclusive to S, i.e. that they do not arise from any other star-cutset $S' \neq S$. Hence none of the arcs of L^* can cross, and thus we have obtained a planar drawing of L^*. □

Given a planar graph and an integer k, Baker [1] showed that there is an algorithm that finds a vertex cover $\mathcal{C}$ of size $|\mathcal{C}| \leq \frac{k+1}{k}|\mathcal{C}^*|$, where $\mathcal{C}^*$ is an minimum vertex cover. The algorithm has runtime $O(8^k n)$, where n is the number of vertices in the graph. Taking k to be $\log n$, we get a polynomial time algorithm with an approximation factor converging to 1.

We close this subsection by remarking that the procedure described above adapts easily to the case where a set of protected edges F is given: First, construct L^* from G. Second, add the nodes of L^* corresponding to protected edges to the cover, and remove the arcs incident to them. Finally, search for a minimum vertex cover in the remaining subgraph of L^*.

3.3 The Recursion

Finally, suppose that Case 2.4(b) holds; that is, we have an uncrossed star-cutset S such that $G - S$ has at least two non-trivial components. This is the most complex case, and in this situation the algorithm will call itself recursively on specific subgraphs of G. We already know how to solve the subgraphs occurring at the leaves of the recursion; they are the cases dealt with in Section 3.1 and Section 3.2.

The main technial issue with our recursion is that, computationally, its natural implementation takes exponential time. Consequently, we cannot try all combinations when we attempt to piece together solutions to subproblems. Rather, to obtain a polynomial time algorithm we may only consider a selection of the possible combinations. The key will be in making the choice carefully to ensure that we do consider at least one combination that leads to a provably good solution.

Let's now describe the recursion. Number the components of $G-S$ as $C_1, C_2, \ldots, C_r$, where the labelling is such that the strongly attached components appear before all the weakly attached components. Recall that there are either zero, one or two strongly attached components. This gives three cases to deal with in the recursion. Recall, though, that Lemma 2.8 implies that if there exists a weakly attached component, then $xy \notin E(G) - F$. This will actually lead us to consider four distinct cases. For the case of two strongly attached components, we will treat separately the case in which there is a weakly attached component and the case in which there are no weakly attached components.

3.3.1 No strongly attached components

`Case I.` So in our first case $G - S$ has no strongly attached components. For each i let $G_i = G[S \cup C_i]$, the subgraph of G induced by $S \cup C_i$, and let $F_i = F \cap E(G_i)$. For each i such that $|C_i| > 1$, call the algorithm recursively on (G_i, F_i) to obtain an extension A_i of F_i. For each i such that $C_i = \{c_i\}$, let $A_i = \{c_i x\}$. Return $A = \cup_i A_i \cup \{xv\}$.

We remark that $E(G_i) \cap E(G_j) = \{e, f\}$ whenever $i \neq j$; again, here e and f refer to the edges xv and vy, respectively. This fact will be used repeatedly in the analysis.

Theorem 3.4. *Assuming that each recursive call returns an extension A_i of F_i such that $|A_i| \leq \alpha |J_i|$, where J_i is an optimal extension of F_i in G_i, we have that A is an extension of F in G such that $|A| \leq \alpha |Z|$, where Z is an optimal extension of F in G.*

Proof. We begin by proving that the algorithm is well defined. To this end, we must first show that each G_i is a 2-connected planar graph. The planarity is obvious, so it remains to show the 2-connectivity.

Claim 3.5. *For all i, the graph G_i is 2-connected.*

Proof. Let $u \in V(G_i)$. If u is x, y or v, then clearly u is not a cut-vertex in G_i. If $u \notin S$ is a cut-vertex in G_i, then u separates some $w \in C_i$ from S and is, therefore, a cut-vertex in G, contradiction. □

Next, we must show that the set of edges A returned by the algorithm is a feasible extension of F.

Claim 3.6. *Every star-cutset S' of G is a star-cutset in G_i for some i, unless $S' = S$ or $S' = \{w, x, y\}$ where $\{w\}$ is a trivial component of $G - S$.*

Proof. Let $S' \neq S$. First observe that since S is uncrossed, S' must be contained in $V(G_i)$ for some i.

By Lemma 2.8, $xy \notin E(G) - F$. Thus, if S' contains x and y then it must be the case that $S' = \{w, x, y\}$ for some $w \in C_i$, since $S' \neq S$. So unless C_i is trivial (and equals $\{w\}$), S' separates v from the vertices in $C_i - w$.

So now suppose that S' contains at most one of x and y. Without loss of generality, assume that S' doesn't contain x. Then clearly all vertices in C_j for all $j \neq i$ are in the same component as x in $G - S'$. But since S' is assumed to be a cutset, there exists u which is not in the same component as x, and this u must be in $V(G_i)$. But if u is separated from x in $G - S'$, then clearly u is also separated from x in $G_i - S'$, since G_i is a subgraph of G. □

Claim 3.6 implies that the set A returned by the algorithm is a feasible solution. The star-cutset S and any star-cutsets S' of the form $S' = \{w, x, y\}$ where $\{w\}$ is a trivial component of $G - S$ are taken care of by adding the edges vx and wx to A. All other star-cutsets are star-cutsets in one of the graphs G_i we recurse on; consequently, they will be intersected by the respective partial extension A_i of F_i.

To analyse the approximation guarantee of the algorithm, we need the following converse to Claim 3.6.

Claim 3.7. *For any i, a star-cutset S' in G_i is also a star-cutset in G.*

Proof. Let S' be a star-cutset in G_i. We have two possibilities.
(i) Assume S' contains both x and y. Then S' cannot be centered at v, otherwise $S' = S$ and S is not a star-cutset in G_i. By Lemma 2.8, $xy \notin E(G) - F$ and so S' cannot be centered at x or y either. Thus, S' must be centered at some vertex $w \in C_i$, and so $S' = \{w, x, y\}$. Then S' is still a star-cutset in G, as it separates v from the vertices in C_j, where $j \neq i$.
(ii) Assume, wlog, that S' does not contain x. Then some vertex u is separated from x in $G_i - S'$. Our goal is to show that u is also separated from x in $G - S'$. For a contradiction, suppose not. So there is a path P from u to x in $G - S$. Since P is not in G_i, it must use vertices from some component C_j with $j \neq i$, and must therefore contain y.

Since P is a path in $G - S'$ containing y, it follows that S' contains neither x nor y. Consequently, it cannot contain v either because v has no edges into C_i. Therefore, we can shortcut P to a path in $G_i - S'$ by using the edges yv and vx. This contradicts the assumption that S' separates x and u. □

Claim 3.8. *If $C_i = \{c_i\}$ is a trivial component, then $S' = \{c_i, x, y\}$ is a star-cutset in G.*

Proof. We're assuming that there exist at least 2 non-trivial components, but no strongly attached components of $G - S$. So, let C_j and C_k be two non-trivial weakly attached components, and let $C_i = \{c_i\}$ be trivial. Then clearly $S' = \{c_i, x, y\}$ separates C_j from C_k in G. □

Now, recall that we assume that a recursive call on (G_i, F_i) returns an extension A_i such that $|A_i| \leq \alpha |J_i|$ where J_i is an optimal extension of F_i. Suppose Z is an optimal extension of F, and define $Z_i = Z \cap E(G_i)$. Note that $Z_i \cap Z_j \subseteq \{e, f\}$ for $i \neq j$, since $E(G_i) \cap E(G_j) = \{e, f\}$.

Our goal is to show that the extension A returned by the algorithm in Case I satisfies $|A| \leq \alpha |Z|$.

We begin with a lemma which will also be used in Cases II and III.

Lemma 3.9. *If $G_i = G[S \cup C_i]$, where C_i is a weakly attached component, then neither e nor f is a ray in a vulnerable star-cutset in G_i.*

Proof. Suppose $e = xv$ is a ray of a star-cutset S'. Again, by Lemma 2.8, $xy \notin E(G) - F$. This implies that $S' = \{x, v, w\}$ for some $w \in C_i$. As S' is assumed to be a cutset, there is a vertex u separated from y in $G_i - S'$. Because v has no edges into the weakly attached component C_i, it follows that $\{x, w\}$ separates y and u in G. This contradicts Assumption 2.2. □

Claim 3.10. *Z contains exactly one of e and f, and*

$$|Z| = 1 + \sum_i (|Z_i| - 1).$$

Proof. Observe that Z must contain at least one of e and f, since S is vulnerable. In addition, S is not a star-cutset in G_i, for any i. Moreover, by Lemma 3.9, neither e nor f can appear in a star-cutset in G_i, for any i. It follows that Z must contain exactly one of e and f: if it contained both then we would get a strictly smaller extension by discarding one of them.

Without loss of generality, assume that Z contains e but not f. Then $Z_i \cap Z_j = \{e\}$ for $i \neq j$. Thus, Z is the disjoint union of the sets $Z_i - \{e\}$ together with $\{e\}$. So we obtain that $|Z| = 1 + \sum_i(|Z_i| - 1)$, as claimed. □

We may now complete the proof of Theorem 3.4. Again, wlog, assume that Z contains e but not f. Claim 3.7 implies that Z_i is also an extension of F_i. Now e does not occur in any star-cutset in G_i, by Lemma 3.9, so $Z_i - e$ is an extension of F_i. Thus, $|J_i| \leq |Z_i| - 1$. So if we recurse on G_i then, by induction $|A_i| \leq (|Z_i| - 1)$. If we didn't recurse on G_i because C_i was trivial, then $|A_i| = 1$ and so $|A_i| \leq |Z_i| - 1 \leq \alpha(|Z_i| - 1)$, since by Claim 3.8, Z has to contain one of the edges from the trivial component to the set $\{x, y\}$ in addition to e. Hence, the size of the extension returned by the algorithm is

$$1 + \sum_i |A_i| \leq 1 + \alpha \sum_i (|Z_i| - 1) \leq \alpha(1 + \sum_i (|Z_i| - 1)) = \alpha|Z|.$$

This concludes the proof of Theorem 3.4 □

3.3.2 One strongly attached component

`Case II.` Now consider the case where $G - S$ has exactly one strongly attached component, C_1. As $G - S$ has at least two non-trivial components, we may assume that the (weakly attached) component C_2 is non-trivial. We then define our recursive problems as follows. Set G_1 to be the graph obtained from $G[S \cup C_1 \cup C_2]$ by contracting the component C_2 into a single vertex v_1^*; set $G_i = G[S \cup C_i]$ for each $i \geq 2$. Then set $F_1 = F \cap E(G_1) \cup \{v_1^* x, v_1^* y\}$ and set $F_i = F \cap E(G_i)$, for each $i \geq 2$.

Apply the algorithm recursively on (G_1, F_1) to obtain an extension A_1; call the algorithm recursively on (G_i, F_i) to obtain an extension A_i of F_i, for each $i \geq 2$ with $|C_i| > 1$; finally, for each $i \geq 2$ where C_i is a singleton, set $A_i = \{c_i x\}$. Return the output $A = \cup_i A_i$.

Again, by Lemma 2.8, the existence of a weakly attached component implies that $xy \notin E(G) - F$.

Lemma 3.11. *Assuming that each recursive call returns an extension A_i of F_i such that $|A_i| \leq \alpha|J_i|$, where J_i is an optimal extension of F_i in G_i, we have that A is an extension of F in G such that $|A| \leq \alpha|Z|$, where Z is an optimal extension of F in G.*

Proof. Again, we begin by showing that the recursion is well defined by proving each G_i to be a 2-connected planar graph. The planarity is clear, so it remains to show the 2-connectivity.

Claim 3.12. *For all i, the graph G_i is 2-connected.*

Proof. The proof is virtually the same as in Claim 3.5. We make the additional remark that the vertex v_1^* in G_1 obtained from contracting the weakly attached component C_2 cannot be a cut-vertex in G_1. □

Next, we must show that the set of edges A returned by the algorithm is a feasible extension of F.

Claim 3.13. *Every star-cutset S' of G is a star-cutset in G_i for some i, unless $S' = \{w, x, y\}$ where $\{w\}$ is a weakly attached trivial component of $G - S$.*

Proof. If $S' = S$, then clearly S' is a cutset in G_1, since it separates v_1^* from the vertices of C_1.

So, take $S' \neq S$. Since S is uncrossed, S' is contained in $V(G_i)$ for some i. We then have two cases:
(i) S' contains both x and y. Hence, $S' = \{w, x, y\}$ for some $w \in C_i$ because $S' \neq S$. If C_i is weakly attached to S then S' separates v from the vertices in $C_i - w$, unless C_i is trivial and equals $\{w\}$. If C_i is strongly attached to S, then S' separates v^* from v in G_i.

(ii) S' does not contain x. All the vertices in C_j, for each $j \neq i$, are in the same component as x in $G-S$. But S' separates some vertex w in C_i from x. However, if w is separated from x in $G-S'$ then clearly they are also separated in $G_i - S'$. □

Claim 3.13 implies that the set A returned by the algorithm is a feasible solution. Any star-cutset of the form $S' = \{w, x, y\}$ where $\{w\}$ is a trivial component of $G-S$ is taken care of by adding the edge wx to A, and all other star-cutsets are star-cutsets in one of the graphs G_i we recurse on, so they will be intersected by the respective partial extension A_i of F_i.

Again, we need the following converse to Claim 3.13.

Claim 3.14. *A star-cutset S' in G_i is also a star-cutset in G.*

Proof. Let S' be a star-cutset in G_i. If $S' = S$, then there is nothing to show. Take $S' \neq S$. If S' contains both x and y, then S' cannot be centered at v. In addition, S' cannot be centered at x or y as $xy \notin E(G) - F$, by Lemma 2.8. Thus, S' is centered at some vertex $w \in C_i$. Because there is at least one non-trivial weakly attached component, S' is a star-cutset in G.

Next, suppose that S' does not contain, without loss of generality, x. Let u be a vertex separated from x in $G_i - S'$. We treat the cases $i = 1$ and $i \geq 2$ separately.

Case (a): $i = 1$. We may assume that $u \neq v_1^*$ because $v_1^* x$ is an edge, so S' cannot separate x from v_1^*. Note also that each edge incident to v_1^* is protected, so S' cannot contain v_1^*. Suppose, for a contradiction, that there is a path P from u to x in $G - S'$. Then P would have to contain vertices from a component C_j with $j \neq 1$. This implies that P contains y, since v has no edges into any component C_j with $j \neq 1$. But then we obtain a path P_1 from u to x in G_1 by shortcutting P using the protected edges yv_1^* and $v_1^* x$. This gives the desired contradiction.

Case (b): $i \geq 2$. Here, S' cannot contain v without containing y as well, since v has no edges into C_i. Any path P from u to x in $G - S'$ must go through y. Consequently, we can shortcut P using the edges yv and vx, contradicting the fact that u and x are separated in $G_i - S'$. □

Claim 3.15. *If $C_i = \{c_i\}$ is a trivial weakly attached component then $S' = \{c_i, x, y\}$ is a star-cutset in G.*

Proof. We have one strongly attached component, but at least two non-trivial components. Since C_2 is non-trivial, $S' = \{c_i, x, y\}$ separates v from C_2 in G. □

To finish the proof of Lemma 3.11, suppose Z is an optimal extension of F, and define $Z_i = Z \cap E(G_i)$. Our goal is to show that the extension A returned by the algorithm in this Case II satisfies $|A| \leq \alpha|Z|$.

Claim 3.16. *If Z contains exactly one of e and f, then*

$$|Z| = 1 + \sum_i (|Z_i| - 1).$$

If Z contains both e and f, then

$$|Z| = 2 + \sum_i (|Z_i| - 2).$$

Proof. Observe that Z must contain at least one of e and f as S is vulnerable. Also, note that $E(G_i) \cap E(G_j) = \{e, f\}$ for any i, j with $i \neq j$; thus, $Z_i \cap Z_j \subseteq \{e, f\}$.

Now suppose that Z contains exactly one of e and f, wlog e. Then Z is the disjoint union of the sets $Z_i - \{e\}$ together with $\{e\}$, and $|Z| = 1 + \sum_i (|Z_i| - 1)$, as claimed.

If Z contains both e and f, then Z is the disjoint union of the sets $Z_i - \{e, f\}$ together with $\{e, f\}$, and $|Z| = 2 + \sum_i (|Z_i| - 2)$, as claimed. □

So now assume that a recursive call on a pair (G_i, F_i) returns an extension A_i of F_i such that $|A_i| \le \alpha |J_i|$, where J_i is an optimal extension of F_i in G_i. Suppose first that Z contains e but not f. Claim 3.14 implies that for all $i \ge 1$, Z_i is also an extension of F_i. In particular, this gives $|J_1| \le |Z_1|$.

Using arguments similar to those in Case I, the edge e cannot appear in any star-cutsets of G_i for $i \ge 2$, so $Z_i - e$ is still a feasible extension. Thus, $|J_i| \le |Z_i| - 1$ for $i \ge 2$.

Finally, if $C_i = \{c_i\}$ is a trivial weakly attached component then, by Claim 3.15, $\{c_i, x, y\}$ is a star-cutset in G. So, when $A_i = \{c_i x\}$, we also have $|A_i| \le |Z_i| - 1$.

Hence the solution returned by the algorithm has size

$$\sum_{i\ge1} |A_i| = |A_1| + \sum_{i\ge2} |A_i| \le \alpha|Z_1| + \alpha \sum_{i\ge2}(|Z_i| - 1) \le \alpha(1 + \sum_{i\ge1}(|Z_i| - 1)) \le \alpha|Z|.$$

If Z contains e and f, similar arguments imply that $|J_i| \le |Z_i| - 2$, for all $i \ge 2$, and that $|J_1| \le |Z_1|$. Hence, the size of the solution A returned by the algorithm satisfies

$$|A| = |A_1| + \sum_{i\ge2} |A_i| \le \alpha|Z_1| + \alpha \sum_{i\ge2}(|Z_i| - 2) \le \alpha(2 + \sum_{i\ge1}(|Z_i| - 2)) = \alpha|Z|.$$

This concludes the proof of Lemma 3.11 □

3.3.3 Two strongly attached components but no weakly attached components

`Case III.` For our third case, assume that $G - S$ contains exactly two strongly attached components, namely C_1 and C_2, but contains no weakly attached components. That is, $V(G) = S \cup C_1 \cup C_2$.

Recall that we're in Case 2.4(b), i.e. $G - S$ has at least 2 non-trivial components, and thus C_1 and C_2 are both non-trivial. To define our subproblems, set G_1 to be the graph obtained from G by contracting C_2 into a vertex v_1^*, and set G_2 to be the graph obtained from G by contracting C_1 into a vertex v_2^*. Set $F_i = F \cap E(G_i) \cup \{v_i^* x, v_i^* y, v_i^* v\}$.

Algorithmically, dealing with two strongly attached components is more complex that the previous two cases. Essentially, this is because both components may prefer different solutions on their intersection S. We could deal with this by trying all possibilities on $E(S)$ in each subproblem, but this would lead to an exponential time recursion. Hence we will need to be careful in what subproblems we choose to solve. Towards this goal, let $m_i = |E(G_i)|$. Without loss of generality, we may assume that $m_1 \le m_2$. Our basic idea will be to find solutions to a variety of problems on G_1, and use this information to decide on a single problem to solve on G_2. This will produce a polynomial time algorithm, but we need to ensure that such an approach does not lead to a poor approximation guarantee.

A second issue that we need to deal with here is that there may be an edge $g = xy \in E(G) - F$. (Recall that in the previous cases, this could not occur as there were weakly attached components.) Such an edge implies that vxy is a triangle that gives rise to three vulnerable star-cutsets. We will deal with the existence/non-existence of edge xy separately.

`Condition 1:` First suppose that $xy \notin E(G) - F$. We recursively call the algorithm on $(G_1, e \cup F_1), (G_1, f \cup F_1)$, and $(G_1, \{e, f\} \cup F_1)$ to obtain extensions A_e, A_f, and A_{ef}, respectively.

(a) If $|A_{ef}| = |A_e| - 1 = |A_f| - 1$, call the algorithm on $(G_2, \{e, f\} \cup F_2)$ to obtain an extension B_{ef}, and return $A_{ef} \cup B_{ef} \cup \{e, f\}$.

(b) Exactly one of A_e and A_f is equal to A_{ef} in cardinality. Without loss of generality, suppose that $|A_e| = |A_{ef}| = |A_f| - 1$. Call the algorithm on $(G_2, e \cup F_2)$ to obtain an extension B_e, and return $A_e \cup B_e \cup e$.

(c) If $|A_e| = |A_f| = |A_{ef}|$, call the algorithm on (G_2, F_2) to obtain an extension $B_\emptyset$. If $B_\emptyset$ contains e return $A_e \cup B_\emptyset$, otherwise return $A_f \cup B_\emptyset$.

`Condition 2`: Suppose that $g = xy \in E(G) - F$. Recursively call the algorithm on $(G_1, \{e, f\} \cup F_1), (G_1, \{f, g\} \cup F_1), (G_1, \{e, g\} \cup F_1)$ and $(G_1, \{e, f, g\} \cup F_1)$ to obtain extensions A_{ef}, A_{fg}, A_{eg} and A_{efg}, respectively.

(a) If $|A_{efg}| = |A_{ef}| - 1 = |A_{fg}| - 1 = |A_{eg}| - 1$, call the algorithm on $(G_2, \{e, f, g\} \cup F_2)$ to obtain an extension B_{efg}, and return $A_{efg} \cup B_{efg} \cup \{e, f, g\}$.

(b) Exactly one of A_{ef}, A_{eg}, A_{fg} is equal to A_{efg} in cardinality. Without loss of generality, suppose that $|A_{ef}| = |A_{efg}| = |A_{fg}| - 1 = |A_{eg}| - 1$. Call the algorithm on $(G_2, \{e, f\} \cup F_2)$ to obtain an extension B_{ef}, and return $A_{ef} \cup B_{ef} \cup \{e, f\}$.

(c) Exactly two of A_{ef}, A_{eg}, A_{fg} are equal to A_{efg} in cardinality. Suppose wlog that $|A_{ef}| = |A_{fg}| = |A_{efg}| = |A_{eg}| - 1$. Call the algorithm on $(G_2, f \cup F_2)$ to obtain an extension B_f. If B_f contains e return $A_{ef} \cup B_f \cup f$, otherwise return $A_{fg} \cup B_f \cup f$.

(d) If $|A_{ef}| = |A_{fg}| = |A_{eg}| = |A_{efg}|$, call the algorithm on (G_2, F_2) to obtain an extension $B_\emptyset$. If $B_\emptyset$ contains e and f return $A_{ef} \cup B_\emptyset$, if $B_\emptyset$ contains e and g return $A_{eg} \cup B_\emptyset$, otherwise return $A_{fg} \cup B_\emptyset$.

We omit the easy proof that the graphs G_1 and G_2 are again 2-connected planar graphs, and instead begin by showing that the algorithm covers all possible cases.

Claim 3.17. *If $g = xy \notin E(G) - F$, we may assume that*

$$|A_e| - 1 \leq |A_{ef}| \leq |A_e| \text{ and } |A_f| - 1 \leq |A_{ef}| \leq |A_f|.$$

Proof. If $|A_e| \geq |A_{ef}| + 2$, set A_e to be $A_{ef} \cup f$. Then clearly A_e is an extension of $e \cup F_1$, and $|A_e| - 1 \leq |A_{ef}|$. If $|A_{ef}| \geq |A_e| + 1$, set A_{ef} to be $A_e - f$. Then clearly A_{ef} is an extension of $ef \cup F_1$, and $|A_{ef}| \leq |A_e|$. The proof for A_f is analogous. □

Claim 3.18. *If $g = xy \in E(G) - F$, we may assume that*

$$|A_{ef}| - 1 \leq |A_{efg}| \leq |A_{ef}|$$

and

$$|A_{fg}| - 1 \leq |A_{efg}| \leq |A_{fg}|$$

and

$$|A_{eg}| - 1 \leq |A_{efg}| \leq |A_{eg}|.$$

Proof. Similar to the proof of Claim 3.17. □

We now establish that in the situations where a recursive call is made, the algorithm returns a feasible solution.

Claim 3.19. *Any star-cutset S' in G is a star-cutset in G_1 or G_2.*

Proof. S is a cutset in G_1 and G_2, so if $S' = S$ we are done. No star-cutset crosses S, so S' is contained in either $V(G_1)$ or $V(G_2)$. Without loss of generality, assume the former. Because $S' \neq S$, we know that at least one of $\{v, x, y\}$, say x, is not contained in S'. Clearly every vertex in C_2 has path to x in $G - S'$. Since S' is a cutset in G, there must be some vertex w separated from x in $G - S'$. As w cannot be in C_2, it must be in $S \cup C_1$. But clearly there can be no path from w to x in $G_1 - S'$ if there is no path in $G - S'$, so S' is a star-cutset in G_1. □

Claim 3.19 implies that if there still is a cutset in G none of whose edges are protected, this cutset is also present in either G_1 or G_2, contradicting the fact that the recursive calls return feasible solutions. Also note that when combining an extension in G_1 with an extension in G_2, we also add at least one of e and f (respectively two of e, f, and g if $xy \in E(G) - F$).

The difficult part of the analysis of the performance guarantee is to show that two extensions in G_1 and G_2 that are within α of their respective local optimal extension combine to give an extension of F in G that is within α of an optimal extension, i.e. that there is no increase in the approximation factor at the recursive step. This is achieved by the following claim and its corollary.

Claim 3.20. *Every star-cutset in G_1 or G_2 is also a star-cutset in G.*

Proof. Let S' be a star-cutset in G_1. The edges from v_1^* to the vertices of S are protected edges in F_1, so v_1^* does not occur in any star-cutset in G_1. Thus, S' is also a star in G. Take two vertices u and w in different components of $G_1 - S'$. If S' is not a star-cutset in G, then there is a path P from u to w in G; this path must use vertices from C_2 in $G - S$. Now any subpath of P through C_2 has both its endpoints in S. Then we can shortcut this subpath, since all three vertices of S are pairwise connected via the protected edges incident to v_1^*. So we obtain a path from u to w in G_1, a contradiction. By symmetry, every star-cutset in G_2 is also a star-cutset in G. □

In essence, this implies that optimal extensions in G_1 and G_2 combine to form an optimal extension of F in G. This is made precise by the following corollary.

Corollary 3.21. *Let Z be an optimal extension of F in G, and let $Z_1 = Z \cap E(G_1)$ and $Z_2 = Z \cap E(G_2)$. Denote by $J_\diamond$ an optimal extension of $\diamond \cup F_1$ in G_1, and by $K_\diamond$ an optimal extension of $\diamond \cup F_2$ in G_2.*

If $g = xy \notin E(G) - F$, then the following hold:

- *If Z contains e but not f, then $|J_e| + |K_e| + 1 \leq |Z|$.*
- *If Z contains e and f, then $|J_{ef}| + |K_{ef}| + 2 \leq |Z|$.*

(The first assertion extends to the symmetric case in the obvious manner.)

Proof. By Claim 3.20, Z_1 is an extension of F_1 in G_1. Suppose that Z contains exactly one of e and f, wlog e. Then $Z_1 - e$ is an extension of $e \cup F_1$ in G_1. Since J_e is an optimal extension of $e \cup F_1$, we have that $|J_e| \leq |Z_1| - 1$. By an analogous argument, we also have $|K_e| \leq |Z_2| - 1$. Moreover, if $e \in Z$ but $f \notin Z$, we have that $|Z| = |Z_1| + |Z_2| - 1$. It follows that

$$|J_e \cup K_e \cup e| \leq |J_e| + |K_e| + 1 \leq |Z_1| - 1 + |Z_2| - 1 + 1 \leq |Z|$$

as claimed. The other case is similar. □

Similary we may prove that:

Corollary 3.22. *Let Z be an optimal extension of F in G, and let $Z_1 = Z \cap E(G_1)$ and $Z_2 = Z \cap E(G_2)$. Denote by $J_\diamond$ an optimal extension of $\diamond \cup F_1$ in G_1, and by $K_\diamond$ an optimal extension of $\diamond \cup F_2$ in G_2.*

If $g = xy \in E(G) - F$, then the following hold:

- *If Z contains e and f but not g, then $|J_{ef}| + |K_{ef}| + 2 \leq |Z|$.*
- *If Z contains e, f and g, then $|J_{efg}| + |K_{efg}| + 3 \leq |Z|$.*

(The first assertion extends to the other symmetric cases in the obvious manner.)

The following claims give inequalities for the optimal extensions that will be used in the analysis. It turns out that, just like the approximate solutions returned by the recursive calls, the sizes of the optimal extensions in G_1 (and G_2) don't differ by much.

The proofs of the following two claims are similar to the proof of Claim 3.17.

Claim 3.23. *Let $J_\diamond$ be an optimal extension of $\diamond \cup F_1$ in G_1. If $g = xy \notin E(G) - F$ then the following hold:*

(a) $|J_e| - 1 \leq |J_{ef}| \leq |J_e|$ *and* $|J_f| - 1 \leq |J_{ef}| \leq |J_f|$.

(b) $|J_\emptyset| \leq |J_e| + 1$ *and* $|J_\emptyset| \leq |J_f| + 1$.

Analogous inequalities hold for optimal extensions $K_\diamond$ of $\diamond \cup F_1$ in G_2. □

Claim 3.24. *Let $J_\diamond$ be an optimal extension of $\diamond \cup F_1$ in G_1. If $g = xy \in E(G) - F$, then the following hold:*

(a) $|J_{ef}| - 1 \leq |J_{efg}| \leq |J_{ef}|$.

(b) $|J_e| \leq \min(|J_{ef}|, |J_{eg}|) + 1 \leq |J_{efg}| + 2$.

(c) $|J_\emptyset| \leq \min(|J_{ef}|, |J_{fg}|, |J_{eg}|) + 2 \leq |J_{efg}| + 3$.

(These inequalities extend to the other symmetric cases and to optimal extensions in G_2 in the obvious manner.) □

We split the analysis into several cases, according to which solution the algorithm returns. In all cases, we assume that a recursive call on G_1 returns an extension $A_\diamond$ of $\diamond \cup F_1$ such that $|A_\diamond| \leq \alpha |J_\diamond|$, where $J_\diamond$ is an optimal extension, and that each recursive call on G_2 returns an extension $B_\diamond$ of $\diamond \cup F_2$ such that $|B_\diamond| \leq \alpha |K_\diamond|$. We will say more about the value of α later, but the following analysis is true for any $\alpha \geq 1$.

We need to distinguish between the cases $g = xy \notin E(G) - F$ and $g = xy \in E(G) - F$. In what follows, we will only discuss asymmetric cases, i.e. we will omit for example the case where the algorithm returns $A_f \cup B_f \cup f$, as it is analogous to the case where the algorithm returns $A_e \cup B_e \cup e$.

`Condition 1.` First assume that $g = xy \notin E(G) - F$.

(a) The algorithm returns $A_{ef} \cup B_{ef} \cup ef$, a solution of size $|A_{ef}| + |B_{ef}| + 2$. This only happens if $|A_{ef}| = |A_e| - 1 = |A_f| - 1$.

- If there exists an optimal solution Z containing e and f, the size of our solution satisfies

$$|A_{ef}| + |B_{ef}| + 2 \leq \alpha(|J_{ef}| + |K_{ef}|) + 2 \leq \alpha(|J_{ef}| + |K_{ef}| + 2) \leq \alpha |Z|.$$

by Corollary 3.21.

- If there is an optimal solution Z containing e but not f, then

$$|A_{ef}| + |B_{ef}| + 2 = |A_e| - 1 + |B_{ef}| + 2 \leq \alpha(|J_e| + |K_{ef}|) + 1 \leq \alpha(|J_e| + |K_{ef}| + 1) \leq \alpha |Z|$$

since $|K_{ef}| \leq |K_e|$.

Similarly if there is an optimal solution Z containing f but not e.

(b) The algorithm returns $A_e \cup B_e \cup e$, which only happens if $|A_e| = |A_{ef}| = |A_f| - 1$. The size of this solution is $|A_e \cup B_e \cup e| \leq |A_e| + |B_e| + 1$.

- If there is an optimal solution Z containing e and f, we have that

$$|A_e| + |B_e| + 1 = |A_{ef}| + |B_e| + 1 \leq \alpha(|J_{ef}| + |K_e|) + 1 \leq \alpha(|J_{ef}| + |K_e| + 1) \leq \alpha |Z|.$$

since $|K_e| \leq |K_{ef}| + 1$.

- If there is an optimal solution Z containing e but not f, we have that

$$|A_e| + |B_e| + 1 \leq \alpha(|J_e| + |K_e|) + 1 \leq \alpha(|J_e| + |K_e| + 1) \leq \alpha |Z|.$$

- If there is an optimal solution Z containing f but not e, then

$$|A_e| + |B_e| + 1 = |A_f| + |B_e| \leq \alpha(|J_f| + |K_e|) \leq \alpha |Z|$$

since $|K_e| \leq |K_f| + 1$.

(c) The algorithm returns $A_e \cup B_\emptyset$, which only happens if $|A_e| = |A_f| = |A_{ef}|$ and $e \in B_\emptyset$. The size of this solution is $|A_e \cup B_\emptyset| \leq |A_e| + |B_\emptyset|$.

- If there is an optimal solution Z containing e and f, we have that

$$|A_e| + |B_\emptyset| = |A_{ef}| + |B_\emptyset| \leq \alpha(|J_{ef}| + |K_{ef}| + 2) \leq \alpha |Z|.$$

since $|K_\emptyset| \leq |K_{ef}| + 2$.

- If there is an optimal solution Z containing e but not f, we have that

$$|A_e| + |B_\emptyset| \leq \alpha(|J_e| + |K_e| + 1) \leq \alpha|Z|.$$

since $|K_\emptyset| \leq |K_e| + 1$.

- If there is an optimal solution Z containing f but not e, then

$$|A_e| + |B_\emptyset| = |A_f| + |B_\emptyset| \leq \alpha(|J_f| + |K_f| + 1) \leq \alpha|Z|.$$

since $|K_\emptyset| \leq |K_f| + 1$.

`Condition 2.` Now suppose that $g = xy \in E(G) - F$.

(a) The algorithm returns $A_{efg} \cup B_{efg} \cup efg$, which only happens if $|A_{efg}| = |A_{ef}| - 1 = |A_{fg}| - 1 = |A_{eg}| - 1$.

- If there is an optimal solution Z containing e, f, g, then

$$|A_{efg}| + |B_{efg}| + 3 \leq \alpha(|J_{efg}| + |K_{efg}| + 3) \leq \alpha|Z|.$$

- If there is an optimal solution Z containing e and f, but not g then

$$|A_{efg}| + |B_{efg}| + 3 = |A_{ef}| - 1 + |B_{efg}| + 3 \leq \alpha(|J_{ef}| + |K_{efg}| + 2) \leq \alpha|Z|$$

since $|K_{efg}| \leq |K_{ef}|$. Similarly for optimal solutions containing e, g but not f or f, g but not e.

(b) The algorithm returns $A_{ef} \cup B_{ef} \cup ef$, which happens if $|A_{ef}| = |A_{efg}| = |A_{fg}| - 1 = |A_{eg}| - 1$. This solution has size $|A_{ef} \cup B_{ef} \cup ef| \leq |A_{ef}| + |B_{ef}| + 2$.

- If there is an optimal solution Z containing e, f, g, then

$$|A_{ef}| + |B_{ef}| + 2 = |A_{efg} + |B_{ef} + 2 \leq \alpha(|J_{efg}| + |K_{ef}| + 2) \leq \alpha|Z|$$

since $|K_{ef}| \leq |K_{efg}| + 1$.

- If there is an optimal solution Z containing e and f, but not g then

$$|A_{ef}| + |B_{ef}| + 2 \leq \alpha(|J_{ef}| + |K_{ef}| + 2) \leq \alpha|Z|.$$

- If there is an optimal solution Z containing f and g, but not e then

$$|A_{ef}| + |B_{ef}| + 2 = |A_{fg}| - 1 + |B_{ef}| + 2 \leq \alpha(|J_{fg}| + |K_{ef}| + 1) \leq \alpha|Z|$$

since $|K_{ef}| \leq |K_{fg}| + 1$. Similarly if Z contains e and g, but not f.

(c) The algorithm returns $A_{ef} \cup B_f \cup f$, which happens if $|A_{ef}| = |A_{fg}| = |A_{efg}| = |A_{eg}| - 1$ and B_f contains e. This solution has size $|A_{ef} \cup B_f \cup f| \leq |A_{ef}| + |B_f| + 1$.

- If there is an optimal solution Z containing e, f, g, then

$$|A_{ef}| + |B_f| + 1 = |A_{efg}| + |B_f| + 1 \leq \alpha(|J_{efg}| + |K_f| + 1) \leq \alpha|Z|$$

since $|K_f| \leq |K_{efg}| + 2$.

- If there is an optimal solution Z containing e and f, but not g then

$$|A_{ef}| + |B_f| + 1 \leq \alpha(|J_{ef}| + |K_f| + 1) \leq \alpha|Z|$$

since $|K_f| \le |K_{ef}| + 1$.

- If there is an optimal solution Z containing f and g, but not e then

$$|A_{ef}| + |B_f| + 1 = |A_{fg}| + |B_f| + 1 \le \alpha(|J_{fg}| + |K_f| + 1) \le \alpha|Z|$$

since $|K_f| \le |K_{fg}| + 1$.

- If there is an optimal solution Z containing e and g, but not f then

$$|A_{ef}| + |B_f| + 1 = |A_{eg}| - 1 + |B_f| + 1 \le \alpha(|J_{eg}| + |K_f|) \le \alpha|Z|$$

since $|K_f| \le |K_e| + 1 \le |K_{eg}| + 2$.

(d) The algorithm returns $A_{ef} \cup B_\emptyset$, which happens if $|A_{ef}| = |A_{fg}| = |A_{eg}| = |A_{efg}|$ and $B_\emptyset$ contains e and f. This solution has size $|A_{ef} \cup B_\emptyset| \le |A_{ef}| + |B_\emptyset|$.

- If there is an optimal solution Z containing e, f, g then

$$|A_{ef}| + |B_\emptyset| = |A_{efg}| + |B_\emptyset| \le \alpha(|J_{efg}| + |K_\emptyset|) \le \alpha|Z|$$

since $|K_\emptyset| \le |K_{efg}| + 3$.

- If there is an optimal solution Z containing e and f but not g, then

$$|A_{ef}| + |B_\emptyset| \le \alpha(|J_{ef}| + |K_\emptyset|) \le \alpha|Z|$$

since $|K_\emptyset| \le |K_{ef}| + 2$.

- If there is an optimal solution Z containing f and g but not e, then

$$|A_{ef}| + |B_\emptyset| = |A_{fg}| + |B_\emptyset| \le \alpha(|J_{fg}| + |K_\emptyset|) \le \alpha|Z|$$

since $|K_\emptyset| \le |K_{fg}| + 2$. Similarly if there is an optimal solution Z containing e and g but not f.

This concludes the proof of the fact that the recursive step combines two extensions in G_1 respectively G_2 that are within α of an optimal local extension to form an extension of F in G that is within α of an optimal extension.

3.3.4 Two strongly attached components and a weakly attached component

`Case IV.` For the fourth case, we have two strongly attached components and at least one weakly attached component.

In this case, the graphs G_1 and G_2 we recurse on are slightly different than in Case III(a), but the analysis is virtually identical. Consider the graph $H = G[S \cup C_1 \cup C_2 \cup C_3]$. Let G_1 be the graph obtained from H by contracting C_2 into a vertex v_1^*, and by contracting C_3 into a vertex w_1^*. Similarly, let G_2 be the graph obtained from H by contracting C_1 into a vertex v_2^* and C_3 into a vertex w_2^*. So G_1 and G_2 are as in Case III(a), except that both have an additional vertex which has edges to x and y only. Define $m_i = |E(G_i)|$.

Let F_1 be the set of edges obtained from $F \cap E(G_1)$ by adding the three edges from v_1^* to x, y, v and the two edges from w_1^* to x, y. Define F_2 similarly.

For $i \ge 3$, define $G_i = G[S \cup C_i]$, and let $F_i = F \cap E(G_i)$.

On G_1 and G_2, call the algorithm recursively as described in Case III, where we may assume that $m_1 \le m_2$. Call the algorithm on $(G_1, e \cup F_1)$, $(G_1, f \cup F_1)$ and $(G_1, \{e, f\} \cup F_1)$, and then call the algorithm on G_2 accordingly. Observe that since there exists a weakly attached component, we need not consider the case $xy \in E(G) - F$. Suppose this returns a set of edges A' that intersects all star-cutsets that are either in G_1 or G_2.

For $i \ge 3$ such that C_i is non-trivial, call the algorithm recursively on (G_i, F_i) to obtain extensions A_i. For $i \ge 3$ such that $C_i = \{c_i\}$, set $A_i = \{c_i x\}$. If there is only one weakly attached component C_3 and it is trivial, set $A_3 = \emptyset$. Return $A := \cup_{i \ge 3} A_i \cup A'$.

It is easy to prove that the graphs G_i are again 2-connected planar graphs. The proof of the fact that the extensions in the smaller graph G_1 differ by at most 1 and that the recursive step, therefore, doesn't miss any cases is exactly as in Case III.

Claim 3.25. *Any star-cutset S' in G is a star-cutset in some G_i, unless $S' = \{w, x, y\}$ where $C_i = \{w\}$ is a trivial weakly attached component of $G - S$.*

Proof. Note that S is a star-cutset in G_1 and G_2; so, if $S' = S$ we are done. Take $S' \neq S$. Since S is uncrossed, we must have $S' \subset V(G_i)$ for some i. There are two cases:
(i) If S' contains both x and y, then we must have $S' = \{w, x, y\}$, since xy is not an edge. Here $w \in C_i$ for some i; if C_i is weakly attached and non-trivial, then S' separates v from $C_i - S'$ in G_i, and we're done. If C_i is strongly attached, then S' separates w_i^* from v_1^*, regardless of whether C_i is trivial and equals $\{w\}$ or not.
(ii) S' contains at most one of x and y. Without loss of generality, suppose S' doesn't contain x. Clearly every vertex in C_j with $j \neq i$ is in the same component as x in $G - S'$. But S' separates some w from x in $G - S'$, so w must lie in $C_i \cup S$. Moreover, there can be no path from w to x in $G_i - S'$, since there is none in $G - S'$. Hence S' is a star-cutset in G_i, as desired. □

Claim 3.25 ensures that if we assume that all the recursive calls on the subgraphs return feasible extensions A_i and A', then $A = \cup_i A_i \cup A'$ is an extension of F in G. If not, there is a star-cutset in G none of whose edges belong to A, and this cutset would also be present in one of the subgraphs.

The converse of this is also true, and we shall need it for the analysis of the approximation guarantee.

Claim 3.26. *If S' is a star-cutset in some G_i, then it is also a star-cutset in G.*

Proof. Just as in the previous versions of this claim, the proof follows from the foolowing two facts. Namely, that S' must be contained in $S \cup C_i$ for some i, because the edges from v_i^* and w_i^* to S are protected, and that any path in $G - S'$ using vertices from some C_j with $j \neq i$ must pass through S and can therefore be short-cutted to a path in $G_i - S'$ using either v or one of the contracted vertices v_i^* or w_i^*. □

Let Z be an optimal extension of F in G. Let $Z_i = Z \cap E(G_i)$, and define $Z' = Z_1 \cup Z_2$. The analysis presented in Case III shows that the set of edges A' returned by the recursion on G_1 and G_2 satisfies $|A'| \leq \alpha |Z'|$. We assume that a recursive call on (G_i, F_i) for $i \geq 3$ returns an extension A_i of F_i such that $|A_i| \leq \alpha |J_i|$, where J_i is an optimal extension.

If Z contains exactly one of e and f, we have that $|J_i| \leq |Z_i| - 1$, as argued in Case I. When we set $A_i = \emptyset$, clearly $|A_i| \leq |Z_i| - 1$, and if we set $A_i = \{c_i x\}$, we also have $|A_i| \leq |Z_i| - 1$. Hence the size of our solution satisfies

$$|A' \cup \cup_{i \geq 3} A_i| \leq |A'| + \sum_{i \geq 3} |A_i| \leq \alpha |Z'| + \alpha (\sum_{i \geq 3} (|Z_i| - 1) \leq \alpha |Z|.$$

If Z contains e and f, we have that $|J_i| \leq |Z_i| - 2$, as argued in Case II. When we set $A_i = \emptyset$, clearly $|A_i| \leq |Z_i| - 2$, and if we set $A_i = \{c_i x\}$, we also have $|A_i| \leq |Z_i| - 2$. Hence the size of our solution satisfies

$$|A' \cup \cup_{i \geq 3} A_i| \leq |A'| + \sum_{i \geq 3} |A_i| \leq \alpha |Z'| + \alpha (\sum_{i \geq 3} (|Z_i| - 2) \leq \alpha |Z|.$$

3.4 Running Time Analysis

Before analysing the runtime of the algorithm, we give a brief summary.

Given G and a set of edges F, we can list in polynomial time all the vulnerable minimal star-cutsets. If there are none, we return $A = \emptyset$ as an optimal extension of F. Otherwise, let B be the set of vulnerable separating edges. If all vulnerable star-cutsets of G are separating edges, we return B as an optimal extension of F and stop. If B is not empty, but not all minimal star-cutsets are separating edges, add the edges in B to F.

So now we may assume that all minimal star-cutsets of G consist of 3 vertices. If G is a wheel graph, we find an extension as described in Section 3.1. If all star-cutsets S are such that at most one component of $G - S$ is non-trivial, we find an extension A of F as described in Section 3.2. If neither of these two cases holds, we know by Theorem 2.4 that there must be an uncrossed star-cutset S such that $G - S$ has at least 2 non-trivial components. We find such an S and recurse as described in Section 3.3.

Now, if G is a wheel graph, it's clear that the algorithm runs in polynomial time. If every star-cutset S is such that at most one component of $G - S$ is non-trivial, it is also straightforward to verify that the algorithm runs in polynomial time: Obtaining the graph L^* from G takes polynomial time; since L is a subgraph of the line graph of G, we have $|V(L^*)| \leq |E(G)| \leq 3|V(G)| - 6$, as G is planar. As discussed in Section 3.2, finding an approximate vertex cover in a planar graph can be done in polynomial time.

Having shown that the algorithm performs in polynomial time on the leaves of the recursion tree, it remains to prove that the recursive steps can also be implemented efficiently.

Claim 3.27. *The algorithm terminates.*

Proof. As noted above, we have a well-defined algorithm that runs in polynomial time on the leaves of the recursion tree. To complete the proof, observe that regardless of whether we are in Case I, II, III or IV, all the graphs G_i we recurse on satisfy $|V(G_i)| < |V(G)|$. Hence, there is no possibility of looping, and the algorithm must eventually terminate. □

In order to analyze the runtime, it will turn out to be more convenient to argue in terms of edges. So, let $m = |E(G)|$, let S and the components C_i be as in the description of the algorithm presented in Section 3.3. For each i, let $m_i = |E(G_i)|$.

Theorem 3.28. *Let $T(m)$ be the running time of the algorithm. If the algorithm recurses, then we have*

$$T(m) \leq 4T(m_1) + \sum_{i=2}^{r} T(m_i) + O(m),$$

where r is the number of components of $G - S$. Moreover, the m_i satisfy

$$\sum_{i \geq 1} m_i \leq m + 2r + 8 \quad \text{and} \quad m_1 \leq \frac{m}{2} + 5.$$

Proof. `Case I.` All components are weakly attached. By definition of the graphs G_i, we have that

$$\sum_{i \geq 1} m_i = m + 2(r-1) = m + 2r - 2 \leq m + 2r + 8,$$

where the first equality holds because e and f are counted r times. Moreover, we lose no generality in assuming that $m_1 = \min_i(m_i)$, which implies that

$$m_1 \leq \frac{1}{r} \sum_i m_i = \frac{m + 2r - 2}{r} \leq \frac{m}{r} + 2 \leq \frac{m}{2} + 5$$

since $r \geq 2$.

Given the partial extensions A_i, we can combine them in time $O(m)$. So $T(m)$ satisfies

$$T(m) = \sum_i T(m_i) + O(m) \leq 4T(m_1) + \sum_{i \geq 2} T(m_i) + O(m),$$

as claimed.

`Case II.` There is exactly one strongly attached component. Then we get

$$\sum_{i \geq 1} m_i = m + 2(r-1) + 2 = m + 2r \leq m + 2r + 8,$$

where the first equality holds because e and f are counted r times, and we have to take into account the edges from $S \cup C_1$ to the contracted component C_2. As above, combining the partial extensions can be done in time $O(m)$, so the total running time is

$$T(m) = \sum_i T(m_i) + O(m).$$

Now it is not necessarily true that $m_1 = \min_i(m_i)$. However, setting $i_0 = \operatorname{argmin}(m_i)$, we have that

$$m_{i_0} \leq \frac{m+2r}{r} = \frac{m}{r} + 2 \leq \frac{m}{2} + 5,$$

and so

$$T(m) = \sum_i T(m_i) + O(m) \leq 4T(m_{i_0}) + \sum_{i \neq i_0} T(m_i) + O(m).$$

Swapping the indices 1 and i_0 now yields the assertions of the theorem.

`Case III.` There are two strongly attached components, and no weakly attached components. If $g = xy \in E(G) - F$ then

$$m_1 + m_2 = m + 9,$$

since we're counting e, f and g twice, and there are three edges from v_1^* to S in G_1 and three edges from v_2^* to S in G_2. If $xy \notin E(G) - F$, then

$$m_1 + m_2 = m + 8.$$

In either case, we obtain that

$$m_1 + m_2 \leq m + 2r + 8,$$

as claimed.

Next, we had labelled G_1 and G_2 so that $m_1 \leq m_2$, which implies that

$$m_1 \leq \frac{m+9}{2} \leq \frac{m}{2} + 5,$$

as required.

Now the algorithm makes one recursive call on G_2, and either 3 or 4 calls on G_1, so

$$T(m) \leq 4T(m_1) + T(m_2) + O(m).$$

`Case IV.` There are two strongly attached components and at least one weakly attached component. Now

$$\sum_i m_i = m + 2(r-1) + 10 = m + 2r + 8$$

since e and f are counted r times, and we use an extra 5 edges for G_1 and 5 edges for G_2 (namely the edges from v_i^* and w_i^* to S).

We labelled G_1 and G_2 so that $m_1 \leq m_2$; thus, $m_1 \leq \frac{m_1+m_2}{2}$. Now observe that $m_1 + m_2 = m + 2r + 8 - \sum_{i \geq 3}^r m_i$, and that $m_i \geq 4$ for all i. So

$$m_1 + m_2 \leq m + 2r + 8 - 4(r-2) = m - 2r + 16 \leq m + 10$$

since $r \geq 3$. Hence $m_1 \leq \frac{m}{2} + 5$ as claimed.

The algorithm makes 3 recursive calls on G_1, and at most 1 on G_i for $i \geq 2$, so

$$T(m) \leq 4T(m_1) + \sum_{i \geq 2} T(m_i) + O(m)$$

as claimed. □

So we obtain the bound
$$T(m) \leq 4T(m_1) + \sum_{i\geq 2} T(m_i) + O(m)$$
where $m_1 \leq \frac{m}{2} + 5$ and $\sum_{i\geq 1} m_i \leq m + 2r + 8$. Observe further that by construction of the graphs G_i, we have $m_i \leq m - 1$ for all i and $m_i \geq 4$ for all i. Then, by consideration of the base cases and applying standard techniques, we obtain that the recursive running time of the algorithm is polynomial.

Corollary 3.29. *The algorithm runs in time $O(m^3)$.* □

3.5 A Hardness Result

We do not know if a PTAS is the best possible result for the network protection problem in planar graphs: It is an open problem to determine whether or not the problem is NP-complete. In fact, our analysis of Section 3.3 shows that, to obtain a polynomial time algorithm, it suffices to find a efficient algorithm for the very special case in which every star-cutset induces at most one non-trivial component, i.e. Case (a) of Theorem 2.4. So it is certainly plausible that there is a polynomial time algorithm for the problem in planar graphs.

Instead, in this section we present a weaker hardness result. We show that the vertex variant of network protection problem in planar graphs is indeed NP-complete. In the vertex version we protect vertices rather than edges; consequently, our goal is to find the smallest cardinality set of vertices P such that P intersects the vertex set of every star-cutset in the planar graph G.

Theorem 3.30. *The vertex variant of the network protection problem is NP-complete in planar graphs.*

Proof. To prove the result, we use a straightforward reduction from the vertex cover problem. Let $G = (V, E)$ be a 2-vertex connected planar graph for which we wish to find an optimal vertex cover. We show how to do this by finding an optimal edge protection set in an auxiliary graph G'. We construct G' by replacing each edge $e = (u, v)$ of G with a 4-cycle u, e_1, v, e_2 plus the chord (e_1, e_2). We will call u and v *original* vertices and the e_1 and e_2 *dummy* vertices. It is clear that any planar embedding of G yields a planar embedding of G'; so G' is planar.

We will show that an optimal solution to the vertex network protection problem in G' would give an optimal vertex cover in G.

Claim 3.31. *A star-cutsets in G' is minimal if and only if it is of the form $\{e_i, u, v\}$, where $i \in \{1, 2\}$ and $e = (u, v)$ is an edge of G.*

Proof. For any edge $e = (u, v)$, the star $\{e_i, u, v\}$ separates e_{3-i} from the rest of G'. So this star it is a cutset. It is minimal for the following reason. G' is 2-connected (because G is) and so a smaller star-cutset must contain both u and v, but uv is not an edge in G' so cannot be a separating edge.

So it remains to show that any minimal star-cutset has this form. As G is 2-connected, the structure of the reduction implies that no original vertex v can centre a star-cutset in G'. So each star-cutset in G' must be centred at a vertex e_1 or e_2 corresponding to an edge $e = (u, v)$ of G. If $\{e_i, e_{3-i}, u, v\}$ is a star cutset, then it cannot be minimal as we have seen that $\{e_i, u, v\}$ is a star-cutset. □

So the minimal star-cutsets in G' are those stars $\{e_i, u, v\}$ corresponding to edges uv in G. This implies that a vertex cover in G is a feasible solution to the vertex protection problem in G'. The following claim establishes the converse.

Claim 3.32. *There is an optimal solution to the vertex network protection problem in G' that contains only vertices of G.*

Proof. From Claim 3.31, each e_i is in exactly one minimal star-cutset, $\{e_i, u, v\}$. This implies that if e_i is in some optimal solution P then we can replace e_i by either u or v to obtain an optimal solution with one less dummy vertex. □

Claim 3.32 implies that a solution P^* to the vertex protection problem in G' yields a vertex cover of the same cardinality in G.

Hence, if we could solve the vertex protection problem optimally in an arbitrary planar graph G, we could find a minimum vertex cover in an arbitrary 2-connected planar graph. Given a candidate solution $A \subseteq V(G)$, it is easy to verify in polynomial time whether the vertices in A hit all the minimal star-cutsets of G. It follows that the vertex protection problem is NP-complete, since finding a minimum vertex cover is NP-complete even in a 3-connected cubic planar graph ([12]). □

4 Non-planar Graphs

Here we settle the complexity of the network protection problem for non-planar graphs by providing logarithmic lower and upper approximability bounds[3]. Here we show how to obtain an $2(c+1)\log n$ approximation algorithm, where $n = |V(G)|$ and c is an absolute constant ($c \approx 1.6$). We complement this by providing a lower bound of $\log n$, showing that, upto a constant factor, our result is the best possible.

4.1 An $O(\log n)$ inapproximability result

Our hardness result is obtained by showing that the inapproximability results for the set cover problem apply for the network protection problem.

Theorem 4.1. *For any $\epsilon > 0$, there is no $(1-\epsilon)\log n$-approximation algorithm for the network protection problem unless $NP \subset TIME(n^{O(\log\log n)})$.*

Proof. Take an instance of Set Cover with ground set $V = \{v_1, \ldots, v_n\}$ and a collection of subsets $\mathcal{S} = \{S_1, \ldots, S_m\}$ of V.

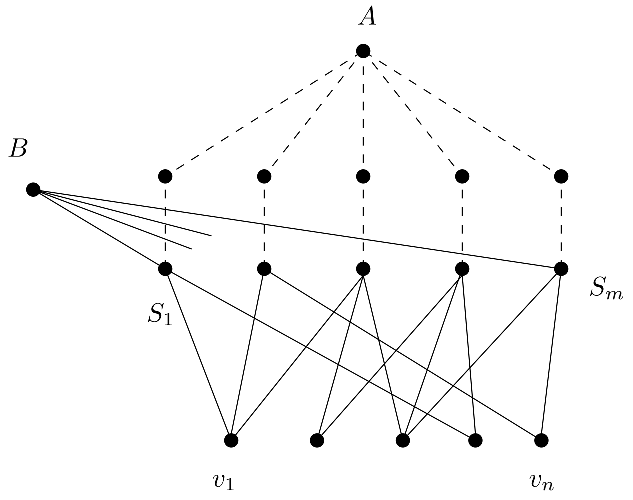


Figure 4: Reduction graph

Figure 4.1 shows the corresponding reduction graph H. It has a vertex for each set S_j, a vertex for each element of the ground set v_i, and two additional vertices A and B. For each set S_j there is a path of length one (that is, an edge) to B and a path of length two to A. Furthermore, there is an edge (v_i, S_j) if and only if element v_i is in the set S_j.

[3]We remark that both these results generalise to the weighted case where we desire a minimum weight protection set rather than just a minimum cardinality protection set.

We may make a couple of simple assumptions. Firstly, each element v_i is in at least two sets; if it is in only one set then that set must be chosen in any feasible solution. Secondly, for no pair of elements, v_i and v_j, may it be the case that each set containing v_i also contains v_j; otherwise we can instead consider the set cover instance $(V - v_j, \mathcal{S})$. Obsevre that these two assumptions imply that neither the v_i vertices nor the S_j vertices S_j, nor the vertices on the paths from the S_j to A can centre a star-cutset. It is easy to see that A cannot centre a star-cutset. Thus, the *only* vertex that can possibly centre a star-cutset is B.

Now suppose we have a feasible set cover $\mathcal{C} = \{S_1, \ldots, S_r\}$. Let $F = \{(B, S_1), (B, S_2), \ldots, (B, S_r))$, that is, the set of edges from B to the vertices corresponding to the sets in the cover. We claim that F is a solution to the network protection problem. To see this, let T be a star centered at B that does not intersect F. Now, for any v_i, we know that $v_i \in S_j$ for some $j \leq r$, because $\mathcal{C}$ is a set cover. Thus, there is still a path from v_i to A in $H - T$. Clearly, for any $S_k \notin T$, there is still a path from S_k to A in $H - T$. Hence, $H - T$ is connected. Observe, also, that the solution to the network protection problem obtained in this fashion has the same cardinality as the set cover.

Conversely, a solution F to the network protection problem yields a set cover as follows. We have seen that only edges incident to B can be contained in a minimal solution. Consequently, wlog, let $F = \{(B, S_1), (B, S_2), \ldots, (B, S_r))$. Then $\mathcal{C} = \{S_j \,|\, 1 \leq j \leq r\}$ must be a set cover. If not, there is some vertex v_i that is in none of the sets $S_j \in \mathcal{C}$. So consider the star T centred at B containing each S_k with $k > r$; it is vulnerable with respect to F, and contains all the vertices corresponding to sets that contain v_i. Thus, v_i is separated from A in $G - T$, a contradiction. Again, note that the cardinality of the set cover obtained is precisely the cardinality of the solution to the network protection problem.

Hence, the cost of an optimal solution to the network protection problem equals the cost of an optimal set cover. Feige [4] showed that unless $NP \subset TIME(n^{O(\log \log n)})$, there is no polynomial time algorithm to approximate Set Cover to within $(1 - \epsilon') \log n'$ for any $\epsilon' > 0$, where n' is the number of elements of the ground set of the Set Cover instance.

Suppose for a contradiction that there were a polynomial time approximation algorithm for the network protection problem with an approximation guarantee of $(1 - \epsilon) \log n$, where $\epsilon > 0$ and n is the number of vertices in the given graph. In his hardness proof, Feige only uses instances of set cover such that the number of sets s is less than the number n' of elements in the ground set. Thus, we may assume that the reduction graph from Theorem 4.1 has at most $n = 3n' + 2$ vertices; one for each element of the ground set, and two for each set, plus the vertices A and$B$. Then, since $\log n$ and $\log n'$ only differ by a constant additive term, it follows that a $(1 - \epsilon) \log n$ approximation algorithm for the network protection problem would yield a $(1 - \epsilon') \log n'$ approximation algorithm for Set Cover for some $\epsilon' > 0$. $\square$

4.2 An $O(\log n)$ approximation algorithm

In this section, we present a polynomial time algorithm for the network protection problem whose approximation guarantee is $O(\log n)$, where n is the number of vertices of the graph.

The approach is simple. We break the problem up into n subproblems - one for each vertex $v \in V$. For each such *local problem*, our goal is to protect a set of edges $F_v \subseteq \delta(v)$ so that there are no vulnerable star-cutsets centered at v. Clearly, the union of the solutions to each local problem gives a feasible global solution $F = \cup_v F_v$. The only difficulty, therefore, is to show how to apply this approach to produce a good approximate global solution. Towards this goal, we will use the following notation. Let $\delta(v)$ denote the edges incident to v, let $\Gamma(v)$ denote the set of neighbours of v, and let $N(v) = \Gamma(v) \cup \{v\}$. We denote by OPT the cardinality of the optimal global solution and by $\text{OPT}(v)$ the cardinality of the optimal solution for the local problem at vertex v.

Theorem 4.2. *We can find local solutions F_v such that the global solution $F = \bigcup_{v \in B} F_v$ satisfies $|F| \leq O(\log n)\, \text{OPT}$.*

Proof. We begin with a straightforward observation.

Observation 4.3. *If F^* is an optimal global solution, then for any v, the set $F^* \cap \delta(v)$ is a local solution.* $\square$

Observation 4.3 implies that $|F^* \cap \delta(v)| \geq \text{OPT}(v)$.

Claim 4.4. *For each v, we can find a local solution F_v of size $|F_v| \leq (c+1)\log(n)\text{OPT}(v)$, where n is the number of vertices of G and c is a constant.*

Claim 4.4 establishes the approximation guarantee. The size of our solution is

$$\begin{aligned} |F| &\leq \sum_{v \in B} |F_v| \\ &\leq (c+1)\log n \sum_v \text{OPT}(v) \\ &\leq (c+1)\log n \sum_v |F^* \cap \delta(v)| \\ &\leq 2(c+1)\log n \cdot |F^*| \\ &= 2(c+1)\log n \cdot \text{OPT} \end{aligned}$$

where the last inequality follows from the fact that since edge of F^* is counted at most twice.

Thus, it remains to prove Claim 4.4. The trick here is give a one-to-one correspondence between local solutions and *connected dominating sets* in an auxiliary graph H_v. The result will then follow by applying an approximation algorithm for the connected dominating set problem.

So fix a vertex v, and let $C_1, \ldots, C_k$ be the components of $G - N(v)$. The auxiliary graph H_v is just the the minor of $G - v$ obtained by contracting each component C_i into a vertex t_i. We assign the vertices t_i weight 0, and each vertex $u \in \Gamma(v)$ weight 1. With these weights, we run an algorithm to find a minimum weight connected dominating set D in H_v. Let $F_v = \{(v, w) \mid w \in D \cap \Gamma(v)\}$. We claim that F_v is a local solution. If not, take a star S_v in G centred at v that is vulnerable with respect to F_v.

Clearly $S_v \cap D = \emptyset$ by definition. Thus, D remains connected in $G - S_v$. Furthermore, any vertex $u \in H_v - S_v$ that is not in D must be adjacent to a vertex of D, since D is a dominating set in H_v. So $H_v - S_v$ is connected, a contradiction.

The weight of each t_i is 0, so the cardinality of the local solution obtained is exactly the weight of D.

Conversely, given a local solution F_v we construct a connected dominating set D in H_v as follows: Let D contain all the vertices t_i and each $w \in \Gamma(v)$ such that $vw \in F_v$. Consequently, D has weight $|F_v|$, since each t_i has zero weight.

Now we need to show that D is connected and that it is a dominating set. To begin, let's show that D is a dominating set, let $u \in H_v - D$. Since D contains all the vertices t_i, u must be in $\Gamma(v) - D$. To show that u must be adjacent to a vertex of D, we have two possibilities. Firstly, if u is adjacent to some vertex t_i the clearly u is dominated. So we may assume that u is not adjacent to any vertex t_i. This implies that $N(u) \subseteq N(v)$. Again, we have two cases to deal with.

(a) $N(u) - v = N(v)$. Since F_v must contain at least one edge (v, w), it follows that u is dominated by w.

(b) $N(u) - v \neq N(v)$. Suppose, for a contradiction, that (v, w) is not in F_v for any vertex $w \in N(u)$. Then $\Gamma(u)$ is a star centred at v whose removal disconnects u from the non-empty set $N(v) - N(u)$. This contradicts the assumption that F_v is a local solution.

To conclude, we next show that D is connected. Because F_v is a local solution, $S_v = \{v\} \cup \{w \mid (v, w) \notin F_v\}$ is not a star-cutset in G. But then $G - S_v = G[D]$ is a a single component; thus D is connected.

So we have the desired bijection. Hence, if $\text{OPT}(CDS)$ is the weight of an optimal connected dominating set in G_v, we have $opt_{CDS} = opt_v$. Guha and Khuller [6] gave a $(c+1)\log(n)$-approximation algorithm for the connected dominating set problem, were c is an absolute constant ($c \approx 1.6$). So the local solution we get from the dominating set has cost at most $(c+1)\log(n) \cdot opt(v)$. This concludes the proof of the claim and, hence, of Theorem 4.2. □

Acknowledgements. We thank Elliot Anshelevich for suggesting the Network Protection Problem to us.

References

[1] Brenda S. Baker, Approximation Algorithms for NP-Complete problems on Planar Graphs, *JACM* **41** (1), 1994, pp. 153-180.

[2] D. Bienstock and S. Mattia, Using mixed-integer programming to solve power grid blackout problems, *Discrete Optimization* **4** (1), 2007, pp. 115-141.

[3] V. Chvátal, Star cutsets and perfect graphs, *JCT(B)* **39**, 1985, pp. 189-199.

[4] U. Feige, A Threshold of $\ln n$ for Approximating Set Cover, *JACM* **45** (4) , 1998, pp. 634-652.

[5] S. Finbow and A. King and G. MacGillivray and R. Rizzi, The firefighter problem for graphs of maximum degree three, *Discrete Mathematics* **307** (16), 2007, pp. 2094-2105.

[6] S. Guha and S. Khuller, Approximation Algorithms for Connected Dominating Sets, *Algorithmica* **20** (4), 1998, pp. 374-387.

[7] B.L. Hartnell, Firefighter! An application of domination, *24th Manitoba Conference on Combinatorial Mathematics and Computing* (presentation), 1995.

[8] D. Kempe and J. Kleinberg and É. Tardos, Maximizing the spread of influence through a social network, *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 137-146.

[9] B. Reed, "Skew partitions in perfect graphs", *Discrete Applied Mathematics*, **156** (7), 2008, pp. 1150-1156.

[10] N. Sonnerat and A. Vetta, Network Connectivity and Malicious Attacks, pre-print.

[11] N. Sonnerat and A. Vetta, Galaxy cutsets in graphs, pre-print.

[12] R. Uehara, NP-complete problems on a 3-connected cubic planar graph and their applications, *Technical Report TWCU-M-0004*, Tokyo Woman's Christian University, 1996.