Network Connectivity and Malicious Attacks

N. Sonnerat [a,*,1] A. Vetta [b,2]

[a] *Department of Mathematics and Statistics, McGill University.*

[b] *Department of Mathematics and Statistics, and School of Computer Science, McGill University.*

Abstract

We consider the following approach to model malicious attacks on a network G. An attack occurs at a vertex v in G and may then spread to vertices within a distance r of v; thus, the affected vertices are spanned by a subtree of depth at most r. Our goal is to build low cost networks that remain connected when the network is attacked maliciously. Specifically, given a graph G we want to find the smallest spanning subgraph H which contains no cutsets of radius r, i.e. cutsets spanned by trees of depth r. This problem, which we call the r-malconnectivity problem, generalises the 2-vertex connectivity problem. However, from a complexity point of view it is more difficult. We show that it is NP-hard to decide whether or not a graph contains a spanning subgraph with no cutsets of radius at most r, where $r \geq 4$. This implies that, in sharp contrast to vertex connectivity problems, no approximation algorithms exist for the r-malconnectivity problem, unless P=NP. Bi-criteria approximation algorithms can arise, however. We show that, given a graph G with no cutset of radius 3, there is a polynomial time algorithm which returns a spanning subgraph that has no cutset of radius 1 and that has size at most $\frac{11}{6}$ times that of the minimum such subgraph. Finally, we show that the recognition problem for testing whether a graph G itself has no cutsets of radius r is easy.

Preprint submitted to Elsevier *3 March 2009*

Key words:

1 Introduction

Graph connectivity is a fundamental concept in network design and has been extremely well studied from a complexity viewpoint (see, for example, [12], [10], [15], [2], [8], [16] amongst others). The motivation for desiring highly connected networks is to provide resilience against network failures at vertices and/or edges. For example, these failures may arise due to equipment malfunctions or due to malicious attacks on the network. Here we are concerned with the latter problem, except we generalise the form an attack may take. Specifically, instead of attacks just affecting isolated vertices/edges we allow an attack to spread/propagate through the network. Thus, our goal is to attempt to model virus-like malicious attacks. As we will see, allowing attacks to take this more general form makes the task of efficiently designing resilient networks much more difficult.

Formally, our model is as follows. We are given a network $G = (V, E)$ that is susceptible to malicious attacks. These attacks take the following form: An attack occurs at a vertex v and then begins to infect the rest of the network. Any vertex within a distance r of the attacked vertex is susceptible to infection. Thus an attack corresponds to a subtree of depth at most r rooted

* Corresponding author.
Email addresses: `sonnerat@math.mcgill.ca` (N. Sonnerat),

`vetta@math.mcgill.ca` (A. Vetta).

1 Supported in part by MITACS NCE grant G209680.

2 Supported in part by NSERC grant 28833-04 and FQRNT grant NC-98649.

at v. Observe that we will not assume that every vertex within distance r of v must be infected; however, for a vertex u to be infected there must be a path of length at most r from v to u consisting entirely of infected vertices.

We say that a graph H is r-*malconnected* if $H - V(T)$ is connected for any tree $T \subseteq H$ of depth at most r. If R is a set of vertices of H that is spanned by a tree of depth r rooted at some vertex $v \in R$ and $H - R$ is disconnected, we say that R is a cutset of radius r centred at v; if $r = 1$ we say that R is a star-cutset. This leads to the following optimization problem.

The Minimum r-Malconnected Spanning Subgraph Problem. Find a minimum sized spanning subgraph $H = (V, E')$ of $G = (V, E)$ such that H remains connected in the face of any malicious attack. That is, $H - R$ is connected for any subgraph R of radius at most r.

Observe that the minimum 0-malconnected spanning subgraph problem is equivalent to the minimum 2-vertex connected spanning subgraph problem. Consequently, the minimum r-malconnected spanning subgraph problem (or the r-malconnectivity problem) is NP-hard. As mentioned, however, things here are actually more difficult than for the standard vertex-connectivity problems. In particular, the design of efficient networks that are resistant to malicious attacks is severely hampered by the fact that the property of r-malconnectedness is *not* monotonic. Monotonicity, though, is a fundamental property of other connectivity design criteria such as vertex connectivity. Moreover, monotonicity has important consequences in network design. Monotonicity is extremely helpful in that it means every edge is useful. That is, the addition of an edge to a network can only make the network more connected. Thus, whilst the addition of an edge may increase costs, it helps towards the ultimate goal of increasing, say, vertex connectivity.

In contrast, to see that the property of malconnectedness is not monotonic, consider the simple example of a cycle C. Observe that C is r-malconnected, for any r. Any subgraph of radius r is just a path, and its removal leaves a path. However, if we add a single chord (u, v) to the cycle the graph is no longer r-malconnected. In fact, the resulting graph is not even 1-malconnected as the removal of a star centred at either u or v will disconnect the graph. This poses many difficulties in terms of network design. In particular, adding an edge not only increases costs, it may also make the ultimate goal of increasing malconnectivity impossible. The reason for this problem is simple. The addition of an edge does help increase basic connectivity *but* it also helps increase the spread and severity of a malicious attack. Our task, therefore, is to increase connectivity without making the network susceptible to malicious attacks. The purpose of this paper is to investigate the complexity of finding a favourable trade-off between these competing factors; that is, can we efficiently build highly connected networks through which malicious attacks can only spread slowly?

Our first result is a very negative one. We show, in Section 2, that the lack of monotonicity is indeed a serious problem. Specifically, we prove that it is NP-hard to decide whether a graph G even contains an r-malconnected spanning subgraph H, where $r \geq 4$. Again, contrast this with the situation of vertex (or edge) connectivity. By monotonicity, a graph G contains an r-connected spanning subgraph if and only if G itself is r-connected. Moreover, by the maxflow-mincut theorem, it is simple to test efficiently whether G is r-connected. The lack of monotonicity for malconnectedness implies that even if G is not r-malconnected it may contain a spanning subgraph H that is.

Unfortunately, our result shows that we cannot test efficiently whether or not this is the case. This, of course, implies that even designing approximation algorithms for the r-malconnectivity problem is impossible (provided P $\neq$ NP).

In the rest of the paper, we present some positive algorithmic results. We show, in Section 3, that at least we can test efficiently whether or not a graph G itself is r-malconnected. Then, in Section 4, we present a bi-criteria result. We give a factor $\frac{11}{6}$-approximation algorithm for the 1-malconnectivity problem in graphs that are 3-malconnected. Specifically, given a graph G that contains no cutsets of radius 3, we can find in polynomial time a spanning subgraph H that contains no star-cutsets and has cardinality at most $\frac{11}{6}$ times larger than the optimal such subgraph. We conclude in Section 5 with directions for future work and a set of open problems.

2 A Hardness Result

We show that the problem of deciding whether a graph G contains a r-malconnected spanning subgraph H is NP-hard by giving a reduction from 3SAT. For convenience, we assume that there are n Boolean variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$, where for each x_i, both x_i and $\bar{x}_i$ occur in exactly $s \geq 2$ clauses. The graph G we use for the reduction has the following properties. G itself is not r-malconnected, and only certain spanning subgraphs of G can possibly be r-malconnected. We will be able to set up a correspondence between these subgraphs and true-false assignments to $x_1, \ldots, x_n$. Furthermore, it will turn out that such a subgraph is actually r-malconnected if and only if its corresponding truth assignment is a satisfying assignment.

The basic idea behind the construction of G is as follows. For each variable x_i, there is a vertex x_i, a vertex $v(x_i)$, and a *variable gadget* g_i. Also, for each clause C_j there is a vertex C_j and a vertex $v(C_j)$. (It will always be clear from the context whether we are talking about a variable or clause or the corresponding vertex in the graph.) The vertices $v(x_i)$ [3] will ensure consistency: If a spanning subgraph H of G does not correspond to a valid true-false assignment, some $v(x_i)$ will be the centre of a cutset of radius r. Moreover, given a subgraph H that does correspond to a true-false assignment, the vertex $v(C_j)$ will be the centre of a cutset of radius r if and only if clause C_j is not satisfied under this assignment. Consequently, the vertices $v(C_j)$ will verify satisfiability.

We now describe the construction in detail, beginning with the the gadgets and how they fit together. For each x_i, the corresponding variable gadget g_i will resemble a complete bipartite graph, where vertices on opposite sides of the partition are connected via paths. We have a vertex labelled $[x_i, C_j]$ if x_i appears in clause C_j; these vertices make up the left side of the bipartition of g_i. Similarly, there is a vertex $[\bar{x}_i, C_k]$ if $\bar{x}_i$ appears in C_k; these vertices make up the right side of the bipartition of g_i. For every clause C_j containing x_i we connect $[x_i, C_j]$ to all the vertices $[\bar{x}_i, C_k]$ via disjoint paths of length $4r$; we label the middle vertex of such a path $z_i^{j,k}$.

We now describe how the vertices and gadgets are connected to one another. By a *long path* we will mean a path of length $4r$, all of whose internal vertices have degree 2. There is a special vertex $\mathcal{X}$ which is connected to all the $v(x_i)$ via disjoint long paths. Next, for every i, $v(x_i)$ is connected to x_i via a path of length $r-1$. Each vertex x_i is in turn connected to all the vertices $[x_i, C_j]$

[3] Here, v stands for "verifier".

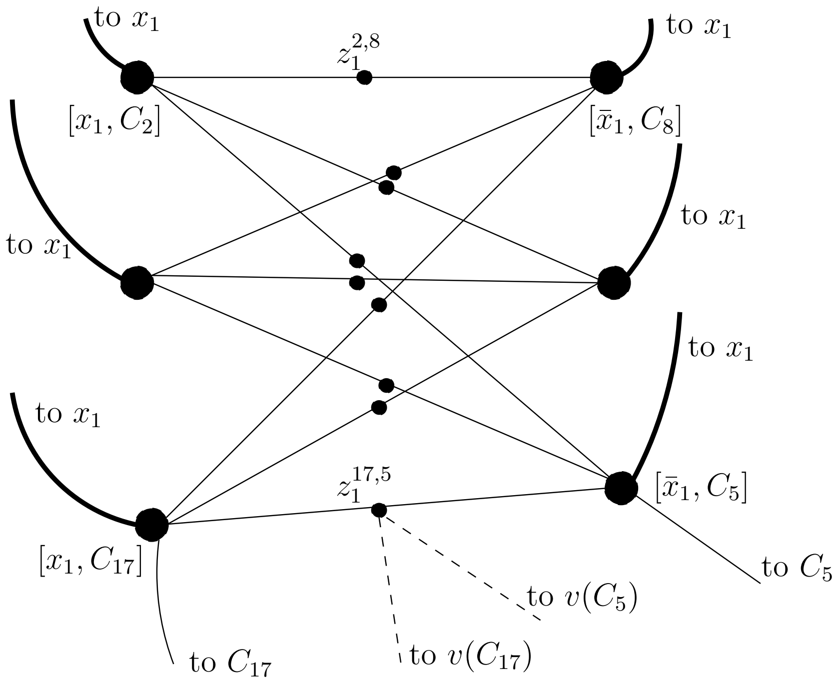

Fig. 1. A variable gadget. The solid lines are paths of length $4r$, the bold lines single edges, the dashed lines paths of length r.

and $[\bar{x}_i, C_k]$ in the corresponding gadget g_i by a single edge.

There is another special vertex $\mathcal{C}$ which is connected to $\mathcal{X}$ via a long path. The vertex $\mathcal{C}$ is also connected to all the $v(C_k)$ via disjoint long paths. Each vertex $v(C_k)$ is joined to C_k by a long path. In turn, a vertex C_k is connected through long paths to three more vertices, determined as follows: If the clause C_k contains the literal x_i then the vertex C_k is connected to the vertex $[x_i, C_k]$; if the clause C_k contains the literal $\bar{x}_i$ then vertex C_k is connected to the vertex $[\bar{x}_i, C_k]$.

The intuition behind these long paths is that, for most choices of vertex $v \in G$, any tree of depth at most r centred at v will contain only one vertex of degree at least 3. This will make the task of checking whether v centres a cutset of radius r straightforward.

To complete the construction, recall that inside every variable gadget we have paths of length $4r$ from every $[x_i, C_j]$ to every $[\bar{x}_i, C_k]$. The middle vertex $z_i^{j,k}$ on such a path is now connected to $v(C_j)$ and $v(C_k)$ via paths of length r. The vertices in the middle of these two paths will be labelled $[z_i^{j,k}, v(C_j)]$ and $[z_i^{j,k}, v(C_k)]$, respectively; we connect these two vertices to $v(x_i)$ using paths of length r. Observe that these additional paths are incident to verifier vertices. They are needed to ensure that the verifiers do their jobs with respect to consistency and satisfiability. This concludes our description of the graph G, which is illustrated in Figure 2.

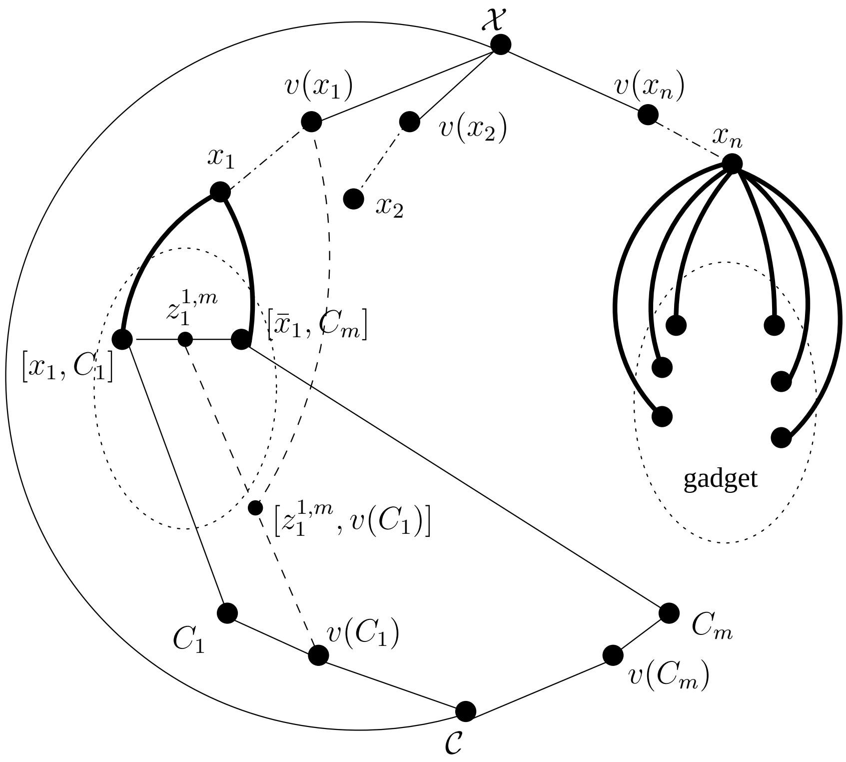

Fig. 2. The graph G. The solid lines are long paths, the bold lines single edges, the dashed lines paths of length r, and the dash-dotted lines are paths of length $r-1$.

We now proceed to show that finding a r-malconnected spanning subgraph H of G is equivalent to finding an assignment of true-false values to $x_1, \ldots, x_n$ that satisfies $C_1 \wedge \cdots \wedge C_m$. Observe that G itself is not r-malconnected, since the tree of depth r centred at $v(x_1)$, for instance, is a cutset.

For a gadget corresponding to x_i, we call the vertices $[x_i, C_j]$ *left* vertices and the vertices $[\bar{x}_i, C_k]$ *right* vertices. We also call the edges joining x_i to left vertices *left* edges, and the edges joining x_i to right vertices *right* edges. We now prove a few technical lemmas that will allow us to set up a one-to-one correspondence between true-false assignments to the variables x_i and certain spanning subgraphs of G.

Lemma 2.1 *Let H be a spanning subgraph of G, and let $e = (x, y)$ be an edge of G which is not labelled left or right. If e is not in H, then H cannot be r-malconnected.*

Proof. First note that at most one of x and y can be a vertex of degree greater than 2. This follows because all the vertices of degree at least 3 are joined by paths of length at least $r/2$, unless they are connected by a left or right edge. Assume, without loss of generality, that x has degree 2, i.e. x is an internal vertex on some path joining two vertices of degree at least 3. But then x can be disconnected from the rest of the graph simply by removing its other neighbour, so we have a cutset of radius 0. $\square$

Hence a candidate for an r-malconnected spanning subgraph must contain all of the edges of G, except possibly the edges labelled left and right. Note that the proof of this lemma relied on the assumption that $r \geq 4$.

Lemma 2.2 *Let H be a spanning subgraph of G. If there exists an x_i such that H contains both a left edge from x_i and a right edge from x_i, then H cannot be r-malconnected.*

Proof. Suppose H contains a left edge from x_i to $[x_i, C_j]$ and a right edge to

$[\bar{x}_i, C_k]$, and let T be the tree of depth r rooted at $v(x_i)$. Any path from $z_i^{j,k}$ (the middle vertex of the path from $[x_i, C_j]$ to $[\bar{x}_i, C_k]$) to $\mathcal{X}$ has to go through one of the vertices $[z_i^{j,k}, v(C_j)]$, $[z_i^{j,k}, v(C_k)]$, $[x_i, C_j]$, or $[\bar{x}_i, C_k]$. But all of these vertices are in T, so $z_i^{j,k}$ and $\mathcal{X}$ are in different components in $G - T$. $\square$

Thus the vertices $v(x_i)$ ensure that any candidate for an r-malconnected spanning subgraph will contain either only left edges or only right edges for each x_i. As we shall see, this will correspond to setting x_i to be true or false.

The next lemma shows that if H contains a left edge (respectively, right edge) from some x_i to its gadget, then we can include in H all the left edges (respectively, right edges).

Lemma 2.3 *Let H be an r-malconnected subgraph of G. If, for some i, H contains a left edge (respectively, right edge) from x_i to its gadget, then the graph H' obtained by adding all the remaining left edges (respectively, right edges) from x_i is still r-malconnected.*

Proof. It is sufficient to show that if H is an r-malconnected spanning subgraph containing a left edge from some x_i to some $[x_i, C_j]$, then adding another left edge from x_i preserves r-malconnectivity. So suppose that x_i is contained in clauses C_j and C_k, and that H contains the left edge $(x_i, [x_i, C_j])$ but not $(x_i, [x_i, C_k])$. Now adding an edge e can only turn vertices at distance at most $r - 1$ from either endpoint of e into the centres of cutsets of radius r. So, in order to prove the lemma, it will suffice to verify that none of the vertices at distance at most r from the vertex $[x_i, C_k]$ can become the centre of a cutset when we add the edge $(x_i, [x_i, C_k])$. Let v be a vertex at distance at most r from $[x_i, C_k]$, and T a subgraph of radius at most r centred at v. Observe that we cannot disrupt the paths from the right vertices $[\bar{x}_i, C_l]$ of the gadget to their corresponding C_l, because H cannot contain any right edges. We can't disrupt the path from $\mathcal{X}$ to $\mathcal{C}$ either, nor the paths from C_j or C_k to $\mathcal{C}$ via $v(C_j)$ or $v(C_k)$, respectively. Hence, any vertex that could possibly be separated from $\mathcal{C}$ by removing T, that is, any vertex at distance at most $2r$ from $[x_i, C_k]$, will still be able to reach $\mathcal{C}$ through either $\mathcal{X}$, $v(C_j)$, $v(C_k)$, or some right vertex of the gadget corresponding to x_i. $\square$

Observe that it must also be the case that for each x_i, there is at least one left edge or one right edge from x_i to its gadget. Otherwise we can disconnect x_i from the rest of the graph by removing its neighbour on the path to $v(x_i)$.

Now we can set up the correspondence between true-false assignments to the variables x_i and potentially r-malconnected spanning subgraphs H. Given an assignment, let H be the subgraph of G consisting of all the unlabelled edges of G, plus, for each i, all the left edges from x_i to its gadget if x_i is true and all the right edges if x_i is false. Conversely, given a spanning subgraph H that contains either all left edges or all right edges for each x_i, we obtain a true-false assignment by setting x_i = true if the left edges are present, and false otherwise.

Before stating and proving our main theorem, we record the following useful fact as a lemma.

Lemma 2.4 *If H is a spanning subgraph corresponding to an assignment, i.e. with either all left edges or all right edges present for each x_i, then no vertices other than the $v(x_i)$ and $v(C_k)$ can be centres of cutsets of radius r.*

Proof. It is certainly true that neither $\mathcal{X}, \mathcal{C}$, nor any of the C_k can be centres of cutsets of radius r, because these vertices only have long paths emanating from them. Also observe that only the vertices $v(x_i)$ are within distance r of both left or right vertices of a gadget and a middle vertex $[z_i^{j,k}, v(C_k)]$, and only the vertices $v(C_k)$ are within distance r of more than one gadget. Showing that all internal vertices, and also the vertices x_i, $[x_i, C_j]$, or $[\bar{x}_i, C_k]$ cannot be centres of cutsets of radius at most r is straightforward. The only non-trivial cases are the vertices $z_i^{j,k}$ and $[z_i^{j,k}, v(C_j)]$. Removing a cutset of radius at most r centred at $z_i^{j,k}$ doesn't affect $[x_i, C_j]$ or $[\bar{x}_i, C_k]$, but it does affect $v(C_j)$ and $v(C_k)$. However, both C_j and C_k are still connected to $\mathcal{X}$ and hence $\mathcal{C}$ through other gadgets. Similarly, removing a cutset of radius at most r centred at $[z_i^{j,k}, v(C_j)]$ may take out $z_i^{j,k}$, $v(x_i)$ and $v(C_j)$, but everything is still connected to $\mathcal{C}$ either through the right side of the gadget or C_j and one of the other gadgets it is connected to. □

The NP-hardness of finding a r-malconnected spanning subgraph H is now established by the following theorem.

Theorem 2.5 *There exists a satisfying true-false assignment if and only if G contains a r-malconnected spanning subgraph H.*

Proof. Suppose that the formula is not satisfiable. So given any true-false assignment, there is at least one clause that is not satisfied. Without loss of generality, let this clause be $C_k = (x_{k_1} \vee x_{k_2} \vee x_{k_3})$. We claim that the tree of depth r centred at $v(C_k)$ is a cutset in the graph H corresponding to the assignment. Since C_k is not satisfied, we have that $x_{k_1} = \text{false}$, $x_{k_2} = \text{false}$, $x_{k_3} = \text{false}$. So how could we possibly get from the vertex C_k to $\mathcal{X}$? Well, not through $v(C_k)$, because that vertex lies in T. We can't go through the vertices x_{k_1}, x_{k_2}, or x_{k_3} either, because these are not joined to $[x_{k_1}, C_k], [x_{k_2}, C_k]$, or $[x_{k_3}, C_k]$. Finally, the middle vertices of the paths from $[x_{k_1}, C_k], [x_{k_2}, C_k]$, and $[x_{k_3}, C_k]$ to the other side of their respective gadget are contained in T as well. Hence C_k is disconnected from $\mathcal{X}$ in $H - T$.

Conversely, if there exists a satisfying assignment, then we claim that the corresponding graph H is r-malconnected. None of the vertices $v(x_i)$ can be the centre of a cutset of radius r, because for each x_i, the graph H either contains all left or all right edges from x_i to its gadget. This means that if all the left edges are chosen for some x_i, then the long paths from the right vertices $[\bar{x}_i, C_k]$ to C_k cannot be interrupted, and similarly if the right edges are chosen. Moreover, none of the vertices $v(C_k)$ can be the centre of a cutset of radius r. This follows from the fact that we have a satisfying assignment. Again without loss of generality, let $C_k = (x_{k_1} \vee x_{k_2} \vee x_{k_3})$ be a clause, and suppose we remove the tree of depth r centred at $v(C_k)$. This will disrupt the long paths from $[x_{k_1}, C_k], [x_{k_2}, C_k]$, and $[x_{k_3}, C_k]$ to all the vertices on the other side of the corresponding gadget. However, we must have at least one of the three edges $(x_{k_1}, [x_{k_1}, C_k]), (x_{k_2}, [x_{k_2}, C_k])$ and $(x_{k_3}, [x_{k_3}, C_k])$ and thus we maintain connectivity. By Lemma 2.4, none of the other vertices can be centres of cutsets of radius r, so the result follows. $\square$

Thus, we obtain our hardness result.

Theorem 2.6 *The problem of deciding whether a graph G contains a r-malconnected spanning subgraph H is NP-hard for $r \geq 4$.*

3 A Recognition Algorithm

Whilst it is hard to decide whether or not a graph contains a r-malconnected subgraph, it is easy to test whether a graph itself is r-malconnected. Here is a very simple polytime algorithm which achieves this.

Theorem 3.1 *There is an $O(n^3m)$ time algorithm to test whether a graph is r-malconnected.*

Proof. Suppose that G contains a cutset T of radius r centred at a vertex v. As $G - T$ contains at least two components there are two non-adjacent vertices x and y in different components of $G - T$. In particular, if we grow a maximum tree T' of depth r in $G - \{x, y\}$ centred at v, then T' is also a cutset separating x and y.

Thus, it suffices to proceed as follows: For every triple of vertices $\{x, y, v\}$, grow a maximum sized breadth first search tree T of depth r centred at v in $G - \{x, y\}$. If x and y are in different components of $G - T$ then G is not r-malconnected. If no such tree exists for any triple then G is r-malconnected. The algorithm is linear time for each triple; hence the overall running time is $O(n^3m)$. $\square$

We remark that this implies that the decision problem of whether G contains an r-malconnected subgraph H (of, say, a specific size) is in NP.

4 A Bi-Criteria Result

As we have seen, it is unlikely that we can obtain approximation algorithms for the minimum r-malconnected subgraph problem. This suggests that some form of bi-criteria approximation algorithm may be the best we can hope for. In this section, we show that such bi-criteria results are possible. Specifically, assuming that G contains no tree-cutset of depth at most three, we give a polytime algorithm to find an approximately minimum sized subgraph containing no star-cutsets.

Theorem 4.1 *Let G be a graph that does not contain a tree-cutset of depth 3. Then there is a factor $\frac{11}{6}$-approximation algorithm for the minimum 1-malconnected subgraph problem in G.*

In order to prove Theorem 4.1, our approach is to iteratively add vertices and edges to a subgraph H of G maintaining the property that H is 1-malconnected. We begin with a cycle $H = H_0$ which is locally maximum in the sense defined below. The first step in the algorithm is to add ears to H. We grow the ears carefully to maintain 1-malconnectivity. (We remark that ear decompositions have been used in designing efficient algorithms for network design problems with low connectivity requirements, for example see [1] and [7].) The first phase will terminate when $V - V(H)$ is a stable set S. In the second phase we will iteratively add the vertices of S along with specially chosen edges to H to avoid creating new star-cutsets.

Phase I: Ear-Growing

Let H_i be the current 1-malconnected subgraph. An ear E_i, whose endpoints, u_i and v_i, are in H_i and whose internal vertices are in $V - H_i$, is then added to H_i to give the subgraph H_{i+1}. The actual choice of ear E_i is made according to the following rules.

(1) The priority of an ear is determined according to the distance, $d_{H_i}(u_i, v_i)$, of u_i and v_i in H_i. Specifically, the order of desirable distances is $\{1, 3, 4, 5, \ldots, 2, 0\}$.

(2) Given Rule (1), the ear E_i is chosen to be "locally maximum" in terms of the number of vertices it contains in $V - H_i$.

(3) Phase I terminates when $V - H_i$ is a stable set.

First we should make it clear what we mean by "locally maximum". Suppose we have an ear E_i chosen according to Rule (1). Take an edge (u, v) of E_i and look for a path P from u to v all of whose internal vertices lie in $V - H_i - E_i$. If there is such a path, set E_i to be $E_i + P - (u, v)$ and repeat. Next, take two adjacent edges $e = (u, v)$ and $f = (v, w)$ in E_i, and look for a path P of length at least 3 from u to w all of whose internal vertices are in $V - H_i - E_i$. If there is one, set E_i to be $E_i + P - (e + f)$ and repeat. This can clearly be done in polynomial time, as the size of ear E_i grows with each local improvement.

Now let's show that Phase I does terminate in the manner required. This is guaranteed by the next lemma.

Lemma 4.2 *If $G - H_i$ contains an edge $e = (x, y)$ then we can always find an ear E_i with at least three edges.*

Proof. Since G contains no cut-vertices (otherwise G is not 3-malconnected), we can find two vertex disjoint paths from $\{x, y\}$ to H_i. Together with e these form an ear E_i with at least three edges.

Our goal now is to show that H_i remains 1-malconnected throughout the course of this first phase. To show this we will need the following claims.

Claim 4.3 *The chosen ear E_i will always satisfy $d_{H_i}(u_i, v_i) \geq 3$.*

Proof. Suppose there is an ear E_i such that $d_{H_i}(u_i, v_i) = 1$. Then $e = (u_i, v_i)$ was chosen in some ear E_j where $j < i$. But then $E_j - e + E_i$ contradicts the choice of E_j by Rule (2).

Suppose then that all possible choices for ear E_i satisfy $d_{H_i}(u_i, v_i) \in \{0, 2\}$. Take an ear E_i with $d_{H_i}(u_i, v_i) = 2$, if there is one. So there is a vertex $v \in H_i$ that is adjacent to both u_i and v_i. Let $e_1 = (v, u_i)$ and $e_2 = (v, v_i)$. If both e_1 and e_2 are in the same ear E_j where $j < i$ then $E_j - e_1 - e_2 + E_i$ could have been chosen as a larger ear instead of E_j, contradicting Rule (2). This follows as E_i contains at least two internal vertices, otherwise there is a cut-vertex in G. Thus, $e_1 \in E_k$ and $e_2 \in E_j$ where $k < j < i$. It follows that $v = v_j$ is an endpoint of the ear E_j. Thus, v_i is an internal vertex of E_j and so $v_i \notin H_j$. Consequently, $v_i \notin H_k$ and $E_k - (u_i, v) + (v, v_i) + E_i$ contradicts the choice of E_k.

Finally, suppose that all possible choices for ear E_i satisfy $d_{H_i}(u_i, v_i) = 0$. Then clearly G contains a cut-vertex. $\square$

Claim 4.4 *If H_i is 1-malconnected and $d_{H_i}(u_i, v_i) \geq 3$, then H_{i+1} is 1-malconnected.*

Proof. So H_i contains no star-cutset. Now consider H_{i+1}. Let v be an internal vertex in E_i. First, it is not possible for v to be the centre of a star-cutset in H_{i+1}. To see this, observe that v is at least distance two from one of u_i or v_i, otherwise the phase would have already terminated. There are no cut-vertices in H_i. Consequently, a star S_v centred at v (and, thus, containing at most two edges) cannot be a cutset in H_{i+1} as it removes at most one vertex from H_i, and $E_i - S_v$ remains connected to the rest of H_{i+1}.

So let v be a vertex in H_i. As v is not the centre of a star-cutset in H_i, it can only become the centre of a star-cutset in H_{i+1} if it separates an internal vertex of E_i from the rest of H_{i+1}. To do this S_v must contain both u_i and v_i. But then $d_{H_i}(u_i, v_i) \leq 2$, a contradiction. $\square$

Corollary 4.5 *For each i, the subgraph H_i is* 1*-malconnected.*

We remark that in Phase I we use only the fact that G contains no cut-vertices, which follows from the assumption that G is 3-malconnected.

Phase II: Incorporating the Stable Set

So at the end of Phase I we have a 1-malconnected subgraph H and a stable set $S = V - V(H)$. In the second phase we iteratively add the vertices of S to H, again, maintaining the property that the resulting subgraph remains 1-malconnected. Moreover, we can add a vertex to H so that it has two neighbours whose distance in H is at least 4.

Lemma 4.6 *Either G is Hamiltonian or, for any vertex $v \in S$, there are vertices $a, b \in \Gamma(\{v\}) \cap H$ such that $d_H(a, b) \geq 4$.*

Proof. Since S is a stable set we know that all of the neighbours of v are in H. As G has no cut-vertices we know that v has at least two neighbours in H. Take a neighbour u of v and suppose $d_H(u, w) \leq 3$ for all $w \neq u \in \Gamma(\{v\})$. So there is a tree $\mathcal{T}$ of depth 3 in H, centred at u, containing every vertex of $\Gamma(\{v\})$. We have two possibilities:
(i) If there is a vertex $x \neq v$ that is not in $\mathcal{T}$ then we have a tree of depth 3 whose removal from G separates x and v, a contradiction.
(ii) Otherwise $\mathcal{T}$ contains every vertex in $V - \{v\}$; in particular, note that this implies that $S = \{v\}$. Let $D_i = \{w : d_H(u, w) = i\}$, for $i = 1, 2, 3$. Observe that D_3 is a clique. Otherwise, let x, y be non-adjacent vertices in D_3. Since v is a neighbour of u there is a tree of depth 3 centred at u containing every vertex except x and y, that is, a tree-cutset. Moreover, $D_3 \subseteq \Gamma(\{v\})$. If not, let $x \in D_3 - \Gamma(\{v\})$. Then there is a tree-cutset of depth 3 centred at u containing

every vertex except x and v.

Next note that every vertex in D_2 has distance at most 3 from v. Since every vertex in D_3 has distance one to v, it follows by similar arguments that D_2 must also be a clique. In this case there is a path of length at most three from v to any vertex in D_2 that just uses vertices in $D_2 \cup D_3$. Consequently D_1 must also be a clique. It is then easy to find a Hamiltonian cycle in G. $\square$

So when we consider $v \in S$ we either output a Hamiltonian cycle in polynomial time or we simply connect v to two neighbours a and b whose distance is at least four in H. In the latter situation we have two possibilities: either $\{a, b\}$ does not form a cutset in H or $\{a, b\}$ forms a cutset in H. In the first case things are simple. We just set $H' = H + (a, v) + (v, b)$. Clearly H' has no star-cutsets so we may then repeat on H' and $S' = S - \{v\}$.

Consider the second case. Let the components of $H - \{a, b\}$ be $C_1, C_2, \ldots C_k$. Let G_i denote the subgraph of $H' = H+(a, v)+(v, b)$ restricted to $C_i \cup \{a, v, b\}$. Label by C_i^r the vertices of C_i whose shortest path distance to $\{a, b\}$ is exactly r, and label by C_i^{r+} the vertices of C_i whose shortest path distance to $\{a, b\}$ is at least r.

Claim 4.7 C_i^{2+} *is non-empty for all* $1 \le i \le k$.

Proof. Suppose C_i^{2+} is empty for some $1 \le i \le k$, then every vertex in C_i is adjacent to either a or b. Consequently, as C_i is connected, there is a path of length 3 from a to b through C_i. This is a contradiction by Lemma 4.6. $\square$

We will try to augment H' to make it 1-malconnected. To do so, we will consider the following Steiner tree problem. We have an auxiliary graph $\mathcal{G}$. There is a terminal node in $\mathcal{G}$ for each set of vertices C_i^{2+}; there is a Steiner node $\mathcal{G}$ for each vertex in $S - \{v\}$. Two nodes in $\mathcal{G}$ are adjacent if there is at least one edge between their corresponding vertex sets in G.

Claim 4.8 *There is a Steiner tree T in $\mathcal{G}$.*

Proof. The set of vertices $\{v, a, b\} \cup \bigcup_{i=1}^{k} C_i^1$ is spanned by a tree of depth 2 centred at v. As G is 3-malconnected, this tree of depth 2 is not a cutset, which implies the claim. $\square$

Lemma 4.9 *If T is a Steiner tree in $\mathcal{G}$ then $H' \cup T$ is 1-malconnected.*

Proof. Clearly v is not the centre of a star-cutset in $H'' = H' \cup T$ (here T also refers to an edge set in G that corresponds to the Steiner tree T in $\mathcal{G}$). Observe then that a star-cutset in H'' must be centred at a vertex incident to an edge in T. To show that such a cutset cannot exist we will need the fact that G_i is 2-vertex connected. Now take a Steiner node s in T. Note that s is incident to at most one vertex in any G_i. Moreover, such a vertex is in C_i^{2+} and thus is not a or b. So the removal of a star centred at s cannot disconnect G_i and all the G_is remain connected to one another via $\{v, a, b\}$.

Similarly take a vertex t in a terminal node, say C_1^{2+}. Observe that t does not centre a star-cutset in G_1, otherwise it gave a star-cutset in H. Now t is adjacent to at most one vertex in $G_i, i \geq 2$. Hence, as before the removal of the neighbours of t cannot disconnect any of the other G_i and all the G_is remain connected to one another via $\{v, a, b\}$. $\square$

So we add the Steiner tree to H' and then repeat the process with a new vertex in S. Clearly both Phase I and Phase II can be implemented in polynomial time so it remains to analyse the performance guarantee of the algorithm.

Proof of Theorem 4.1. In the ear decomposition phase, we add at most two edges for each vertex we add to H. In the stable set phase we charge the edges (a, v) and (b, v) to $v \in S$. We charge two edges of T to each Steiner node in T. So in the worst case we have no Steiner nodes in T and then we have $k-1$ edges left to charge for the k terminal nodes in T. How do we do this?

First, label the indices of $C_1, C_2, \ldots, C_k$ (the components of $H - \{a, b\}$) according to the order in which the algorithm first incorporated a vertex of C_i into H.

Claim 4.10 *For $i \geq 3$, the vertices a and b are the endpoints of an ear, say E_i, added in Phase I through C_i. Also, a and b either both lie on the initial cycle, or on an ear E_2 through C_2.*

Proof. Let C^* be the cycle found at the very beginning of Phase I. If some vertex $w \in C_2$ lies on C^*, then both a and b must lie on C^* as well, otherwise $\{a, b\}$ cannot separate C_1 from C_2. If no vertex of C_2 lies on C^*, let w_2 be the first vertex of C_2 added to H. Thus w_2 must be an internal vertex of some ear E_2, and E_2 must contain a and b (not necessarily as endpoints), otherwise $\{a, b\}$ cannot separate C_1 from C_2. It follows that when the algorithm adds the first vertex w_i of C_i, for $i \geq 3$, the current subgraph H_{k_i} already contains the vertices a and b. This implies that a and b must be the endvertices of the ear containing w_i, because otherwise removing $\{a, b\}$ would not separate C_i from all the other components. $\square$

Moreover, it can't be the case that C_i, $i \geq 1$, only contains vertices that were in S at the end of Phase I, because since S is a stable set, we would get the contradiction $d_H(a, b) = 2$.

Now we charge one edge of T to each such ear E_i, $i \geq 3$. Depending on whether the first vertex of C_2 lies on the initial cycle C^* or an ear E_2, we charge the remaining edge of T to C^* or E_2.

Claim 4.11 *The initial cycle C^* is charged at most $\frac{|C^*|-6}{2}$ edges, and each ear E is charged at most $\frac{n_E-1}{2}$ edges, where n_E is the number of internal vertices of E.*

The proof relies on the the following two lemmas. Given a cycle C, we say that a pair of vertices (u,v) *overlaps* a pair (x,y) if either $x \in P_1$, $y \in P_2$ or $x \in P_2$, $y \in P_1$, where P_1, P_2 are the two components of $C - \{u,v\}$. (Observe that (u,v) overlaps (x,y) if and only if (x,y) overlaps (u,v).) Given a path P, we say that a pair of vertices (u,v) *overlaps* a pair (x,y) if either $x \in P_1$, $y \in P_2 \cup P_3$ or $y \in P_1$, $x \in P_2 \cup P_3$, where P_1 is the interior of the subpath from u to v and P_2, P_3 are the two components of $P - (P_1 \cup \{u,v\})$. (Again, (u,v) overlaps (x,y) if and only if (x,y) overlaps (u,v).) We say that a family $\mathcal{F}$ of pairs of vertices is *non-overlapping* if

(i) No two pairs in $\mathcal{F}$ overlap each other.

(ii) $d(u,v) \geq 4 \; \forall (u,v) \in \mathcal{F}$.

(iii) $\min[d(u,x)+d(v,y), d(u,y)+d(v,x)] \geq 2 \; \forall (u,v),(x,y) \in \mathcal{F}$.

Figure 4(a) shows a path with a non-overlapping family (the vertices in a pair are joined by a dotted line), and in Figure 4(b), the pairs (u,v) and (u,w) fail to satisfy condition (iii) and (s,t) overlaps (i,j).

Lemma 4.12 *The size of a non-overlapping family $\mathcal{F}$ is at most* $\max[\frac{n_P-1}{2}, 0]$ *for a path P and* $\max[\frac{|C|-6}{2}, 0]$ *for a cycle C, where n_P is the number of internal vertices of P.*

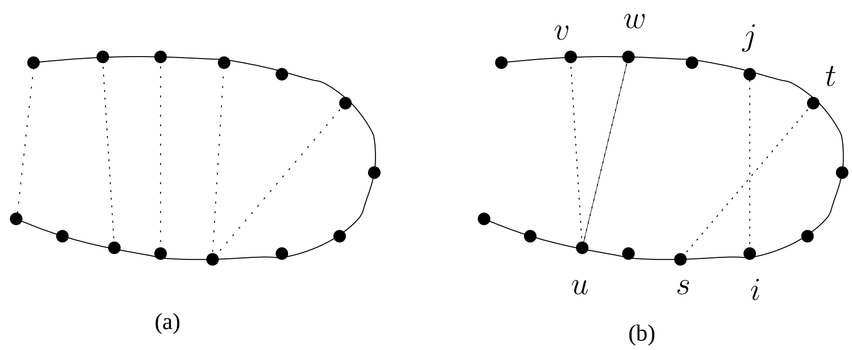

Fig. 3.

Proof. Let P be a path, from a vertex s to a vertex t, say, and let $\mathcal{F}$ be a non-overlapping family of maximum size for this path. We use induction on n_P. If $n_P \leq 2$, the statement of the lemma is true, because no two vertices of P are at distance at least 4. So suppose $n_P > 2$. If neither s nor t belongs to a pair other than possibly (s,t) in $\mathcal{F}$, we're done by considering $P - \{s,t\}$ and applying induction to find that the size of $\mathcal{F}$ is at most $\frac{n_P-2-1}{2}+1 = \frac{n_P-1}{2}$. So assume without loss of generality that $(s,u) \in \mathcal{F}$ for some $u \neq t$. Choose $u \neq t$ so as to maximize $d(s,u)$, and consider the $s-u$ path P_1 and the $u-t$ path P_2. If $(s,t) \in \mathcal{F}$, it is the only pair not contained in either P_1 or P_2. If $(s,t) \notin \mathcal{F}$, every pair in $\mathcal{F}$ is contained in either P_1 or P_2, since $\mathcal{F}$ is a non-overlapping family. Now $(s,t) \in \mathcal{F}$ implies $n_{P_2} \geq 1$, so by induction we have that the size of $\mathcal{F}$ is at most $\frac{n_{P_1}-1+n_{P_2}-1}{2}+1 = \frac{n_P-3}{2}$ as required. If $(s,t) \notin \mathcal{F}$ and $n_{P_2} \geq 1$, the same argument works. Finally, if $(s,t) \notin \mathcal{F}$ and $n_{P_2} = 0$, then P_2 cannot contain a pair of $\mathcal{F}$ and $P_1 = P - \{t\}$, so we can apply induction to P_1.

Now let C be a cycle, and $\mathcal{F}$ a non-overlapping family of maximum size for C. If $\mathcal{F}$ is empty, the assertion of the lemma is true. Otherwise, let (u,v) be a pair in $\mathcal{F}$ that minimizes $d(u,v)$. This splits C into two $u-v$ paths P_1 and P_2. By minimality of $d(u,v)$ and the fact that $\mathcal{F}$ is a non-overlapping family, either none of the internal vertices of P_1 occur in pairs belonging to $\mathcal{F}$, or none of the internal vertices of P_2 occur in pairs belonging to $\mathcal{F}$. Suppose without loss of generality that this is true for P_1. Since $d(u, v) \geq 4$, there must be at least 3 such internal vertices. Using the assertion of the lemma for paths proved above, we get that the size of $\mathcal{F}$ is at most $\frac{n_{P_2}-1}{2} \leq \frac{|C|-6}{2}$, as required. $\square$

Lemma 4.13 *The number of edges we charge to a given ear or the initial cycle in Phase II is at most the size of a maximal non-overlapping family for that ear or the cycle.*

Proof. We only charge an edge to the initial cycle C^* if C^* runs through components C_1 and C_2 with respect to the cutset $\{a, b\}$ we picked. Suppose that at a later iteration of Phase II, we pick vertices a' and b' on C^*. If the pair (a', b') overlaps the pair (a, b), the cycle does not run through two components with respect to the cutset $\{a', b'\}$, because when we considered the cutset $\{a, b\}$, we added a vertex $v \in S$ and edges (v, a) and (v, b). So C^* is not charged an edge for $\{a', b'\}$. Since the distance between a' and b' in the current subgraph H is at least 4, but a and b are now at distance 2 from each other, we see that the two conditions on distances in the definition of a non-overlapping family are satisfied. So the pairs of vertices we pick during Phase II that result in C^* being charged an edge form a non-overlapping family.

Similarly for an ear E. We charge an edge to E if a and b are the endpoints of E and E runs through some component C_i with $i \geq 3$. This can only occur once, since after the iteration where we consider $\{a, b\}$, a and b are joined by a path of length 2 via some vertex v from the stable set S. We also charge an edge to E if E runs through components C_1 and C_2 with respect to $\{a, b\}$. Suppose that in a later iteration we pick a cutset $\{a', b'\}$. If (a', b') overlaps (a, b), then E does not run through components C_1 and C_2 with respect to

$\{a', b'\}$ and therefore is not charged anything. We also know that a' and b' are at distance at least 4 from each other in the subgraph we have in the later iteration. These two facts imply that the pairs of vertices that result in E being charged form a non-overlapping family for E. $\square$

Let n_E denote the number of internal vertices of an ear E. So E consists of $n_E + 1$ edges. By Claim 4.11, the ear E is charged at most $\frac{n_E-1}{2}$ edges during Phase II. So for an ear E with at least 3 internal vertices, the sum of the edges belonging to E and the edges charged to E is at most $n_E+1+\frac{n_E-1}{2} = \frac{3}{2}n_E+\frac{1}{2}$. If E has only 2 internal vertices it is not charged anything in Phase II, and has $n_E + 1 = 3 = \frac{3}{2}n_E$ edges. The initial cycle C^* contains $|C^*|$ edges and is charged at most $\frac{|C^*|-6}{2}$ edges in Phase II. So together with the edges charged to the vertices of the stable set S, we have a total of

$$|C^*| + \frac{|C^*| - 6}{2} + \sum \frac{3}{2}n_E + \sum \frac{1}{2} + 2|S|$$

edges, where the first sum is taken over all ears added in Phase I, and the second sum is taken over all ears with at least 3 internal vertices. This is at most

$$\frac{3}{2}(n - |S|) + \frac{1}{6}(n - |S|) + 2|S|,$$

since the number of ears with at least 3 internal vertices is at most $\frac{1}{3}(n-|S|)$. Thus the total number of edges is at most $\frac{5}{3}(n - |S|) + 2|S|$.

Next, observe that $\max[n, 2|S|]$ is a lower bound on the optimal solution as each vertex must be adjacent to at least two edges. Since

$$\frac{\frac{5}{3}(n - |S|) + 2|S|}{\max[n, 2|S|]} \leq \frac{11}{6},$$

this gives an approximation guarantee of $\frac{11}{6}$ as required. $\square$

5 Conclusion

We have considered a new approach for modelling malicious attacks on a network, and have presented a hardness result and a bi-criteria approximation algorithm for the resulting network design problem. However, many interesting open problems remain.

Firstly, can the results in this paper be improved? We have seen that it is NP-hard to test whether a graph contains a r-malconnected subgraph, for $r \geq 4$. What about the special case of $r = 1$; that is, is it NP-hard to decide whether or not a graph contains a subgraph with no star-cutsets? Can the approximation guarantee for our bi-criteria result be improved?

Secondly, can our results be extended to more complex situations? What happens if edge costs are incorporated? In this situation, we are looking for minimum cost rather than minimum cardinality subgraphs. What happens if we allow more than one malicious attack? Here, we need to maintain graph connectivity when *numerous* subtrees are removed from the graph.

Finally, this work focuses on wired networks. What if we consider wireless networks? One way to do this would be to incorporate power costs into the model (for example, see [4] and [9]). Do corresponding results hold for such wireless networks?

References

[1] J, Cheriyan, A. Sebo and Z. Szigeti, "Improving on the 1.5-approximation of a smallest 2-edge connected spanning subgraph", *SIAM J. Discrete Math*, **14**, pp170-180, 2001.

[2] J. Cheriyan, S. Vempala and A. Vetta, "Approximation algorithms for minimum-cost k-vertex connected subgraphs", *SIAM Journal on Computing*, **32(4)**, pp1050-1055, 2003.

[3] J. Cheriyan and A. Vetta, "Approximation algorithms for network design with metric costs", *Proceedings of the 37th ACM Symposium on Theory of Computing*, pp167-175, 2005.

[4] A. Clementi, P. Penna and R. Silvestri, "Hardness results for the power range assignment problem in packet radio networks", *Proceedings of the 2rd International Workshop on Approximation Algorithms for Combinatorial Optimization*, pp197-208, 1999.

[5] A. Frank, "Connectivity augmentation problems in network design", *Mathematical Programming: State of the Art*, pp34-63, 1995.

[6] A. Frank and T. Jordan, "Minimal edge-coverings of pairs of sets", *Journal of Combinatorial Theory B*, **65**, pp73-110, 1995.

[7] H. Gabow, "An ear decomposition approach to approximating the smallest 3-edge connected spanning subgraph of a multigraph", *SIAM Journal of Discrete Mathematics*, **18(1)**, pp41-70, 2004.

[8] H. Gabow, M. Goemans, E. Tardos and D. Williamson, "Approximating the smallest k-edge connected spanning subgraph by LP-rounding", *Proceedings of the 16th Symposium on Discrete Algorithms*, pp562-571, 2005.

[9] M. Hajiaghayi, G. Kortsarz, V. Mirrokni and Z. Nutov, "Power optimization for connectivity problems", *Proceedings of the 11th Conference on Integer Programming and Combinatorial Optimization*, pp349-361, 2005.

[10] K. Jain, "A factor 2 approximation algorithm for the generalized Steiner network problem", *Combinatorica*, **21(1)**, pp39-60, 2001.

[11] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network", *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp137-146, 2003.

[12] S. Khuller, "Approximation algorithms for finding highly connected subgraphs", in D. Hochbaum (ed.), *Approximation Algorithms for NP-hard Problems*, pp236-265, 1995.

[13] S. Khuller and U. Vishkin, "Biconnectivity approximations and graph carvings", *Journal of the ACM*, **41(2)**, pp214-235, 1994.

[14] L. Kirousis, E. Kranakis, D. Krizanc and A. Pelc, "Power consumption in packet radio networks", *Theoretical Computer Science*, **243**, pp289-305, 2000.

[15] G. Kortsarz, R. Krauthgamer and J. Lee, "Hardness of approximation algorithm for vertex-connectivity network design problems", *SIAM Journal on Computing*, **33(3)**, pp704-720, 2004.

[16] G. Kortsarz and Z. Nutov, "Approximation algorithm for k-node connected subgraphs via critical graphs", *Proceedings of the 36th ACM Symposium on Theory of Computing*, pp138-145, 2004.

[17] S. Vempala and A. Vetta, "Factor 4/3 approximations for minimum two-connected subgraphs", *Proceedings of the 3rd International Workshop on Approximation Algorithms for Combinatorial Optimization*, pp262-273, 2000.