Galaxy cutsets and graph connectivity: variations on a theme

by

Nicolas Sonnerat

Department of Mathematics and Statistics
Faculty of Science
McGill University, Montréal, Québec

June 2010

A thesis submitted to McGill University
in partial fulfillment of the requirements
of the degree of Doctor of Philosophy

Abstract

In this thesis we consider cutsets in graphs which can be expressed as unions of sets each of which is spanned by a tree of diameter at most $d-1$ for some integer $d \geq 1$; we call these sets *galaxy cutsets*. These galaxy cutsets generalise both star-cutsets and vertex-cuts, and serve as simple models for virus-like attacks on or cascading failures in networks, the crucial property being that neighbours of affected vertices may also fail and cease to function. We approach our subject from four different points of view. We begin by exploring the connection between galaxies and a suitable type of flow, proving a min-max result for planar graphs. Then, after tackling the fundamental issue of recognising whether a given graph is susceptible to virus-like attacks, i.e. whether it contains a galaxy cutset, we consider a weighted version of the flows that are dual to the galaxies, and prove $\Theta(\log n)$ lower and upper approximability bounds for the problem of finding a maximum such flow. We then investigate the problem of network design, that is to say, the problem of constructing low cost spanning subgraphs of a given graph which are not vulnerable to cascading failures. Finally, we embark on a detailed analysis of the structure of star-cutsets in planar graphs and use our results to derive a polynomial time algorithm for the problem of neutralising every star-cutset by protecting edges.

Abrégé

Dans cette thèse, nous considerons des séparateurs dans les graphes qui peuvent être éxprimés sous forme d'une union d'ensembles de sommets dans laquelle chaque ensemble est couvert par un arbre de diamètre $d-1$ pour un nombre entier $d \geq 1$; nous appellons ces séparateurs des *galaxies séparatrices*. Les galaxies séparatrices genéralisent les étoiles séparatrices et les séparateurs formés par un ensemble de sommets, et elles servent comme simple modèle pour des attaques de virus sur ou des cascades de défaillances dans un réseau, la proprieté distinguante étant que les voisins des sommets qui sont affectés peuvent eux aussi faillir. Nous approchons le sujet depuis quatre points de vue différents. Nous commençons par explorer le lien entre les galaxies et un type de flot approprié, et nous prouvons un résultat de type minmax pour les graphes planaires. Ensuite, après avoir résolu la question fondamentale de reconnaitre si un graphe donné est sensible aux attaques de virus, c'est-à-dire s'il contient une galaxie séparatrice, nous introduisons des capacités dans les flots correspondants aux galaxies, et demontrons une borne d'approximabilité inférieure et supérieure de $\Theta(\log n)$ pour le problème de trouver un flot maximum. Ensuite, nous enquêtons sur le problème de dessein de réseau, c'est-à-dire le problème de construire des sous-graphes couvrants peu coûteux qui ne sont pas sensibles aux cascades de défaillances. Finalement, nous nous lançons dans une analyse détaillée de la structure des étoiles séparatrices dans les graphes planaires, et nous utilisons nos résultats pour développer un algorithme polynomial qui résout le problème de neutraliser toutes les étoiles séparatrices en protégeant des arêtes.

Acknowledgments

First and foremost, I would like to thank my supervisors, Adrian Vetta and Bruce Reed. To them, I am grateful for many things: their continuous support, both academic and financial; the many things I learned from them during courses and research discussions; their patience; the invitations to various workshops; and the occasional work sessions that took place over a pint during the half-time break of a football match.

I would also like to acknowledge three professors with a great gift for teaching whose courses I was fortunate to take at McGill: David Avis, Luc Devroye, and Bruce Shepherd.

Lastly, I would like to thank all my friends in both the mathematics and computer science departments at McGill for making graduate school an enjoyable experience: Simon Gemmrich, Michael Wong, Geva Maimon, David Cottrell, Neil Olver, Joel Phillips, Andrew and Jamie King, Sean Kennedy, Nicolas Broutin, Ross Kang, Louigi Addario-Berry, and Conor Meagher.

Declaration

This thesis does not contain any material which has been accepted, in whole or in part, for any other degree or diploma. The results of Chapters 3, 4, 5, 6 and 7 constitute an original contribution to knowledge, unless explicitly stated otherwise.

Chapters 3, 4, 6 and 7 are based on joint work with my supervisor, Adrian Vetta, and have either appeared in print ([90]), or have been submitted to appear in print ([89], [88]), or will be submitted in the near future ([87], an extended abstract of this paper has appeared in [91]).

Chapter 5 is based on joint work with Guyslain Naves and Adrian Vetta, and has been submitted to appear in print ([78]).

Contents

List of Figures

Chapter 1

Introduction

In this thesis, we investigate a special type of cutsets in graphs, which we call *galaxy cutsets*. In its most general form, a galaxy cutset is a union of k sets of vertices, each of which is spanned by a tree of diameter at most $d-1$, where both k and d are positive integers. The parameter k is called the *order* of the galaxy. We will study both the case of galaxy cutsets that separate two fixed designated vertices s and t, and the case where the galaxy disconnects the graph, i.e. we only require that there exist some pair of vertices that is separated. Cuts in graphs are often intimately related to flows, and we shall see that galaxy cutsets are no exception to this pattern. In the case of galaxies separating two given vertices s and t, the dual flow corresponding to the galaxies will turn out to have a particularly nice combinatorial structure.

The motivation for studying galaxies is that they serve as models for virus-like spreading through or cascading failures in networks: The infection begins at the centres of the trees spanning the sets in the galaxy, and then propagates a certain distance along the edges of the graph. While this sort of spreading behaviour can in principle be positive, as in the spread of information through a network, we shall adopt a pessimistic view and focus on malicious attacks and failures. Hence, we will assume that the vertices belonging to a galaxy are destroyed or cease to function.

The question then is what sort of properties of the graph that remains after we remove a galaxy we ought to study. In graphs modelling communication or transportation networks, a property of prime importance is always connectivity, i.e. after the occurrence of vertex or edge failures, the unaffected vertices still ought to be able to transfer information or goods between one another. We therefore choose connectivity as our criterion for the functionality of a network after disaster has struck in the form of a galaxy.

The degree of connectivity is generally quantified by the number of vertices or edges that may fail simultaneously without disconnecting the graph. However, these breakdowns are essentially independent of one another. Our galaxy cutsets, on the other hand, capture the idea that failures might propagate through the graph, in the sense that neighbours of affected vertices also break down and cease to function. Indeed, the fact that two vertices of a graph are adjacent usually expresses that there is some kind of special relationship between them, or that they are close together in some sense. It therefore stands to reason that a breakdown at a vertex would affect its neighbours. To make this more concrete, let us consider the following examples:

In a social network, the vertices represent individuals, while the edges indicate some kind of relation, e.g. friendship, kinship, or working at the same office. If a member of the network catches a virus, like the flu, this virus will spread through the network along the edges, starting at the initially infected vertex.

For another example with a similar flavour, consider a computer network, say internet servers and the physical wires connecting them. In this setting, a computer virus spreads in much the same way as a biological virus would, which is of course how computer viruses obtained their name.

However, viruses do not provide the only example. In an electricity network, a component failure often results in a cascade of secondary failures,

since electrical current cannot simply disappear and will thus begin to overload other components.

As a further example, consider a spy network, where the vertices are covert agents, joined by an edge if they directly communicate with one another. Here the infection-like spreading behaviour can model the situation of an agent being subverted by the enemy and turned into a double agent. This could conceivably affect the reliability of the information obtained not just from the subverted agent, but also from his or her neighbours in the network, i.e. the spies that were in direct contact with the double agent.

The central theme of this thesis is how failures or breakdowns which propagate through a graph affect the connectivity. Each chapter can be viewed as a variation on this central theme, where we approach the phenomenon from a different angle. In the remainder of this introduction, we define the core concepts, then outline the different approaches taken in each chapter and summarise our results. We then conclude the chapter by placing our results into the context of related work.

Given a graph $G = (V, E)$, a set of vertices X is called a *d-galaxy of order k* if it can be written as a union of k sets of vertices $Z_1 \cup \ldots \cup Z_k$, each of which is spanned by a tree of diameter at most $d - 1$[1]. A d-galaxy of order $k = 1$ will be called a *d-star*. Sometimes it will be convenient to use the depth of the trees spanning the stars instead of the diameter, and we therefore make the following additional definitions: If S is a vertex set spanned by a tree of depth r, we will refer to it as a *star of radius r*, and a set which is the union of k stars of radius r is a *galaxy of order k and radius r*. We will limit the confusion that might arise from the two similar definitions by emphasising in each chapter whether we are using diameters or radii.

[1]The reason why we use $d - 1$ instead of d will become clear when we investigate the dual flows corresponding to galaxies

1.1 Results

In Chapter 2, we prepare the ground by fixing notation and the definitions that are necessary to state and prove our results. Since many of our results pertain to computational complexity, we also list the standard NP-complete problems we will use for reductions.

In Chapter 3 we explore the connection between galaxy cutsets and a type of flow called a *d-noninterfering flow*. We can obtain an intuitive idea of how this flow should be defined as follows: The vertex version of Menger's theorem asserts that the minimum number of vertices one has to remove from a graph to disconnect given non-adjacent vertices s and t equals the maximum number of vertex-disjoint $s-t$ paths, i.e. the maximum cardinality of a collection of paths no two of which pass through a common vertex (apart from s and t of course). Since vertices can be thought of as cutsets of radius 0, one might be tempted to conjecture that the minimum number of stars (sets of radius 1) that have to be removed in order to disconnect s and t equals the maximum cardinality of a collection of $s-t$ paths with the property that no two paths intersect a common star. We will show that this conjecture almost holds in planar graphs, but that in graphs which are not planar the gap between the minimum and the maximum can be arbitrarily large. More precisely, given a graph with non-adjacent vertices s and t and an integer $d \geq 1$, we will say that two $s-t$ paths are *d-noninterfering* if they are pairwise at distance at least d in $G - \{s, t\}$. A collection of pairwise d-noninterfering paths will be called a *d-noninterfering flow*. The main result of the chapter is:

Theorem 3.1 In a planar graph, the minimum number of d-stars we must remove in order to separate s and t is at most one more than the maximum cardinality of a d-noninterfering flow between s and t, for any $d \geq 1$.

This packing/covering result can be viewed as an extension of a weak-

ened version of Menger's theorem. For non-planar graphs, we exhibit an example where the maximum number of 2-noninterfering paths is 1, while the minimum number of 2-stars we must remove is of order $\Omega(n)$. We also show that for fractional flows and galaxies, the minimum and the maximum are equal by the linear programming duality theorem, regardless of whether the graph is planar or not.

In Chapter 4, we tackle the fundamental problem of recognising whether a given graph is vulnerable to virus-like attacks in the form of galaxy cutsets. Our main result is:

Theorem 4.5 The problem of deciding whether there is a galaxy cutset of order k and radius 1 is NP-complete if the parameter k is part of the input.

However, we will show that if the parameter k is constant, then there is a straightforward recognition algorithm that runs in polynomial time for any r. For the special case $k = 1$, we exhibit a more sophisticated algorithm that runs in time $O(rnm)$, which significantly improves on the running time of the generic trivial algorithm ($O(mn^3)$).

In Chapter 5, we consider an extension to weighted flows of the d-noninterfering flows studied in Chapter 3. We begin by showing that for $d = 2$, the problem of finding a maximum 2-noninterfering flow is as hard as Stable Set in non-planar graphs (even in the unweighted case). We then prove the following inapproximability result:

Theorem 5.1 For undirected planar graphs, the hardness of approximation for the maximum 1- or 0-noninterfering weighted flow problem is $\Theta(\log n)$, unless $P = NP$.

For the case $d = 0$, we define a flow to be 0-noninterfering if the paths in it are pairwise edge-disjoint.

In Chapter 6, we turn to the problem of designing networks that are not vulnerable to cascading failures in the shape of galaxy cutsets. In graph

theoretic terms, designing a network traditionally means finding a spanning subgraph having certain desirable properties. The full graph specifies which edges could possibly be built, and their cost, and the spanning subgraph is the network that is eventually realised. The simplest example of this is to find a minimum weight spanning tree in a graph with weighted edges, i.e. a minimum weight 1-vertex connected spanning subgraph. This problem admits several elegant and efficient solutions; in fact, one could argue (somewhat tongue-in-cheek) that pretty much any reasonable approach leads to a provably optimal solution. However, even in an unweighted graph finding a minimum 2-connected spanning subgraph is NP-hard, since such a subgraph is a Hamiltonian cycle if the graph is Hamiltonian. Hence, an algorithm that constructs a minimum 2-connected spanning subgraph could be used to solve the NP-complete decision problem of deciding if a graph is Hamiltonian. Unfortunately, in our case the situation is even more difficult, even if we restrict ourselves to considering galaxy cutsets of order $k = 1$:

Theorem 6.6 It is NP-complete to determine if a given graph has a spanning subgraph without cutsets spanned by trees of depth at most r when $r \geq 4$.

As this effectively rules out approximation algorithms, we turn to bicriteria results, and show that if we make the stronger assumption that G does not contain a cutset of radius 3, we can find a spanning subgraph without star-cutsets with no more than $\frac{11}{6}$ times as many edges as the optimal such subgraph. The main difficulty in constructing graphs without star-cutsets of radius $r \geq 1$ is the lack of monotonicity: While adding edges increases the connectivity, it may also create new star-cutsets. Hence, one must take great care when adding edges, which is not the case in the design of k-connected graphs, where every additional edge can only increase the connectivity.

Chapter 7 consists of two main parts. The first part is of a purely graph theoretic nature, as we investigate in detail the structure of star-cutsets of

radius 1 in planar graphs. In the second part, we use the structural results obtained in the first part to solve a problem that belongs to the domain of optimisation on graphs. More concretely, the problem is to choose a minimum cardinality set of edges that intersects the set of rays of every star-cutset. (The *rays* of a star are the edges from the centre to the other vertices.) The problem thus belongs to the class of hitting set problems. and it is motivated by the virus-like attacks modelled by star-cutsets. By protecting certain edges we prevent the virus from propagating along them, so protecting a collection of edges intersecting every set of rays effectively neutralises all the star-cutsets in the graph. Our structural result and the algorithm only hold in planar graphs however; for general graphs, we present an $O(\log n)$ approximation algorithm, together with an approximation-preserving reduction from Set Cover that show this guarantee to be tight up to a constant factor. We also show that the similar problem of protecting vertices in order to neutralise star-cutsets is NP-complete, even in planar graphs.

In Chapter 8 we conclude by listing open problems related to galaxy cutsets and to weighted d-noninterfering flows.

1.2 Related work

While there exists an extensive body of literature dealing with the spread of epidemics or cascading failures in graphs, most of the existing results differ from ours in the following two important aspects. First, the underlying graphs considered are often random, for example the Erdös-Renyi graphs $G_{n,p}$ or $G_{n,m}$, or graphs that fit into the preferential attachment framework. Second, the measure of functionality of the graph after a failure tends to be either the size of the largest component, or the number of affected vertices.

However, while the literature on the spread of epidemics through graphs is not very closely related to our work for the most part, there are many

results in the areas of network flows, connectivity of graphs, network design, optimisation on networks, and structural graph theory which are relevant and which we shall briefly discuss in the remainder of this chapter.

The study of the d-noninterfering flows which are the subject of Chapter 3 was initiated by McDiarmid, Reed, Schrijver and Shepherd in [71] and [72], and our result builds upon their work. Given the applicability of network flows there is a vast literature optimising flows given additional constraints. These side-constraints may arise from the application itself, but they can also arise due to restrictions induced by available technology or by the choice of routing protocol; see [86] for a survey illustrating some of these issues. The work most closely related to the results of Chapter 5 concerns k-splittable flows introduced by Baier, Köhler and Skutella [5]. A k-splittable flow is a flow that can be routed along k paths - note that these paths are not required to be disjoint. Thus, Kleinberg's unsplittable flows [62] can be viewed as 1-splittable flows. Baier et al. present a 2-approximation algorithm for the k-splittable single-commodity flow problem. For further examples of the connection between flows and cuts and of approximate maxflow-mincut theorems (for multi-commodity flows) the reader may also wish to consult Garg, Vazirani and Yannakakis [38].

The computational hardness of identifying galaxy cutsets is in stark contrast to the problem of deciding whether a graph is k-vertex or k-edge connected. Many of the algorithms to solve this problem use Ford and Fulkerson's maximum flow algorithm ([29]), or a refinement of it, in one way or another. Currently the fastest run-times for determining the vertex-connectivity κ or the edge-connectivity λ of an undirected graph are $O((n + \min\{\kappa^{\frac{5}{2}}, \kappa n^{\frac{3}{4}}\})\kappa n)$ and $O(m + \lambda^2 n \log(n/\lambda))$; both are due to Gabow ([35] respectively [33]). We refer the reader to Chapters 9 and 15 of Schrijver [84] for a more detailed description of methods and algorithms for determining the vertex- and edge-connectivity of graphs.

As far as network design is concerned, we already mentioned that finding a minimum weight spanning tree in a graph is a tractable problem, and that finding a minimum weight k-connected spanning subgraph is NP-hard when $k \geq 2$. Consequently, researchers have focused on approximation algorithms, and there is a wealth of results on the subject. We only list the best-known approximation ratios for unit costs and general costs in undirected graphs. Other cases that have been considered in the literature include metric costs and fixed small values of k for both directed and undirected graphs. The survey by Kortsarz and Nutov [66] contains results for the cases omitted here. For general non-negative costs, the algorithm with the currently best approximation guarantee for finding a k-edge connected spanning subgraph is due to Jain [55] and has an approximation factor of 2. For k-vertex connectivity, Cheriyan, Vempala and Vetta [14] gave a $O(\log k)$ algorithm for graphs having at least $6k^2$ vertices. Nutov [79] gave an $O(\log k \cdot \log \frac{n}{n-k})$ algorithm, with no restrictions on the input graph. This guarantee is of order $O(\log k)$ unless k is of order $n - o(n)$. If all the edge costs are equal to 1, the best-known bounds for undirected graphs are $1+\frac{2}{k}$ for edge-connectivity (Gabow, Goemans, Tardos and Williamson [36]) and $1 + \frac{1}{k}$ for vertex-connectivity (Cheriyan and Thurimella [13]).

Regarding graph theoretic results, the structure of star-cutsets (of radius 1) has played a major role in the theory of perfect graphs. Chvátal showed in [18] that no minimal imperfect graph contains a star-cutset, a result that became known as the Star-cutset Lemma. In the same article, Chvátal also gave a polynomial time algorithm for finding a star-cutset, the run-time being $O(nm)$. Kratsch and Spinrad improved on this run-time, and in [67] gave an $O(n^{2.79})$ algorithm that uses periodic matrix multiplication. Before Chudnovski, Robertson, Seymour and Thomas ([16]) proved the Strong Perfect Graph Conjecture, the Star-cutset Lemma was a powerful tool for determining whether certain classes of graphs were perfect. For

example, Hayward showed in his Ph.D. thesis [53] that every weakly triangulated graph contains a star-cutset, and deduced that weakly triangulated graphs constituted a class of perfect graphs. Also in [18], Chvátal defined a generalisation of star-cutsets called skew-partitions, and conjectured that a generalisation of his Star-Cutset Lemma held for skew-partitions. The validity of his conjecture is implied by the proof of the Strong Perfect Graph Conjecture ([16]). For a concise summary of the development from star-cutset to (balanced) skew-partitions and the part they played in the proof of the Strong Perfect Graph Conjecture, see Reed [81].

Cutsets which are unions of stars were considered by Gunther in [40], where he gave a characterisation of graphs that cannot be disconnected by removing $k-1$ stars. However, he made the additional assumption that the graph be k-regular and contain a k-clique, and his definition of a star slightly differs from ours. This work was later extended by Gunther, Hartnell and Nowakowski in [44], and by Gunther and Hartnell in [43]. In [42], Gunther and Hartnell investigate graphs such that the expected number of affected vertices after an attack consisting of k randomly chosen stars is minimised. Finbow and Hartnell ([26]) later generalised this to unions of sets of radius two. In the same spirit of studying graphs that are resilient to virus-like attacks, Hartnell and Kocay in [50] characterise graphs G such that $G-S$ is minimally k-connected for any star S in the graph, where k can take the values $1, 2$ or 3.

From an algorithmic rather than structural point of view, the idea of protecting vertices to guard against a threat that spreads through a graph is not new; a case in point is the Firefighter problem, proposed by Hartnell in [49]. Here, a fire rages through the graph, successively spreading to neighbours of vertices that are burning in discrete time steps. In each round, a fixed number of vertices may be protected by the firefighter, and the process ends when the fire can no longer spread to unprotected neighbours of burning vertices.

The objective is to minimise the number of vertices that are burning at the end of the process. Results on the firefighter problem include [27] and [59], see also the recent survey by Finbow and MacGillivray [28]. The firefighter problem differs from the problems we consider in two important aspects: First, we focus exclusively on connectivity, i.e. the number of infected vertices is of no concern to us, as long as the graph induced on the unaffected vertices remains connected. Second, we do not explicitly consider dynamic processes, meaning we cannot modify the graph once a virus-like failure has started spreading. In our model, the dynamic aspect is implicitly given by the parameter r (or d, as the case may be). Two possible interpretations are that after time r, a way of stopping the viral spread is found, or that the strength of the infection is attenuated the farther it travels from the initially infected vertices. The problem of containing epidemics or cascading failures in networks has also been considered by Holme [54] and Motter [74], among others. The former investigates the problem of which vertices to vaccinate in order to reduce the effect of an epidemic, and the latter approaches the problem by allowing the intentional removal of vertices or edges after the initial infection/failure. In either case, the measure of success is the size of the largest connected component of the graph remaining after the epidemic or cascading failure.

Chapter 2

Preliminaries

Any graph theoretic terms not explicitly defined here can be found in Bollobás [7].

2.1 Graphs

Unless specified otherwise, throughout this thesis a graph $G = (V, E)$ will be understood to be finite, simple, and undirected. We generally let $n := |V(G)|$ denote the number of vertices, and $m := |E(G)|$ the number of edges. The *neighbourhood* of a vertex u will be denoted by $\Gamma(u)$ and is defined to be $\Gamma(u) := \{v \in V(G) \mid (u, v) \in E(G)\}$. A *subgraph* H of G is a graph such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. Note that H is a graph in its own right, so the endpoints of an edge $e \in E(H)$ must both belong to $V(H)$. Given a set $S \subseteq V$, we denote by $G[S]$ the graph induced on the vertices S, i.e. $V(G[S]) = S$ and $E(G[S]) = \{(u, v) \in E(G) \mid u \in S \wedge v \in S\}$. A *minor* of a graph G is a graph G^* obtained from G by a series of vertex deletions and edge contractions. Contracting an edge $e = (u, v)$ means replacing u and v by a new vertex u^* adjacent to all the vertices in $(\Gamma(u) \cup \Gamma(v)) - \{u, v\}$.

A set of vertices S is called a *stable set* or *independent set* if no two

vertices of S are adjacent. A set of vertices K is called a *clique* if any two vertices of K are adjacent.

If T is a set of vertices containing a vertex v such that v is adjacent to all $x \in T, x \neq v$, we say that T is a *star* with *centre* v. We define the set of *rays* of a star T centred at v to be $R(T) := \{(v, x) \mid x \in T, x \neq v\}$. We point out that according to our definition, singleton vertices are considered to be stars (with no rays), and that a star need not contain all the neighbours of the centre.

2.2 Paths and connectivity

A *path* from a vertex u to a vertex v is a sequence of distinct vertices $\{u_1, u_2, \ldots, u_k\}$ such that $u_1 = u, u_k = v$ and for each $i < k$, the pair (u_i, u_{i+1}) is an edge. A closed path, i.e. a path such that (u_k, u_1) is an edge, will be referred to as a *cycle*. The *length* of a path is the number of edges it contains. If there exists a path from u to v, we define the *distance* $d(u, v)$ between u and v to be the length of a shortest path between u and v. If H is a subgraph of G and u and v are in $V(H)$, then $d_H(u, v)$ denotes the distance in H, i.e. the length of a shortest path that uses only vertices and edges belonging to H. The *diameter* of a subgraph H is defined to be the maximum distance $d_H(u, v)$ over all pairs of vertices u and v in H. Two paths P and Q are called *edge-disjoint* if $E(P) \cap E(Q) = \emptyset$, and *vertex-disjoint* if $V(P) \cap V(Q) = \emptyset$. Two paths P and Q with common endpoints u and v are said to be *internally vertex-disjoint* if $V(P) \cap V(Q) = \{u, v\}$.

Given an integer $k \geq 1$, we say that a graph G is *k-vertex connected* if, for all non-adjacent pairs of vertices $\{u, v\}$, there exist k internally vertex-disjoint paths from u to v. Similarly, we say that G is *k-edge connected* if for all pairs of vertices $\{u, v\}$, there exist k edge-disjoint paths from u to v. If a graph G is 1-vertex connected or, equivalently, 1-edge connected, we

simply say that G is *connected*, and if there exist vertices u and v such that there is no path from u to v, we say that G is *disconnected*. Consider the equivalence relation defined on the vertices of a graph by $u \sim v$ if and only if there exists a path from u to v. The equivalence classes of this relation are precisely the maximally connected subgraphs of G and are called the *connected components* of G. A set $S \subseteq V(G)$ is a *cutset* if $G[V - S]$ is disconnected. A cutset S is said to *separate* two vertices u and v if they belong to distinct components of $G[V - S]$.

A set of vertices X that can be written as a union of k sets, each of which is spanned by a tree T of diameter at most $d-1$ for an integer $d \geq 1$, is called a *d-galaxy of order k*. A d-galaxy of order 1 will also be called a *d-star*. Sometimes it will be more convenient to argue in terms of depth rather than diameter, and so we define a set X to be a *galaxy of order k and radius r* if X can be written as a union of k sets each of which is spanned by a tree of depth at most r, for some $r \geq 0$. A galaxy of order 1 and radius r will also be called a *star of radius r*. In order to avoid any confusion that could arise from the difference between d-galaxies and galaxies of radius r, we will always recall the definitions further on, when they are used.

Note that in order to be precise, we should really define d-galaxies (or galaxies of radius r) in terms of the smallest integer d (respectively r) such that all the stars in the galaxies are spanned by a tree of diameter $d-1$ (respectively of depth r). However, we can afford to be a little bit sloppy in this regard, because we will only ever be interested in galaxies spanned by trees of diameter at most, but not necessarily exactly $d-1$ (respectively of depth at most r). The reason we use $d-1$ instead of d in the definition of d-galaxies will become clear in Chapter 3.

Observe that stars of radius 0 correspond to vertices, and stars of radius 1 correspond to stars according to the standard definition. Also observe that for even values of d, a set S is a d-star if and only if it is a star of radius $\frac{d}{2}$.

We end this section by stating Menger's theorem, which relates the size of minimum cutsets to the size of maximum collections of vertex-disjoint paths.

Theorem 2.1 (Menger)**.** *Let G be a graph with non-adjacent vertices s and t. Then the maximum number of internally vertex-disjoint $s-t$ paths equals the minimum cardinality of a vertex set separating s and t.*

We remark that Menger first formulated his theorem in terms of topology ([73]); however, it is in the form above that it is usually encountered in the literature on graph theory.

2.3 Planarity

To draw a graph on a surface, for instance the plane $\mathbb{R}^2$ or the sphere S^2, one represents a vertex $u \in V(G)$ by a point x_u in the surface, and an edge $e = (u, v)$ by a continuous curve γ_e with endpoints x_u and x_v. For convenience, we will not distinguish between vertices and edges and their representation in the surface. A graph is said to be *planar* if there exists a drawing of it in the plane $\mathbb{R}^2$ such that if e and f are distinct edges, then the images of γ_e and γ_f do not intersect, except possibly at their endpoints.

Let K_5 be the complete graph on five vertices, i.e. the graph on five vertices whose vertex set is a clique, and let $K_{3,3}$ be the graph on six vertices consisting of two stable sets A and B such that $|A| = |B| = 3$ and every vertex of A is adjacent to every vertex of B. Then Kuratowski's theorem ([68]) gives a purely combinatorial characterization of the graphs that can be drawn in the plane without crossing edges:

Theorem 2.2 (Kuratowski)**.** *A graph G is planar if and only if it does not contain K_5 or $K_{3,3}$ as a minor.*

Since the plane $\mathbb{R}^2$ is homeomorphic to the sphere S^2 minus one point, there is a natural way to move back and forth between drawings of a planar graph G on either surface.

2.4 NP-completeness and approximation algorithms

In order to prove the NP-hardness of some of the problems we consider in this thesis, we use reductions from the following standard NP-complete problems (see e.g. [37]).

3SAT: Given Boolean variables $x_1, \ldots, x_n$ and disjunctive clauses $C_1, \ldots, C_m$ containing 3 literals each, is there an assignment of true-false values to the variables such that the formula $C_1 \wedge C_2 \wedge \cdots \wedge C_m$ is satisfied?

Set Cover: Given a finite set S, a collection $\mathcal{U}$ of subsets of S, and an integer k, is there a set $\mathcal{C} = \{U_1, \ldots, U_k\} \subseteq \mathcal{U}$ of size k such that every $x \in S$ is in some $U_i \in \mathcal{C}$?

Vertex Cover: This is a special case of the Set Cover problem. Given a graph G and an integer k, is there a set of vertices C with $|C| = k$ such that every edge contains a vertex from C?

Stable Set: Given a graph G and an integer k, does G contain a stable set of size k?

Dominating Set: Given a graph G and an integer k, does G contain a set of vertices D with $|D| = k$ such that every $v \in V(G)$ either belongs to D or is adjacent to a vertex of D?

Another important NP-complete problem which we will refer to is

Steiner Tree: Given an undirected graph $G = (V, E)$ with nonnegative edge costs, whose vertices are partitioned into two sets, the *required* vertices

and the *Steiner* vertices, find a minimum cost tree in G that contains all the required vertices (and any subset of the Steiner vertices).

Faced with an NP-complete or NP-hard problem, one often turns to *approximation algorithms*. Consider the generic optimisation problem

$$\max\{f(x) \,|\, x \in X\},$$

where $f(x) \geq 0$ for all $x \in X$. Further suppose that $x^* \in X$ maximises f. Then if $\alpha > 1$, we call an algorithm an *α-approximation algorithm* if it runs in polynomial time and returns a solution y satisfying

$$f(y) \geq \frac{f(x^*)}{\alpha}.$$

Similarly, for a minimisation problem of the form

$$\min\{g(x) \,|\, x \in X\},$$

with $g(x) \geq 0$ for all $x \in X$ and optimal solution x^*, an α-approximation algorithm ($\alpha > 1$) would run in polynomial time and return y such that

$$g(y) \leq \alpha g(x^*).$$

We close the chapter by introducing the asymptotic notation we use to describe the runtimes of the various algorithms presented in this thesis.

Let f and g be non-negative functions defined on the set of natural numbers. We say that $f \in O(g)$ if there exists a positive constant k such that $f(n) \leq kg(n)$ for large enough n. We say that $f \in \Omega(g)$ if the reversed inequality holds, i.e. if $f(n) \geq kg(n)$ for some k and large enough n. If f is asymptotically bounded by g both from above and from below, we write $f \in \Theta(g)$.

Chapter 3

Galaxy Cutsets and Noninterfering Flows

Flows and cuts are often intimately related, either in the form of min-max results, or in the form of cut-conditions, where the value of the cut provides an upper bound for the value of the flow. As we show in this chapter, galaxy cutsets are no exception to this pattern. After defining the flows corresponding to the galaxy cutsets, we proceed to prove a min-max theorem that holds for planar graphs, and then provide an example of a graph showing that the flow-cut gap can be arbitrarily large if the assumption of planarity is omitted.

Given an integer d, and designated non-adjacent vertices s and t, a pair of $s-t$ paths P, Q is called *d-noninterfering* if there is no path of length $d-1$ in $G-\{s,t\}$ between an internal vertex of P and an internal vertex of Q. A *d-noninterfering flow* is a collection $\mathcal{P}$ of pairwise d-noninterfering $s-t$ paths. The value of a d-noninterfering flow is the number of paths in $\mathcal{P}$. For simplicity, we also refer to a d-noninterfering flow as a *d-flow*. We denote by ρ_d the maximum value of a d-flow. Equivalently, ρ_d is the size of a maximum cardinality packing of d-noninterfering $s-t$ paths; consequently we shall call ρ_d the *packing number*.

Recall that a vertex set Z is a *d-star* if it is spanned by a tree T_Z of diameter at most $d-1$. A 1-star, for example, is just a single vertex; a 2-star is a pair of vertices $\{u, v\}$ that induce an edge; a 3-star corresponds to the standard definition of a star. Observe that in particular any path of length $d-1$ is a d-star. Also note that a star of radius r is a d-star for $d = 2r+1$. This is the reason we use diameters of spanning trees rather than depth in this chapter; otherwise we would only obtain d-stars with odd value of d, and as we shall see, this would restrict us to d-flows with odd values of d in the dual packing problem.

Next, recall that a vertex set $\mathcal{G}$ is a *d-galaxy of order k* if it can be written as a union of k d-stars. Given designated non-adjacent vertices s and t, a d-galaxy $\mathcal{G}$ is a *d-galaxy cutset* if s and t lie in different components of $G[V - \mathcal{G}]$. We let τ_d denote the minimum order of a d-galaxy cutset, and call this the *covering number*. The main result of this chapter now is:

Theorem 3.1. *In a planar graph we have $\rho_d \leq \tau_d \leq \rho_d + 1$, for any $d \geq 1$.*

To see how this relates to Menger's theorem, consider the case $d = 1$. A 1-galaxy of order k is just a set of at most k vertices, and a 1-flow between s and t is simply a collection of internally vertex-disjoint $s-t$ paths. Hence Menger's theorem asserts that $\rho_1 = \tau_1$, whether the underlying graph is planar or not. At the end of Section 3.1.3, we will exhibit a simple planar graph showing that the inequality $\tau_d \leq \rho_d + 1$ is best possible when $d \geq 2$.

3.0.1 Packing, covering, and linear programming

Theorem 3.1 is a packing-covering result. So before proving the theorem, we explore this underlying duality in terms of linear programming. First, let us formulate the d-flow problem as an integer linear program. Letting $\mathcal{P}$ be the set of $s-t$ paths in G and $\mathcal{S}_d$ be the set of d-stars in $G - \{s, t\}$,

consider the following integer linear program:

$$\begin{aligned} \max \quad & \sum_{P \in \mathcal{P}} y_P & & \\ \text{s. t.} \quad & \sum_{P : P \cap Z \neq \emptyset} y_P \leq 1 & & \forall Z \in \mathcal{S}_d \\ & y_P \in \{0,1\} & & \forall P \in \mathcal{P} \end{aligned}$$

The constraints imply that at most one path can intersect any d-star; thus, we do indeed obtain the desired integer program. Relaxing the integrality constraints produces the following linear program and its dual.

$$\begin{aligned} (P) \quad \max \quad & \sum_{P \in \mathcal{P}} y_P & & \\ \text{s. t.} \quad & \sum_{P : P \cap Z \neq \emptyset} y_P \leq 1 & & \forall Z \in \mathcal{S}_d \\ & y_P \geq 0 & & \forall P \in \mathcal{P} \end{aligned} \qquad \begin{aligned} (D) \quad \min \quad & \sum_{Z \in \mathcal{S}_d} x_Z & & \\ \text{s. t.} \quad & \sum_{Z : Z \cap P \neq \emptyset} x_Z \geq 1 & & \forall P \in \mathcal{P} \\ & x_Z \geq 0 & & \forall Z \in \mathcal{S}_d \end{aligned}$$

An integral solution to the dual corresponds to a collection of d-stars. Furthermore, the dual constraints then state that every $s-t$ path P must be hit by at least one of the selected d-stars. Thus, every feasible integral dual solution gives a d-galaxy cutset.

Strong duality implies that the maximum value of a fractional d-flow equals the minimum cardinality of a fractional d-galaxy cutset. This equality holds in any graph. Theorem 3.1 then asserts that, for planar graphs, the optimal integral solution to the primal and the optimal integral solution to the dual differ by at most 1.

We remark that we only need a polynomial number of constraints in the primal (and, thus, a polynomial number of dual variables). The reason is that we can restrict our attention to d-stars that are spanned by maximal trees of diameter $d-1$. By a maximal tree $T \subset G - \{s,t\}$ of diameter $d-1$ we mean a tree with the property that we increase the diameter to d if we add a neighbour $u \in G - (T \cup \{s,t\})$ of some vertex $w \in T$ to the tree T.

These maximal trees either have a vertex as their centre[1], if d is odd, or have an edge as their centre, if d is even.

Clearly there can be at most one maximal tree with a given vertex or edge as its centre, so the number of constraints in the primal is bounded above by $n = |V(G)|$ if d is odd and by $m = |E(G)|$ if d is even.

However, even though the number of constraints in the primal is polynomial, the number of variables is exponential, and it is an open problem whether or not the LP can be solved in polynomial time for a non-planar input graph G and $d \geq 2$.

3.1 Proving the minmax theorem

We now prove Theorem 3.1. To do this, we first discuss, in Section 3.1.1, a class of curves in planar graphs called *d-alternate closed curves*. As we will see, these curves relate closely to both galaxy cutsets and non-interfering flows.

3.1.1 d-alternate curves

In a planar graph G with designated non-adjacent vertices s and t, a *d-alternate closed curve* $\mathcal{C} = \{Q_1, C_1, Q_2, C_2, \ldots, Q_l, C_l\}$ is a sequence where each Q_i is a simple path of length at most $d-1$ in $G - \{s,t\}$ and C_j is a curve in the plane which connects the last vertex of Q_j to the first vertex of $Q_{j+1 \,(\mathrm{mod}\, l)}$ and which does not intersect G in any edges or vertices other than its endvertices. Equivalently, for any j the last vertex of Q_j and the first vertex of $Q_{j+1 \,(\mathrm{mod}\, l)}$ lie on a common face.

[1]To obtain the *centre* of a tree T, one removes all the leaves of T to obtain a tree T'. Iterating this procedure results in either a single vertex or a single edge, called the *centre*.

Two properties of a d-alternate closed curve $\mathcal{C}$ are of particular interest here: its length and its winding number. The *length* $l(\mathcal{C})$ is simply the number of path segments $\mathcal{C}$ contains. Note that we can embed G on the sphere S^2 and think of $\mathcal{C}$ as a continuous closed curve in the geometric sense, that is, as a continuous map $\gamma : [0,1] \Longrightarrow S^2 - \{s,t\}$. Interpreted like this, $\mathcal{C}$ has a winding number $w(\mathcal{C})$ - loosely speaking, this is the number of times $\mathcal{C}$ "separates" s from t. To define the winding number precisely, take any continuous curve K from s to t on S^2 and choose an orientation for $\mathcal{C}$. Now let $\lambda(\mathcal{C})$ be the number of times $\mathcal{C}$ crosses K from left to right, and let $\mu(\mathcal{C})$ be the number of times $\mathcal{C}$ crosses K from right to left. The *winding number* $w(\mathcal{C})$ is defined to be $\mu(\mathcal{C}) - \lambda(\mathcal{C})$. (It is not hard to see that $w(\mathcal{C})$ is well-defined; specifically, it is independent of the choice of path K.) It is a well-known fact of topology (proven for example in [76]) that the curve $\mathcal{C}$ separates s from t (that is, s and t are in different components of the plane after the removal of $\mathcal{C}$) if and only if $|w(\mathcal{C})| \geq 1$.

In Section 3.1.2, we establish a relationship between the minimum order of a d-galaxy cutset and the minimum length of a d-alternate closed curve that separates s from t. This relationship will form the base case for an inductive proof, given in Section 3.1.3, that bounds the minimum order of a galaxy cutset from above by the minimum possible ratio of length to winding number of a d-alternate closed curve. Finally, we invoke a theorem by McDiarmid et al. that relates the maximum number of d-noninterfering paths to this same ratio. Theorem 3.1 then follows.

3.1.2 Minimum length curves and galaxy cutsets

Here we show that the minimum length of a d-alternate closed curve that separates s from t equals the minimum order of a d-galaxy cutset.

Theorem 3.2. *The minimum length l of a d-alternate closed curve $\mathcal{C}$ with $|w(\mathcal{C})| \geq 1$ equals the minimum order τ_d of a d-galaxy cutset.*

Proof. First, let $\mathcal{C}$ be a d-alternate closed curve of length l. Recall that each path segment in $\mathcal{C}$ has length at most $d-1$. Thus each has diameter at most $d-1$ and so, by definition, is a d-star. Thus the union of these path segments is a d-galaxy $\mathcal{G}$ of order l. Furthermore, as $|w(\mathcal{C})| \geq 1$, the removal of $\mathcal{G}$ separates s from t. Thus, $\mathcal{G}$ is a d-galaxy cutset and $\tau_d \leq l$.

Now let $\mathcal{G}$ be a d-galaxy cutset of order τ_d. To complete the theorem we need to construct a d-alternate closed curve $\mathcal{C}$ of length τ_d that has non-zero winding number. We do this by first considering the special case $d = 1$, and then reducing the general case to it.

If $d = 1$, a d-galaxy cutset is simply an $s - t$ separator, i.e. $\mathcal{G} = \{v_1, v_2, \ldots, v_{\tau_d}\}$. Since τ_d is the minimum order of a d-galaxy cutset, it follows that $\mathcal{G} - v_i$ is not an $s - t$ separator for any i between 1 and τ_d. Hence, there exists a collection of internally vertex disjoint $s - t$ paths $\{P_1, P_2, \ldots, P_{\tau_d}\}$ such that, for each i, the path P_i contains v_i but none of the vertices $v_j, j \neq i$. Since the paths P_i share the same endpoints and are internally vertex-disjoint, we may assume, by planarity, that they do not cross. So we can order them anti-clockwise around s as $P_1, P_2, \ldots, P_{\tau_d}$. This induces a corresponding ordering of the vertices $v_1, v_2, \ldots, v_{\tau_d}$ of $\mathcal{G}$.

Claim 3.3. *For each i, v_i lies on a common face with $v_{i+1\,(mod\,\tau_d)}$.*

Proof. Let S and T denote the components of $G - \mathcal{G}$ containing s respectively t. It is clear that each v_i must have an edge into both S and T, for otherwise $\mathcal{G} - v_i$ would be a smaller $s - t$ separator.

Consider the graph G' obtained by contracting S and T into super-vertices s' and t', respectively. This contraction transforms each path P_i, $1 \leq i \leq \tau_d$, into an an $s' - t'$ path P_i' of length 2 via v_i in G'. Moreover,

since $\mathcal{G}$ was an $s-t$ separator, any such path P_i' must intersect v_i for some i. Clearly the ordering of the paths P_i in G induces the same ordering of the paths P_i'.

Now suppose that v_i and v_{i+1} were not on a common face in G. This implies that there is a cycle W separating them. Since G is planar and both v_i and v_{i+1} have edges into S and T, W must contain vertices from both S and T. Then contracting S and T turns W into a 4-cycle W' separating v_i and v_{i+1} in G' (note that we do not contract any of the edges incident to the vertices of $\mathcal{G}$). But this is impossible, since there can be no path P_j' between P_i' and P_{i+1}'. □

Claim 3.3 allows us to construct a 1-alternate closed curve $\mathcal{C} = \{Q_1, C_1, Q_2, C_2, \ldots, Q_{\tau_d}, C_{\tau_d}\}$ from $\mathcal{G}$ as follows: Each path segment Q_i is given by v_i. Since v_i and v_{i+1} share a face, it is clear that this is a valid 1-alternate curve. As the removal of $\mathcal{C}$ separates s and t, it follows that we must have $|w(\mathcal{C})| \geq 1$.

Hence in the special case $d = 1$ the length l of a minimum d-alternate closed curve with non-zero winding number is at most τ_d, as claimed.

It remains to extend this to the general case. Suppose we have a d-galaxy cutset $\mathcal{G} = \{Z_1, Z_2, \ldots, Z_{\tau_d}\}$ of minimum order, where now $d \geq 2$. Our strategy will be to contract the d-stars into vertices, and then apply the result for the case $d = 1$. For this strategy to be successful, we need the d-stars to be disjoint. The following two claims show that we may assume this to be the case.

Claim 3.4. *The d-stars Z_i may be assumed to be paths.*

Proof. We shall assume that, among all d-galaxy cutsets of order τ_d, our galaxy $\mathcal{G}$ is minimum with respect to the number of vertices it contains.

For each i, let T_i be a tree of diameter at most $d-1$ spanning Z_i. Denote by S respectively U the component of $G - \mathcal{G}$ containing s respectively t.

For $i \neq j$, if some u is a leaf of T_i and of T_j, or a leaf of T_i and a non-leaf of T_j, we replace T_i by $T_i' := T_i - \{u\}$. This does not change the vertex set of the galaxy $\mathcal{G}$, and T_i' is clearly still a d-star. We may therefore assume that if u is a leaf of some T_i, then it does not belong to any other tree T_j with $j \neq i$.

We shall call a vertex $u \in \mathcal{G}$ *essential* if it has an edge into S and an edge into U. Suppose that some leaf $u \in T_i$ is not essential. Then, since u is not in any other tree, $\mathcal{G}' := \mathcal{G} - \{u\}$ is still a d-galaxy of order at most τ_d separating s and t, contradicting the minimality (with respect to the number of vertices) of $\mathcal{G}$. Hence, every leaf of every tree T_i must be essential.

If, for an index i, the tree T_i has two leaves, then T_i is a path, and there is nothing more to show. So assume that T_i has at least three leaves, say u, v and w. By the above argument, u, v and w must be essential. But then G contains a $K_{3,3}$ minor, which is obtained by contracting S, U and $T_i - \{u, v, w\}$ into vertices and deleting $V - (S \cup U \cup T_i)$. Hence, each T_i can have at most two leaves, i.e. it must be a path. □

Claim 3.5. *The d-stars Z_i may be assumed to be disjoint.*

Proof. By Claim 3.4, we may assume the trees T_i spanning the d-stars to be paths. Suppose that for $i \neq j$, there is a vertex $u \in T_i \cap T_j$. If u is a leaf of T_i, we can simply replace T_i by $T_i' := T_i - \{u\}$. This does not change the set of vertices of $\mathcal{G}$, and T_i' is still a d-star. Similarly if u is a leaf of T_j.

However, if u is an internal vertex of both paths T_i and T_j, we again obtain a $K_{3,3}$ minor: One stable set of the $K_{3,3}$ minor consists of $\{v_i, w_i, w_j\}$ where v_i, w_i are leaves of T_i and w_j is a leaf of T_j, and the other stable set is obtained by contracting $(T_i \cup T_j) - \{v_i, w_i, w_j\}$, S and U (recall that the leaves of T_i and T_j must be essential). □

Now, consider the graph G^* obtained by contracting each Z_i into a vertex v_i^*. Since we can assume the Z_i to be disjoint, the vertices v_i^* are distinct.

Clearly G^* is again a planar graph, and now $\mathcal{G}^* = \{v_1^*, v_2^*, \ldots, v_{\tau_d}^*\}$ is an $s-t$ separator of order τ_d in G^*. Moreover, τ_d must be the minimum order of an $s-t$ separator in G^*, for if $\mathcal{G}'$ were a smaller $s-t$ separator in G^*, we could obtain a d-galaxy cutset of order less than τ_d in G. To see this, take each vertex u of $\mathcal{G}'$ which did not arise from a contracted d-star (so it is also a vertex in G), and for every vertex v_i^* in $\mathcal{G}'$ which did come from a contracted d-star Z_i, we take Z_i.

Now we know that in G^* we can find a 1-alternate curve of length at most τ_d separating s and t. But since in G^* the vertex v_i^* shares a face with v_{i+1}^*, it follows that some vertex $b_i \in Z_i$ must share a face with some vertex $a_{i+1} \in Z_{i+1}$. As each Z_i has diameter at most $d-1$, we can construct a d-alternate curve $\{C_1, Q_1, \ldots, C_{\tau_d}, Q_{\tau_d}\}$ in G by taking the path segment C_i to be the subpath of Z_i between the vertices a_i and b_i. □

3.1.3 Galaxy cutsets and curves with winding number at least two

We now use Theorem 3.2 to relate the minimum order of a d-galaxy cutset to the minimum ratio between the length and winding number of a d-alternate closed curve. The reason we do this is because this ratio also bounds the packing number, thus allowing us to derive the minmax result. For the remainder of this section, we shall without loss of generality assume winding numbers to be non-negative - indeed, if a curve $\mathcal{C}$ has winding number w, then the curve $\mathcal{C}^*$ obtained by traversing $\mathcal{C}$ in the opposite direction has winding number $-w$.

Theorem 3.6. *Let $\mathcal{C}$ be a d-alternate closed curve with strictly positive winding number. Then there is a d-galaxy cutset of order at most $\lceil \frac{l(\mathcal{C})}{w(\mathcal{C})} \rceil$.*

Proof. We apply induction on w. If $w(\mathcal{C}) = 1$, the assertion follows immediately from Theorem 3.2. So suppose that $w(\mathcal{C}) > 1$. The following lemma allows us to complete the inductive step.

Lemma 3.7. *Given a closed d-alternate curve $\mathcal{C}$ of length $l = l(\mathcal{C})$ and winding number $w = w(\mathcal{C}) \geq 2$, we can find two d-alternate closed curves $\mathcal{C}_1$, with $l_1 = l(\mathcal{C}_1), w_1 = w(\mathcal{C}_1) = 1$ and $\mathcal{C}_2$ with $l_2 = l(\mathcal{C}_2), w_2 = w(\mathcal{C}_2) = w - 1$, such that $l_1 + l_2 \leq l + 1$.*

With Lemma 3.7 in hand, we can complete the proof of Theorem 3.6. If the length of $\mathcal{C}_1$ satisfies $l_1 \leq \lceil \frac{l(\mathcal{C})}{w(\mathcal{C})} \rceil$ then, as $w_1 = 1$, by Theorem 3.2 there is a d-galaxy cutset of order at most $\lceil \frac{l(\mathcal{C})}{w(\mathcal{C})} \rceil$ and we are done. So suppose that $l_1 \geq \lceil \frac{l(\mathcal{C})}{w(\mathcal{C})} \rceil + 1$. This implies that

$$l_2 \;\leq\; l + 1 - l_1 \;\leq\; l - \lceil \frac{l(\mathcal{C})}{w(\mathcal{C})} \rceil \;\leq\; l - \frac{l}{w} = \frac{w-1}{w} l.$$

By induction, there is a d-galaxy cutset of order at most

$$\lceil \frac{l_2}{w_2} \rceil \;\leq\; \lceil \frac{w-1}{w} l \cdot \frac{1}{w-1} \rceil \;=\; \lceil \frac{l}{w} \rceil,$$

as required. □

It remains to prove Lemma 3.7. Before doing so, we remark that we cannot prove it directly by splitting $\mathcal{C}$ into curves $\mathcal{C}_1$ and $\mathcal{C}_2$ (at an arbitrary point of self-intersection) and then applying induction on the winding number. This is because it is quite possible for $w(\mathcal{C}_1)$ and $w(\mathcal{C}_2)$ to be larger than $w(\mathcal{C})$ in absolute value.

Proof of Lemma 3.7. Consider an embedding of G on the sphere S^2, and let U be the universal covering space of $S := S^2 - \{s, t\}$, with projection map $\pi : U \to S$. A *covering space* of S is a space U together with a map $\pi : U \to S$ having the following property: S has an open cover $\{W_\alpha\}$ such that for each α, the preimage $\pi^{-1}(W_\alpha)$ is a disjoint union of open sets,

each of which is homeomorphically mapped to W_α by π. The *universal covering space* of a space is the unique (up to isomorphism) covering space that is simply connected, meaning that every continuous closed path on it is homotopic to a point. Roughly speaking, it has no holes. We refer the interested reader to [52] and [76] for more details on covering spaces. For our purposes, it will be enough to think about U in the following simple terms: since the sphere minus two points is homeomorphic to the cylinder, U can be pictured as an infinite strip which is "wrapped around" the cylinder by the projection map π. Then one of the boundaries, say the "upper" boundary, of the strip corresponds to s, and the "lower" boundary corresponds to t. For a point $x \in U$, we denote by $x' \in U$ the element of the preimage of $\pi(x)$ "to the right" of x. Formally, x' can be defined as follows: Let L be a closed curve starting and ending at $\pi(x)$ such that $w(L) = 1$ and L is oriented anti-clockwise around s. Then, letting $\mathcal{L}$ be the unique curve on U that starts at x and is mapped to L by π, the point x' is defined to be the endpoint of $\mathcal{L}$. The curve $\mathcal{L}$ is called a *lifting*[2] of L, and the uniqueness of a lifting starting at a given point of U (in this case x) is a fundamental fact of topology (see [76] for a proof). Figure 3.1 illustrates these concepts.

Given an integer $k \geq 1$, define $x^{(k)} := x'$ if $k = 1$ and $x^{(k)} := (x^{(k-1)})'$ if $k \geq 2$. For an integer $k < 0$, we define $x^{(k)}$ to be the point $y \in U$ such that $y^{(-k)} = x$, and for notational convenience define $x^{(0)} := x$. Similarly, for a curve K on U and an integer k, define $K^{(k)} := \{x^{(k)} \,|\, x \in K\}$.

Now, let K be a lifting of $\mathcal{C}$, and consider the curve

$$\mathcal{K} := \bigcup_{k \in \mathbb{Z}} K^{(k \cdot w(\mathcal{C}))}.$$

Informally speaking, $\mathcal{K}$ is obtained by taking infinitely many liftings of $\mathcal{C}$ one after the other, as shown in Figure 3.2.

[2]More formally, given a continuous map $\gamma : [0,1] \to S$, a lifting of γ is a continuous map $\tilde{\gamma} : [0,1] \to U$ such that $\pi \circ \tilde{\gamma} = \gamma$.

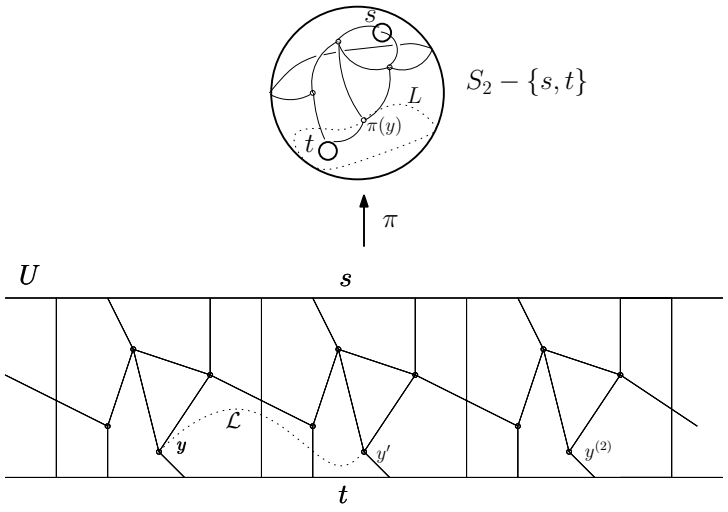

Figure 3.1: The universal covering space of $S^2 - \{s, t\}$

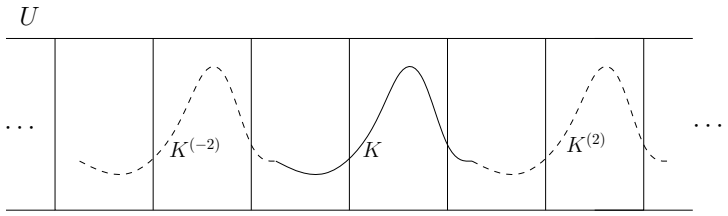

Figure 3.2: Obtaining the curve $\mathcal{K}$

Claim 3.8. *The curves $\mathcal{K}$ and $\mathcal{K}'$ intersect.*

Proof. Consider the points of maximal latitude of $\mathcal{K}$, i.e. the points minimizing the Euclidean distance to the upper boundary of U. If $\mathcal{K}$ and $\mathcal{K}'$ did not intersect, then without loss of generality $\mathcal{K}$ would always lie above $\mathcal{K}'$. But $\mathcal{K}'$ is just $\mathcal{K}$ shifted to the right, so the maximal latitudes of $\mathcal{K}$ and $\mathcal{K}'$ are the same. Hence, at a point of maximum latitude of $\mathcal{K}'$, $\mathcal{K}$ cannot lie above $\mathcal{K}'$, contradiction. [Note that since $\mathcal{C}$ is a continuous curve, any lifting of $\mathcal{C}$ is also continuous, i.e. a continuous map from $[0, 1]$ to U, as shown in [76]. It is then easy to see that the distance of K to the upper boundary of U is a continuous function with domain $[0, 1]$, which implies that points of maximal latitude do indeed exist.] □

Now, by Claim 3.8, $\mathcal{K}$ and $\mathcal{K}'$ must intersect, and since both $\mathcal{K}$ and $\mathcal{K}'$ project to $\mathcal{C}$, this gives a crossing of $\mathcal{C}$ with itself. Let x be a point on U where $\mathcal{K}$ and $\mathcal{K}'$ intersect.

Since $w(\mathcal{C}) > 0$ we chose the orientation for $\mathcal{C}$ that yields a positive winding number. This orientation of $\mathcal{C}$ induces orientations of $\mathcal{K}$ and $\mathcal{K}'$. Because $\mathcal{K}$ goes through x, $\mathcal{K}'$ must go through x'. But as $\mathcal{K}'$ also goes through x, we get a curve $\mathcal{C}_1$ of winding number 1 by projecting down onto S^2 the piece of $\mathcal{K}'$ from x to x'. On the other hand, we can also switch from $\mathcal{K}'$ to $\mathcal{K}$ at x (preserving the orientations) and project the resulting curve onto S^2, which gives a curve $\mathcal{C}_2$ of winding number $w-1$. Since $w(\mathcal{C}_2) \geq 1$, this curve still separates s and t. This is illustrated in Figure 3.3.

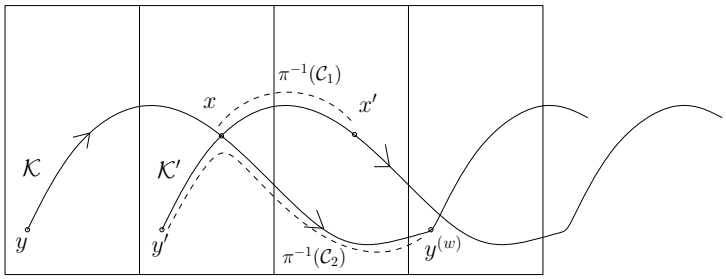

Figure 3.3: Decomposing a curve

The next step is to analyse the nature of the crossing of $\mathcal{C}$ with itself. This will prove that $\mathcal{C}_1$ and $\mathcal{C}_2$ are indeed valid d-alternate curves with the properties described in Lemma 3.7. Suppose first that $\pi(x)$ is not a vertex of G. Since $\mathcal{C}$ only intersects G in vertices, this implies that $\pi(x)$ lies in the interior of a face of G. Thus, it indicates a crossing of two jumps C_i and C_j of $\mathcal{C}$. If C_i connects vertices u_i and w_i, and C_j connects u_j to w_j, then we can simply reroute by jumping from u_i to w_j and u_j to w_i. Thus we obtain two d-alternate curves whose lengths satisfy $l_1 + l_2 = l$.

Next, suppose that $\pi(x) = v \in V(G)$. So v lies in the intersection of two path segments Q_i and Q_j of $\mathcal{C}$. Let Q_i' and Q_j' be the paths obtained

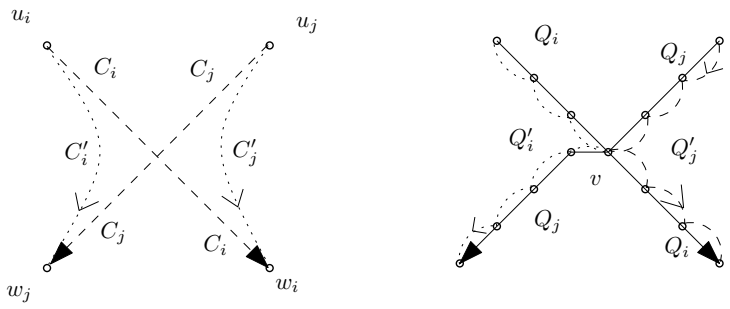

Figure 3.4: Rerouting the jumps and paths

by switching from Q_i to Q_j respectively from Q_j to Q_i at v. Since both Q_i and Q_j contain at most d vertices, we get that $|V(Q'_i)| + |V(Q'_j)| \leq 2d$. If $|V(Q'_i)| \leq d$ and $|V(Q'_j)| \leq d$, then $\mathcal{C}_1$ and $\mathcal{C}_2$ are d-alternate curves with $l_1 + l_2 = l$.

Next, note at most one of Q'_i and Q'_j can contain more than d vertices. Suppose Q'_i does. Then we can separate Q'_i into two path segments of length at most d by adding a jump from an interior vertex of Q'_i to itself. In this case, we obtain d-alternate curves whose lengths satisfy $l_1 + l_2 = l + 1$. Again note that when we reroute, we must do so in an orientation-preserving way, as argued above. This concludes the proof of Lemma 3.7. □

We are now ready to prove our main result, Theorem 3.1. Theorem 3.6 gives an upper bound on the covering number. Furthermore, we have a lower bound on the packing number due to the following theorem of McDiarmid et al. (Theorem B in [71]).

Theorem 3.9 (McDiarmid, Reed, Schrijver, Shepherd)**.** *The maximum number of pairwise d-noninterfering $s-t$ paths equals the minimum of $\lfloor \frac{l(\mathcal{C})}{w(\mathcal{C})} \rfloor$, over all closed d-alternate curves $\mathcal{C}$.* □

We remark that the minimum of $\frac{l(\mathcal{C})}{w(\mathcal{C})}$ must be finite. This follows as it is always possible to find a d-galaxy cutset of finite order, and Theorem 3.2 implies that there must exist an associated d-alternate closed curve with

winding number 1. Theorem 3.1 then follows immediately from the next result.

Theorem 3.10. *Let $\mathcal{C}$ be a d-alternate closed curve minimizing $\lfloor \frac{l(\mathcal{C})}{w(\mathcal{C})} \rfloor$. Then*

$$\rho_d \;=\; \lfloor \frac{l(\mathcal{C})}{w(\mathcal{C})} \rfloor \;\leq\; \tau_d \;\leq\; \lceil \frac{l(\mathcal{C})}{w(\mathcal{C})} \rceil \;\leq\; \rho_d + 1$$

Proof. The equality is Theorem 3.9. The inequality $\tau_d \leq \lceil \frac{l(\mathcal{C})}{w(\mathcal{C})} \rceil$ is Theorem 3.6. The other two inequalities are trivial. □

The planar graph G given in Figure 3.5 shows that the inequality $\tau_d \leq \rho_d + 1$ is best possible: Any two $s - t$ paths are at distance at most 1 from one another, so the maximum value of a 2-flow between s and t is 1. On the other hand, we cannot disconnect s and t by removing a single 2-star in $G - \{s, t\}$, and thus we have that $\tau_2 \geq \rho_2 + 1$.

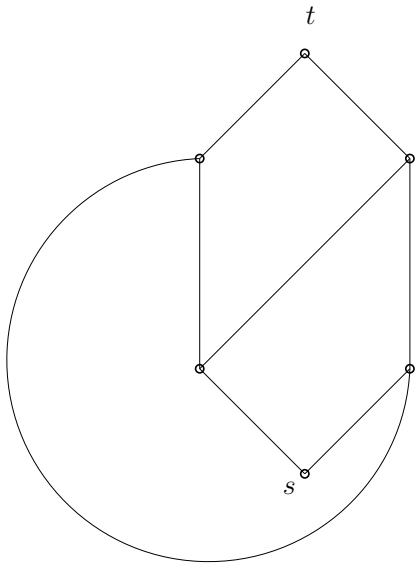

Figure 3.5: Example of a planar graph with $\tau_2 = \rho_2 + 1$

Finally, we remark that τ_d must actually equal $\lceil \frac{l(\mathcal{C})}{w(\mathcal{C})} \rceil$, where $\mathcal{C}$ is a curve minimising the ratio $\frac{l(\mathcal{C})}{w(\mathcal{C})}$. This is because it follows from the work of McDiarmid et al. ([72]) that $\frac{l(\mathcal{C})}{w(\mathcal{C})}$ is the optimal fractional solution of the

linear program we introduced at the beginning of the chapter. Hence, if $\frac{l(\mathcal{C})}{w(\mathcal{C})}$ is not integral, it must be the case that

$$\tau_d = \lceil \frac{l(\mathcal{C})}{w(\mathcal{C})} \rceil = \lfloor \frac{l(\mathcal{C})}{w(\mathcal{C})} \rfloor + 1,$$

for otherwise the optimal integral dual solution would be strictly smaller than the optimal fractional primal solution, which is impossible.

3.1.4 Algorithmic considerations

The proof of Theorem 3.1 is algorithmic. In fact, for any d, given k noninterfering $s-t$ paths, the algorithm of [71] either finds a collection of $k+1$ noninterfering paths, or finds a curve $\mathcal{C}$ with $\lfloor \frac{l(\mathcal{C})}{w(\mathcal{C})} \rfloor = k$. Such a curve acts as a certificate for the optimality of k. Moreover, it is sufficient to consider curves of winding number at most $n = |V(G)|$. This implies that the number of possible crossings of path segments Q_i or jumps C_j is polynomial in n. Thus, our inductive proof of Theorem 3.6 can be turned into a polynomial time algorithm that extracts a simple curve $\mathcal{C}^*$ of length $\lceil \frac{l(\mathcal{C})}{w(\mathcal{C})} \rceil$. From the curve $\mathcal{C}^*$ an optimal d-galaxy is easily obtained.

3.2 An unbounded flow-cut gap

Recall that the value of an optimal fractional d-flow equals the order of an optimal fractional d-galaxy cutset in general graphs. For $d = 1$, the corresponding integral solutions are also equal, by Menger's theorem. However, for larger d, the gap between the integral solutions is easily shown to be unbounded. This is true even for the case $d = 2$.

Theorem 3.11. *There are graphs for which* $\rho_2 \leq \frac{2}{n-2} \cdot \tau_2$.

Proof. Consider the graph G shown in Figure 3.6. It contains a complete bipartite subgraph formed by stable sets A and B of size N. In addition,

there is a vertex s that is adjacent to every vertex in A, and a vertex t that is adjacent to every vertex in B.

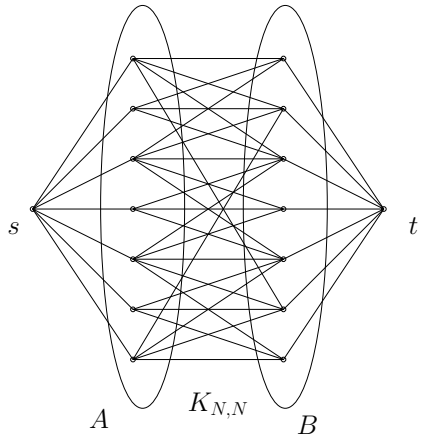

Figure 3.6: An unbounded flow-cut gap

Let $d = 2$. Observe that there cannot be a pair of 2-noninterfering $s - t$ paths, say P_1 and P_2. This follows because each path would then contain internal vertices in both A and B. Consequently, there would be a path of length $1 = d - 1$ between an internal vertex of P_1 and an internal vertex of P_2. Hence, the maximum number of 2-noninterfering paths that can be packed is exactly one.

On the other hand, the order of a minimum 2-galaxy cutset is N. To see this, recall that a 2-star is just a pair of vertices that induce an edge. Now, if we remove at most $N - 1$ pairs of adjacent vertices of $G - \{s, t\}$ then we would be left with at least one vertex $a \in A$ and one vertex $b \in B$. As a and b are adjacent there would remain one $s - t$ path that was not hit by the galaxy. Since G has $n = 2N + 2$ vertices, the result follows. □

Chapter 4

Identifying Galaxy Cutsets

In this chapter, we tackle the fundamental problem of deciding whether a given input graph is vulnerable to virus-like attacks or cascading network failures in the form of galaxy cutsets. For this problem, it will be more convenient to argue in terms of depth rather than diameter of the trees spanning the galaxies. Recall that we defined a set X to be a galaxy of order k and radius r if X can be written as $X = \bigcup_{i=1}^{k} Z_i$, where each Z_i is spanned by a tree of depth at most r. The question we are concerned with is whether a given graph G has a cutset which is a galaxy of order k and radius r.

Intuitively, we are trying to model a scenario where k simultaneous virus attacks occur at the centres of the stars Z_i, and from there spread to vertices at distance at most r from the centres. Since our criterion for functionality of the network after such an attack is whether the unaffected vertices can still communicate with one another, we are naturally led to the question of finding galaxy cutsets. We emphasise that in this chapter, we are not considering the particular case of galaxies separating two given designated vertices s and t.

Not surprisingly, whether or not the galaxy cutsets can be identified in

polynomial time depends on the parameters k and r. We begin by showing in Section 4.1 that if k is assumed to be a fixed constant, we can solve the problem in polynomial time for any r. As the algorithm achieving this is straightforward, the main result of Section 4.1 is a more sophisticated algorithm for the special case $k = 1$ with a run-time of $O(rnm)$, which is significantly faster than the run-time of the trivial algorithm ($O(mn^3)$).

However, as we show in Section 4.2, if k is part of the input the problem of deciding whether there is a galaxy cutset is NP-complete, even if r is fixed to be 1.

4.1 Finding star-cutsets of radius r

We begin by showing that if k is a fixed constant, testing whether a graph G has a galaxy cutset of order k and radius r can easily be done in time $O(n^{k+2}m)$ as follows. Suppose that v and w are vertices of G, and that there exists a galaxy X of order k and radius r such that v and w are in different components of $G - X$. Let $\{u_1, \ldots, u_k\}$ be the centres of the k stars in X. Then if X' is the galaxy consisting of all vertices at distance at most r from one of $u_1, \ldots, u_k$ in $G - \{v, w\}$, it is clear that X' must also separate v and w. Thus, for every $(k + 2)$-tuple $\{v, w, u_1, u_2, \ldots, u_k\}$ we let X be the set of vertices at distance at most r from one of the centres u_i in $G - \{v, w\}$. This set X can be found in time $O(m)$, e.g. by growing BFS-trees of depth r rooted at each of the centres u_i. Then we check whether v and w are in the same component of $G - X$. As there are $O(n^{k+2})$ possible $(k+2)$-tuples and both the BFS algorithm and checking connectivity require $O(m)$ time, the total run-time of this algorithm is $O(n^{k+2}m)$, as claimed.

We will now show that we can significantly improve on this trivial algorithm in the special case where $k = 1$. In other words, we are now concerned with finding a single star-cutset of radius r. The idea of the algorithm is to

decide, for each vertex v, whether v centres a star-cutset of radius r. We will be able to achieve this in time $O(rm)$ using a modified BFS-algorithm, thus obtaining a total run-time of $O(rnm)$.

So, let the graph G and the integer $r \geq 1$ be given. For a vertex $v \in G$, let $D_i(v)$ denote the vertices at distance i from v, and $\tilde{D}(v)$ the vertices at distance at least $r+1$. We will just write D_i and $\tilde{D}$ if the vertex v is fixed and there is no risk of confusion. Note that $\bigcup_{i=0}^{r} D_i$ is spanned by a tree of depth r rooted at v, e.g. a BFS-tree.

The algorithm relies on the following structural results: First, if G itself is a star of radius r, i.e. if $\tilde{D}(v) = \emptyset$ for some v, then G has no star-cutset of radius r if and only if it is a cycle or a clique. Second, if the set of vertices $\tilde{D}(v)$ is non-empty for all vertices v, we will show that G has a star-cutset of radius r centred at v unless the set $\tilde{D}(v)$ is a connected subgraph and every vertex w in $D_i(v)$ can "escape" to $\tilde{D}(v)$ via a safe path consisting of vertices that are at distance strictly greater than r from v in $G - \{w\}$.

We begin with the case where G itself is spanned by a tree of depth at most r.

Lemma 4.1. *Let G be a graph such that $\tilde{D}(v)$ is empty for some $v \in G$, and let $r \geq 1$ be an integer. Then G has no star-cutset of radius r if and only if G is a cycle or a clique.*

Proof. Suppose G has no star-cutset of radius r, and let v be such that $\tilde{D}(v) = \emptyset$. Let $t \leq r$ be the largest integer such that $D_t \neq \emptyset$. The case $t = 0$ is trivial, since then $G = \{v\}$ is obviously a clique. If $t = 1$, G must be a clique as well, because if D_1 contained two non-adjacent vertices x and y, then $V(G) - \{x, y\}$ would be a star-cutset of radius 1. So suppose $t \geq 2$. It is easy to see that for $1 \leq i \leq t-1$ every vertex $w \in D_i$ must have a neighbour in D_{i+1}, otherwise we get a star-cutset of radius i. It is also easy to see that no D_i for $1 \leq i \leq t-1$ can be a clique, since otherwise D_i would

be a star-cutset of radius 1 separating v from the vertices in D_t. Moreover, D_t must be a clique, because $G-\{x,y\}$ is a star of radius t for any $x, y \in D_t$.

We claim that D_i must consist of two non-adjacent vertices for each $1 \leq i \leq t-1$. To see this, suppose that some D_i, $1 \leq i \leq t-1$, contains three vertices $\{x, y, z\}$. Since D_i is not a clique, we may without loss of generality assume that x and y are not adjacent. If $i > t-i$, let w be a vertex in D_{t-i} that lies on a shortest path from v to z. If $i < t-i$, pick $w \in D_{t-i}$ such that z lies on a shortest path from v to w. Such a w must exist, since z has a neighbour in D_{i+1}, which in turn has a neighbour in D_{i+2}, and so on up to D_{t-i}. If $i = t-i$, i.e. if $i = \frac{t}{2}$, pick $w = z$. Note that in all three cases, we have $w \in D_{t-i}$.

In order to derive a contradiction, we will show that in $G-\{x,y\}$, every vertex is at distance at most t from w. To see this, observe that since $(t-i)+i = t$, every vertex of $\{v\} \cup D_1 \cup \cdots \cup D_i$ lies within distance t of w, via a path through v. Moreover, such a path can be chosen to go through z if $i < t-i$ and thus contains neither x nor y. Also, since D_t is a clique, every vertex of $D_{i+1} \cup \cdots D_t$ lies within distance at most $i+1+(t-(i+1)) = t$ from w, via a path through D_t. Again, such a path can be chosen to go through z if $i > t-i$ and thus contains neither x nor y. It follows that the non-adjacent vertices x and y could be separated by removing $G-\{x,y\}$, a star-cutset of radius $t \leq r$ centred at w.

Let u and w be the two vertices of D_{t-1}. To complete the proof, we must show that either $D_t = \{x\}$ for some x adjacent to both u and w, or $D_t = \{x, y\}$ for some x and y such that x is adjacent to u but not to w, and y is adjacent to w but not to u (this completes the cycle). Observe that if, for some vertex $x \in D_t$, each neighbour of x in D_{t-1} is also adjacent to some other vertex $y \in D_t$, then the set $D_t - \{x\} \cup D_{t-1}$ is a star-cutset of radius 2 separating x from v. It follows immediately that $|D_t| \leq 2$, since we know that $|D_{t-1}| = 2$. If $D_t = \{x\}$, it is clear that x must be adjacent to

both vertices of D_{t-1}, and if $D_t = \{x, y\}$, then x and y cannot have common neighbours in D_{t-1}, i.e. either x is adjacent to u and y to w, or vice-versa. Recall that we argued earlier that D_t had to be a clique, so (x, y) must be an edge, which completes the cycle.
For the converse, simply observe that cliques and cycles have no star-cutsets of radius r, for any r. □

So now assume that for every $v \in G$, there is at least one vertex at distance at least $r + 1$ from v, i.e. $\tilde{D}(v) \neq \emptyset$. Given v, denote by $E(D_i)$ the set of edges between vertices in D_i, by W_i the set of vertices $D_i \cup \ldots \cup D_r$, and by G_i the induced subgraph on W_i. Let G'_i be the graph obtained from G_i by removing the edges $E(D_i)$. We say that $w \in D_i$ has an *exclusive* neighbour $u \in D_{i+1}$ if $(w, u) \in E(G)$ and $d_{G'_i}(u, w') > r - i$ for all vertices $w' \in D_i$, $w' \neq w$. Another way of saying this is that u is at distance strictly greater than r from v in $G - \{w\}$. The intuition behind this definition is that these exclusive neighbours provide a safe path to the set $\tilde{D}$ of vertices at distance greater than r from v, if they exist. We formalise this in the following lemma:

Lemma 4.2. *G has no star-cutset of radius r centred at v if and only if $\tilde{D}$ is connected, every vertex in D_r has a neighbour in $\tilde{D}$, and for every $t \leq r-1$, every vertex in D_t has an exclusive neighbour in D_{t+1}.*

Proof. Suppose G has no star-cutset of radius at most r centred at v. It is clear that $\tilde{D}$ must be connected. If there were a vertex $w \in D_r$ without a neighbour in $\tilde{D}$, then $v \cup \bigcup_i D_i - \{w\}$ would be a star-cutset of radius r, contradiction. If some $w \in D_t$ did not have an exclusive neighbour in D_{t+1}, we could find a star-cutset of radius r as follows: The set of vertices $\bigcup_{i=1}^{t} D_i - \{w\}$ is spanned by a tree of depth t rooted at v. Since all of w's neighbours in D_{t+1} are at distance at most $r - t$ from some other vertex of

D_t, there is a tree rooted at v containing $\bigcup_{i=1}^{t+1} D_i - \{w\}$, and this tree spans a star-cutset of radius r.

To prove the converse, it suffices to show that if we remove a tree T of depth at most r rooted at v, there exists a path from every $w \notin T$ to $\tilde{D}$ using only vertices of $V(G) - T$. Clearly this is true if $w \in \tilde{D}$. If $w \in D_r$, then w has a neighbour in $\tilde{D}$ by assumption. If $w \in D_t$ for some $t \leq r-1$, note that its exclusive neighbour w' cannot be contained in any tree of depth r rooted at v that does not also contain w. But then w's exclusive neighbour has an exclusive neighbour w'' in D_{t+2} which cannot be reached by a tree rooted at v not containing w', and so on. This gives a path from w all the way to $\tilde{D}$. □

We remark that for the special case $r = 1$, Lemma 4.2 was stated and proved by Chvátal in [18].

Lemmas 4.1 and 4.2 lead to a fast algorithm for checking whether a graph has a star-cutset of radius r. We proceed to show that the conditions on the graph stated in the lemmas can be tested for in polynomial time.

We write D_t and $\tilde{D}$ instead of $D_t(v)$ and $\tilde{D}(v)$ as there is no risk of confusion, and begin by observing that we can check in time $O(m)$ if G is a clique or a cycle. If that is the case, we know that G is has no star-cutset of radius r and we are done.

If G is neither a clique nor a cycle, we do the following for every vertex v: First we grow a BFS-tree rooted at v. This gives us the sets $D_1, \ldots, D_r$ and $\tilde{D}$. If $\tilde{D}$ is empty, we know G has a star-cutset of radius r by Lemma 4.1, and we are done. If $\tilde{D} \neq \emptyset$, we check if $\tilde{D}$ is connected, and if every vertex in D_r has a neighbour in $\tilde{D}$, which we can do in time $O(m)$.

The most complex step is to check that for each i such that $1 \leq i \leq r-1$, every vertex in D_i has an exclusive neighbour in D_{i+1}. Doing this requires $r-1$ phases, each of which is a modified BFS-algorithm. In phase i, we will

process the level D_i. Phase i in turn will consist of $r - i$ steps. Let i be such that $1 \leq i \leq r - 1$, and suppose that $D_i = \{w_1, \ldots w_p\}$. For a vertex u, denote by $\mathcal{L}^i(u)$ the set of *labels* of u in phase i. At the start of phase i, all vertices in W_{i+1} are unlabelled, i.e. $\mathcal{L}^i(u) = \emptyset$ for all $u \in W_{i+1}$. We use the superscript i to emphasise that the labels in the $r - 1$ phases are independent of one another.

We shall label all the vertices of $W_{i+1} = D_{i+1} \cup \ldots \cup D_r$ using the indices of the vertices in D_i in such a way that at the end of the process, we can simply read off the exclusive neighbours (in D_{i+1}) of the vertices in D_i from the labels. Labelling all the vertices in W_{i+1} will require $r - i$ steps.

The first of those $r - i$ steps is to scan the neighbours of the vertices $w_j \in D_i$ in D_{i+1}, proceeding in a breadth-first-search fashion. When we scan a vertex $u \in D_{i+1}$ adjacent to $w_j \in D_i$, either u has at most one label, or u already has two distinct labels. We update $\mathcal{L}^i(u)$ according to the following rules:

1. If $\mathcal{L}^i(u) = \emptyset$ or $\mathcal{L}^i(u) = \{j'\}$ with $j' \neq j$, set $\mathcal{L}^i(u) := \mathcal{L}^i(u) \cup \{j\}$.
2. If $\mathcal{L}^i = \{j\}$ or $|\mathcal{L}^i(u)| \geq 2$, we do not modify $\mathcal{L}^i(u)$.

We observe that after this first step, all the vertices in D_{i+1} will have one label if they are adjacent to only one vertex of D_i, and two distinct labels if they are adjacent to at least two vertices of D_i. Note that if $i = r - 1$, there is only one step to be performed, so the algorithm terminates here.

In steps 2 to $r - i$, we only work in the graph G_{i+1}, the subgraph of G induced on the vertex set $W_{i+1} = \bigcup_{l=i+1}^{r} D_l$. At each step, we scan the neighbours of vertices in W_{i+1} whose set of labels was modified in the previous step. Suppose we scan a neighbour u of a vertex x. We add labels to u according to the following rules:

1. If $\mathcal{L}^i(u) = \emptyset$ we set $\mathcal{L}^i(u) := \mathcal{L}^i(x)$.

2. If $\mathcal{L}^i(u) = \{j\}$ and $\mathcal{L}^i(x)$ contains some $j' \neq j$, set $\mathcal{L}^i(u) := \mathcal{L}^i(u) \cup \{j'\}$.

3. If $\mathcal{L}^i(u) = \mathcal{L}^i(x) = \{j\}$ or $|\mathcal{L}^i(u)| \geq 2$ we do not modify $\mathcal{L}^i(u)$.

The following claim establishes the connection between our labelling scheme and exclusive neighbours of the vertices $w_j \in D_i$.

Claim 4.3. *After t steps, the unlabelled vertices are the vertices at distance greater than t from D_i, the vertices with one label, say j, are the vertices at distance at most t from some $w_j \in D_i$, but at distance greater than t from all the other vertices in D_i, and the vertices with two labels j, l are the vertices at distance at most t from at least two vertices in D_i, namely w_j and w_l.*

Proof. We use induction on t. We have already observed that the base case is true just after the description of step 1 of the algorithm. So now suppose we just completed step t, and let u be any vertex in W_{i+1}.

If u is unlabelled, it must be at distance greater than t from D_i. Otherwise it would be adjacent to a vertex x at distance at most $t-1$ from D_i, which by induction hypothesis would have a least one label. But then u would have been labelled in step t.

Suppose u has two labels, say j and j'. If u already had label j after step $t-1$, then it is at distance at most $t-1 < t$ from $w_j \in D_i$. If it did not have label j, it inherited it from some vertex x that is at distance at most $t-1$ from w_j, by the induction hypothesis. The same is true for label j'. So after step t, u must be at distance at most t from w_j and $w_{j'}$.

Suppose u only has label j. So it is at distance at most t from w_j, either because it already had label j the step before, or because it inherited it from a vertex at distance at most $t-1$ during step t. If u were at distance at most t from a second vertex $w_{j'}$, then it would also have inherited the label j' by step t. □

For each i with $1 \leq i \leq r-1$, we run the labelling process described above for $t := r - i$ steps. Claim 4.3 then implies that if $u \in D_{i+1}$ is a neighbour of $w_j \in D_i$, then u is an exclusive neighbour if and only if $\mathcal{L}^i(u) = \{j\}$ after $r - i$ steps.

To summarise, each iteration of the labelling process identifies the exclusive neighbours of all the vertices in D_i for some i. Hence, we need $r - 1$ iterations of the labelling process to handle all the vertices in $D_1 \cup \cdots \cup D_{r-1}$. It remains to determine the run-time of each iteration of the labelling algorithm.

Claim 4.4. *Each edge of G_i is considered at most four times during the labelling process.*

Proof. Let $e = (u, \bar{u})$ be an edge in G_i. We only consider e when either u's or $\bar{u}$'s label changes. The label of any vertex can change at most twice, so we consider e at most four times. □

It follows that we can process each level D_i in time $O(m)$. Since there are r levels, we can determine if all the vertices in $W_1 = D_1 \cup D_2 \cup \cdots \cup D_r$ have an exclusive neighbour in time $O(rm)$. Each vertex $v \in V$ could be the centre of a star-cutset of radius r, so the total running time is $O(rnm)$.

4.2 The hardness of finding galaxy cutsets of radius 1

In this section, we show that if the order k of a galaxy is not fixed, but part of the input, then the decision problem of determining whether there is a galaxy cutset of order k is NP-complete, even if all the stars in the galaxy have radius 1.

We remark that every set of vertices X can trivially be written as a galaxy of order $|X|$, and that the problem of finding the minimum k for which X is a galaxy of order k is NP-hard, since the Dominating Set problem can be reduced to it. However, this is not the problem that we are concerned with; we are not a priori given a cutset X as a candidate for the galaxy, but rather have to decide whether the graph can be disconnected by removing up to k stars or not.

Theorem 4.5. *Given a graph G and an integer k, it is NP-complete to decide whether G contains a cutset which is a galaxy of order k.*

We begin by remarking that if we are given a collection of k stars, it is easy to check in polynomial time whether removing those stars disconnects the graph. Hence, the decision problem is in NP. We now prove the hardness by a reduction from Vertex Cover. Let H be a graph with vertices $\{v_1, \ldots, v_n\}$ and edges $\{e_1, \ldots, e_m\}$, and let $k \geq 1$ be an integer. We will construct an auxiliary graph G such that H has a vertex cover of size k if and only if G has a cutset X which is a galaxy of order k. Note that we may assume that $m > k$, as otherwise picking one endpoint of each edge yields a vertex cover of size at most k. If $k \leq 2$ or $k \geq n - 2$, we can decide if there is a vertex cover of size k in polynomial time using brute force, so we shall also assume that $2 < k < n - 2$.

In order to understand the construction of the reduction graph G better, it is helpful to observe two things about graphs without cutsets that are galaxies of order k. Firstly, every vertex must have degree at least $k + 1$, for otherwise we can disconnect that vertex by taking X to consist of all its neighbours. Secondly, if for some vertex v there are vertices whose neighbourhoods have large intersections with the neighbourhood of v, then it is intuitively more likely that there will be a galaxy cutset of order k disconnecting v from the rest of the graph. Our reduction graph consists, roughly speaking, of two trees whose leaves we shall connect by edges in a

specific way. The reason why trees are helpful is that in a tree there is a unique path between any two vertices u and v, and it suffices to remove a single vertex on that path to disconnect u and v. Thus, in a certain sense removing a star in a tree does little more damage than removing a single vertex. The tree-like structure and the way the leaves of the trees are joined by edges will allow us to control how the neighbourhoods of different vertices intersect and as a consequence to place restrictions on which sets of k vertices can possibly form galaxy cutsets.

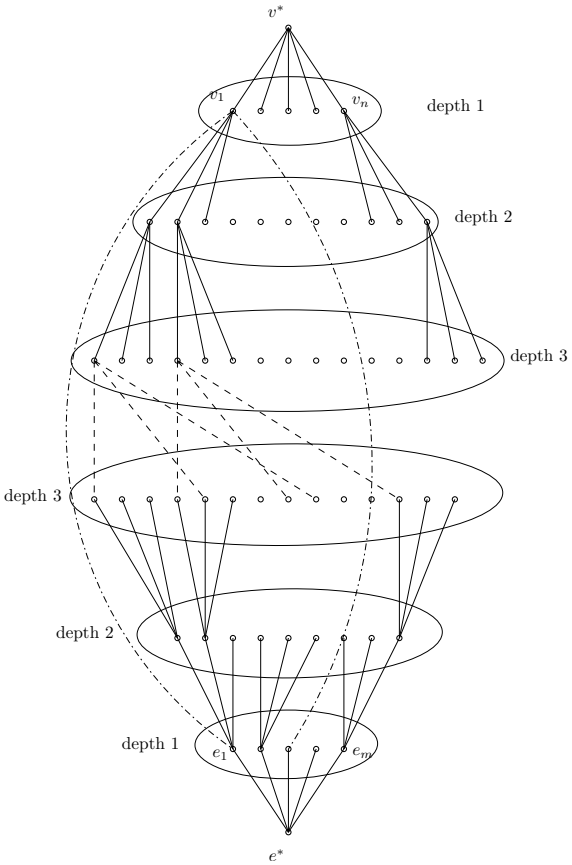

Figure 4.1: Reduction graph for the Vertex Cover reduction

Figure 4.1 shows the high-level structure of G. There are two vertices v^* and e^*. These will be the roots of two trees of depth 3, which we will connect to one another at the leaves. The vertex e^* has m children $\{e_1, \ldots, e_m\}$, corresponding to the edges of H, and the vertex v^* has n children $\{v_1, \ldots, v_n\}$,

corresponding to the vertices of H. If $n \leq 2k$, we add $2k - n + 1$ dummy vertices $\{v_{n+1}, \ldots, v_{2k+1}\}$ as additional children of v^* and set $n' := 2k + 1$, otherwise we add no dummy vertices and set $n' = n$. We call these vertices of depth 1. Next, each vertex v_i has a_i children, and each vertex e_j has b_j children, where the a_i and b_j are chosen as follows. Pick a prime p such that

$$p \geq \max(n' + 1, m + 1, 2k + 4) \tag{4.1}$$

Such a prime can be found in time polynomial in n, since there is always a prime between $N := \max(n' + 1, m + 1, 2k + 4)$ and $2N$. Now let each a_i equal either $\lfloor \frac{p^2}{n'} \rfloor$ or $\lceil \frac{p^2}{n'} \rceil$ and let each b_j equal either $\lfloor \frac{p^2}{m} \rfloor$ or $\lceil \frac{p^2}{m} \rceil$, so that they satisfy $\sum_i a_i = \sum_j b_j = p^2$. Note that we will have $a_i \geq n' \geq 2k + 1 \geq k + 1$ for all i and $b_j \geq m \geq k + 1$ for all j. We call the children of the v_i the grandchildren of v^*, and the children of the e_j the grandchildren of e^*.

Since $\sum_i a_i = p^2$ by construction, there are p^2 grandchildren of v^*, and we label them using the elements of the Abelian group $\mathbb{Z}_p \times \mathbb{Z}_p$ in lexicographic order, i.e.

$$(0, 0)_{v^*}, (0, 1)_{v^*}, \ldots, (0, p - 1)_{v^*}, (1, 0)_{v^*}, \ldots, (p - 1, p - 1)_{v^*}.$$

The p^2 grandchildren of e^* are labeled in the same way, except that the subscript is e^*.

Finally, for all i each child of v_i has $k + 1$ children. These children are numbered and said to have type $1, 2, \ldots, k + 1$ according to their number. Similarly, for all j, each child of e_j has $k + 1$ children numbered 1 to $k + 1$. The grandchildren of v^* and e^* are said to have depth 2, and the greatgrandchildren are said to have depth 3.

The greatgrandchildren inherit the label of their parent, to which we add a superscript to indicate the type, so the children of $(i, j)_{v^*}$ are labeled

$$(i, j)^1_{v^*}, (i, j)^2_{v^*}, (i, j)^3_{v^*}, \ldots, (i, j)^{k+1}_{v^*},$$

and the children of the grandchild labeled $(i,j)_{e^*}$ are

$$(i,j)^1_{e^*}, (i,j)^2_{e^*}, (i,j)^3_{e^*}, \ldots, (i,j)^{k+1}_{e^*}.$$

To complete G, we add a few more edges. If v_i is an endpoint of e_j in H, we add the edge (v_i, e_j) to G. (We do not add edges from the dummy vertices to children of e^*.) The greatgrandchildren of v^* are connected to the greatgrandchildren of e^* as follows: For a greatgrandchild $(i,j)^l_{v^*}$, we add edges to the greatgrandchildren of e^* which are labeled $(i+x, j+x^2)^l_{e^*}$ for $x = 0, \ldots, p-1$. Note that a greatgrandchild of v^* of type l only has edges to greatgrandchildren of e^* which are of the same type. This concludes our description of the graph G.

The hardness of our decision problem is established by the following theorem.

Theorem 4.6. *The graph H has a vertex cover of size k if and only if G has a cutset X which is a galaxy of order k.*

Proof. One direction is straightforward. Suppose H has a vertex cover C of size k. Without loss of generality, assume $C = \{v_1, v_2, \ldots, v_k\}$. Let X be the set consisting of $\{v_1, \ldots, v_l\}$ together with their neighbours among the children of e^*. Then X is the union of k stars. Since C is a vertex cover, X contains all the children of e^*, and so e^* and v^* are in different components of $G - X$, i.e. X is a galaxy cutset of order k in G.

For the other direction, we will show that if H does not have a vertex cover of size k, then G does not have a galaxy cutset. We prove this in two steps. First we establish in Lemma 4.7 that if H has no vertex cover of size k, then any vertex of depth $0, 1$ or 2 in $G - X$ has a child which is also in $G - X$. In other words, for any choice of galaxy X, any vertex in $G - X$ will have a path to a vertex of depth 3 in $G - X$. Then we show in Lemma 4.8 that all the vertices of depth 3 are in the same component of $G - X$.

Lemma 4.7. *Let X be a galaxy of order k in G. Then all of the following hold:*

(1) If $v^* \notin X$, then v^* has a neighbour of depth 1 in $G - X$.

(2) Any vertex of depth 1 in $G - X$ has a neighbour of depth 2 in $G - X$.

(3) If $e^* \notin X$, then e^* has a neighbour of depth 1 in $G - X$ unless H has a vertex cover of size k.

(4) Any vertex of depth 2 in $G - X$ has a neighbour of depth 3 in $G - X$.

Proof. (1) Suppose $v^* \notin X$. A star centred at a grandchild of v^* contains at most one of the vertices v_i, namely its parent, and a star centred at a vertex e_j contains at most two vertices v_i, v_l, corresponding to its endpoints. A star centred at a vertex v_i contains only one child of v^*, namely itself. If a star is centred at a vertex that is neither v^*, a child of v^*, a grandchild of v^*, nor a child of e^*, it cannot contain any of the vertices v_i. So X contains at most $2k$ of the children of v^*. Since we added dummy vertices $\{v_{n+1}, \ldots, v_{2k+1}\}$ in the case $n \leq 2k$, it follows that v^* will have a neighbour of depth 1 in $G - X$.

(2) Let w be a vertex of depth 1 in $G - X$. By construction, any star that is not centred at w contains at most one of w's children. But w has at least $k + 1$ children, so there must be at least one left in $G - X$.

(3) A star centred at a grandchild of e^* can contain at most one child of e^*, namely its parent. A star centred at a vertex v_i can contain at most $d(v_i)$ children of e^*, where $d(v_i)$ is the degree of v_i in H. A star centred at a vertex e_j contains only one child of e^*, namely itself. If a star is centred at a vertex which is neither e^*, a child of e^*, a grandchild of e^* nor a child of v^*, then it cannot contain any of the vertices e_j. So X contains at most M of the vertices e_j, where M is the maximum

number of edges covered by a vertex cover of size k in H. It follows that if $M < m$, i.e. if H has no vertex cover of cardinality k, then e^* will have a neighbour of depth 1 in $G - X$.

(4) For the last case, take a vertex $w \in G - X$ of depth 2, and suppose w is a grandchild of v^*. By construction, w has $k + 1$ children of depth 3. We claim that at most k of them can be contained in X. To see this, observe that a star that contains any of the children of w has to be centred at a vertex u of depth 3 (or at w, but we are assuming that $w \notin X$). If u is a greatgrandchild of v^*, it has to be a child of w, and then the star contains no other children of w other than u itself. If u is a greatgrandchild of e^*, say of type i, then the star can only contain a child of w that is also of type i, and there is only one such child. Hence X contains at most k of the children of w, so w will have a neighbour of depth 3. A similar argument shows that a vertex of depth 2 which is a grandchild of e^* will have a neighbour of depth 3 in $G - X$.

□

Next, we show that the middle layer of depth 3 vertices remains connected after the removal of any galaxy.

Lemma 4.8. *Let X be a galaxy of order k in G. Then the following hold:*

(1) There exists an i_0 such that every vertex of depth 3 and type i_0 in $G - X$ is connected to every other vertex of type i_0 in $G - X$.

(2) For every i and j with $i \neq j$, every vertex of depth 3 and type i in $G - X$ is connected to some vertex of depth 3 and type j in $G - X$.

Note that these two lemmas together imply the theorem. If H has no vertex cover of size k, any vertex not in X is connected to some vertex of depth 3 by Lemma 4.7. But then Lemma 4.8 implies that all the vertices of

depth 3 not in X are in the same component of $G - X$. So $G - X$ must be connected. The proof of Lemma 4.8 relies on the following claim.

Claim 4.9. *For each l, let F_l be the bipartite subgraph of G induced by the vertices of depth 3 and type l. Then F_l has the following properties:*

(a) F_l is p-regular.

(b) If u, w are distinct vertices belonging to the same stable set of F_l, then $|\Gamma(u) \cap \Gamma(w)| \leq 1$.

(c) F_l is $(k+1)$-vertex connected.

Proof. We will fix l and omit the superscript in the vertex labels. We also let A be the stable set of F_l whose vertices are the greatgrandchildren of v^*, and B the stable set of F_l whose vertices are the greatgrandchildren of e^*.

(a) A vertex $(i,j)_{v^*}$ is adjacent to $(i+x, j+x^2)_{e^*}$ for $x = 0, 1, \ldots, p-1$, and a vertex $(i', j')_{e^*}$ has an edge to $(i'-x, j'-x^2)_{v^*}$ for $x = 0, 1 \ldots, p-1$. So F_l is a p-regular bipartite graph. (Addition and subtraction are taken mod p.)

(b) Suppose $u = (i,j)_{v^*}$ and $w = (i',j')_{v^*} \neq (i,j)_{v^*}$ have two distinct common neighbours. So there exist $x, y, \tilde{x}$, and $\tilde{y}$ such that

$$\begin{aligned} i + x &= i' + \tilde{x} && (4.2) \\ j + x^2 &= j' + \tilde{x}^2 && (4.3) \\ i + y &= i' + \tilde{y} && (4.4) \\ j + y^2 &= j' + \tilde{y}^2 && (4.5) \end{aligned}$$

Subtracting (4.4) from (4.2) and (4.5) from (4.3), we see that

$$\begin{aligned} x - y &= \tilde{x} - \tilde{y} && (4.6) \\ x^2 - y^2 &= \tilde{x}^2 - \tilde{y}^2 && (4.7) \end{aligned}$$

Now if $x = y$, the two neighbours of $(i, j)_{v^*}$ are not distinct, which contradicts our assumption. However, if $x \neq y$, we may divide (4.7) by $x - y$ to obtain $x+y = \tilde{x}+\tilde{y}$, which together with (4.6) implies that $x = \tilde{x}$ and $y = \tilde{y}$. In this case, we get that $i = i'$ and $j = j'$, which is also a contradiction. A similar proof shows that (b) holds for two vertices u, w in B.

(c) The $(k + 1)$-vertex connectivity follows from the first two properties. Again, suppose first that u, w are two vertices in A. Let

$$U := \left(\bigcup_{u' \in \Gamma(u)} \Gamma(u') \right) - \{u\}, \; W := \left(\bigcup_{w' \in \Gamma(w)} \Gamma(w') \right) - \{w\},$$

i.e. $U \subset A$ is the set of neighbours of neighbours of u, excluding u itself, and $W \subset A$ is the set of neighbours of neighbours of w, excluding w itself. Let $Z_1 := U \cap W$. Since the neighbourhoods of two neighbours of u only have u in their intersection by (b), we have that $|U| = (p - 1)p$. By the same argument, $|W| = (p-1)p$. But $|A| = p^2$, so we must have $|Z_1| \geq p^2-2p \geq 0$. Now we claim that we can find $k+1$ internally vertex disjoint paths of length 4 between u and w. Pick a vertex u_1 in Z_1. So u_1 is adjacent to a neighbour z_1 of u and to a neighbour z_1' of w, by definition of U and W. This gives the first path $P_1 = \{u, z_1, u_1, z_1', w\}$ of length 4 between u and w. The degenerate case where $z = z'$ can only happen once because $|\Gamma(u) \cap \Gamma(w)| \leq 1$; in that case we get a path of length 2. Now remove z_1, z_1' and all their neighbours except u and w from the graph. So we remove at most $2(p-1)$ vertices from Z_1. Call the resulting set Z_2. Pick a vertex $u_2 \in Z_2 \subset U \cap W$. Then u_2 is adjacent to a neighbour z_2 of u and a neighbour z_2' of w. This gives a path P_2 of length 4 between u and w, and P_2 is internally vertex disjoint from P_1. Doing this $k + 1$ times yields $k + 1$ vertex disjoint paths $P_1, \ldots, P_{k+1}$. Since $|U \cap W| \geq p(p - 2) \geq 2(p - 1)(k + 1)$ because $p \geq N \geq 2k + 4$ (Equation 4.1), all of the sets $Z_1, Z_2, \ldots, Z_k$ will be non-empty. A similar argument shows that we can find $k + 1$ vertex disjoint paths between any two vertices u' and w' of B. Finally, any vertex $u \in A$ has $p > k + 1$ neighbours in B,

so no cutset of size k in F_l can contain all of the neighbours of u. Similarly for $u' \in B$. Hence F_l is $(k+1)$-vertex connected. $\square$

We now proceed to prove part (1) of Lemma 4.8. The proof is based on two simple observations. Firstly, if a star T is centred at a vertex u of depth 3, then T can contain at most $p+1$ vertices of depth 3, namely u itself and all its neighbours. But the crucial point is that T will only contain depth 3 vertices of one type, the same type as the centre u. Secondly, if a star T is centred at a vertex w of depth 2, it can contain at most $k+1$ vertices of depth 3, namely the children of w. The crucial point here is that among the children of w, for each $i = 1, \ldots, k+1$ there will be at most one vertex of type i. A star centred at a vertex which is not of depth 2 or 3 cannot contain any vertices of depth 3.

So now let X be a galaxy of order k, say $X = \bigcup_{j=1}^{k} T_j$, where each T_j is a star centred at w_j. We need to show that there exists an i_0 such that all the vertices of type i_0 which are not in X are in the same component of $G - X$. Now if a centre w_j is of depth 3 and type i, ignore all vertices of that type completely. Since there are $k+1$ types, there is an i_0 such that no centre w_j is of type i_0. By Claim 4.9, the subgraph F_{i_o} spanned by the vertices of type i_0 is $(k+1)$-connected. Since there are no stars with a centre of type i_0, each star contains at most one vertex of type i_0 by the second observation above. So X contains at most k vertices of type i_0, and thus $F_{i_0} - X$ is connected, so all vertices of type i_0 are in the same component of $G - X$, as claimed.

For the proof of (2), let $u \in G - X$ be a vertex of type i and depth 3, and assume u is a greatgrandchild of v^*. The vertex u has p neighbours among the greatgrandchildren of e^*. Any two of these only have u as a common neighbour, so u is joined to $p(p-1)$ greatgrandchildren of v^* through vertex disjoint paths of length 2. It follows that u is joined to $p(p-1)$ grandchildren of v^* through vertex disjoint paths of length 3. We think of the grandchildren

of v^* as *hubs* that let us reach vertices of type $j \neq i$. We claim that $G - X$ contains at least one hub and all its children.

Call a hub z *useless* if z or one of its children is contained in X, or if a vertex on one of the $u - z$ paths described above is contained in X. We shall bound the maximum number of useless hubs. To this end, let x_1 be the number of stars in X centred at vertices of depth 1, let x_2 be the number of stars centred at vertices of depth 2, let x_3 be the number of stars centred at vertices of depth 3 and type i, and let x_3' be the number of vertices centred at vertices of depth 3 and type $j \neq i$.

Now note that a star centred at a vertex of depth 1 can render at most $\lceil \frac{p^2}{n'} \rceil$ hubs useless. A vertex of depth 2 can render at most one hub useless. A vertex of depth 3 and type i can render at most p hubs useless, and a vertex of type 3 and type $j \neq i$ can render at most p hubs useless. Thus the total number of useless hubs is bounded above by

$$x_1 \lceil \frac{p^2}{n'} \rceil + x_2 + x_3 p + x_3' p.$$

Note that $x_1 + x_2 + x_3 + x_3' \leq k$. Clearly the worst case occurs when $x_1 = k$, and then there are $k\lceil \frac{p^2}{n'} \rceil$ useless hubs. Since $n' - k > 2$, we have that $\frac{n'-k}{n'} > \frac{2}{n'} \geq \frac{1}{p} + \frac{k}{p^2}$ (as $p \geq n'$ and $k \leq n'$), and so $p^2(1 - \frac{k}{n'}) > p + k$ and thus

$$p(p-1) > k \lceil \frac{p^2}{n'} \rceil.$$

So u will be connected to a vertex of depth 2 and all of its children in $G - X$, as required.

A analogous argument works when u is a greatgrandchild of e^*, and we have thus proved Lemma 4.8 □

This completes the proof of Theorem 4.6, and thus establishes Theorem 4.5. □

We close this chapter by remarking that the proof of Theorem 4.5 does

not yield any information about the hardness of approximation of the problem of finding a minimum galaxy cutset. This is because there is no one-to-one correspondence between vertex covers in the Vertex Cover instance and galaxy cutsets in the reduction graph we have used. Indeed, for any Vertex Cover instance with input parameter k, our reduction graph will have galaxy cutsets of order $k+2$, since any vertex of depth 2 can be disconnected from v^* by removing its $k+2$ neighbours.

Chapter 5

Weighted Noninterfering Flows

In this chapter, we prove a $\Theta(\log n)$ approximability bound for a weighted extension of the d-flows that we defined in Chapter 3. In order to introduce this weighted version of d-noninterfering flows recall the LP formulation of the unweighted flow problem:

$$\begin{aligned} \max \quad & \sum_{P \in \mathcal{P}} y_P & & \\ \text{s. t.} \quad & \sum_{P: P \cap Z \neq \emptyset} y_P \;\leq\; 1 & & \forall Z \in \mathcal{S}_d \\ & y_P \;\in\; \{0,1\} & & \forall P \in \mathcal{P} \end{aligned}$$

where $\mathcal{S}_d$ is the set of all d-stars in $G - \{s,t\}$, and $\mathcal{P}$ is the set of all $s-t$ paths.

Now, to define the weighted problem, suppose that we have non-negative weights or capacities on the edges of the graph, and consider the integer linear program

$$
\begin{array}{lll}
\max & \displaystyle\sum_{P\in\mathcal{P}} w_P y_P & \\
\text{s. t.} & \displaystyle\sum_{P:P\cap Z\neq\emptyset} y_P \leq 1 & \forall Z \in \mathcal{S}_d \\
& y_P \in \{0,1\} & \forall P \in \mathcal{P}
\end{array}
$$

where the weight of an $s-t$ path P is defined to be $w_P := \min\{w(e) \mid e \in P\}$.

This method of determining weights for paths arises naturally if we assume that we first have to decide on a collection of d-noninterfering paths, and that once the paths have been fixed, we send as much flow as possible along each path. Thus, the maximum amount of flow that a path can carry is the minimum capacity of an edge on the path. The total value of the flow is then the sum of the flows along each path, $\sum_{P\in\mathcal{P}} w_P$. Our goal is to obtain a flow $\mathcal{P}$ consisting of d-noninterfering paths that has maximum value.

As pointed out in Chapter 3, the special case $d = 1$ corresponds to vertex-disjoint paths from s to t. For $d = 0$, we define a d-noninterfering flow to be a flow consisting of pairwise edge-disjoint paths. Recall that our minmax result, Theorem 3.1, only holds in planar graphs, and moreover a simple argument shows that the problem of finding an unweighted d-flow in a general graph is hopeless even for $d = 2$. To see this, observe that if we take any given graph G and add vertices s and t adjacent to every $v \in V(G)$, then finding a maximum 2-flow from s to t is equivalent to finding a maximum cardinality stable set in G, and the Stable Set problem cannot be approximated to better than $O(n^{1-\varepsilon})$ unless NP=ZPP (see [51]). We remark that McDiarmid et al. in [71] pointed out that the NP-hardness of finding a maximum 2-flow follows readily from the NP-completeness of the problem of finding a chordless circuit passing through two given vertices

s and t. The NP-completeness of this latter problem was proven by Fellows ([25]). However, the reduction from the Stable Set problem we have given above shows in addition that, if P $\neq$ NP, there is no hope of obtaining a non-trivial approximation algorithm.

Consequently, in studying the disjoint weighted d-flow problem we restrict ourselves to the cases $d = 1$ and $d = 0$, i.e. to paths that are either vertex- or edge-disjoint. We study in detail the case $d = 0$ in undirected graphs, since the results carry over to $d = 1$ and directed graphs using standard reductions, and call the problem of finding a maximum weight collection of edge-disjoint paths between s and t the *Disjoint Weighted Flow Problem.*

We present $\Theta(\log n)$ lower and upper approximation bounds for the Disjoint Weighted Flow problem, where n is the number of vertices. Our lower bound applies even for the special case of planar graphs.

5.1 A lower bound of $\Omega(\log n)$

In this section we present our main result:

Theorem 5.1. *For undirected planar graphs, the hardness of approximation for the maximum disjoint weighted flow problem is* $\Omega(\log n)$*, unless* $P = NP$*.*

Before proving Theorem 5.1 we outline the structure of the proof. First, we introduce a graph G_N that has a maximum disjoint weighted flow of value equal to the harmonic number $H_N \approx \log N$. But if we use a slightly modified weight function for the paths then G_N has a maximum disjoint weighted flow of value one.

We then build a new network $\mathcal{G}$ by replacing each node of G_N by an instance of an NP-hard routing problem. The routing problem will be chosen to have the following properties. If it is a YES-instance then path weightings

for the disjoint weighted flow problem on $\mathcal{G}$ will correspond to the original weighting scheme on G_N. In contrast, if it is a NO-instance, then path weightings for the disjoint weighted flow problem on $\mathcal{G}$ will correspond to the modified weighting scheme on G_N.

It follows that an approximation algorithm with guarantee better than logarithmic would allow us to distinguish between YES- and NO-instances of our routing problem, giving a lower bound of $\Omega(\log N)$. We will see that this bound is equal to $\Theta(\log n)$.

Furthermore, the graphs G_N and $\mathcal{G}$ will be undirected planar graphs. Theorem 5.1 will follow.

5.1.1 A half-grid graph

Let us begin by defining the graph G_N. There are N rows (numbered from top to bottom) and N columns (numbered from left to right). All the edges in the ith row and all the edges in the ith column have weight $\frac{1}{i}$. The ith row extends as far as the ith column and vice versa; thus, we obtain a "half-grid" that is a weighted version of the network considered by Guruswami et al [45]. Finally we add a source s and a sink t. There are edges of weight $\frac{1}{i}$ from s to the first vertex in row i and from t to the last vertex in column i, for $1 \leq i \leq N$. The complete construction is shown in Figure 5.1.

Note that there is a unique $s - t$ path P_i consisting only of edges of weight $\frac{1}{i}$, that is, the hook-shaped path that goes from s along the ith row and then down the ith column to t. Moreover, for $i \neq j$, the path P_i intersects P_j precisely once. Clearly each path P_i has weight $w(P_i) = \frac{1}{i}$, so the collection of edge-disjoint paths $\mathcal{P}^* = \{P_1, P_2, \ldots, P_N\}$ gives a flow of total value $H_N = 1 + \frac{1}{2} + \ldots \frac{1}{N}$. Since every edge incident to s is used in $\mathcal{P}^*$ with its maximum weight, this solution is optimal. Similarly, if we are constrained to use flows that decompose into at most k disjoint paths then

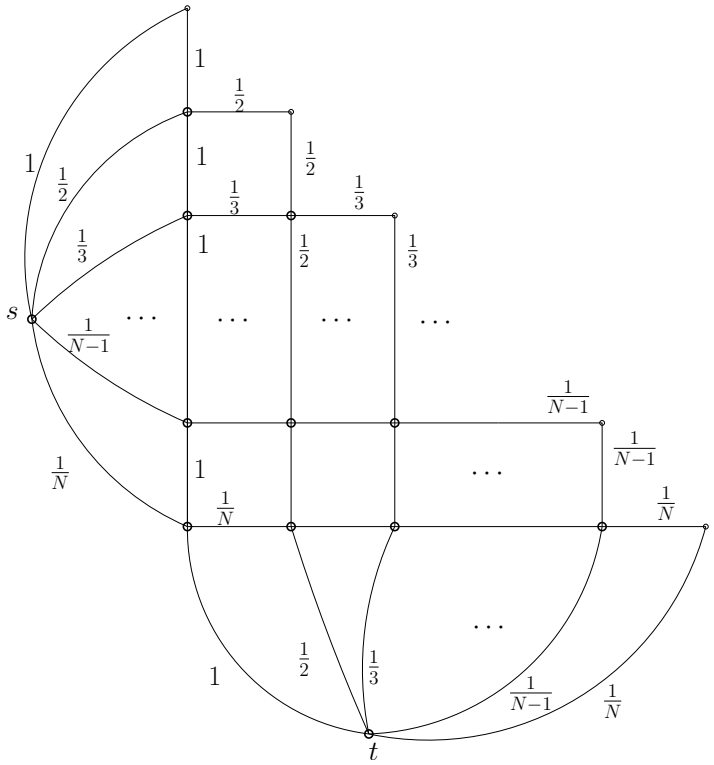

Figure 5.1: Grid Graph G_N.

the optimal flow has weight H_k.

Now consider what happens when we modify the weight function for the paths. Given a collection $\mathcal{P}$ of edge-disjoint paths, let the *modified weight*, $\hat{w}_{\mathcal{P}}(P)$, of a path $P \in \mathcal{P}$ be the minimum weight amongst its edges and those edges incident to a vertex at which P crosses another path $Q \in \mathcal{P}$. Formally,

$$\hat{w}_{\mathcal{P}}(P) = \min\{w_{uv} \,|\, v \in P, uv \in Q \text{ for some } Q \in \mathcal{P}\}$$

where we will omit the subscript if $\mathcal{P}$ is clear.

The maximum value of a flow is significantly reduced if we use these modified weights.

Lemma 5.2. *The maximum value of a weighted flow in G_N under the modified weights is* 1.

Proof. Define the *rank* of a path P to be the index j for which this path uses the weight $\frac{1}{j}$ edge incident to t. Suppose $\frac{1}{i}$ is the maximum modified weight of any path in a flow $\mathcal{P}$. Let j be the rank of some path $Q \in \mathcal{P}$ of modified weight $\frac{1}{i}$. Then set $\mathcal{P}^+$ to be the collection of paths in $\mathcal{P}$ with ranks greater than j, and $\mathcal{P}^-$ to be the paths with ranks less than j.

Observe that Q must contain as a sub-path all the edges in column j that lie below row i. Otherwise, Q would contain an edge in a row of lower weight than $\frac{1}{i}$, contradicting the fact that Q has modified weight $\frac{1}{i}$. Similarly, no other other path in $\mathcal{P}$ crosses Q on this sub-path, as this would reduce Q's modified weight. This implies that any path in $\mathcal{P}^+$ must use one edge of the columns $j+1$ to i between row i and row $i+1$. Consequently, $|\mathcal{P}^+| \leq i - j$. Obviously $|\mathcal{P}^-| \leq j - 1$ and so $|\mathcal{P}| \leq 1 + (i - j) + (j - 1) = i$. Since each path has modified weight at most $\frac{1}{i}$, this gives an upper bound of 1 on the modified value of the flow.

For $\mathcal{P}^* = \{P_1, P_2, \ldots, P_N\}$, we see that $\hat{w}(P_i) = \frac{1}{N}$, for all i. Thus, this collection of paths obtains the maximum modified value of one. □

5.1.2 The 2-edge-disjoint weighted paths problem

The next step is to replace, in G_N, each vertex at the crossing of two paths P_i and P_j with an instance of an NP-hard routing problem. To define this routing problem, let H be an undirected graph whose edges have weight either a or b, where $b > a$. Given two pairs of vertices (s_1, t_1) and (s_2, t_2), we wish to find a path P_1 from s_1 to t_1 and a path P_2 from s_2 to t_2 with the properties that

(i) P_1 and P_2 are edge-disjoint.

(ii) P_2 may only use edges of weight b (P_1 may use either weight a or weight b edges).

We call this the *Two Edge-Disjoint Weighted Paths Problem*, or 2-EDWP.

Evidently, we will be most interested in the case where the graph H is planar. Then we have:

Theorem 5.3. PLANAR-2-EDWP *is NP-hard, even if the pairs of terminals lie on the outer face of H in the order s_1, s_2, t_1, t_2.*

We remark that Theorem 5.3 immediately tells us that the maximum disjoint weighted flow problem is hard in planar graphs. Simply take an instance of PLANAR-2-EDWP and add a super-source s and a super-sink t. Then connect s to s_1 and s_2 with edges of weights a and b, respectively. Similarly, connect t to t_1 and t_2 with edges of weights a and b, respectively. Then there is a disjoint weighted $s-t$ flow of value $a+b$ if and only if there are paths P_1 and P_2 satisfying properties (i) and (ii). Of course, we desire a much stronger hardness result than this, but this observation will be useful in motivating the subsequent construction.

Theorem 5.3 is closely related to a result on unsplittable flows by Guruswami, Khanna, Rajamaran, Shepherd, and Yannakakis. In [45], Guruswami et al. denote by Undir-Node-USF the unsplittable flow problem in an undirected, node-capacitated graph, and prove the following theorem:

Theorem 5.4 (Guruswami et. al [45])**.** *Given an instance of Undir-Node-USF with two source-sink pairs, it is NP-hard to decide if both pairs can be feasibly routed, even if all node capacities are* 1 *or* 2*, and the two demands are* 1 *and* 2*.*

Theorem 5.3 is essentially the edge-version of Theorem 5.4, and moreover we prove that the problem is hard even in planar graphs.

We will need a geometric result to prove Theorem 5.3. An edge $uv \in E(G)$ is a *separating edge* if $G-\{u,v\}$ is disconnected. A triangle $uv, vw, wu \in E(G)$ is a *separating triangle* if $G-\{u,v,w\}$ is disconnected. Our geometric result relies on the following theorem:

Theorem 5.5 (Whitney [95], 1931)**.** *Every maximal planar graph with no separating triangle is Hamiltonian.*

It was shown in [4] that such a Hamiltonian cycle can be computed in linear-time.

Lemma 5.6. *Let $G = (V, E)$ be an embedded planar graph, such that G has girth 4 and has no separating edge. Let $\phi_e \subseteq \mathbb{R}^2$ be the open curve corresponding to the embedding of e, for each edge $e \in E$. Then there is a simple closed curve in $\mathcal{R} = \mathbb{R}^2 \setminus \bigcup_{e \in E} \phi_e$ that intersects the image of every vertex.*

Proof. We define the graph G' obtained form G by adding a new vertex in each face. Each of these new vertices is adjacent to every vertex on the boundary of its face. G' is then obviously a maximal planar graph. In order to apply Theorem 5.5 to G', we must prove that G' does not contain a separating triangle. First assume it. Then we get a Hamiltonian cycle on G'. By slightly shifting this Hamiltonian cycle, we get a curve intersecting G' exactly once on each vertex, and only there. Then, the lemma is proved by simply ignoring the new vertices.

To conclude the proof, we show that G' has no separating triangle. We denote by E' the set of new edges. First, every triangle T has exactly two edges in E', because G has girth 4 (so $|T \cap E'| \neq 0$), the graph induced by E' is bipartite (so $|T \cap E'| \neq 3$), and there is no edge in E' with both extremities on $V(G)$ (so $|T \cap E'| \neq 1$). Suppose that T were a separating triangle, and let e be its edge in G. Then each component of $G' - T$ would contain a vertex of G, since each new vertex is still adjacent to a vertex of G. So e would be a separating edge of G, contradicting the assumption. □

In the following, we identify vertices, edges and graphs with their respective images on the plane. For $\gamma \in \{a, b\}$, we call an edge of weight γ a *γ-edge*.

Proof of Theorem 5.3. We give a reduction from Planar-3-Sat to Planar-2-EDWP. Let $\mathcal{C}$ be a set of clauses over the variables $\mathcal{X}$, such that the bipartite graph $G = (\mathcal{X} \cup \mathcal{C}, \{xC \ : \ x \in \mathcal{X}, C \in \mathcal{C}, x \in C \vee \overline{x} \in C\})$ is planar. Without loss of generality, we can suppose that each variable appears at most three times. To see this, observe that if x appears in $k \geq 4$ clauses we may introduce k new variables, $x_1, \ldots, x_k$, and new clauses $\overline{x_1} \vee x_2$, $\overline{x_2} \vee x_3$, $\ldots$, $\overline{x_k} \vee x_1$, and replace each occurrence of x by an occurrence of one of the x_i. Without loss of generality, we can also suppose that each variable appears exactly once negatively. These transformations can clearly be implemented whilst preserving the planarity of G. Thus, we obtain a formula whose corresponding bipartite graph G has maximum degree 3.

Now take a planar embedding for G. By Lemma 5.6, we may find a closed curve $\mathcal{D}$ intersecting the embedding of G exactly on its vertices.

We will transform $(G, \mathcal{D})$ into an instance of Planar-2-EDWP in polynomial-time. To do this, we need to build an auxiliary edge-weighted planar graph G' for the routing problem. Towards this goal, we first take G and use $\mathcal{D}$ to induce an additional set of embedded a-edges whose endpoints are in $V(G)$.

Then we replace each edge $e = uv \in E(G)$ by a 4-cycle consisting of b-edges us_e, ut_e, vs_e, vt_e, where s_e and t_e are new vertices.

Next we replace each *variable* vertex $x \in \mathcal{X}$ by a variable gadget and each *clause* vertex by a clause gadget. Each variable vertex x of degree three is replaced by one of four possible *variable gadgets*; the actual choice is dependent upon the relative position of $\mathcal{D}$ with respect to the edges incident to x and upon the sign of x in the adjacent clauses. These four gadgets are illustrated in Figure 5.2, where the edges corresponding to $\mathcal{D}$ and the other a-edges are dashed, the edges corresponding to $E(G)$ and the other b-edges are bold (recall there must be two edges out of the gadget for each edge in G as we initially replaced such edges by a 4-cycle). The $+$ and $-$ signs indicate

whether the variable appears positively or negatively in the adjacent clause.

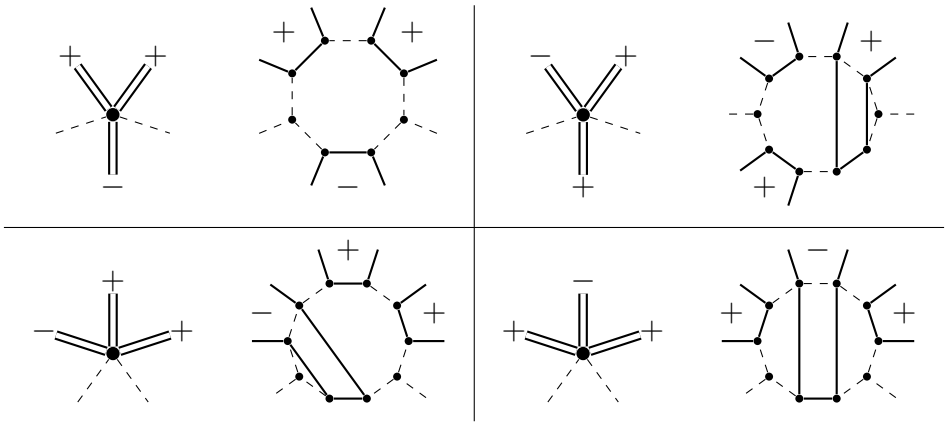

Figure 5.2: Variable gadgets

Each clause vertex C of degree three is replaced by one of two possible *clause gadgets*; again, the actual choice is dependent upon the relative position of $\mathcal{D}$ with respect to the edges incident to C. These two gadgets are shown in Figure 5.3. The gadgets for clauses with two literals and for variables occurring only twice are similar to those presented, but simpler.

To complete the construction we need to specify the sources and the sinks. To do so, we first specify a multi-commodity flow formulation with many source-sink pairs. Later we will show how to implement it as a flow with just two source-sink pairs. Towards the former goal, we will have a source-sink pair (s_e, t_e) for each edge $e \in G$. Furthermore, we will have one additional source-sink pair (s_a, t_a). To define this pair, arbitrarily choose one of the edges uv of $\mathcal{D}$. Then replace uv by two edges us_a and vt_a each with weight a. Observe that s_a and t_a are on the boundary of a common face of the resultant planar graph G'.

This multi-commodity flow problem relates to the planar 3-SAT instance in the following manner.

Claim 5.7. *The formula is satisfiable if and only if there are edge-disjoint paths $\{P_e\}_{e \in E(G)}$ and Q in G', with the following properties.*

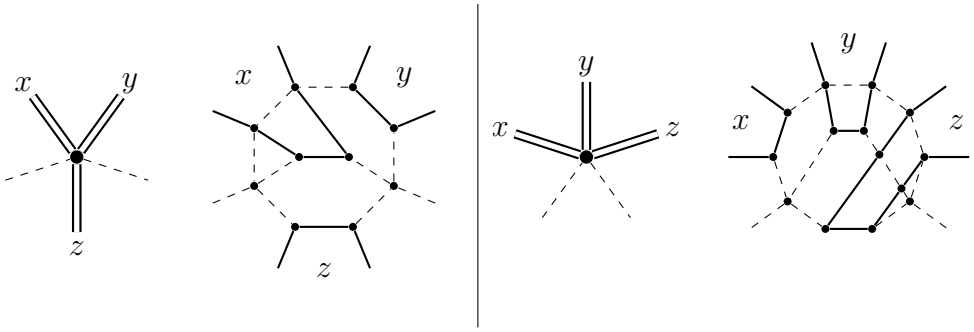

Figure 5.3: Clause gadgets, with the same convention as in Figure 5.2.

(*i*) *P_e has endpoints s_e and t_e and uses only b-edges.*

(*ii*) *Q has endpoints s_a and t_a.*

Proof. First, note the 4-cycles of b-edges that initially replaced each edge have become larger under the construction but are still b-cycles. Moreover, these b-cycles (call them H_e, for each $e \in G$) are edge disjoint and their union covers all of the b-edges in G'.

Now suppose that all the paths exist. There are only two possible routes in H_e between s_e and t_e that P_e can take; if $e = xC$ then one route passes through the variable gadget x and the other passes through the clause gadget C. Since s_e and t_e have degree two, it follows immediately that Q cannot use any of the edges incident to them. Consequently, Q must *follow* the curve $\mathcal{D}$. We will show how to obtain from Q a satisfying truth-assignment.

For any edge $e = xC$, we say that the cycle H_e is *positive* if x appears positively in C, *negative* otherwise. Then, for a variable gadget x, it is easy to see that if Q does not intersect the unique negative cycle going through the gadget then it must use at least one edge of each of the positive cycles H_e going through that gadget. If it intersects the negative cycle, set variable x to *true*, otherwise set it to *false*.

To see that this does produce a satisfying assignment, take any clause C, say over the variables x, y and z. Since Q follows $\mathcal{D}$ it must pass through

each clause gadget. Consequently, Q intersects at least one of H_{xC}, H_{yC}, and H_{zC}. Without loss of generality, let it intersect H_{xC}. This means that P_{xC} cannot go through the clause gadget C and, hence, must go through the variable gadget x. But, again, as Q follows $\mathcal{D}$ it must pass through the variable gadget x too. Therefore, Q cannot intersect H_{xC} in the variable gadget x. This precisely means that x is *true* if x appears positively in C, and x is *false* if it appears negatively. So C is satisfied by x.

On the other hand, given a satisfying assignment, it is easy to find a collection of feasible paths. This is because, for each variable gadget, there is a sub-path that intersects only the positive cycles in that gadget and there is a sub-path that intersects only the negative cycle. Therefore, Q can always follow the appropriate sub-path. □

To complete the proof of Theorem 5.3 we need to reduce the number of commodities in the flow to two. For this, we will keep the source-sink pair (s_a, t_a) but group into one all of the pairs (s_e, t_e) via the use of a new source sink pair (s_b, t_b). To accomplish this, we first need to position the new vertices s_b and t_b in G'. Let $\mathcal{B}$ be a closed curve that intersects G' on s_a and t_a only. Then add s_b arbitrarily on the "upper" path between s_a and t_a induced by $\mathcal{B}$. Similarly add t_b on the "lower" path between s_a and t_a induced by $\mathcal{B}$.

Our goal now is to force any path of b-edges between s_b and t_b to follow b-paths between s_e and t_e for every $e \in G$. To do this, let $e_1, e_2, \ldots, e_m$ be any ordering of the edges of G. For a cycle H_e, we define its *inside* as the connected component of $\mathbb{R}^2 \setminus H_e$ that does not contain any vertex of G'. Then set $\mathcal{R}$ to be the union of the inside of every cycle H_e plus $V(G')$ and the inside of $\mathcal{B}$. Observe that $\mathcal{R}$ is a union of disjoint balls, so its complement is connected. Let P be a path between s_b and s_{e_1} in this complement. Build a path of b-edges along P and add them to G', inserting new vertices whenever P crosses an a-edge (note that these are the only edges P can cross). Next

add P to $\mathcal{R}$; this does not change the connectedness of its complementary set. In this manner, we may iteratively add paths of b-edges between t_i and s_{i+1}, for $1 \leq i \leq m-1$, and finally between between t_m and t_b. By construction, these paths are disjoint and cross only a-edges. We thus obtain a new planar graph G'' with four terminals on the same face, as desired.

Clearly this new instance of Planar-2-EDWP is equivalent to the previous multi-commodity flow problem. To see this, simply note that the new b-edges are isthmi in the subgraph consisting of the b-edges. Consequently, the (s_b, t_b)-path must use each of these new b-edges and then, as before, in each H_e route through either the variable gadget or through the clause gadget. This completes the reduction. □

5.1.3 The hardness result

We can now complete the proof of the approximation hardness. Observe that any vertex of degree four in G_N is incident to two edges of weight $\frac{1}{i}$ and to two edges of weight $\frac{1}{j}$, for some $i \neq j$. We construct a graph $\mathcal{G}$ by replacing each vertex of degree four with the routing graph H. We do this in such a way that the weight $\frac{1}{i}$ edges of G_N are incident to s_1 and t_1, and the weight $\frac{1}{j}$ edges are incident to s_2 and t_2. Moreover, for that copy of H placed at the intersection of P_i and P_j, we then let $a = \frac{1}{i}$ and $b = \frac{1}{j}$, where we may assume that $j < i$.

The hardness result will follow once we see how this construction relates to the original and modified weight functions.

Lemma 5.8. *If H is a YES-instance then the optimal disjoint weighted flow in $\mathcal{G}$ has value H_N. If H is a NO-instance then the optimal disjoint weighted flow in $\mathcal{G}$ has value at most* 1.

Proof. It is clear that if H is a YES-instance, then paths in $\mathcal{G}$ induce paths in G_N which are free to cross at any vertex without restrictions on their

values. This means we obtain a flow of value H_N by using the canonical paths P_i, $1 \leq i \leq N$.

However, if H is a NO-instance, then it contains only an $s_1 - t_1$ path, or only an $s_2 - t_2$ path, or the $s_2 - t_2$ path is forced to use a lower weight a-edge. This implies that the induced paths $\mathcal{P}$ in G_N either do not cross at all, or if they cross then the weight of the path using the $\frac{1}{j}$-edge is forced down to a weight of $\frac{1}{i}$ (recall $j < i$). But this means that the weight of a path is upper bounded by the modified weight function $\hat{w}$. This allows us to apply Lemma 5.2, and hence the value of an optimal flow in this case is at most 1. □

Proof. Proof of Theorem 5.1. It follows that if we could approximate the maximum disjoint weighted flow problem in $\mathcal{G}$ to a factor better than H_N, we could determine whether the optimal solution is 1 or H_N. This in turn would allow us to determine whether H is a YES- or a NO-instance.

Note that $\mathcal{G}$ has $n = \Theta(pN^2)$ edges, where $p = |V(H)|$. If we take $N = \Theta(p^{\frac{1}{2}(\frac{1}{\varepsilon}-1)})$, where $\varepsilon > 0$ is a small constant, then $\log n = \Theta(H_N) = \Theta(\log p)$. This gives our lower bound of $\Omega(\log n)$. □

Similarly, if we are restricted to consider only flows that decompose into k disjoint paths then it is not hard to see that:

Theorem 5.9. *For undirected, planar networks, there is a $\Omega(\log k)$ hardness of approximation, unless $P = NP$, for the problem of finding a maximum flow that decomposes into at most k edge-disjoint paths.* □

5.2 Matching the lower bound

Our lower bound is tight to within a constant factor - there is a simple approximation algorithm that gives an almost matching upper bound.

Theorem 5.10. *For any network, there is an $O(\log n)$ approximation algorithm for the maximum disjoint weighted flow problem.*

Proof. To begin, round each edge weight down to the nearest power of 2. This can only cost us a factor 2 in our approximation guarantee. Next, we claim that we may assume that every edge weight lies between $1 = 2^0$ and 2^t where $t = 1 + \lceil \log n \rceil$. To see this, first note that there can be at most n edge-disjoint $s - t$ paths in any flow. Therefore, for any j, the total contribution from all paths that contain an edge of weight 2^j or less is upper bounded by $n2^j$. Now, let 2^{j_0} be the highest edge weight such that there exists a path of weight 2^{j_0}. Deleting the edges of weight 2^j for all indices j where $2^j < \frac{1}{n}2^{j_0-1}$ loses us at most 2^{j_0-1} in weight, that is, half of the optimal flow value. The lowest remaining edge weight, 2^{j_1}, then satisfies $j_1 \geq j_0 - 1 - \lceil \log n \rceil$. Scaling down the edge weights by a factor 2^{j_1} gives the claim.

The approximation algorithm now proceeds as follows. For each i such that $0 \leq i \leq t = 1 + \lceil \log n \rceil$, let E_i be the edges of weight at least 2^i, and let $G_i = (V, E_i)$. Let ϕ_i be the maximum number of edge-disjoint $s - t$ paths in G_i. Clearly, these paths induce a weighted disjoint flow of value at least $2^i\phi_i$ in G. Furthermore the optimal weighted disjoint flow must have value at most $\sum_{i=0}^{t} 2^i\phi_i$. To see this, note that the paths of weight 2^i in the optimal solution together form a feasible solution for the disjoint paths problem in G_i. Then, since $t = 1 + \lceil \log n \rceil$, one of the G_i produces a weighted disjoint flow whose value is at least a logarithmic fraction of the optimal flow value. As we can easily solve the maximum disjoint paths problem in G_i in polynomial time, this gives the claimed $O(\log n)$ approximation algorithm. □

Corollary 5.11. *There is an $O(\log k)$ approximation algorithm for the problem of finding a maximum flow that decomposes into at most k edge-disjoint paths.*

Proof. This previous argument applies. The approximation guarantee, however, improves to $O(\log k)$ because now the paths of weight at most 2^j can only contribute a total value of at most $k2^j$. □

Chapter 6

Constructing Subgraphs without Star-cutsets

Given the interpretation of galaxy cutsets as malicious virus-like attacks on a network, a natural question that arises is whether it is possible to design networks that are not vulnerable to such attacks. We therefore turn our attention to the problem of finding a spanning subgraph without galaxy cutsets in a given graph. The full input graph G represents the potential network, i.e. it specifies the feasible edges, and the spanning subgraph H represents the actual network that is eventually built.

Recall from Chapter 4 that the problem of determining if a given graph has a galaxy cutset of order k is NP-complete, even if all the stars have radius 1. This implies that there is little hope of being able to find a spanning subgraph without galaxy cutsets efficiently. Indeed, if P $\neq$ NP, then the problem is not even in NP, as we cannot determine in polynomial time if a given candidate solution is a YES- or a NO-instance. We therefore restrict ourselves to the more tractable special case where the order of the galaxy cutsets is fixed to be $k = 1$, i.e. we wish to find a spanning subgraph that does not contain any star-cutsets of radius r.

As in Chapter 4, we have a negative hardness result and a positive algorithmic result. The hardness result asserts that if $r \geq 4$ it is NP-complete to decide if a given graph has any spanning subgraph without star-cutsets of radius r.

In light of such a strong negative result, we turn to bi-criteria approaches. In Section 6.2 we exhibit a polynomial time approximation algorithm that finds a spanning subgraph without star-cutsets in a graph that has no star-cutsets of radius 3. The number of edges of the spanning subgraph found is at most $\frac{11}{6}$ times the number of edges of an optimal spanning subgraph.

6.1 The hardness of finding subgraphs without star-cutsets of radius r

By the results of Chapter 4, we can test in polynomial time whether a given candidate subgraph of G has a star-cutset of radius r or not, and hence the decision problem is in NP. We will show that the problem of deciding whether G contains a spanning subgraph H without star-cutsets of radius r is NP-hard by reducing 3SAT to it. Given a 3SAT instance with n Boolean variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$, we assume without loss of generality that for each x_i, both x_i and $\bar{x}_i$ occur in at least 2 clauses.

The graph G we use for the reduction has the following properties. G itself has star-cutsets of radius r, as do spanning subgraphs that do not correspond to valid true-false assignments of the Boolean variables. Moreover, any spanning subgraph that does correspond to a valid assignment will have a star-cutset of radius r if and only if the formula has no 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_i, and a variable gadget g_i. Also,

for each clause C_j there is a vertex C_j and a vertex s_j. The vertices v_i will ensure consistency: If a spanning subgraph H of G does not correspond to a valid true-false assignment, some v_i will be the centre of a star-cutset of radius r. Moreover, given a subgraph H that does correspond to a true-false assignment, the vertex s_j will be the centre of a star-cutset of radius r if and only if clause C_j is not satisfied under this assignment. Consequently, the vertices s_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 resembles a complete bipartite graph, where vertices on opposite sides of the partition are connected via paths. We have a vertex labelled a_{ij} 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 b_{ik} if $\bar{x}_i$ appears in C_k; these vertices make up the right side of the bipartition of g_i. We connect every vertex a_{ij} to every vertex b_{ik} via disjoint paths of length $4r$ and label the middle vertex of such a path $z_i^{j,k}$.

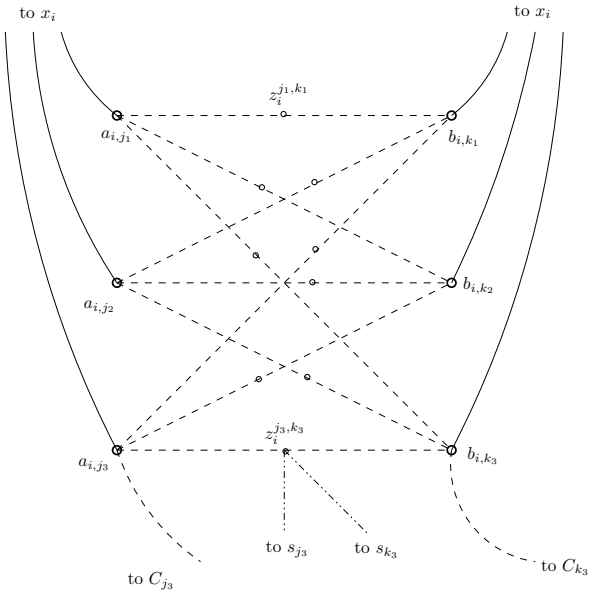

Figure 6.1: A variable gadget

We now describe how the vertices and gadgets are connected to one another. By a *long* path we 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_i via disjoint long paths. Next, for every i, v_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 a_{ij} and b_{ik} 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 s_k via disjoint long paths. Each vertex s_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 a_{ik}; if the clause C_k contains the literal $\bar{x}_i$ then vertex C_k is connected to the vertex b_{ik}.

The intuition behind these long paths is that, for most choices of vertex $v \in G$, any star of radius 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 star-cutset of radius r straightforward.

To complete the construction, recall that inside every variable gadget we have paths of length $4r$ from every a_{ij} to every b_{ik}. The middle vertex $z_i^{j,k}$ on such a path is now connected to s_j and s_k via paths of length r. The vertices in the middle of these two paths will be labelled $[z_i^{j,k}, s_j]$ and $[z_i^{j,k}, s_k]$, respectively; we connect these two vertices to v_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 6.2. In the figure, solid lines represent long paths, dashed lines represent paths of length $r-1$, dash-dotted lines represent paths of length r, and single edges are drawn with fat lines.

We now proceed to show that finding a spanning subgraph H of G with-

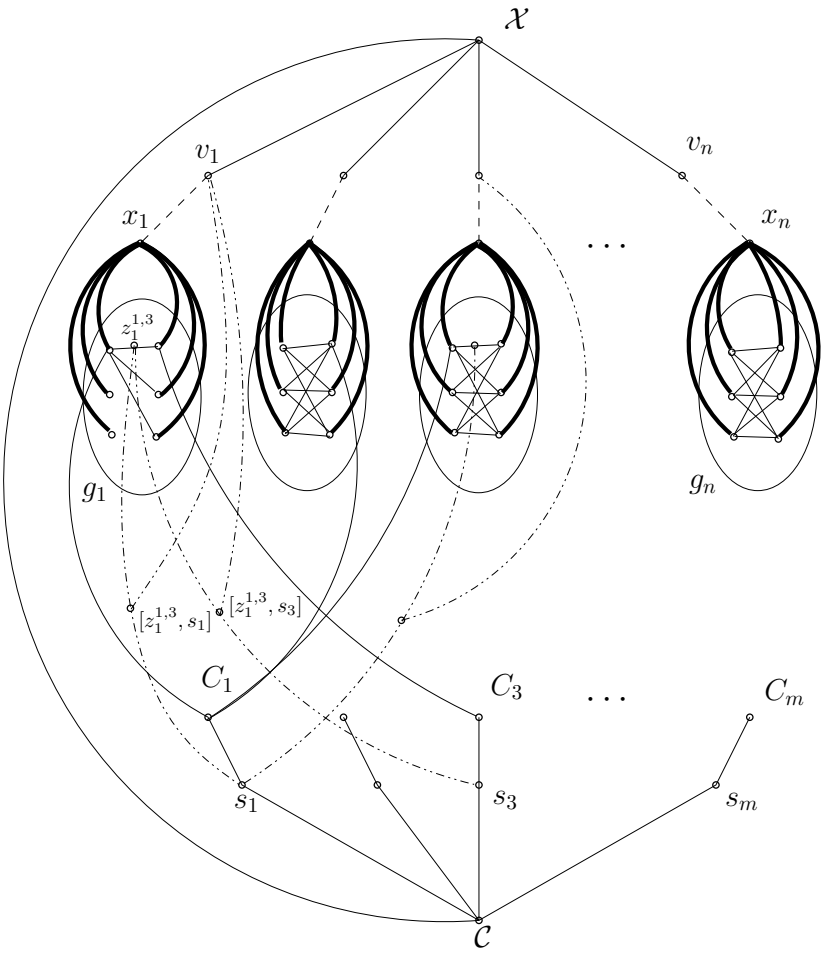

Figure 6.2: Reduction graph for the 3SAT reduction

out star-cutsets of radius r 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 has star-cutsets of radius r, for instance the tree of depth r centred at v_1, which separates the vertices on the long internal paths of g_1 from $\mathcal{X}$.

For each gadget g_i, we call the vertices a_{ij} *left* vertices and the vertices b_{ik} *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 6.1. *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 has a star-cutset of radius r.*

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 3 or more are joined by paths of length at least $r/2 \geq 2$, 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. Denote by w the neighbour of x which is not y. Then, if (x, y) is not in H, we can separate x from $\mathcal{X}$ simply by removing its other neighbour w, so we have a star-cutset of radius 0.

Note that this proof relied on the assumption that $r \geq 4$, without which we could not have deduced that one of the endpoints of $e = (x, y)$ had to have degree 2. □

Hence any candidate for a suitable spanning subgraph must contain all of the edges of G, except possibly the edges labelled left and right. Next, we show that any candidate for a spanning subgraph without radius r star-

cutsets 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.

Lemma 6.2. *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 has a star-cutset of radius r.*

Proof. Suppose H contains a left edge from x_i to a_{ij} and a right edge to b_{ik}, and let T be the tree of depth r rooted at v_i. Any path from $z_i^{j,k}$ (the middle vertex of the path from a_{ij} to b_{ik}) to $\mathcal{X}$ has to go through one of the vertices $[z_i^{j,k}, s_j]$, $[z_i^{j,k}, s_k]$, a_{ij}, or b_{ik}. But all of these vertices are in T, so $z_i^{j,k}$ and $\mathcal{X}$ are in different components in $G - T$. □

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 6.3. *Let H be a spanning subgraph of G that has no star-cutsets of radius r. If, for some i, H contains a left edge (respectively, right edge) from x_i to its gadget g_i, then the graph H' obtained by adding all the remaining left edges (respectively, right edges) from x_i to g_i does not have a radius r star-cutset either.*

Proof. It is sufficient to show that if H is a spanning subgraph without radius r star-cutsets containing a left edge from some x_i to some a_{ij}, then adding another left edge from x_i does not create a star-cutset of radius 3. So suppose that x_i is contained in clauses C_j and C_k, and that H contains the left edge (x_i, a_{ij}) but not (x_i, a_{ik}). Adding the edge $e = (x_i, a_{ik})$ can only turn vertices at distance at most $r-1$ from either endpoint of e into centres of star-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 a_{ik} can become the centre of a star-cutset when we add the edge $e = (x_i, a_{ik})$.

Let v be a vertex at distance at most r from a_{ik}, and T a star of radius r centred at v. Observe that we cannot disrupt the paths from the right vertices b_{il} of the gadget to their corresponding clause vertex C_l, because H cannot contain any right edges. We cannot disrupt the path from $\mathcal{X}$ to $\mathcal{C}$ either, nor the paths from C_j or C_k to $\mathcal{C}$ via s_j or s_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 a_{ik}, will still be able to reach $\mathcal{C}$ through either $\mathcal{X}$, s_j, s_k, or some right vertex of the gadget corresponding to x_i. □

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_i.

Now we can set up the correspondence between true-false assignments to the variables x_i and potential 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 the left edges or all the 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 6.4. *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_i and s_k can be centres of star-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 star-cutsets of radius r, because these vertices only have long paths ema-

nating from them. Also observe that only the vertices v_i are within distance r of both left or right vertices of a gadget and a middle vertex $[z_i^{j,k}, s_k]$, and only the vertices s_k are within distance r of more than one gadget. Showing that all internal vertices, and also the vertices x_i, a_{ij}, or b_{ik} cannot be centres of star-cutsets of radius at most r is straightforward. The only non-trivial cases are the vertices $z_i^{j,k}$ and $[z_i^{j,k}, s_j]$. Removing a star-cutset of radius at most r centred at $z_i^{j,k}$ does not affect a_{ij} or b_{ik}, but it does affect s_j and s_k. However, both C_j and C_k are still connected to $\mathcal{X}$ and hence $\mathcal{C}$ through other gadgets. Similarly, removing a star-cutset of radius at most r centred at $[z_i^{j,k}, s_j]$ may take out $z_i^{j,k}$, v_i and s_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 $g_{i'}$ with $i' \neq i$ it is connected to. □

The NP-hardness of finding a spanning subgraph H without star-cutsets of radius r is now established by the following theorem.

Theorem 6.5. *There exists a satisfying true-false assignment if and only if G contains a spanning subgraph H without star-cutsets of radius r.*

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 s_k is a cutset in the graph H corresponding to the assignment. Since C_k is not satisfied, we have that $x_{k_1} = x_{k_2} = x_{k_3} = \text{false}$ and so none of the left edges to the gadgets g_{k_1}, g_{k_2} and g_{k_3} are present. So any path from C_k to $\mathcal{X}$ must go through either s_k or through the middle vertices on the paths from a_{k_i,C_k} to the other side of the gadget g_{k_i} for $i = 1, 2, 3$. But all of those vertices are contained in T, and 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 has no star-cutsets of radius r. None of the vertices

v_i can be the centre of such a cutset, 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 $b_{i,k}$ to C_k cannot be interrupted, and similarly if the right edges are chosen. Moreover, none of the vertices s_k can be the centre of a star-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 s_k. This will disrupt the long paths from $a_{k_1,k}, a_{k_2,k}$, and $a_{k_3,k}$ to all the vertices on the other side of the corresponding gadget. However, we must have at least one of the three edges $(x_{k_1}, a_{k_1,k}), (x_{k_2}, a_{k_2,k})$ and $(x_{k_3}, a_{k_3,k})$ and thus we maintain connectivity. By Lemma 6.4, none of the other vertices can be centres of star-cutsets of radius r, so the result follows. □

Thus, we obtain our hardness result.

Theorem 6.6. *The problem of deciding whether a graph G contains a spanning subgraph H with no star-cutsets of radius r is NP-complete for $r \geq 4$.*

6.2 A bi-criteria approach to finding spanning subgraphs without star-cutsets

As we have seen, it is unlikely that we can obtain approximation algorithms for the problem of finding a minimum spanning subgraph without star-cutsets of radius r when $r \geq 4$. 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 star-cutset of radius at most three, we give a polynomial time algorithm to find an approximately minimum spanning subgraph containing no star-cutsets.

Theorem 6.7. *Let G be a graph that does not contain a star-cutset of radius 3. Then there is a factor $\frac{11}{6}$-approximation algorithm for the minimum star-cutset free spanning subgraph problem in G.*

Our approach to proving Theorem 6.7 is to iteratively add vertices and edges to a subgraph H of G, maintaining the property that H has no star-cutsets. The main difficulty is that this property is non-monotonic, as can be seen immediately by considering the example of a cycle to which a chord is added. This difficulty does not arise in the design of k-vertex or k-edge connected spanning subgraph, where adding edges can only increase the connectivity.

The assumption that G does not contain star-cutsets of radius 3 implies that G is 2-vertex connected. We will use this fact repeatedly. We start constructing our subgraph with a cycle $H = \mathcal{E}_0$ 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 the property of being star-cutset free. We remark that ear decompositions have been used in designing efficient algorithms for network design problems with low connectivity requirements, for example see [12] and [34]. The first phase will terminate when $V - V(H)$ is a stable set $\mathcal{S}$. In the second phase we will iteratively add the vertices of $\mathcal{S}$ along with carefully chosen edges to H, again avoiding the creation of new star-cutsets.

6.2.1 Phase I: Ear growing

First we define formally what is meant by an ear. Given a graph G and a subgraph H, an *ear* $\mathcal{E}$ is a path whose endpoints are in H and whose internal vertices are in $V - H$. By a *long* ear we mean an ear with at least 3 edges. We call an ear $\mathcal{E}$ *locally maximum* if no edge of the ear can be replaced by a path of length at least 2 all of whose internal vertices lie in $V - (H \cup \mathcal{E})$, and

if no pair of adjacent edges of the ear can be replaced by a path of length at least 3 with internal vertices in $V - (H \cup \mathcal{E})$. The definition of locally maximum extends in an obvious way to the initial cycle $\mathcal{E}_0$.

In iteration i, an ear $\mathcal{E}_i$ is added to the current subgraph H_i to obtain the subgraph H_{i+1}. The actual choice of ear $\mathcal{E}_i$ is made according to the following rules.

(1) $\mathcal{E}_i$ contains at least 3 edges, i.e. it is a long ear, and its endpoints u_i and v_i satisfy $d_{H_i}(u_i, v_i) \geq 3$.

(2) Given Rule (1), the ear $\mathcal{E}_i$ is chosen to be locally maximum.

We now show that the procedure described above is well-defined and terminates when $V - H_i$ is a stable set.

Claim 6.8. *While $V - H_i$ is not a stable set, there is a long ear $\mathcal{E}_i$ satisfying $d_{H_i}(u_i, v_i) \geq 3$.*

Proof. Let x and y be vertices of $V - H_i$ such that $e = (x, y)$ is an edge. Since G is 2-connected, there must be 2 vertex-disjoint paths from $\{x, y\}$ to H_i. It follows that there exists a long ear. Let $\mathcal{E}_i$ be a long ear with endpoints u_i, v_i, and suppose that $d_{H_i}(u_i, v_i) \leq 2$.

If $d_{H_i}(u_i, v_i) = 1$, then $e = (u_i, v_i)$ was chosen in some ear $\mathcal{E}_j$ where $j < i$. But then $\mathcal{E}_j - e + \mathcal{E}_i$ could have been chosen instead of $\mathcal{E}_j$, contradicting Rule (2).

Next, if $d_{H_i}(u_i, v_i) = 2$, 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 $\mathcal{E}_j$ where $j < i$ then $\mathcal{E}_j - e_1 - e_2 + \mathcal{E}_i$ could have been chosen as a larger ear instead of $\mathcal{E}_j$, contradicting Rule (2). Thus, $e_1 \in \mathcal{E}_k$ and $e_2 \in \mathcal{E}_j$ where $k < j < i$. It follows that $v = v_j$ is an endpoint of the ear $\mathcal{E}_j$. Thus,

v_i is an internal vertex of $\mathcal{E}_j$ and so $v_i \notin H_j$. Consequently, $v_i \notin H_k$ and $\mathcal{E}_k - (u_i, v) + (v, v_i) + \mathcal{E}_i$ contradicts the choice of $\mathcal{E}_k$.

Hence the endpoints of every long ear must satisfy either $d_{H_i}(u_i, v_i) \geq 3$ or $d_{H_i}(u_i, v_i) = 0$. But if $d_{H_i}(u_i, v_i) = 0$ for every long ear, then clearly G contains a cut-vertex, leading again to a contradiction. □

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

Claim 6.9. *If H_i has no star-cutset and $d_{H_i}(u_i, v_i) \geq 3$, then H_{i+1} has no star-cutset.*

Proof. Let v be an internal vertex in $\mathcal{E}_i$. 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, since we showed that there is always an ear with at least three edges in Phase I.

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 $\mathcal{E}_i - S_v$ 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 $\mathcal{E}_i$ 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. □

Since the algorithm starts with a cycle $\mathcal{E}_0$ and cycles do not contain star-cutsets, it follows that the subgraph H we have when Phase I terminates cannot contain a star-cutset either.

We remark that in Phase I we use only the fact that G contains no cut-vertices, which follows from the assumption that G has no star-cutsets of radius at most 3.

6.2.2 Phase II: Incorporating the stable set

Let H be the subgraph obtained after Phase I, and let $\mathcal{S} = V - V(H)$ be the stable set of remaining vertices. In the second phase we iteratively add the vertices of $\mathcal{S}$ to H, again taking care not to create any star-cutsets. In fact, we can add always a vertex to H that has two neighbours whose distance in H from one another is at least 4.

Lemma 6.10. *Let G be a graph without star-cutsets of radius 3, let H be a connected subgraph of G, and let $v \in V - V(H)$ be a vertex all of whose neighbours are in H. Then either v has two neighbours a and b such that $d_H(a, b) \geq 4$, or G is Hamiltonian.*

Proof. As G has no cut-vertices we know that v has at least two neighbours in H. Let u be a neighbour of v and suppose $d_H(u, w) \leq 3$ for all $w \in \Gamma(v)$. So there is a tree Z of depth 3 in H, centred at u, containing every vertex of $\Gamma(v)$. We have two possibilities:
(i) There is a vertex $x \neq v$ that is not in Z, in which case we have a tree of depth 3 whose removal from G separates x and v, a contradiction.
(ii) $\Gamma(v)$ (and thus Z) contains every vertex in $V - \{v\}$. Let $D_i = \{w : d_H(u, w) = i\}$, for $i = 1, 2, 3$. We will show that D_i must be a clique for $i = 1, 2, 3$.

If D_3 is not a clique, 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 star-cutset of radius at most 3.

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, we could separate two non-adjacent vertices x and y of D_2 by removing $V - \{x, y\}$, a tree of depth at most 3.

But if D_2 is a clique, 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. □

The graph H obtained after Phase I is connected, and $\mathcal{S} = V - H$ is a stable set, so we have $\Gamma(v) \subseteq H$ for every $v \in \mathcal{S}$. Hence, Lemma 6.10 applies, and so when we consider $v \in \mathcal{S}$ 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. This yields a subgraph H' which is again connected, and clearly $\mathcal{S}' := \mathcal{S} - \{v\}$ is still a stable set, so the conditions of Lemma 6.10 will be satisfied in all subsequent iterations as well.

If we don't find G to be Hamiltonian and we add edges (v, a) and (v, b), there are two possibilities: Either the set $\{a, b\}$ forms a cutset in H, or it does not.

In the latter case things are simple. We just set H' to be the graph with vertex set $V(H') = V(H) \cup \{v\}$ and edge set $E(H') = E(H) \cup \{(v, a), (v, b)\}$. Since $\{a, b\}$ is not a cut in H, we have that $\{v, a, b\}$ is not a star-cutset in H'. Moreover, if H' contained a star-cutset $S \neq \{v, a, b\}$, this would show that H contained a star-cutset, too, which is a contradiction.

has no star-cutsets so we may then iterate on H' and $\mathcal{S}' := \mathcal{S} - \{v\}$.

If $\{a, b\}$ is a cutset, adding the edges (v, a) and (v, b) creates a star-cutset, which we must neutralise by adding additional edges. 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 distance to $\{a, b\}$ is exactly r, and label by C_i^{r+} the vertices of C_i whose distance to $\{a, b\}$ is at least r.

It is clear that each G_i is again 2-vertex connected, for a cutvertex in G_i would still also be a cutvertex in G.

Claim 6.11. *C_i^{2+} is non-empty for all $1 \leq i \leq k$.*

Proof. Suppose C_i^{2+} is empty for some $1 \leq i \leq k$, 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 contradicts Lemma 6.10. □

In order to neutralise the star-cutset $\{v, a, b\}$, we will consider the following Steiner tree problem. Consider the auxiliary graph $\mathcal{G}$ obtained by creating a terminal node t_i for each set of vertices C_i^{2+}, and a Steiner node s_u for each vertex $u \in \mathcal{S} - \{v\}$. Two nodes in $\mathcal{G}$ are adjacent if there is at least one edge between their corresponding vertex sets in G.

Claim 6.12. *There is a Steiner tree $\mathcal{T}$ 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 has no star-cutsets of radius 3, this tree of depth 2 is not a cutset, which implies the claim. □

Let H'' be the graph obtained from H' by adding the Steiner tree $\mathcal{T}$, i.e. we add the Steiner nodes used by $\mathcal{T}$, and, for each edge between nodes of $\mathcal{T}$, we add an edge between the corresponding vertex sets in G.

Lemma 6.13. *If $\mathcal{T}$ is a Steiner tree in $\mathcal{G}$ then H'' has no star-cutsets.*

Proof. Clearly v is not the centre of a star-cutset in H''. Observe then that a star-cutset in H'' must be centred at a vertex incident to an edge in $\mathcal{T}$. To show that such a cutset cannot exist we will need the fact that G_i is 2-vertex connected. Let s be a Steiner node in $\mathcal{T}$. 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, let t be a vertex in a set C_j^{2+} corresponding to a terminal node . Observe that t does not centre a star-cutset in G_j, otherwise it gave a star-cutset in H. Now t is adjacent to at most one vertex in $G_i, i \neq j$. Hence, as before the removal of the neighbours of t cannot disconnect any of the other

graphs G_i, and moreover all these graphs remain connected to one another via $\{v, a, b\}$. □

After adding the Steiner tree to H', we update H and $\mathcal{S}$ and iterate. Clearly both Phase I and Phase II can be implemented in polynomial time so it remains to analyse the performance guarantee of the algorithm. This will require a fairly intricate charging argument, which we detail in the remainder of this section.

Proof of Theorem 6.7. 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 (v, a) and (v, b) to $v \in \mathcal{S}$. We charge two edges of T to each Steiner node in T (recall that the Steiner nodes also belong to the current stable set $\mathcal{S}$). 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. We will charge these edges to the initial cycle and the ears from Phase I as follows:

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 6.14. *For $i \geq 3$, the vertices a and b are the endpoints of an ear $\mathcal{E}_i$, added in Phase I through C_i. Moreover, 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 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. □

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

Now we charge one edge of $\mathcal{T}$ to each such ear $\mathcal{E}_i$, $i \geq 3$. 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 $\mathcal{T}$ to C^* or E_2.

Claim 6.15. *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 idea of the proof is simple: We shall bound from above, for each ear and for the initial cycle, the number of times that a pair $\{a, b\}$ which induces a cut in Phase II (and thus requires augmenting the current subgraph) can occur on the cycle or the ear.

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

In conditions (ii) and (iii), the distance is taken to be the distance on the cycle or path. Figure 6.3(a) shows a path with a non-overlapping family (the vertices constituting a pair are joined by a dotted line), and in Figure 6.3(b), the pairs (u,v) and (u,w) fail to satisfy condition (iii) and (s,t) overlaps (i,j).

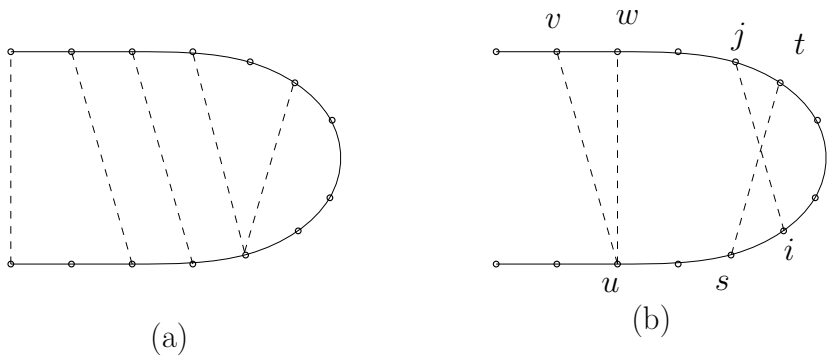

Figure 6.3: Non-overlapping and overlapping families

Lemma 6.16. *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 $v \in \mathcal{S}$ 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.

The argument for an ear $\mathcal{E}$ is similar. We charge an edge to $\mathcal{E}$ if a and b are the endpoints of $\mathcal{E}$ and $\mathcal{E}$ 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 $\mathcal{S}$. We also charge an edge to $\mathcal{E}$ if $\mathcal{E}$ 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 $\mathcal{E}$ 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 $\mathcal{E}$ being charged form a non-overlapping family for $\mathcal{E}$. □

Having established the correspondence between non-overlapping families and the number of edges charged to an ear or cycle, it remains to bound from above the size of a non-overlapping family.

Lemma 6.17. *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.*

Proof. We begin by proving the assertion of the lemma for paths, and then deduce it for cycles. We use induction on the number of internal vertices. Let P be a path with endvertices s and t, and let $\mathcal{F}$ be a non-overlapping family of maximum size for this path. 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 are 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 maximise $d(s,u)$, 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 minimises $d(u,v)$. 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. □

This concludes the proof of Claim 6.15. The remaining analysis that shows the approximation guarantee to be $\frac{11}{6}$ is straightforward. Let $n_{\mathcal{E}}$ denote the number of internal vertices of an ear $\mathcal{E}$, so $\mathcal{E}$ consists of $n_{\mathcal{E}}+1$ edges. By Claim 6.15, the ear $\mathcal{E}$ is charged at most $\frac{n_{\mathcal{E}}-1}{2}$ edges during Phase II. So for an ear $\mathcal{E}$ with at least 3 internal vertices, the sum of the edges belonging to $\mathcal{E}$ and the edges charged to $\mathcal{E}$ is at most $n_{\mathcal{E}}+1+\frac{n_{\mathcal{E}}-1}{2} = \frac{3}{2}n_{\mathcal{E}}+\frac{1}{2}$. If $\mathcal{E}$ has only 2 internal vertices it is not charged anything in Phase II, and has $n_{\mathcal{E}}+1 = 3 = \frac{3}{2}n_{\mathcal{E}}$ edges. The initial cycle C^* contains $|C^*|$ edges and is charged at most $\frac{|C^*|-6}{2}$ edges in Phase II. Hence, together with the edges

charged to the vertices of the stable set $\mathcal{S}$, we have a total of

$$|C^*| + \frac{|C^*| - 6}{2} + \sum_1 \frac{3}{2} n_{\mathcal{E}} + \sum_2 \frac{1}{2} + 2|\mathcal{S}|$$

edges, where the first sum $\sum_1$ is taken over all ears added in Phase I, and the second sum $\sum_2$ is taken over all ears with at least 3 internal vertices. This is at most

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

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

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

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

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

Chapter 7

Neutralising Galaxy Cutsets

In this chapter, we will consider the problem of neutralising galaxy cutsets in graphs. This problem is similar in spirit to the issue of designing resilient graphs as discussed in Chapter 6, but the approach is different in that instead of constructing a graph, we assume that it is given and that we have to fix existing vulnerabilities. Hence, it falls into the domain of optimisation on networks rather than network design. We will restrict ourselves to the case where both the order of the galaxy and the radius of the stars in it are equal to 1. In other words, we are concerned with star-cutsets according to the standard definition.

We begin by formally defining our task as a hitting set problem, after which we provide a more intuitive explanation justifying why this is a sensible approach. Recall that if S is a star of radius 1 with centre v, we call the set of edges $R(S) := \{(v, x) \,|\, x \in S, x \neq v\}$ emanating from the centre the *rays* of the star. The hitting set problem we wish to solve is the following:

> `The Network Protection Augmentation 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 $R(S)$ for every star-cutset S.

We say that a star-cutset S is *vulnerable* with respect to F if $R(S) \cap F = \emptyset$. An alternative formulation of the Network Protection Augmentation Problem then is: Given G and $F \subset E(G)$, find the smallest set A such that G has no star-cutsets which are vulnerable with respect to $A \cup F$.

As briefly touched upon in the introduction, our motivation for considering this problem is the following. Suppose we can prevent a hypothetical virus from propagating along an edge $e = (u, v)$ by *protecting* that edge in some way, incurring a (unit) cost for each edge we choose to protect. Under this assumption, finding a set A such that $A \cup F$ intersects $R(S)$ for every star-cutset S neutralises every star-cutset, in that there will be no more star-cutsets containing only unprotected edges (i.e. edges along which the virus can travel) as rays.

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 additional edges we need to protect 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.

In some practical applications, it might be more natural to protect vertices instead of edges, and we will give this variant some consideration. Unfortunately, it turns out to be NP-hard even in planar graphs. On the other hand, while the edge variant is NP-hard in non-planar graphs, we will see that in planar graphs minimal star-cutsets have a very interesting structure which allows us to solve the Network Protection Augmentation Problem in polynomial time using a recursive algorithm. We give a detailed analysis of this structure in Section 7.1, and present our algorithm in Section 7.2. Then, in Section 7.3, we give $\Theta(\log n)$ lower and upper approximation bounds for the Network Protection Augmentation Problem in general graphs; here n is

the number of vertices. The hardness proof for the vertex case can be found in Section 7.2.6.

Before launching into the investigation of the structure of minimal star-cutsets in planar graphs, we give a brief high-level description of the algorithm. We attempt to find a star-cutset that allows us to define subproblems on certain minors of the input graph. We then call the algorithm recursively on those minors, and combine the solutions to the subproblems to obtain a solution for the original problem. The main structural theorem asserts that if no suitable star-cutset can be found, then the input graph must have a very special form which allows us to solve the problem directly in polynomial time.

7.1 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 algorithm in Section 7.2. Before presenting this structural result, though, we examine how large *minimal*[1] 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.

7.1.1 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 either two or three vertices. We begin with some simple observations. Suppose that G contains a cut-vertex v.

[1] 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 $X \subset \{x_1, \ldots, x_k\}$.

Thus, there is a star-cutset centred at v containing no rays; consequently, protecting edges cannot neutralise 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 7.1. *G is 2-connected.*

Next, we use the planarity of G to bound the size of a minimal star-cutset from above.

Lemma 7.2. *Every minimal star-cutset contains at most 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 shows G to contain a $K_{3,k}$ minor. By planarity, this implies that $k \leq 2$ for any minimal star-cutset. □

We call a star-cutset $S = \{u, v\}$ consisting of exactly two vertices that induce an edge a *separating-edge*. Either u or v may be thought of as the centre, and there is a single ray (u, v). For the purposes of our algorithm, separating-edges will be particularly easy to deal with. This is because separating-edges let us divide the network protection problem into subproblems in a natural way, and proving that optimal solutions to those subproblems can be combined to give an optimal global solution is straightforward, as we shall show in Section 7.2.3.

A minimal star-cutset $S = \{v, x, y\}$ such that $(v, x), (v, y)$ and (x, y) are edges of G will be called a *triangle cutset*. Note that at least two of the three edges of a triangle cutset must be protected, as any one of the three vertices may serve as the centre.

We remark that Lemma 7.2 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.

Henceforth, we will assume that all star-cutsets we encounter are minimal, without mentioning it explicitly in every theorem, lemma or claim, except when we wish to emphasise the minimality.

7.1.2 The Structural Theorem

To state the structural theorem we need several definitions. Let S and S' be distinct minimal star-cutsets. Then S' *crosses* S if S' is centred at a vertex of S and contains vertices from two distinct components of $G - S$; we say that S is *uncrossed* if no such S' exists. A *wheel graph* consists of an induced cycle (i.e. there are no chords), together with an additional *hub vertex* adjacent to all the vertices in the cycle. The edges from the hub vertex to vertices in the cycle are called *spokes*. A *padded wheel* is obtained from a wheel graph by performing one of the following two operations on some of the cycle edges:

(O1) Adding one path of length 2 between adjacent vertices of the cycle. In this case, the internal vertex on the new path must be adjacent to the hub of the wheel.

(O2) Deleting an edge (u, v) of the cycle and adding internally vertex-disjoint paths of length 2 between u and v. There must be at least one such path, and at most one of the internal vertices on the new paths may have an edge to the hub.

Operations (O1) and (O2) may both be applied in the same graph, but not on the same edge. Observe that a wheel graph is clearly planar, and that the conditions on the length 2 paths we add in operations (O1) and

(O2) preserve the planarity. Hence, a padded wheel is also a planar graph. Figure 7.1 shows both a wheel and a padded wheel.

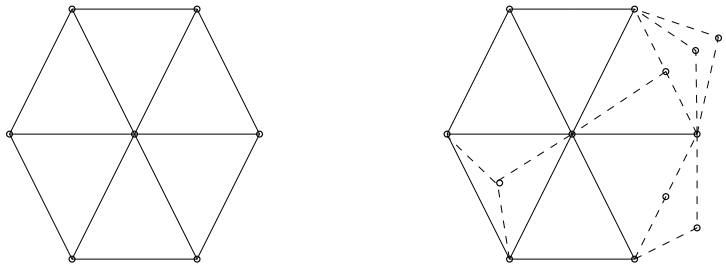

Figure 7.1: A wheel and a padded wheel

We can now state our structural theorem, which is the main result of this section.

Theorem 7.3. *Let $G = (V, E)$ be a 2-vertex connected planar graph, and let $F \subseteq E$. Then one of the following must be true:*

(P1) G has no vulnerable star-cutsets.

(P2) G contains a separating-edge $\{u, v\}$ (not necessarily vulnerable).

(P3) For every vulnerable star-cutset S, at most one of the components of $G - S$ is non-trivial [2].

(P4) G contains an uncrossed vulnerable star-cutset S such that at least two components of $G - S$ are non-trivial.

(P5) G is a padded wheel.

In Case (P1), there is nothing to do. We simply return $\emptyset$ as an optimal extension of F. Case (P2) will also turn out to be rather simple from an algorithmic point of view, so for the remainder of the section we will make the following assumption:

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

Assumption 7.4. *Every minimal star-cutset contains at least* 3 *vertices.*

Thus, by Lemma 7.2, we may actually assume that every star-cutset contains exactly three vertices. Now to prove Theorem 7.3 we need to delve deeper into how minimal star-cutsets can relate to each other. We begin with a straightforward observation.

Observation 7.5. *Let $S = \{v, x, y\}$ be a star-cutset centred at v, and let C be a component of $G - S$. 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 (respectively y) into C, then (v, y) (respectively (v, x)) would be a separating-edge, contradicting both the minimality of S and the assumption that there are no separating-edges. □

Next we show that star-cutsets can cross only in restricted circumstances.

Theorem 7.6. *Let $S = \{v, x, y\}$ be a vulnerable star-cutset centred at v, and suppose T is a vulnerable star-cutset that crosses S. Then all of the following hold:*

(i) T is centred at v.

(ii) S crosses T.

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

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

In the latter case we may assume that $T = \{x, a, b\}$ where a and b lie in distinct components C respectively D of $G - S$. Let $u \in G - T$. We claim that there must be a path from u to y in $G - T$. If u is v or y, this is trivially true. 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 $G - T$ by Observation 7.5. So, without loss of generality, assume that u lies in C. If $T = \{x, a, b\}$ intersects every path from u to y, then the set $\{x, a\}$ must also intersect every $u - y$ path, as there is no path from u to $\{v, y\}$ in $G - T$. But there is an edge between x and a and so, since T is a vulnerable star-cutset, (x, a) is a vulnerable separating-edge, contradicting both Assumption 7.4 and the assumption that T is minimal. Hence, every vertex $u \in G - T$ is connected to y. But this contradicts the assumption that T is a cutset, and thus T must also be centred at v, i.e. (i) holds.

For the former case, let $T = \{v, a, b\}$ with $a \in C, b \in D$, and again let $u \in G - T$. We claim that there is either a path from u to x or a path from u to y in $G - T$. 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 a path from u to x and a path from u to y in $G - T$. So assume, without loss of generality, that $u \in C$. If $T = \{v, a, b\}$ intersects every path from u to x and every path from u to y, then the set $\{v, a\}$ must also intersect every $u - x$ path and every $u - y$ path, since S separates u from b. But then $\{v, a\}$ would be a separating-edge. Hence, every vertex is connected to either x or y in $G - T$. It follows that x and y must be in distinct components of $G - T$, otherwise $G - T$ would be connected. However, if x and y are in distinct components of $G - T$, then S crosses T. 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 $G - T$. □

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

It will be useful to partition the components of a 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. If C has no edge to

the centre v, we say that it is *weakly attached* to S.

A star-cutset cannot induce many strongly attached components:

Lemma 7.7. *Let $S = \{v, x, y\}$ be a star-cutset in G. Then $G - S$ has at most 2 components that are strongly attached to S.*

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

Lemma 7.8. *If $S = \{v, x, y\}$ is a vulnerable star-cutset centred at v such that $e = (x, y) \in E(G)$, 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 $e = (x, y) \in E$, then e is a separating-edge, contradicting Assumption 7.4. □

The following corollary of Theorem 7.6 will be applied repeatedly in the proof of our main structural result, Theorem 7.3.

Corollary 7.9. *Suppose $S = \{v, x, y\}$ is a vulnerable star-cutset and $T = \{v, a, b\}$ is a vulnerable star-cutset crossing S, with a and b in distinct components C respectively 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. Suppose there were a path P from x to y all of whose internal vertices lay in C and that P did not contain a. Then x and y would be in the same component of $G - T$, that is, S would not cross T. □

Let us now return to Theorem 7.3. It must be the case that there exists a vulnerable star-cutset S such that there are at least two non-trivial components in $G - S$, otherwise Case (P3) of 7.3 holds. If S is not crossed by any

other vulnerable star-cutset, (P4) holds. So suppose that every vulnerable S such that $G - S$ has at least two non-trivial components is crossed by at least one other vulnerable star-cutset. Our goal for the remainder of the section is to show that, in this case, G must be a padded wheel.

7.1.3 Discovering the Wheel

Let $S = \{v, x, y\}$ be a crossed, non-trivial vulnerable star-cutset with centre v, and consider the collection $\mathcal{S}$ of all the vulnerable star-cutsets that cross S. Observe that if S is crossed, then Theorem 7.6 implies that $G - S$ has exactly two components, say C and D. Let $A := \{a_1, \ldots, a_r\}$ be the vertices of C that occur in some star-cutset belonging to $\mathcal{S}$, and let $B := \{b_1, \ldots, b_t\}$ be the vertices of D that occur in some star-cutset belonging to $\mathcal{S}$. Note that we must have $r \geq 1$ and $t \geq 1$, since by assumption S is crossed by at least one other star-cutset. A path P from x to y all of whose internal vertices belong to C (respectively D) shall be more succinctly referred to as an $x - y$ path *through* C (respectively D).

Given an embedding of G in the plane, label the vertices $a_1, a_2, \ldots, a_r$ and $b_1, b_2, \ldots, b_t$ in clockwise order around v as shown in Figure 7.2. Claim 7.10 establishes formally that this ordering is justified. For notational convenience, we also define $a_0 = b_{t+1} := x$ and $a_{r+1} = b_0 := y$.

We briefly sketch an outline of the argument before getting started on the details. The vertices $S \cup A \cup B$ will provide the framework for the wheel. The difficulty then lies in proving that there cannot be much "between" those vertices. More precisely, we will show that for any i, a_i and a_{i+1} (respectively b_i and b_{i+1}) are joined either by an edge, or by paths of length 2, in a manner consistent with the definition of a padded wheel.

Claim 7.10. *There are no indices i, k, j with $i < k$ such that b_j lies between a_i and a_k in the clockwise ordering of the vertices $a_0, \ldots, a_{r+1}, b_0, \ldots, b_{t+1}$*

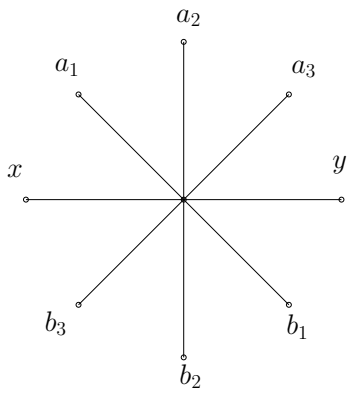

Figure 7.2: The wheel structure of the crossing star-cutsets

around v.

Proof. Suppose there are indices i, k and j as in the statement of the claim. Let P and Q be paths from x to y through C and D, respectively. From Corollary 7.9, we know that P must intersect both a_i and a_k, so let P' be the subpath of P between a_i and a_k. But now the cycle Z consisting of P' together with the edges va_i and va_k separates b_j from x and y; see Figure 7.3 (a). Since b_j must be contained in Q, it follows that Q must intersect Z. By planarity Q must then cross Z at a vertex, but 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 does not 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. □

Lemma 7.11. *For each pair a, b with $a \in A$ and $b \in B$, the set $S' = \{v, a, b\}$ is a star-cutset.*

Proof. This follows immediately from Corollary 7.9. There are only two components in $G - S$, namely C and D, and every $a \in A$ intersects all the $x-y$ paths through C, and every $b \in B$ intersects all the $x-y$ paths through D. Therefore there cannot be an $x - y$ path in $G - \{v, a, b\}$. □

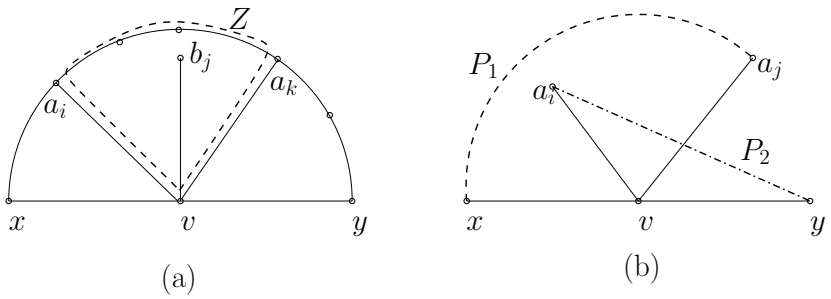

Figure 7.3: The cyclic ordering

Claim 7.12. *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 all internal vertices in C, and suppose that it visits a_j before a_i for some pair $i < j$. Note that we are assuming 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 Z consisting of P_1 together with the edges a_jv and (v, x). This cycle separates a_i and y and so, by planarity, P_2 must cross Z at some vertex. It cannot cross Z at x or v, since all of P's interior vertices lie in C. Furthermore, P_2 cannot cross Z at a vertex belonging to P_1, otherwise P would visit a vertex twice. This gives us the desired contradiction. Figure 7.3(b) illustrates the proof. An analogous proof works for the b_i. □

Claim 7.13. *Let $u \in C - A$. Then there exists a unique index i with $0 \leq i \leq r$ such that there is a path in $C - A$ from u to a_i and to a_{i+1}, but not to any other vertex of $A \cup \{x, y\}$. Similarly, for every $w \in D - B$ there exists a unique index j with $0 \leq j \leq t$ such that there is a path in $D - B$ from w to b_j and to b_{j+1}, but not to any other vertex of $B \cup \{x, y\}$.*

Proof. If there were no path from u to any vertex of $A \cup \{x, y\}$, then either G would be disconnected or the hub v would be a cut-vertex, in either case contradicting the assumption of 2-vertex connectivity. If there were a path from u to precisely one vertex $a_i \in A \cup \{x, y\}$, then either a_i would be a cut-vertex or (v, a_i) would be a separating-edge, again leading to a contradiction. Hence there must be paths in $C - A$ from u to at least 2 vertices a_i and a_j of $A \cup \{x, y\}$. Without loss of generality, we may assume that $i < j$. Now if $j \neq i+1$, then the existence of the paths from u to a_i and a_j imply that a_{i+1} misses an $x - y$ path, contradicting Corollary 7.9. Finally, if u had a third path to a_k with $k \notin \{i, i+1\}$, then either $|k - i| \geq 2$ or $|k - (i+1)| \geq 2$, and we could find a vertex a_j which did not intersect all the $x - y$ paths.

The proof for $w \in D - B$ is analogous. □

In the light of Claim 7.13, it makes sense to define $U_i := \{u \in C - A \mid u \text{ has a path to } a_i \text{ and } a_{i+1}\}$ for $0 \leq i \leq r$ and $W_j := \{w \in D - B \mid w \text{ has a path to } b_j \text{ and } b_{j+1}\}$ for $0 \leq j \leq t$. Then $V(G)$ is the disjoint union of $S \cup A \cup B \cup \bigcup_i U_i \bigcup_j W_j$.

Lemma 7.14. *For every i and k with $0 \leq i < k \leq r+1$, $\{v, a_i, a_k\}$ 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 for every i, k with $0 \leq i < k \leq t+1$ 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 since then $\{v, a_i, a_k\} = S$. Consider a path P from x to y all of whose internal vertices are in C. By Claim 7.12, P intersects a_i before it intersects a_k. Let $P^{i,k}$ be the set of vertices of P between a_i and a_k. If $P^{i,k}$ is non-empty, then $\{v, a_i, a_k\}$ is a star-cutset separating $P^{i,k}$ from the vertices in D. Otherwise, either a_i or a_k would miss some $x - y$ path through C. If $P^{i,k}$ is empty, then $e = (a_i, a_k)$ is an edge, and this edge is used in P. It follows that $\{v, a_i, a_k\}$ is a star-cutset

unless $P^{i,k}$ is empty for every path P, in which case (a_i, a_k) is an edge on every $x - y$ path through C. An analogous proof works for the b_i. □

Corollary 7.15. *If U_i is not empty, then $S_i := \{v, a_i, a_{i+1}\}$ is an uncrossed star-cutset, and if W_j is not empty, then $S_j := \{v, b_j, b_{j+1}\}$ is an uncrossed star-cutset.*

Proof. The fact that S_i disconnects U_i immediately follows from Claim 7.13, since the vertices of U_i can have no paths to vertices other than a_i, a_{i+1} and v in $C - A$. If there were a star-cutset T crossing S_i, then T would have to contain a vertex $u \in U_i$ intersecting every path from a_i to a_{i+1} in C. But then u would also intersect every $x - y$ path through C, implying that it should be in A, which it is not. The proof for S_j is analogous. □

With these results in hand, we can now prove Theorem 7.3.

Proof of Theorem 7.3. Recall that we may assume that G contains at least one non-trivial vulnerable star-cutset and that every non-trivial vulnerable star-cutset is crossed. Again we take such a star-cutset S along with the collection $\mathcal{S}$ of star-cutsets crossing S, and define A, B, the U_i and the W_j as before. We will now show that G must be a padded wheel.

For each i from 0 to r, consider the star $S_i = \{v, a_i, a_{i+1}\}$ centred at v. If $U_i = \emptyset$, then (a_i, a_{i+1}) must be an edge of G. If U_i is not empty, then S_i is an uncrossed star-cutset by Corollary 7.15. This implies that there can be no edges between vertices of U_i, for otherwise S_i would be an uncrossed star-cutset yielding at least two non-trivial components. Hence U_i must consist of vertices that are adjacent to a_i and a_{i+1}, and possibly to v, but to no other vertices. An analogous statement holds for $W_j, 0 \leq j \leq t$ by considering the stars $S_j = \{v, b_j, b_{j+1}\}$.

Claim 7.16. *At most one vertex of each U_i and W_j can be adjacent to v.*

Proof. Suppose we had two such vertices u_1 and u_2, and let Q be a path from a_i to a_{i+1} going first to x, then to y through D, and then from y to a_{i+1}. Let C_1 and C_2 be the two cycles obtained by adding u_1, respectively u_2, to Q. Then either C_1 separates v and u_2, or C_2 separates v and u_1. In either case we obtain a contradiction.

The proof for the W_j is analogous. □

This leaves three possibilities in the case where U_i is not empty. If $U_i \neq \emptyset$ and (a_i, a_{i+1}) is an edge of G, then every $u \in U_i$ must be adjacent to v, for otherwise (a_i, a_{i+1}) would be a separating-edge. Then Claim 7.16 implies that U_i must consist of a single vertex u_i.

If $|U_i| \neq \emptyset$ and (a_i, a_{i+1}) is not an edge, either no vertex of U_i is adjacent to v, or exactly one vertex of U_i is adjacent to v. In conclusion, for each i one of the cases depicted in Figure 7.4 must hold true, which implies that G is a padded wheel. □

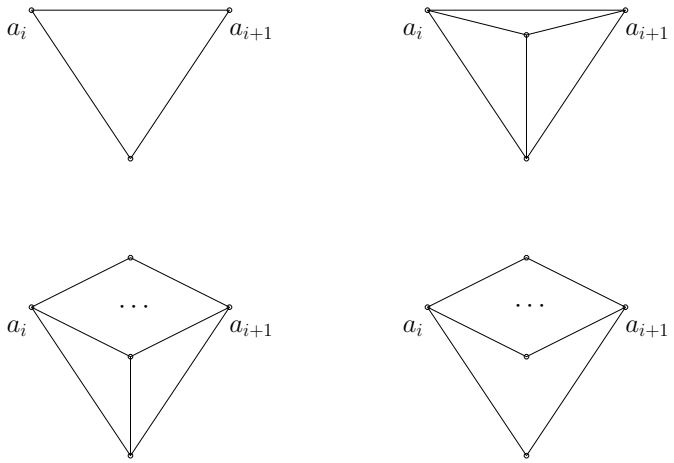

Figure 7.4: Allowed configurations in a padded wheel

This concludes the proof of Theorem 7.3, which is at the core of the recursive algorithm presented in Section 7.2.

7.2 A polynomial time algorithm

In this section we present our polynomial time algorithm for the Network Protection Augmentation Problem in planar graphs. The algorithm is inspired by our structural result, Theorem 7.3. A detailed description of the algorithm and the quantitative performance bound is provided in Sections 7.2.1 to 7.2.4. The algorithm is recursive and so, to complete the result, we show in Section 7.2.5 that the recursion can indeed 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 $|A|$ has minimum cardinality.

Again, all the star-cutsets will be assumed to be minimal. Also, 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 is not, then it returns that there is no extension of F and stops. So we may assume that G is 2-vertex 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.

Next, we determine the set B consisting of all the separating-edges (vulnerable or not). If $B \neq \emptyset$, we choose one separating-edge and recurse on subgraphs of G.

If B is empty, then by Theorem 7.3 we have three remaining cases to deal with. If 7.3(P4) holds, the algorithm will find an extension by making recursive calls on certain minors of G. If either 7.3(P3) or 7.3(P5) holds,

the algorithm finds an extension directly, without recursing. We begin by discussing the cases where no recursion is required, followed by the case where G has a separating edge, and finally the most complex case where G has a non-trivial uncrossed vulnerable star-cutset.

7.2.1 Exploiting the wheel

In this section, we consider the simplest remaining case in the structural theorem; namely, Case 7.3(P5) holds and so G is a padded wheel graph. Recall that a wheel consists of an induced cycle together with a hub vertex adjacent to the vertices in the cycle, and that a padded wheel graph is obtained from a wheel by replacing some of the cycle edges by paths of length 2; the allowed configurations are shown in Figure 7.1.

We adopt the notation used in Section 7.1.3, i.e. $S = \{v, x, y\}$ is a non-trivial star-cutset which is crossed, C and D are the two components of $G - S$, $\mathcal{S}$ is the collection of star-cutsets crossing S, $A = \{a_1, \ldots, a_r\}$ and $B = \{b_1, \ldots, b_t\}$ consist of the vertices of C respectively D occurring in some star-cutset of $\mathcal{S}$. Set $a_0 = b_{t+1} := x$ and $a_{r+1} = b_0 := y$. For each i such that $1 \leq i \leq r$ and j such that $1 \leq j \leq t$, let U_i and W_j denote the vertices "between" a_i and a_{i+1} respectively b_j and b_{j+1}.

Theorem 7.17. *Let T be a star-cutset in G. Then T must take one of the following three shapes:*

(1) $T = \{v, a, b\}$ with a and b in $A \cup B \cup \{x, y\}$.

(2) $T = \{a_i, v, a_{i+1}\}$ (or $T = \{b_j, v, b_{j+1}\}$) for some $i \leq r$ ($j \leq t$) and T is a triangle cutset.

(3) $T = \{u_i, a_i, a_{i+1}\}$ for some $i \leq r$ and $u_i \in U_i$ or $T = \{w_j, b_j, b_{j+1}\}$ for some $j \leq t$ and $w_j \in W_j$.

To prove the theorem, observe that we can partition the vertices into three types: The hub v, the set $A \cup B \cup \{x, y\}$, and the set $\bigcup_i U_i \cup \bigcup_j W_j$. We shall analyse how edges incident to vertices in these three categories can occur in star-cutsets.

Lemma 7.18. *If a star-cutset T is centred at v, then T cannot contain a vertex $u \in U_i$ or $w \in W_j$.*

Proof. Suppose T contains an edge (v, u) with $u \in U_i$ for some i. Then u cannot intersect all $x - y$ paths through C, so any two vertices of $G - T$ will still be connected by a path going through either C or D, depending on where the third vertex of T lies.

The proof for an edge (v, w) with $w \in W_j$ is analogous. □

Lemma 7.18 implies that no optimal solution will contain edges of the form (v, u) and (v, w) with u and w in some U_i respectively W_j. Hence our algorithm will never take these edges into consideration for protection.

Star-cutsets that are centred at some vertex a_i or b_i are equally easy to deal with.

Lemma 7.19. *If T is a star-cutset centred at a vertex $a \in A \cup \{x, y\}$, then T must be a triangle cutset of the form $T = \{a_i, v, a_{i+1}\}$ for some i with $0 \leq i \leq r$. Moreover, the set U_i must consist of a single vertex. An analogous statement holds for star-cutsets T centred at a vertex $b \in B \cup \{x, y\}$.*

Proof. Suppose that T is centred at a_i, where $0 \leq i \leq r + 1$. It is clear that none of the following can be a star-cutset:

(1) $\{a_i, u, u'\}$ with $u \in U_{i-1}$ and $u' \in U_i$ (where $i \geq 1$);

(2) $\{a_i, a_{i-1}, u\}$ with $u \in U_i$ or $u \in U_{i-1}$ (where $i \geq 1$);

(3) $\{a_i, a_{i+1}, u\}$ with $u \in U_{i-1}$ (where $1 \leq i \leq r$) or $u \in U_i$ (where $0 \leq i \leq r$);

(4) $\{a_i, a_{i-1}, a_{i+1}\}$ (where $1 \leq i \leq r$);

(5) $\{a_i, u, u'\}$ with $u, u' \in U_i$ or $u, u' \in U_{i-1}$ (where $i \geq 1$);

(6) $\{a_i, v, u\}$ with $u \in U_i$ or $u \in U_{i-1}$ (where $i \geq 1$).

This leaves the possibility that $T = \{a_i, v, a_{i+1}\}$ or $T = \{a_i, v, a_{i-1}\}$, where (a_i, a_{i+1}) or (a_i, a_{i-1}) is an edge. In the former case, since (a_i, a_{i+1}) cannot be a separating-edge, it follows that every vertex of U_i must be adjacent to v, which implies that U_i contains a single vertex by Claim 7.16. Note that U_i cannot be empty, for then T would not be a cutset. The proof for the latter case ($T = \{a_i, v, a_{i-1}\}$) is analogous, as is the proof for star-cutsets centred at $b \in B \cup \{x, y\}$. □

Lemma 7.19 implies that the only star-cutset centred at some a_i or b_i is in fact a triangle cutset, and that the edge (a_i, a_{i+1}) is contained only in the two cutsets arising from the triangle by regarding either a_i or a_{i+1} to be the centre. Since in a triangle cutset two of the three edges must be protected, we may therefore assume that any optimal solution protects (v, a_i) and (v, a_{i+1}), which is what the algorithm will do when it encounters this particular configuration.

Lemma 7.20. *$T = \{u, a_i, a_{i+1}\}$ is a star-cutset centred at $u \in U_i$ for some i if and only if U_i contains a vertex $w \neq u$ which is not adjacent to v. Moreover, there can be no other star-cutsets centred at u.*

An analogous statement holds for $w \in W_j$ for some j.

Proof. If U_i contains a vertex $w \neq u$ which is not adjacent to v, then $T = \{u, a_i, a_{i+1}\}$ disconnects w from v. Conversely, if $U_i - \{u\}$ is empty or every vertex $w \neq u$ in U_i is adjacent to v, then $T = \{u, a_i, a_{i+1}\}$ is not a cutset.

Finally, it is clear that neither $\{u, v, a_i\}$ nor $\{u, v, a_{i+1}\}$ can be a star-cutset, so if u centres a star-cutset, it has to be $\{u, a_i, a_{i+1}\}$. □

Combining Lemmas 7.19 and 7.20, we can see that the edges (u, a_i) and (u, a_{i+1}) with $u \in U_i$ can be in at most one star-cutset, namely (u, a_i, a_{i+1}). It follows that an optimal solution will protect precisely one of those two edges, and switching to the other one does not change the cardinality of the solution. Hence our algorithm will arbitrarily pick one of (u, a_i) and (u, a_{i+1}) in the case where u centres a star-cutset.

Finally, the algorithm needs to decide which of the edges incident to v and to a vertex a_i or b_i to protect.

Lemma 7.21. *Let a, b be distinct vertices from the set $A \cup B \cup \{x, y\}$ such that the edges from v to a, b are unprotected. Then $T = \{v, a, b\}$ is a vulnerable star-cutset centred at v unless, for some i, $a = a_i$ and $b = a_{i+1}$ and $U_i = \emptyset$.*

Proof. If $a = x$ and $b = y$, then $T = S$ so T is a star-cutset. If $a \in A$ and $b \in B$, then T is a star-cutset, because a intersects every $x - y$ path through C and b intersects every $x - y$ path through D. So suppose a and b are both in $A \cup \{x, y\}$, i.e. $a = a_i$ and $b = a_j$ with $0 \leq i < j \leq r + 1$ (the case where a and b are in $B \cup \{x, y\}$ is analogous). If $j > i + 1$, we already showed in Section 7.1.3 that T must be a star-cutset. The same is true if $j = i + 1$ and $U_i \neq \emptyset$. The only remaining case is that $j = i + 1$ and $U_i = \emptyset$, in which case it is clear that T is not a star-cutset. □

Lemmas 7.18, 7.19, and 7.20 imply Theorem 7.17, and together with 7.21 establish the correctness of the following algorithm for solving the Network Protection Problem, given a padded wheel G and a set of already protected edges F:

(1) For every i such that $\{v, a_i, a_{i+1}\}$ ($\{v, b_i, b_{i+1}\}$) is a vulnerable triangle

cutset, add the two edges (v, a_i) and (v, a_{i+1}) ((v, b_i) and (v, b_{i+1})) to F.

(2) For every i and $u \in U_i$ ($w \in W_i$) such that $\{u, a_i, a_{i+1}\}$ ($\{w, b_i, b_{i+1}\}$) is a star-cutset, choose one of the two edges (u, a_i) and (u, a_{i+1}) ((w, b_i) and (w, b_{i+1})) and add it to F.

(3) If there exists an index i such that $U_i = \emptyset$ and both (v, a_i) and (v, a_{i+1}) are unprotected, add all edges (v, a_j) with $j \notin \{i, i+1\}$ and all edges (v, b_k) to F. If $U_i \neq \emptyset$ for all i, but $W_j = \emptyset$ for some j, add all edges (v, b_i) with $i \notin \{j, j+1\}$ and all edges (v, a_k) to F. If there is no index i such that either U_i or W_i is empty and a_i, a_{i+1} are unprotected, pick an unprotected edge from v to $A \cup B \cup \{x, y\}$ and add all the other edges from v to $A \cup B \cup \{x, y\}$ to F.

The correctness of the algorithm can be seen as follows: In any triangle cutset of the shape $\{v, a_i, a_{i+1}\}$ or $\{v, b_i, b_{i+1}\}$, at least two edges must be protected in any feasible solution. Our algorithm chooses the two edges incident to v, since those edges can occur in other star-cutsets, whereas the edge (a_i, a_{i+1}) respectively (b_i, b_{i+1}) cannot. Next, in any cutset of the form $\{u_i, a_i, a_{i+1}\}$ or $\{w_i, b_i, b_{i+1}\}$, an optimal algorithm will protect precisely one of the two edges incident to u_i respectively w_i. As both edges cannot occur in other star-cutsets, we are free to pick one of them arbitrarily. For the final case of a star-cutset T centred at v, observe that T cannot contain a vertex from some U_i or W_j by Lemma 7.18. Lemma 7.21 then implies that any feasible algorithm must protect all but at most two of the edges from v to the vertices in $A \cup B \cup \{x, y\}$. If there is an index i such that U_i or W_i is empty, we can leave (v, a_i) and (v, a_{i+}) (respectively (v, b_i) and (v, b_{i+1}) unprotected. If there is no such i, the best we can do is to leave exactly one edge from v to the set $A \cup B \cup \{x, y\}$ unprotected.

We remark that it is easy to implement the above to run in linear time.

Clearly the triangle cutsets of the form $\{v, a_i, a_{i+1}\}$ and the star-cutsets $\{u, a_i, a_{i+1}\}$ can be found and protected in time $O(m)$. The same holds true for finding two vulnerable adjacent spokes (v, a_i) and (v, a_{i+1}) such that $\{v, a_i, a_{i+1}\}$ is not a cutset.

7.2.2 Dealing with trivial components

Next, suppose that Case 7.3(P3) holds, i.e. 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 solved optimally via a reduction to the Minimum Vertex Cover problem in a line graph. Consider the graph L^* defined as follows:

$$V(L^*) := \{v_e \,|\, e \in E(G) \text{ is in a star-cutset of } G\}.$$

and

$$E(L^*) := \{(v_e, v_f) \,|\, e \text{ and } f \text{ share a vertex } \textit{and} \text{ form a star-cutset in } G\}$$

Observe that there is an obvious one-to-one correspondence between solutions to the network protection problem in G and vertex covers in L^*: If C^* is a vertex cover in L^*, let $F := \{e \in E(G) \,|\, v_e \in C^*\}$. Then, by the definition of L^* and because C^* is a vertex cover, F must contain at least one ray of every star-cutset. Conversely, given a solution F to the network protection problem, we can define $C^* := \{v_e \in V(L^*) \,|\, e \in F\}$. Since F intersects the set of rays of every star-cutset, we must have that C^* is a vertex cover.

Now, the vertex cover problem is solvable in polynomial time in line graphs, and L^* is evidently a subgraph of the line graph L of G. At first glance, this does not seem to help us, as subgraphs of line graphs are generally not line graphs themselves. We can, however, prove that L^* is a line

graph and thus can also solve the network protection problem in polynomial time when Case 7.3(P3) holds.

Theorem 7.22. *Let G be a planar graph such that any star-cutset S yields at most one non-trivial component. Then L^* is a line graph.*

Lemma 7.23. *We have $d(v_e) \leq 4$ for all $v_e \in L^*$. Moreover, if $d(v_e) = 3$, it must be the case that v_e is contained in an edge and a triangle, and if $d(v_e) = 4$, then v_e is contained in two triangles (see Figure 7.5).*

Figure 7.5: $d(v_e)$ is 3 or 4

Proof of Theorem 7.22. Lemma 7.23 immediately implies Theorem 7.22. This follows as the lemma means that the edges of L^* can be covered by cliques (more specifically, by edges and triangles) such that each vertex is contained in at most 2 cliques. This, in turn, is equivalent to saying that L^* is a line graph (see, for example, [47]). □

Thus, it remains to prove Lemma 7.23. Before doing so, we present three claims that will be useful.

Claim 7.24. *Let $S_1 = \{v, x, y_1\}$, $S_2 = \{v, x, y_2\}$ and $S_3 = \{v, x, y_3\}$ be distinct star-cutsets in G each containing the edge (x, v). Then for each $i = 1, 2, 3$ there exists a singleton component $\{w_i\}$ of $G - S_i$ such that $w_1, w_2, w_3, y_1, y_2, y_3$ are all distinct.*

Proof. It is clear that the w_i exist, because $G - S_i$ has at most one non-trivial component for each i, by our assumption on G. Also, by definition y_1, y_2, y_3 are distinct, and $w_1 \neq y_1, w_2 \neq y_2$ and $w_3 \neq y_3$.

Next, $\{v, x, w_1\}$ cannot be a star-cutset. To see this, note that w_1 is a singleton component in $G - S_1$ and all the other components of $G - S_1$ must have an edge to y_1, as y_1 is not the centre of S_1. Therefore every vertex of $G - \{x, v, w_1\}$ is in the same component as y_1, and $G - \{v, x, y_1\}$ is connected. It follows that $w_1 \neq y_2$ and $w_1 \neq y_3$. Analogously, we have that $w_2 \neq y_1$ and $w_2 \neq y_3$, and that $w_3 \neq y_1$ and $w_3 \neq y_2$. Thus it remains to show that w_1, w_2, w_3 are distinct. To see this, note that w_1 and y_1 are adjacent vertices in $G - S_2$, and therefore w_1 cannot form the singleton component $\{w_2\}$ of $G - S_2$. Similarly, we obtain that $w_1 \neq w_3$, and that $w_2 \neq w_3$, completing the proof. □

We remark that this claim does not require the centres of each S_i (which is either v of x) to be specified. In particular, the claim allows any S_i to be a triangle cutset, i.e. a set of vertices $\{x, v, y_i\}$ that induces a 3-cycle in the graph and whose removal leaves a disconnected graph[3].

Claim 7.25. *Let $S_1 = \{v, x, y_1\}, S_2 = \{v, x, y_2\}$ and $S_3 = \{v, x, y_3\}$ be distinct star-cutsets in G each containing the edge (x, v). For each $i \neq j$, the edge (w_i, y_i) is in the non-trivial component of $G - S_j$.*

Proof. As above, (w_i, y_i) is an edge as y_i is not the centre of S_i. Then, by Claim 7.24, as neither w_i nor y_i equals y_j, both vertices must be in the same component of $G - S_j$. Thus this component must be the non-trivial component. □

Claim 7.26. *Let $S = \{v, x, y\}$ be a triangle-cutset, and let w be a singleton component of $G - S$. Then the 3-cycle $v - x - y$ is a face of $G - \{w\}$.*

Proof. As (x, y) is an edge, it must be the case that $\{w\}$ is strongly attached to S, otherwise there would be a separating-edge. So w is adjacent to v, x

[3]Recall that a triangle cutset $\{v, x, y\}$ gives rise to 3 star-cutsets, as each of the 3 vertices can be regarded as the centre.

and y, but not to any other vertices. Consider any planar drawing of G. There can be no vertices of G inside the 3-cycle $x - y - w$, since otherwise (x, y) would be a separating-edge. By symmetry, there can be no vertices inside the 3-cycles $x - v - w$ and $y - v - w$ either. Hence $x - y - v$ must be a face of $G - \{w\}$. □

Proof of Lemma 7.23. Let $e = (x, v) \in E(G)$ be in a minimal star-cutset $S_1 = \{v, x, y_1\}$. We split our analysis into two parts, depending upon whether or not e is in a triangle cutset.

Case I. *The edge e is not in a triangle cutset.*

Now, if e is in at most two star-cutsets then v_e has degree at most two in L^*, and there is nothing more to show. We may therefore assume that e lies in at least two other cutset $S_2 = \{x, v, y_2\}$ and $S_3 = \{x, v, y_3\}$. By assumption, neither of these are triangle cutsets and so y_1, y_2 and y_3 are distinct.

Since v is the centre of S_1, observe that (x, y_1) is not an edge.

We now attempt to derive a contradiction. To this end, let w_i be a singleton component of $G - S_i$ for $i = 1, 2, 3$. By Claim 7.24, the six vertices $y_1, y_2, y_3, w_1, w_2, w_3$ are distinct. Thus, by Claim 7.25, w_1, y_1, w_2 and y_2 all lie in the unique non-trivial component of $G - S_3$. So there must be a path from w_1 to w_2 in $G - S_3$. Moreover, w_1's neighbour on this path must be y_1, and w_2's neighbour must be y_2, since w_1 and w_2 have no other neighbours in $G - S_3$.

We now need to distinguish two further non-symmetric subcases depending upon the centres of S_1, S_2 and S_3. As they are not triangle cutsets, none of them are centred at a y_i as xv is an edge. So, without loss of generality either two or three of them are centred at v.

(1) *S_1, S_2 and S_3 are centred at v.*

This means that w_1 and w_2 must both be adjacent to x, and therefore

we can extend the $w_1 - w_2$ path exists in $G - S_3$ to a cycle C by adding x. The situation is depicted in Figure 7.6. By planarity, without loss of generality, both v and y_3 lie inside of C. Firstly, if y_3 lies inside the cycle C' obtained by the subpath of C from y_2 to y_1 together with v, then w_3 cannot have edges to both x and y_3, a contradiction. Secondly, if y_3 lies inside the cycle $x - v - y_1 - w_1$, then $G - S_1$ has at least two non-trivial components (one containing $\{y_3, w_3\}$ and one containing $\{y_2, w_2\}$), a contradiction. Finally, if y_3 lies inside the cycle x, v, y_2, w_2 then $G - S_2$ has at least two non-trivial components, (one containing $\{y_3, w_3\}$ and one containing $\{y_1, w_1\}$), a contradiction.

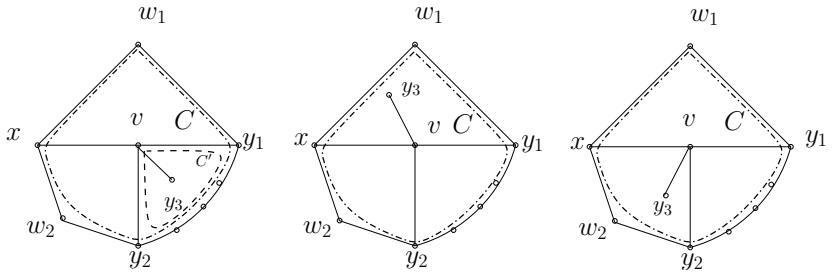

Figure 7.6: Illustration of case I.a

(2) *S_1 and S_2 are all centred at v, but S_3 is centred at x.*

This is depicted in Figure 7.7. The vertex y_3 cannot lie outside of C, because then w_3 could not have edges to both y_3 and v. But it cannot lie inside one of the cycles $x - v - y_1 - w_1$ or $x - v - y_2 - w_2$ either, because then either $G - S_1$ or $G - S_2$ would have at least two non-trivial components.

Consequently, for Case I, we have $d(v_e) \leq 2$ in L^*. We now turn our attention to the case where e is in a triangle cutset.

Case II. *The edge e is in a triangle cutset.*

Let the triangle cutset be $S_1 = \{x, v, y_1\}$. Therefore, by Lemma 7.8 ,all the components of $G - S_1$ must be strongly attached; consequently, by Lemma

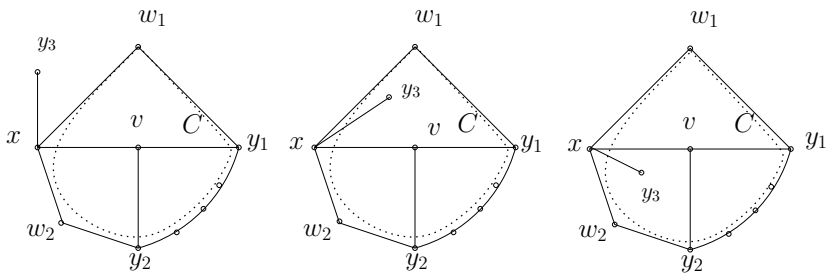

Figure 7.7: Illustration of case I.b

7.7, it contains exactly two components. If e is in no other star-cutsets, then $d(v_e) = 2$ and there is nothing to show. So suppose that e is in at least one other star-cutset $S_2 = \{x, v, y_2\}$ with $y_2 \neq y_1$.

(1) *S_2 is a triangle cutset.*

Thus xv is in two triangle cutsets and, hence, in four distinct star-cutsets, as shown in Figure 7.8. Suppose e is in a further (fifth) star-cutset $S_3 = \{x, v, y_3\}$ with $y_3 \neq y_1, y_3 \neq y_2$. Without loss of generality, assume that S_3 is centred at x.

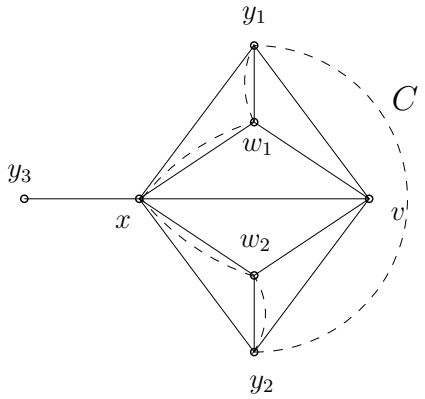

Figure 7.8: Illustration of case II.a

Let w_i be a singleton component of $G - S_i$. Again, the vertices $w_1, w_2, w_3, y_1, y_2, y_3$ are distinct by Claim 7.24.

By applying Claim 7.26 twice, we may assume that the vertices w_1 and w_2 are inside the triangles $\{x, v, y_1\}$ and $\{x, v, y_2\}$, respectively.

This follows as both are faces of $G-\{w_1, w_2\}$ and we may choose some other face to be the infinite one.

By Claim 7.25, w_1 and w_2 must be in the same component of $G - S_3$. Therefore, there is an $w_1 - w_2$ path in $G - S_3$, which can be extended to a cycle C by adding x. By Claim 7.26, y_3 must lie outside the two triangles. Hence, there cannot be both an edge from w_3 to y_3 and an edge from w_3 to v.

Thus e cannot lie in a fifth star-cutset, and so $d(v_e) = 4$. Moreover, v_e is in two triangles, as desired, since e is in two triangle cutsets.

(2) *S_2 is not a triangle cutset.*

We may assume that S_2 is centred at x. Thus (x, y_2) is an edge but (v, y_1) is not. If e s in no other cutsets then we are done. So, suppose e is in a fourth cutset $S_3 = \{x, v, y_3\}$. As before, let w_i be a singleton component of $G - S_i$. Invoking Claim 7.24, $w_1, w_2, w_3, y_1, y_2, y_3$ must be distinct.

Note that we lose no generality by drawing w_1 inside the triangle $\{x, v, y_1\}$, since by Claim 7.26 $\{x, v, y_1\}$ is a face of $G - \{w_1\}$. This also implies that neither y_2 nor y_3 can lie inside this triangle. We then have two cases left to deal with.

(i) *S_3 is centred at x.*

This situation is as shown in Figure 7.9. By Claim 7.25, we can find a path from y_2 to y_1 in $G - S_3$, which together with x gives a cycle C. If y_3 lies outside of C, then w_3 cannot have edges to both b and v. If y_3 lies inside of C, it must lie within the cycle $x - v - w_2 - y_1$, and then $G - S_2$ has at least two non-trivial components, one containing $\{y_3, w_3\}$ and one containing $\{y_1, w_1\}$.

(ii) *S_3 is centred at v.*

This is depicted in Figure 7.10. By Claim 7.25, we can find a path

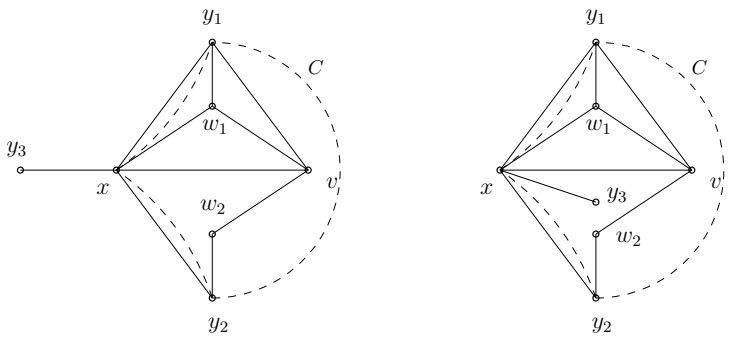

Figure 7.9: Illustration of case II.b(i).

from w_2 to y_1 in $G - S_3$, which together with v gives a cycle C. If y_3 lies inside of C, then w_3 cannot have edges to both y_3 and v. If y_3 lies outside C, it must lie within the cycle $x - v - w_2 - y_2$, and then $G - S_2$ produces at least two non-trivial components, one containing $\{y_3, w_3\}$ and one containing $\{y_1, w_1\}$.

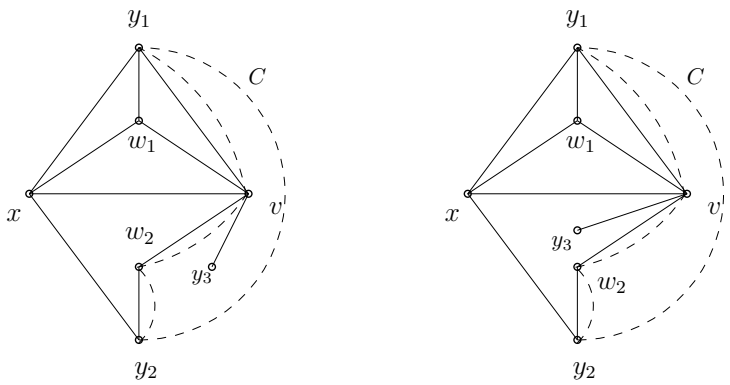

Figure 7.10: Illustration of case II.b(ii)

These contradictions imply that v_e has degree 3. Moreover, it is in one triangle and one edge, as desired. □

We conclude this section with two remarks. First, the vertex cover problem in a line graph $L(G)$ of a graph G is solved by observing that the set of edges $F \subseteq E(G)$ is a matching in G if and only if $V(L) - X_F$ is a vertex cover in L, where X_F is the set of vertices of L corresponding to the edges

in F. Hence a minimum vertex cover $\mathcal{C}$ in L can be obtained by first finding a maximum matching $\mathcal{M}$ in G and then letting $\mathcal{C} = V(L) - X$, where $X = \{v_e \in V(L) \,|\, e \in \mathcal{M}\}$. The maximum matching can be found in polynomial time; furthermore, the graph H of which our graph L^* is the line graph can be constructed in linear time, as shown in [82].

Second, the above can easily be extended to the situation where a set of vertices of L^* is given and must be extended to a valid vertex cover, as is the case in our algorithm. To see this, just observe that the vertices of L^* in a vertex cover correspond to edges of G that are not in the corresponding matching. So given a partial vertex cover in L^*, we simply remove the corresponding edges from G and then find a maximum matching in the remaining graph.

7.2.3 Splitting along a separating-edge

As we shall see in this section, a separating-edge offers a very natural way of splitting the Network Protection Problem into two subproblems on subgraphs of G. An alternative approach would be to add all the separating-edges to F; however, being able to assume that a given graph has no separating-edges at all, whether vulnerable or not, leads to a simplified statement and proof of the structural result and the analysis of the algorithm for the last three cases of Theorem 7.3.

So, let G be given, and let $S = \{u, v\}$ be a separating-edge. Let $C_1, \ldots, C_r$ be the components of $G - S$, and define $G_i := G[C_i \cup \{u, v\}]$.

Theorem 7.27. *Every minimal star-cutset $T \neq S$ in G is a star-cutset in G_i for a unique i. Conversely, for any i, a star-cutset T in G_i is a star-cutset in G.*

The first step towards proving the theorem is to show that any star-cutset $T \neq S$ is contained (as a set of vertices) in $V(G_i)$ for some i.

Lemma 7.28. *S is not crossed by a minimal star-cutset.*

Proof. Suppose there were a minimal star-cutset T crossing S. Without loss of generality, we may assume that T is centred at u, so $T = \{u, x, y\}$ where x and y are in distinct components C respectively D of $G - S$. Let w be any vertex of $G - T$. Our goal is to show that there must be a path from w to v in $G - T$, contradicting the assumption that T is a cutset.

If $w = v$ or w is in a component of $G - S$ other than C or D, this is trivial. So assume w is in $C - x$. If x intersects every $w - v$ path contained in D, then either $\{u, x\}$ is a separating-edge, contradicting the minimality of T, or x is a cut-vertex, contradicting the 2-vertex connectivity of G. Hence there must be a path from w to v in $G - T$. An analogous argument works if $w \in D$. □

Lemma 7.29. *Every minimal star-cutset $T \neq S$ of G is a star-cutset in G_i for some i.*

Proof. Let i be the unique index such that $V(G_i)$ contains T, which must exist by Lemma 7.28. Since T is minimal, S cannot be contained in T, so without loss of generality suppose that $v \notin T$. As T is a cutset, there must be some w disconnected from v in $G - T$. Every vertex of G_j with $j \neq i$ is clearly still connected to v, hence this w must lie in G_i. But then it follows that T is a star-cutset in G_i, as claimed. □

Lemma 7.30. *For any i, a cutset T of G_i is a cutset in G.*

Proof. Let x, y be two vertices of G_i such that there is no $x - y$ path in $G_i - T$. If there were an $x - y$ path P in $G - T$, then P would have to go through u and v, in which case we could shortcut it using the edge (u, v) to obtain an $x - y$ path in G_i. □

Lemma 7.30 implies that any star-cutset in G_i is also a star-cutset in G. It also implies that each subgraph G_i must be 2-vertex connected, since a cut-vertex in G_i would be a cut-vertex in G.

For each i let $F_i := F \cap E(G_i)$. Then we can solve the Network Protection Problem by calling the algorithm recursively on each G_i to obtain an extension A_i of F_i, and get an extension A of F by setting $A := \cup_i A_i \cup (u, v)$.

Since any star-cutset in G is a star-cutset in some G_i, we have that A intersects $R(S)$ for every star-cutset S of G, i.e. A is a feasible solution.

Claim 7.31. *The extension A obtained from the partial solutions is optimal.*

Proof. Let Z be an optimal extension of F in G. By Lemma 7.30, $Z_i := Z \cap E(G_i)$ is an extension of F_i in G_i. Note that Z must contain the separating-edge $e = (u, v)$, but e cannot be a separating-edge in G_i. So assuming that the recursive calls return optimal partial extensions, we have that $|A_i| \leq |Z_i| - 1$ for all i, which implies

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

□

7.2.4 The recursion

Finally, suppose that Case 7.3(P4) 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 minors of G. We already know how to solve the subproblems occurring at the leaves of the recursion; they are the cases dealt with in Section 7.2.1 and Section 7.2.2. The main technical 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 from the subproblems. Rather, to obtain a polynomial time algorithm we may only choose to 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 optimal solution.

Let us 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. However, Lemma 7.8 implies that if there exists a weakly attached component, then $(x, y) \notin E(G)$. 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 also a weakly attached component and the case in which there are no weakly attached components.

Although the four cases all require to be handled slightly differently, the basic procedure is always the same, and we outline it now in order to make the following detailed case analysis appear less daunting. As a first step, we will always show that the minors defining the subproblems are again 2-vertex connected planar graphs, so that making recursive calls of the algorithm is well defined. The second step is to show that each star-cutset of G (other than the cutset S that defines the subproblems) appears as a star-cutset in one of the subproblems, thus ensuring that the algorithm does not miss any star-cutsets and returns a feasible solution. To prove the performance guarantee, we will need the fact that a star-cutset in one of the subproblems is also a star-cutset in G. This means that the partial solutions will not choose edges to protect that are not really needed in the original graph. Lastly, we will show that the size of the combined solution returned by the algorithm will always be at most the size of an optimal solution.

In all the cases, the star-cutset giving rise to the subproblems will be denoted by $S = \{v, x, y\}$, with v being the centre in the cases where $(x, y) \notin E(G) - F$. The edges of S will be denoted by $e := xv$, $f := yv$ and $g := (x, y)$ (when $(x, y) \in E(G) - F$).

No strongly attached components

`Case I.` In the first case we consider, $G - S$ 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 $i \neq j$. This fact will be used repeatedly in the analysis.

Theorem 7.32. *Assuming that each recursive call returns an optimal extension A_i of F_i in G_i, we have that A 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-vertex connected planar graph. The planarity is obvious, so it remains to show the 2-connectivity.

Claim 7.33. *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 7.34. *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 7.8, $(x, y) \notin E(G) - F$. 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 7.34 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 (v, x) 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 show that the algorithm returns an optimal solution, we need the following converse to Claim 7.34.

Claim 7.35. *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 7.8, $(x, y) \notin$

$E(G) - F$ 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, without loss of generality, 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 (v, x). This contradicts the assumption that S' separates x and u. □

Claim 7.36. *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 optimal extension A_i 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 $|A| \leq |Z|$.

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

Lemma 7.37. *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 7.8, $(x, y) \notin E(G) - F$. 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 7.4. □

Claim 7.38. *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 7.37, 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 7.32. Again, without loss of generality, assume that Z contains e but not f. Claim 7.35 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 7.37, so $Z_i - e$ is an extension of F_i. So if we recurse on G_i then, by induction $|A_i| \leq |Z_i| - 1$. If we did not recurse on G_i because C_i was trivial, then $|A_i| = 1$ and so $|A_i| \leq |Z_i| - 1$, since by Claim 7.36, 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

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

This concludes the proof of Theorem 7.32 □

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 $C_i = \{c_i\}$ is a singleton, set $A_i = \{c_i x\}$. Return the output $A = \cup_i A_i$.

Again, by Lemma 7.8, the existence of a weakly attached component implies that $(x, y) \notin E(G)$.

Lemma 7.39. *Assuming that each recursive call returns an optimal extension A_i of F_i we have that A 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 7.40. *For all i, the graph G_i is 2-connected.*

Proof. The proof is virtually the same as in Claim 7.33. 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 7.41. *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 7.41 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 7.41.

Claim 7.42. *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 $(x, y) \notin E(G) - F$, by

Lemma 7.8. 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 (v, x), contradicting the fact that u and x are separated in $G_i - S'$. □

Claim 7.43. *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 7.39, 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 $|A| \leq |Z|$.

Claim 7.44. *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, without loss of generality 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 optimal extension A_i of F_i. Suppose first that Z contains e but not f. Claim 7.42 implies that for all $i \geq 1$, Z_i is also an extension of F_i. In particular, this gives $|A_1| \leq |Z_1|$.

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

Finally, if $C_i = \{c_i\}$ is a trivial weakly attached component then, by Claim 7.43, $\{c_i, x, y\}$ is a star-cutset in G. So, when $A_i = \{c_i x\}$, we also have $|A_i| \leq |Z_i| - 1$.

Hence the solution returned by the algorithm has size

$$\sum_{i \geq 1} |A_i| = |A_1| + \sum_{i \geq 2} |A_i| \leq |Z_1| + \sum_{i \geq 2} (|Z_i| - 1) \leq 1 + \sum_{i \geq 1} (|Z_i| - 1) = |Z|.$$

If Z contains e and f, similar arguments imply that $|A_i| \leq |Z_i| - 2$, for all $i \geq 2$, and that $|A_1| \leq |Z_1|$. Hence, the size of the solution A returned

by the algorithm satisfies

$$|A| = |A_1| + \sum_{i \geq 2} |A_i| \leq |Z_1| + \sum_{i \geq 2} (|Z_i| - 2) \leq 2 + \sum_{i \geq 1} (|Z_i| - 2) = |Z|.$$

This concludes the proof of Lemma 7.39 □

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 are in Case 7.3(P4), 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 than 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 $m_1 \leq m_2$. 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 optimal solutions to the subproblems are combined to give a global optimal solution.

A second issue that we need to deal with here is that there may be an edge $g = (x, y) \in E(G) - F$. (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 (x, y) separately.

`Condition 1:` If $(x, y) \notin E(G) - F$ then 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.

(1) 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 $A_{ef} \cup B_{ef} \cup \{e, f\}$.

(2) 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 $A_e \cup B_e \cup e$.

(3) 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:` If $g = (x, y) \in E(G) - F$ then 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.

(1) 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 $A_{efg} \cup B_{efg} \cup \{e, f, g\}$.

(2) 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 $A_{ef} \cup B_{ef} \cup \{e, f\}$.

(3) Exactly two of A_{ef}, A_{eg}, A_{fg} are equal to A_{efg} in cardinality. Suppose without loss of generality 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 $A_{fg} \cup B_f \cup f$.

(4) 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 optimal solutions to the subproblems cannot differ by much. This will be used shortly to prove that the partial extensions returned by the recursive calls can be combined to give an optimal global solution.

Claim 7.45. *Let $A_\diamond$ be an optimal extension of $\diamond \cup F_1$ in G_1. If $g = (x, y) \notin E(G) - F$ then the following hold:*

(1) $|A_e| - 1 \leq |A_{ef}| \leq |A_e|$ *and* $|A_f| - 1 \leq |A_{ef}| \leq |A_f|$.

(2) $|A_\emptyset| \leq |A_e| + 1$ *and* $|A_\emptyset| \leq |A_f| + 1$.

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

Proof. If $|A_e| \geq |A_{ef}| + 2$, set $A'_e := A_{ef} \cup f$. 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 $A'_{ef} := A_e - f$. Then clearly A'_{ef} is an extension of $ef \cup F_1$, and $|A'_{ef}| \leq |A_e|$. The proofs for A_f and for extensions $B_\diamond$ in G_2 are analogous. □

Note that the first assertion of Claim 7.45 implies that the cases (a) to (c) of Condition 1 above exhaust all the possibilities, i.e. that the algorithm is well defined.

Using analogous arguments, we can prove:

Claim 7.46. *Let $A_\diamond$ be an optimal extension of $\diamond \cup F_1$ in G_1. If $g = (x, y) \in E(G) - F$, then the following hold:*

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

(2) $|A_e| \leq \min(|A_{ef}|, |A_{eg}|) + 1 \leq |A_{efg}| + 2$.

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

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

Again observe that the first assertion implies that the cases (a) to (d) of Condition 2 above exhaust all the possibilities.

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

Claim 7.47. *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 7.47 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 $(x, y) \in E(G) - F$).

The difficult part of the analysis of the performance guarantee is to show that two optimal extensions in G_1 and G_2 combine to give an optimal extension of F in G. This is achieved by the following claim and its corollary.

Claim 7.48. *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 7.49. *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 $A_\diamond$ an optimal extension of $\diamond \cup F_1$ in G_1, and by $B_\diamond$ an optimal extension of $\diamond \cup F_2$ in G_2.*

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

- *If Z contains e but not f, then $|A_e| + |B_e| + 1 \leq |Z|$.*
- *If Z contains e and f, then $|A_{ef}| + |B_{ef}| + 2 \leq |Z|$.*

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

Proof. By Claim 7.48, Z_1 is an extension of F_1 in G_1. Suppose that Z contains exactly one of e and f, without loss of generality e. Then $Z_1 - e$ is an extension of $e \cup F_1$ in G_1. Since A_e is an optimal extension of $e \cup F_1$, we have that $|A_e| \leq |Z_1| - 1$. By an analogous argument, we also have $|B_e| \leq |Z_2| - 1$. Moreover, if $e \in Z$ but $f \notin Z$, we have that $|Z| = |Z_1| + |Z_2| - 1$. It follows that

$$|A_e \cup B_e \cup e| \leq |A_e| + |B_e| + 1 \leq |Z_1| - 1 + |Z_2| - 1 + 1 = |Z|$$

as claimed. The other case is similar. □

Similarly we may prove that:

Corollary 7.50. *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 $A_\diamond$ an optimal extension of $\diamond \cup F_1$ in G_1, and by $B_\diamond$ an optimal extension of $\diamond \cup F_2$ in G_2.*

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

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

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

We move on to show that the algorithm does indeed combine optimal partial extensions to form an optimal global extension. 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 optimal extension $A_\diamond$ of $\diamond \cup F_1$, and that each recursive call on G_2 returns an optimal extension $B_\diamond$ of $\diamond \cup F_2$.

As always, we need to distinguish between the cases $g = (x, y) \notin E(G) - F$ and $g = (x, y) \in E(G) - F$. 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:` $g = (x, y) \notin E(G) - F$.

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

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

$$|A_{ef}| + |B_{ef}| + 2 \leq |Z|$$

by Corollary 7.49.

(ii) If there is an optimal solution Z containing e but not f, then

$$|A_{ef}| + |B_{ef}| + 2 = |A_e| - 1 + |B_{ef}| + 2 = |A_e| + |B_{ef}| + 1 \leq |Z|$$

since $|B_{ef}| \leq |B_e|$.

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

(2) 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 $|A_e \cup B_e \cup e| \leq |A_e| + |B_e| + 1$.

(i) If there is an optimal solution Z containing e and f, we have that

$$|A_e| + |B_e| + 1 = |A_{ef}| + |B_e| + 1 \leq |A_{ef}| + |B_{ef}| + 2 \leq |Z|$$

since $|B_e| \leq |B_{ef}| + 1$.

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

$$|A_e| + |B_e| + 1 \leq |Z|.$$

(iii) If there is an optimal solution Z containing f but not e, then

$$|A_e| + |B_e| + 1 = |A_f| + |B_e| \leq |A_f| + |B_e| \leq |Z|$$

since $|B_e| \leq |B_f| + 1$.

(3) 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 $|A_e \cup B_\emptyset| \leq |A_e| + |B_\emptyset|$.

(i) If there is an optimal solution Z containing e and f, we have that

$$|A_e| + |B_\emptyset| = |A_{ef}| + |B_\emptyset| \leq |A_{ef}| + |B_{ef}| + 2 \leq |Z|$$

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

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

$$|A_e| + |B_\emptyset| \leq |A_e| + |B_e| + 1 \leq |Z|$$

since $|B_\emptyset| \leq |B_e| + 1$.

(iii) If there is an optimal solution Z containing f but not e, then

$$|A_e| + |B_\emptyset| = |A_f| + |B_\emptyset| \leq |A_f| + |B_f| + 1 \leq |Z|$$

since $|B_\emptyset| \leq |B_f| + 1$.

`Condition 2:` $g = (x, y) \in E(G) - F$.

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

(i) If there is an optimal solution Z containing e, f, g, then

$$|A_{efg}| + |B_{efg}| + 3 \leq |Z|.$$

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

$$|A_{efg}|+|B_{efg}|+3 = |A_{ef}|-1+|B_{efg}|+3 \leq |A_{ef}|+|B_{ef}|+2 \leq |Z|$$

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

(2) 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 $|A_{ef} \cup B_{ef} \cup ef| \leq |A_{ef}| + |B_{ef}| + 2$.

(i) If there is an optimal solution Z containing e, f, g, then

$$|A_{ef}| + |B_{ef}| + 2 = |A_{efg} + |B_{efg}| + 3 \leq |Z|$$

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

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

$$|A_{ef}| + |B_{ef}| + 2 \leq |Z|.$$

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

$$|A_{ef}| + |B_{ef}| + 2 = |A_{fg}| - 1 + |B_{ef}| + 2 \leq |A_{fg}| + |B_{fg}| + 2 \leq |Z|$$

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

(3) 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 $|A_{ef} \cup B_f \cup f| \leq |A_{ef}| + |B_f| + 1$.

(i) If there is an optimal solution Z containing e, f, g, then

$$|A_{ef}| + |B_f| + 1 = |A_{efg}| + |B_f| + 1 \leq |A_{efg}| + |B_{efg}| + 3 \leq |Z|$$

since $|B_f| \leq |B_{efg}| + 2$.

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

$$|A_{ef}| + |B_f| + 1 \leq |A_{ef}| + |B_{ef}| + 2 \leq |Z|$$

since $|B_f| \leq |B_{ef}| + 1$.

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

$$|A_{ef}| + |B_f| + 1 = |A_{fg}| + |B_f| + 1 \leq |A_{fg}| + |B_{fg}| + 2 \leq |Z|$$

since $|B_f| \leq |B_{fg}| + 1$.

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

$$|A_{ef}| + |B_f| + 1 = |A_{eg}| - 1 + |B_f| + 1 \leq |A_{eg}| + |B_{eg}| + 2 \leq |Z|$$

since $|B_f| \leq |B_e| + 1 \leq |B_{eg}| + 2$.

(4) 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 $|A_{ef} \cup B_\emptyset| \leq |A_{ef}| + |B_\emptyset|$.

(i) If there is an optimal solution Z containing e, f, g then

$$|A_{ef}| + |B_\emptyset| = |A_{efg}| + |B_\emptyset| \leq |A_{efg}| + |B_{efg}| + 3 \leq |Z|$$

since $|B_\emptyset| \leq |B_{efg}| + 3$.

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

$$|A_{ef}| + |B_\emptyset| \leq |A_{ef}| + |B_{ef}| + 2 \leq |Z|$$

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

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

$$|A_{ef}| + |B_\emptyset| = |A_{fg}| + |B_\emptyset| \leq |A_{fg}| + |B_{fg}| + 2 \leq |Z|$$

since $|B_\emptyset| \leq |B_{fg}| + 2$. 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 optimal extensions in G_1 respectively G_2 to form an optimal extension of F in G.

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 $i \geq 3$, 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 $m_1 \leq m_2$. 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 $(x, y) \in E(G) - F$. Suppose this returns a set of edges A' that intersects all star-cutsets that are either in G_1 or G_2.

For $i \geq 3$ such that C_i is non-trivial, call the algorithm recursively on (G_i, F_i) to obtain extensions A_i. For $i \geq 3$ 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 $A := \cup_{i \geq 3} A_i \cup A'$.

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, does not miss any cases is exactly as in Case III.

Claim 7.51. *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 (x, y) 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 are 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' does not 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 7.51 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 7.52. *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 is implied by the following 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-cut 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 $|A'| \leq |Z'|$. 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 $|A_i| \leq |J_i|$, 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

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

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

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

7.2.5 Running time analysis

Before analysing the run-time 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 separating edges. If $B \neq \emptyset$, we recurse as described in Section 7.2.3. If $B = \emptyset$, we may assume that all minimal star-cutsets of G consist of 3 vertices. If G is a padded wheel, we find an extension as described in Section 7.2.1. 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 7.2.2. If neither of these two cases holds, we know by Theorem 7.3 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 7.2.4.

Now, if G is a padded wheel, the run-time of the algorithm is bounded above by $O(m)$, as argued at the end of Section 7.2.1. 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: Given a list of vulnerable star-cutsets in G, the graph L^* can be constructed in time $O(m)$. Next, the graph H for which L^* is the line graph can be constructed in linear time, and then a maximum matching in H can be found in time $O(m\sqrt{n})$. Thus, the running time in this case is dominated by the time it takes to obtain the list of vulnerable star-cutsets; for our purposes, a brute-force $O(m^3)$ algorithm that tests, for every pair $(v, x), (v, y)$ of incident edges, whether $G - \{v, x, y\}$ is connected will suffice. (Note that given a pair $(v, x), (v, y)$, we can check in constant time whether (x, y) is an unprotected edge.)

Having shown that the algorithm performs in time at most $O(m^3)$ on the leaves of the recursion tree, it remains to prove that the recursive steps

can also be implemented efficiently.

Claim 7.53. *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 recurse on minors obtained from the components of an uncrossed star-cutset or the subgraphs obtained from the components of a separating-edge, 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 run-time, 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 Sections 7.2.3 and 7.2.4. For each i, let $m_i = |E(G_i)|$.

Theorem 7.54. *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. The assertions are easily seen to be true if we recurse on the subgraphs obtained from a separating-edge: The algorithm is called only once on each G_i, so

$$T(m) \leq \sum_{i=1}^{r} T(m_i) + O(m) \leq 4T(m_1) + \sum_{i=2}^{r} T(m_i) + O(m)$$

and

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

since the separating-edge is counted r times. After renumbering, we can assume that m_1 is the smallest of the m_i and so clearly $m_1 \leq \frac{m}{2} + 5$.

So now suppose we recurse on an uncrossed star-cutset.

`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 $i_0 = \text{argmin}(m_i)$, 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 $g = (x, y) \in E(G) - F$ then

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

since we are 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 $(x, y) \notin E(G) - F$, 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.

Given the bounds on the m_i and assuming inductively that the run-time of each subproblem is at most $O(m_i^3)$, we can use the recursive bound on $T(m)$ above to show that $T(m)$ is also bounded by $O(m^3)$. The proof proceeds by analyzing the maxima of the convex function $f((m_1, m_2, \ldots, m_r)) = C[4m_1^3 + \sum_{i\geq 2} m_i^3 + m]$ over the convex domain given by the bounds on the m_i; we omit the details.

Corollary 7.55. *The run-time of the algorithm is bounded above by $O(m^3)$.*

□

7.2.6 Hardness of the vertex variant

In this section we show that the vertex variant of network protection problem in planar graphs is indeed NP-hard. 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 7.56. *The vertex variant of the network protection problem is NP-hard 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 7.57. *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 (u, v) 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 (u, v) 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 7.58. *There is an optimal solution to the vertex network protection problem in G' that contains only vertices of G.*

Proof. From Claim 7.57, 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 7.58 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-hard, since finding a minimum vertex cover is NP-complete even in a 3-connected cubic planar graph ([92]). □

7.3 Neutralising star-cutsets in general graphs

Here we settle the complexity of the network protection problem for non-planar graphs by providing logarithmic lower and upper approximability bounds[4]. 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

[4]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.

complement this by providing a lower bound of $\log n$, showing that, up to a constant factor, our result is essentially best possible.

7.3.1 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 7.59. *For any $\varepsilon > 0$, there is no $(1-\varepsilon)\log n$-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.

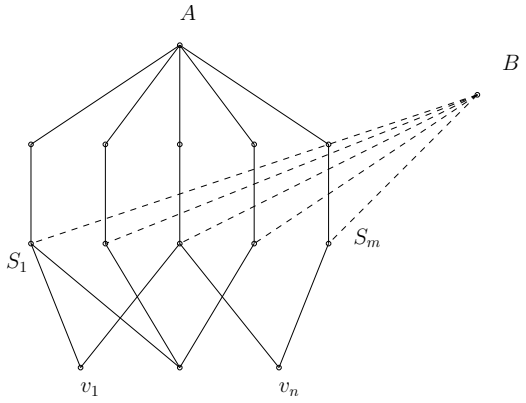

Figure 7.11: Reduction graph for the Set Cover reduction

Figure 7.3.1 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.

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 $(V - v_j, \mathcal{S})$. Observe 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, without loss of generality, 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 [24] showed that unless $NP \subset TIME(n^{O(\log \log n)})$, there is no polynomial time algorithm to approximate

Set Cover to within $(1-\varepsilon')\log n'$ for any $\varepsilon' > 0$, 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 $(1-\varepsilon)\log n$, where $\varepsilon > 0$ 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 7.59 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 $(1-\varepsilon)\log n$ approximation algorithm for the network protection problem would yield a $(1-\varepsilon')\log n'$ approximation algorithm for Set Cover for some $\varepsilon' > 0$. □

7.3.2 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. The set $\delta(v)$ denotes the edges incident to v. Clearly, the union of the solutions to each local problem gives a feasible global solution $F = \bigcup_v F_v$. 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. Recall that $\Gamma(v)$ denotes the set of neighbours of v; we then set $N(v) = \Gamma(v) \cup \{v\}$. Further, 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 7.60. *We can find local solutions F_v such that the global solution $F = \bigcup_{v \in B} F_v$ satisfies $|F| \leq O(\log n)\,\mathrm{OPT}$.*

Proof. We begin with a straightforward observation.

Observation 7.61. *If F^* is an optimal global solution, then for any v, the set $F^* \cap \delta(v)$ is a local solution.* □

Observation 7.61 implies that $|F^* \cap \delta(v)| \geq \mathrm{OPT}(v)$.

Claim 7.62. *For each v, we can find a local solution F_v of size $|F_v| \leq (c+1)\log(n)\mathrm{OPT}(v)$, where n is the number of vertices of G and c is an absolute constant.*

Claim 7.62 establishes the approximation guarantee. The size of our solution is

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

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

Thus, it remains to prove Claim 7.62. We will achieve this by giving 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 us 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.

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

(2) $N(u) - v \neq N(v)$. 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, $S_v = \{v\} \cup \{w \,|\, (v,w) \notin F_v\}$ is not a star-cutset in G. But then $G - S_v = G[D]$ is a a single component; thus D is connected.

This gives us the desired bijection. Hence, if $\textsc{opt}(CDS)$ is the weight of an optimal connected dominating set in G_v, we have $opt_{CDS} = opt_v$. Guha and Khuller [39] 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 7.60. □

Chapter 8

Conclusion

We studied galaxy cutsets as simple models for virus-like attacks and cascading failures in graphs, giving attention to the problems of identifying vulnerabilities in a given graph, designing resilient graphs, fixing existing vulnerabilities by protecting edges, and relating the cutsets to a suitable type of flow. The numerous hardness results we obtained indicate that researchers concerned with practical applications on real-world networks might do well to consider approximation algorithms, bi-criteria approaches, or even just heuristics without provable performance guarantees.

We close by listing open problems related to cutsets of radius r and galaxy cutsets, and also indicate possible avenues for future work concerning the weighted d-noninterfering flows that were the subject of Chapter 5. We begin with the list of open problems:

Open Problem 8.1. *What is the computational complexity of determining whether a given graph has a spanning subgraph without star-cutsets of radius r for $1 \leq r \leq 3$?*

Open Problem 8.2. *Find an approximately minimum spanning subgraph without star-cutsets of radius* 1 *in a graph that has no star-cutsets of radius* 1.

Open Problem 8.3. *Give a non-trivial approximation algorithm for the vertex variant of the Network Protection Problem. Also, what is the hardness of approximation for this variant, in planar or general graphs?*

Open Problem 8.4. *Can the run-time of the algorithm for finding star-cutsets of radius r be improved? In particular, can the dependence on r in the run-time be removed?*

Open Problem 8.5. *Investigate bi-criteria results for galaxy cutsets. For example, assuming that a graph has a galaxy cutset of order k, can we find a galaxy cutset of order ck, where $c > 1$ is a constant?*

An interesting direction for future work concerning the weighted d-flows would be to investigate the multi-commodity case, where we wish to find weighted flows between s_i and t_i, for $i = 1, \ldots, k$, that are d-noninterfering and maximise total weight. By the techniques of Section 5.2, we can easily obtain an upper bound of $O(\alpha \log n)$, where α is the approximation achievable in the unweighted case. Unfortunately, the unweighted version is extremely hard to approximate since it is the edge-disjoint paths problem studied by Guruswami et al. [45]. They show this problem is inapproximable to within $\alpha = m^{\frac{1}{2}-\varepsilon}$, for any $\varepsilon > 0$, in directed graphs and give an approximation algorithm that essentially matches this lower bound.

Another promising avenue would be to introduce congestion into the picture, i.e. rather than require the paths in a flow to be d-noninterfering, we would allow a constant (or logarithmic) number of them to intersect any one d-star. Routing with congestion has been a very fruitful area of research, particularly with regards to multi-commodity flows; picking just a few examples from the vast literature, we point the interested reader to the work of Andrews et al. ([3]), Chekuri et al. ([9]), and Chuzhoy et al. ([17]).

References

[1] M. Agrawal, N. Kayal, and N. Saxena. Primes is in P. *Annals of Mathematics*, 160(2):781–793, 2004.

[2] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, 1993.

[3] M. Andrews, J. Chuzhoy, V. Guruswami, S. Khanna, K. Talwar, and L. Zhang. Inapproximability of edge-disjoint paths and low congestion routing on undirected graphs. Technical Report TR14-113, Electronic Colloquium on Computational Complexity, 2007.

[4] T. Asano, S. Kikuchi, and N. Saito. An efficient algorithm to find a hamiltonian circuit in a 4-connected maximal planar graph. *Graph Theory and Algorithms*, pages 182–195, 1981.

[5] G. Baier, E. Kohler, and M. Skutella. The k-splittable flow problem. *Algorithmica*, 42:231–248, 2005.

[6] D. Bienstock and S. Mattia. Using mixed-integer programming to solve power grid blackout problems. *Discrete Optimization*, 4:115–141, 2007.

[7] B. Bollobás. *Modern Graph Theory*. Springer, New York, 1998.

[8] J. Bourgain. On Lipschitz embedding of finite metric spaces in Hilbert spaces. *Israeli Journal of Mathematics*, 52:46–52, 1985.

[9] C. Chekuri, S. Khanna, and B. Shepherd. Edge-disjoint paths in planar graphs with constant congestion. *SIAM Journal on Computing*, 39:281–301, 2009.

[10] J. Chen, R. Kleinberg, L. Lovasz, R. Rajaraman, R. Sundaram, and A. Vetta. (almost) tight bounds and existence theorems for confluent flows. In *Proceedings of the 36th ACM Symposium on Theory of Computing (STOC)*, pages 529–538, 2004.

[11] J. Chen, R. Rajaraman, and R. Sundaram. Meet and merge: approximation algorithms for confluent flow. In *Proceedings of the 35th ACM Symposium on Theory of Computing (STOC)*, pages 373–382, 2003.

[12] J. Cheriyan, A. Sebo, and Z. Szigeti. Improving on the 1.5-approximation of a smallest 2-edge connected spanning subgraph. *SIAM Journal on Discrete Mathematics*, 14:170–180, 2001.

[13] J. Cheriyan and R. Thurimella. Approximating minimum-size k-connected spanning subgraphs via matching. *SIAM Journal on Computing*, 30:528–560, 2000.

[14] J. Cheriyan, S. Vempala, and A. Vetta. Approximation algorithms for minimum-cost k-vertex connected subgraphs. *SIAM Journal on Computing*, 32(4):1050–1055, 2003.

[15] J. Cheriyan and A. Vetta. Approximation algorithms for network design with metric costs. In *Proceedings of the 37th ACM Symposium on Theory of Computing*, pages 167–175, 2005.

[16] M. Chudnovski, N. Robertson, P. Seymour, and R. Thomas. The strong perfect graph theorem. *Annals of Mathematics*, 164:51–229, 2006.

[17] J. Chuzhoy, V. Guruswami, S. Khanna, and K. Talwar. Hardness of routing with congestion in directed graphs. In *Proceedings of the 39th ACM Symposium on Theory of Computing (STOC)*, pages 165–178, 2007.

[18] V. Chvátal. Star-cutsets and perfect graphs. *Journal of Combinatorial Theory B*, 39:189–199, 1985.

[19] A. Clementi, P. Penna, and R. Silvestri. Hardness results for the power range assignment problem in packet radio networks. In *Proceedings of the*

3rd International Workshop on Approximation Algorithms for Combinatorial Optimization, pages 197–208, 1999.

[20] G. Cornuéjols. The strong perfect graph theorem. *Optima*, 70:2–6, 2003.

[21] P. Crucitti, V. Latora, and M. Marchiori. Model for cascading failures in complex networks. *Physical Review E*, 69(4):045104, 2004.

[22] R. Diestel. *Graph Theory*. Springer, 2005.

[23] Y. Dinitz, N. Garg, and M. Goemans. On the single-source unsplittable flow problem. *Combinatorica*, 19:17–41, 1999.

[24] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45:634–652, 1998.

[25] M. Fellows. The robertson-seymour theorems: a survey. *Contemporary Mathematics*, 89, 1989.

[26] S. Finbow and B. Hartnell. On designing a network to defend against random attacks of radius two. *Networks*, 19:771–792, 1989.

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

[28] S. Finbow and G. MacGillivray. The firefighter problem: a survey of results, directions and questions. *The Australasian Journal of Combinatorics*, 43, 2009.

[29] L. Ford and D. Fulkerson. A simple algorithm for finding maximal network flows and an application to the Hitchcock problem. *Canadian Journal of Mathematics*, 9:210–218, 1957.

[30] S. Fortune, J. Hopcroft, and J. Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10:111–121, 1980.

[31] A. Frank. Connectivity augmentation problems in network design. In J. Birge and K. Murty, editors, *Mathematical Programming: State of the Art*, pages 34–63. University of Michigan Press, 1994.

[32] A. Frank and T. Jordan. Minimal edge-coverings of pairs of sets. *Journal of Combinatorial Theory B*, 65:73–110, 1995.

[33] H. Gabow. A matroid approach to finding edge connectivity and packing arborescences. *Journal of Computer and System Sciences*, 50:259–273, 1995.

[34] H. Gabow. An ear decomposition approach to approximating the smallest 3-edge connected spanning subgraph of a multigraph. *SIAM Journal on Discrete Mathematics*, 18(1):41–70, 2004.

[35] H. Gabow. Using expander graphs to find vertex-connectivity. *Journal of the ACM*, 53(5):800–844, 2006.

[36] H. Gabow, M. Goemans, E. Tardos, and D. Williamson. Approximating the smallest k-edge connected spanning subgraph by LP-rounding. In *Proceedings of the 16th Symposium on Discrete Algorithms*, pages 562–571, 2005.

[37] M. Garey and D. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[38] N. Garg, V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM Journal on Computing*, 25:235–251, 1996.

[39] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20:374–387, 1998.

[40] G. Gunther. Neighbour-connectivity in regular graphs. *Discrete Applied Mathematics*, 11(3):233–243, 1985.

[41] G. Gunther and B. Hartnell. On minimizing the effects of betrayals in a resistance movement. *Congressus Numerantium*, 22:285–306, 1978.

[42] G. Gunther and B. Hartnell. Optimal k-secure graphs. *Discrete Applied Mathematics*, 2:225–231, 1980.

[43] G. Gunther and B. Hartnell. Security of underground resistance movements. In N. Memon, J. Farley, D. Hicks, and T. Rosenorn, editors, *Mathematical Methods in Counterterrorism*, pages 185–204. Springer, 2009.

[44] G. Gunther, B. Hartnell, and R. Nowakowski. Neighbour-connected graphs and projective planes. *Networks*, 17(2):241–247, 2006.

[45] V. Guruswami, S. Khanna, R. Rajaraman, B. Shepherd, and M. Yannakakis. Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. *Journal of Computer and System Sciences*, 67:473–496, 2003.

[46] M. Hajiaghayi, G. Kortsarz, V. Mirrokni, and Z. Nutov. Power optimization for connectivity problems. In *Proceedings of the 11th Conference on Integer Programming and Combinatorial Optimization*, pages 349–361, 2005.

[47] F. Harary. *Graph Theory.* Perseus Books, 1963.

[48] B. Hartnell. The optimum defense against random subversions in a network. In *Proceedings of the 10th Southeast conference on Combinatorics, Graph Theory and Computing*, pages 494–499, 1979.

[49] B. Hartnell. Firefighter! an application of domination. Presentation. 24th Manitoba Conference on Combinatorial Mathematics and Computing, 1995.

[50] B. Hartnell and W. Kocay. On minimal neighbourhood-connected graphs. *Discrete Mathematics*, 92:95–105, 1991.

[51] J. Hastad. Clique is hard to approximate within $n^{1-\varepsilon}$. In *Proceedings of the 37th IEEE annual symposium on foundations of computer science*, pages 627–636, 1996.

[52] A. Hatcher. *Algebraic Topology.* Cambridge University Press, 2001.

[53] R. Hayward. *Two classes of perfect graphs.* PhD thesis, McGill University, 1986.

[54] P. Holme. Efficient local strategies for vaccination and network attack. *Europhysics Letters*, 68:908–914, 2004.

[55] K. Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[56] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *KDD '03: Proceedings of the 9th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 137–146, New York, NY, USA, 2003. ACM.

[57] S. Khuller. Approximation algorithms for finding highly connected subgraphs. In D. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, pages 236–265. PWS Pub. Co., Boston, 1995.

[58] S. Khuller and U. Vishkin. Biconnectivity approximations and graph carvings. *Journal of the ACM*, 41(2):214–235, 1994.

[59] A. King and G. MacGillivray. The firefighter problem for cubic graphs. *Discrete Mathematics*, 310(3):614–621, 2010.

[60] L. Kirousis, E. Kranakis, D. Krizanc, and A. Pelc. Power consumption in packet radio networks. *Theoretical Computer Science*, 243:289–305, 2000.

[61] P. Klein, S. Plotkin, and S. Rao. Excluded minors, network decomposition and multicommodity flow. In *Proceedings of the 25th Symposium on the Theory of Computing (STOC)*, pages 682–690, 1993.

[62] J. Kleinberg. Single-source unsplittable flow. In *Proceedings of the 37th on Foundations of Computer Science (FOCS)*, pages 68–77, 1996.

[63] R. Koch and I. Spenke. Complexity and approximability of k-splittable flows. *Theoretical Computer Science*, 369:338–347, 2006.

[64] G. Kortsarz, R. Krauthgamer, and J. Lee. Hardness of approximation algorithm for vertex-connectivity network design problems. *SIAM Journal on Computing*, 33(3):704–720, 2004.

[65] G. Kortsarz and Z. Nutov. Approximation algorithm for k-node connected subgraphs via critical graphs. In *Proceedings of the 36th ACM Symposium on Theory of Computing*, pages 138–145, 2004.

[66] G. Kortsarz and Z. Nutov. Approximating minimum cost connectivity problems. In T. Gonzalez, editor, *Handbook on Approximation Algorithms and Metaheuristics*, chapter 58. Chapman & Hall / CRC, 2007.

[67] D. Kratsch and J. Spinrad. Between $O(nm)$ and $O(n^\alpha)$. In *Proceedings of the 14th annual ACM-SIAM symposium on discrete algorithms*, pages 709–716, 2003.

[68] C. Kuratowski. Sur le problème des courbes gauches en topologie. *Fund. Math.*, 15:271–283, 1930.

[69] T. Leighton and S. Rao. Multi-commodity max-flow min-cut theorems and their use in approximation algorithms. *Journal of the ACM*, 46:787–832, 1999.

[70] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15:215–245, 1995.

[71] C. McDiarmid, B. Reed, A. Schrijver, and B. Shepherd. Non-interfering dipaths in planar digraphs. Technical report, Centrum voor Wiskunde en Informatica, 1991.

[72] C. McDiarmid, B. Reed, A. Schrijver, and B. Shepherd. Non-interfering network flows. In *Proceedings of the 3rd Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 245–257. Springer-Verlag, 1992.

[73] K. Menger. Zur allgemeinen Kurventheorie. *Fund. Math.*, 10:96–115, 1927.

[74] A. Motter. Cascade control and defense in complex networks. *Physical Review Letters*, 93(9):098701, 2004.

[75] A. Motter and Y. Lai. Cascade-based attacks on complex networks. *Physical Revue E*, 66(6):065102, 2002.

[76] J. Munkres. *Topology.* Prentice Hall, 2000.

[77] H. Nagamochi and T. Ibaraki. *Algorithmic aspects of graph connectivity.* Cambridge University Press, 2008.

[78] G. Naves, N. Sonnerat, and A. Vetta. Flows on disjoint paths. Submitted.

[79] Z. Nutov. An almost $O(\log k)$-approximation for k-connected subgraphs. In *SODA '09: Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 912–921, 2009.

[80] H. Okamura and P. Seymour. Multicommodity flows in planar graphs. *Journal of Combinatorial Theory B*, 31:75–81, 1981.

[81] B. Reed. Skew partitions in perfect graphs. *Discrete Applied Mathematics*, 156:1150–1156, 2008.

[82] N. Roussopoulos. A max{m,n} algorithm for determining the graph H from its line graph G. *Information Processing Letters*, 2:108–112, 1973.

[83] F. Salazar and M. Skutella. Single-source k-splittable min-cost flows. *Operations Research Letters*, 37:71–74, 2009.

[84] A. Schrijver. *Combinatorial Optimization.* Springer, 2003.

[85] P. Seymour. On odd cuts and plane multicommodity flows. *Proceedings of the London Mathematical Society*, 42(3):178–192, 1991.

[86] B. Shepherd. Single-sink multicommodity flow with side constraints. In W. Cook, L. Lovasz, and J. Vygen, editors, *Research Trends in Combinatorial Optimization*, pages 429–450. Springer, 2009.

[87] N. Sonnerat and A. Vetta. Edge immunization in planar graphs. In preparation.

[88] N. Sonnerat and A. Vetta. A minmax theorem for non-interfering flows and galaxy cutsets in planar graphs. Submitted.

[89] N. Sonnerat and A. Vetta. Network connectivity and malicious attacks. Submitted.

[90] N. Sonnerat and A. Vetta. Galaxy cutsets in graphs. *Journal of Combinatorial Optimization*, 19(3):415–427, 2010.

[91] Nicolas Sonnerat and Adrian Vetta. Defending planar graphs against star-cutsets. *Electronic Notes in Discrete Mathematics*, 34:107 – 111, 2009. European Conference on Combinatorics, Graph Theory and Applications (EuroComb 2009).

[92] 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.

[93] V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[94] S. Vempala and A. Vetta. Factor 4/3 approximations for minimum two-connected subgraphs. In *Proceedings of the 3rd International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 262–273, 2000.

[95] H. Whitney. A theorem on graphs. *Annals of Mathematics*, 32(2):378–390, 1931.