Approximating Minimum-Size 2-Edge-Connected and 2-Vertex-Connected Spanning Subgraphs

by

Vishnu V. Narayan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2017

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

We study the unweighted 2-edge-connected and 2-vertex-connected spanning subgraph problems. A graph is 2-edge-connected if it is connected on removal of an edge, and it is 2-vertex-connected if it is connected on removal of a vertex. The problem of finding a minimum-size 2-edge-connected (or 2-vertex-connected) spanning subgraph of a given graph is NP-hard.

We present a $\frac{4}{3}$-approximation algorithm for unweighted 2ECSS on 3-vertex-connected input graphs, which matches the best known approximation ratio due to Sebő and Vygen for the general unweighted 2ECSS problem, but our analysis is with respect to the $D2$ lower bound. We also give a $\frac{17}{12}$-approximation algorithm for unweighted 2VCSS on graphs of minimum degree at least 3, which is lower than the best known ratios of $\frac{3}{2}$ by Garg, Santosh and Singla and $\frac{10}{7}$ by Heeger and Vygen for the general unweighted 2VCSS problem. These algorithms are accompanied by new theorems about the known lower bounds.

Acknowledgements

Most of all, I would like to thank my friends and family for all of their support and encouragement.

I am very lucky to have a great supervisor in Joseph Cheriyan, and am thankful for everything that he has done for me and for everyone else over the past few years.

I'm thankful for the amazing C&O department and all of its faculty and staff. I'm especially thankful for all of my course instructors and office colleagues. I've learned more things than I ever expected to learn in these last two years at Waterloo.

Finally, I'm very grateful to have William Cook and Jochen Koenemann read my thesis and provide invaluable comments and suggestions.

Table of Contents

List of Figures

Chapter 1

Introduction

Real world networks are prone to failures. At a high level, the goal of *survivable network design* is to design networks that survive the failure of some of their constituents, which are often nodes or connections between nodes. More formally, survivable network design problems are a collection of problems where the goal is to design networks of minimum cost, subject to some survivability constraints. Usually, we require that some connectivity property be satisfied by these networks.

Most networks can very naturally be modelled as graphs. A graph G is an ordered pair (V, E), where V is a set of *vertices* or *nodes* and E is a set of *edges*, which are unordered pairs of these vertices. A natural question to ask is whether we can design networks of minimum cost that can survive the failure of one vertex or one edge. Graphs that survive these failures are called *2-vertex-connected* and *2-edge-connected* graphs respectively. The main focus of this thesis is the problem of constructing minimum-size 2-edge-connected and 2-vertex connected graphs on some vertex set, when we are given the set of possible edges that we are allowed to use. Since these problems are NP-hard, we focus on the problem of finding approximate solutions to them.

Formally, a graph is connected if there is a path between every pair of vertices in the graph. It is 2-edge-connected if the graph obtained by removing any edge is connected, and it is 2-vertex-connected (or 2-connected) if the graph obtained by removing any vertex, and all edges incident on that vertex, is connected. Equivalently, a graph is 2-edge-connected (2-vertex-connected) if and only if for every pair of vertices u, v in the graph there are 2 edge-disjoint (vertex-disjoint, respectively) paths in the graph with endpoints at u and v. Clearly, every 2-vertex-connected graph is 2-edge-connected.

This notion extends to integers larger than 2. More generally, a graph is k-edge-

connected (respectively, k-vertex-connected) if there does not exist a subset of $k-1$ edges (respectively, vertices) whose removal disconnects the remainder of the graph.

The k-edge-connectivity problem is the following problem: given a graph G, find a spanning subgraph of G of minimum size (number of edges) that is k-edge-connected. k-vertex-connectivity problem asks to find a spanning subgraph of minimum size that is k-vertex-connected. For $k = 1$, these problems reduce to the problem of finding a smallest connected subgraph that is a spanning tree. It is well known that this case can be solved in polynomial time.

The case $k = 2$ is of special interest, since this is the smallest value of k for which the problem is NP-hard (see e.g. [13]). NP-hardness for this case can be seen by a reduction from the Hamiltonian cycle problem: a graph has a Hamiltonian cycle if and only if the size of a minimum-size 2-edge-connected (or 2-vertex-connected) spanning subgraph is equal to the number of vertices in the graph.

The 2-edge-connectivity problem (also called the 2-edge-connected spanning subgraph problem, or 2ECSS) and 2-vertex-connectivity problem (2VCSS) have been well studied for this reason. Several constant-factor approximation algorithms have been found for both problems, and we have seen many improvements as recently as in the past five years.

In this thesis, we study the 2ECSS and 2VCSS problems. We describe some of the known lower bounds on the size of an optimal solution to these problems, and present two new approximation algorithms for special cases of these problems.

The first of these algorithms is for the 2ECSS problem: we present a $\frac{4}{3}$-approximation algorithm for the special case of 2ECSS where the instance is 3-vertex-connected. This ratio is equal to the lowest ratio known at present for the general 2ECSS problem (due to Sebő and Vygen [28]), but uses a different lower bound (the $D2$ lower bound, described in Section 2.2). Our result proves that the size of an optimal 2ECSS is within a factor of $\frac{4}{3}$ of the size of a minimum-$D2$ of G when G is 3-vertex-connected.

The second algorithm achieves a ratio of $\frac{17}{12}$ for the restricted version of the 2VCSS problem where the instance has no vertices of degree 2 (and hence has minimum degree at least 3). This improves on the lowest ratios known at present for the general problem (which is $\frac{3}{2}$, due to Garg, Santosh and Singla [14]; a recent result of Heeger and Vygen [16], to appear in SIAM J. Discrete Math., improves this to $\frac{10}{7}$). Our second result is accompanied by an interesting structural theorem on the existence of ear-decompositions with certain properties: we show that every 2-vertex-connected graph with no vertices of degree 2 has a *nice* ear-decomposition (see [28]) that is also open.

The specializations of 2ECSS and 2VCSS for which we present these algorithms are

also NP-hard. Garey, Johnson and Tarjan [12] show that 2ECSS is NP-hard in 3-vertex-connected graphs. Lemma A.1 in the appendix shows that the specialized version of 2VCSS (with minimum degree at least 3) is NP-hard.

1.1 Related Work

There are many natural generalizations of these problems. For instance, the weighted versions of the 2-edge-connectivity and 2-vertex-connectivity problems introduce nonnegative edge-weights ($c_e \in \mathbb{R}_{\geq 0} : e \in E$), and ask to find a subgraph that satisfies the corresponding connectivity property and minimizes the sum of these edge-weights over all the edges in the subgraph. The unweighted case can be obtained by setting $c_e = 1$ for every edge $e \in E$.

Another generalization of the 2-edge-connectivity problem is the Generalized Steiner Network Problem (GSNP). The input to this problem is an undirected multigraph where each edge e has weight $c_e \in \mathbb{R}_{\geq 0}$, and for each pair of vertices $u, v \in V$, there is a connectivity requirement $r_{u,v} \in \mathbb{Z}_{\geq 0}$. A feasible solution to this problem is a subgraph that has at least $r_{u,v}$ edge-disjoint paths between u and v, and the problem asks to find a feasible solution of minimum cost. It is easy to see that if we set $r_{u,v} = 2$ for every pair of vertices $u, v \in V$, then this problem reduces to the 2-edge-connectivity problem. Similar problems have been defined and studied for the case where the connectivity requirement asks for vertex-disjoint paths.

Both the 2ECSS and 2VCSS problems have been well studied. For the 2ECSS problem, Khuller and Vishkin [22] gave a $\frac{3}{2}$-approximation algorithm. Cheriyan, Sebő and Szigeti [4] improved the approximation ratio to $\frac{17}{12}$. Recently, Sebő and Vygen [28] gave a $\frac{4}{3}$-approximation algorithm for this problem. Better approximation ratios have been claimed, but to the best of our knowledge, no complete proof has been published. For the weighted version of this problem, several 2-approximation algorithms exist. Khuller and Vishkin [22] gave a 2-approximation algorithm for the weighted problem. The 2-approximation due to Jain [17] for the more general GSNP also gives a 2-approximation for the weighted 2-edge-connected spanning subgraph problem. No algorithm with an approximation ratio better than 2 is known for the weighted 2-edge-connectivity problem.

For the 2VCSS problem, Khuller and Vishkin [22] gave a $\frac{5}{3}$-approximation algorithm. Garg, Santosh and Singla [14] improved this ratio to $\frac{3}{2}$. Very recently, Heeger and Vygen [16] obtained an approximation ratio of $\frac{10}{7}$ for the general unweighted 2VCSS problem. Our work was carried out independently at the same time, and achieves a slightly better ratio

for a specialization of this problem. For the weighted version of this problem, a ratio of 2 has been achieved by several authors (see e.g. [23]). This ratio has not yet been improved.

Vempala and Vetta [29] reported a ratio of $\frac{4}{3}$ for both problems, but to the best of our knowledge, their analysis for both problems remains incomplete. This paper introduced the $D2$-lower bound on the size of a 2ECSS solution that we use in our algorithm in Chapter 3. Jothi, Raghavachari and Varadarajan [19] reported a ratio of $\frac{5}{4}$ for 2VCSS, and Gubbala and Raghavachari [15] announced a result of $\frac{9}{7}$, but subsequently both results have been withheld from publication (see [15] and [16]).

Both problems have been shown to be APX-hard (see [5] and [9]) in the general case.

1.2 Organization of the Thesis

In Chapter 2, we discuss some preliminaries, define the notation and terms used in this thesis, and describe some of the known lower bounds on the cost of an optimal 2ECSS or 2VCSS solution.

In Chapter 3, we present a $\frac{4}{3}$-approximation algorithm for 2ECSS, restricted to instances where the graph is 3-vertex-connected. We prove that the approximation guarantee is satisfied with respect to the $D2$ lower bound for this class of graphs.

In Chapter 4, we present a $\frac{17}{12}$-approximation algorithm for 2VCSS, restricted to instances where the graph has no vertices of degree 2. This algorithm is based on the results of Sebő and Vygen [28] for approximating 2ECSS. We show that an approach similar to the one in [28], but applied to the 2-vertex-connectivity problem, fails for general graphs, but an adaptation of this approach succeeds for graphs without vertices of degree 2.

Chapter 2

Preliminaries and Lower Bounds

In this chapter we give an introduction to the 2ECSS and 2VCSS problems, and describe the notation and terminology used in this thesis. We also present some high-level ideas that will be useful for the rest of the thesis. Finally, we describe in detail the known lower bounds on the cost of an optimal solution to these problems.

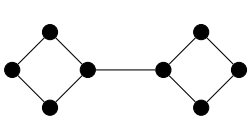
2.1 Preliminaries

We begin with some basic definitions and terminology.

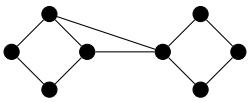
Unless stated otherwise, we assume that graphs are simple and have no loops or multi-edges (otherwise we call them multigraphs). Let $G = (V, E)$ be a graph. For some vertex $v \in V$, we denote by $G \setminus v$ the graph obtained from G by removing the vertex v and all of the edges incident on v. For some edge $e \in E$ with endpoints u and v, we use e and uv interchangeably to refer to the edge e, and we denote by $G \setminus e$ or by $G \setminus uv$ the graph obtained from G by removing the edge e.

A *bridge* of a graph is an edge whose removal disconnects the graph. A *cut-vertex* is a vertex whose removal disconnects the remainder of the graph. If a graph contains a bridge, it is not 2-edge-connected, and if it contains a cut-vertex, it is not 2-vertex-connected. Figure 2.1 shows some examples related to these definitions.

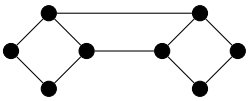
A *path* P in G is a nonempty graph of the form $V = \{x_0, x_1, \ldots, x_k\}$, $E = \{x_0x_1, x_1x_2, \ldots, x_{k-1}x_k\}$, where the x_i are all distinct. We call x_0 and x_k the *endpoints* of P and $x_1, \ldots, x_{k-1}$ the *internal vertices* of P. The *length* of a path is the number of edges in the path. Let u, v be two vertices in V. A *u, v-path* in G is a path with endpoints at u and v.



(a) A graph with a bridge



(b) A 2-edge-connected graph with a cut-vertex



(c) A 2-vertex-connected graph

Figure 2.1: 2-edge-connected and 2-vertex-connected graphs

Formally, a graph $G = (V, E)$ is *2-edge-connected* if $|V| \geq 3$ and for every edge $e \in E$, $G \setminus e$ is connected. G is *2-vertex-connected* if $|V| \geq 3$ and for every vertex $v \in V$, $G \setminus v$ is connected.

Let $G = (V, E)$ be an unweighted 2-edge-connected graph. The 2-edge-connected spanning subgraph problem (2ECSS) on the instance G is the problem of finding a 2-edge-connected spanning subgraph of G with the minimum number of edges. The 2-vertex-connected spanning subgraph problem (2VCSS) on the instance G is the problem of finding a 2-vertex-connected spanning subgraph of G with the minimum number of edges.

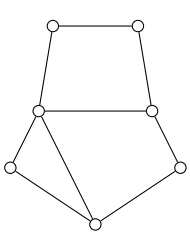
As a preliminary remark, we note that for the 2ECSS problem, we may assume that the input graph is 2-vertex connected: an optimal solution to 2ECSS on G is a union of optimal solutions to 2ECSS on its blocks (see Proposition 1.4 in [28]).

An *ear* of G is a subgraph P of G with at least one edge, such that P is either a path (with two end vertices) or a cycle with one vertex chosen as its end vertex. An ear is *open* if it is a path and *closed* otherwise. It is *trivial* if it has a single edge, *short* if it has 2 or 3 edges, and *long* otherwise. A *k-ear* is an ear with exactly k edges. A k-ear is *even* if k is even, and is *odd* otherwise. At times, we abuse the notation for trivial ears, and write uv for the ear corresponding to the path containing the vertices u, v and the edge uv.

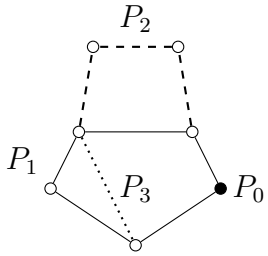
Let P be an ear of G. The vertices of P that are not endpoints of P are called internal vertices of P, and their set is denoted by $in(P)$.

An *ear-decomposition* of G is a sequence P_0,P_1,. . . ,P_k, where P_0 is a vertex and P_1,. . . ,P_k are ears such that P_i shares exactly its one endpoint (or two endpoints in case P_i is open) with the vertices of $P_0 \cup \ldots \cup P_{i-1}$. We refer the reader to Figure 2.2 for an example. The black node represents the *starting vertex* P_0, the solid edges represent the closed *starting ear* P_1, the dashed edges represent the open ear P_2, and the dotted edge represents the trivial ear P_3.

An ear-decomposition D of G is *open* if every ear in D except the ear P_1 is open. The following lemma due to Whitney [30] characterizes the graphs that have ear-decompositions (respectively, open ear-decompositions). We state the lemma in 2 parts.



(a) A graph G



(b) An ear-decomposition of G

Figure 2.2: An ear-decomposition

Lemma 2.1.

1. *A graph is 2-edge-connected if and only if it has an ear-decomposition.*
2. *A graph is 2-vertex-connected if and only if it has an open ear-decomposition.*

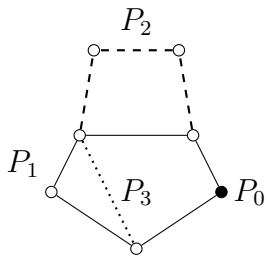
Proof. We will prove part 2, the proof of part 1 is similar and we omit it. It is easy to see that a graph with an open ear-decomposition D is 2-vertex-connected.

For the other direction, we start with a 2-vertex-connected graph and construct an ear-decomposition $D = (P_0, P_1, \ldots, P_k)$. We choose an arbitrary start vertex, and let P_0 be this vertex. Since G is 2-vertex-connected, it contains a cycle containing this vertex. We use such a cycle to construct the starting ear P_1. We then repeatedly add an ear to our partial ear-decomposition D in the following manner. Let $V' \subseteq V$ be the union of the internal vertices of the ears in D. As long as $V' \neq V$, there exists an edge $e = uv$ in G with exactly one endpoint (say u) in V'. Let P be a path that contains e, with one endpoint at u and the other endpoint at some vertex $x \in V' \setminus \{u\}$, such that the internal vertices of P are in $V \setminus V'$. Clearly, such a path exists: if not, u is a cut-vertex of G. We construct an open ear out of this path and add it to D. When $V' = V$, we add each of the remaining edges of G as a trivial ear. Then D is an open ear-decomposition of G. □

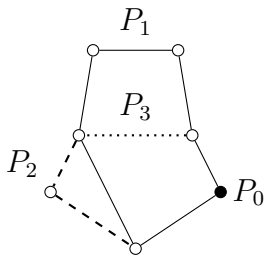
Observe that a given graph G can have multiple ear-decompositions. Figure 2.3 shows two different ear-decompositions of the same graph, using the example from Figure 2.2. This fact, along with the above lemma, hints at a possible technique for finding 2-edge-connected and 2-vertex-connected spanning subgraphs of low cost. Let $G = (V, E)$ be the input graph. Suppose that we find an ear-decomposition D of G. We discard the trivial ears of D, and add the remaining edges of D (those in nontrivial ears) to our

solution. The resulting graph is 2-edge-connected (2-vertex-connected if D is an open ear-decomposition). We can exploit this fact by attempting to compute an ear-decomposition that has a relatively large number of trivial ears. The nontrivial ears of this decomposition correspond to a solution of low cost.

In section 2.2, we will further exploit the fact that increasing the number of trivial ears decreases the cost of our solution to give a lower bound on the cost of an optimal solution. Both the $\frac{17}{12}$-approximation for 2ECSS due to Cheriyan, Sebő and Szigeti [4], and the $\frac{4}{3}$-approximation for 2ECSS due to Sebő and Vygen [28], make use of ear-decompositions to construct a solution. The techniques used in Chapter 4 are inspired by those in [4] and [28]. In this chapter, we present a $\frac{17}{12}$-approximation algorithm for 2VCSS for instances without degree-2 vertices, that uses ear-decompositions to compute a feasible solution.



(a) An ear-decomposition D_1 with no even ears



(b) An ear-decomposition D_2 with 2 even ears

Figure 2.3: Two ear-decompositions

2.2 Lower bounds on the cost of an optimal solution

We denote by $OPT(G)$ the size of an optimal 2ECSS solution for an instance G. Since there is no easy way to compute $OPT(G)$, it is important to find a good lower bound on its value. In this section, we will describe several lower bounds on $OPT(G)$.

We denote by $OPT_{2VC}(G)$ the size of an optimal 2VCSS solution for an instance G. Since every 2-vertex-connected spanning subgraph of a graph G is also 2-edge-connected, every lower bound on the cost of an optimal 2ECSS solution also bounds the cost of an optimal 2VCSS solution. The lower bounds that we discuss are all lower bounds on $OPT(G)$, and are therefore also valid for $OPT_{2VC}(G)$.

2.2.1 A Linear Programming Relaxation

There is a natural LP relaxation of the 2ECSS problem, and the optimal value of this LP gives a lower bound on $OPT(G)$. For any subset S of V, let $\delta(S)$ be the set of edges in E with exactly one endpoint in S, and $x(\delta(S))$ be the sum of the x-values over these edges.

$$\begin{array}{ll} \min & \sum_{e \in E} x_e \\ \text{subject to} & x(\delta(S)) \geq 2 \text{ for all } \varnothing \subsetneq S \subsetneq V \\ & x_e \geq 0 \qquad \text{for all } e \in E \end{array}$$

In the above LP, V and E are the vertex and edge sets of the input graph respectively. We have a nonnegative variable $x_e \in \mathbb{R}_{\geq 0}$ for each edge e. The cut-constraints impose the restriction that the edges in every cut have a total x-value of at least 2. Since any 2-edge-connected spanning subgraph of G has at least 2 edges in every cut, it defines a feasible solution for this LP.

If we let $LP(G)$ denote the optimal value of this LP for graph G, then $LP(G) \leq OPT(G)$, since an optimal solution is feasible for the LP.

We can obtain a simple combinatorial lower bound from the constraints of this LP. Consider the cut-constraints corresponding to the singletons. For a vertex v, the constraint corresponding to the set $\{v\}$ is of the form $x(\delta(\{v\})) \geq 2$. If we sum these inequalities over all the vertices, we find that the objective value of every feasible LP solution is at least $|V|$.

The next two subsections describe two (usually) stronger lower bounds on the value of $LP(G)$.

2.2.2 The L_φ-Lower Bound

Recall that a given graph G can have multiple ear-decompositions. It is possible that these ear-decompositions differ in the number of even ears (see Figure 2.3). We denote by $\varphi(G)$ the minimum (possible) number of even ears of an ear-decomposition, over all of the ear decompositions of G.

We begin with the following definitions. An ear-decomposition D of G is *evenmin* if the number of even ears in D is equal to $\varphi(G)$. An ear-decomposition is *odd* if it has no even ears. We define an indicator function φ from the ears of D to the set $\{0, 1\}$ as follows. For any ear P, we let $\varphi(P) = 1$ if P is even and $\varphi(P) = 0$ otherwise. A nontrivial

ear P is a *pendant* ear if no other nontrivial ear has an endpoint in $in(P)$, otherwise it is non-pendant.

A graph $G = (V, E)$ is called *factor-critical* if for every node $v \in V$, $G \setminus v$ has a perfect matching. Lovász [24] showed an interesting characterization of the factor-critical graphs by their ear-decompositions.

Theorem 2.2. *(Lovász) A graph is factor-critical if and only if it has an odd ear- decomposition.*

Lovász and Plummer [25] also showed that it is possible to naturally combine this theorem with Lemma 2.1.

Theorem 2.3. *(Lovász, Plummer) A 2-vertex-connected factor critical graph has an open odd ear-decomposition.*

Their proof of this theorem is constructive, and gives a method to compute an open odd ear-decomposition in polynomial time.

Let $G = (V, E)$ be a graph and $T \subseteq V$ with $|T|$ even. A set $F \subseteq E$ is called a *T-join* if T is exactly the set of odd-degree vertices of the graph (V, F). The next result is due to Frank [10].

Theorem 2.4. *(Frank) Let $G = (V, E)$ be a 2-edge-connected graph. Then there exists $T \subseteq V$, $|T|$ even, such that the minimum cardinality of a T-join is $\frac{1}{2}(|V(G) + \varphi(G) - 1)$. Such a T and an ear-decomposition with $\varphi(G)$ even ears can be computed in time $O(|V| \cdot |E|)$.*

Cheriyan, Sebő and Szigeti [4], in their result which gave a $\frac{17}{12}$-approximation algorithm for 2ECSS, used the above theorems to prove the following results. The first of these results is used in the approximation algorithms in [4] and [28], as well as in the algorithm presented in Chapter 4 of this thesis. It states that for any 2-vertex-connected graph G, there exists an ear-decomposition of G that is both open and evenmin. Further, such an ear-decomposition can be computed in polynomial time.

Theorem 2.5. *(Cheriyan, Sebő, Szigeti) Let G be a 2-vertex-connected graph. An open ear-decomposition D of G having $\varphi(G)$ even ears can be computed in time $O(|V| \cdot |E|)$.*

Proof. By Frank's Theorem (Theorem 2.4), we can compute an evenmin ear-decomposition D in time $O(|V| \cdot |E|)$. Starting with D, we subdivide an edge on each even ear to obtain an odd ear-decomposition D' of the resulting graph G'. Since D' is odd, by Theorem 2.2,

G' is factor critical. Since this subdivision does not introduce cut-vertices, G is 2-vertex-connected. Hence we can compute an open odd ear-decomposition $\bar{D}$ of G' by Theorem 2.3. Finally, if we undo the subdivisions, this ear-decomposition remains open (since both edges resulting from a subdivision remain in the same ear of $\bar{D}$, and contracting an edge in a nontrivial open ear does not introduce cut vertices). Undoing the subdivisions also reintroduces $\varphi(G)$ even ears. The resulting ear-decomposition of G is both open and evenmin. □

The second result gives a new lower bound on the cost of an optimal solution to 2ECSS for a graph G, in terms of the value of $\varphi(G)$. For any 2-edge-connected graph G, let $L_\varphi(G) := |V| + \varphi(G) - 1$.

Theorem 2.6. *(Cheriyan, Sebő, Szigeti) Let $G = (V, E)$ be a 2-edge-connected graph. Then $OPT(G) \geq L_\varphi(G)$.*

Proof. Let H be a 2ECSS of G. We claim that every ear-decomposition of H has at least $\varphi(G)$ even ears. Suppose not, and let D be an ear-decomposition of H with fewer than $\varphi(G)$ even ears. If we add all edges of $E(G) \setminus E(H)$ as trivial ears to the end of D, then the resulting sequence is an ear-decomposition of G with less than $\varphi(G)$ even ears, a contradiction. Consequently, every ear-decomposition of an optimal 2ECSS has at least $\varphi(G)$ even ears, and hence at least $\varphi(G)$ nontrivial ears.

Let $\bar{H}$ be an optimal 2ECSS of G and $\bar{D}$ be an ear-decomposition of $\bar{H}$. Let P be a nontrivial ear of $\bar{D}$. If P is the starting cycle P_1, then $|E(P)| = |in(P)|$. Otherwise, $|E(P)| = |in(P)| + 1$. Summing over all nontrivial ears of $\bar{D}$, of which there are at least $\varphi(G)$, we get $|E(\bar{H})| \geq |V| + \varphi(G) - 1$. □

In their recent paper, Sebő and Vygen [28] use the same lower bound as part of their analysis, but prove a stronger result: that this lower bound is, in fact, a lower bound on $LP(G)$. Their proof involves more work, and we direct the reader to Section 4, Theorem 5 of [28] for this proof. Observe that the above lower bound is stronger than the naïve lower bound of $|V|$ for graphs G with $\varphi(G) > 1$. Figure 2.4 shows an example of such a graph G (with $\varphi(G) = 2$).

2.2.3 The L_μ-Lower Bound

To achieve the approximation guarantee of $\frac{4}{3}$ for 2ECSS, Sebő and Vygen [28] used two lower bounds in conjunction - the L_φ-bound, and a second lower bound on $LP(G)$ that

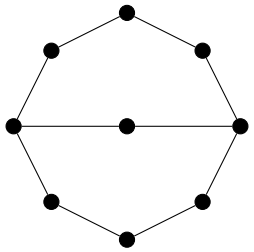


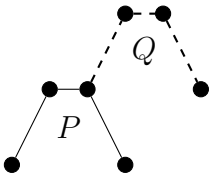
Figure 2.4: A graph G with $\varphi(G) = 2$

we call the L_μ-bound. In [28], the authors present multiple approximation algorithms that use the same lower bounds. In order to do this, each lower bound is defined as generally as possible. In this subsection, we will show a specialization of the lower bound in [28], that applies to the optimal 2ECSS and 2VCSS solutions.

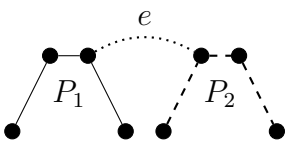
To describe this second bound, we need ear-decompositions with particular properties. Let G be a 2-edge-connected graph. An ear-decomposition D of G is called *nice* if it has the following properties:

1. D is evenmin;
2. all short ears of D are pendant ears;
3. the internal vertices of different short ears of D are non-adjacent in G.

In this paper, we refer to properties (2) and (3) of the above definition collectively as the *short-ears properties*. Figures 2.5a shows an example of a non-pendant short ear P with a nontrivial ear Q having an endpoint in $in(P)$. Figure 2.5b shows two pendant short ears P_1 and P_2 which are adjacent in G (by the edge e). A nice ear-decomposition is an evenmin ear-decomposition that does not have any short-ear configurations of these types.



(a) A non-pendant short ear P



(b) Two pendant short ears P_1 and P_2 whose internal vertices are adjacent

Figure 2.5: Short ear configurations that are not "nice"

Sebő and Vygen (Lemma 2.5 of [28]) show that for any 2-vertex-connected graph G, there exists a nice ear decomposition of G, and that this nice ear-decomposition can be computed in $O(|V(G)| \cdot |E(G)|)$ time. The ear-decomposition constructed by their algorithm may not be open. At a high level, their algorithm starts with an open and evenmin ear decomposition (see Theorem 2.5). It makes a sequence of local changes to this ear-decomposition in order to obtain the short-ears properties, while maintaining the property that the ear-decomposition is evenmin (but possibly creating closed ears). Finally, their algorithm makes a few more local changes to the short ears of this ear-decomposition.

During the course of our investigation, we attempted to use similar ideas to obtain an approximation algorithm for 2-vertex-connectivity. Unfortunately, the algorithm in [28] loses the property of openness while gaining the short-ears properties. Consequently, the solution so obtained (after discarding trivial ears) is not necessarily 2-vertex-connected, and the above theorem does not work for open ear-decompositions. In fact, there are graphs for which there are no ear-decompositions that are open, evenmin, and satisfy the short-ears properties. The example in Figure 2.6 shows a graph G for which every open and evenmin ear-decomposition contains either a non-pendant 3-ear consisting of the dashed edges, or a non-pendant 3-ear consisting of the dotted edges.

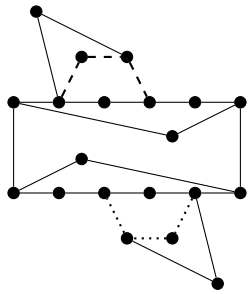


Figure 2.6: A graph with no open evenmin ear-decomposition satisfying the short-ears properties

In Chapter 4, we present a $\frac{17}{12}$-approximation algorithm for 2VCSS on instances with minimium degree at least 3. This algorithm constructs an open and nice ear-decomposition for these instances, showing that such ear-decompositions always exist for 2-vertex-connected graphs of minimum degree at least 3. This was the most general class of graphs for which we could achieve the result. It remains to characterize precisely the graphs for which such ear-decompositions exist.

In [28], the authors construct a nice ear-decomposition of the given 2-vertex-connected graph G. Using this ear-decomposition, they find another lower bound on the cost of an

optimal 2ECSS solution, which we call the L_μ-lower bound. In fact, they prove that this lower bound also applies to $LP(G)$. We will now describe a simplified version of their lower bound, applicable to an optimal solution to the 2ECSS problem.

Let G be a 2-vertex-connected graph, and let D be a nice ear-decomposition of G. The *eardrum* associated with G and D is the set M of connected components of the subgraph induced by the internal vertices of the short ears of D (see Figure 2.7; dashed edges form short pendant ears and dotted edges form a long pendant ear). Let V_M be the vertex set of this subgraph. Every component in M is either an isolated vertex or a pair of vertices with an edge. Let V_L be the set of internal vertices of the pendant ears that are not short, and let V_I be the internal vertices of the non-pendant ears. Observe that V_M, V_L and V_I are disjoint, and $V = V_M \cup V_L \cup V_I$.

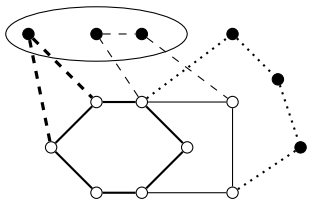


Figure 2.7: A nice ear-decomposition. The eardrum M is circled. The white nodes represent vertices in V_I and the black nodes outside M represent vertices in V_L

In general, there may be trivial ears that have one endpoint in V_M and the other endpoint in V_I. It is possible to replace a pendant short ear by another pendant short ear with the same internal vertices by using these trivial ears. As an example, consider the 3-ear with vertex set $\{u_1, u_2, u_3, u_4\}$, as shown in Figure 2.8. Due to the presence of the two vertices $x, y \in V_I$ and the trivial ears u_2x and u_3y, we may choose to replace the pendant 3-ear $\{u_1 - u_2 - u_3 - u_4\}$ by any of the pendant 3-ears $\{x - u_2 - u_3 - u_4\}$, $\{u_1 - u_2 - u_3 - y\}$ or $\{x - u_2 - u_3 - y\}$, all of which result in valid nice ear-decompositions of G.

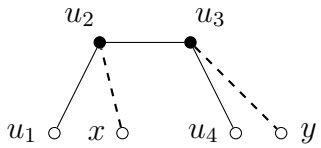


Figure 2.8: A short ear which can be replaced

In fact, making such a replacement in a clever manner allows us to find a new lower bound on $OPT(G)$. We will first restate the above result formally. Let D be the nice

ear-decomposition that we start with. For any $f \in M$, let P_f be the ear with f as its internal vertices. Let Q_f be any path in G with f as its internal vertices. If we replace the ear P_f of D with the ear Q_f, and change the trivial ears accordingly, the resulting ear-decomposition is valid for G and is a nice ear-decomposition of G.

Let $\mathcal{P}_f$ be the set of all possible paths Q_f for each $f \in M$. Suppose that we temporarily remove all edges that have neither endpoint in V_M from G and D. We want to pick exactly one element R_f from each set $\mathcal{P}_f$, such that we need to add as few further edges to the graph $(V(G), \cup_{f \in M} E(R_f))$ as possible in order to make this graph connected. In other words, our goal is to choose paths R_f such that the graph $(V(G), \cup_{f \in M} E(R_f))$ has as few components as possible.

This motivates the definition of an earmuff (Definition 3.1 in [28]). Given G and an associated ear-decomposition D and eardrum M, an *earmuff* for M in G is a set of paths $\{R_f : f \in F\}$, where $F \subseteq M$ such that $(V(G), \cup_{f \in M} E(R_f))$ is a forest. A *maximum earmuff* is one in which $|F|$ is maximum, and this maximum is denoted by $\mu(G, M)$.

Sebő and Vygen [28] showed that earmuffs are the common independent sets of two matroids, whose ground set is $\cup_{f \in M} \mathcal{P}_f$. On the one hand, the sets $\mathcal{P}_f$ partition this ground set. Since an earmuff contains at most one element per partition class, the subsets of $\cup_{f \in M} \mathcal{P}_f$ that contain at most one element per class define a *partition matroid* on the ground set (see e.g. [11]). The subsets of $\cup_{f \in M} \mathcal{P}_f$ whose union is a forest form the *cycle matroid* of a graph if we represent each path $R \in \mathcal{P}_f$ by the set of its two endpoints (see Section 3.1 in [28]). The earmuffs are those sets that are independent in both of these matroids. Thus earmuff maximization reduces to matroid intersection, and it can be solved in polynomial time [7].

We remark that the algorithm by Sebő and Vygen actually solves this subproblem and computes a maximum earmuff. Our algorithm (Chapter 4) does not need to solve this problem, but we use the following lower bound that arises from its structure in our analysis.

Given a graph G and associated nice ear-decomposition D with eardrum M, we let $L_\mu(G, M) := |V| + |M| - \mu(G, M) - 1$. The authors in [28] show that this expression gives a lower bound on the value of $LP(G)$. We direct the reader to Section 4, Theorem 6 in [28] for a proof of this result.

Theorem 2.7. *(Sebő, Vygen) Let G be a 2-vertex-connected graph. Let D be an open nice-ear decomposition of G and let M be the associated eardrum. Then $L_\mu(G, M) \leq LP(G)$.*

If $|M| - \mu(G, M) > 1$ for some ear-decomposition of G, this lower bound is stronger than the naïve lower bound of $|V|$. See Figure 2.9 for an example. If we let M be the

eardrum corresponding to the circled vertices, then $\mu(G, M) = 1$: if we add any pair of short ears, then the edges of these ears form a cycle, thus any earmuff can have at most one such ear. But $|M| = 3$, thus $L_\mu(G, M) = |V| + 1$.

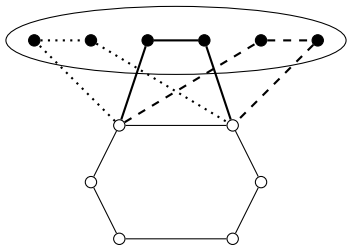


Figure 2.9: A graph G and associated eardrum M with $L_\mu(G, M) = |V| + 1$

2.2.4 The $D2$-Lower Bound

Let G be a 2-edge-connected graph. The $D2$ problem (also called the minimum-size 2-edge-cover problem) is the problem of finding a subgraph of minimum size such that the minimum degree of a vertex of this subgraph is at least 2. A solution to this problem can be found in polynomial time (see e.g. [27]).

Vempala and Vetta [29] were the first authors to use the $D2$ problem as a tool for computing an approximation to the 2ECSS problem.

Let H be a 2-edge-connected spanning subgraph of G. Clearly, every vertex of H has degree at least 2. Thus H is feasible for the D2 problem. We call such a graph (of minimum degree at least 2) a $D2$ of G, and we call an optimal solution a *min-*$D2$ of G.

In Chapter 3, we consider the 2ECSS problems on instances that are 3-vertex-connected. Our algorithm first computes a min-$D2$ of G, and uses the size of this min-$D2$ as a lower bound on $OPT(G)$. Starting with the min-$D2$, we construct a 2-edge-connected spanning subgraph of size at most $\frac{4}{3}$ the size of the min-$D2$.

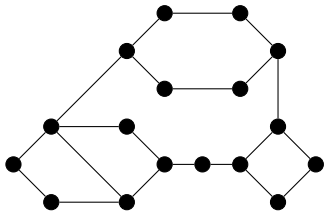
The algorithm in [29] constructs a spanning tree out of the connected components of a min-$D2$, and then iterates over these components. In each iteration, it connects one component to its parent by two edges. Our algorithm works differently, and instead bears some similarity to the algorithm for a weighted version of 2ECSS in [3]: we start with a min-$D2$ and first construct a *bridgeless* $D2$, which is a $D2$ whose connected components are each 2-edge-connected. After this, we merge these 2-edge-connected components into a single large component using the edges of the input graph. As mentioned earlier, our

results show that if G is 3-vertex-connected, then $OPT(G)$ is at most $\frac{4}{3}$ of the size of a min-$D2$ of G.

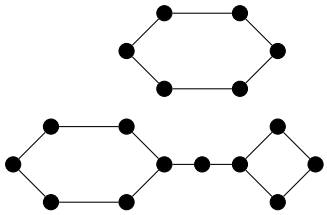
In the rest of this subsection, we will describe the structure of a $D2$, and introduce some terminology and notation that will be used in the rest of this thesis.

Let J be a graph. A *2ec-block* of J is a maximal (with respect to edge and vertex inclusions) 2-edge-connected subgraph of J. We define the *underlying forest* of J as follows. Starting with the graph J, we contract all of the edges in each 2ec-block. The resulting graph contains a node u' for every node $u \in V(J)$ that was not in a 2ec-block, and a node v' for every 2ec-block of J. We call the latter nodes the *compound nodes* corresponding to the 2ec-blocks of J. By the maximality of 2ec-blocks, the graph obtained in this manner is acyclic.

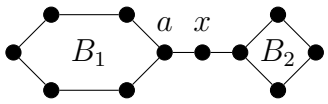
Let G be a 2-edge-connected graph, and H be a $D2$ of G (a subgraph of G with minimum degree at least 2). For any subgraph H' of H, we denote the underlying forest of H' by $\mathcal{T}(H')$. Clearly, for every connected component C of H, $\mathcal{T}(C)$ is a tree, which we call the *underlying tree* of C. For any underlying forest T obtained through these contractions, we denote by $\mathcal{T}^{-1}(T)$ the subgraph of H obtained by reversing the contractions. The figures below show a graph G, a $D2$ H of G, a component C of H, and the underlying tree $\mathcal{T}(C)$ of C.



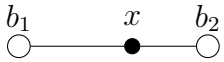
(a) A 2-edge-connected graph G



(b) A $D2$ H of G



(c) A component C of H, with 2ec-blocks B_1 and B_2



(d) The graph $\mathcal{T}(C)$

Figure 2.10: The structure of a $D2$

Let v be a vertex of $T = \mathcal{T}(H)$ (compound or otherwise). A *neighbour* or *neighbouring 2ec-block* of v in T is a compound node u in T in the same component as v, such that none of the internal vertices of the unique u, v-path in T is a compound node.

Observe that since H is a $D2$, every leaf node of $T = \mathcal{T}(H)$ is a compound node. Finally, we remark that every edge of T directly corresponds to a bridge of its corresponding component in $\mathcal{T}^{-1}(T)$; for example, the edge b_1x in Figure 2.10d corresponds to the bridge ax in Figure 2.10c.

Chapter 3

A $\frac{4}{3}$-Approximation for Restricted 2ECSS

In this chapter, we present a $\frac{4}{3}$-approximation algorithm for the unweighted version of the 2-edge-connected spanning subgraph problem (2ECSS), restricted to input graphs that are 3-vertex-connected.

The unweighted 2-edge-connectivity problem is well studied. Khuller and Vishkin [22] gave a simple but clever $\frac{3}{2}$-approximation algorithm for this problem. Using more complex techniques, Cheriyan, Sebő and Szigeti [4] improved this ratio to $\frac{17}{12}$. Recently, Sebő and Vygen [28] improved this ratio even further, to $\frac{4}{3}$. This last ratio was achieved with respect to the 2ECSS-LP (see Chapter 2), by an analysis that used the two lower bounds, the L_φ-lower bound and the L_μ-lower bound, in conjunction.

We present a new algorithm for approximating 2ECSS, restricted to 3-connected graphs, which achieves the approximation guarantee of $\frac{4}{3}$ with respect to another lower bound: the $D2$-lower bound, which is the size of a minimum-size subgraph with minimum vertex degree at least 2 (a min-$D2$ – see Chapter 2). In [29], Vempala and Vetta gave a $\frac{4}{3}$ approximation algorithm for 2ECSS with respect to the $D2$-lower bound, but to the best of our knowledge, their analysis remains incomplete. They use a DFS-based approach, computing a depth first search tree of components of a min-$D2$ and then modifying this tree with local changes to each component to achieve the result.

Our methods are different from those in [29]. We start with a min-$D2$ of the input graph and modify its components to obtain a *bridgeless-$D2$* of G, which is a $D2$ whose connected components are 2-edge connected. We then proceed to merge these components

into a single 2-edge-connected component, which is our solution. The following theorem is a consequence of our algorithm.

Theorem 3.1. *Let G be a 3-vertex-connected graph. The size of a smallest 2-edge-connected spanning subgraph of G is at most $\frac{4}{3}$ the size of a min-$D2$ of G.*

We remark that the restriction to 3-vertex-connected graphs is necessary for our algorithm (without major modifications) to work: the example in Figure 3.1 shows that the cost of an optimal 2ECSS can be as high as $\frac{3}{2}$ of the cost of a min-$D2$ on graphs which are not 3-vertex-connected. This example consists of a repeated "gadget": a 4-cycle, connected to the previous 4-cycle by two edges. The example shows that for each $k \in \mathbb{Z}_{>0}$, there exists a graph G_k on $4k$ vertices, with $6k-2$ edges, such that the size of a min-$D2$ is $4k$ (the solid edges in the figure form a $D2$), but the size of an optimal 2-edge-connected subgraph is $6k-2$ (every solution includes all the edges, since every edge is in a 2-edge-cut).

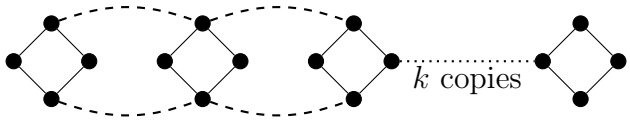
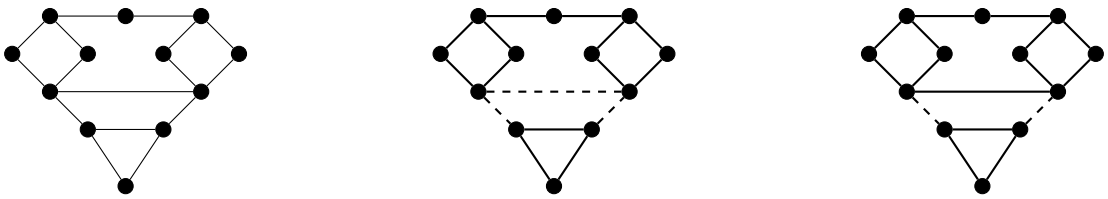


Figure 3.1: The graph G_k, where the ratio between the size of an optimal 2ECSS and the size of a min-$D2$ approaches $\frac{3}{2}$

3.1 Algorithm Overview

Before we begin to explain the details of our algorithm, we require the following definitions.

A *bridgeless-$D2$* of G is a subgraph of minimum-degree 2 (a $D2$), with the added property that every connected component of this subgraph is 2-edge-connected (bridgeless). In other words, a bridgeless-$D2$ of G is a subgraph of G that is a disjoint union of 2-edge-connected graphs. The solid edges in Figure 3.2 show an example of a $D2$ containing a component with a bridge, and a bridgeless-$D2$.



(a) A graph G (b) A $D2$ of G (c) A bridgeless-$D2$ of G

Figure 3.2: A $D2$ and a bridgeless-$D2$

Let $G = (V, E)$ be the input graph. By assumption, G is 3-connected. In the analysis of our algorithm, we use the following token argument. We start with a min-$D2$ $\mathcal{D}$ of G, and assign $\frac{1}{3}$ tokens to each edge of $\mathcal{D}$. We then make a sequence of changes to this $D2$, by adding and deleting some edges, such that the resulting graph is 2-edge-connected. The tokens allow us to count the number of edges in our final solution, and show that this number is at most $\frac{4}{3}$ times the size of $\mathcal{D}$.

Let $H = (V_H, E_H)$ be a subgraph of G. We denote by $t(H) = t(E_H) = \sum_{e \in E_H} t(e)$ the number of tokens on the edges of H, where $t(e)$ is the number of tokens on the edge e.

Let H and H' be two subgraphs of G. When we *buy* an edge $e \in E(G) \setminus E(H)$, we pay one token for it and add it to $E(H)$. When we *sell* an edge $e \in E(H)$, we recover one token for it and remove it from $E(H)$. We *convert* H to H' when we buy all edges in $E(H') \setminus E(H)$ and sell all edges in $E(H) \setminus E(H')$.

Our algorithm consists of a few steps that can be summarized as follows. We let H be the partial solution constructed by our algorithm at any point during its execution.

1. [**Preprocessing**] We construct a min-$D2$ $\mathcal{D}$ of G. This can be done in polynomial time (see e.g. [27]). We set $H := \mathcal{D}$, and assign $\frac{1}{3}$ tokens to each edge of H.

2. [**Bridge-covering**] Starting with the $D2$ H, we buy and sell suitable edges using the tokens assigned to the edges of H, to convert H to a bridgeless-$D2$ with some additional spare tokens on each component.

3. [**Gluing**] Starting with the bridgeless-$D2$ H and the tokens on its components, we buy and sell suitable edges using these tokens, to convert H to a 2ECSS of G.

Since we begin with a min-$D2$ of G with $\frac{1}{3}$ tokens on each edge, and buy and sell edges at the cost of one token per edge, our final solution has at most $\frac{4}{3}$ as many edges as the min-$D2$. This gives us the approximation guarantee.

The goal of the bridge-covering step is to convert H to a bridgeless-$D2$ of G that satisfies the following invariant.

Gluing Invariant

- Let C be a component of H. Then $t(C) \geq \min(2, \frac{|E(C)|}{3})$.

In each iteration of the gluing step, we will merge components of this bridgeless-$D2$ together, while maintaining the Gluing Invariant and the property that H is a bridgeless-$D2$ of G.

The concept of *bridge-covering* was first introduced in [3], and the methods used in the bridge-covering step of the algorithm are inspired by the methods in [3]. At a high level, the bridge-covering process starts with the min-$D2$ H with $\frac{1}{3}$ tokens on each edge, and iterates over the components of H that have a bridge. In each iteration, we highlight one component as the *growing* component C_0 for that iteration. We process the growing component for as long as it contains a bridge, so that at the end of this iteration, the growing component is bridgeless.

We do this one bridge at a time. As long as C_0 contains a bridge e, we find a path in G which covers that bridge, and buy its edges. More formally, we find a path P in G such that if we buy the edges of P that are not yet in our solution, then e is no longer a bridge in the resulting graph (we say that the path P *covers* the bridge e in our solution). During this process, we sometimes sell some redundant edges (which are not required for 2-edge-connectivity) in order to obtain a few extra tokens. We refer the reader to Figure 3.3a for a pictorial example, where thick edges are in the growing component, solid edges are in the current solution, and dashed edges are edges of G that are not in the current solution. The path P is shown in Figure 3.3b by the thick dotted edges. As the name *growing component* suggests, this process could result in the addition of some new vertices to C_0,

since the components that contain vertices of P are merged with the growing component. The details pertaining to our choice of P, and the supporting token argument bounding the number of tokens we use for this process, are deferred to Section 3.3 of this chapter.

(a) The graphs G and H (b) The path P

Figure 3.3: Bridge-covering

We will show that on termination of the bridge covering step, the solution maintains the Gluing Invariant. The gluing step of our algorithm starts with this graph, and constructs a 2-edge-connected spanning subgraph of G using the tokens on its components to buy some edges of G. In each iteration of the gluing step, we choose one component of our partial solution, and find an ear in G with endpoints in this component, whose vertices are in multiple components. We use the tokens on these components to buy the edges that are not yet in our solution.

In both of the above steps, we find a path through multiple components. Our algorithm often needs to find such a path, starting and ending at particular components or vertices, with the added property that the path must "enter" and "exit" each component at most once (in other words, for each component, at most two edges of the path have exactly one endpoint in that component). In all of these cases, as long as we find a path we can assume that we have attained this property, because we can "shortcut" this path in order to achieve the required property.

More formally, let G be the input graph and H (a subgraph of G) be the partial solution constructed so far. Let P be a path in H, and let $\mathcal{C}_P$ be the set of components of H that contain at least one vertex of P. Then, when we *shortcut* P, we replace it with a path P' that "enters" and "exits" each component at most once, in the following manner. Let $C \in \mathcal{C}_P$ be a component such that more than 2 edges of P have exactly one endpoint in C. Let $e_1 = ab$ and $e_2 = cd$ respectively be the first and last edges of P with this property, such that b and c are in C. We delete the b, c-subpath of P, and replace it with a b, c-path in the connected component of H. After repeating the above procedure for any component that

P enters multiple times, we attain the above property. We call this process *shortcutting*, and say that we have *shortcut* P.

Before we begin, we require the following lemma, which we will use repeatedly in the analysis of our algorithm.

Lemma 3.2. *Let $G = (V, E)$ be a 2-edge-connected graph and $e = uv$ be an edge of G such that there exists a collection of 3 edge-disjoint u, v-paths in G. Then $G \setminus e$ is 2-edge-connected.*

Proof. Since the u, v-paths are edge disjoint, at most one of them contains the edge e. Thus the other two paths (call these P_1 and P_2) are u, v-paths in $G \setminus e$. Since G is 2-edge-connected, $G \setminus e$ is connected. Suppose, for the sake of finding a contradiction, that $G \setminus e$ is not 2-edge-connected. Then it has a bridge e'. Let S be the set of vertices of one connected component of $(G \setminus e) \setminus e'$. Then $\delta(S)$ is the cut corresponding to the bridge e' in $G \setminus e$.

If u and v are both in S (or both in $V \setminus S$) then $e \notin \delta(S)$, and every edge in G that is in $\delta(S)$ is also in $G \setminus e$. Hence G is not 2-edge-connected, contradicting out assumption. In the only other case, exactly one of u and v is in S (suppose without loss of generality that u is in S). Since P_1 and P_2 are edge-disjoint u, v-paths in $G \setminus e$, there exist distinct edges $e_1 \in E(P_1)$ and $e_2 \in E(P_2)$ such that $e_1, e_2 \in \delta(S)$ in $G \setminus e$, contradicting our assumption that e' is a bridge. □

We will first describe the gluing step, then the bridge-covering step. The input to the gluing step is a bridgeless-$D2$ with some tokens on its components. This graph maintains the Gluing Invariant, which says that components with at least 6 edges have at least 2 tokens, and components with 3, 4 or 5 edges have exactly $\frac{1}{3}$ tokens per edge. Recall that each component of a bridgeless-$D2$ is 2-edge-connected.

Section 3.2 discusses the Gluing process in detail, and Section 3.3 discusses the bridge-covering process in detail.

3.2 Gluing

We begin by describing the gluing procedure at a high level. Let H be a bridgeless-$D2$ that maintains the Gluing Invariant.

We repeat the following procedure until H is connected (and therefore 2-edge-connected). Denote by G' the multigraph obtained from G by contracting all of the edges in H. Since G is 2-edge-connected, G' is 2-edge-connected. We find a closed ear in G', and use tokens from the components corresponding to the vertices of this ear to buy the edges in G that correspond to edges of this ear of G'. As a result, these components are merged into a single large 2-edge-connected component. During this process, we sometimes sell redundant edges (edges that are not required to maintain 2-edge-connectivity of a component) to obtain extra tokens. At any point during this step, the partial solution H is a bridgeless-$D2$ of G.

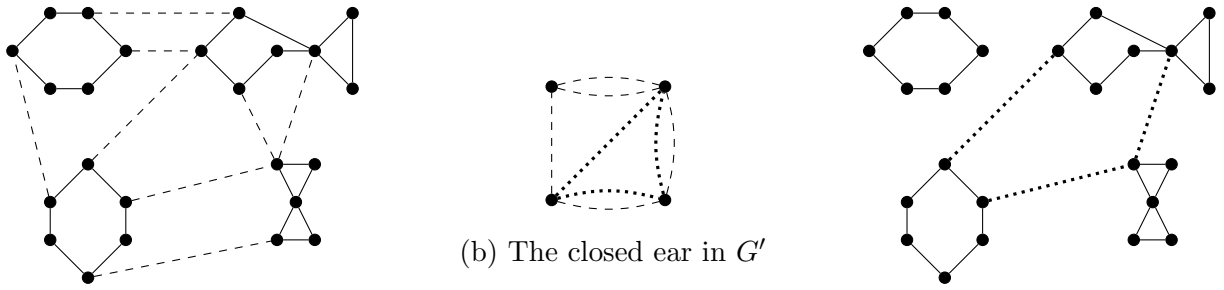
If we assume that every component has at least 2 tokens, then this process is straightforward. The components that have less than 2 tokens require a more complicated algorithm to handle them. In this step, our algorithm first handles these hard cases, and then proceeds to the simple phase where every component has at least 2 tokens. In the first phase, we process all of the "small" components of H (components with at most 5 edges), by starting with each such component and choosing a closed ear containing this component. When we buy the edges of the closed ear, we ensure that the resulting component is 2-edge-connected and has at least 6 edges and 2 tokens. At the end of this phase, every component of H has at least 2 tokens. In the second phase, we merge all of the remaining components of H into a single 2-edge-connected spanning subgraph of G.

We will now describe the details of both phases of the gluing process. We will start by describing the second phase, which is much simpler. For this phase, we may assume that every component has at least 2 tokens.

Second Phase

At the start of this phase, $H = (V, E_H)$ is a bridgeless-$D2$ with at least 2 tokens on each component. We contract each component C_i into a vertex v_i. If the resulting multigraph has at least 2 vertices, then it is 2-edge-connected. We find a cycle $v_1, \ldots, v_k$ with $k \geq 2$, and buy the edges of G corresponding to edges of this cycle (see Figure 3.4). We add these edges to the set E_H. This costs k tokens, and we have $2k \geq k+2$ tokens available on the components of this cycle, each of which becomes a part of the new 2-edge-connected component. Hence we are left with at least 2 spare tokens, which we assign to the new component in H, and the Gluing Invariant holds.

We iterate this process until we are left with a single 2-edge-connected component, which is our final solution. Figure 3.4 shows an example of one iteration of this phase of the gluing step. The solid edges of Figure 3.4a represent edges of H, and the dashed edges represent edges of G that are not in H. The dotted edges of Figures 3.4b and 3.4c represent edges of the closed ear.



(a) The graphs G and H

(b) The closed ear in G'

(c) The closed ear in G

Figure 3.4: The second phase of the gluing step

We next describe the first phase. In this phase, we individually handle all of the small cycles (with 3, 4 or 5 edges) separately. We first process the 3-cycles, then the 4-cycles, and finally the 5-cycles of H. We iterate this process until every 2-edge-connected component of H has at least 6 edges and at least 2 tokens (after which we may proceed to the second phase described earlier).

Before we begin to describe this phase of our algorithm, we require the following definitions and lemmas (which we will prove at the end of this section).

Let P be a path in G. We denote by $\mathcal{C}(P)$ the set of components of H that contain an internal vertex of P.

Lemma 3.3. *Let $G = (V, E)$ be a 3-connected graph and $R, S \subseteq V$ with $|R|, |S| \geq 3$ and $R \cap S = \varnothing$. Then there exists a set of 3 vertex-disjoint R, S-paths in G that can be computed in polynomial time.*

Lemma 3.4. *Let H be a bridgeless-D2 of G and B_1 and B_2 be two components of H. Let P be a simple B_1, B_2-path in G that shares only its endpoints with $V(B_1) \cup V(B_2)$. Suppose that for each $C \in \mathcal{C}(P)$, P has exactly two edges with exactly one endpoint in C.*

Then $|E(P) \setminus E(H)| \leq 1 + \sum_{C \in \mathcal{C}(P)} t(C)$.

The above lemma states that if we are given one extra token, we can buy the edges of $E(P) \setminus E(H)$ using the tokens on the components in $\mathcal{C}(P)$ and this extra token.

This lemma is proved at the end of the section. The proof is simple, and implicitly provides an algorithm to obtain the tokens required and to buy the edges of the path, while maintaining the Gluing Invariant.

Processing 3-cycles

We repeat this step as long as H contains a component that is a 3-cycle. We choose a 3-cycle R. Let $V(R) = \{a, b, c\}$. We choose an edge e with exactly one endpoint in $V(R)$. Without loss of generality, assume $e = ax$. Let B be the component of H that contains x.

Since G is 3-connected, by Lemma 3.3, we find a set $\mathcal{P} = \{P_1, P_2, P_3\}$ of vertex-disjoint paths in G between the sets $V(R)$ and $V(B)$. We have the following cases for B:

Case 1. B is a 3-cycle. Let $V(B) = \{x, y, z\}$.

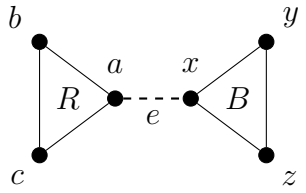


Figure 3.5: Processing 3-cycles: Case 1

At least one of the three paths in $\mathcal{P}$ has one endpoint in $\{b, c\}$ and the other endpoint in $\{y, z\}$. Denote this path by P, and suppose without loss of generality that P ends at b and y. We shortcut P as explained earlier (note that shortcutting might change the set $\mathcal{C}(P)$).

By Lemma 3.4, we require one token to buy P. We require an additional token to buy the edge e, and another 2 tokens to assign to the resulting component to maintain the Gluing Invariant, for a total of 4 tokens.

We obtain one token from each of R and B. Since a and b have at least 3 edge-disjoint paths between them, we can sell the edge ab by Lemma 3.2 to recover one token. Since x and y have at least 3 edge-disjoint paths, we sell xy to recover another token. Hence we have the required 4 tokens for this case.

Note that the vertices a and x are in the same 2ec-block of H after this iteration.

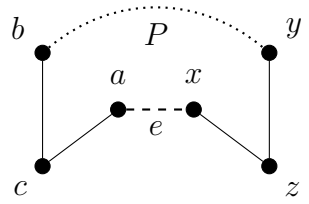


Figure 3.6: Processing 3-cycles: Case 1

Case 2. B is a 4-cycle. Let $V(B) = \{x, y, z, w\}$, such that x is adjacent to y and z in H.

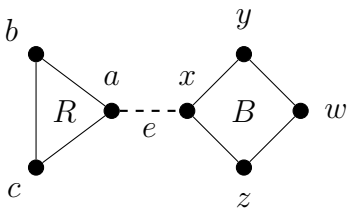


Figure 3.7: Processing 3-cycles: Case 2

If there exists a path in $\mathcal{P}$ with one endpoint in $\{b, c\}$ (without loss of generality, say b) and the other endpoint in $\{y, z\}$ (without loss of generality, say y), we shortcut this path as before. To buy this path, by Lemma 3.4, we require one token. We require an additional token to buy the edge ax, and two tokens to assign to the resulting component, for a total of 4 tokens. We obtain one token from each of B and R, and sell the edges ab and xy to obtain two more tokens. As explained earlier, the resulting component is 2-edge-connected and maintains the gluing invariant.

Otherwise, the two paths in $\mathcal{P}$ with an endpoint in $\{b, c\}$ have their other endpoints in $\{x, w\}$. Suppose without loss of generality that P_1 is a b, x-path, P_2 is a c, w-path, and P_3 is an a, y-path in G.

If $\mathcal{C}(P_1) \cap \mathcal{C}(P_3) \neq \varnothing$, let C be a component of H that is in both of these sets. We construct a b, y-path P by taking the subpath of P_1 from b to C and the subpath of P_3 from C to y, and an appropriate subpath in C connecting these two paths. We shortcut P as before. By Lemma 3.4, we require one token to buy the edges of P not already in H. We require an additional token to buy e and two more tokens to maintain the Gluing Invariant. We buy the edges of P and the edge e. We obtain one token from each of R and B, and sell ab and xy

for two additional tokens.

Otherwise, $\mathcal{C}(P_1) \cap \mathcal{C}(P_3) = \varnothing$. We shortcut the paths P_1 and P_2 and buy their edges using the tokens from R and B and Lemma 3.4. As before, we sell the edges ab and xy by Lemma 3.2, and recover two tokens to maintain the Gluing Invariant.

Case 3. B is a 5-cycle. Let $V(B) = \{x, y, w, w', z\}$, labelled in order around B.

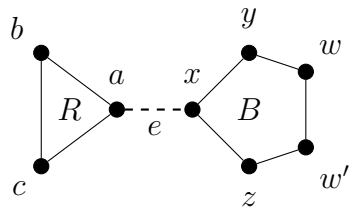


Figure 3.8: Processing 3-cycles: Case 3

Since B has 5 vertices, some pair of paths in $\mathcal{P}$ have endpoints that are adjacent in B. Let P_1 and P_2 be these paths. Clearly, the endpoints of P_1 and P_2 are adjacent in R, since R is a 3-cycle. We shortcut P_1 and P_2 as described earlier.

If $\mathcal{C}(P_1) \cap \mathcal{C}(P_2) = \varnothing$, then we use one token from each of R and B to buy these paths (by Lemma 3.4). We sell the edge between the endpoints of P_1 and P_2 in R, and the edge between the endpoints of P_1 and P_2 in B, since they have at least 3 edge-disjoint paths between them (by Lemma 3.2). We recover two tokens to assign to the resulting component. This component is 2-edge-connected and maintains the Gluing Invariant.

Otherwise, $\mathcal{C}(P_1) \cap \mathcal{C}(P_2) \neq \varnothing$. Let $C \in \mathcal{C}(P_1) \cap \mathcal{C}(P_2)$ be a common component with the maximum number of tokens in this set.

If $t(C) \geq \frac{4}{3}$, we take $\frac{1}{3}$ tokens from C to use later on in our analysis. C is now left with 1 token. We buy the path P_1 and the edge e, for which (by Lemma 3.4) we require 2 tokens. To maintain the Gluing Invariant, we require 2 more tokens, for a total requirement of 4 tokens. We refer the reader to Figure 3.9 for this sub-case. We obtain $\frac{5}{3}$ tokens from B, $\frac{1}{3}$ from C and 1 from R, for a total of 3 tokens. We then sell the edge ab, maintaining 2-edge-connectivity by Lemma 3.2, to obtain an extra token. The resulting component is 2-edge-connected and has at least 2 tokens, and hence maintains the Gluing Invariant.

If $t(C) = 1$, then every component of $\mathcal{C}(P_1) \cap \mathcal{C}(P_2)$ is a 3-cycle. Since P_1 and P_2 are vertex-disjoint, they are edge-disjoint. To buy all of the edges in $(E(P_1) \cup E(P_2)) \setminus E(H)$, by Lemma 3.4, we require one extra token for each

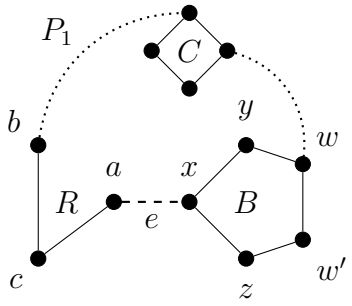


Figure 3.9: Processing 3-cycles: Case 3

component $C \in \mathcal{C}(P_1) \cap \mathcal{C}(P_2)$, since each of these components has only one token to begin with (and we require 2 tokens to apply Lemma 3.4 twice). Suppose that we have exactly one extra token per component in this set (we will describe how to obtain these later). We use these tokens and one token from each of R and B, and we buy all of the edges in P_1 and P_2 (by Lemma 3.4).

We will now describe how we obtain one extra token per component in the set. Let C be a component of $\mathcal{C}(P_1) \cap \mathcal{C}(P_2)$. We denote by $f_{P_1}(C)$ the edge of P_1 "entering" C, which is the first edge of P_1 with exactly one endpoint in $V(C)$. We denote by $e_{P_1}(C)$ the edge of P_1 "leaving" C, which is the second (and last) edge of P_1 with exactly one endpoint in $V(C)$. We define $f_{P_2}(C)$ and $e_{P_2}(C)$ in a similar fashion for P_2. Upto relabelling P_1, P_2 and the vertices of C, there are exactly two possible configurations of these edges that can occur (see Figure 3.10), since C has exactly 3 vertices and the two paths are vertex disjoint. Either each path contains exactly one vertex of C, or exactly one path contains two vertices of C.

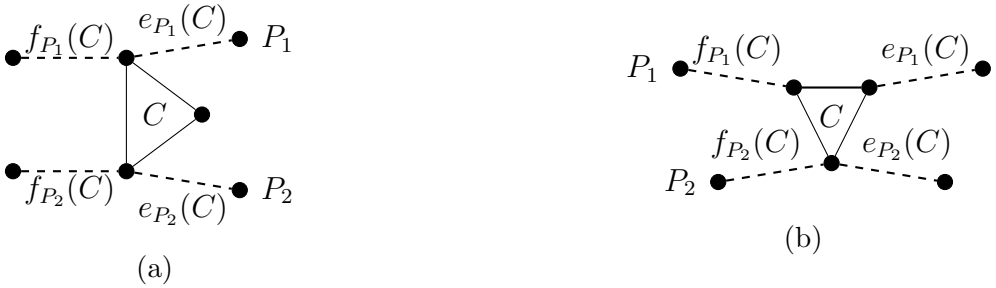


Figure 3.10: Processing 3-cycles: Case 3

Let $V(C) = \{u_1, u_2, u_3\}$, such that u_1 is an endpoint of $f_{P_1}(C)$ and u_2 is an

endpoint of $f_{P_2}(C)$. Observe that the edge u_1u_2 does not belong to the edge sets of either P_1 or P_2. In the new 2-edge-connected component created by buying these paths, the vertices u_1 and u_2 have at least 3 edge-disjoint paths between them (the edge u_1u_2, a path through R that uses subpaths of P_1 and P_2, and a path through B that uses subpaths of P_1 and P_2), so we sell the edge u_1u_2 to recover one token by Lemma 3.2. After repeating this process for all components of $\mathcal{C}(P_1) \cap \mathcal{C}(P_2)$, we obtain the required number of tokens to buy both paths.

Finally, we sell the edges between the endpoints of P_1 and P_2 in R, and the endpoints of P_1 and P_2 in B, which maintains 2-edge-connectivity by Lemma 3.2, and we recover 2 tokens to assign to the resulting component. This component is 2-edge-connected and maintains the Gluing Invariant.

Case 4. $t(B) \geq 2$. Let $P_1 \in \mathcal{P}$ be the path with an endpoint at b. We simplify the path P_1 as explained before and buy its edges, and we buy the edge e. We require a total of 4 tokens to maintain the Gluing Invariant. We obtain one token from R, two from B, and one by selling the edge ab, which is possible by Lemma 3.2.

Processing 4-cycles

At this stage, H does not have any component that is a 3-cycle. We repeat this step as long as H contains a component that is a 4-cycle. We choose a 4-cycle R. Let $V(R) = \{a, b, c, d\}$, such that a is adjacent to b and c in H. We choose an edge e with exactly one endpoint in $V(R)$. Without loss of generality, we may assume $e = ax$. Let B be the component of H that contains x.

Since G is 3-connected, by Lemma 3.3, we find a set $\mathcal{P} = \{P_1, P_2, P_3\}$ of vertex-disjoint paths in G between the sets $V(R)$ and $V(B)$. We have the following cases for B:

Case 1. B is a 4-cycle. Let $V(B) = \{x, y, z, w\}$, such that x is adjacent to y and z.

Since each of the sets $V(R)$ and $V(B)$ contains 4 vertices, there are 2 pairs of paths in $\mathcal{P}$ whose endpoints in $V(R)$ are adjacent in H, and 2 pairs whose endpoints in $V(B)$ are adjacent in H. Thus there is at least one pair of paths whose endpoints in both sets are adjacent in H. Let P_1 and P_2 be these paths. If $\mathcal{C}(P_1) \cap \mathcal{C}(P_2) = \varnothing$, then we shortcut each of these paths as described earlier, and use one token from each of R and B to buy their edges, by Lemma 3.4. We sell the edge of R connecting their endpoints in $V(R)$ and the edge of

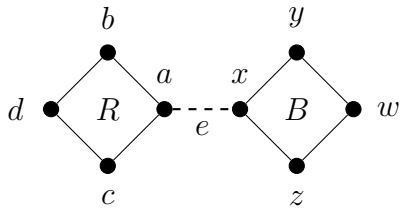


Figure 3.11: Processing 4-cycles: Case 1

B connecting their endpoints in $V(B)$, which maintains 2-edge-connectivity by Lemma 3.2, to recover 2 tokens (see Figure 3.12). Then the resulting component is 2-edge-connected and has at least 2 tokens, hence the Gluing Invariant holds.

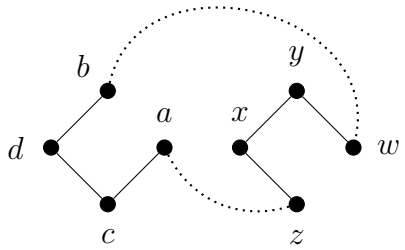


Figure 3.12: Processing 4-cycles: Case 1

Otherwise $\mathcal{C}(P_1) \cap \mathcal{C}(P_2) \neq \varnothing$. Since the endpoints in $V(R)$ of P_1 and P_2 are adjacent in H, at least one of them (without loss of generality, say P_1) has an endpoint in $\{b, c\}$. Let b be the endpoint of P_1. We shortcut P_1 as we did earlier. Since we had $\mathcal{C}(P_1) \cap \mathcal{C}(P_2) \neq \varnothing$ before shortcutting P_1, we have $\mathcal{C}(P_1) \neq \varnothing$ after shortcutting P_1. Let $C \in \mathcal{C}(P_1)$ be a component of this path. Since $|E(C)| \geq 4$, $t(C) \geq \frac{4}{3}$. We take $\frac{1}{3}$ tokens from C to use later on, leaving one token remaining on C. We buy the edges of P and the edge e, for which we require 2 tokens (by Lemma 3.4). We require an additional 2 tokens to maintain the gluing invariant. We obtain $\frac{1}{3}$ tokens from C, and $\frac{4}{3}$ tokens from each of R and B. We then sell the edge ab to recover one token for a total of 4 tokens.

Case 2. B is a 5-cycle. Let $V(B) = \{x, y, w, w', z\}$, labelled in order around B. At least one path in $\mathcal{P}$ has an endpoint in $\{b, c\}$. Let P_1 be this path, and suppose without loss of generality that P_1 ends at b. We shortcut and buy P_1 and e. By Lemma 3.4, we require 4 tokens in total to maintain the Gluing

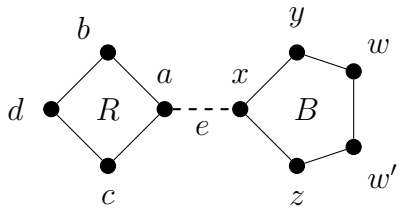


Figure 3.13: Processing 4-cycles: Case 2

Invariant. We obtain $\frac{4}{3}$ tokens from R and $\frac{5}{3}$ from B, and we sell ab to recover one token for a total of 4 tokens.

Case 3. $t(B) \geq 2$. There exists at least one path in $\mathcal{P}$ with an endpoint at either b or c. Let $P_1 \in \mathcal{P}$ be such a path, and without loss of generality, let b be an endpoint of P_1. We simplify the path P_1 as explained before, and buy this path and the edge e. We sell the edge ab to recover one token, and obtain 2 tokens from B and 1 token from R to get the required 4 tokens to maintain the Gluing Invariant.

Processing 5-cycles

At this stage, H does not have any component that is a 3-cycle or 4-cycle. We repeat this step as long as H contains a component that is a 5-cycle.

We choose a 5-cycle R, and an edge $e = ax$ with exactly one endpoint a in $V(R)$. Let B be the component of H that contains x.

Since G is 3-connected, by Lemma 3.3, we find a set $\mathcal{P} = \{P_1, P_2, P_3\}$ of vertex-disjoint paths in G between the sets $V(R)$ and $V(B)$. Since R is a 5-cycle, there exists a pair of paths in $\mathcal{P}$ whose endpoints in $V(R)$ are adjacent in H. Let $\{P_1, P_2\}$ be this pair.

If $\mathcal{C}(P_1) \cap \mathcal{C}(P_2) = \varnothing$, then we shortcut each of these paths as described earlier, and use two tokens to buy their edges, by Lemma 3.4. We sell the edge of R connecting their endpoints in $V(R)$. We require an additional 2 tokens to maintain the Gluing Invariant, for a total requirement of 4 tokens. We obtain 1 token from selling an edge of R, and at least $\frac{5}{3}$ tokens from each of R and B, for a total of at least 4 tokens.

Otherwise, $\mathcal{C}(P_1) \cap \mathcal{C}(P_2) \neq \varnothing$. We shortcut P_1, as explained earlier. Let $C \in \mathcal{C}(P_1)$ be a component containing a vertex of P_1. Since $t(C) \geq \frac{5}{3}$, we take $\frac{2}{3}$ tokens from C, leaving one token on this component. We buy the edges of P and the edge e, for which we require 2 tokens. We obtain at least $\frac{5}{3}$ tokens from each of R and B and

$\frac{2}{3}$ tokens from C, and we assign the two extra tokens to the resulting component to maintain the Gluing Invariant.

On termination of the above process, the graph H is 2-edge-connected.

Observe that every step of the above algorithm can be carried out in polynomial time. In each step, we merge at least two components, thus the number of steps is polynomial in the size of the input, and the overall running time is polynomial.

We now prove Lemmas 3.3 and 3.4. To prove Lemma 3.3, we require *Menger's Theorem* (see e.g. [6]).

Theorem 3.5. *(Menger's Theorem) Let $G = (V, E)$ be a graph and $R, S \subseteq V$. Then the minimum number of vertices separating A from B in G is equal to the maximum number of vertex-disjoint R, S-paths in G.*

Lemma 3.3. *Let $G = (V, E)$ be a 3-connected graph and $R, S \subseteq V$ with $|R|, |S| \geq 3$ and $R \cap S = \varnothing$. Then there exists a set of 3 vertex-disjoint R, S-paths in G that can be computed in polynomial time.*

Proof. Construct the auxiliary digraph $G' = (V', A')$ of G as follows. For each $v \in V$, add two vertices v_{in} and v_{out} to V', and add an arc from v_{in} to v_{out} to A'. For every edge $uv \in E$, add the two arcs $u_{out}v_{in}$ and $v_{out}u_{in}$ to A'. Add two vertices r and s to V'. For each $v \in R$, add an arc from s to v_{in}. Finally, for each $v \in S$, add an arc from v_{out} to s.

Assign a capacity of 1 to each arc, and find an r, s-maximum flow F in the resulting graph. This flow consists of a set of internally vertex-disjoint paths from r to s. This follows from the fact that all of the inflow at a vertex u_{in} leaves that vertex on the $u_{in}u_{out}$ arc, and that all of the inflow at a vertex u_{out} enters that vertex on the $u_{in}u_{out}$ arc.

By Menger's Theorem, G contains at least 3 disjoint R, S paths, which correspond to 3 internally disjoint r, s paths in G'. Thus the maximum flow value is at least 3. By traversing flow paths in F, we can construct a set of at least 3 vertex disjoint r, s paths in G', corresponding to a similar set in G. □

Lemma 3.4. *Let H be a bridgeless-D2 of G and B_1 and B_2 be two components of H. Let P be a simple B_1, B_2-path in G that shares only its endpoints with $V(B_1) \cup V(B_2)$. Suppose that for each $C \in \mathcal{C}(P)$, P has exactly two edges with exactly one endpoint in C.*

Then $|E(P) \setminus E(H)| \leq 1 + \sum_{C \in \mathcal{C}(P)} t(C)$.

Proof. Every edge of $E(P) \setminus E(H)$ connects two different components of $\mathcal{C}(P) \cup \{B_1, B_2\}$, and there are $|\mathcal{C}(P)| + 1$ such edges. If we map each component of $\mathcal{C}(P)$ to the edge of this set entering that component, and use the token on that component to pay for that edge, then we are left with having to pay for only the last edge of P in this set, for which we require one extra token. $Thus |E(P) \setminus E(H)| \leq 1 + \sum_{C \in \mathcal{C}(P)} t(C)$. □

3.3 Bridge-covering

The notion of *bridge-covering* was introduced in [3], in which the authors consider a weighted version of the 2-edge-connectivity problem, and provide an approximation algorithm that starts with a min-$D2$ and constructs a bridgeless-$D2$ in a similar manner. The techniques used in this section are inspired by those in [3]. The bridge-covering step starts with a min-$D2$ H of the given 3-connected graph G, with $\frac{1}{3}$ tokens on each edge, and converts it to a bridgeless-$D2$ that maintains the Gluing Invariant.

A high level explanation of this step is presented in Section 3.1 of this chapter. Briefly, we repeat the following procedure until H is a bridgeless-$D2$ of G that satisfies the Gluing Invariant. We choose a component of H that contains a bridge (call this component C_0 or the growing component), and process this component until it is 2-edge-connected, through a sequence of steps. In each step, we choose a bridge e of C_0 and find a path that covers e, such that if we buy the edges of this path and add them to H, then e is no longer a bridge in the resulting graph. At any point of the bridge-covering step, the partial solution H is a $D2$ of G.

For this section, we call a 2ec-block *small* if it has at most 5 edges, and *large* otherwise. Observe that since $\mathcal{D}$ is a min-$D2$, the small 2ec-blocks with k edges are k-cycles.

During the bridge-covering step, we use two invariants to track the number of tokens on each 2ec-block. The first of these is the General Invariant.

General Invariant

- Let B be a 2ec-block of C. Then $t(B) \geq \min(2, \frac{|E(B)|}{3})$.

For any component C of H other than the growing component C_0, it is easy to see that this invariant holds for C at the start of the algorithm. In fact, the invariant holds for C until the first iteration in which C either gets chosen as the growing component, or merges into the growing component, since our algorithm does not affect the tokens on components that do not merge with the growing component in any iteration.

Once we choose a growing component C_0, the bridge-covering process for this component occurs in 2 phases, which we call Phase 0 and Phase 1. The goal of Phase 0 is to buy and sell suitable edges so that C_0 and some 2ec-block R of C_0 satisfy the second invariant, which we call the Bridgeless Invariant.

Bridgeless Invariant

1. The root 2ec-block R has at least 2 tokens (that is, $t(R) \geq 2$).

2. Let N be a neighbour of R. Then $t(N) \geq \frac{1}{2}$.

3. Let N be a 2ec-block that is not a neighbour of R. Then $t(N) \geq \min(2, \frac{|E(N)|}{3})$.

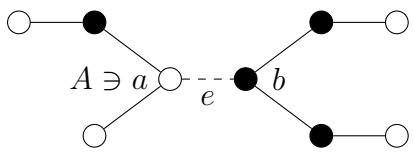
Suppose that we have chosen a growing component C_0. Let $\mathcal{B}$ be the set of 2ec-blocks of C_0. If $\mathcal{B}$ contains a large 2ec-block B, then the pair (C_0, B) maintains the Bridgeless Invariant, and we skip Phase 0 for C_0 and proceed directly to Phase 1. Otherwise, we use one iteration of Phase 0 to buy and sell some edges so that C_0 and some 2ec-block R maintain the Bridgeless Invariant, and then proceed to Phase 1.

When we enter Phase 1 of the bridge-covering step, we remain in this phase until the growing component C_0 is 2-edge-connected.

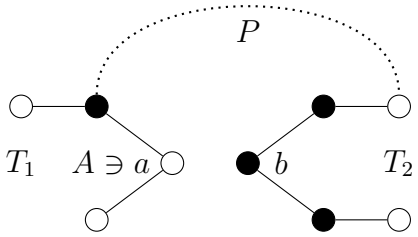
In addition to the above invariants, we have the following property on the bridges of the components of H. Throughout the operation of the bridge-covering step, every bridge has at least $\frac{1}{3}$ tokens, because we do not use the tokens on a bridge of H unless we cover that bridge in that iteration.

Before we begin to describe the two phases, we will explain how we choose paths that cover each bridge, and how we pay for these paths.

Let C_0 be the growing component, and let $e = ab$ be a bridge of C_0. Let $T = \mathcal{T}(C_0)$ be the underlying tree of C_0 (we refer the reader to Chapter 2 for a description of the structure of a $D2$ and its underlying forest). Let T_1 and T_2 be the two trees obtained from T by removing the edge e, such that a is in T_1 (see Figures 3.14a and 3.14b).



(a) An underlying tree example (white nodes are compound nodes)



(b) A fair path P in G covering e in H

Figure 3.14: Fair paths

We call a path P in G a *fair* path covering e in H, if the following hold:

(i) P has one endpoint in $\mathcal{T}^{-1}(T_1)$ and the other endpoint in $\mathcal{T}^{-1}(T_2)$.

(ii) The internal vertices of P are disjoint from $V(C_0)$.

(iii) For any component $C \neq C_0$ of H, if P contains a vertex of C, then P has exactly two edges with exactly one endpoint in C (in other words, P "enters" and "exits" the component C at most once).

We will first show the existence of a fair path in G covering e. Since G is 2-edge connected, $G \setminus e$ is connected and has an a, b-path $P_{a,b}$.

Suppose $P_{a,b}$ intersects a component C' of H multiple times. We may shortcut $P_{a,b}$ as described earlier to attain property (iii).

Let y be the last vertex of $P_{a,b}$ that intersects T_1, and y' be the first vertex that intersects T_2. If we discard the subpaths from a to y and from y' to b in $P_{a,b}$, the resulting path has properties (i) and (ii). Further, since T_1 and T_2 are disjoint, this path has at least one edge, and is hence a fair path covering e.

Let $\mathcal{P}$ be a set of fair paths covering e in H. A path $P \in \mathcal{P}$ with endpoints $x \in \mathcal{T}^{-1}(T_1)$ and $y \in \mathcal{T}^{-1}(T_2)$ is called *T_2-maximum* for the set $\mathcal{P}$ if the number of bridges of C_0 in a path from y to b in C_0 is maximum among all fair paths in $\mathcal{P}$ (in other words, if the path covers the maximum number of bridges of T_2).

The following fact and lemmas will be used in the analysis of our algorithm to show that we can find fair paths covering a bridge, and to explain how we pay for these paths. We defer the proofs of the lemmas to the end of this section.

Fact 3.6. *Let G be a 2-edge-connected graph and H be a min-D2 of G. Let B be a 2ec-block of H with vertex $a \in V(B)$, and suppose there is a bridge $e = ab$ incident on B. Then b has degree 2 in H.*

Since a is adjacent to at least two vertices of B and the vertex b, it has degree at least 3. If b has degree at least 3, then $H \setminus ab$ is a $D2$ of G, contradicting the minimality of H.

Lemma 3.7. *Let G be a 2-edge-connected graph and H be a $D2$ of G. Let $e = ab$ be a bridge of some component C_0 of H. Let $T = \mathcal{T}(C_0)$ and let T_1 and T_2 be the components of $T \setminus e$ (such that a is in T_1 or in a compound node of T_1). Let $V_1 \subseteq \mathcal{T}^{-1}(T_1)$ and $V_2 \subseteq \mathcal{T}^{-1}(T_2)$ be disjoint subsets of $V(C_0)$. A fair path that is T_2-maximum for the set of all fair paths covering e with endpoints in V_1 and V_2 can be computed in polynomial time.*

During the bridge-covering procedure, all of the bridges that we choose to cover have one endpoint in a 2ec-block of H. Let R be a 2ec-block containing the endpoint a of e, and suppose C_0 and R satisfy (2) and (3) of the Bridgeless Invariant. Then we have the following lemma.

Lemma 3.8. *Let P be a T_2-maximum fair path (for some set) covering e in H. Let $\mathcal{C}$ be the set of components containing a vertex of P. Then*

1. $|E(P) \setminus E(H)| \leq 1 + \sum_{C \in \mathcal{C}} t(C)$.

2. We can use the tokens on components in $\mathcal{C}$ to buy the edges of P such that the resulting growing component and 2ec-block containing e satisfy (2) and (3) of the Bridgeless Invariant.

The above lemma states that if we are given one extra token, then we can buy the edges in $E(P) \setminus E(H)$ using this extra token and some tokens on the 2ec-blocks of the components containing internal vertices of P, such that the resulting growing component and resulting 2ec-block containing e satisfy (2) and (3) of the Bridgeless Invariant.

We will now describe the two phases of the algorithm - Phase 0 and Phase 1. Recall that Phase 0 is performed on a main component C_0 that does not contain a large 2ec-block. Otherwise, C_0 and the large 2ec-block R maintain the Bridgeless Invariant, so we skip Phase 0.

To buy a fair path covering an edge e of the growing component, we sometimes use tokens from the 2ec-blocks that neighbour a vertex of this fair path in some component (as described in the proof of Lemma 3.8). Since these 2ec-blocks become neighbours of the new 2ec-block covering e in the new growing component, the loss of tokens on them is accommodated by the Bridgeless Invariant. After the following analysis, it will be easy to observe that in this process, we do not take any fractional tokens from a 2ec-block more than one time.

Since Phase 1 is much easier to describe, we will start with this phase.

Phase 1

At the start of this phase, the growing component C_0 and some block R of C_0 that we have already selected maintain the Bridgeless Invariant. At the end of this phase, the growing component will be 2-edge connected (bridgeless), and have at least 2 spare tokens. In this phase, we repeat the following procedure as long as the growing component contains a bridge. In each iteration, we select a bridge $e = ab$ such that a is a vertex of the root block R, and we buy a path to cover this edge.

By Fact 3.6, b has degree 2 in C_0. Let c be the other vertex adjacent to b.

Let T be the tree $\mathcal{T}(C_0)$, and T_1 and T_2 be the components of $T \setminus e$ as before. We find a fair path P covering e that is T_2-maximum for the set of all such paths. We then buy the path P, for which we require one extra token (by Lemma 3.8), which we

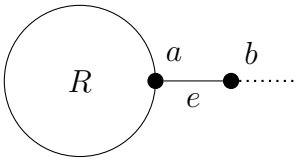


Figure 3.15: Phase 1

will obtain later. Let $p \in \mathcal{T}^{-1}(T_2)$ be an endpoint of P. Since $G \setminus \{b\}$ is connected, $p \neq b$.

If c is in a 2ec-block, then the resulting root 2ec-block has at least 3 tokens: 2 from R, $\frac{1}{3}$ from each of the edges ab and bc, and at least $\frac{1}{2}$ from the 2ec-block containing c. We use one token to buy P and assign the other two tokens to the new 2ec-block to maintain the Bridgeless Invariant.

Otherwise, the p, b-path in C_0 has at least 2 bridges: otherwise, $p = c$, but $G \setminus \{b, c\}$ is connected, contradicting the assumption that P is T_2-maximum for the set of all fair paths covering e. Hence the resulting 2ec-block has at least 3 bridges, and has at least 3 tokens: 2 from R, and $\frac{1}{3}$ from each of the 3 bridges. We use one token to pay for P and assign the others to the new 2ec-block to satisfy the Bridgeless Invariant.

Phase 0

We carry out this phase of the algorithm if the growing component has no large 2ec-block. At the end of this phase, the growing component C_0 (and some block R of C_0) satisfy the Bridgeless Invariant. We start by choosing a leaf node of $\mathcal{T}(C_0)$. Let R be the corresponding 2ec-block of this component. Since R is a leaf, it is incident with exasctly one bridge e of C_0. Let $e = ab$ with $a \in V(R)$. The following lemma will be useful in our analysis of Phase 0.

Lemma 3.9. *Let P be a fair path covering e. Suppose that we buy the edges of P to create a new 2ec-block R'. Let k be the number of bridges covered by P, and t_C be the number of tokens on all of the 2ec-blocks of C_0 that are now in R'. If $t(R) + \frac{1}{3}k + t_C \geq 3$, we can terminate Phase 0.*

Proof. By Lemma 3.8, we require one extra token to buy the edges of $E(P) \setminus E(H)$ and satisfy (2) and (3) of the Bridgeless Invariant. To satisfy (1), we require that the number of tokens that we can assign to R' is at least 2, hence we need a total of 3 tokens on R'. If we obtain these tokens from R, the k bridges, and the 2ec-blocks of C_0 that merge with R, then we can terminate Phase 0. □

Case 1. R is a 3-cycle.

We choose R to be the root block for this iteration of Phase 0. Let $V(R) = \{a, x, y\}$. Since R is a 2ec-block in a component that has a bridge, and since the corresponding compound node is a leaf of the underlying tree, there is exactly one bridge $e = ab$ incident on a vertex of R (without loss of generality, incident on a).

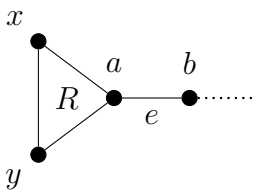


Figure 3.16: Phase 2, Case 1

Let $T = \mathcal{T}(C_0)$, and let T_1 and T_2 be the two components of $T \setminus e$, such that the compound node of R is in T_1.

By linear-time graph search from the vertices x and y, we find a fair path P in G covering e, in polynomial time, that is T_2-maximum for the set of fair paths that end at either x or y (see Lemma 3.7). Without loss of generality, let x be the endpoint of P. We buy the edges of P, for which we require one extra token, which we will obtain later. Since the pair of vertices (x, a) has 3 edge disjoint paths: the edge xa, the path $xy - ya$, and the path through P, we sell the edge xa to obtain an extra token and assign this token to R. Since C_0 maintains the General Invariant before the start of Phase 0, R had at least 1 token, so now R has 2 tokens. The resulting component has a 2ec-block R' containing e.

Let p be the other endpoint of P. If the p, b-path in T_2 either contains a compound node, or has at least 2 bridge edges, then by Lemma 3.9 C_0 and R' maintain the Bridgeless Invariant and we can terminate Phase 0.

Otherwise, the p, b-path in T_2 has exactly 1 bridge (since G is 3-connected, $G \setminus \{a, b\}$ is connected, hence there exists a fair path covering at least 2 bridges in this case). Let $c = p$. Since c is not in a 2ec-block, it is adjacent to at least two bridges of the main component.

Let $\tilde{e} = bc$, and let T_1' and T_2' be the components of $T \setminus \tilde{e}$. By Fact 3.6, b has degree 2 in H. Hence the subtree $T_1' \cap T_2$ contains just the isolated vertex b. Consider the graph $G' = G \setminus \{a, c\}$. Since G is 3-connected, this graph is connected, thus every pair of vertices have a path in G' between them. Hence there is a path Q in G' with one endpoint at b and the other endpoint in $T_2' \setminus \{c\}$,

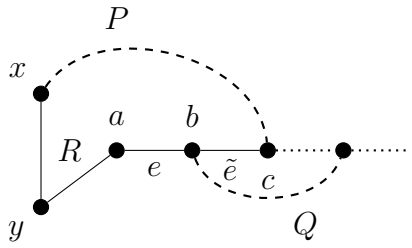


Figure 3.17: Phase 2, Case 1

whose internal vertices are not in the growing component. This path is a fair path covering some bridge cd incident on c in H, with the additional property that the set of components of the internal vertices of Q is disjoint from the set of components of the internal vertices of P (otherwise, P is not T_2-maximum, since there exists a fair path covering e through this common component, that covers more bridges than P), hence each of these components maintains the General Invariant before this operation (and we have the required tokens to buy Q by Lemma 3.8).

Using Lemma 3.8, we buy the fair path Q, for which we require one extra token. We can sell the edge bc and recover one token, since the resulting block is still 2-edge-connected, because b and c had 3 edge-disjoint paths between them before this operation. Let R'' be the new 2ec-block containing a, b and c. Now, paths P and Q together cover at least 3 bridges (e, $\tilde{e}$, and cd), which contribute a total of at least 1 token to R''. We get an additional 2 tokens from R and one token from selling bc, for a total of 4 tokens. We use 2 tokens to buy paths P and Q and assign 2 tokens to the resulting 2ec-block R'', which maintains the Bridgeless Invariant.

Case 2. R is a 4-cycle.

We choose R to be the root block for this iteration of Phase 0. Let $V(R) = \{a, x, z, y\}$, such that a is adjacent to x and y as shown in Figure 3.18. Since R is a 2ec-block in a component that has a bridge, and since the corresponding compound node is a leaf of the underlying tree, there is exactly one bridge $e = ab$ incident on a vertex of R (without loss of generality, incident on a).

Let $T = \mathcal{T}(C_0)$. Let T_1 and T_2 be the two components of $T \setminus e$, such that the compound node of R is in T_1.

Consider the connected graph $G \setminus \{z, a\}$. As explained earlier, we find a T_2-maximum fair path P covering e, for the set of fair paths with an endpoint at

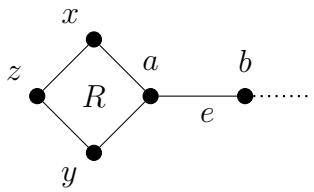


Figure 3.18: Phase 2, Case 2

either x or y (assume without loss of generality that P has an endpoint at x in G). Let $p \in \mathcal{T}^{-1}(T_2)$ be the other endpoint of P. We buy the edges of P, for which (by Lemma 3.8) we require an extra token, which we obtain later. We sell the edge xa and assign its token to R. Since C_0 maintains the General Invariant at the start of Phase 0, R had at least $\frac{4}{3}$ tokens, so R now has at least $\frac{7}{3}$ tokens. As described earlier, this operation creates a new 2ec-block R' containing e and the vertices of R and P along with some vertices of the growing component.

If the p, b-path in T_2 has at least 1 bridge edge, then by Lemma 3.9 C_0 and R' maintain the Bridgeless Invariant and we can terminate Phase 0.

Otherwise, $p = b$. Since b has degree 2, it has 2 bridges ab and bc of the main component incident on it. Consider the graph $G \setminus \{z, b\}$. Since G is 3-connected, $G \setminus \{z, b\}$ is connected, hence there exists a path Q in G with one endpoint at a and the other endpoint at $b' \in \mathcal{T}^{-1}(T_2) \setminus \{b\}$. Q ends at a and not x or y, since otherwise it would contradict the assumption that P is T_2-maximum. Hence Q is a fair path covering cd in the main component. Since P is T_2-maximum, the set of components containing internal vertices of P is disjoint from the set of components containing internal vertices of Q.

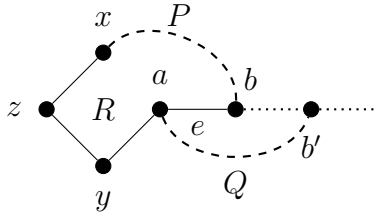


Figure 3.19: Phase 2, Case 2

Using Lemma 3.8, we buy the fair path Q, for which we require an extra token. Let R'' be the resulting 2ec-block.

We sell the edge e to recover one token, for a total of 4 tokens ($\frac{7}{3}$ from R, $\frac{1}{3}$ from each of ab and bc, and 1 from selling e). We use 2 tokens to pay for P and Q,

and assign the remaining 2 tokens to R'' to maintain the Bridgeless Invariant with C_0 and R''.

Case 3. R is a 5-cycle.

We choose R to be the root block for this iteration of Phase 0. Let $V(R) = \{a, w, x, y, z\}$, labelled in order around the cycle as shown in Figure 3.20. Let $e = ab$ be the bridge of the main component incident on a vertex of R (without loss of generality, incident on a).

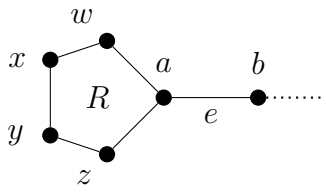


Figure 3.20: Phase 2, Case 3

Let $T = \mathcal{T}(C_0)$. Let T_1 and T_2 be the two components of $T \setminus e$, such that the compound node R is in T_1.

We find a fair path P covering e, starting at any vertex of R that is T_2-maximum for the set of all such paths. Let $p \in V(\mathcal{T}^{-1}(T_2))$ be the other endpoint of P. We buy the path P, which requires one extra token by Lemma 3.8, which we will obtain later. Let F' be the new wec-block containing e. Since C_0 maintains the General Invariant at the start of this phase, R has at least $\frac{5}{3}$ tokens.

If the p, b-path in T_2 either contains a compound node or has at least 3 bridge edges, then by Lemma 3.9 C_0 and R' maintain the Bridgeless Invariant and we can terminate Phase 0.

Otherwise, the p, b-path in T_2 has exactly 2 bridge edges. To see this, let c be the neighbour of b on this path. Since G is 3-connected, $G \setminus \{b, c\}$ is connected and hence contains a path from a vertex of R to a vertex of $\mathcal{T}^{-1}(T_2) \setminus \{b, c\}$ which is a fair path covering e. Hence p cannot be b or c, and p is a neighbour d of c (see Figure 3.21). Since d is not in a 2ec-block, it has at least 2 bridges of the main component incident on it.

Let T_1' and T_2' be the underlying trees of $T \setminus cd$, where T_1' contains the compound node corresponding to R. Consider the graph $G \setminus \{b, d\}$. This graph is connected, and thus has a path Q from c to a vertex of T_2'. Since Q covers cd, some subpath Q' of Q is a fair path covering cd in the main component. Since P was chosen to be T_2-maximum, the set of components containing internal vertices of Q' is disjoint from the set of components containing internal vertices of P.

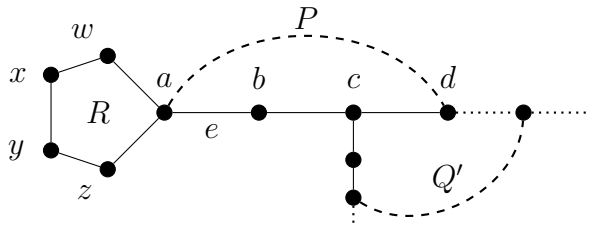


Figure 3.21: Phase 2, Case 3

Using Lemma 3.8, we buy the fair path Q' for one extra token. The paths P and Q' together cover at least 4 bridges of C_0.

We sell the edge cd to obtain an extra token. We now have at least 4 tokens ($\frac{5}{3}$ from R, 1 from selling cd, and $\frac{4}{3}$ from the 4 bridges of C_0 that we covered with P and Q'). We use two tokens to buy P and Q and assign the remaining two tokens to the resulting 2ec-block so that it maintains the Bridgeless Invariant.

On termination of the above algorithm, the graph H is a bridgeless-$D2$ of G. Further, every component of H that was a main component at some stage of the algorithm has at least 2 tokens. Every component of H that was never a main component at some stage was bridgeless in $\mathcal{D}$, and hence maintains the General Invariant. Let C be a component of H that was never a main component of the above algorithm. If $|E(C)| \geq 6$, then $t(C) \geq 2$ by the General Invariant. Otherwise, if $|E(C)| \in \{3, 4, 5\}$, then $t(C) = \frac{1}{3}|E(C)|$ by the General Invariant. Hence the solution H output by the above algorithm satisfies the Gluing Invariant, and can be used in the Gluing step.

Observe that every step of the above algorithm can be carried out in polynomial time. In each step, we cover at least one bridge, thus the number of steps is polynomial in the size of the input, and the overall running time is polynomial.

We will now prove Lemmas 3.7 and 3.8.

Lemma 3.7. *Let G be a 2-edge-connected graph and H be a $D2$ of G. Let $e = ab$ be a bridge of some component C_0 of H. Let $T = \mathcal{T}(C_0)$ and let T_1 and T_2 be the components of $T \setminus e$ (such that a is in T_1 or in a compound node of T_1). Let $V_1 \subseteq \mathcal{T}^{-1}(T_1)$ and $V_2 \subseteq \mathcal{T}^{-1}(T_2)$ be disjoint subsets of $V(C_0)$. A fair path that is T_2-maximum for the set of all fair paths covering e with endpoints in V_1 and V_2 can be computed in polynomial time.*

Proof. We first compute T_1 and T_2. Then, we delete all edges of C in G, then perform a linear-time graph search in the resulting graph from each vertex in V_1. Each time that we

reach a vertex in V_2, we find the length of the T_2-path from that vertex or its compound node to b. We find a path with the longest such length, and shortcut this path as before. We return the resulting path.

Let P be a T_2-maximum fair path (for the given set of fair paths covering e) with endpoint $x_P \in V_1$. This algorithm finds P or a path of equal length by linear-time graph search from x_P, and hence computes a T_2-maximum fair path for the set of all fair paths covering e, in polynomial time. □

Lemma 3.8. *Let P be a T_2-maximum fair path (for some set) covering e in H. Let $\mathcal{C}$ be the set of components containing a vertex of P. Then*

1. $|E(P) \setminus E(H)| \leq 1 + \sum_{C \in \mathcal{C}} t(C)$.

2. We can use the tokens on components in $\mathcal{C}$ to buy the edges of P such that the resulting growing component and 2ec-block containing e satisfy (2) and (3) of the Bridgeless Invariant.

Proof. Let P be a fair path covering e. Consider any component $C \in \mathcal{C}$. Since P is a fair path, there is a unique edge $f_P(C)$ of P that "enters" C (with one end in C and the other in the subpath of P between T_1 and C) and a unique edge $e_P(C)$ that "exits" C (with one end in C and the other in the subpath of P between C and T_2).

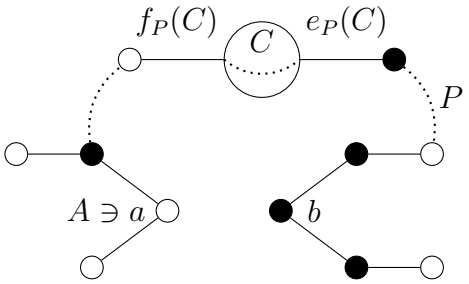


Figure 3.22: $f_P(C)$ and $e_P(C)$ for some component C containing a vertex of P (white nodes are compound nodes)

Let s and t be the vertices in C that are incident with $f_P(C)$ and $e_P(C)$ respectively. If s or t is in a 2ec-block B of C, we take one token from B and use it to buy the edge $f_P(C)$. Otherwise, consider the underlying tree $\mathcal{T}(C)$ of C. This tree has a unique path Q from s (or a compound node containing s) to t (or a compound node containing t), all of whose edges are in P. If this path contains a compound node internally or at an endpoint, then we use one token from the corresponding 2ec-block and use it to buy the edge $f_P(C)$.

Otherwise, this path does not contain any compound nodes. We find a neighbouring 2ec-block $B(s)$ of s and a neighbouring 2ec-block $B(t)$ of t such that $B(s)$ and $B(t)$ are distinct (see Figure 3.23). Observe that such blocks exist: we may delete the edges of Q from the underlying tree of C, and traverse the resulting trees containing s and t from the nodes s and t respectively (with a linear-time graph search), until the distinct compound nodes $B(s)$ and $B(t)$ are found. Since every leaf of the underlying tree is a compound node, this procedure will always terminate and find suitable 2ec-blocks. We take 0.5 tokens from each of $B(s)$ and $B(t)$ and buy the edge $f_P(C)$.

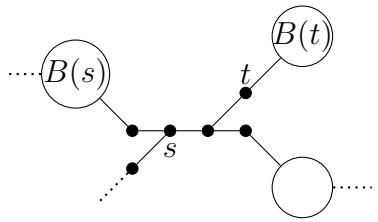


Figure 3.23: Neighbours $B(s)$ and $B(t)$ of nodes s and t in the component C

By repeating the above process for every component that contains an internal node of P, we can buy all edges of $E(P) \setminus E(H)$ except one, hence $|E(P) \setminus E(H)| \leq 1 + \sum_{C \in \mathcal{C}} t(C)$.

Any 2ec-block from which we take 1 token during this procedure becomes part of the block containing the edge e. Any 2ec-block from which we take $\frac{1}{2}$ tokens becomes a neighbour of this block. Hence the growing component and this block satisfy (2) and (3) of the Bridgeless Invariant. □

Chapter 4

A $\frac{17}{12}$-Approximation for Restricted 2VCSS

In this chapter, we present a polynomial-time $\frac{17}{12}$-approximation algorithm for the minimum-cost 2-vertex-connected spanning subgraph problem (2VCSS), restricted to input graphs of minimum degree at least 3. Our algorithm uses the framework of ear-decompositions for approximating connectivity problems, which was previously used in algorithms for finding the smallest 2-edge-connected spanning subgraph by Cheriyan, Sebő and Szigeti [4], who gave a $\frac{17}{12}$-approximation algorithm for the general 2ECSS problem, and by Sebő and Vygen [28], who improved the approximation ratio to $\frac{4}{3}$.

The unweighted 2VCSS problem has been well studied. Khuller and Vishkin [22] gave a $\frac{5}{3}$-approximation algorithm. Garg, Santosh and Singla [14] improved this ratio to $\frac{3}{2}$. Very recently, Heeger and Vygen [16] reported an approximation ratio of $\frac{10}{7}$. Our research was carried out independently in the same time period.

We present a new $\frac{17}{12}$-approximation algorithm for 2VCSS, restricted to instances with no degree-2 vertices. Our algorithm is inspired by the $\frac{4}{3}$-approximation algorithm for 2ECSS in [28], and our methods are similar. However, as explained in section 2.2, our goal is to compute an open nice ear-decomposition. Our algorithm manages to do so for graphs without degree-2 vertices. Consequently, we have the following theorem.

Theorem 4.1. *Let G be a 2-connected graph with no degree-2 vertices. Then G has an open nice ear-decomposition. Such an ear-decomposition can be computed in polynomial time.*

4.1 Algorithm Overview

Our algorithm consists of a few steps, summarized as follows.

1. We construct an open evenmin ear-decomposition D of G.
2. We modify D to get an open evenmin ear-decomposition with the property that all of its short ears are pendant ears.
3. We modify D to get an open evenmin ear-decomposition that is nice.
4. We delete all edges in trivial ears. The resulting graph is a 2-vertex-connected spanning subgraph of G with at most $\frac{17}{12}OPT_{2VC}(G)$ edges.

Our analysis is detailed in the rest of this chapter.

The following lemma (Lemma 4.2) allows us to replace a given ear-decomposition on a graph with another ear-decomposition on the same graph, by locally changing a pair of ears in order to reduce the number of non-pendant ears in the ear-decomposition. It is used multiple times in our algorithm. The lemma is very simple to prove, and its proof is omitted. The reader may find Figure 4.1 useful for understanding the statement of the lemma.

Lemma 4.2. *Let D be an ear-decomposition of a graph G. Suppose P and Q are nontrivial ears of D such that Q is the first nontrivial ear of D with an endpoint in an internal vertex of P. Further, suppose that only one of the endpoints of Q is an internal vertex of P, and that this endpoint is adjacent, by an edge of P, to an endpoint of P. Let x and y be the end vertices of P and w and z be the end vertices of Q, such that w is the internal vertex of P adjacent to y.*

Let P' be the ear with endpoints x and z and consisting of all edges of P and Q except wy. Let D' be the ear-decomposition constructed from D by deleting the ears P and Q, adding the ear P' in the position of Q, and adding the trivial ear wy at the end of the ear-decomposition. Then D' is a valid ear-decomposition of G.

Figure 4.1: A local change that reduces the number of non-pendant ears

4.2 Making Short Ears Pendant

Theorem 4.3. *Every 2-vertex-connected graph G with minimum degree at least 3 has an open ear-decomposition with $\varphi(G)$ even ears in which all short ears are pendant. Such an ear-decomposition can be computed in polynomial time.*

Using Proposition 3.2 of Cheriyan, Sebő and Szigeti [4], we construct an open ear-decomposition $D = (P_0, P_1, \ldots, P_k)$ of G with $\varphi(G)$ even ears.

Suppose the closed ear P_1 is short (that is, P_1 is a 3-ear). Since every vertex of G has degree at least 3, G has at least 4 vertices, hence D has at least one open ear. Suppose u and v are the end vertices of P_2, then there is a u, v-path of length 2 in P_1. Let P' be the union of the u, v-path in P_1 and the u, v-path in P_2. We delete the ears P_1 and P_2 from D, and add the ear P' in the position of P_1 in D, and the trivial ear uv at the end of D. Now the closed ear in D is a long ear, and D is still evenmin. We set $k := k - 1$ and relabel the new ears of D such that $D = (P_0, P_1, \ldots, P_k)$.

Figure 4.2: The cycle-ear P_1 is a short ear

We proceed to make all other short ears pendant, starting with 2-ears. As long as D has a non-pendant 2-ear, we repeat the following procedure. We choose the first non-pendant 2-ear P in D. Since P is non-pendant, there exists a nontrivial ear in D with one end incident on the internal vertex z of P. Let Q be the first such ear in D.

Let u and v be the end vertices of P and x and z be the end vertices of Q such that $u \neq x$. We delete ears P and Q from D, and construct the ear P' with ends at u and x, containing internally the internal vertices of both P and Q, as shown by the thick line in Figure 4.3b. We add the ear P' to D in the position of Q. We add the trivial ear vz at the end of D. By Lemma 4.2, D is still a valid ear-decomposition of G. Since Q was a nontrivial ear, P' has length at least 3, hence this procedure reduces the number of 2-ears in D by one. Further, P' is an open ear, thus D is still an open ear-decomposition. If Q

Figure 4.3: P is a 2-ear

was an even ear, then this procedure reduced the number of even ears by 2, contradicting our assumption that D is evenmin. Hence Q was an odd ear, and the number of even ears remains unchanged in D.

After repeating the above procedure for all non-pendant 2-ears, all 2-ears in D are pendant. Next, we make all 3-ears pendant. As long as D has a non-pendant 3-ear, we repeat the following procedure. Prior to each iteration, we relabel the ears in D such that the i^{th} ear is labelled P_i.

Let P be the first non-pendant 3-ear in D. Let x and z be the endpoints of P, and let v and y be the internal vertices of P adjacent to x and z respectively (as shown in Figure 4.4).

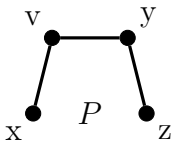


Figure 4.4: P is a 3-ear

Case 1. There exists a nontrivial ear Q with endpoints v and y.

Let P' be the ear with endpoints x and z consisting of all of the edges of Q and the edges vx and yz (as shown by the thick dashed line in Figure 4.5b). We delete the ears P and Q from D, and add the ear P' to D in the position of P, and the trivial ear xy at the end of D. The resulting ear-decomposition D is valid for G. Since Q is nontrivial, P' has length at least 4 and is a long ear. Further, D is still an open ear-decomposition, and since the length of P' has the same parity as the length of Q, D is still evenmin.

Figure 4.5: There exists a nontrivial ear Q with endpoints v and y

Case 2. There exist ears Q_1 and Q_2 such that Q_1 has endpoints x and y, Q_2 has endpoints v and z, and at least one ear in $\{Q_1, Q_2\}$ is nontrivial.

Let P' be the ear with endpoints x and z consisting of all the edges of Q_1 and Q_2 and the edge vy (as shown by the thick dashed line in Figure 4.6b). We delete the ears P, Q_1 and Q_2 from D, and add the ear P' to D in the position of P, and the trivial ears ax and by at the end of D. The resulting open ear-decomposition D is valid for G. If both Q_1 and Q_2 are even ears, then this procedure reduces the number of even ears in D by 2, contradicting our assumption that D was evenmin. If either zero or exactly one of these ears is even, then D remains evenmin.

Figure 4.6: There exist ears Q_1 from x to y and Q_2 from v to z, not both trivial

Case 3. Otherwise, let Q be the first nontrivial ear with an endpoint at an internal vertex of P (say y). Let w be the other endpoint of Q (as shown in Figure 4.7).

Case 3a. $w \neq x$. Let P' be the ear with endpoints x and w, consisting of the edges of P and Q except yz. We delete the ears P and Q from D, add the ear

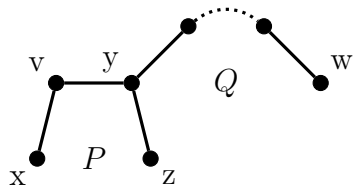


Figure 4.7: Case 3

P' to D in the position of Q, and the trivial ear yz at the end of D. P' is both open and long, and the resulting ear-decomposition D is valid for G by Lemma 4.2. Since the length of P' has the same parity as the length of Q, the number of even ears remains the same.

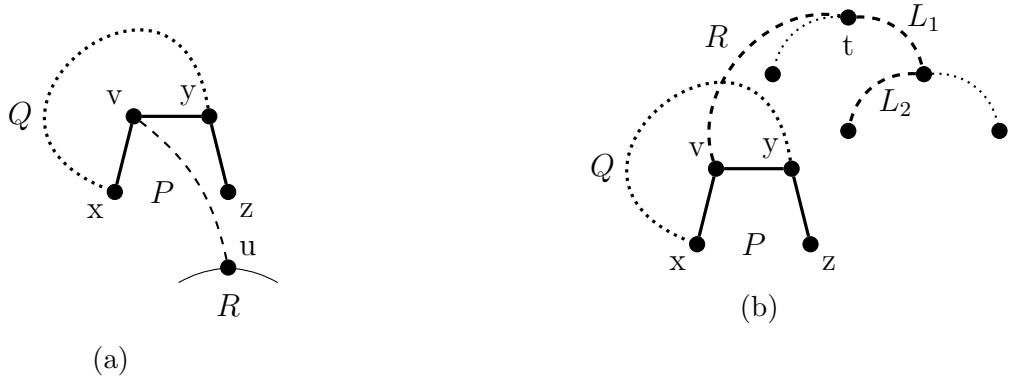


Figure 4.8: Case 3

Case 3b. $w = x$, and v is the endpoint of a trivial ear uv such that $u \in X$. We refer the reader to Figure 4.8a for this case.

Let R be the ear containing u internally. If R is a short ear, then it is pendant (since P is the first non-pendant short ear). We have the following cases:

(i) R is a 2-ear. We choose an endpoint a of R that does not coincide with z, and let P' be the ear $au \cup uv \cup vy \cup yz$. We delete P and R from D and add the 4-ear P' in the position of P, and the new trivial ears at the end of D.

(ii) R is a 3-ear. We choose the endpoint a of R that is not adjacent to u in R. Let R' be the ear of length 2 in R with endpoints a and u. If a does

not coincide with x, let P' be the ear $R' \cup uv \cup vy \cup Q$, which has the same parity as Q. We delete R, P and Q from D and add P' in the position of P in D and the trivial ears at the end of D. If a coincides with x, let P' be the ear $R' \cup uv \cup vy \cup yz$ of length 5. We delete R and P from D and add P' in the position of P in D and the trivial ears at the end of D.

In both of the above cases, we do not create extra even ears, and the resulting ear-decomposition is open, evenmin, and valid for G.

If R is a long ear, let P' be the ear $Q \cup yv \cup vu$. We delete P and Q from D and add P' in the position of P in D, and the trivial ears at the end of D. This ear has the same parity as the ear Q, so the resulting ear-decomposition is open, evenmin, and valid for G.

Case 3c. Otherwise, since the graph has minimum degree at least 3, v is adjacent to a vertex $u \notin X \cup \{y\}$. Observe that this is the only remaining case.

Let R be the ear containing the edge uv, and let t be the other endpoint of R. In particular, if uv is a trivial ear, then t is the vertex u. We refer the reader to Figure 4.8b, which will be useful throughout the following analysis.

The following sub-procedure constructs three sets of ears ($\mathcal{F}^{old}$, $\mathcal{F}_1^{new}$ and $\mathcal{F}_0^{new}$), that are later used to modify the ear-decomposition in order to add the internal vertices of P to a new long ear. The procedure adds some of the existing ears of D to the set $\mathcal{F}^{old}$, and constructs sets of new ears $\mathcal{F}_1^{new}$ and $\mathcal{F}_0^{new}$. When suitable sets are found, the ears in $\mathcal{F}^{old}$ are deleted from D and replaced with the ears in $\mathcal{F}_0^{new}$, along with a suitably constructed long ear.

We repeat the following sub-procedure until t is in $X \cup \{v, y\}$. We initialize $\mathcal{F}^{old}$, $\mathcal{F}_1^{new}$ and $\mathcal{F}_0^{new}$ with the empty set. Let S be the ear that internally contains t, with endpoints c and d. We add S to $\mathcal{F}^{old}$. We partition the edges of S into the ears S_1^{new} (with endpoints c and t) and S_0^{new} (with endpoints t and d). If S is an even ear, either S_1^{new} and S_0^{new} are both odd or they are both even. If S is an odd ear, suppose without loss of generality that S_1^{new} is even and S_0^{new} is odd. We add S_1^{new} to $\mathcal{F}_1^{new}$ and S_0^{new} to $\mathcal{F}_0^{new}$. We set $t = c$.

The following observations will be useful in our analysis.

- If S_0^{new} is even, then S is even (thus replacing S with S_0^{new} in any ear-decomposition will not, by itself, increase the number of even ears in that ear-decomposition).
- If S_1^{new} is odd, then S_0^{new} is odd and S is even.

When this sub-procedure terminates, we have the following cases:

(i) $t \notin \{v, y, z\}$. Let P' be the ear $zy \cup yv \cup R \cup L_1 \cup \ldots \cup L_k$, where $\mathcal{F}_1^{new} = \{L_1, \ldots, L_k\}$. We delete P and all ears in $\mathcal{F}^{old}$ from D, and for each ear in $\mathcal{F}^{old}$, replace it with the corresponding ear in $\mathcal{F}_0^{new}$ at the same position in D (this does not add any extra even ears, but might create new non-pendant short ears; observe that these ears occur later in the ear-decomposition than the newly created long ear in this iteration). In the position of P, add the ear P'. We add the trivial ear xv at the end of D. The resulting ear decomposition is valid because the sub-procedure is terminated when a vertex in X is encountered, thus every ear in $\mathcal{F}^{old}$ appeared after P in D, hence every ear in $\mathcal{F}_0^{new}$ appears after P'.
If $\mathcal{F}_1^{new}$ contains only even ears, then P' has the same parity as R and we do not introduce any extra even ears. If not, then $\mathcal{F}_1^{new}$ contains at least one odd ear, in which case the corresponding ear in $\mathcal{F}^{old}$ is even, and the corresponding ear in $\mathcal{F}_0^{new}$ is odd. Since we have already reduced the number of even ears by at least one, P' is even and we do not introduce extra even ears.

(ii) $t = v$. Discard the previous sets $\mathcal{F}^{old}$, $\mathcal{F}_1^{new}$ and $\mathcal{F}_0^{new}$. We choose u to be the neighbour of v on the last ear that was labelled S. Let R be this ear and let t be its other endpoint. Since every choice of R that we make in this manner appears strictly earlier in the ear decomposition than all the previous choices, the sub-procedure can only be repeated $O(n)$ times before we no longer have this case.

(iii) $t = y$. Let $\mathcal{F}_1^{new} = \{L_1, \ldots, L_k\}$, and let P' be the ear $xv \cup R \cup L_1 \cup \ldots \cup L_k \cup yz$. We delete P and all ears in $\mathcal{F}^{old}$ from D, and for each ear in $\mathcal{F}^{old}$, replace it with the corresponding ear in $\mathcal{F}_0^{new}$ at the same position in D. In the position of P, add the ear P'. We add the trivial ear vy at the end of D. This ear-decomposition is valid, as explained earlier.
If all of the ears in L are even, then P' has the same parity as R, and this step does not introduce any extra even ears. If not, then L contains at least one odd ear, in which case P' is even and we do not introduce extra even ears (as explained earlier).

(iv) $t = z$. Let P' be the ear $Q \cup vy \cup R \cup L_1 \cup \ldots \cup L_k$, where $\mathcal{F}_1^{new} = \{L_1, \ldots, L_k\}$. We delete P, and in the position of P, add the ear P'. We delete all ears in $\mathcal{F}^{old}$ from D, and for each ear in $\mathcal{F}^{old}$, replace it with the corresponding ear in $\mathcal{F}_0^{new}$ at the same position in D. We add the trivial ears xv and yz at the end of D. As explained earlier, this ear-decomposition is valid. If all the ears in $\mathcal{F}_1^{new}$ are even, we have the following cases:

(a) Q and R are odd. In this case, P' is odd and we do not introduce extra even ears.

(b) Exactly one of Q and R is even. In this case, P' is even and we do not introduce extra even ears.

If Q and R are both even, P' is odd, contradicting the assumption that D was evenmin; this case cannot occur.

If $\mathcal{F}_1^{new}$ contains an odd ear, then the corresponding ear in $\mathcal{F}_0^{new}$ is odd and the corresponding ear in $\mathcal{F}^{old}$ is even. Since we have already reduced the number of even ears by at least one, we do not introduce extra even ears.

In all of the above cases, the internal vertices of P are added to a long ear in D. If P was a 3-ear, then it is possible that we created new non-pendant short ears that appear after P' in the new ear-decomposition. These short ears are handled in future iterations of the above procedure, in the same manner as above (that is, first we handle all non-pendant 2-ears, then we handle the first non-pendant 3-ear).

In each iteration, the above procedure takes time polynomial in $|V(G)|$ for the non-pendant short ear under consideration. Further, if this short ear is a 3-ear, the internal vertices of this short ear are added to long ears, and are never again added to a non-pendant short ear until termination. As a consequence, the set X in each iteration is a strict superset of the corresponding set in any previous iteration. Hence the running time for the whole procedure is polynomial.

On termination of this procedure, the ear-decomposition D is open and has $\varphi(G)$ even ears, and all of its short ears are pendant.

4.3 Making Short Ears Non-Adjacent

Theorem 4.4. *Given a 2-vertex-connected graph G with minimum degree at least 3, and an associated evenmin ear-decomposition D in which all short ears are pendant, an open nice ear-decomposition of G can be computed in polynomial time.*

Since D is open and evenmin, and all short ears of D are pendant, it remains to obtain the property that there are no edges connecting an internal vertex of one short ear to an internal vertex of another short ear of D.

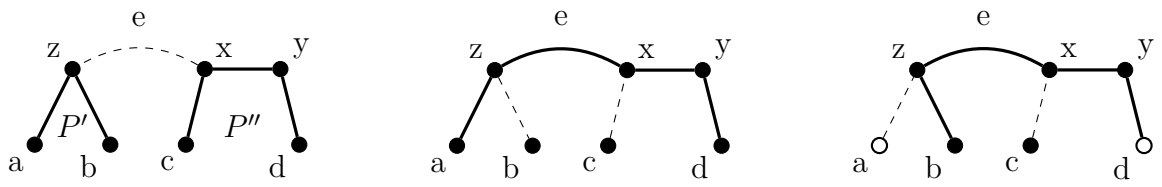
Since D has $\varphi(G)$ even ears, there are no edges connecting the internal vertices of 2-ears. If not, we could replace the 2-ears and the trivial ear connecting their internal vertices by a pendant 3-ear and two trivial ears, reducing the number of even ears by two, contradicting the assumption that D is evenmin. Since we have two choices for each end vertex of such a 3-ear, we can always choose its end vertices such that it is open.

As long as D has two short pendant ears P' and P'' with an edge e connecting an internal vertex of P' with an internal vertex of P'', we repeat the following procedure.

Case 1. One of the ears P' and P'' is a 2-ear.

Without loss of generality, assume P' is a 2-ear and P'' is a 3-ear.

Let a and b be the endpoints of P' and z be the internal vertex of P'. Let c and d be the endpoints of P'' and x and y be the internal vertices of P'' such that x is adjacent to both c and z (Figure 4.9a). We construct the ear S as shown by the thick paths in Figures 4.9b and 4.9c, that is, S consists of the edges az, zx, xy and yd if the vertices a and d are distinct, and the edges bz, zx, xy and yd if they coincide.



(a) (b) a and d are distinct (c) a and d coincide

Figure 4.9: P' is a 2-ear

We remove the ears P' and P'' from D, and add the ear S in place of the ear P', followed by trivial ears consisting of the remaining edges from P' and P'' that are

not in S. Since P' and P'' are both pendant ears, the new ear-decomposition is a valid ear-decomposition of G. Since the end vertices of S are distinct, it is open, and since we deleted a 2-ear from D before adding a 4-ear to it, the number of even ears in D remains equal to $\varphi(G)$.

Case 2. Both P' and P'' are 3-ears.

Let a and b be the endpoints of P' and let v and w be its internal vertices adjacent to a and b respectively. Let c and d be the endpoints of P'' and let x and y be its internal vertices adjacent to c and d respectively (Figure 4.10). Suppose v and y are adjacent. We have the following cases.

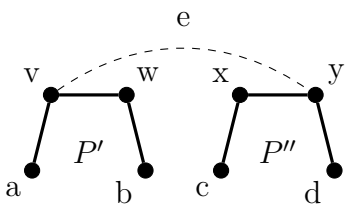


Figure 4.10: Both P' and P'' are 3-ears

Case 2a. The vertices b and c are distinct.

We construct the ear S with endpoints b and c and edges bw, wv, vy, yx and xc (as shown by the thick path in Figure 4.11). We remove the ears P' and P'' from D, add the ear S in place of the ear P', and add the trivial ears consisting of the remaining edges from P' and P'' that are not in S at the end of D. Since P' and P'' are both pendant ears, the new ear-decomposition is a valid ear-decomposition of G. Since the end vertices of S are distinct, it is open, and since S is an odd ear, the number of even ears in D remains equal to $\varphi(G)$.

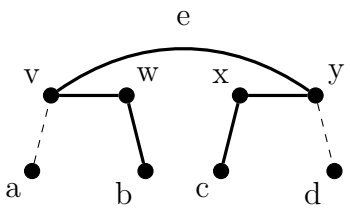


Figure 4.11: b and c are distinct

Case 2b. The vertices b and c coincide, as shown in Figure 4.12.

Since every vertex of the graph has degree at least 3, x is adjacent to some vertex not in the set $\{b, y\}$.

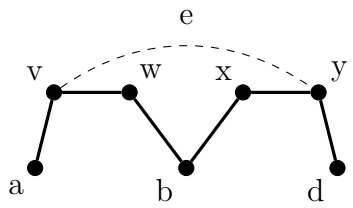
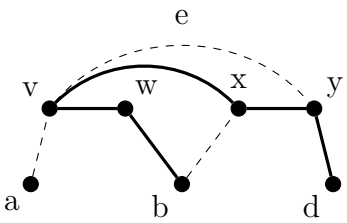


Figure 4.12: b and c coincide

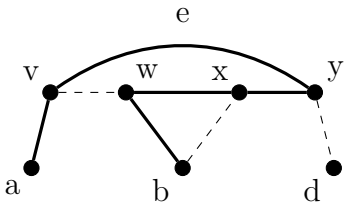
Case 2b.I. x is adjacent to an internal vertex of P'.

If x is adjacent to v, we construct the ear S with endpoints b and d and edges bw, wv, vx, xy and yd (as shown by the thick path in Figure 4.13a). Otherwise, if x is adjacent to w, we construct the ear S with endpoints a and b and edges av, vy, yx, xw and wb (as shown by the thick path in Figure 4.13b). In either case, We delete P' and P'' from D, add S to D in place of P', and add all of the remaining edges (dashed edges in the corresponding figure) in trivial ears at the end of D.

In both cases, the ear S is an odd long ear with distinct end points, hence the new ear-decomposition is open and is valid for G, and the number of even ears remains equal to $\varphi(G)$.



(a) x is adjacent to v



(b) x is adjacent to w

Figure 4.13: x is adjacent to an internal vertex of P'

Case 2b.II. x is adjacent to an internal vertex z of an ear R not equal to P'.

Since the input graph is simple and does not have parallel edges, z does not coincide with b or y. If R is a long ear, we construct the ear S with endpoints b and z and edges bw, wv, vy, yx and xz (as shown by the thick

path in Figure 4.14a). We delete P' and P'' from D, add S to D in place of P', and add all of the dashed edges in the corresponding figure in trivial ears at the end of D.

Otherwise, if R is a short ear, then it is pendant. If it is a 2-ear (Figure 4.14b), we construct the ear S as shown by the thick path in the figure. Observe that we have two choices for one end of this ear: we choose to end the ear at either g or h, so as to ensure that it is an open ear. The example in the figure shows S ending at g, with edges gz, zx, xy, yv, vw and wb. We remove P', P'' and R from D, add S to D in place of P', and add all of the dashed edges in trivial ears at the end of D.

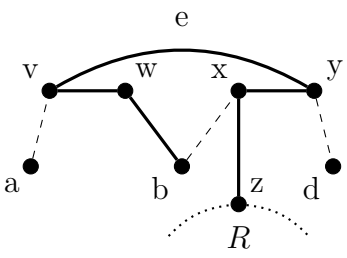
Otherwise, R is a 3-ear. Let g and h be the endpoints of R, and i and z be its internal vertices adjacent to g and h respectively (Figures 4.14c and 4.14d). We have two cases: either g and b are distinct, or they coincide. If they are distinct, we construct the ear S as shown by the thick path in Figure 4.14c, with edges bw, wv, vy, yx, xz, zi and ig. We delete P', P'' and R from D, and add S to D in place of P', and all of the dashed edges in trivial ears at the end of D.

Otherwise, the vertices g and b coincide. We construct the ear S as shown by the thick path in Figure 4.14d, with edges gi, iz, zx, xy and yd. We delete the ears P'' and R from D and add S to D in place of P'', and all of the dashed edges in trivial ears at the end of D.

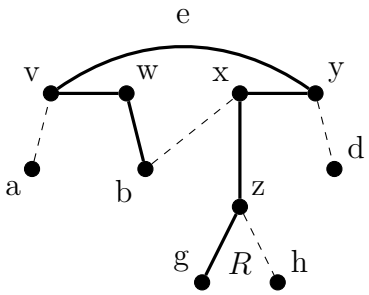
In all cases, the number of even ears remains equal to $\varphi(G)$, since the only case where S is an even ear is when R is a 2-ear. Additionally, S is an open pendant ear, hence the new ear-decomposition is valid for G.

Since the above procedure takes constant time for every pair of pendant ears with adjacent internal vertices, the running time for the whole procedure is polynomial.

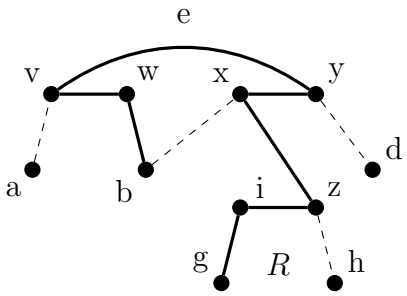
On termination of this procedure, the ear-decomposition D has $\varphi(G)$ even ears, and is both open and nice.



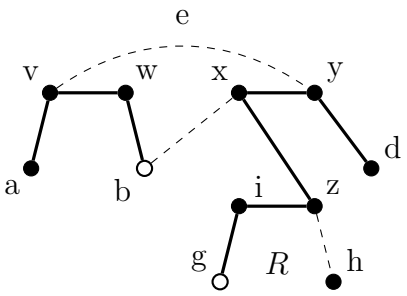
(a) R is a long ear



(b) R is a 2-ear



(c) R is a 3-ear, and b and g are distinct



(d) R is a 3-ear, and b and g coincide

Figure 4.14: x is adjacent to a vertex outside P'

4.4 Finishing Up

In this subsection, we show that the graph obtained by taking the non-trivial ears of the open nice ear-decomposition that we constructed has size within a factor of $\frac{17}{12}$ of the cost of an optimal solution.

Lemma 4.5. *Let D be an open nice ear-decomposition of a 2-vertex-connected graph G, and M be the associated eardrum of D. Denote by V_I the set of internal vertices of non-pendant ears, and let $\mu(G, M)$ be the size of the maximum earmuff for the eardrum M. Then $\mu(G, M) \leq |V_I| - 1$.*

Proof. Suppose not. Then $\mu(G, M) \geq |V_I|$. Consider the graph H on the vertex set V_I with edge (u, v) present in $E(H)$ if and only if there is a path with its endpoints at u and v in the maximum earmuff for M. Since $\mu(G, M) \geq |V_I|$, this graph has at least $|V_I|$ edges, and is hence not a forest. Since any cycle in this graph is a cycle in the maximum earmuff, we have a contradiction. Hence $\mu(G, M) \leq |V_I| - 1$. □

Theorem 4.6. *The above algorithm is a $\frac{17}{12}$-approximation algorithm for 2VCSS on graphs with minimum degree at least 3.*

We construct an open nice ear-decomposition D for G. Let π denote the number of pendant ears and π_3 the number of (pendant) 3-ears in this ear-decomposition. We have $\pi_3 \leq \pi$. Let H be the graph obtained by deleting from G all edges that are in trivial ears in this ear-decomposition. Since the nontrivial ears of D form an open ear-decomposition for H, H is 2-vertex-connected (see Lemma 2.1), and has at most $\frac{17}{12}$LP(G) edges, which we show using the following claims.

Claim 4.7. *The number of edges in nontrivial ears is at most $\frac{5}{4}LP(G) + \frac{1}{2}\pi$.*

Proof. For any ear P with $|E(P)| \geq 5$, we have $|E(P)| \leq \frac{5}{4}|in(P)|$. For any 4-ear or 2-ear P we have $|E(P)| \leq \frac{5}{4}|in(P)| + \frac{3}{4}$. For any 3-ear P we have $|E(P)| \leq \frac{5}{4}|in(P)| + \frac{1}{2}$.

Let E' be the set of edges in nontrivial ears. Since the total number of 4- and 2-ears in D is at most $\varphi(G)$, and $\pi_3 \leq \pi$, the total number of edges in nontrivial ears is at most $\frac{5}{4}(|V(G)| - 1) + \frac{3}{4}\varphi(G) + \frac{1}{2}\pi \leq \frac{5}{4}L_\varphi(G) + \frac{1}{2}\pi$, which is at most $\frac{5}{4}LP(G) + \frac{1}{2}\pi$. □

Claim 4.8. *The number of edges in nontrivial ears is at most $\frac{3}{2}LP(G) - \frac{1}{4}\pi$.*

Proof. Since D is an open nice ear-decomposition, the graph induced in G by the internal vertices V_M of the pendant short ears of D has degree at most 1. Let M be the set of its components, then M is an eardrum in G. Let V_L be the set of internal vertices of pendant long ears and let $V_I = V \setminus (V_M \cup V_L)$. Denote by φ_M, φ_L and φ_I the number of even ears in the sets of pendant short ears, pendant long ears and non-pendant ears respectively.

Let E_1 be the set of edges in pendant short ears. For every pendant short ear P, we have $|E(P)| = \frac{3}{2}|in(P)| + \frac{1}{2}\varphi(P)$. Summing over all pendant short ears, we have $|E_1| = \frac{3}{2}|V_M| + \frac{1}{2}\varphi_M$.

Let E_2 be the set of edges in pendant long ears. For every pendant long ear P, we have $|E(P)| \leq \frac{3}{2}|in(P)| + \frac{1}{2}\varphi(P) - 1$. Summing over all pendant long ears, we have $|E_2| \leq \frac{3}{2}|V_L| + \frac{1}{2}\varphi_D - (\pi - |M|)$.

Let E_3 be the set of edges in non-pendant ears. For every non-pendant ear P, since P is a long ear, we have $|E(P)| \leq \frac{5}{4}|in(P)| + \frac{1}{2}\varphi(P)$. For the starting vertex P_0, $|E(P_0)| = 0$ and $|in(P_0)| = 1$. Summing over all non-pendant ears and P_0, we have $|E_3| \leq \frac{5}{4}|V_I - 1| + \frac{1}{2}\varphi_I$.

Let $E' = E_1 \cup E_2 \cup E_3$ be the set of edges in nontrivial ears. Summing the above inequalities, we get

$$\begin{aligned}
|E'| &\leq \frac{3}{2}|V(G)| + \frac{1}{2}\varphi(G) - \pi + |M| - \frac{1}{4}|V_I| - \frac{5}{4} \\
&= [|V(G)| + |M| - \mu(G,M) - 1] \\
&\quad + \frac{1}{2}[|V(G)| + \varphi(G) - 1] \\
&\quad - \pi \\
&\quad + \left(\frac{1}{4} + \mu(G,M) - \frac{1}{4}|V_I|\right) \\
&= L_\mu(G,M) + \frac{1}{2}L_\varphi(G) - \pi + \left(\frac{1}{4} + \mu(G,M) - \frac{1}{4}|V_I|\right) \\
&\leq \frac{3}{2}LP(G) - \pi + \left(\frac{1}{4} + \mu(G,M) - \frac{1}{4}|V_I|\right) \\
&\leq \frac{3}{2}LP(G) - \pi + \frac{3}{4}\mu(G,M) \qquad \text{using Lemma 4.5} \\
&\leq \frac{3}{2}LP(G) - \pi + \frac{3}{4}\pi \qquad \text{since } \mu(G,M) \leq |M| \leq \pi \\
&\leq \frac{3}{2}LP(G) - \frac{1}{4}\pi.
\end{aligned}$$

□

If $\pi \leq \frac{1}{3}LP(G)$, then from Claim 1, $|E'| \leq \frac{5}{4}LP(G) + \frac{1}{2}\pi \leq \frac{17}{12}LP(G) \leq \frac{17}{12}OPT_{2VC}(G)$.

If $\pi > \frac{1}{3}LP(G)$, then from Claim 2, $|E'| \leq \frac{3}{2}LP(G) - \frac{1}{4}\pi < \frac{17}{12}LP(G) \leq \frac{17}{12}OPT_{2VC}(G)$.

Applying Theorems 4.3 and 4.4 to G, and deleting all edges in trivial ears, we obtain a 2-vertex-connected spanning subgraph of cardinality at most $\frac{17}{12}OPT_{2VC}(G)$ in polynomial time.

Concluding Remarks

We presented two approximation algorithms, one for a specialization of the 2ECSS problem and another for a specialization of the 2VCSS problem. We also gave some new structural results about the optimal solutions and known lower bounds for these problems.

Both problems have been studied for nearly three decades, and while progress on them has been slow over this time period, new approximation algorithms, albeit highly complicated ones, continue to appear.

The weighted versions of these problems continue to be difficult to approximate. Although several people have worked on them, there is no known ratio better than 2 even for very specialised choices of edge weights. Since we do not know of any examples that prevent a (2-ϵ)-approximation algorithm for the weighted 2-edge-connectivity problem for some $\epsilon > 0$, this seems to be the best related problem to attempt to solve at the moment. Another problem arising from this thesis is to characterize the graphs that have open and nice ear-decompositions. Any such characterization would immediately provide an approximation algorithm for this class of graphs. I hope (and expect) to see a lot of progress on these and other connectivity problems in the next few years.

References

[1] *39th Annual Symposium on Foundations of Computer Science, FOCS '98, November 8-11, 1998, Palo Alto, California, USA*. IEEE Computer Society, 1998.

[2] *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA*. ACM/SIAM, 2003.

[3] J. Cheriyan, J. Dippel, F. Grandoni, A. Khan, and V.V. Narayan. On the matching augmentation problem. *(Unpublished Manuscript)*.

[4] Joseph Cheriyan, András Sebö, and Zoltán Szigeti. Improving on the 1.5-approximation of a smallest 2-edge connected spanning subgraph. *SIAM J. Discrete Math.*, 14(2):170–180, 2001.

[5] Béla Csaba, Marek Karpinski, and Piotr Krysta. Approximability of dense and sparse instances of minimum 2-connectivity, TSP and path problems. In Eppstein [8], pages 74–83.

[6] R. Diestel. *Graph Theory*. Springer-Verlag Berlin Heidelberg, 2010.

[7] Jack Edmonds. Submodular functions, matroids, and certain polyhedra. In Jünger et al. [21], pages 11–26.

[8] David Eppstein, editor. *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA*. ACM/SIAM, 2002.

[9] Cristina G. Fernandes. A better approximation ratio for the minimum size k-edge-connected spanning subgraph problem. *J. Algorithms*, 28(1):105–124, 1998.

[10] András Frank. Conservative weightings and ear-decompositions of graphs. *Combinatorica*, 13(1):65–81, 1993.

[11] András Frank. *Connections in Combinatorial Optimization*. Oxford University Press, 2011.

[12] M. R. Garey, David S. Johnson, and Robert Endre Tarjan. The planar hamiltonian circuit problem is np-complete. *SIAM J. Comput.*, 5(4):704–714, 1976.

[13] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[14] Naveen Garg, Santosh Vempala, and Aman Singla. Improved approximation algorithms for biconnected subgraphs via better lower bounding techniques. In Ramachandran [26], pages 103–111.

[15] Prabhakar Gubbala and Balaji Raghavachari. Approximation algorithms for the minimum cardinality two-connected spanning subgraph problem. In Jünger and Kaibel [20], pages 422–436.

[16] K. Heeger and J. Vygen. Two-connected spanning subgraphs with at most $\frac{10}{7}$opt edges. *eprint arXiv:1609.00147*, 2016.

[17] Kamal Jain. Factor 2 approximation algorithm for the generalized steiner network problem. In *39th Annual Symposium on Foundations of Computer Science, FOCS '98, November 8-11, 1998, Palo Alto, California, USA* [1], pages 448–457.

[18] Klaus Jansen and Samir Khuller, editors. *Approximation Algorithms for Combinatorial Optimization, Third International Workshop, APPROX 2000, Saarbrücken, Germany, September 5-8, 2000, Proceedings*, volume 1913 of *Lecture Notes in Computer Science*. Springer, 2000.

[19] Raja Jothi, Balaji Raghavachari, and Subramanian Varadarajan. A 5/4-approximation algorithm for minimum 2-edge-connectivity. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA.* [2], pages 725–734.

[20] Michael Jünger and Volker Kaibel, editors. *Integer Programming and Combinatorial Optimization, 11th International IPCO Conference, Berlin, Germany, June 8-10, 2005, Proceedings*, volume 3509 of *Lecture Notes in Computer Science*. Springer, 2005.

[21] Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi, editors. *Combinatorial Optimization - Eureka, You Shrink!, Papers Dedicated to Jack Edmonds, 5th International Workshop, Aussois, France, March 5-9, 2001, Revised Papers*, volume 2570 of *Lecture Notes in Computer Science*. Springer, 2003.

[22] Samir Khuller and Uzi Vishkin. Biconnectivity approximations and graph carvings. *J. ACM*, 41(2):214–235, 1994.

[23] Guy Kortsarz and Zeev Nutov. Approximating node connectivity problems via set covers. *Algorithmica*, 37(2):75–92, 2003.

[24] László Lovász. A note on factor-critical graphs. *Studia Sci. Math. Hungar.*, 7:279–280, 1972.

[25] László Lovász and Michael D. Plummer. *Matching Theory*. North Holland, 1986.

[26] Vijaya Ramachandran, editor. *Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 25-27 January 1993, Austin, Texas*. ACM/SIAM, 1993.

[27] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24. Springer-Verlag Berlin Heidelberg, 2003.

[28] András Sebö and Jens Vygen. Shorter tours by nicer ears: 7/5-approximation for the graph-tsp, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs. *Combinatorica*, 34(5):597–629, 2014.

[29] Santosh Vempala and Adrian Vetta. Factor 4/3 approximations for minimum 2-connected subgraphs. In Jansen and Khuller [18], pages 262–273.

[30] H. Whitney. Non-separable and planar graphs. *Transactions of the American Mathematical Society*, 34:339–362, 1932.

Appendix

Lemma A.1. *The 2-vertex-connectivity problem is NP-hard when restricted to instances with minimum degree at least 3.*

Proof. Let G be the input graph of an instance of the general unweighted 2-vertex- connectivity problem. Denote by $n(G)$ the number of vertices with degree 2 in G.

Consider the graph G' constructed in the following manner. We replace every vertex with degree 2 in G by an instance of K_4 (the complete graph on 4 vertices), such that the two edges incident on the degree-2 vertex in G are incident on two distinct vertices of the K_4 instance in G'. Then G' has minimum degree at least 3, and every 2-vertex-connected spanning subgraph H of G, with $|E(H)|$ edges, corresponds to a 2-vertex-connected spanning subgraph H' of G' (constructed by adding a path of length 3 between the degree-4 nodes of every K_4 instance created by replacement), with $|E(H')| = |E(H)|+3n(G)$ edges, and vice-versa. Further, any optimal solution to this problem in G' uses at least 3 edges of each K_4 instance.

Consequently, any algorithm that solves the 2-vertex-connectivity problem in polynomial time on graphs with minimum degree at least 3 can be used to solve the unrestricted problem in polynomial time; hence this restricted problem is NP-hard. □