The Matching Augmentation Problem: A $\frac{7}{4}$-Approximation Algorithm

J.Cheriyan [*] J.Dippel [†] F.Grandoni [‡] A.Khan [§] V.V.Narayan [¶]

December 28, 2017

Abstract

We present a $\frac{7}{4}$ approximation algorithm for the matching augmentation problem (MAP): given a multi-graph with edges of cost either zero or one such that the edges of cost zero form a matching, find a 2-edge connected spanning subgraph (2-ECSS) of minimum cost.

We first present a reduction of any given MAP instance to a collection of well-structured MAP instances such that the approximation guarantee is preserved. Then we present a $\frac{7}{4}$ approximation algorithm for a well-structured MAP instance. The algorithm starts with a min-cost 2-edge cover and then applies ear-augmentation steps. We analyze the cost of the ear-augmentations using an approach similar to the one proposed by Vempala & Vetta for the (unweighted) min-size 2-ECSS problem ("Factor 4/3 approximations for minimum 2-connected subgraphs," APPROX 2000, LNCS 1913, pp.262–273).

Keywords: 2-edge-connected graph, 2-edge covers, approximation algorithms, bridges, connectivity augmentation, forest augmentation problem, matching augmentation problem, network design.

1 Introduction

A basic goal in the area of *survivable network design* is to design real-world networks of low cost that provide connectivity between pre-specified pairs of nodes even after the failure of a few edges/nodes. Many of the problems in this area are NP-hard, and significant efforts have been devoted in the last few decades to the design of approximation algorithms.

One of the fundamental problems in the area is the minimum-cost 2-edge connected spanning subgraph problem (abbreviated as min-cost 2-ECSS): given a graph together with non-negative costs for the edges, find a 2-edge connected spanning subgraph (abbreviated as 2-ECSS) of minimum cost. This problem is closely related to the famous Traveling Salesman Problem (TSP), and some of the earliest papers in the area of approximation algorithms address the min-cost 2-ECSS problem [4, 5]. In the context of approximation algorithms, this research led to the discovery of algorithmic paradigms such as the *primal-dual method* [8, 19] and the *iterative rounding method* [10, 13], and led to dozens of publications. Under appropriate assumptions, these methods achieve an approximation guarantee of 2 for several key problems in survivable network design, including min-cost 2-ECSS. Unfortunately, these generic methods do not achieve approximation guarantees

[*]University of Waterloo, Waterloo, Canada
[†]University of Waterloo, Waterloo, Canada
[‡]IDSIA, USI-SUPSI, Lugano, Switzerland
[§]Technical University of Munich, Garching, Germany
[¶]McGill University, Montreal, Canada

below 2. Significant research efforts have been devoted to achieving approximation guarantees below 2 for specific problems in the area of survivable network design. For example, building on earlier work, an approximation guarantee of $\frac{4}{3}$ has been achieved for unweighted (*min-size*) 2-ECSS [16], where each edge of the input graph has cost one and the goal is to find a 2-ECSS with the minimum number of edges.

There is an important obstacle beyond unweighted problems, namely, the special case of min-cost 2-ECSS where the (input) edges have cost of zero or one, and the aim is to design an algorithm that achieves an approximation guarantee below 2. This problem is called the *Forest Augmentation Problem* (FAP). In more detail, we are given an undirected graph $G = (V,\ E_0 \cup E_1)$, where each edge in E_0 has cost zero and each edge in E_1 has cost one; the goal is to compute a 2-ECSS $H = (V, F)$ of minimum cost. We denote the cost of an edge e of G by $\text{cost}(e)$, and for a subgraph G' of G, $\text{cost}(G')$ denotes $\sum_{e \in E(G')} \text{cost}(e)$. Observe that $\text{cost}(H) = |F \cap E_1|$, so the goal is to augment E_0 to a 2-ECSS by adding the minimum number of edges from E_1. Intuitively, the zero-edges define some existing network that we wish to augment (with edges of cost one) such that the augmented network is resilient to the failure of any one edge. W.l.o.g. we may contract each of the 2-edge connected subgraphs formed by the zero-edges, and hence, we may assume that E_0 induces a forest: this motivates the name of the problem.

A key special case of FAP is the *Tree Augmentation Problem* (TAP), where the edges of cost zero form a spanning tree. Nagamochi [9] first obtained an approximation guarantee below 2 for TAP, and since then there have been several advances including recent work, see [1, 3, 12, 7].

We focus on a different special case of FAP called the *matching augmentation problem* (MAP): given a multi-graph with edges of cost either zero or one such that the edges of cost zero form a matching, find a 2-ECSS of minimum cost. Note that loops are not allowed; multi-edges (parallel edges) are allowed. From the view-point of approximation algorithms, MAP is "orthogonal" to TAP in the sense that the forest of zero-cost edges has many connected components in MAP, whereas this forest has only one connected component in TAP. In our opinion, MAP (like TAP) is an important special case of FAP in the sense that none of the previous approaches (including approaches developed for TAP over two decades) give an approximation guarantee below 2 for MAP.

1.1 Previous literature & possible approaches for MAP

There is extensive literature on approximation algorithms for problems in survivable network design, and also on the minimum-cost 2-ECSS problem including its key special cases (including the unweighted problem, TAP, etc.). To the best of our knowledge, there is no previous publication on FAP or MAP, although the former is well known to the researchers working in this area.

Let us explain briefly why previous approaches do not help for obtaining an approximation guarantee below 2 for MAP. Let G denote the input graph, and let n denote $|V(G)|$. Let *opt* denote the optimal value, i.e., the minimum cost of a 2-ECSS of the given instance. Recall that the standard cut-covering LP relaxation of the min-cost 2-ECSS problem has a non-negative variable x_e for each edge e of G, and for each nonempty set of nodes S, $S \neq V$, there is a constraint requiring the sum of the x-values in the cut $(S, V - S)$ to be ≥ 2; the objective function is to minimize $\sum_{e \in E} \text{cost}(e) x_e$.

The primal-dual method and the iterative rounding method are powerful and versatile methods for rounding LP relaxations, but in the context of FAP, these methods seem to be limited to proving approximation guarantees of 2 (or more).

Several intricate combinatorial methods that may also exploit lower-bounds from LP relaxations have been developed for approximation algorithms for unweighted 2-ECSS, e.g., the $\frac{4}{3}$-approximation algorithm of [16]. For unweighted 2-ECSS, there is a key lower bound of n on *opt*

(since any solution must have $\geq n$ edges, each of cost one). This no longer holds for MAP; indeed, the analogous lower bound on opt is $\frac{1}{2}n$ for MAP. It can be seen that an α-approximation algorithm for unweighted 2-ECSS implies a $(3\alpha-2)$-approximation algorithm for MAP. (We sketch the reduction: let M denote the set of zero-cost edges in an instance of MAP; observe that $|M| \leq opt$; we subdivide (once) each edge in M, then we change all edge costs to one, then we apply the algorithm for unweighted 2-ECSS, and finally we undo the initial transformation; the optimal cost of the unweighted 2-ECSS instance is $\leq opt + 2|M|$, hence, the solution of the MAP instance has cost $\leq \alpha(opt + 2|M|) - 2|M| = \alpha opt + (2\alpha - 2)|M| \leq (3\alpha - 2)opt$.) Thus the $\frac{4}{3}$-approximation algorithm of [16] for unweighted 2-ECSS gives a 2-approximation algorithm for MAP. (Although preliminary results and claims have been published on achieving approximation guarantees below $\frac{4}{3}$ for unweighted 2-ECSS, there are no refereed publications to date, see [16].)

Over the last two decades, starting with the work of [9], a few methods have been developed to obtain approximation guarantees below 2 for TAP. The recent methods of [1, 3] rely on so-called bundle constraints defined by paths of zero-cost edges. Unfortunately, these methods (including methods that use the bundle constraints) rely on the fact that the set of zero-cost edges forms a connected graph that spans all the nodes, see [9, 12, 3]. Clearly, this property does not hold for MAP.

1.2 Hardness of approximation of MAP and FAP

MAP is a generalization of the unweighted 2-ECSS problem (consider the special case of MAP with $M = \emptyset$). The latter problem is known to be APX-hard; thus, it has a "hardness of approximation" threshold of $1 + \epsilon$ where $\epsilon > 0$ is a constant, see [6]. Hence, MAP is APX-hard.

Given the lack of progress on approximation algorithms for FAP, one may wonder whether there is a "hardness of approximation" threshold that would explain the lack of progress. Unfortunately, the results and techniques from the area of "hardness of approximation" are far from the known approximation guarantees for many problems in network design. For example, even for the notorious Asymmetric TSP (ATSP), the best "hardness of approximation" lower bound known is around $\frac{75}{74} \approx 1.014$, see [11].

1.3 Our method for MAP

We first present a reduction of any given instance of MAP to a collection of well-structured MAP instances such that the approximation guarantee is preserved, see Sections 2, 3, 4. Then we present a $\frac{7}{4}$ approximation algorithm for a well-structured MAP instance, see Sections 3, 5, 6. Our algorithm starts with a so-called D2 (this is a min-cost 2-edge cover) and then applies ear-augmentation steps. We analyze the cost of the ear-augmentations using an approach similar to the one proposed by Vempala & Vetta for the unweighted 2-ECSS problem [17]. Our presentation is self-contained and formally independent of Vempala & Vetta's manuscript; also, we address a weighted version of the 2-ECSS problem and our challenge is to improve on the approximation guarantee of 2, whereas Vempala & Vetta's goal is to achieve an approximation guarantee of $\frac{4}{3}$ for the unweighted 2-ECSS problem.

An outline of the paper follows. Section 2 has standard definitions and some preliminary results. Section 3 presents an outline of our algorithm for MAP, and explains what is meant by a well-structured MAP instance. Section 4 presents the pre-processing steps that give an approximation preserving reduction from any instance of MAP to a collection of well-structured MAP instances; some readers may prefer to skip this section (and refer back to the results/details as needed). Sections 5, 6 present the $\frac{7}{4}$ approximation algorithm for well-structured MAP instances, and prove the approximation guarantee. Section 7 presents examples that give lower bounds on our results

on MAP. The first example gives a construction such that $opt \approx \frac{7}{4}\tau$, where τ denotes the minimum cost of a 2-edge cover. The second example gives a construction such that the cost of the solution computed by our algorithm is $\approx \frac{7}{4}opt$.

2 Preliminaries

This section has definitions and preliminary results. Our notation and terms are consistent with [2], and readers are referred to that text for further information.

Let $G = (V, E)$ be a (loop-free) multi-graph with edges of cost either zero or one such that the edges of cost zero form a matching. We take G to be the input graph, and we use n to denote $|V(G)|$. Let M denote the set of edges of cost zero. We call an edge of M a *zero-edge* and we call an edge of $E - M$ a *unit-edge*. We call a pair of parallel edges a $\{0, 1\}$-edge-pair if one of the two edges of the pair has cost zero and the other one has cost one.

We use the standard notion of contraction of an edge, see [15, p.25]: Given a multi-graph H and an edge $e = vw$, the contraction of e results in the multi-graph $H/(vw)$ obtained from H by deleting e and its parallel copies and identifying the nodes v and w. (Thus every edge of H except for vw and its parallel copies is present in $H/(vw)$; we disallow loops in $H/(vw)$.)

For a graph H and a set of nodes $S \subseteq V(H)$, $\delta_H(S)$ denotes the set of edges that have one end node in S and one end node in $V(H) - S$; moreover, $H - S$ denotes $H[V(H) - S]$, the subgraph of H induced by $V(H) - S$. For a graph H and a set of edges $F \subseteq E(H)$, $H - F$ denotes the graph $(V(H),\ E(H) - F)$. We use relaxed notation for singleton sets, e.g., we use $\delta_H(v)$ instead of $\delta_H(\{v\})$, and we use $H - v$ instead of $H - \{v\}$.

We denote the cost of an edge e of G by $\text{cost}(e)$. For a set of edges $F \subseteq E(G)$, $\text{cost}(F) := \sum_{e \in F} \text{cost}(e)$, and for a subgraph G' of G, $\text{cost}(G') := \sum_{e \in E(G')} \text{cost}(e)$.

2.1 2EC, 2NC, bridges and D2

A multi-graph H is called k-edge connected if $|V(H)| \geq 2$ and for every $F \subseteq E(H)$ of size $< k$, $H - F$ is connected. Thus, H is 2-edge connected if it has ≥ 2 nodes and the deletion of any one edge results in a connected graph. A multi-graph H is called k-node connected if $|V(H)| > k$ and for every $S \subseteq V(H)$ of size $< k$, $H - S$ is connected. We use the abbreviations *2EC* for "2-edge connected," and *2NC* for "2-node connected."

We assume w.l.o.g. that the input G is 2-edge connected. Moreover, we assume w.l.o.g. that there are ≤ 2 copies of each edge (in any multi-graph that we consider); this is justified since an edge-minimal 2-ECSS cannot have three or more copies of any edge (see Proposition 1 below).

For any 2EC graph H, let $OPT(H)$ denote a min-cost 2-ECSS of H, and let $opt(H)$ denote the minimum cost of a 2-ECSS of H. When there is no danger of ambiguity, we use OPT rather than $OPT(H)$, and we use opt rather than $opt(H)$.

By a *bridge* we mean a cut edge, i.e., an edge of a connected (sub)graph whose removal results in two connected components, and by a *cut node* we mean a node of a connected (sub)graph whose deletion results in two or more connected components. We call a bridge of cost zero a *zero-bridge* and we call a bridge of cost one a *unit-bridge*.

By a *2ec-block* we mean a maximal connected subgraph with two or more nodes that has no bridges. Thus, a 2ec-block could be a connected component that has no bridges, or it could be a bridgeless subgraph B of a connected component C such that there is a bridge (of C) incident to B. (Observe that the 2ec-blocks of a graph H consist of the connected components with ≥ 2 nodes of the graph obtained from H by deleting all bridges.) We call a 2ec-block *pendant* if it is incident to exactly one bridge.

The next result characterizes edges that are not essential for 2-edge connectivity.

Proposition 1. *Let H be a 2EC graph and let $e = vw$ be an edge of H. If $H - e$ has two edge-disjoint v, w paths, then $H - e$ is 2EC.*

By a *2-edge cover* (of G) we mean a set of edges F of G such that each node v is incident to at least two edges of F (i.e., $F \subseteq E(G) : |\delta_F(v)| \geq 2, \forall v \in V(G)$). By D2($G$) we mean any minimum-cost 2-edge cover of G (G may have several minimum-cost 2-edge covers, and D2(G) may refer to any one of them); we use $\tau(G)$ to denote the cost of D2(G); when there is no danger of ambiguity, we use D2 rather than D2(G), and we use τ rather than $\tau(G)$. Note that D2 may have several connected components, and each may have one or more bridges; moreover, if a connected component of D2 has a bridge, then it has two or more pendant 2ec-blocks.

The next result follows from Theorem 34.15 in [15, Chapter 34].

Proposition 2. *There is a polynomial-time algorithm for computing* D2.

The next result states the key lower bound used by our approximation algorithm.

Fact 3. *Let H be any 2EC graph. Then we have $opt(H) \geq \tau(H)$.*

By a *bridgeless 2-edge cover* (of G) we mean a 2-edge cover (of G) that has no bridges; note that we have no requirements on the cost of a bridgeless 2-edge cover. We mention that the problem of computing a bridgeless 2-edge cover of minimum cost is NP-hard (there is a reduction from TAP), and there is no approximation algorithm known for the case of nonnegative costs.

2.2 Redundant 4-cycles

By a *redundant 4-cycle* we mean a cycle C consisting of four nodes and four edges of G such that $V(C) \neq V(G)$, two of the (non-adjacent) edges of C have cost zero, and two of the nonadjacent nodes of C have degree two in G. For example, a 4-cycle $C = u_1, u_2, u_3, u_4, u_1$ with zero-edges u_1u_2, u_3u_4 and unit-edges u_2u_3, u_4u_1 of a 2EC graph G is a redundant 4-cycle provided all edges between $V(C)$ and $V(G) - V(C)$ are incident to either u_1 or u_3.

Observe that every 2-edge cover of G must contain the edges of every redundant 4-cycle. Also, observe that two different redundant 4-cycles are disjoint (i.e., any node is contained in at most one redundant 4-cycle).

2.3 Ear decompositions

An *ear decomposition* of a graph is a partition of the edge set into paths or cycles, $P_0, P_1, \ldots, P_k$, such that P_0 is the trivial path with one node, and each P_i ($1 \leq i \leq k$) is either (1) a path that has both end nodes in $V_{i-1} = V(P_0) \cup V(P_1) \cup \ldots \cup V(P_{i-1})$ but has no internal nodes in V_{i-1}, or (2) a cycle that has exactly one node in V_{i-1}. Each of $P_1, \ldots, P_k$ is called an *ear*; note that P_0 is not regarded as an ear. We call $P_i, i \in \{1, \ldots, k\}$, an *open ear* if it is a path, and we call it a *closed ear* if it is a cycle. An *open* ear decomposition $P_0, P_1, \ldots, P_k$ is one such that all the ears $P_2, \ldots, P_k$ are open. (The ear P_1 is always closed.)

Proposition 4 (Whitney [18])**.** *(i) A graph is 2EC iff it has an ear decomposition.*

(ii) A graph is 2NC iff it has an open ear decomposition.

2.4 Bad-pairs and bp-components

By a *bad-pair* we mean a pair of nodes $\{v, w\}$ of the input graph G such that the edge vw is present and has zero cost, and moreover, the deletion of both nodes v and w results in a disconnected graph.

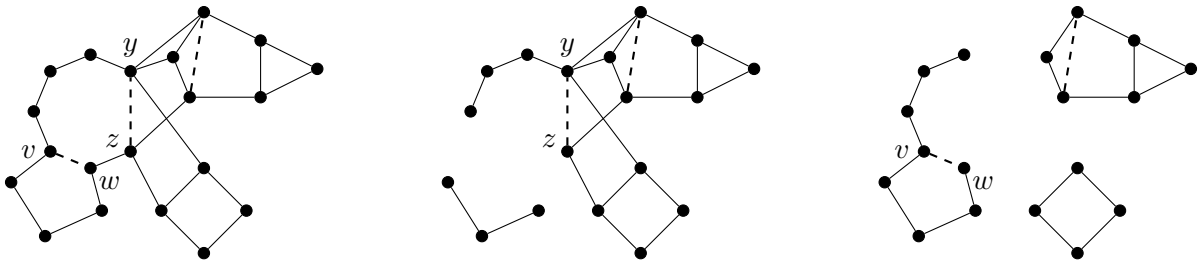
By a *bp-component* we mean one of the connected components resulting from the deletion of a bad-pair from G; see Figure 1. The set of all bp-components (of all bad-pairs) is a cross-free family, see [15, Chap. 13.4]. Consider two bad-pairs $\{v, w\}$ and $\{y, z\}$. One of the bp-components of $\{v, w\}$ contains $\{y, z\}$; call it C_1. Then it can be seen that all-but-one of the bp-components of $\{y, z\}$ are contained in C_1; the one remaining bp-component of $\{y, z\}$, call it C'_1, contains $\{v, w\}$ and all of the bp-components of $\{v, w\}$ except C_1. Thus, the union of the two bp-components C_1 and C'_1 contains $V(G)$.

The following fact is analogous to the fact that every tree on ≥ 2 nodes contains a node v such that all-but-one of the neighbors of v are leaves. (To see this, consider a longest path P of a tree, and take v to be the second node of P.)

Fact 5. *Suppose that G has at least one bad-pair. Then there exists a bad-pair such that all-but-one of its bp-components are free of bad-pairs.*

Remark: Let us sketch an algorithmic proof of Fact 5. For any subgraph G' of G, let $\#\mathrm{bp}(G')$ denote the number of bad-pairs of G'. For any bad-pair $\{v, w\}$ and its bp-components $C_1, \ldots, C_k$, let us assume that the indexing is in non-increasing order of $\#\mathrm{bp}(C_i)$, thus, we have $\#\mathrm{bp}(C_1) \geq \#\mathrm{bp}(C_2) \geq \cdots \geq \#\mathrm{bp}(C_k)$.

We start by computing all the bad-pairs of G. Then we choose any bad-pair $\{v_1, w_1\}$ and compute its bp-components $C_1^{(1)}, \ldots, C_{k_1}^{(1)}$. If $\#\mathrm{bp}(C_2^{(1)}) = 0$, then we are done; our algorithm outputs $\{v_1, w_1\}$. Otherwise, we pick $C_2^{(1)}$ and choose any bad pair $\{v_2, w_2\}$ contained in $C_2^{(1)}$. We compute the bp-components $C_1^{(2)}, \ldots, C_{k_2}^{(2)}$ of $\{v_2, w_2\}$ (in G). As stated above, $C_1^{(2)}$ contains $C_1^{(1)}$ as well as $\{v_1, w_1\}$; moreover, $\#\mathrm{bp}(C_2^{(1)}) > \#\mathrm{bp}(C_2^{(2)}) + \cdots + \#\mathrm{bp}(C_{k_2}^{(2)})$, since the bad-pair $\{v_2, w_2\}$ is in $C_2^{(1)}$ but it is not in any of its own bp-components. We iterate these steps (i.e., if $\#\mathrm{bp}(C_2^{(2)}) \neq 0$, then we pick any bad-pair $\{v_3, w_3\}$ of $C_2^{(2)}$, ...), until we find a bad-pair $\{v_\ell, w_\ell\}$ such that $\#\mathrm{bp}(C_2^{(\ell)}) = 0$; then we are done. Clearly, this algorithm is correct, and it terminates in $O(n)$ iterations.



(a) A graph with bad-pairs (indicated by dashed lines).

(b) bp-components for the bad-pair $\{v, w\}$.

(c) bp-components for the bad-pair $\{y, z\}$.

Figure 1: Illustration of bad-pairs and bp-components.

3 Outline of the algorithm

This section has an outline of our algorithm. We start by defining a well structured MAP instance.

Definition 1. *An instance of MAP is called* well-structured *if it has*

- *no* $\{0,1\}$*-edge-pairs,*
- *no redundant 4-cycles,*
- *no cut nodes, and*
- *no bad-pairs.*

Section 4 explains how to "decompose" any instance of MAP G into a collection of well-structured MAP instances $G_1, \ldots, G_k$ such that a 2-ECSS H of G can be obtained by computing 2-ECSSes $H_1, \ldots, H_k$ of $G_1, \ldots, G_k$, and moreover, the approximation guarantee is preserved, i.e., the approximation guarantee on G is $\leq$ the maximum of the approximation guarantees on $G_1, \ldots, G_k$ (in other words, $\frac{\text{cost}(H)}{opt(G)} \leq \max_{i=1,\ldots,k} \left\{ \frac{\text{cost}(H_i)}{opt(G_i)} \right\}$).

Algorithm (outline):

(0) apply the pre-processing steps (reductions) from Section 4 to obtain a collection of well-structured MAP instances $G_1, \ldots, G_k$;

for each G_i $(i = 1, \ldots, k)$, apply steps (1),(2),(3):

(1) compute D2(G_i) in polynomial time (w.l.o.g. assume D2(G_i) contains all zero-edges of G_i);

(2) then apply "bridge covering" from Section 5 to D2(G_i) to obtain a bridgeless 2-edge cover $\widetilde{H}_i$ of G_i;

(3) then apply the "gluing step" from Section 6 to $\widetilde{H}_i$ to obtain a 2-ECSS H_i of G_i;

(4) finally, output a 2-ECSS H of G from the union of $H_1, \ldots, H_k$ by undoing the transformations applied in step (0).

The pre-processing of step (0) consists of four reductions:

(pp1) handle $\{0,1\}$-edge-pairs,

(pp2) handle redundant 4-cycles,

(pp3) handle cut-nodes, and

(pp4) handle bad-pairs;

the first three reductions are discussed in Sections 4.1–4.3, and the last one is discussed in Section 4.4. Step (0) applies (pp1) to obtain a collection of MAP instances; after that, there is no further need to apply (pp1). Then, we iterate: while the collection of MAP instances has one or more of the latter three "obstructions" (redundant 4-cycles, cut nodes, bad-pairs), we apply (pp2), (pp3), (pp4) in sequence. After $\leq n$ iterations, we have a collection of well-structured MAP instances G_i. Then, we compute a near-optimal 2-ECSS H_i for each G_i using the algorithm of Theorem 6 (below), and finally, we use the H_i subgraphs to construct a 2-ECSS of G.

Our $\frac{7}{4}$ approximation algorithm for MAP follows from the following key theorem, and the fact that the algorithm runs in polynomial time.

Theorem 6. *Given a well-structured instance of MAP G', there is a polynomial-time algorithm that obtains a 2-ECSS H' from* D2(G') *(by adding edges and deleting edges) such that* $\text{cost}(H') \leq \frac{7}{4}\tau(G')$.

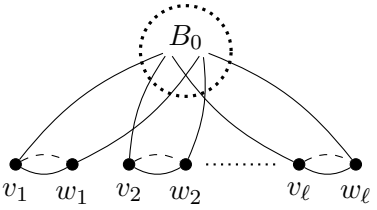
We use a credit scheme to prove this theorem; the details are presented in Sections 5 and 6. The algorithm starts with D2 as the current subgraph; we start by assigning $\frac{7}{4}$ initial credits to each unit-edge of D2; each such edge keeps one credit to pay for itself and the other $\frac{3}{4}$ credits are taken to be working credits available to the algorithm; the algorithm uses these working credits to pay for the augmenting edges "bought" in steps (2) or (3) (see the outline); also, the algorithm may "sell" unit-edges of the current subgraph (i.e., such an edge is permanently discarded and is not contained in the 2-ECSS output by the algorithm) and this supplies working credits to the algorithm (see Sections 5, 6).

In an ear-augmentation step, we may add either a single ear or a double ear (i.e., a pair of ears); see Section 5 (double ears may be added in Case 3, page 22) and Section 6 (double ears may be added in Case 2, page 26). Although this is not directly relevant, we mention that double ear augmentations are essential in matching theory, see [14, Ch.5.4]. As discussed above, in some of the ear-augmentation steps, we may (permanently) delete some edges from the current subgraph; see Section 5 (edges are deleted in double ear augmentations in Case 3, page 22) and Section 6 (edges are deleted in both Cases 1, 2).

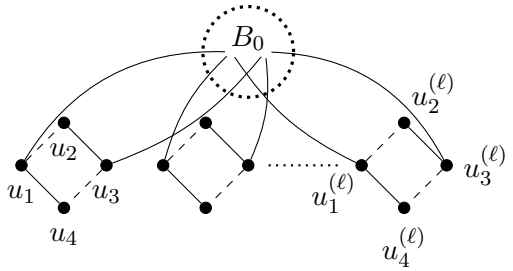
Remark: The following examples show that when we relax the definition of a well-structured MAP instance, then the inequality in Theorem 6 could fail to hold. See Figure 2 for illustrations.
(1) $\{0,1\}$-edge-pairs (i.e., parallel edges of cost zero and one) are present. Then $\frac{opt}{\tau} \approx 2$ is possible. Our construction consists of a root 2ec-block B_0, say a 6-cycle of cost 6, and $\ell \gg 1$ copies of the following gadget that are attached to B_0. The gadget consists of a pair of nodes v, w and two incident edges: a copy of edge vw of cost zero, and a copy of edge vw of cost one. Moreover, we have an edge between v and B_0 of cost one, and an edge between w and B_0 of cost one. Observe that a (feasible) 2-edge cover of this instance consists of B_0 and the two parallel edges (i.e., the two copies of the edge vw) of each copy of the gadget, and it has cost $6 + \ell$. Observe that any 2-ECSS contains the two edges between $\{v, w\}$ and B_0. Thus, $opt \geq 2\ell$, whereas $\tau \leq 6 + \ell$.
(2) Redundant 4-cycles are present. Then $\frac{opt}{\tau} \approx 2$ is possible. Our construction consists of a root 2ec-block B_0, say a 6-cycle of cost 6, and $\ell \gg 1$ copies of the following gadget that are attached to B_0. The gadget consists of a 4-cycle $C = u_1, \ldots, u_4, u_1$ that has two zero-edges u_1u_2, u_3u_4 and two unit-edges u_2u_3, u_4u_1; moreover, we have an edge between u_1 and B_0 of cost one, and an edge between u_3 and B_0 of cost one. Observe that a (feasible) 2-edge cover of this instance consists of B_0 and the 4-cycle C of each copy of the gadget, and it has cost $6 + 2\ell$. Observe that for any 2-ECSS and for each copy of the gadget, the two edges between C and B_0 as well as the four edges of C are contained in the 2-ECSS. Thus, $opt \geq 4\ell$, whereas $\tau \leq 6 + 2\ell$.
(3) Cut nodes are present. Then $\frac{opt}{\tau} \approx 2$ is possible. Our construction consists of ℓ copies of a 3-cycle $C = u_1, u_2, u_3, u_1$ where u_1u_2 is a zero edge and the other two edges have cost one. We "string up" the ℓ copies, i.e., node u_3 of the ith copy is identified with node u_1 of the $(i+1)$th copy. The optimal solution has all the edges, so $opt = 2\ell$, whereas a (feasible) 2-edge cover consists of a Hamiltonian path together with two more edges incident to the two ends of this path, and it has cost $\ell + 2$; thus $\tau \leq 2 + \ell$.
(4) Bad-pairs are present. Then $\frac{opt}{\tau} \approx 2$ is possible. An example can be obtained by modifying example (2) above.

4 Pre-processing

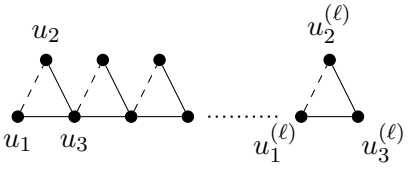
This section presents the four reductions used in the pre-processing step of our algorithm, namely, the handling of the $\{0,1\}$-edge-pairs, the redundant 4-cycles, the cut nodes, and the bad-pairs.



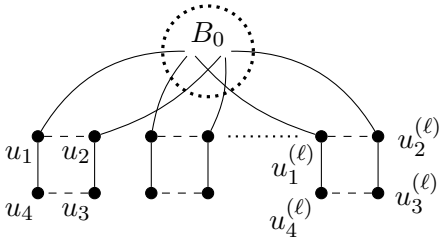
(a) Example (1) of Remark.



(b) Example (2) of Remark.



(c) Example (3) of Remark.



(d) Example (4) of Remark.

Figure 2: MAP instances G that are *not* well-structured such that $\frac{opt(G)}{\tau(G)} \approx 2$. Edges of cost zero and one are illustrated by dashed and solid lines, respectively.

4.1 Handling $\{0,1\}$-edge-pairs

We apply the following pre-processing step to eliminate all $\{0,1\}$-edge-pairs. We start with a simple result.

Fact 7. *Let H be an (inclusion-wise) edge-minimal 2EC graph, and let e_1, e_2 be a pair of parallel edges. Then $H - \{e_1, e_2\}$ has precisely two connected components, and each of these connected components is 2EC.*

Proof. Let v and w be the end nodes of e_1, e_2. By Proposition 1, H does not have three edge-disjoint v, w paths, hence, $H - \{e_1, e_2\}$ is disconnected. It follows that $H - \{e_1, e_2\}$ has precisely two connected components (deleting one edge from a connected graph results in a graph with ≤ 2 connected components). Let C_1, C_2 be the two connected components of $H - \{e_1, e_2\}$; clearly, each of C_1, C_2 contains precisely one of the nodes v, w. Suppose that C_1 is not 2EC; then it has a bridge f. Since the parallel edges e_1, e_2 have exactly one end node in C_1, f stays a bridge of $C_1 \cup C_2 \cup \{e_1, e_2\}$. This is a contradiction, since $H = C_1 \cup C_2 \cup \{e_1, e_2\}$ is 2EC. □

Let H be any 2EC graph. We call a $\{0,1\}$-edge-pair *essential* if its deletion results in a disconnected graph. When we delete an essential $\{0,1\}$-edge-pair then we get two connected components and each one is 2EC, by arguing as in the proof of Fact 7. Hence, when we delete all the essential $\{0,1\}$-edge-pairs, then we get a number of connected components $C_1, \ldots, C_k$ such that each one is 2EC. Clearly, an approximately optimal 2-ECSS of H can be computed by returning the union of approximately optimal 2-ECSSes of $C_1, \ldots, C_k$ together with all the essential $\{0,1\}$-edge-pairs of H; moreover, it can be seen that the approximation guarantee is preserved, that is, the approximation guarantee on H is $\leq$ the maximum of the approximation guarantees on $C_1, \ldots, C_k$.

The next observation allows us to handle the inessential $\{0,1\}$-edge-pairs.

Fact 8. *Suppose that H is 2EC and it has no essential $\{0,1\}$-edge-pairs. Then there exists a min-cost 2-ECSS of H that does not contain any unit-edge of any $\{0,1\}$-edge-pair.*

Proof. Consider a min-cost 2-ECSS H' of H that contains the minimum number of $\{0,1\}$-edge-pairs, i.e., among all the optimal subgraphs, we pick one that has the fewest number of parallel edges e, f such that $\text{cost}(e) + \text{cost}(f) = 1$. If H' has no $\{0,1\}$-edge-pair, then the fact holds. Otherwise, we argue by contradiction. We pick any $\{0,1\}$-edge-pair e, f. Deleting both e, f from H' results in two connected components C_1, C_2 such that each is 2EC, by Fact 7. Now, observe that e, f is not essential for H, hence, $H - \{e, f\}$ has an edge e'' between C_1 and C_2. We obtain the graph H'' from H' by replacing the unit-edge of e, f by the edge e''. Clearly, H'' is a 2-ECSS of H of cost $\leq \text{cost}(H')$, and moreover, it has fewer $\{0,1\}$-edge-pairs than H'. Thus, we have a contradiction. □

Now, focus on the input graph G. By the discussion above, we may assume that G has no essential $\{0,1\}$-edge-pairs. Then we delete the unit-edge of each $\{0,1\}$-edge-pair. By Fact 8, the resulting graph stays 2EC and the optimal value is preserved. Thus, we can eliminate all $\{0,1\}$-edge-pairs, while preserving the approximation guarantee.

Proposition 9. *Assume that G has no essential $\{0,1\}$-edge-pairs. Let $\hat{G}$ be the multi-graph obtained from G by eliminating all $\{0,1\}$-edge-pairs (as discussed above). Then $opt(G) = opt(\hat{G})$.*

Moreover, an α-approximation guarantee for $\hat{G}$ implies the same approximation guarantee for G.

In what follows, we continue to use G to denote the multi-graph obtained by eliminating all $\{0,1\}$-edge-pairs (for the sake of notational convenience).

The next fact states that the restriction on the zero-edges of G is preserved when we contract a set of edges E' such that none of the end nodes of the zero-edges in $E - E'$ is incident to an edge of E' (i.e., the end nodes of the zero-edges of G/E' are "original nodes" rather than "contracted nodes.")

Fact 10. *Let $G = (V, E)$ satisfy the restriction on the zero-edges (i.e., G has no $\{0,1\}$-edge-pairs, and the zero-edges form a matching). Suppose that we contract a set of edges $E' \subset E$ such that there exists no node that is incident to both an edge in E' and a zero-edge in $E - E'$. Then the restriction on the zero-edges continues to hold for the contracted multi-graph G/E'.*

4.2 Handling redundant 4-cycles

We contract all of the redundant 4-cycles in a pre-processing step. Recall that two distinct redundant 4-cycles have no nodes and no edges in common. We first compute all the redundant 4-cycles and then we contract each of these to a single node (i.e., we contract all four edges of each redundant 4-cycle).

Proposition 11. *Suppose that G has q redundant 4-cycles. Let $\hat{G}$ be the multi-graph obtained from G by contracting all redundant 4-cycles. Then $opt(G) = opt(\hat{G}) + 2q$. Moreover, an α-approximation guarantee for $\hat{G}$ implies the same approximation guarantee for G.*

4.3 Handling cut nodes

Let H be any 2EC graph. Then H can be decomposed into blocks $H_1, \ldots, H_k$ such that each block is either 2NC or else it consists of two nodes with two parallel edges between the two nodes. (Thus, $E(H)$ is partitioned among $E(H_1), \ldots, E(H_k)$ and any two of the blocks H_i, H_j are either disjoint or they have exactly one node in common.) It is well known that an approximately optimal 2-ECSS of H can be computed by taking the union of approximately optimal 2-ECSSes of $H_1, \ldots, H_k$; moreover, the approximation guarantee is preserved, see [16, Proposition 1.4].

Proposition 12. *Suppose that G has cut nodes; let $H_1, \ldots, H_k$ be the blocks of G. Then $opt(G) = \sum_{i=1}^{k} opt(H_i)$. Moreover, an α-approximation guarantee for each of $H_1, \ldots, H_k$ implies the same approximation guarantee for G.*

4.4 Pre-processing for bad-pairs

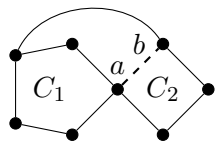
This sub-section presents the pre-processing step of our algorithm that handles the bad-pairs. This step partitions the edges of G among a number of sub-instances G_i such that each sub-instance is 2NC and has no bad-pairs. We ensure the key property that the union of the 2-ECSSes of the sub-instances G_i forms a 2-ECSS of G (see Fact 16).

4.4.1 Bad-pairs and bp-components

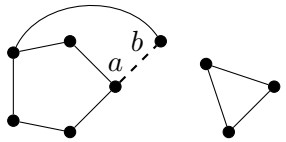
Recall that a *bad-pair* is a pair of nodes $\{v, w\}$ of G such that G has an edge vw of zero cost, and the deletion of both nodes v and w results in a disconnected graph.

For a bad-pair $\{v, w\}$ and one of its bp-components C we use $C^{\{v,w\}}$ to denote the subgraph of G induced by $V(C) \cup \{v, w\}$; thus, we have $C^{\{v,w\}} = G[V(C) \cup \{v, w\}]$; moreover, if C has ≥ 2 nodes, then we use $C^{\oslash}$ to denote the multi-graph obtained from $C^{\{v,w\}}$ by contracting the zero-edge vw, whereas if C has only one node then we take $C^{\oslash}$ to be the same as $C^{\{v,w\}}$ (to ensure that $C^{\oslash}$ is 2NC, see Fact 13, we have to ensure that it has ≥ 3 nodes; if C has only one node, then note that $C^{\{v,w\}}/\{vw\}$ has only 2 nodes).

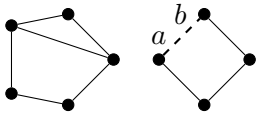
We sketch our plan for handling the bad-pairs in this informal and optional paragraph; readers interested in the formal presentation may skip this paragraph. Assume that G has two or more bad-pairs. We traverse the "tree of bp-components and bad-pairs"; at each iteration, we pick a bad-pair $\{v_\ell, w_\ell\}$ such that all-but-one of its bp-components are free of bad-pairs, see Fact 5. Let C_1 denote the (unique) bp-component that has one or more bad-pairs (w.l.o.g. assume C_1 exists), and let $C_2, \ldots, C_k$ denote the bp-components free of bad-pairs; we call these the *minimal* bp-components. Recall that for any 2EC graph H, we use $\tau(H)$ to denote the cost of a D2 subgraph of H. It is easily seen that for a bp-component C_i, the subgraph of any optimal solution induced by $V(C_i) \cup \{v, w\}$ has cost $\geq \tau({C_i}^{\oslash})$ (see Fact 15). Now, focus on an optimal solution and let F^* denote its set of edges, i.e., (V, F^*) is a 2-ECSS of G of minimum cost. Since (V, F^*) is 2EC, it can be seen that there exists a $j \in \{1, \ldots, k\}$ such that the graph $({C_j}^{\{v_\ell,w_\ell\}} - v_\ell w_\ell)$ contains a v_ℓ, w_ℓ path (see the proof of Lemma 17); then, it follows that $F^* \cup \{v_\ell w_\ell\}$ induces a 2-ECSS of ${C_j}^{\{v_\ell,w_\ell\}}$. In other words, we may assume w.l.o.g. that the zero-edge $v_\ell w_\ell$ (of the bad-pair) is in F^*, and we may "allocate" it to one of the bp-components. Informally speaking, our plan is to return k sub-instances such that one of the sub-instances is of the form ${C_j}^{\{v_\ell,w_\ell\}}$ while the other sub-instances are of the form ${C_i}^{\oslash}$; this can be viewed as "allocating" the zero-edge $v_\ell w_\ell$ to one carefully chosen bp-component C_j by "mapping" C_j to the sub-instance ${C_j}^{\{v_\ell,w_\ell\}}$, while all other bp-components C_i ($i \neq j$) are "mapped" to sub-instances ${C_i}^{\oslash}$. We "allocate" the zero-edge $v_\ell w_\ell$ as follows: For each $i = 1, \ldots, k$, we compute $\tau({C_i}^{\{v_\ell,w_\ell\}})$ and $\tau({C_i}^{\oslash})$; these two numbers are either the same or they differ by one (see Fact 14). Suppose that there is an index $j \in \{2, \ldots, k\}$ such that these two numbers are the same for C_j. Then we "allocate" the zero-edge to C_j; in case of ties, we pick any j such that $\tau({C_j}^{\{v_\ell,w_\ell\}}) = \tau({C_j}^{\oslash})$. (It can be seen that this allocation is best.) On the other hand, if the two numbers differ for each $j \in \{2, \ldots, k\}$, then we "allocate" the zero-edge to C_1. (Although this allocation may disagree with the allocation used by the optimal solution, it turns out that we incur no "loss".) This brings us to the end of the iteration for the bad-pair $\{v_\ell, w_\ell\}$; the algorithm applies the same method to the "remaining graph," namely, either ${C_1}^{\{v_\ell,w_\ell\}}$ or ${C_1}^{\oslash}$. We mention that C_1 plays a special role in our pre-processing algorithm; this will become clear when we prove its correctness (see Lemma 18 below). See the example in Figure 3.



(a) A bad-pair $\{a, b\}$ whose bp-components are C_1 and C_2.



(b) The graphs $C_1^{\{a,b\}}$ (left) and $C_2^{\oslash}$ (right).



(c) The graphs $C_1^{\oslash}$ (left) and $C_2^{\{a,b\}}$ (right).

Figure 3: An example illustrating a "bad allocation" of the zero-edge of a bad-pair. The graph (left subfigure) has one bad-pair $\{a, b\}$ and its bp-components are C_1 (with 4 nodes) and C_2 (with 2 nodes); all edges have cost one, except ab; note that $opt = 8$. If ab is "allocated" to C_1, the sum of the costs of the D2 subgraphs of the resulting sub-instances is 9.

Remark: Recall that k denotes the number of bp-components of the bad-pair $\{v, w\}$. Readers may focus on the key case of $k = 2$ for the rest of this section; our presentation is valid for any $k \geq 2$.

Fact 13. *Let v, w, C be as stated above, and let G be 2NC. Then both $C^{\{v,w\}}$ and $C^{\oslash}$ are 2NC.*

Proof. First, consider $C^{\{v,w\}}$; observe that it has ≥ 3 nodes; since G is 2NC and C is a connected component of $G - \{v, w\}$, for each node $z \in V(C)$, there exist two openly disjoint paths between z and $\{v, w\}$ in $C^{\{v,w\}}$; hence, it can be seen that $C^{\{v,w\}}$ is 2NC.

Now, consider $C^{\oslash}$, and let v^* denote the node resulting from the contraction of vw; by definition, $C^{\oslash}$ has ≥ 3 nodes; arguing as above, for each node $z \in V(C)$, there exist two openly disjoint paths between z and v^* in $C^{\oslash}$; moreover, $C^{\oslash} - v^*$ has a single connected component; hence, it can be seen that $C^{\oslash}$ is 2NC. □

Fact 14. *Let v, w, C be as stated above, and let G be 2NC. We have,*

$$\tau(C^{\oslash}) \quad \leq \quad \tau(C^{\{v,w\}}) \quad \leq \quad 1 + \tau(C^{\oslash}).$$

Proof. Suppose that C has ≥ 2 nodes (if C has one node, then $C^{\oslash} = C^{\{v,w\}}$). To show the first inequality, we start with a D2 subgraph of $C^{\{v,w\}}$ and contract the zero-edge vw; this results in a 2-edge cover of $C^{\oslash}$. Clearly, the cost of this 2-edge cover is $\geq \tau(C^{\oslash})$.

To show the second inequality, we start with a D2 subgraph of $C^{\oslash}$, then "uncontract" the zero-edge vw, and then (if needed) add an edge (of cost one) incident to either v or w. □

Fact 15. *Let $\{v, w\}$ be a bad-pair, and let $C_1, \ldots, C_k$ be all of its bp-components. Let H be a 2-ECSS of G. Consider any C_i, where $i \in \{1, \ldots, k\}$. Then $H[V(C_i) \cup \{v, w\}]$ is connected, and moreover, either it is 2EC or it has exactly one bridge, namely, vw. Therefore, $H[V(C_i{}^{\oslash})]$ is 2EC. Hence, $\text{cost}(H[V(C_i) \cup \{v, w\}]) \geq opt(C_i{}^{\oslash}) \geq \tau(C_i{}^{\oslash})$.*

The above fact is essential for our analysis. It states that for any 2-ECSS H, the subgraph induced by $V(C_i) \cup \{v, w\}$ is either 2EC or it is connected and has vw as its unique bridge. (To see this, observe that for any node $u \in V(C_i)$ there exist two edge-disjoint paths that start at u, end at either v or w, and have all internal nodes in $V(C_i)$.) Therefore, the subgraph obtained from $H[V(C_i) \cup \{v, w\}]$ by contracting the edge vw is 2EC. This implies the key lower bound, $\text{cost}(H[V(C_i) \cup \{v, w\}]) \geq opt(C_i{}^{\oslash})$.

Fact 16. *Let $\{v, w\}$ be a bad-pair, and let $C_1, \ldots, C_k$ be all of its bp-components. Suppose that we pick one of these bp-component C_j and compute a 2-ECSS H_j of ${C_j}^{\{v,w\}}$. For each of the other bp-components C_i, $i \in \{1, \ldots, k\} - \{j\}$, we compute a 2-ECSS H_i of ${C_i}^{\oslash}$. Then the union of the edge sets of H_j and $H_1, \ldots, H_{j-1}, H_{j+1}, \ldots, H_k$ gives a 2-ECSS of G.*

Let $\{v, w\}$ be a bad-pair, and let $C_1, \ldots, C_k$ be its bp-components. Clearly, we have

$$opt(G) \quad \geq \quad \sum_{i=1}^{k} \tau({C_i}^{\oslash}).$$

The next lemma gives a better lower bound on $opt(G)$.

Lemma 17. *Let $\{v, w\}$ be a bad-pair, and let $C_1, \ldots, C_k$ be its bp-components. Suppose that for $j = 2, \ldots, k$, we have $\tau({C_j}^{\oslash}) \quad < \quad \tau({C_j}^{\{v,w\}})$. Then we have*

$$opt(G) \quad \geq \quad \tau({C_1}^{\{v,w\}}) \quad + \quad \sum_{i=2}^{k} \tau({C_i}^{\oslash}).$$

Proof. Let F^* be the set of edges of an optimal solution. We partition $F^* - vw$ into k sets $F_1^*, ..., F_k^*$ by placing an edge $e \in F^* - vw$ in F_i^* iff both ends of e are in $V(C_i) \cup \{v, w\}$. By Fact 15, F_i^* induces a 2-ECSS of ${C_i}^{\oslash}$, $\forall i \in \{1, \ldots, k\}$.

W.l.o.g. we may assume that the zero-edge vw is in F^*. Since the optimal subgraph (V, F^*) is 2EC, there exists a $j \in \{1, \ldots, k\}$ such that the graph $({C_j}^{\{v,w\}} - vw)$ contains a v, w path. By Fact 15, $F_j^* \cup \{vw\}$ induces a 2-ECSS of ${C_j}^{\{v,w\}}$, hence, $\text{cost}(F_j^*) \geq \tau({C_j}^{\{v,w\}})$. Therefore, we have $opt(G) \geq \tau({C_j}^{\{v,w\}}) \quad + \quad \sum_{i=1, i \neq j}^{k} \tau({C_i}^{\oslash})$.

If $j = 1$, then we have proved the inequality in the lemma. Otherwise, observe that $\tau({C_j}^{\{v,w\}}) = 1 + \tau({C_j}^{\oslash})$, and $\tau({C_1}^{\{v,w\}}) \leq 1 + \tau({C_1}^{\oslash})$ (by Fact 14), hence, we obtain the required inequality as follows:

$$opt(G) \quad \geq \quad 1 \; + \; \sum_{i=1}^{k} \tau({C_i}^{\oslash}) \quad \geq \quad \tau({C_1}^{\{v,w\}}) \; + \; \sum_{i=2}^{k} \tau({C_i}^{\oslash}).$$

□

4.4.2 Pre-processing algorithm

Suppose that G is 2NC and it has one or more bad-pairs.

Pre-processing Algorithm (outline):

(0) pick a bad-pair $\{v, w\}$ that satisfies the condition in Fact 5, and let its bp-components be $C_1, C_2, \ldots, C_k$, where $C_2, \ldots, C_k$ are minimal bp-components (free of bad-pairs);

(1) for each $i = 2, \ldots, k$, we compare $\tau({C_i}^{\oslash})$ and $\tau({C_i}^{\{v,w\}})$; if the former is strictly smaller than the latter for all $i = 2, \ldots, k$, then we return the list of graphs ${C_2}^{\oslash}, \ldots, {C_k}^{\oslash}$ and ${C_1}^{\{v,w\}}$; informally speaking, we allocate the zero-edge vw to C_1;

(2) otherwise, we have at least one $j \in \{2, \ldots, k\}$ such that $\tau({C_j}^{\oslash}) = \tau({C_j}^{\{v,w\}})$; then we return the list of graphs ${C_1}^{\oslash}, \ldots, {C_{j-1}}^{\oslash}, {C_{j+1}}^{\oslash}, \ldots, {C_k}^{\oslash}$ and ${C_j}^{\{v,w\}}$; informally speaking, we allocate the zero-edge vw to C_j;

(3) let G' denote the "remaining graph," either ${C_1}^{\{v,w\}}$ or ${C_1}^{\oslash}$; **stop** if G' has no bad-pairs, otherwise, apply the same pre-processing iteration to G'.

Let $\tau^{alg}(G)$ denote the sum of the costs of the D2 subgraphs of the graphs G_i returned by the pre-processing algorithm, where each G_i is of the form $C_j{}^{\{v_\ell,w_\ell\}}$ or $C_j{}^{\oslash}$ w.r.t. some bad-pair $\{v_\ell, w_\ell\}$; thus, $\tau^{alg}(G) = \sum_i \tau(G_i)$. We use the same notation $\tau^{alg}(\cdot)$ for other 2NC graphs.

Lemma 18.

$$opt(G) \quad \geq \quad \tau^{alg}(G)$$

Proof. We use induction on the number of bad-pairs. It is easily seen, via Lemma 17, that the result holds if either G has no bad-pairs or G has exactly one bad-pair. Now, assume that G has two or more bad-pairs.

Let $\{v, w\}$ be the initial bad-pair for our pre-processing, let $C_1, \ldots, C_k$ denote all of its bp-components, and let $C_2, \ldots, C_k$ be minimal bp-components (free of bad-pairs). Let $H_1, \ldots, H_k$ denote the graphs corresponding to $C_1, \ldots, C_k$ returned by the pre-processing (i.e., each H_i is either $C_i{}^{\oslash}$ or $C_i{}^{\{v,w\}}$, where $i \in \{1, \ldots, k\}$). For each $i = 1, \ldots, k$, let V_i denote the set of nodes $V(C_i) \cup \{v, w\}$; thus, V_i denotes the node-set of $C_i{}^{\{v,w\}}$.

We have $\tau^{alg}(G) \;=\; \tau^{alg}(H_1) + \sum_{i=2}^{k} \tau^{alg}(H_i)$; to see this, note that the execution of the algorithm on G consists of two stages: the first stage applies to the graphs $H_2, \ldots, H_k$ and it computes $\tau^{alg}(H_2), \ldots, \tau^{alg}(H_k)$, while the second stage applies to the graph H_1 and it computes $\tau^{alg}(H_1)$.

Let G^* denote an optimal solution, i.e., a 2-ECSS of minimum cost, and w.l.o.g. assume that G^* contains all zero-edges of G. We have $opt(G) = \text{cost}(G^*) = \sum_{i=1}^{k} \text{cost}(G^*[V_i])$.

By the induction hypothesis, we have $opt(H_i) \geq \tau^{alg}(H_i)$, $\forall i \in \{1, \ldots, k\}$.

By Fact 15 and the induction hypothesis, we have

if H_i is of the form $C_i{}^{\oslash}$, then

$$\text{cost}(G^*[V_i]) \geq opt(H_i) \geq \tau^{alg}(H_i), \quad \forall i \in \{1, \ldots, k\}. \qquad (*)$$

We complete the proof by examining a few cases.

Case 1: the zero-edge vw is allocated to C_1: First, suppose that $G^*[V_1]$ is 2EC. Then, we have $\text{cost}(G^*[V_1]) \geq opt(H_1) \geq \tau(H_1) = \tau^{alg}(H_1)$; combining this with $(*)$ we have

$$opt(G) = \sum_{i=1}^{k} \text{cost}(G^*[V_i]) \geq \sum_{i=1}^{k} opt(H_i) \geq \sum_{i=1}^{k} \tau^{alg}(H_i) = \tau^{alg}(G).$$

Now, suppose that $G^*[V_1]$ is not 2EC; then, by Fact 15, it is connected and has only one bridge, namely, vw. Moreover, it can be seen that G has an edge $\hat{e}$ whose end nodes are in two different connected components of $G^*[V_1] - vw$. (To see this, note that $G[V_1]$ is 2NC, hence, $(G[V_1] - vw)$ has a v, w path, and one of the edges of this path satisfies the requirement on $\hat{e}$.) Thus, adding $\hat{e}$ to $G^*[V_1]$ results in a 2EC graph of cost $1 + \text{cost}(G^*[V_1]) \geq opt(H_1) \geq \tau(H_1) = \tau^{alg}(H_1)$. Also, observe that G^* has two edge-disjoint v, w paths, hence, one of the subgraphs $G^*[V_\ell] - vw$, where $\ell \in \{2, \ldots, k\}$, has a v, w path. Hence, $\text{cost}(G^*[V_\ell]) \geq \tau(C_\ell{}^{\{v,w\}}) = 1 + \tau(C_\ell{}^{\oslash}) = 1 + \tau^{alg}(H_\ell)$ (we used the fact that $\tau(C_i{}^{\{v,w\}}) > \tau(C_i{}^{\oslash})$, $\forall i \in \{2, \ldots, k\}$). By the above two inequalities and $(*)$, we have

$$opt(G) = \sum_{i=1}^{k} \text{cost}(G^*[V_i]) \geq$$
$$(\tau^{alg}(H_1) - 1) + (\tau^{alg}(H_\ell) + 1) + \sum_{i \in \{2,\ldots,\ell-1,\ell+1,\ldots,k\}} \tau^{alg}(H_i) = \tau^{alg}(G).$$

Case 2: the zero-edge vw is allocated to C_j, $j \in \{2, \ldots, k\}$: Thus, we have $\tau(C_j{}^{\oslash}) = \tau(C_j{}^{\{v,w\}})$ (otherwise, vw cannot be allocated to C_j). Then, by Fact 15, we have

$$\text{cost}(G[V_j]) \geq \tau(C_j{}^{\oslash}) = \tau(C_j{}^{\{v,w\}}) = \tau^{alg}(H_j).$$

This, together with $(*)$, implies that $opt(G) \geq \tau^{alg}(G)$.

This completes the proof of the lemma. □

Proposition 19. *Suppose that a 2-ECSS of cost $\leq \alpha\, \tau(G_i)$ can be computed in polynomial time for any well-structured MAP instance G_i. Then the reduction for handling bad-pairs (namely, (pp4) in Section 3) computes a 2-ECSS of cost $\leq \alpha\, \tau(G)$ for G, assuming that each of the sub-instances G_i computed by the reduction is well-structured (recall that each G_i is of the form $C_j{}^{\{v_\ell, w_\ell\}}$ or $C_j{}^{\oslash}$ w.r.t. some bad-pair $\{v_\ell, w_\ell\}$). Moreover, the reduction can be implemented in polynomial time.*

Proof. Lemma 18 states that $\tau^{alg}(G) \leq opt(G)$. This inequality implies the first part of the proposition. To see this, observe that we can compute a 2-ECSS of cost $\leq \alpha\, \tau(G_i)$ for each of the sub-instances G_i computed by the reduction; the union of all these 2-ECSSes is a 2-ECSS of G (by Fact 16), and it has cost $\leq \alpha\ \sum_i \tau(G_i) \leq \alpha\ \tau^{alg}(G) \leq \alpha\ opt(G)$ (in fact, by Theorem 6, we may fix $\alpha = \frac{7}{4}$).

It can be seen that this reduction runs in polynomial time. □

5 Bridge covering

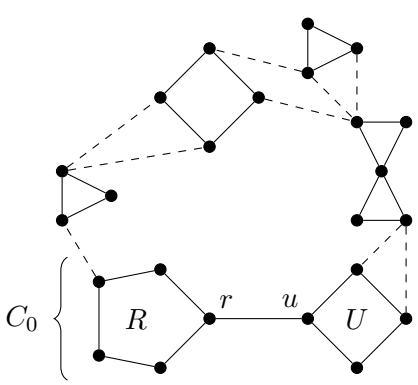
In this section and in Section 6, we assume that the input is a well-structured MAP instance.

We start by illustrating our method for bridge covering on a small example. After computing D2, recall that we give 1.75 initial credits to each unit-edge of D2, thereby giving each of these edges 0.75 working credits (we keep aside one credit to buy the unit-edge for our solution). Consequently, each 2ec-block of D2 gets ≥ 1.5 working credits (this is explained below). We want to buy more edges to add to D2 such that all bridges are "covered", and we pay for the newly added edges via the working credits.

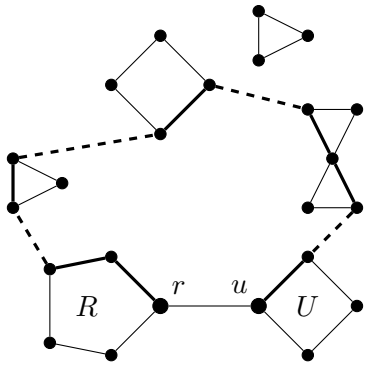
Observe that each 2ec-block of D2 has ≥ 1.5 working credits, because it has ≥ 2 unit-edges; to see this, suppose that a 2ec-block B has b nodes; if $b = 2$, then B has two parallel unit-edges; otherwise, B has $\geq b$ edges and has $\leq \lfloor b/2 \rfloor$ zero-edges, so B has $\geq \lceil b/2 \rceil$ unit-edges, and we have $b \geq 3$.

In what follows, we use the term credits to mean the working credits of the algorithm; this excludes the unit credit retained by every unit-edge of the current solution subgraph; for example, a 6-cycle of D2 that consists of four unit-edges has 3 credits (and it has 7 initial credits).

Consider an example such that D2 has a connected component C_0 that has one bridge and two 2-ec blocks R and U; let ru denote the unique bridge where r is in R and u is in U. (It can be seen that ru is a zero-bridge, but we will not use this fact.) Since G is assumed to be 2NC, $G - ru$ is connected, hence, it contains a u, r path; let us pick a u, r path Y of $G - ru$ that has only its prefix and suffix in common with C_0 and that has the minimum number of non-D2 edges. Our plan is to augment D2 by adding the edge set $E(Y) - E(\text{D2})$, thus "covering" the bridge ru. We may view this as an "ear-augmentation step" that adds the ear Y. We have to pay for the non-D2 edges of Y by using the credits available in D2. Let us traverse Y from u to r, and each time we see a non-D2 edge of Y, then we will pay for this edge. For the sake of illustration, consider the example in Figure 4b; note that the u, r path Y (in the right subfigure) has $\ell = 4$ non-D2 edges (indicated by dashed lines). When we traverse Y starting from u, then observe that each of the first $(\ell - 1)$ of these edges has its last node in a distinct 2ec-block of D2 (moreover, none of these



(a) The D2 subgraph is indicated by solid lines; it has 5 connected components, one has the bridge ru and the other 4 are bridgeless. The dashed lines indicate the edges of $E(G) - E(\text{D2})$.



(b) The thick lines indicate a "shortest" u, r path of $G - ru$; this path has the minimum number of non-D2 edges (4 dashed lines); our goal is to "pay" for these 4 non-D2 edges via the credits available in D2.

Figure 4: Illustration of bridge covering on a simple example.

$(\ell - 1)$ 2ec-blocks is in C_0). We pay for these $(\ell - 1)$ edges by borrowing one credit from the credit of these $(\ell - 1)$ 2ec-blocks. We need one more credit to pay for the last non-D2 edge of Y. We get this credit from the prefix of Y between its start and its first non-D2 edge; in particular, we borrow one credit from the credit of U. Thus, we can pay for all the non-D2 edges of Y. Observe that by adding the edge set $E(Y) - E(\text{D2})$ to D2, we have merged several 2ec-blocks (including R and U) into a new 2ec-block. We give the new 2ec-block (that contains R and U) the credit of R as well as any unused credit of the other 2ec-blocks incident to Y.

The general case of bridge covering is more complicated.

Remark: For the sake of exposition, we may impose a direction on a path, cycle, or ear (e.g., we traverse Y from u to r in the discussion above). Nevertheless, the input G is an undirected graph, so (formally speaking) there is no direction associated with paths or cycles of G.

5.1 Post-processing D2

Immediately after computing D2, we apply a post-processing step that replaces some unit-edges of D2 by other unit-edges to obtain another D2 that we denote by $\widehat{\text{D2}}$ that satisfies the following key property:

> *Every pendant 2ec-block B of $\widehat{\text{D2}}$ that is incident to a zero-bridge has* $\text{cost}(B) \geq 3$, *and hence, has ≥ 2.25 credits (see part (2) of the credit invariant below).*

In other words, if a 2ec-block of $\widehat{\text{D2}}$ has ≤ 2 unit-edges, then either the 2ec-block is not pendant (i.e., it is incident to no bridges or ≥ 2 bridges) or it is pendant and is incident to a unit-bridge.

For any subgraph G' of G, let $F_0(G')$ denote the set of zero-bridges (of G') that are incident to pendant 2ec-blocks (of G') of cost ≤ 2 (the notation F_0 is used only in this subsection); moreover, let $\#\text{comp}(G')$ denote the number of connected components of G'. Thus, the goal of the post-processing step is to compute $\widehat{\text{D2}}$ such that $F_0(\widehat{\text{D2}})$ is empty.

The post-processing step is straightforward. W.l.o.g. assume that the initial D2 contains all the zero-edges; we start with this D2 and iterate the following step. Let D2^{old} denote the D2 at the start of the iteration. If $F_0(\text{D2}^{old})$ is empty then we are done, we take D2^{old} to be $\widehat{\text{D2}}$. Otherwise, we

pick any zero-bridge v_0v_1 in $F_0(\mathrm{D2}^{old})$, and we take B to be a pendant 2ec-block with $\mathrm{cost}(B) = 2$ that is incident to v_0v_1; note that B exists by the definition of F_0. Now, observe that all of the edges of B incident to v_0 have cost one, hence, it can be seen that B has either 2 or 3 nodes; then, since G is 2NC, it can be seen that G has a unit-edge e between $V(B) - \{v_0\}$ and $V - V(B)$, and moreover, B has a unit-edge f that is incident to e and to v_0. We replace f by e. It is easily seen that the resulting subgraph is a 2-edge cover of the same cost; we denote it by $\mathrm{D2}^{new}$.

We claim that $F_0(\mathrm{D2}^{new}) \subseteq F_0(\mathrm{D2}^{old})$, and moreover,

$$|F_0(\mathrm{D2}^{new})| + \#\mathrm{comp}(\mathrm{D2}^{new}) \;<\; |F_0(\mathrm{D2}^{old})| + \#\mathrm{comp}(\mathrm{D2}^{old}).$$

To see this, first suppose that the end nodes of e are in two different connected components of $\mathrm{D2}^{old}$; then it can be seen that $F_0(\mathrm{D2}^{new}) \subseteq F_0(\mathrm{D2}^{old})$, and hence, the claimed inequality holds; otherwise, (the end nodes of e are in the same connected component of $\mathrm{D2}^{old}$) it can be seen that v_0v_1 is not a bridge of $\mathrm{D2}^{new}$ and it can be seen that $F_0(\mathrm{D2}^{new})$ is a proper subset of $F_0(\mathrm{D2}^{old})$, and hence, the claimed inequality holds.

Thus, after $O(n)$ iterations, we find $\widehat{\mathrm{D2}}$ that satisfies the key property. See Figure 5 for illustrations.

Proposition 20. *There is a polynomial-time algorithm that finds a 2-edge cover $\widehat{\mathrm{D2}}$ such that $F_0(\widehat{\mathrm{D2}})$ is empty; thus, $\widehat{\mathrm{D2}}$ satisfies the key property.*

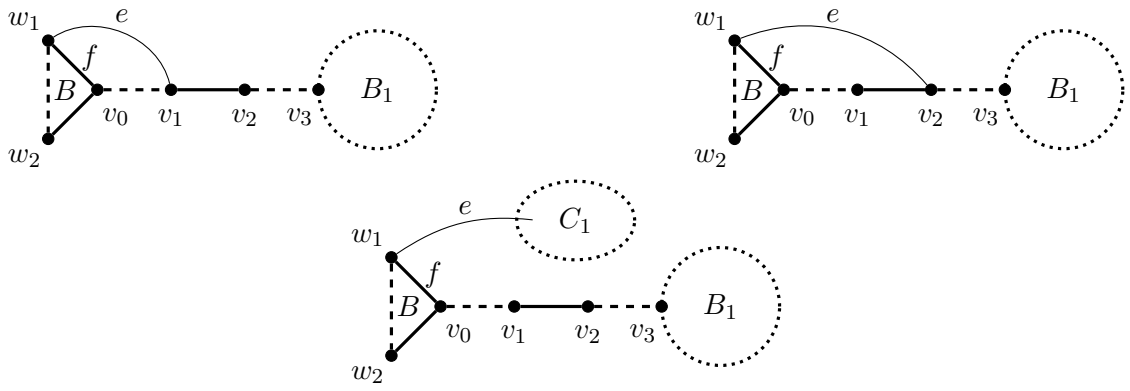


Figure 5: Illustrations of some cases that could arise during post-processing. Edges of D2 are indicated by thick lines. Edges of cost zero and one are indicated by dashed and solid lines, respectively.

5.2 Credit invariant and charging lemma

Let $H = (V, F)$ denote the current graph of picked edges; thus, at the start, H is the same as $\widehat{\mathrm{D2}}$, and we may assume that H has one or more bridges (otherwise, bridge covering is trivial). By an *original* 2ec-block B of H we mean a 2ec-block (of the current H) such that B is also a 2ec-block of $\widehat{\mathrm{D2}}$ (i.e., the set of edges incident to $V(B)$ is the same in both $\widehat{\mathrm{D2}}$ and the current H). By a *new* 2ec-block of H we mean a 2ec-block (of the current H) that is not an original 2ec-block. Similarly, we define an original/new connected component of H.

We call a node v of H a *white* node if v belongs to a 2ec-block of H, otherwise, we call v a *black* node.

Fact 21. *Suppose that H has one or more black nodes. Let v be a black node of H. Then all edges of H incident to v are bridges of H, and v is incident to ≥ 2 bridges of H. Every maximal path of H that starts with v contains a white node.*

Remark: Let H' denote the graph obtained from H by contracting each 2ec-block; thus, each 2ec-block of H maps to a "contracted" white node of H', each black node of H maps to a black node of H', and each bridge of H maps to a bridge of H'. Clearly, H' is a forest; it may have isolated nodes (these correspond to bridgeless connected components of H). Clearly, H' has ≥ 2 edges incident to each black node.

By a *b-path* of H we mean a path consisting of bridges that starts and ends with arbitrary nodes (white or black) such that all internal nodes are black nodes. We say that two 2ec-blocks of a connected component of H are *b-adjacent* if there exists a b-path whose terminal nodes are in these two 2ec-blocks.

Initially, our algorithm picks a connected component C_0 of $H = \widehat{\mathrm{D2}}$ that has one or more bridges, and then picks any pendant 2ec-block of C_0 and designates it as the *root 2ec-block* R (in this section, R always denotes the root 2ec-block). Also, the algorithm picks the unique bridge of C_0 incident to R; we denote this bridge by ru, where r is in R. If ru is a zero-bridge, then since C_0 is an original connected component, R has ≥ 2.25 credits (by the key property of $\widehat{\mathrm{D2}}$); otherwise, R has ≥ 1.5 credits. Immediately after designating R, we ensure that R has ≥ 2 credits. If R is short of credit (i.e., it has only 1.5 credits), then we borrow 0.5 credits from a b-adjacent 2ec-block of C_0.

The bridge covering step maintains the following invariant. (At the end of this section, we argue that this invariant is preserved in each iteration, see Proposition 24.)

Credit invariant:

(1) *Each original bridgeless connected component has* ≥ 1.5 *credits, and each new bridgeless connected component has* ≥ 2 *credits. Moreover, each unit-bridge of* H *has* 0.75 *credits.*

(2) *Within each original connected component of* H *each pendant 2ec-block that is incident to a zero-bridge has* ≥ 2.25 *credits.*

(3) *Suppose that the root 2ec-block* R *is well defined. Then either* R *is incident to a unit-bridge and has* ≥ 2 *credits, or* R *has* ≥ 2.25 *credits. Moreover, each 2ec-block that is b-adjacent to* R *has* ≥ 1 *credits, and every other 2ec-block of* H *has* ≥ 1.5 *credits.*

In an arbitrary iteration of the algorithm, either R is a pendant 2ec-block of an original connected component C_0 of H, or R has been designated as the root 2ec-block by the previous iteration. If possible, the algorithm chooses ru to be a unit-bridge incident to R, otherwise, ru is a zero-bridge incident to R.

When we remove the edge ru from C_0 then we get two connected components; let us denote them by C_0^r (it contains r but not u) and C_0^u (it contains u but not r). Recall that $G - ru$ has a path between C_0^u and C_0^r. Let P be such a path of $G - ru$ that starts at some node a_0 of C_0^u, ends at some node z_0 of C_0^r, has no internal nodes in C_0, and (subject to the above) minimizes $|E(P) - E(H)|$. Note that there could be many choices for the nodes a_0 and z_0; although the choice of these nodes is important for our analysis (see Lemma 23 and its proof), the next lemma (Lemma 22) and its proof apply for all valid choices of these two nodes. See Figure 6.

Let H^{new} denote $H \cup (E(P) - E(H))$ and let B^{new} denote the 2ec-block of H^{new} that contains a_0, z_0, and R. We designate B^{new} as the root 2ec-block of H^{new}, provided B^{new} is incident to a bridge of H^{new}.

Each iteration may be viewed as an ear-augmentation step that adds either one or two open ears to C_0, e.g., the path P may be viewed as an open ear w.r.t. C_0. On the other hand, our charging scheme (for paying for the edges of $E(P) - E(H)$) views each iteration as adding an ear

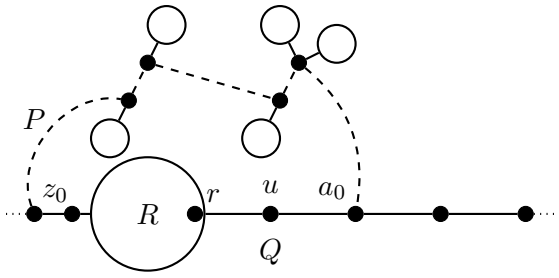


Figure 6: Illustration of plan for "bridge covering." Large circles indicate 2ec-blocks. Dashed lines indicate P. Q is the path of C_0 between r and a_0, where a_0 is the unique node of P in C_0^u.

w.r.t. R, that is, we take the ear to be the union of three paths, namely, an r, a_0 path of C_0 that contains ru, the path P, and a path of C_0 between z_0 and R (we mention that our charging scheme also uses the credits available in the first of these three paths). To avoid confusion, we call these the R-ears, and unless mentioned otherwise, an "ear" means an ear w.r.t. C_0.

The next lemma explains how we can use the credits available in H to pay for all-but-one of the unit-edges added by an ear-augmentation step. We use credits(H) to denote the (working) credit of H (i.e., the sum of the (initial) credits of the unit-edges of H minus the number of unit-edges of H).

Lemma 22. *Let H, C_0 be as stated above, and suppose that H satisfies the credit invariant. Let P be an open ear w.r.t. C_0 with end nodes a_0, z_0 such that $|E(P) - E(H)|$ is minimum. Let $C_1, \ldots, C_k$ denote the connected components of H that contain (at least) one internal node of P (thus, $C_0 \neq C_i$, $\forall i = 1, \ldots, k$). Then, the credits of $C_1, \ldots, C_k$ can be redistributed such that the credit invariant holds for H^{new}, and we have*

$$|E(P) - E(H)| - 1 \leq \text{credits}(H) - \text{credits}(H^{new}).$$

Proof. Our goal is to show that we can pay via the credits of $C_1, \ldots, C_k$ for all-but-one of the edges of $E(P) - E(H)$, while ensuring that the credit invariant holds for H^{new}. When we traverse P from a_0 to z_0, then observe that each of the edges of $E(P) - E(H)$, except the last such edge, has its last end node in a connected component $C_i \neq C_0$ of H; we use the credits available in that connected component to pay the cost of the edge. The rest of the proof shows how we can obtain one unit from the credit of the relevant connected component while ensuring that the credit invariant continues to hold for H^{new}. (Proposition 24 below shows that the credit invariant is preserved in each iteration.)

Consider any original connected component $C \neq C_0$ of H that contains one of the internal nodes of P. See Figure 7. By our choice of P, there is a unique edge of P that "enters" C and there is a unique edge of P that "exits" C, i.e., there is a unique edge f with one end node in C and the other end node in the subpath of P between a_0 and C, and similarly, there is a unique edge e with one end node in C and the other end node in the subpath of P between C and z_0.

Let s_0 denote the end node of f in C, and let t_0 denote the end node of e in C. Possibly, $s_0 = t_0$. Let $P(s_0, t_0)$ denote the s_0, t_0 sub-path of P. Clearly, $P(s_0, t_0)$ is contained in H.

First, suppose that either s_0 or t_0 is a white node; there is a 2ec-block of C, call it B, that contains this white node; then we take one unit from the credit of B and use that to pay for the edge f. (Recall that B^{new} is designated as the root 2ec-block of H^{new}, and note that B^{new} contains both s_0 and t_0; moreover, the 2ec-block B (of H) is also contained in B^{new}; it can be seen that the credit invariant is maintained in H^{new} although we borrowed one credit from B.)

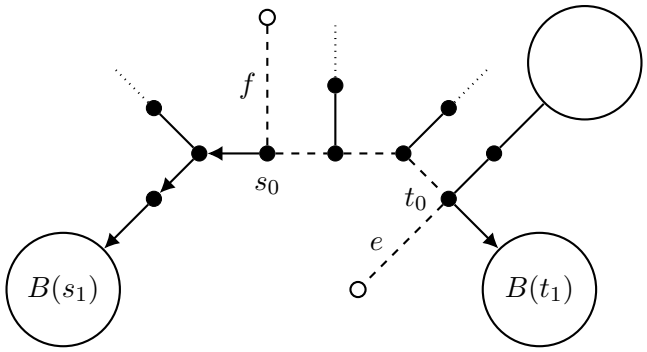


Figure 7: Illustration of the notation in Lemma 22. A connected component $C \neq C_0$ of H is shown (C does not contain R). The large circles indicate 2ec-blocks of C. The path P is indicated by dashed lines; P contains 3 edges of C. The arrows indicate maximal b-paths ending at 2ec-blocks.

If both s_0 and t_0 are black nodes, then we resort to a more complex scheme. Let $C(s_0, s_1)$ be any maximal b-path of $C - E(P(s_0, t_0))$ that starts with s_0 and ends with a white node s_1, and let $B(s_1)$ denote the 2ec-block that contains the white node s_1; note that $B(s_1)$ has no nodes in common with B^{new}. Similarly, let $B(t_1)$ denote a 2ec-block that contains the terminal white node t_1 (where, $t_1 \neq s_1$) of a maximal b-path of $C - E(P(s_0, t_0))$ that starts with t_0; it can be seen that $B(t_1)$ has no nodes in common with B^{new}.** We take 0.5 credits from each of $B(s_1)$ and $B(t_1)$ and use that to pay for the edge f. (In H^{new}, observe that both s_0 and t_0 are contained in the root 2ec-block B^{new}; moreover, $B(s_1)$ and $B(t_1)$ are 2ec-blocks, and moreover, both these 2ec-blocks are b-adjacent to B^{new}; hence, the credit invariant is maintained in H^{new} although we borrowed 0.5 credits from each of $B(s_1)$ and $B(t_1)$.)

Finally, consider any new connected component $C \neq C_0$ of H that contains one of the internal nodes of P. Our algorithm ensures that every new connected component of H, except for C_0, is 2EC, hence, C is 2EC. We take one unit from the credit of C and use that to pay for the unique edge of P that "enters" C. (The credit invariant is maintained in H^{new} because C is contained in B^{new}, the designated root 2ec-block of H^{new}.) □

5.3 Algorithm and analysis for bridge covering

We present the algorithm and analysis of bridge covering based on Lemma 22.

Recall that H satisfies the credit invariant initially, and that Lemma 22 allows us to pay for all-but-one of the unit-edges added by an ear-augmentation step. We charge the remaining cost (of one) to a prefix of the R-ear that we denote by Q; Q is the maximal path of C_0 contained in the R-ear and starting with the edge ru. Let a_0 denote the other end node of Q (thus, when we traverse the edges (and nodes) of the R-ear starting with the edge ru, then a_0 is the first node incident to an edge of the R-ear in $E(G) - E(H)$). Since our goal is to collect as much credit as possible from Q we choose the R-ear such that either (i) Q has a white node w ($w \neq r$) or (ii) Q has no white nodes (other than r), Q has the maximum cost possible, and subject to this, Q has the maximum number of bridges possible. In other words, we choose (the 3-tuple) P, a_0, z_0 such that P is a path of $G - ru$ with one end node a_0 in C_0^u and the other end node z_0 in C_0^r, none of the internal nodes of P is in C_0, the associated prefix Q satisfies condition (i) or (ii) (stated above), and (subject to all the above requirements) P has the minimum number of edges from $E(G) - E(H)$. We can easily compute (the 3-tuple) P, a_0, z_0 in polynomial time via standard methods from graph algorithms; this is discussed in the next lemma.

**If $s_0 = t_0$, then note that s_0 is incident to ≥ 2 bridges of $C = C - E(P(s_0, t_0))$, hence, we can ensure that $t_1 \neq s_1$.

Lemma 23. *P, a_0, z_0 satisfying the requirements stated above can be computed in polynomial time.*

Proof. For each node $v \in C_0^u$, we define $\gamma(v)$ as follows: $\gamma(v) = \infty$ if every path of C_0 between v and r contains a white node w, $w \neq r$; otherwise, $\gamma(v) = |E(C_0(v,r))| + n \cdot \text{cost}(C_0(v,r))$, where $C_0(v,r)$ denotes the unique b-path of C_0 between v and r. (Informally speaking, $\gamma(v)$ assigns a "rank" to each node v of C_0^u; if every v,r path of C_0 contains two or more white nodes, then the rank is ∞, otherwise, the rank is determined by the unique b-path of C_0 between v and r, and we rank according to the 2-tuple consisting of the cost and the number of bridges of the relevant path.)

Then, we construct the following weighted directed graph: the directed graph has two oppositely oriented edges for each edge of $G - V(C_0)$, it has an edge oriented out of $V(C_0^r)$ for each edge of $G - ru$ in the cut $\delta_G(V(C_0^r))$, and it has an edge oriented into $V(C_0^u)$ for each edge of $G - ru$ in the cut $\delta_G(V(C_0^u))$ (there are no oriented edges corresponding to other edges of G); for example, an edge $vx \in E(G - ru)$ with $v \in V(C_0^r)$ and $x \in V - V(C_0^r)$ is replaced by the (single) arc (v,x), an edge $v'y \in E(G - ru)$ with $v' \in V(C_0^u)$ and $y \in V - V(C_0^u)$ is replaced by the (single) arc (y,v'), and an edge $xy \in E(G - ru)$ with both $x, y \in V - V(C_0)$ is replaced by the pair of arcs $(x,y),(y,x)$. We assign weights of zero to the oriented edges associated with the edges of H, and weights of one to the other oriented edges (associated with the edges of $E(G) - E(H)$). Then, we apply a reachability computation, taking all the nodes in C_0^r to be the sources. We claim that a node $v \in C_0^u$ is reachable from C_0^r (in the directed graph) iff $G - ru$ has a path between C_0^r and v such that no internal node of the path is in C_0.

Thus, we can find a_0 and z_0 that satisfy the requirements: we choose a_0 to be a node $v \in C_0^u$ that is reachable from C_0^r (in the directed graph) and that has the maximum $\gamma()$ value, and then we choose z_0 to be a node in C_0^r such that the directed graph has a path from this node to a_0. Then, we take P to be a shortest z_0, a_0 path in the (weighted) directed graph. □

An outline of the bridge covering step follows.

Bridge covering (outline):

(0) compute D2, then post-process D2 to obtain $\widehat{\text{D2}}$, and let $H = \widehat{\text{D2}}$;

(1) pick any (original) connected component C_0 of H that has a bridge, then pick any pendant 2ec-block of C_0 and designate it as the root 2ec-block R;

(2) **repeat**

> if possible, pick a unit-bridge incident to R, otherwise, pick any zero-bridge incident to R, and denote the picked bridge by ru;
> apply one ear-augmentation step (add one or two ears) to cover a sub-path of bridges starting with ru, and let B^{new} denote the resulting 2ec-block that contains R and ru;
> let $R := B^{new}$;

until $R = B^{new}$ has no incident bridges;

(3) **stop** if H has no bridges, otherwise, **go to** (1).

Suppose that the prefix Q (excluding r) of the R-ear contains a white node w (thus, $w \neq r$). Let $B \neq R$ denote the 2ec-block that contains w. Observe that B^{new} contains B and the credits of both B and Q are available to pay for the current ear-augmentation step and to re-establish the credit invariant. Note that we must pay one unit for the ear-augmentation step and we may have

to add another 0.25 units to the credit of B^{new} (if B^{new} is a 2ec-block that is incident to one or more zero-bridges and to no unit-bridge). If the sum of the credits of B and Q is < 1.25, then it can be seen that B has one credit and Q has zero credit. Thus, Q has only one zero-bridge ru. Then by our choice of ru and the credit invariant, R already has ≥ 2.25 credits, hence, we need only one credit for the ear-augmentation step (since B^{new} gets all the credits of R).

Now, assume that the prefix Q (excluding r) has no white nodes; in particular, the nodes u and a_0 are black. It is easily seen that Q has ≥ 2 bridges. (Since G is 2NC, $G - u$ has a path between the connected component of $C_0 - u$ that contains R and each of the other connected components of $C_0 - u$; this implies that there exists a choice of an R-ear such that the associated prefix Q' has ≥ 2 bridges and we have $\text{cost}(Q') \geq 1$.)

Suppose that $\text{cost}(Q) \geq 2$, thus, Q has ≥ 2 unit-bridges. Then, we have ≥ 1.5 credits available in Q. This suffices to pay for the current ear-augmentation step and to re-establish the credit invariant.

Now, we may assume that $\text{cost}(Q) = 1$. Let us denote the node sequence of Q by $v_0, v_1, v_2, \ldots$, where $v_0 = r$ and $v_1 = u$. Clearly, we have three possibilities:

Case 1: Q consists of 2 bridges and $\text{cost}(v_0v_1) = 1$, $\text{cost}(v_1v_2) = 0$: We argue that this possibility cannot occur. Observe that $v_0 = r$, $v_1 = u$ and $v_2 = a_0$. Observe that $G - \{v_1, v_2\}$ is connected; otherwise, $\{v_1, v_2\}$ would be a bad-pair. Consider the connected components of $C_0 - \{v_1, v_2\}$; let S denote the set of nodes of the connected component that contains R, and let T denote $V(C_0) - \{v_1, v_2\} - S$. Since $G - \{v_1, v_2\}$ is connected, it has path $P_\star$ between a node $z_\star \in S$ and a node $a_\star \in T$ such that $P_\star$ has no internal nodes in C_0. Observe that every path of C_0 between r and $a_\star$ has cost ≥ 2 (because such a path contains the unit-edge ru as well as another edge of the cut $\delta_{C_0}(\{v_1, v_2\})$, and all edges of this cut have cost one); thus, the prefix $Q_\star$ associated with $P_\star$ has cost ≥ 2. This contradicts our choice of P, a_0, z_0 (because the prefix Q has cost one). See Figure 8.

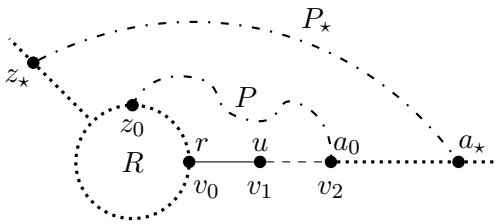


Figure 8: Illustration of Case 1 (bridge covering).

Case 2: Q consists of 3 bridges and $\text{cost}(v_0v_1) = 0$, $\text{cost}(v_1v_2) = 1$, $\text{cost}(v_2v_3) = 0$: Then, we have 0.75 credits available in Q. We argue that this suffices to pay for the current ear-augmentation step and to re-establish the credit invariant. Observe that $v_0 = r$, $v_1 = u$ and $v_3 = a_0$. By our choice of ru and the credit invariant, R has ≥ 2.25 credits. Thus, we can pay one unit for the ear-augmentation step and we have 2 credits (from R) left for B^{new}. Moreover, v_3 is a black node, and it can be seen that one of the unit-bridges of C_0 incident to v_3 becomes a bridge incident to B^{new}; in other words, at the next iteration, when we designate B^{new} as the root 2ec-block, then only 2 credits are required for B^{new}. See Figure 9.

Case 3: Q consists of 2 bridges and $\text{cost}(v_0v_1) = 0$, $\text{cost}(v_1v_2) = 1$: Observe that $v_0 = r$, $v_1 = u$ and $v_2 = a_0$. Note that v_2 is a black node, and it is incident to a bridge other than v_2v_1. There are two subcases, namely, either v_2 is incident to two unit-bridges or not, and we

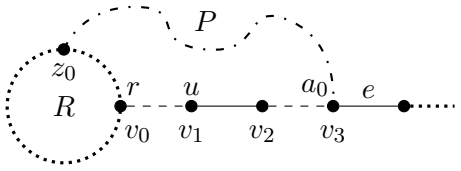


Figure 9: Illustration of Case 2 (bridge covering).

choose v_3 appropriately. In the former subcase, we take v_2v_3 to be a unit-bridge, and in the latter subcase we have to take v_2v_3 to be the unique zero-bridge incident to v_2.

We handle the first subcase (with $\text{cost}(v_2v_3) = 1$) similarly to Case 2 above, that is, we argue that the 0.75 credits of the bridge v_1v_2 suffice to pay for the current ear-augmentation step and to re-establish the credit invariant (we skip the details).

In the second subcase (with $\text{cost}(v_2v_3) = 0$), observe that H has precisely two bridges incident to v_2, because v_2 is incident to only one unit-bridge (by our choice of v_2v_3). Our plan is to add a second ear and then observe that the edge v_1v_2 becomes redundant after the addition of the two ears, hence, we can delete this edge from H thereby gaining 1.75 credits. (Note that when we delete a unit-edge of $\widehat{\text{D2}}$ from our solution subgraph H, then all of the 1.75 credits of that edge become available as credits.) Moreover, we get another 0.75 (or more) credits from the addition of the two ears. Thus we get ≥ 2.5 credits, and this suffices to pay for the addition of two ears and to re-establish the credit invariant. See Figure 10.

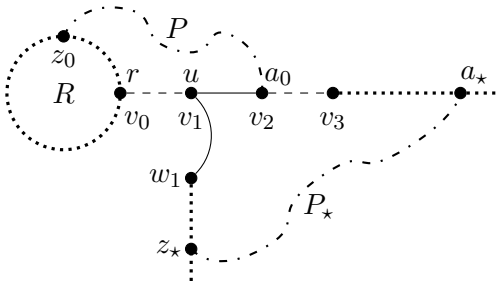


Figure 10: Illustration of Case 3 (bridge covering).

Observe that $G - \{v_2, v_3\}$ is connected; otherwise, $\{v_2, v_3\}$ would be a bad-pair. Consider the connected components of $C_0 - \{v_2, v_3\}$; let S denote the set of nodes of the connected component that contains R, and let T denote $V(C_0) - \{v_2, v_3\} - S$. Since $G - \{v_2, v_3\}$ is connected, it has a path $P_\star$ between a node $z_\star \in S$ and a node $a_\star \in T$ such that $P_\star$ has no internal nodes in C_0. Moreover, w.l.o.g. we assume that $P_\star$ has the minimum number of edges from $E(G) - E(H)$. It can be seen that $P_\star, a_\star, z_\star$ satisfy the following:

(i) Either $z_\star = v_1$ or there is a bridge v_1w_1 (of C_0) such that $v_1w_1 \neq v_1v_0$, $v_1w_1 \neq v_1v_2$, and $z_\star$ is in the connected component of $C_0 - v_1w_1$ that contains w_1. This follows from a contradiction argument; the only other possibility is that $z_\star$ is in C_0^r (the connected component of $C_0 - ur$ that contains r); but then we would define the prefix $Q_\star$ associated with $P_\star$ to be a path of C_0 between r and $a_\star$; note that $Q_\star$ would contain v_3 (since C_0 has only two bridges incident to v_2), hence, $Q_\star$ would have ≥ 3 bridges and we would have $\text{cost}(Q_\star) \geq 1$; this would contradict our choice of P, a_0, z_0.

(ii) Now, we define the prefix $Q_\star$ associated with $P_\star$ to be a path of C_0 between v_1 and $a_\star$. Note that $Q_\star$ contains v_3 (since C_0 has only two bridges incident to v_2). We have two

cases: either v_3 is a black node or it is a white node; in the first case, $Q_\star$ contains a unit-bridge incident to v_3 and we have $\text{cost}(Q_\star) \geq 2$, whereas in the second case, $Q_\star$ contains the white node v_3. Hence, when we add the ear $P_\star$, then either we get ≥ 0.75 credits from a unit-bridge of $Q_\star$ incident to v_3, or we get ≥ 1 credit from the 2ec-block (of C_0) that contains v_3.

(iii) There is no connected component of $H - C_0$ that is incident to both P and $P_\star$. Otherwise, suppose that some connected component $\hat{C}$ of H is incident to both P and $P_\star$. Then there is a path in $(P \cup P_\star \cup \hat{C})$ that starts at z_0 and ends at $a_\star$, and whose prefix $Q_\star \cup \{v_0v_1\}$ in C_0 either has a white node or has cost ≥ 2, contradicting our choice of P, a_0, z_0.

Since no connected component of H other than C_0 is incident to both P and $P_\star$, it follows that P and $P_\star$ have no nodes in common. Hence, we can apply Lemma 22 separately to each of $P_\star$ and P and thus pay for all-but-one of the unit-edges added by each of the two ears.

(iv) After adding the two ears, the edge v_1v_2 can be deleted from H while preserving 2-edge connectivity, by Proposition 1. To see this, observe that $(C_0 - v_1v_2) \cup P$ contains a v_1, v_2 path, and also $(C_0 - v_1v_2) \cup P_\star$ contains a v_1, v_2 path, and moreover, these two paths have no internal nodes in common (i.e., $(C_0 - v_1v_2) \cup P \cup P_\star$ contains a cycle incident to v_1 and v_2).

Summarizing, when v_2v_3 is a zero-edge, then we add the two ears $P, P_\star$ and delete the edge v_1v_2 from H; we have sufficient credits to pay for the addition of the two ears and to re-establish the credit invariant.

Proposition 24. *(i) The credit invariant holds for* $\widehat{\text{D2}}$. *(ii) Every iteration of bridge covering preserves the credit invariant.*

Proof. It is easily seen that (i) holds. Now, focus on any iteration. We use the notation of this section; in particular, we use P to denote the first ear added in an iteration; moreover, we use $\widehat{P}$ to denote the corresponding R-ear (see page 19); also, let R^{new} denote the root 2ec-block of H^{new} (assume it exists).

It is easy to verify that parts (1) and (2) of the credit invariant are preserved by every iteration. Whenever we take away credits from a unit-bridge e of C_0 (e.g., when e is in the prefix Q of $\widehat{P}$), then we retain sufficient credits for e (if e stays in H^{new} then it keeps ≥ 1 credit, otherwise, it keeps ≥ 0 credit); moreover, if we take away credit from a bridge e of C_0, then either e is contained in the 2ec-block R^{new} of H^{new} or else e is deleted from H (thus, at most one iteration can take away credit from a unit-bridge of C_0).

Consider part (3) of the credit invariant. First, let us consider the credits of R and R^{new}. Suppose that R has α credits at the start of an iteration. Then, aside from two exceptions, at the start of the next iteration, R^{new} has $\geq \alpha$ credits; the exceptions occur in cases 2 and 3(first subcase); in these two cases, R has $\alpha \geq 2.25$ credits (since ru is a zero-bridge) while R^{new} has $\geq \alpha - 0.25 \geq 2$ credits and R^{new} is guaranteed to be incident to a unit-bridge; hence, the credit invariant pertaining to R is preserved. In what follows, we ignore the second subcase of case 3 that adds the two ears $P, P_\star$, and we discuss only the single ear-augmentations; our arguments can be extended for the double ear-augmentations, but we skip those details. Consider the credits of the other (non-root) 2ec-blocks of H and H^{new}. Any 2ec-block of H that contains a node of $\widehat{P}$ is "merged" into R^{new}, hence, the credit invariant is not relevant for such 2ec-blocks of H (we argued above that R^{new} satisfies the credit invariant). Consider a 2ec-block B of H such that there is a b-path (of H)

between B and a black node of $\widehat{P}$. Then, in H^{new}, R^{new} is b-adjacent to B, hence, B is required to have ≥ 1 credit (by the credit invariant). Although we may remove credits from such 2ec-blocks (e.g., see the proof of Lemma 22, and note that we take away 0.5 credits from each of $B(s_1)$ and $B(t_1)$), we ensure that each such 2ec-block has at least one credit at the next iteration. Lastly, consider any 2ec-block B of H that is (node) disjoint from $\widehat{P}$ and is not b-adjacent to any black node of $\widehat{P}$. Clearly, B has the same credit in both H and H^{new}. It follows that the part(3) of the credit invariant is preserved in every iteration. □

This concludes the discussion of bridge covering.

Proposition 25. *At the termination of the bridge covering step, H is a bridgeless 2-edge cover and the credit invariant holds (thus, every original 2ec-block of H has ≥ 1.5 credits and every new 2ec-block of H has ≥ 2 credits).*

6 The gluing step

In this section and in Section 5, we assume that the input is a well-structured MAP instance.

In this section, we focus on the last step of the algorithm, namely, the gluing step, and analyze the details with the goal of showing that the credits in H suffice to update H to a 2-ECSS of G by adding some edges and deleting some edges (i.e., the difference between the number of edges added and the number of edges deleted in the gluing step is $\leq$ the credit of H at the start of the gluing step). The following result summarizes this section:

Proposition 26. *At the termination of the bridge-covering step, let H denote the bridgeless 2-edge cover computed by the algorithm and suppose that the credit invariant holds; let γ denote* credits(H). *Then the gluing step augments H to a 2-ECSS by adding $\leq \gamma$ unit-edges.*

It is convenient to define the following multi-graph: let $\hat{G}$ be the multi-graph obtained from G by contracting each 2ec-block B_i of H into a single node that we will denote by B_i (thus, B_i is either a 2ec-block of H or a node of $\hat{G}$). Note that the algorithm "operates" on G and never refers to $\hat{G}$; but, for our discussions and analysis, it is convenient to refer to $\hat{G}$.

At the start of the gluing step, recall that each original 2ec-block of H has ≥ 1.5 credits and each new 2ec-block of H has ≥ 2 credits. We pick any 2ec-block R_0 of H and designate it as the root R; then we apply iterations; each iteration adds to H the edges of an ear whose start node and end node are in R; some iterations may add a second ear. After each iteration, we update the notation so that R denotes the 2ec-block of the current subgraph H that contains R_0. Our plan is to keep adding ears to H until all of the nodes are in R. Thus, we terminate when H is a 2-ECSS. In the following discussion, we assume that R has zero credits, i.e., we ignore the credits available in R while paying for the edges added in ear-augmentation steps.

Consider a cycle $\hat{C}$ of $\hat{G}$ incident to R. Observe that each of the non-root nodes of $\hat{C}$ has ≥ 1.5 credits, hence, we have $\geq 1.5(|\hat{C}| - 1)$ credits available from (the nodes of) $\hat{C}$ and this is $\geq |\hat{C}|$ whenever $|\hat{C}| \geq 3$. Thus, we have enough credit to buy all the edges of $\hat{C}$ whenever $|\hat{C}| \geq 3$. Moreover, if $|\hat{C}| = 2$ and the non-root node of $\hat{C}$ has ≥ 2 credits, then we have enough credit to buy all the edges of $\hat{C}$. Thus, we have insufficient credit only when $|\hat{C}| = 2$ and the non-root node of $\hat{C}$ has exactly 1.5 credits.

In what follows, we assume that all the cycles of $\hat{G}$ incident to R have insufficient credit. Let $\hat{C} = R, B, R$ be a cycle of $\hat{G}$ that has insufficient credit; thus, the non-root node of $\hat{C}$ is denoted by B. Clearly, B is an original 2ec-block of H (otherwise, it would have 2 credits rather than 1.5 credits). Let b denote the number of original nodes of B. It can be seen that $b \in \{2, 3, 4\}$. Moreover, for $b \in \{3, 4\}$, note that B cannot have a cut node (otherwise, B would have $\geq 1 + \lceil b/2 \rceil$

unit-edges), and hence, (since B is 2NC) B must contain a spanning cycle. For $b \in \{3, 4\}$, let $Q(B)$ denote any spanning cycle of B.

The two edges of $\hat{C}$ correspond to two edges of G between B and R; let e denote one of these edges, and let v_e denote the end node of e in B. Since G is 2NC, $G - v_e$ has a path between $(B - v_e)$ and R. Each such path has all its internal nodes in $V(R) \cup (V(B) - \{v_e\})$ (by our assumption on cycles of $\hat{G}$ incident to R), hence, there exists an edge f of G between $(B - v_e)$ and R; let u_f denote the end node of f in $B - v_e$. See Figure 11.

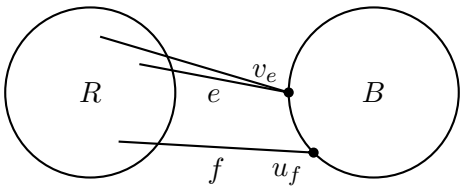


Figure 11: Illustration of R, B, and v_e, u_f.

Now, we have two cases, depending on whether v_e and u_f are adjacent in B or not.

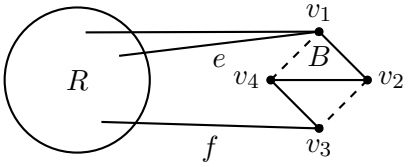
Case 1: e, f can be chosen such that B has an edge between v_e and u_f: In this case, we claim that e, f can be chosen such that B has a unit-edge between v_e and u_f. By way of contradiction, suppose that $v_e u_f$ is a zero-edge. Then we have $b \in \{3, 4\}$ (otherwise, if $b = 2$, then B would consist of two parallel unit-edges), and moreover, $G - \{v_e, u_f\}$ is connected (since $\{v_e, u_f\}$ is not a bad-pair). It can be seen that G has an edge rv_0 such that r is in R and v_0 is in $B - \{v_e, u_f\}$, and moreover, G has an edge between v_0 and $\{v_e, u_f\}$ (we skip the details). W.l.o.g. suppose that G has the edge $v_0 u_f$; this is a unit-edge (since $v_e u_f$ is a zero edge). Then by replacing the pair of edges e, f by rv_0, f, we have two edges between R and B such that their end nodes in B are distinct and there exists a unit-edge of B between these two end nodes. Our claim follows.

Now, observe that the graph $H \cup \{e, f\} - \{v_e u_f\}$ has two edge-disjoint v_e, u_f paths. We buy the edges e, f and sell the unit-edge $v_e u_f$; that is, we add the two edges e, f to H and remove the edge $v_e u_f$ from H. (In the gluing step, when we delete a unit-edge from our solution subgraph H, then one credit of that edge become available to the algorithm; note that unit-edges in $E(H) - E(\widehat{\mathrm{D2}})$ have only one credit.) In the resulting graph H^{new}, the connected component containing R_0 (as well as R and B) is 2EC, by Proposition 1. This step results in a surplus of 0.5 credits (we get 1.5 credits from B, one credit from selling $v_e u_f$, and we pay two credits for the edges e, f).

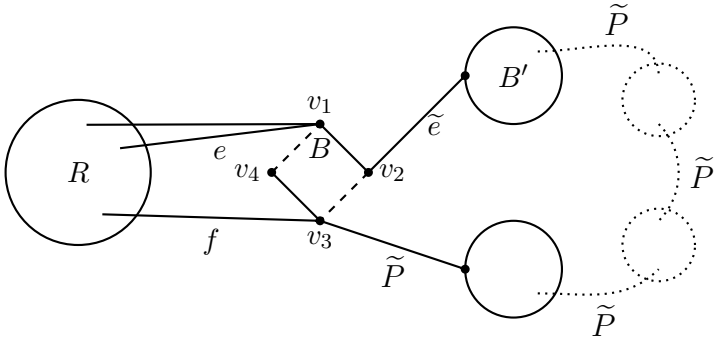
Case 2: for any choice of e, f there is no edge between v_e and u_f in B: Then, clearly $b = |V(B)| > 3$, thus $b = 4$, and B has a spanning cycle $Q(B)$. Let $Q = v_1, v_2, v_3, v_4, v_1$ denote $Q(B)$, where w.l.o.g. $v_e = v_1$ and $u_f = v_3$. Since B has 1.5 credits, two of the (non-adjacent) edges of Q must be zero-edges. There must be one or more edges incident to v_2 or v_4, otherwise, Q would be a redundant 4-cycle of G.

Suppose that G has the edge $v_2 v_4$. See Figure 12a. We buy the three edges $e, f, v_2 v_4$ and we sell the two unit-edges of Q. In the resulting graph H^{new}, the connected component containing R_0 (as well as R and B) is 2EC, by Proposition 1. This step results in a surplus of 0.5 credits (we get 1.5 credits from B, two credits from selling the two unit edges of Q, and we pay three credits for the edges $e, f,\ v_2 v_4$).

Lastly, suppose that v_2 and v_4 are nonadjacent in G. Then it can be seen that G has an edge between another 2ec-block B' of H (where $B' \neq B$ and $B' \neq R$) and one of v_2 or v_4,



(a) Illustration of R, B, and non-adjacent v_e, u_f, where B consists of a 4-cycle and a diagonal. Dashed lines indicate zero-edges.



(b) Illustration of R, B, and non-adjacent v_e, u_f, where B is a 4-cycle with two zero-edges v_2v_3, v_1v_4.

Figure 12: Illustrations for Case 2.

say v_2; let us denote this edge by $\widetilde{e}$. Since G is 2NC, there is a path in $G - \{v_2\}$ between B' and $B - \{v_2\}$. Let $\widetilde{P}$ denote such a path that has the fewest edges of $E(G) - E(H)$. In $\hat{G}$, observe that $\widetilde{P} \cup \{\widetilde{e}\}$ corresponds to a cycle $\hat{C}_{B'}$ that is incident to B and B'. Moreover, in $\hat{G}$, note that $\hat{C}_{B'}$ cannot be incident to R (by our assumption on cycles of $\hat{G}$ incident to R). See Figure 12b. Let e_Q denote the unit-edge of Q that has its end nodes among v_1, v_2, v_3. It can be seen that $H \cup \{e, f, \widetilde{e}\} \cup E(\widetilde{P}) - \{e_Q\}$ has two edge-disjoint paths between the end nodes of e_Q. We buy e, f, we sell e_Q, and moreover, we buy the edges of $\big(E(\widetilde{P}) - E(H)\big) \cup \{\widetilde{e}\}$. In the resulting graph H^{new}, the connected component containing R_0 (as well as R and B) is 2EC, by Proposition 1. It can be seen that this step results in a surplus of credits; note that the sum of the credits of the 2ec-blocks (excluding B) incident to $\widetilde{P}$ minus the size of $\big(E(\widetilde{P}) - E(H)\big) \cup \{\widetilde{e}\}$ is ≥ -0.5.

7 Examples showing lower bounds

This section presents two examples that give lower bounds on our results on MAP. The first example gives a construction such that $opt \approx \frac{7}{4}\tau$. This shows that Theorem 6 is essentially tight. The second example gives a construction such that the cost of the solution computed by our algorithm is $\approx \frac{7}{4} opt$.

7.1 Optimal solution versus min-cost 2-edge cover

Proposition 27. *For any $k \in \mathbb{N}$, there exists a MAP instance G_k such that $\tau(G_k) \leq 4k + 3$ and $opt(G_k) \geq 7k + 3$.*

Proof. The graph G_k consists of a root 2-ec block B_0 and k copies $J_1, \ldots, J_k$ of a gadget subgraph J. The gadget subgraph J consists of 8 nodes $v_1, \ldots, v_8$ and 11 edges; there are four zero-edges $v_1v_4, v_2v_3, v_5v_8, v_6v_7$, and seven unit-edges $v_1v_2, v_1v_7, v_2v_5, v_3v_4, v_3v_8, v_5v_6, v_7v_8$; see the subgraph induced by the nodes $v_1, \ldots, v_8$ in Figure 13; observe that 8 of the 11 edges form two 4-cycles (namely, v_1, v_2, v_3, v_4, v_1 and v_5, v_6, v_7, v_8, v_5) and the other three edges are v_2v_5, v_3v_8, and v_1v_7.

Let B_0 be a 6-cycle $w_1, \ldots, w_6, w_1$ that has 3 unit-edges and 3 zero-edges.

G_k has two unit-edges between each copy of the gadget subgraph J_i ($i = 1, \ldots, k$) and B_0; these two edges are incident to the nodes v_1 and v_3 of J_i (see the illustration in Figure 13) and to the nodes w_1 and w_4 of B_0. Observe that the subgraph of G_k consisting of B_0 and the two 4-cycles (namely, v_1, v_2, v_3, v_4, v_1 and v_5, v_6, v_7, v_8, v_5) of each copy of the gadget subgraph is a (feasible) 2-edge cover of G_k of cost $4k + 3$. Hence, $\tau(G_k) \leq 4k + 3$.

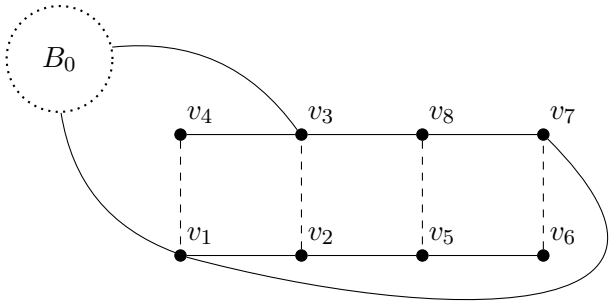


Figure 13: Example graph (with $k = 1$ copy of the 8-node gadget) where $opt \geq 7k + \epsilon$ and $\tau \leq 4k + \epsilon$, where ϵ is a constant. Edges of cost zero and one are illustrated by dashed and solid lines, respectively. B_0 is the root 2ec-block; ϵ is the cost of an optimal solution on B_0.

Finally, we claim that $opt(G_k) = \text{cost}(OPT(G_k)) \geq 7k + 3$. In what follows, we use OPT to denote an optimal solution of G_k. Clearly, OPT has to contain all the edges of B_0 as well as the two edges between B_0 and each copy of the gadget subgraph. Now, we focus on one copy J_i of the gadget subgraph, and let $opt(G_k, J_i)$ denote the cost of the edges of OPT incident to J_i. We will show that $opt(G_k, J_i) \geq 7$, hence, it follows that $opt(G_k) = 3 + \sum_{i=1}^{k} opt(G_k, J_i) \geq 7k + 3$. Since $\deg(v_4) = \deg(v_6) = 2$, OPT must pick the edges v_1v_4 and v_3v_4 as well as the edges v_5v_6 and v_6v_7. Consider the cut $\delta_{G_k}(\{v_5, v_6, v_7, v_8\})$. This cut has three unit-edges: v_1v_7, v_2v_5, v_3v_8. We have the following cases:
Case 1: OPT picks all three edges of the cut. Then $opt(G_k, J_i) \geq 7$.
Case 2: OPT does not pick all three edges of the cut. Then we will show that it must pick two edges from the cut and one more unit-edge, thus giving us $opt(G_k, J_i) \geq 7$. We have three subcases:
Case 2.1: $v_1v_7 \notin OPT$: then OPT must pick v_7v_8 and the other 2 edges of the cut.
Case 2.2: $v_2v_5 \notin OPT$: then OPT must pick v_1v_2 and the other 2 edges of the cut.
Case 2.3: $v_3v_8 \notin OPT$: then OPT must pick v_7v_8 and the other 2 edges of the cut.
Hence, $opt(G_k, J_i) \geq 7$ and we have $opt(G_k) \geq 7k + 3$. This completes the proof. □

7.2 Optimal solution versus algorithm's solution

In this section we present a family of graphs G_k, $k = 1, 2, 3 \ldots$ for which the ratio of the cost of a solution obtained by our algorithm and the cost of an optimal solution approaches $\frac{7}{4}$. Let the solution subgraph found by applying our algorithm to G_k be denoted by $ALGO(G_k)$.

Proposition 28. *For any $k \in \mathbb{N}$, there exists a MAP instance G_k such that* $\text{cost}(ALGO(G_k)) \geq 7k + 6$ *and* $opt(G_k) \leq 4k + 7$.

Proof. Let J be a gadget on nodes $v_1, v_2, v_3, v_4, w_1, w_2, w_3, w_4$, with edges v_1v_4, v_2v_3, w_1w_2, and w_3w_4 of cost zero, and edges v_1v_2, v_3v_4, w_1w_4, w_2w_3, $g = v_4w_1$ and $h = v_2v_4$ of cost one, as shown in Figure 14 (where dashed and solid lines represent edges of cost zero and one, respectively).

The graph $G_k = (V_k, E_k)$ is constructed as follows. We start with a 6-cycle $B_0 = b_1, b_2, b_3, b_4, b_5, b_6, b_1$ of unit-edges, of cost 6. We place k copies $J_1, \ldots, J_k$ of the gadget J in the following manner. Let v_j^i and w_ℓ^i denote the nodes v_j and w_ℓ of J_i, and let g_i and h_i denote the edges g and h of J_i. First, we attach J_1 to B_0 by adding the two unit-edges $e_1 = b_1v_1^1$ and $f_1 = b_4v_3^1$. Then, for each $i \in \{2, \ldots, k\}$, we attach J_i to J_{i-1} by adding the two unit-edges $e_i = w_2^{i-1}v_1^i$ and $f_i = w_3^{i-1}v_3^i$. The graph G_k is illustrated in Figure 15. Observe that G_k is a well-structured MAP instance.

Note that the cost of any 2-edge cover is $\geq 4k + 6$, since it contains all edges of B_0, as well as (at least) one unit-edge incident to each of the eight nodes of each gadget. W.l.o.g, $\widehat{D2}$ consists of

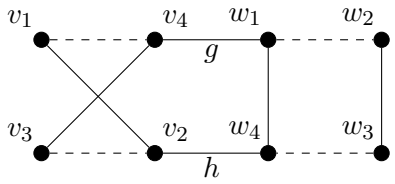


Figure 14: The gadget J. Edges of cost zero and one are illustrated by dashed and solid lines, respectively.

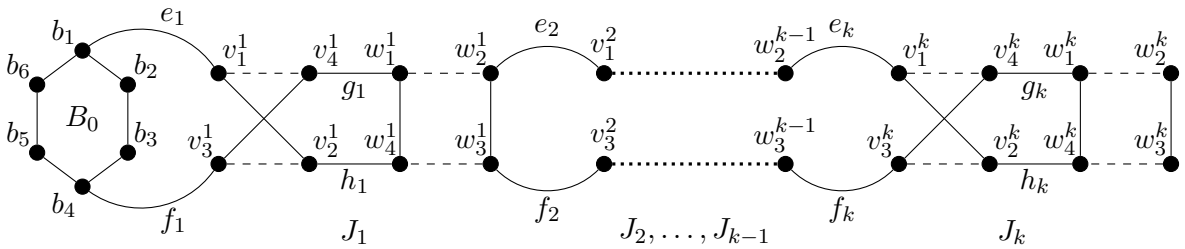


Figure 15: The graph G_k has copies $J_1, \ldots, J_k$ of the gadget. All edges have cost one, except the zero-edges of $J_1, \ldots, J_k$.

B_0 and the two 4-cycles v_1, v_2, v_3, v_4, v_1 and w_1, w_2, w_3, w_4, w_1 of each gadget. Note that our choice of $\widehat{\mathrm{D2}}$ is a bridgeless 2-edge cover.

Consider the working of the algorithm on G_k. Since $\widehat{\mathrm{D2}}$ has no bridges, the algorithm proceeds to the gluing step. We use the notation of Section 6 to describe the working of the gluing step on G_k.

In each iteration of the gluing step, we choose the 2ec-block containing B_0 to be the root 2ec-block. In the first iteration, $R = B_0$ is the root 2ec-block, and the block B is the cycle v_1, v_2, v_3, v_4, v_1 of J_1, i.e., the ear-augmentation step picks the "ear" R, B, R and takes the edges e, f (see Figure 11) to be the edges e_1, f_1 of G_k. Since the end nodes of e_1 and f_1 are non-adjacent in B, we apply case 2 of the gluing step by taking B' to be the 4-cycle w_1, w_2, w_3, w_4, w_1 of J_1 (see Figure 12b); moreover, we take $\tilde{e}$ to be the edge g_1 (of G_k), and we take $\tilde{P}$ to consist of h_1 and its two end nodes (in G_k). The algorithm buys the four edges e_1, f_1, g_1 and h_1 and sells one unit-edge (say v_3v_4), so the algorithm incurs a cost of 7 for J_1. In subsequent iterations, the same case of the gluing step is applied to each of the copies $J_2, \ldots, J_k$ of the gadget, hence, $ALGO(G_k)$ incurs a cost of 7 for each copy of the gadget; thus, we have $\mathrm{cost}(ALGO(G_k)) = 7k + 6$.

On the other hand, the subgraph G^* of G_k (described below) is a 2-ECSS of cost $4k + 7$; $E(G^*)$ consists of the union of $k + 3$ sets of edges, namely, the set of edges $\{e_i, f_i, g_i, h_i\}$ for each $i \in \{1, \ldots, k\}$, the set of zero-edges of G_k, $E(B_0)$, and the singleton $\{w_2^k w_3^k\}$ (the edge $w_2^k w_3^k$ is indicated by the right-most vertical line in Figure 15). Hence, $opt(G_k) \leq 4k + 7$. □

Acknowledgments: We are grateful to several colleagues for their careful reading of preliminary drafts and for their comments.

References

[1] D. Adjiashvili. Beating approximation factor two for weighted tree augmentation with bounded costs. In P. N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2384–2399. SIAM, 2017.

[2] R. Diestel. *Graph Theory (4th ed.)*. Graduate Texts in Mathematics, Volume 173. Springer-Verlag, Heidelberg, 2010.

[3] S. Fiorini, M. Groß, J. Könemann, and L. Sanità. A 3/2-approximation algorithm for tree augmentation via Chvátal-Gomory cuts. *CoRR*, abs/1702.05567, 2017.

[4] G. N. Frederickson and J. JáJá. Approximation algorithms for several graph augmentation problems. *SIAM J. Comput.*, 10(2):270–283, 1981.

[5] G. N. Frederickson and J. JáJá. On the relationship between the biconnectivity augmentation and traveling salesman problems. *Theor. Comput. Sci.*, 19:189–201, 1982.

[6] H. N. Gabow, M. X. Goemans, É. Tardos, and D. P. Williamson. Approximating the smallest k-edge connected spanning subgraph by LP-rounding. *Networks*, 53(4):345–357, 2009.

[7] G.Even, J.Feldman, G.Kortsarz, and Z.Nutov. A 1.8 approximation algorithm for augmenting edge-connectivity of a graph from 1 to 2. *ACM Trans. Algorithms*, 5(2):21:1–17, 2009.

[8] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995.

[9] H.Nagamochi. An approximation for finding a smallest 2-edge connected subgraph containing a specified spanning tree. *Discrete Applied Mathematics*, 126:83–113, 2003.

[10] K. Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[11] M. Karpinski, M. Lampis, and R. Schmied. New inapproximability bounds for TSP. *J. Comput. Syst. Sci.*, 81(8):1665–1677, 2015.

[12] G. Kortsarz and Z. Nutov. A simplified 1.5-approximation algorithm for augmenting edge-connectivity of a graph from 1 to 2. *ACM Trans. Algorithms*, 12(2):23:1–20, 2016.

[13] L. C. Lau, R. Ravi, and M. Singh. *Iterative Methods in Combinatorial Optimization*. Cambridge Texts in Applied Mathematics (No. 46). Cambridge University Press, 2011.

[14] L. Lovàsz and M. D. Plummer. *Matching Theory*, volume 367 of *AMS/Chelsea Publishing*. American Mathematical Society, 2009.

[15] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Algorithms and Combinatorics, Volume 24. Springer-Verlag, Berlin Heidelberg, 2003.

[16] A. Sebö and J. 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.

[17] S. Vempala and A. Vetta. Factor 4/3 approximations for minimum 2-connected subgraphs. In K. Jansen and S. 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*, pages 262–273. Springer, 2000.

[18] H. Whitney. Non-separable and planar graphs. *Trans. Amer. Math. Soc.*, 34:339–362, 1932.

[19] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.