An Improved Approximation Algorithm for the Matching Augmentation Problem

Joseph Cheriyan
Comb. & Opt. Dept., University of Waterloo, Canada

Robert Cummings
Comb. & Opt. Dept., University of Waterloo, Canada

Jack Dippel
Mathematics & Statistics, McGill University, Montreal, Canada

Jasper Zhu
Comb. & Opt. Dept., University of Waterloo, Canada

Abstract

We present a $\frac{5}{3}$-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.

A $\frac{7}{4}$-approximation algorithm for the same problem was presented recently, see Cheriyan, et al., "The matching augmentation problem: a $\frac{7}{4}$-approximation algorithm," *Math. Program.*, 182(1):315–354, 2020.

Our improvement is based on new algorithmic techniques, and some of these may lead to advances on related problems.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases 2-Edge connected graph, 2-edge covers, approximation algorithms, connectivity augmentation, forest augmentation problem, matching augmentation problem, network design

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2021.38

Related Version *Full Version*: `https://arxiv.org/abs/2007.11559`

Funding *Joseph Cheriyan*: NSERC, RGPIN-2019-04197

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

1 Introduction

The design and analysis of algorithms for problems in network design is a core topic in Theoretical Computer Science and Combinatorial Optimization. Algorithmic research on problems such as the minimum spanning tree problem and the Traveling Salesman Problem (TSP) started decades ago and is a thriving area even today. One of the key problems in this area is the minimum-cost 2-ECSS (2-edge connected spanning subgraph) problem: Given an undirected graph $G = (V, E)$ and a nonnegative cost for each edge $e \in E$, denoted cost(e), find a minimum-cost spanning subgraph $H = (V, F)$, $F \subseteq E$, that is 2-edge connected. Throughout, we use $n := |V|$ to denote the number of nodes of G. (Recall that a graph is 2-edge connected if it is connected and has no "cut edges", or equivalently, each of its nontrivial cuts has ≥ 2 edges.) This problem is NP-hard, and the best approximation guarantee known, due to [22], is 2.

On the other hand, the best "hardness of approximation threshold" known is much smaller; for example, it is $(1 + \frac{\rho_{VC3}}{104})$ for the unweighted problem, where $1 + \rho_{VC3}$ is the "hardness of approximation threshold" for the minimum vertex cover problem on a graph with

© Joseph Cheriyan, Robert Cummings, Jack Dippel, and Jasper Zhu;
licensed under Creative Commons License CC-BY 4.0

maximum degree 3, [10, Theorem 5.2]. Also, the best lower bound known on the integrality ratio of the standard LP relaxation (for minimum-cost 2-ECSS) is around 1.5 (thus, well below 2), see [5].

1.1 FAP, TAP and MAP

Given this significant gap between the lower bounds and the upper bounds, research in this area over the last two decades has focused on the case of zero-one cost functions (every edge has a cost of zero or one). Let us call an edge $e \in E$ with $\mathrm{cost}(e) = 0$ a zero-edge, and let us call an edge $e \in E$ with $\mathrm{cost}(e) = 1$ a unit-edge. Intuitively, the zero-edges define some existing network that we wish to augment (with unit-edges) such that the augmented network is resilient to the failure of any one edge. We may assume that the zero-edges form a forest; otherwise, there is at least one cycle C formed by the zero edges, and in that case, we may contract C, solve the problem on the resulting graph G/C, find a solution (edge set) F, and return $F \cup C$ as a solution of the original problem. Consequently, the minimum-cost 2-ECSS problem with a zero-one cost function is called the *Forest Augmentation Problem* or FAP. The challenge is to design an approximation algorithm with guarantee strictly less than 2 for FAP.

A well known special case of FAP is TAP, the *Tree Augmentation Problem*: the set of zero-edges forms a spanning tree. The first publication to break the "2-approximation barrier" for TAP is [14] (2003), and since then there have been several important advances, including recent work, see [11, 17, 1, 19, 4, 9, 13].

Recently, see [2], there has been progress on another important (in our opinion) special case of FAP called the *Matching Augmentation Problem* or MAP: Given a multi-graph with edges of cost either zero or one such that the zero-edges form a matching, find a 2-ECSS of minimum cost. From the view-point of approximation algorithms, MAP is "complementary" to TAP, in the sense that the forest formed on $V(G)$ by the zero-edges has many connected components, each with one node or two nodes, whereas this forest has only one connected component in TAP.

1.2 Previous literature and possible approaches for attacking MAP

There is a large body of work on network design and the design of algorithms (for finding optimal solutions, as well as for finding approximately optimal solutions); see the books in the area [20, 25, 18]. Unfortunately, none of these results and methods has helped to break the "2-approximation barrier" for FAP (to the best of our knowledge).

Powerful and versatile methods such as the *primal-dual method* (see [25, 12]) and the *iterative rounding method* (see [18, 16]) have been developed for problems in network design, but the proveable approximation guarantees for these methods are ≥ 2. (These methods work by rounding LP relaxations; informally speaking, the approximation guarantee is proved via an upper bound of 2 per iteration on the "integral cost incurred" versus the "chargeable LP cost", and it is plausible that the factor of 2 cannot be improved for this type of analysis.)

Combinatorial methods that may also exploit lower-bounds from LP relaxations have been developed for approximation algorithms for unweighted minimum-cost 2-ECSS, e.g., $\frac{4}{3}$-approximation algorithms are presented in [24, 21, 15]. For the unweighted problem, there is a key lower bound of n on *opt* (since any solution must have $\geq n$ edges, each of cost one). This fails to holds for MAP; indeed, the analogous lower bound on *opt* is $\frac{1}{2}n$ for MAP. This rules out any direct extension of these combinatorial methods (for the unweighted problem) to prove approximation guarantees below 2 for MAP.

Recently, Traub and Zenklusen [23] presented an approximation algorithm for Weighted TAP with guarantee $1 + (\ln 2) + \epsilon < 1.7$, via a so-called relative greedy algorithm, based on previous work by Cohen and Nutov [6] that, in turn, is based on earlier work by Zelikovsky on the Steiner Tree problem [26]. While the result of Traub and Zenklusen is a major advance in the area, the core ideas of the relative greedy algorithm are well known, and, as yet, there is no advance on FAP via relative greedy algorithms. (Informally speaking, Weighted TAP is a special case of SCP (the Set Covering Problem), and the method of Cohen and Nutov exploits special properties of the SCP instances associated with TAP, whereas, to the best of our knowledge, this paradigm does not extend to FAP.)

1.3 Our results and techniques

Our main contribution is a $\frac{5}{3}$-approximation algorithm for MAP, improving on the $\frac{7}{4}$ approximation guarantee of [2], see Theorems 12, 13.

At a high level (hiding many important points), our algorithm is based on a "discharging scheme" where we compute a lower bound on *opt* (the optimal value) and fix a "budget" of α times this lower bound (where $\alpha > 1$ is a constant), "scatter" this budget over the graph G, use the budget to buy some edges to obtain a "base graph", then traverse the "base graph" and buy more edges to augment the "base graph", so that (eventually) we have a 2-ECSS whose cost is within the budget of α times our lower bound. We mention that several of the results cited above are based on discharging schemes, e.g., [24, 11, 17, 15, 2]. In some more detail, but still at a high level, we follow the method of [2]. (Our presentation can be read independently of [2]; we have repeated a few definitions and statements of results from [2].) We first pre-process the input instance G, with the goal of removing all "obstructions" (e.g., cut nodes), and we decompose G into a list of "well structured" sub-instances $G_1, G_2, \ldots$ that are pairwise edge-disjoint. Now, consider one of these sub-instances G_i (it has none of the "obstructions"). We compute a subgraph H_i whose cost is a lower bound on $opt(G_i)$. Finally, we augment H_i to make it 2-edge connected, and use a credit-based analysis to prove an approximation guarantee.

A 2-edge cover is a subgraph that has at least two edges incident to every node. The minimum-cost 2-edge cover is the key subgraph used as a lower bound in our algorithm; we refer to it as D2. (D2 can be computed in polynomial time via extensions of Edmonds' algorithm for computing a minimum-cost perfect matching.) Since every 2-ECSS is a 2-edge cover, we have $\text{cost}(\text{D2}) \leq opt$. So, by transforming D2 to a 2-ECSS of cost $\leq \frac{5}{3}\text{cost}(\text{D2})$, we achieve our claimed approximation guarantee.

Our pre-processing includes several new ideas, and moreover, it is essential to handle new "obstructions" that are not handled in [2]; indeed, [2] has tight examples such that $opt/\text{cost}(\text{D2}) \geq \frac{7}{4} - \epsilon$ (for some $\epsilon > 0$). Although our algorithm handles several new "obstructions", our analysis and proofs for the pre-processing are simple. One of our key tools (for our pre-processing analysis) is to prove a stronger guarantee of $\max(opt, \frac{5}{3}opt - 2)$ rather than just $\frac{5}{3}opt$. When we analyze our decomposition of an instance into sub-instance(s), then this additive term of -2 is useful in combining solutions back together at the end of the algorithm (when we "undo" the decomposition of G into sub-instances $G_1, G_2, \ldots$). (Our analysis of pre-processing is omitted in this paper, due to space constraints; it is given in the full paper, [3].)

Our main algorithm (following [2]) has two key subroutines for transforming a D2 of a "well structured" sub-instance G_i to a 2-ECSS of G_i while ensuring that the total cost is $\leq \frac{5}{3}\text{cost}(\text{D2})$.

(i) **Bridge covering step**: The goal is to augment edges such that each connected component of our "current solution graph" H_i is 2-edge-connected; we start with $H_i := \text{D2}(G_i)$. Our analysis is a based on a new and simple credit scheme that bypasses some difficulties in the credit scheme of [2].

(ii) **Gluing step**: Finally, this step merges the (already 2-edge connected) connected components of H_i to form a 2-ECSS of the sub-instance G_i. A key part of this step handles so-called "small 2ec-blocks"; these are cycles of cost 2 that occur as connected components of $\text{D2}(G_i)$ and stay unchanged through the bridge covering step. Observe that a "small 2ec-block" has only $\frac{4}{3}$ credits (it has a "budget" of $(\frac{5}{3})(2)$, and after paying for its two unit-edges, there is only $\frac{4}{3}$ credits available). Our gluing step applies a careful swapping of unit-edges for the "small 2ec-blocks" while it merges the connected components of H_i into a 2-ECSS, and ensures that the net augmentation cost does not exceed the available credit.

There are well-known polynomial time algorithms for implementing all of the basic computations in this paper, see [20]. We state this explicitly in all relevant results (e.g., Theorem 12), but we do not elaborate on this elsewhere.

2 Preliminaries

This section has definitions and preliminary results. Our notation and terms are consistent with [7], 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. Throughout, the reader should keep in mind that M is a matching; this fact is used in many of our proofs without explicit reminders. We call an edge of M a *zero-edge* and we call an edge of $E - M$ a *unit-edge*.

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, $\text{cost}(G') := \sum_{e \in E(G')} \text{cost}(e)$.

For ease of exposition, we often denote an instance G, M by G; then, we do not have explicit notation for the edge costs of the instance, but the edge costs are given implicitly by $\text{cost} : E(G) \to \{0, 1\}$, and M is given implicitly by $\{e \in E(G) : \text{cost}(e) = 0\}$.

For a positive integer k, we use $[k]$ to denote the set $\{1, \ldots, k\}$.

We use the standard notion of contraction of an edge, see [20, p.25]. For a graph H and a set of its nodes S, $\Gamma_H(S) := \{w \in V(H) - S \,:\, v \in S, vw \in E(H)\}$, thus, $\Gamma_H(S)$ denotes the set of neighbours of S. 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 the subgraph of H induced by S, and $H - S$ denotes 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 $(V(H),\ E(H) - F)$. For any subgraph K of a graph H with $V(K) \subsetneq V(H)$, an *attachment* of K is a node of K that has a neighbour in $V(H) - V(K)$.

We may use relaxed notation for singleton sets, and, we may not distinguish between a subgraph and its node set; for example, given a graph H and a set S of its nodes, we use $E(S)$ to denote the edge set of the subgraph of H induced by S.

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 2EC. Moreover, for some (but not all) of our discussions, we assume that there are ≤ 2 copies of each edge (in the multi-graph under consideration); this is justified since an edge-minimal 2-ECSS cannot have three or more copies of any edge (see Proposition 1 below).

For any instance H, 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 $opt(H)$.

By a *bridge* we mean 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. We call a 2ec-block *pendant* if it is incident to exactly one bridge. We call a 2ec-block *small* if it has ≤ 2 unit-edges, and we call it *large* otherwise.

For a 2EC graph G and a cut node v of G, a 2ec-v-block means the subgraph of G induced by $\{v\} \cup V(C)$ where C is one of the connected components of $G - v$.

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.*

The next lemma partially characterizes the cuts of size ≤ 2 in a graph obtained by "uncontracting" a set of nodes of a 2EC graph.

▶ **Lemma 2.** *Let H be a 2EC graph and let $C \subsetneq V(H)$ be a set of nodes such that the induced subgraph $H[C]$ is connected. Suppose that H^* is a 2-ECSS of H/C. Let H' be the spanning subgraph of H with edge set $E(C) \cup E(H^*)$. Then H' is a connected graph such that each of its bridges (if any) is in $E(C)$.*

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); when there is no danger of ambiguity, we use D2 rather than D2(G).

By a *bridgeless 2-edge cover* (of G) we mean a 2-edge cover (of G) that has no bridges.

The next result follows from Theorem 34.15 in [20, Chapter 34].

▶ **Proposition 3.** *There is a polynomial-time algorithm for computing* D2*.*

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

▶ **Lemma 4.** *Let H be any 2EC graph. Then we have $opt(H) \geq \text{cost}(\text{D2}(H))$.*

For any fixed positive integer z (thus, $z = O(1)$) and any instance of MAP, in time $O(1)$, we can determine whether the instance has $opt > z$, and if not, then we can find an optimal 2-ECSS of the instance.

▶ **Lemma 5.** *Let H be an instance of MAP, and let z be a fixed positive integer. There is an $O(1)$-time algorithm to determine whether $\mathrm{opt}(H) \geq z$. Moreover, if $\mathrm{opt}(H) \leq z$, then a minimum-cost 2-ECSS of H can be found in $O(1)$ time.*

2.2 Obstructions for the approximation guarantee

There are several obstructions (e.g., cut nodes) that prevent our algorithm (and analysis) from achieving our target approximation factor of $\frac{5}{3}$. We eliminate all such obstructions in a pre-processing step that takes the given instance G of MAP (the input) and replaces it by a list of sub-instances $G_1, G_2, \ldots$, such that (a) none of the obstructions occurs in a sub-instance G_i, (b) the edge-sets of the sub-instances are pairwise-disjoint, and (c) given a 2-ECSS of each sub-instance G_i of approximately optimal cost, we can construct a 2-ECSS of G of cost $\leq \frac{5}{3}opt(G)$. (Precise statements are given later.) The obstructions for our algorithm are:

(i) cut nodes,
(ii) parallel edges,
(iii) zero-cost S2,
(iv) unit-cost S2,
(v) S$\{3,4\}$,
(vi) R4,
(vii) R8.

Below, we formally define each of these obstructions. Four of these obstructions were introduced in [2], and readers interested in a deeper understanding may refer to that paper, in particular, see the remark after [2, Theorem 6] and see [2, Figure 2] for instances G of MAP that contain cut nodes, parallel edges, zero-cost S2s, or R4s such that $opt(G)/\mathrm{cost}(\mathrm{D2}(G)) \approx 2$; informally speaking, an approximation algorithm based on the lower bound $\mathrm{cost}(\mathrm{D2}(G))$ on $opt(G)$ fails to beat the approximation threshold of 2 in the presence of any of these four obstructions.

▶ **Definition 6.** *By a* zero-cost S2 *(also called a bad-pair), we mean a zero-edge e and its end nodes, u, v, such that $G - \{u, v\}$ has ≥ 2 connected components.*

▶ **Definition 7.** *By a* unit-cost S2*, we mean a unit-edge e and its end nodes, u, v, such that $G - \{u, v\}$ has ≥ 2 connected components; moreover, in the graph $G/\{u, v\}$, there exist two distinct 2ec-$\hat{v}$-blocks B_1, B_2 incident to the contracted node $\hat{v}$ such that $\mathrm{opt}(B_i) \geq 3$ and B_i has a zero-edge incident to the contracted node, $\forall i \in [2]$.*

▶ **Definition 8.** *By an* S$\{3,4\}$*, we mean an induced 2NC subgraph C of G with $|V(C)| \in \{3,4\}$ that has a spanning cycle of cost two such that $G - V(C)$ has ≥ 2 connected components, and the cut $\delta(V(C))$ has no zero-edges; moreover, in the graph G/C, there exist two distinct 2ec-$\hat{v}$-blocks B_1, B_2 incident to the contracted node $\hat{v}$ that have $\mathrm{opt}(B_1) \geq 3$ and $\mathrm{opt}(B_2) \geq 3$.*

▶ **Definition 9.** *By an* R4 *(also called a redundant 4-cycle), we mean an induced subgraph C of G with four nodes such that $V(C) \neq V(G)$, C contains a 4-cycle of cost two, and C contains a pair of nonadjacent nodes that each have degree two in G.*

▶ **Definition 10.** *By an* R8*, we mean an induced subgraph C of G with eight nodes such that $V(C) \neq V(G)$, C contains two disjoint 4-cycles C_1, C_2 with $\mathrm{cost}(C_i) = 2, \forall i \in [2]$, C has exactly two attachments a_1, a_2 where $a_i \in C_i, \forall i \in [2]$, and both end nodes of the (unique) unit-edge of $C_i - a_i$ are adjacent to C_{3-i}, $\forall i \in [2]$.*

3 Outline of the algorithm

This section has an outline of our algorithm. We start by defining an instance of MAP$\star$.

▶ **Definition 11.** *An instance of MAP$\star$ is an instance of MAP with ≥ 12 nodes that contains*

- *no cut nodes,*
- *no parallel edges,*
- *no* zero-cost S2*,*
- *no* unit-cost S2*,*
- *no* S$\{3,4\}$*,*
- *no* R4*, and*
- *no* R8.

In this section and Section 4, we sketch how to "decompose" any instance of MAP G with $|V(G)| \geq 12$ into a collection of instances $G_1, \ldots, G_k$ of MAP such that (a) either $|V(G_i)| < 12$ or G_i is an instance of MAP$\star$, $\forall i \in [k]$, (b) the edge sets $E(G_1), \ldots, E(G_k)$ are pairwise disjoint (thus $E(G_1), \ldots, E(G_k)$ forms a subpartition of $E(G)$), and (c) a 2-ECSS H of G can be obtained by computing 2-ECSSes $H_1, \ldots, H_k$ of $G_1, \ldots, G_k$. Moreover, the approximation guarantee is preserved, meaning that $\text{cost}(H) \leq \frac{5}{3}opt(G) - 2$ provided $\text{cost}(H_i) \leq \max(opt(G_i), \frac{5}{3}opt(G_i) - 2), \forall i \in [k]$.

Algorithm (outline).

```
(0)   apply the pre-processing steps (see below and see Section 4) to obtain a collection
      of instances $G_1, \ldots, G_k$ such that either $|V(G_i)| < 12$ or $G_i$ is an instance of MAP$\star$,
      $\forall i \in [k]$;
      for each $G_i$ ($i = 1, \ldots, k$),
      if $|V(G_i)| < 12$
(1)   exhaustively compute an optimum 2-ECSS $H_i$ of $G_i$ via Lemma 5;
      else
(2.1) compute D2($G_i$) in polynomial time (w.l.o.g. assume D2($G_i$) contains all zero-edges
      of $G_i$);
(2.2) then apply "bridge covering" from Section 5 to D2($G_i$) to obtain a bridgeless
      2-edge cover $\widetilde{H}_i$ of $G_i$;
(2.3) then apply the "gluing step" from Section 6 to $\widetilde{H}_i$ to obtain a 2-ECSS $H_i$ of $G_i$;
      endif;
      endfor;
(3)   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 several reductions; most of these reductions are straightforward, but we have to prove that the approximation guarantee is preserved when we "undo" each of these reductions. These proofs are given in the full paper, [3].

Pre-processing – Step (0) of Algorithm.

```
    While the current list of sub-instances $G_1, G_2, \ldots$ has a sub-instance $G_i$ that has $\geq 12$
    nodes and is not an instance of MAP$\star$ (assume that $G_i$ is 2EC):

    if $G_i$ is not 2NC:
(i) (handle a cut-node)
    let $v$ be a cut node of $G_i$, and let $B_1, \ldots, B_k$ be the 2ec-$v$-blocks of $G_i$; replace $G_i$ by
    $B_1, \ldots, B_k$ in the current list;
    else apply exactly one of the following steps to $G_i$:
```

- **(ii)** (handle a pair of parallel edges)
  let $\{e, f\}$ be a pair of parallel edges of G_i (one of the edges in $\{e, f\}$ is a unit-edge); discard a unit-edge of $\{e, f\}$ from G_i;
- **(iii)** (handle an "S obstruction")
  - **(a)** (handle a unit-cost S2)
  - **(b)** (handle a zero-cost S2)
  - **(c)** (handle an S$\{3, 4\}$)

  let C denote a subgraph of G_i that is, respectively, (a) a unit-cost S2, (b) a zero-cost S2, or (c) an S$\{3, 4\}$;
  contract C to obtain G_i/C and let $\hat{v}$ denote the contracted node; let $B_1, \ldots, B_k$ be the 2ec-$\hat{v}$-blocks of G_i/C; replace G_i by $B_1, \ldots, B_k$ in the current list;
- **(iv)** (handle an "R obstruction")
  - **(a)** (handle an R4)
  - **(b)** (handle an R8)

  let C denote a subgraph of G_i that is, respectively, (a) an R4, or (b) an R8;
  contract C to obtain G_i/C, and replace G_i by G_i/C in the current list;

Our $\frac{5}{3}$ approximation algorithm for MAP follows from the following theorem.

▶ **Theorem 12.** *Given an instance of MAP⋆ G', there is a polynomial-time algorithm that obtains a 2-ECSS H' such that* $\text{cost}(H') \leq \max(opt(G'),\ \frac{5}{3}opt(G') - 2)$.

We use a credit scheme to prove this theorem; the details are presented in Sections 5 and 6. The algorithm starts with D2(G') as the current graph, and assigns $\frac{5}{3}$ tokens to each unit-edge of D2(G'); each such edge keeps one unit to pay for itself and the other $\frac{2}{3}$ is taken to be credit of the edge; thus, the algorithm has $\frac{2}{3}\text{cost}(\text{D2}(G'))$ credits at the start; the algorithm uses the credits to pay for the augmenting edges "bought" in steps (2.2) or (2.3) (see the outline); also, the algorithm may "sell" unit-edges of the current graph (i.e., such an edge is permanently discarded and is not contained in the 2-ECSS output by the algorithm).

The factor $\frac{5}{3}$ in our approximation guarantee is tight in the sense that there exists an instance G of MAP⋆ such that $opt(G)/\text{cost}(\text{D2}(G)) \geq \frac{5}{3} - \epsilon$, for any small positive number ϵ. The instance G consists of a root 2ec-block B_0, say a 6-cycle of cost 6, $v_1, \ldots, v_6, v_1$, and $\ell \gg 1$ copies of the following gadget that are attached to B_0. The gadget consists of a 6-cycle $C = u_1, \ldots, u_6, u_1$ of cost 3 that has alternating zero-edges and unit-edges; moreover, there are three unit-edges between C and B_0: v_1u_1, v_3u_3, v_5u_5. Observe that a (feasible) 2-edge cover of this instance consists of B_0 and the 6-cycle C of each copy of the gadget, and it has cost $6 + 3\ell$. Observe that for any 2-ECSS and for each copy of the gadget, the six edges of C as well as (at least) two of the edges between C and B_0 are contained in the 2-ECSS. Thus, $opt(G) \geq 6 + 5\ell$, whereas $\text{cost}(\text{D2}(G)) \leq 6 + 3\ell$.

4 Pre-processing

Due to space constraints, we only the state our main result for pre-processing and skip several lemmas that are needed to prove this result. These lemmas and their proofs are given in the full paper, [3].

▶ **Theorem 13.** *Suppose that there is an approximation algorithm that given an instance H of MAP⋆, finds a 2-ECSS of cost $\leq$* $\max(opt(H),\ \alpha\ opt(H) - 2)$. *Then, given an instance G of MAP, there is a polynomial-time algorithm to find a 2-ECSS of cost $\leq$* $\max(opt(G),\ \alpha\ opt(G) - 2)$.

5 Bridge covering

The results in this section are based on the prior results and methods of [2, 8], but the goal in these previous papers is to obtain an approximation guarantee of $\frac{7}{4}$ for MAP, whereas our goal is an approximation guarantee of $\frac{5}{3}$. Our credit invariant is presented in Section 5.1 below, and it is based on the credit invariant in [8].

In this section and in Section 6, we assume that the input is an instance of MAP$\star$. For notational convenience, we denote the input by G. Recall that G is a simple, 2NC graph on ≥ 12 nodes, and G has no zero-cost S2, no unit-cost S2, no S$\{3,4\}$, no R4, and no R8. Recall that a 2ec-block is called small if it has ≤ 2 unit-edges, and is called large otherwise. Since G is 2NC and simple, a small 2ec-block is either a 3-cycle with one zero-edge and two unit-edges, or a 4-cycle with alternating zero-edges and unit-edges.

Each unit-edge e of D2 starts with $\frac{5}{3}$ tokens, and from this, one unit is kept aside (to pay for e), and the other $\frac{2}{3}$ is defined to be the credit of e. Our overall goal is to find a 2-ECSS H' of G of cost $\leq \frac{5}{3}$cost(D2), and we keep $\frac{2}{3}$cost(D2) from our budget in the form of credit while using the rest of our budget for "buying" the unit-edges of D2. We use the credit for "buying" unit-edges that are added to our current graph during the bridge covering step or the gluing step. (In the gluing step, we may "sell" unit-edges of our current graph, that is, we may permanently discard some unit-edges of our current graph; thus, our overall budgeting scheme does not rely solely on credits.)

We use H to denote the current graph of the bridge covering step; initially, $H =$ D2.

The outcome of the bridge covering step is stated in the following result.

▶ **Proposition 14.** *At the termination of the bridge covering step, H is a bridgeless 2-edge cover; moreover, every small 2ec-block of H has $\geq \frac{4}{3}$ credits and every large 2ec-block of H has ≥ 2 credits. The bridge covering step can be implemented in polynomial time.*

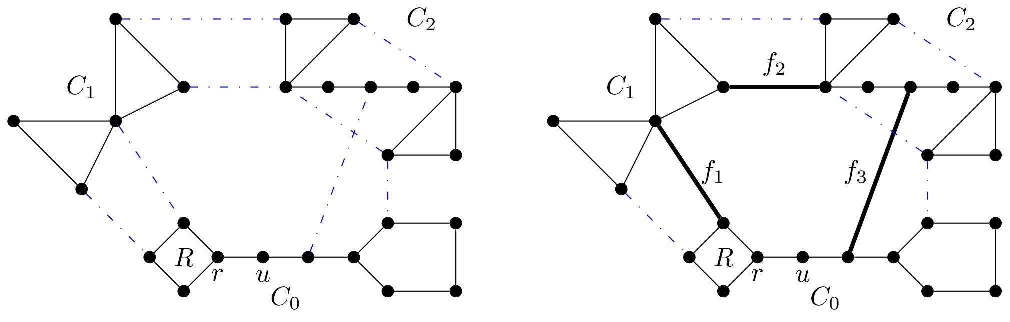

Figure 1 Illustration of an iteration of our bridge-covering step. Solid lines indicate edges of the graph H, and (blue) dash-dotted lines indicate edges of $E(G) - E(H)$. The pseudo-ear $R, f_1, C_1, f_2, C_2, f_3$ covers the bridge ru of C_0 (right subfigure). Thick lines indicate the edges f_1, f_2, f_3 of the pseudo-ear.

A brief overview of the bridge covering step follows: The goal is to add "new" edges to H to obtain a bridgeless 2-edge cover, and to pay for these "new" edges from credits available in H while preserving a credit invariant (stated below). In each iteration, we pick a connected component C_0 of H such that C_0 has a bridge, then we pick any pendant 2ec-block R of C_0, then we add a set of edges $\{f_1, \ldots, f_k\} \subseteq E(G) - E(H)$ that "covers" the unique bridge of C_0 incident to R (possibly, $k = 1$). Informally speaking, this step merges $k-1$ connected components $C_1, C_2, \ldots, C_{k-1}$ of H with C_0 (see the discussion below). Each connected component of H has one unit of so-called c-credit (by the credit invariant stated below), and we take this credit from each of $C_1, C_2, \ldots, C_{k-1}$ and use that to pay for $k-1$ of the newly added edges. The challenge is to find one more unit of credit (since we added k edges), and this is the focus of our analysis given below.

By [2, Section 5.1,Proposition 5.20], we may assume without loss of generality that D2 has the following properties:

(*) D2 contains all the zero-edges. Every pendant 2ec-block of D2 that is incident to a zero-bridge is a large 2ec-block.

Recall that H denotes the current graph, and initially, $H = \mathrm{D2}$. 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.

It is convenient to define the following multi-graphs: let $\widetilde{H}$ be the multi-graph obtained from H by contracting each 2ec-block B_i of H into a single node that we will denote by B_i. Observe that each connected component of $\widetilde{H}$ is a tree (possibly, an isolated node). We call a node v of the multigraph $\widetilde{H}$ black if it is the image of a black node of H, otherwise, we call v a white node. Each 2ec-block of H maps to a white node of $\widetilde{H}$. Each bridge of H maps to a bridge of $\widetilde{H}$. Clearly, each black node of $\widetilde{H}$ is incident to ≥ 2 bridges of $\widetilde{H}$.

Similarly, let $\widetilde{G}$ be the multi-graph obtained from G by contracting each 2ec-block B_i of H into a single node.

5.1 Credit invariant

We re-assign the credits of D2 such that the following credit invariant holds for H at the start/end of every iteration in the bridge covering step.

For a black node v of H, we use $\deg_H^{(1)}(v)$ to denote the number of unit-bridges incident to v in H.

Credit invariant for H

(a) each connected component is assigned at least one credit (called c-credit);

(b) each connected component that is a small 2ec-block is assigned $\frac{1}{3}$ credits (called b-credit);

(c) every other 2ec-block is assigned at least one credit (called b-credit);

(d) each black node v is assigned $\frac{1}{3}\deg_H^{(1)}(v)$ credits (called n-credit).

Note that the four types of credit are distinct, and the invariant gives lower bounds. For example, a connected component that is a large 2ec-block has one c-credit and at least one b-credit.

▶ **Lemma 15.** *The initial credits of* D2 *can be re-assigned such that (the initial) $H = \mathrm{D2}$ satisfies the credit invariant.*

5.2 Analysis of a pseudo-ear augmentation

In this subsection, our goal is to show that a so-called pseudo-ear augmentation can be applied to H whenever a connected component of H has a bridge, such that the cost of the newly added unit-edges is paid from the credits released by the pseudo-ear augmentation, and moreover, the credit invariant is preserved.

In the graph H, let C_0 be a connected component that has a bridge, let R be a pendant 2ec-block of C_0, and let ru be the unique bridge (of C_0) incident to R, where $r \in V(R)$.

▶ **Definition 16.** *A pseudo-ear of H w.r.t. C_0 starting at R is a sequence $R, f_1, C_1, f_2, C_2, \ldots, f_{k-1}, C_{k-1}, f_k$, where $C_0, C_1, \ldots, C_{k-1}$ are distinct connected components of H, $f_1, \ldots, f_k \in E(G) - E(H)$, each f_i, $i \in [k-1]$, has one end node in C_{i-1} and the other end node in C_i, f_1 has an end node in R, and f_k has one end node in C_{k-1} and one end node in $C_0 - V(R)$. The end node of f_k in $C_0 - V(R)$ is called the head node of the pseudo-ear.*

Any shortest (w.r.t. the number of edges) path of C_0 between r and the head node of the pseudo-ear is called the witness path of the pseudo-ear.

Our plan is to find a pseudo-ear (as above) such that for any witness path Q, there is at least one unit of credit in $Q - r$. Let R^{new} denote the 2ec-block that results from the addition of the pseudo-ear; thus, R^{new} contains $R \cup Q$. The b-credit of R is transferred to R^{new}; thus, R^{new} satisfies part (c) of the credit invariant; see Proposition 19 below. After we add the pseudo-ear to H, the credits of $Q - r$ are released (they are no longer needed for preserving the credit invariant, because $Q \cup R$ is merged into R^{new}). Informally speaking, we use the credits released from $Q - r$ to pay for the cost of the last unit-edge added by the pseudo-ear augmentation.

In the graph $\widetilde{G}$, let $\widetilde{C_0}$ denote the tree corresponding to C_0 and let $\widetilde{R}$ denote the leaf of $\widetilde{C_0}$ corresponding to R. Let $\widetilde{P}$ be a shortest (w.r.t. the number of edges) path of $\widetilde{G} - E(\widetilde{C_0})$ that has one end node at $\widetilde{R}$ and the other end node at another node of $\widetilde{C_0}$. Then $\widetilde{P}$ corresponds to a pseudo-ear $R, f_1, C_1, \ldots, C_{k-1}, f_k$; the sequence of edges of $E(\widetilde{G}) - E(\widetilde{H})$ of $\widetilde{P}$ corresponds to $f_1, \ldots, f_k$ and the sequence of trees $\widetilde{C_1}, \ldots, \widetilde{C}_{k-1}$ of $\widetilde{P}$ corresponds to $C_1, \ldots, C_{k-1}$.

It is easy to find a pseudo-ear such that any witness path Q has ≥ 2 edges. To see this, observe that $G - u$ is connected (since G is 2NC); let P be a shortest (w.r.t. the number of edges) path between R and $C_0 - V(R)$ in $G - u$; then P corresponds to our desired pseudo-ear, and the head node is the end node of P in $C_0 - u - V(R)$. Clearly, any path of C_0 between r and the head node has ≥ 2 edges, hence, any witness path of the pseudo-ear has ≥ 2 edges.

In each iteration (of bridge covering), we compute a pseudo-ear using a polynomial-time algorithm that is presented in the proof of Proposition 18, see below.

The next lemma is used to lower bound the credit of a witness path.

▶ **Lemma 17.** *Let Ψ be a pseudo-ear of H w.r.t. C_0 starting at R, let Q be a witness path of Ψ, and let ru be unique bridge of C_0 incident to R. Suppose that Q satisfies one of the following:*

(a) *Q contains a white node distinct from r, or*

(b) *Q contains exactly one white node and ≥ 3 bridges, or*

(c) *Q contains exactly one white node, exactly two bridges, and a black node v such that $\deg^{(1)}_H(v) \geq 2$.*

Then $Q - r$ has at least one credit, and that credit is not needed for the credit invariant of the graph resulting from the pseudo-ear augmentation that adds Ψ to H.

▶ **Proposition 18.** *There is a polynomial-time algorithm for finding a pseudo-ear (of H w.r.t. C_0 starting at R) such that any witness path Q of the pseudo-ear satisfies one of the three conditions of Lemma 17.*

▶ **Proposition 19.** *Suppose that H satisfies the credit invariant, and a pseudo-ear augmentation is applied to H. Then the resulting graph H^{new} satisfies the credit invariant.*

Proof of Proposition 14. The proof follows from Lemmas 15, 17, and Propositions 18, 19, and the preceding discussion.

Each iteration, i.e., each pseudo-ear augmentation, can be implemented in polynomial time, and the number of iterations is $\leq |E(\text{D2})|$.

At the termination of bridge covering, each connected component of H is a 2ec-block that has one c-credit and either one b-credit, or (in the case of a small 2ec-block) $\frac{1}{3}$ b-credits. By summing the two types of credit, it follows that each small 2ec-block has $\frac{4}{3}$ credits and each large 2ec-block has ≥ 2 credits. ◀

6 The gluing step

In this section, we focus on the gluing step, and we assume that the input is an instance of MAP$\star$. For notational convenience, we denote the input by G. Recall that G is a simple, 2NC graph on ≥ 12 nodes, and G has no zero-cost S2, no unit-cost S2, no S$\{3,4\}$, no R4, and no R8. (In this section, we use all the properties of G except the absence of unit-cost S2s.)

There are important differences between our gluing step and the gluing step of [2]. Our gluing step (and overall algorithm) beats the $\frac{7}{4}$ approximation threshold because our pre-processing step eliminates the S$\{3,4\}$ obstruction and the R8 obstruction (these obstructions are not relevant to other parts of our algorithm). In the full version of the paper, see [3, Appendix], we present instances G of MAP that contain S$\{3,4\}$s (respectively, R8s) and contain none of the other six obstructions such that $opt(G)/\text{cost}(\text{D2}(G)) \approx \frac{7}{4}$.

We use H to denote the current graph of the gluing step. At the start of the gluing step, H is a simple, bridgeless graph of minimum degree two; thus, each connected component of H is 2EC; clearly, the 2ec-blocks of H correspond to the connected components of H. Recall that a 2ec-block of H is called small if it has ≤ 2 unit-edges, and is called large otherwise. Observe that a small 2ec-block of H is either a 3-cycle with one zero-edge and two unit-edges, or a 4-cycle with alternating zero-edges and unit-edges.

The following result summarizes this section:

▶ **Proposition 20.** *At the termination of the bridge-covering step, let H denote the bridgeless 2-edge cover computed by the algorithm and suppose that each small 2ec-block of H has $\frac{4}{3}$ credits and each large 2ec-block of H has ≥ 2 credits. Let γ denote* credit(H). *Assume that H contains all zero-edges. Then the gluing step augments H to a 2-ECSS H' of G (by adding edges and deleting edges) such that* $\text{cost}(H') \leq \text{cost}(H) + \gamma - 2$. *The gluing step can be implemented in polynomial time.*

Our gluing step applies a number of iterations. Each iteration picks two or more 2ec-blocks of H, and merges them into a new large 2ec-block by adding some unit-edges and possibly deleting some unit-edges such that the following invariant holds for H at the start/end of every iteration of the gluing step.

Invariants for the gluing step:
- H is a simple, bridgeless graph of minimum degree two (hence, the 2ec-blocks of H correspond to the connected components of H);
- (credit invariant) each small 2ec-block of H has $\frac{4}{3}$ credits and each large 2ec-block of H has ≥ 2 credits.

It is convenient to define the following multi-graph: let $\widetilde{G}$ 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, the notation B_i refers to either a 2ec-block of H or a node of $\widetilde{G}$). Observe that $\widetilde{G}$ is 2EC. We call a node of $\widetilde{G}$ small (respectively, large) if the corresponding 2ec-block of H is small (respectively, large). The gluing step "operates" on G and never refers to $\widetilde{G}$; but, for our discussions and analysis, it is convenient to refer to $\widetilde{G}$. (Note that $\widetilde{G}$ changes in each iteration, since the current graph H changes in each iteration.)

Suppose that $\widetilde{G}$ has ≥ 2 nodes and has no small nodes. Then, we pick any (large) node $\widetilde{v}$ of $\widetilde{G}$. Since $\widetilde{G}$ is 2EC, it has a cycle $\widetilde{C}$ incident to $\widetilde{v}$. Let $|\widetilde{C}|$ denote the number of edges of $\widetilde{C}$; note that $|\widetilde{C}| \geq 2$. Our iteration adds to H the unit-edges corresponding to $\widetilde{C}$. The credit available in H for the 2ec-blocks incident to $\widetilde{C}$ is $\geq 2|\widetilde{C}|$ and the cost of the augmentation is $|\widetilde{C}|$; hence, we have surplus credit of $2|\widetilde{C}| - |\widetilde{C}| \geq 2$. The surplus credit is given to the new large 2ec-block. Clearly, the credit invariant is preserved.

In general, small nodes may be present in $\widetilde{G}$. If we apply the above scheme and find a cycle $\widetilde{C}$ incident only to small nodes with $|\widetilde{C}| \leq 5$, then we fail to maintain the credit invariant (since only $|\widetilde{C}|/3$ credits are available for the new large 2ec-block). Consider a special case when $\widetilde{G}$ has a small node $\mathcal{A}$ that has a unique neighbour B and B is large; clearly, there are ≥ 2 parallel edges between $\mathcal{A}$ and B. Below, we show that $\mathcal{A}$ and B can be merged to form a new large 2ec-block using an augmentation of net cost one, rather than two, by deleting one or more unit-edges of $\mathcal{A}$; then we have surplus credit ≥ 2 for the new large 2ec-block. For example, if $\mathcal{A}$ is a 3-cycle of H, then there exists a unit-edge uw of $\mathcal{A}$ such that G has edges uv_1 and wv_2 where $v_1, v_2 \in B$; so the augmentation adds the unit-edges uv_1 and wv_2 to H and discards uw from H.

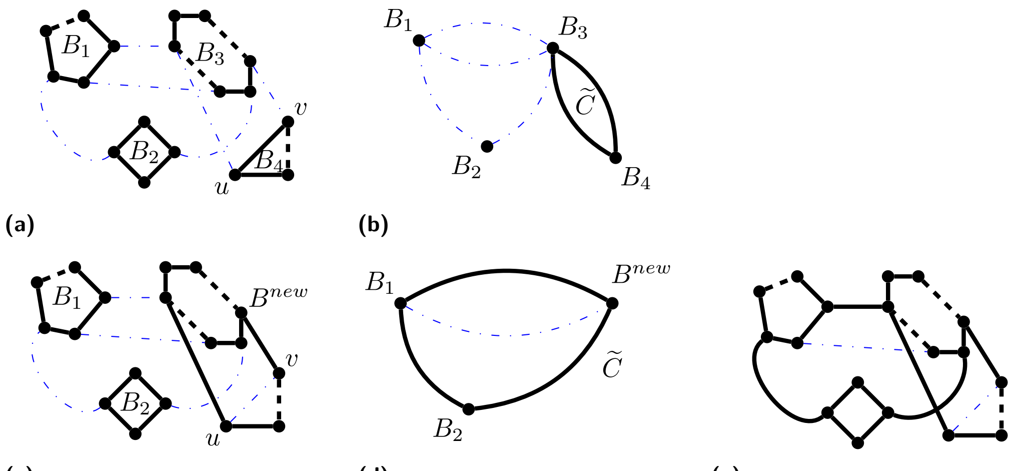

Figure 2 Two iterations of our gluing step are illustrated. At each iteration, solid lines indicate unit-edges of H, dashed lines indicate zero-edges of H, and (blue) dash-dotted lines indicate edges of $E(G) - E(H)$. Subfigures (a), (b) show Iteration 1. (a) The input graph G and a bridgeless 2-edge cover H of G. (b) The graph $\widetilde{G}$ of the graphs G, H in (a). The small node B_4 has a unique neighbour B_3 which is large. The first iteration augments via $\widetilde{C} = B_4, B_3, B_4$. Subfigures (c), (d) show Iteration 2. (c) The graphs G and H after the first iteration. B_3 and B_4 have been merged to form B^{new} by adding two unit-edges to H and deleting the unit-edge uv from H. (d) The graph $\widetilde{G}$ of the graphs G, H in (c). All nodes of $\widetilde{G}$ are large. The second iteration augments via $\widetilde{C} = B_1, B_2, B^{new}, B_1$. Subfigure (e) shows the output 2-ECSS of our gluing step.

We present key definitions and results on small 2ec-blocks in Section 6.1. Our algorithm for the gluing step is presented in Section 6.2.

6.1 Analysis of small 2ec-blocks

In this subsection, we focus on the small 2ec-blocks of H and we present the definitions and results that underlie our algorithm for the gluing step. Recall that G has ≥ 12 nodes.

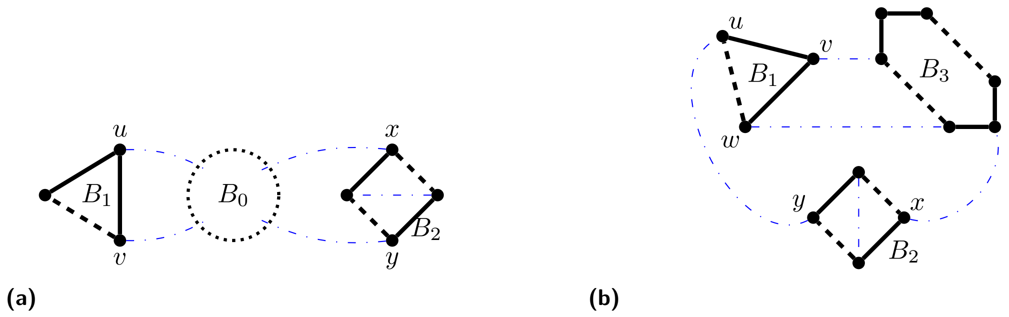

Figure 3 Illustrations of swappable edges and swappable pairs of small 2ec-blocks. Solid lines indicate unit-edges of H, dashed lines indicate zero-edges of H, and (blue) dash-dotted lines indicate edges of $E(G) - E(H)$. (**a**) The 2ec-block B_1 has a swappable edge uv, and the 2ec-block B_2 has a swappable pair $\{x, y\}$. (**b**) The 2ec-block B_1 has two swappable edges: uv is good, and vw is bad. The swappable pair $\{x, y\}$ of the 2ec-block B_2 is good.

▶ **Definition 21.** *Let $\mathcal{A}$ be a small 2ec-block of H. A unit-edge uw of $\mathcal{A}$ is called swappable if both u and w are attachments of $\mathcal{A}$ in G (that is, G has an edge ux where $x \in V(G) - \mathcal{A}$ and G has an edge wy where $y \in V(G) - \mathcal{A}$).*

▶ **Definition 22.** *Let $\mathcal{A}$ be a small 2ec-block of H. A pair of nodes $\{u, w\}$ of $\mathcal{A}$ is called a swappable pair if either (i) uw is a swappable edge of $\mathcal{A}$, or (ii) both u and w are attachments of $\mathcal{A}$ in G, u, w are not adjacent in $\mathcal{A}$ (note that $\mathcal{A}$ is a 4-cycle in this case), and the other two nodes of $\mathcal{A}$ are adjacent in G (that is, $E(G) - E(H)$ has a "diagonal edge" between the other two nodes of $\mathcal{A}$).*

▶ **Definition 23.** *Let $\mathcal{A}$ be a small 2ec-block of H. A swappable pair $\{u, w\}$ of $\mathcal{A}$ is called good if there are distinct 2ec-blocks B_u and B_w ($\mathcal{A} \neq B_u \neq B_w \neq \mathcal{A}$) such that G has an edge ux where $x \in B_u$ and G has an edge wy where $y \in B_w$; otherwise, $\{u, w\}$ is called a bad swappable pair of $\mathcal{A}$. A good (respectively, bad) swappable edge of $\mathcal{A}$ is defined similarly.*

▶ Remark 24. Observe that each iteration merges two or more 2ec-blocks of H (see the discussion following Proposition 20). Consider a small 2ec-block $\mathcal{A}$ of H that stays unchanged over several iterations. After one of these iterations, a swappable pair $\{u, w\}$ of $\mathcal{A}$ may change from good to bad, but $\{u, w\}$ cannot change from bad to good.

▶ **Lemma 25.** *Let $\mathcal{A}$ be a small 2ec-block of H. If $\mathcal{A}$ is adjacent (in G) to a unique 2ec-block B, then B is large. (That is, if there is 2ec-block B such that $\Gamma_G(V(\mathcal{A})) \subseteq V(B)$, then B is large.)*

▶ **Lemma 26.** *Let $\mathcal{A}$ be a small 2ec-block of H. Then $\mathcal{A}$ has at least one swappable pair. Moreover, if $\mathcal{A}$ is a 3-cycle, then $\mathcal{A}$ has at least one swappable edge.*

▶ **Lemma 27.** *Let $\mathcal{A}$ be a small 2ec-block of H. If $\mathcal{A}$ is a 3-cycle, and $\mathcal{A}$ is adjacent (in G) to at least two other 2ec-blocks, then it has a good swappable edge.*

Suppose that the current graph H has no good swappable pairs, that is, for every small 2ec-block $\mathcal{A}$ of H, every swappable pair of $\mathcal{A}$ is bad. To "merge away" the remaining small 2ec-blocks of H, we construct the following auxiliary digraph D^{aux}: there is a node for each 2ec-block of H, and we call the nodes corresponding to the small 2ec-blocks the *red* nodes, and the other nodes the *green* nodes; for each small 2ec-block $\mathcal{A}$ of H and each of its swappable pairs $\{u,w\}$, D^{aux} has an arc $(\mathcal{A},B)$ where B corresponds to the unique 2ec-block B of H such that $\Gamma_G(\{u,w\}) \subseteq V(B) \cup V(\mathcal{A})$. Observe that each red node of D^{aux} has at least one outgoing arc.

▶ **Lemma 28.** *Suppose that there exist no good swappable pairs. Then, D^{aux} does not have a pair of red nodes $\mathcal{A}_1, \mathcal{A}_2$ such that $(\mathcal{A}_1,\mathcal{A}_2)$ is the unique outgoing arc of $\mathcal{A}_1$ and $(\mathcal{A}_2,\mathcal{A}_1)$ is the unique outgoing arc of $\mathcal{A}_2$ (that is, if D^{aux} has a directed 2-cycle C on the red nodes, then one of the red nodes incident to C has ≥ 2 outgoing arcs).*

By the above lemma, D^{aux} either has an arc $(\mathcal{A},B)$ from a red node $\mathcal{A}$ to a green node B, or it has a directed path $\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3$ on three red nodes. In both cases, we can apply a merge step to obtain a new large 2ec-block (i.e., a green node) while preserving the credit invariant. More details are presented in the next subsection.

6.2 Algorithm for the gluing step

In this subsection, we explain the working of the algorithm for the gluing step, based on the results in the previous subsection.

Consider any small 2ec-block $\mathcal{A}$ that has a good swappable pair $\{u,w\}$ such that u is adjacent (in G) to another 2ec-block B_u, and w is adjacent (in G) to another 2ec-block B_w, and $\mathcal{A} \neq B_u \neq B_w \neq \mathcal{A}$. Observe that $G - V(\mathcal{A})$ is connected, otherwise, $\mathcal{A}$ would be an S$\{3,4\}$ of G (the arguments in the proof of Lemma 25 can be used to verify this statement). Hence, $\widetilde{G} - \mathcal{A}$ has a path between B_u and B_w; adding the edges $\mathcal{A}B_u$ and $\mathcal{A}B_w$ to this path gives a cycle $\widetilde{C}$ of $\widetilde{G}$. We merge the 2ec-blocks incident to $\widetilde{C}$ into a new large 2ec-block by adding the unit-edges corresponding $\widetilde{C}$ to H. Moreover, if $uw \in E(\mathcal{A})$, then we discard uw from H, otherwise, $\mathcal{A}$ is a 4-cycle (with two zero-edges) and $E(G) - E(H)$ has a unit-edge f between the two nodes of $\mathcal{A} - \{u,w\}$, and in this case, we add the edge f to H and we discard the two unit-edges of $\mathcal{A}$ from H. The credit available in H for $\widetilde{C}$ is $\geq \frac{4}{3}|\widetilde{C}|$ and the net cost of the augmentation is $|\widetilde{C}|-1$; hence, we have surplus credit of $\frac{1}{3}|\widetilde{C}|+1 \geq 2$ (since $|\widetilde{C}| \geq 3$). The surplus credit is given to the new large 2ec-block.

The gluing step applies the above iteration until there are no good swappable pairs in the current graph H. Then the auxiliary digraph D^{aux} is constructed. By Lemma 28, D^{aux} has either (i) an arc $(\mathcal{A},B)$ from a red node $\mathcal{A}$ to a green node B, or (ii) a directed path $\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3$ on three red nodes.

In the first case, $\mathcal{A}$ is a small 2ec-block, B is a large 2ec-block, and $\mathcal{A}$ has a swappable pair $\{u,w\}$ such that $\mathcal{A} \cup B$ contains all neighbours (in G) of $\{u,w\}$. We merge $\mathcal{A}$ and B into a new large 2ec-block as follows. We add two unit-edges between $\mathcal{A}$ and B to H (one edge is incident to u and the other edge is incident to w). Moreover, if $uw \in E(\mathcal{A})$, then we discard uw from H, otherwise, $\mathcal{A}$ is a 4-cycle (with two zero-edges) and $E(G)-E(H)$ has a unit-edge f between the two nodes of $\mathcal{A} - \{u,w\}$, and in this case, we add the edge f to H and we discard the two unit-edges of $\mathcal{A}$ from H. The credit available in H for $\mathcal{A} \cup B$ is $\geq \frac{4}{3}+2$ and the net cost of the augmentation is one; hence, we have surplus credit of $\frac{1}{3}+2 \geq 2$. The surplus credit is given to the new large 2ec-block. Consider the second case. Then $\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3$ are small 2ec-blocks such that $\mathcal{A}_1$ has a swappable pair u_1w_1 such that $\Gamma_G(\{u_1,w_1\}) \subseteq V(\mathcal{A}_1) \cup V(\mathcal{A}_2)$, and $\mathcal{A}_2$ has a swappable pair u_2w_2 such that $\Gamma_G(\{u_2,w_2\}) \subseteq V(\mathcal{A}_2) \cup V(\mathcal{A}_3)$. We add two unit-edges between $\mathcal{A}_1$ and $\mathcal{A}_2$ to H (one edge is incident to u_1 and the other edge is incident to w_1), and then we either discard one unit-edge from H (if $u_1w_1 \in E(\mathcal{A}_1)$) or we add another edge to H and discard two unit-edges of $\mathcal{A}_1$ from H (if $u_1w_1 \notin E(\mathcal{A}_1)$). We apply a similar augmentation to $\mathcal{A}_2$ and $\mathcal{A}_3$ using the swappable pair $\{u_2, w_2\}$. The credit available in H for $\mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3$ is $\geq (3 \cdot \frac{4}{3}) = 4$ and the net cost of the augmentation is two; hence, we have surplus credit of $\geq 4 - 2$. The surplus credit is given to the new large 2ec-block.

By repeatedly applying the above iteration (that merges red nodes of D^{aux} into green nodes), we obtain a current graph H that has no small 2ec-blocks. As discussed above, the merge step is straightforward when all 2ec-blocks of H are large.

▶ **Lemma 29.** *After every merge step, the subgraph B^{new} constructed by that step (that is a so-called large 2ec-block) is 2EC.*

References

1 David Adjiashvili. Beating approximation factor two for weighted tree augmentation with bounded costs. In Philip 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. doi:10.1137/1.9781611974782.157.

2 Joe Cheriyan, Jack Dippel, Fabrizio Grandoni, Arindam Khan, and Vishnu V. Narayan. The matching augmentation problem: a $\frac{7}{4}$-approximation algorithm. *Math. Program.*, 182(1):315–354, 2020. doi:10.1007/s10107-019-01394-z.

3 Joseph Cheriyan, Robert Cummings, Jack Dippel, and Jasper Zhu. An improved approximation algorithm for the matching augmentation problem. *CoRR*, abs/2007.11559, 2020. arXiv:2007.11559.

4 Joseph Cheriyan and Zhihan Gao. Approximating (unweighted) tree augmentation via lift-and-project, part II. *Algorithmica*, 80(2):608–651, 2018. doi:10.1007/s00453-017-0275-7.

5 Joseph Cheriyan, Howard J. Karloff, Rohit Khandekar, and Jochen Könemann. On the integrality ratio for tree augmentation. *Oper. Res. Lett.*, 36(4):399–401, 2008. doi:10.1016/j.orl.2008.01.009.

6 Nachshon Cohen and Zeev Nutov. A (1 + ln 2)-approximation algorithm for minimum-cost 2-edge-connectivity augmentation of trees with constant radius. *Theor. Comput. Sci.*, 489-490:67–74, 2013. doi:10.1016/j.tcs.2013.04.004.

7 R. Diestel. *Graph Theory (4th ed.)*. Graduate Texts in Mathematics, Volume 173. Springer-Verlag, Heidelberg, 2010. URL: http://diestel-graph-theory.com/.

8 Dippel, Jack. *The Matching Augmentation Problem: A 7/4-Approximation Algorithm.* M.Math. Thesis, C&O Department. UWSpace (University of Waterloo), 2019. URL: http://hdl.handle.net/10012/14700.

9 Samuel Fiorini, Martin Groß, Jochen Könemann, and Laura Sanità. Approximating weighted tree augmentation via Chvátal-Gomory cuts. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 817–831. SIAM, 2018. doi:10.1137/1.9781611975031.53.

10 Harold N. Gabow, Michel X. Goemans, Éva Tardos, and David P. Williamson. Approximating the smallest k-edge connected spanning subgraph by LP-rounding. *Networks*, 53(4):345–357, 2009. doi:10.1002/net.20289.

11 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.

12 Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995. doi:10.1137/S0097539793242618.

13 Fabrizio Grandoni, Christos Kalaitzis, and Rico Zenklusen. Improved approximation for tree augmentation: saving by rewiring. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 632–645. ACM, 2018. doi:10.1145/3188745.3188898.

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

15 Christoph Hunkenschröder, Santosh S. Vempala, and Adrian Vetta. A 4/3-approximation algorithm for the minimum 2-edge connected subgraph problem. *ACM Trans. Algorithms*, 15(4):55:1–55:28, 2019. doi:10.1145/3341599.

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

17 Guy Kortsarz and Zeev 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. doi:10.1145/2786981.

18 Lap Chi Lau, R. Ravi, and Mohit Singh. *Iterative Methods in Combinatorial Optimization*. Cambridge Texts in Applied Mathematics (No. 46). Cambridge University Press, 2011. URL: http://www.cambridge.org/catalogue/catalogue.asp?ISBN=9780521189439.

19 Zeev Nutov. On the tree augmentation problem. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, volume 87 of *LIPIcs*, pages 61:1–61:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ESA.2017.61.

20 A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Algorithms and Combinatorics, Volume 24. Springer-Verlag, Berlin Heidelberg, 2003. URL: http://www.springer.com/gp/book/9783540443896.

21 András Sebö and Jens Vygen. Shorter tours by nicer ears: 7/5-approximation for the graph-TSP, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs. *Combinatorica*, 34(5):597–629, 2014. doi:10.1007/s00493-014-2960-3.

22 S.Khuller and U.Vishkin. Biconnectivity approximations and graph carvings. *Journal of the ACM*, 41(2):214–235, 1994.

23 Vera Traub and Rico Zenklusen. A better-than-2 approximation for weighted tree augmentation. *CoRR*, abs/2104.07114, 2021. arXiv:2104.07114.

24 Santosh Vempala and Adrian Vetta. Factor 4/3 approximations for minimum 2-connected subgraphs. In Klaus Jansen and Samir Khuller, editors, *Approximation Algorithms for Combinatorial Optimization, Third International Workshop, APPROX 2000, Saarbrücken, Germany, September 5-8, 2000, Proceedings*, volume 1913 of *Lecture Notes in Computer Science*, pages 262–273. Springer, 2000. doi:10.1007/3-540-44436-X_26.

25 David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. URL: http://www.cambridge.org/de/knowledge/isbn/item5759340/?site_locale=de_DE.

26 Alexander Zelikovsky. Better approximation bounds for the network and Euclidean Steiner tree problems. Technical report, CS-96-06, Department of Computer Science, University of Virginia, USA, 1996.