RIMS-1731

Covering Cuts in Bridgeless Cubic Graphs

By

Sylvia BOYD, Satoru IWATA, Kenjiro TAKAZAWA

August 2011

Covering Cuts in Bridgeless Cubic Graphs[*]

Sylvia Boyd[†] Satoru Iwata[‡] Kenjiro Takazawa[‡]

August 8, 2011

Abstract

In this paper we are interested in algorithms for finding 2-factors that cover certain prescribed edge-cuts in bridgeless cubic graphs. We present an algorithm for finding a minimum-weight 2-factor covering all the 3-edge cuts in weighted bridgeless cubic graphs, together with a polyhedral description of such 2-factors and that of perfect matchings intersecting all the 3-edge cuts in exactly one edge. We further give an algorithm for finding a 2-factor covering all the 3- and 4-edge cuts in bridgeless cubic graphs. Both of these algorithms run in $\mathrm{O}(n^3)$ time, where n is the number of vertices.

As an application of the latter algorithm, we design a 6/5-approximation algorithm for finding a minimum 2-edge-connected subgraph in 3-edge-connected cubic graphs, which improves upon the previous best ratio of 5/4. The algorithm begins with finding a 2-factor covering all 3- and 4-edge cuts, which is the bottleneck in terms of complexity, and thus it has running time $\mathrm{O}(n^3)$. We then improve this time complexity to $\mathrm{O}(n^2 \log^4 n)$ by relaxing the condition of the initial 2-factor and elaborating the subsequent processes.

Keywords: bridgeless cubic graphs, minimum-weight 2-factor covering 3-edge cuts, polyhedral description of 2-factors covering 3-edge cuts, 2-factor covering 3- and 4-edge cuts, minimum 2-edge-connected subgraphs

Mathematics subject classification (2000): 05C85, 05C70

1 Introduction

The classical theorem of Petersen [29] shows that any bridgeless cubic graph has a 2-factor. The aim of this paper is to provide algorithms for finding 2-factors that cover certain prescribed edge-cuts in bridgeless cubic graphs, and demonstrate their usefulness for developing approximation algorithms for related NP-hard problems for such graphs. In § 3 we give Algorithm W3Cut, which finds a minimum-weight 2-factor covering all 3-edge cuts in a weighted bridgeless cubic graph in $\mathrm{O}(n^3)$ time, where n is the number of vertices, as well as a linear system describing the associated polytope for this problem. Note that this represents the first polynomial-time algorithm for this problem. We also present a polyhedral description of perfect matchings intersecting all 3-edge cuts in exactly one edge, which are a complement of those 2-factors, in bridgeless cubic graphs. In § 4 we give Algorithm 34Cut, which finds a 2-factor that covers all the 3- and 4-edge cuts in bridgeless cubic graphs. This algorithm also runs in $\mathrm{O}(n^3)$ time, and represents the first polynomial-time algorithm for this problem. In § 5 we demonstrate the usefulness of our results by using them to facilitate a new 6/5-approximation algorithm for the problem of finding a minimum 2-edge-connected spanning subgraph in a 3-edge-connected cubic graph, improving upon the previous best worst-case ratio of 5/4 for this problem [18, 19].

[*]This research was partially supported by grants from the Natural Sciences and Engineering Research Council of Canada, and the Kayamori Foundation of Informational Science Advancement.

[†]School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Ontario K1N 6N5, Canada (`sylvia@site.uottawa.ca`).

[‡]Research Institute for Mathematical Sciences (RIMS), Kyoto University, Kyoto 606-8502, Japan (`iwata@kurims.kyoto-u.ac.jp`, `takazawa@kurims.kyoto-u.ac.jp`).

The problem of finding a 2-factor covering edge-cuts in a graph is closely related to the problem of finding 2-factors of G which do not contain any cycles of prescribed lengths (see [17] for an overview of what is known on this problem). We call a cycle of length three a *triangle*, and that of length four a *square*. In a bridgeless cubic graph G, a 2-factor that covers all 3-edge cuts would form a triangle-free 2-factor, and, if G is also 3-edge-connected, a 2-factor that covers all 3- and 4-edge cuts would form a triangle- and square-free 2-factor.

There exist polynomial-time algorithms for finding a minimum-weight triangle-free 2-factor in weighted subcubic graphs [16, 23, 32]. For general weighted graphs, the complexity of this problem is unknown, while a polynomial-time algorithm for the unweighted version of this problem is given in [15]. Polynomial-time algorithms are known for finding a triangle- and square-free 2-factor in subcubic graphs [3, 17]. For general graphs, the complexity of this problem is also unknown. Moreover, Vornberger [32] showed that the problem of finding a minimum-weight triangle- and square-free 2-factor is NP-hard, even in cubic graphs.

The algorithms for finding 2-factors which do not contain any cycles of length three and/or four have proven useful in the design of approximation algorithms for related NP-hard problems (cf. [1, 11, 28]). Surprisingly, despite being closely related and thus also potentially useful, the problem of finding a 2-factor covering 3- and 4-edge cuts has received very little attention in the literature. It can be deduced from polyhedral results of Edmonds [10] that any bridgeless cubic graph has a 2-factor that covers all 3-edge cuts (see § 2), a fact which is shown in [20] and also shown and used in [6] to get the first 4/3-approximation for graph-TSP. Furthermore, Kaiser and Škrekovski [21] showed that any bridgeless cubic graph has a 2-factor covering all 3- and 4-edge cuts, however their result is not algorithmic in nature (see § 4 for more details on this). Note that it is not always possible to find a 2-factor covering all 5-edge cuts in a bridgeless cubic graph (for example, consider the Petersen graph), nor one that covers all 2-edge cuts, and the complexity of finding a minimum-weight 2-factor covering all 3- and 4-edge cuts is currently not known.

Note that a 2-factor that covers all edge-cuts would form a Hamilton cycle. The problem of finding a Hamilton cycle in bridgeless cubic graphs is known to be NP-complete (see [25]). In the weighted case, a 2-factor found by Algorithm W3Cut would provide a lower bound for the TSP in a bridgeless cubic graph. This lower bound would be stronger than the one provided by a minimum-weight 2-factor, or the one provided by a minimum-weight triangle-free 2-factor.

Moreover, Algorithm 34Cut is useful for designing an approximation algorithm for the TSP. By using our algorithm as a preprocess for the algorithm in [14] for the TSP on the metric completion of 3-edge-connected cubic graphs, we can improve the approximation ratio $(3/2 - 5/389)$ to $7/5$, which was further improved to $4/3$ recently [6]. In another related paper, Aggarwal, Garg and Gupta [1] prove $4/3$ specifically for 3-edge-connected cubic graphs in a more elegant way. Their algorithm begins by finding a triangle- and square-free 2-factor. However their algorithm and proof could easily be both simplified and shortened by using our Algorithm 34Cut to instead start with a 2-factor that covers all 3- and 4-edge cuts, as it would eliminate the necessity to deal with the cases of cycles of length five with chords.

In § 5 we further demonstrate the usefulness of Algorithm 34Cut by using it to design a polynomial-time 6/5-approximation algorithm for the *2-edge-connected spanning subgraph problem* for 3-edge-connected cubic graphs. Given any unweighted graph G, the 2-edge-connected spanning subgraph problem (henceforth 2EC) is the problem of finding a 2-edge-connected spanning subgraph of G with as few edges as possible. This problem is known to be MAX SNP-hard even for cubic graphs (see [9]), and has been widely investigated (see [19] for an overview). Krysta and Kumar [24] designed a 21/16-approximation algorithm for cubic graphs, and Csaba, Karpinski and Krysta [9] designed a $(5/4 + \epsilon)$-approximation algorithm for subcubic graphs. For 3-edge-connected cubic graphs, an algorithm achieving the approximation ratio 5/4 was given by Huh [18]. Currently, the best approximation ratio for 2EC for general graphs (and for 3-edge-connected cubic graphs) is 5/4 due to Jothi, Raghavachari and Varadarajan [19]. In [19], they build upon key concepts from the approach taken by Vempala and Vetta [31] for their 4/3-approximation algorithm. Interestingly, in [31], they mention that in terms of possible improvements using their approach, achieving a worst-case ratio of 6/5 would seem to be a barrier, and thus our algorithm that achieves this bound has some significance.

For 2EC in 3-edge-connected cubic graphs, we present two algorithms, Algorithm Apx2EC and Algorithm FastApx2EC, both of which have approximation ratio 6/5. Algorithm Apx2EC begins with a 2-factor covering all 3- and 4-edge cuts found by Algorithm 34Cut. This preprocessing is the bottleneck of the complexity of the algorithm and thus Algorithm Apx2EC has running time $O(n^3)$. In Algorithm FastApx2EC, we make use of yet a third 2-factor algorithm we provide in this paper, which finds, in a given 3-edge-connected cubic graph, a 2-factor which covers all 3-edge cuts and is square-free. Starting with this 2-factor has an advantage: the algorithm to find such a 2-factor is faster than Algorithm 34Cut, and this allows us to improve the time complexity of our 6/5-approximation algorithm to $O(n^2 \log^4 n)$. However, this improvement comes at a cost: The resulting 6/5-algorithm is more complicated and the proof of approximation is more difficult (see § 6). So it seems that Algorithm 34Cut has definite advantages in this regard.

A related approach for finding approximated 2EC solutions is to study the *integrality gap* α(2EC), which is the worst-case ratio between the optimal value for 2EC and the optimal value for the linear programming relaxation of its associated weighted problem, obtained by taking the metric completion. The value α(2EC) gives one measure of the quality of the lower bound provided by the linear programming relaxation. Moreover, a polynomial-time constructive proof for the value α(2EC) would provide an α(2EC)-approximation algorithm for 2EC. For the metric completion of 3-edge-connected cubic graphs, Huh's results [18] imply that α(2EC) is at most 5/4. In § 5 we improve upon this and show that α(2EC) for this class of graphs is at most 6/5. This is significant in that it has been conjectured that $\alpha(\text{2EC}) = 6/5$ for general weighted graphs, however what is known is only that the integrality gap lies somewhere between 6/5 and 3/2 for such graphs (see [2]).

The organization of this paper is summarized as follows. Section 2 provides basic results on 2-factors in bridgeless cubic graphs. In § 3, we present an algorithm for finding a 2-factor covering all 3-edge cuts and a polyhedral description of such 2-factors in weighted bridgeless cubic graphs. In § 4, we describe Algorithm 34Cut, an algorithm for finding a 2-factor covering all 3- and 4-edge cuts in bridgeless cubic graphs. As an example of the usefulness of Algorithm 34Cut, in § 5 we give a 6/5-approximation algorithm for 2EC in 3-edge-connected cubic graphs which runs in $O(n^3)$ time and show that 6/5 is an upper bound for α(2EC). Finally, in § 6 we give a faster $O(n^2 \log^4 n)$-time algorithm for the same problem with the same approximation ratio.

Definitions and notation

Let $G = (V, E)$ be a graph with vertex set V and edge set E. In general, unless stated otherwise, we allow G to have multi-edges (i.e., parallel edges), but no loops. We denote an edge with end vertices u and v by uv. We say that G is *weighted* if each edge $e \in E$ is assigned a real weight c_e. For any vertex subset $S \subseteq V$, let $\bar{S}$ denote $V \setminus S$ and let $\delta(S)$ denote the set of edges connecting S and $\bar{S}$. For each vertex $v \in V$, $\delta(\{v\})$ is simply denoted by $\delta(v)$. A graph in which $|\delta(v)| = 3$ for all $v \in V$ is called *cubic*. For a subgraph H of graph G, the vertex set and edge set of H are denoted by $V(H)$ and $E(H)$, respectively. For $S \subseteq V$, we let $\gamma(S)$ denote the set of edges with both ends in S and let $G[S]$ denote the subgraph of G induced by S, i.e., $G[S] = (S, \gamma(S))$. For $S \subset V$, let $G \times S$ be the graph obtained by shrinking S into a single pseudo-vertex v_S. Note that we keep multiple copies of edges in such a contraction, but remove loops.

A *matching* is a set of vertex-disjoint edges. A matching M is called *perfect* if every vertex of G is incident to an edge in M. A 2-*factor* F is a set of edges $F \subseteq E$ such that every vertex of G is incident to exactly two edges in F. Note that the complement of a 2-factor is a perfect matching in cubic graphs.

An edge subset $D \subseteq E$ of the form $D = \delta(S)$ for some nonempty proper subset $S \subset V$ is called a *cut*. A cut D of cardinality k is called a *k-edge cut* if it is minimal in the sense that D does not contain any cut as a proper subset. A graph G is said to be *k-edge-connected* if every cut of G has cardinality at least k. A graph without any 1-edge cut is called *bridgeless*. We say that a subset F of E *covers* a cut D if $F \cap D \neq \emptyset$.

2 Preliminaries

A well-known theorem of Petersen [29] states that every bridgeless cubic graph contains a perfect matching. Schönberger [30] proved the following strengthened form of Petersen's Theorem:

Theorem 2.1 ([30])**.** *Let $G = (V, E)$ be a bridgeless cubic (multi-)graph with specified edge $e^* \in E$. Then there exists a perfect matching of G that contains e^*.*

A perfect matching containing a specified edge e^* in a bridgeless cubic graph can be found in $O(n \log^4 n)$ time [4]. Taking the complement of the perfect matching in Theorem 2.1 immediately gives the following corollary.

Corollary 2.2. *Let $G = (V, E)$ be a bridgeless cubic graph with specified edge $e^* \in E$. Then there exists a 2-factor of G that does not contain e^*.*

For any edge set $F \subseteq E$, the *incidence vector of* F is the vector $\chi^F \in \mathbf{R}^E$ defined by $\chi^F_e = 1$ for $e \in F$ and $\chi^F_e = 0$ for $e \in E \setminus F$. For any edge set $F \subseteq E$ and $x \in \mathbf{R}^E$, let $x(F)$ denote the sum $\sum_{e \in F} x_e$. Given a graph G, the associated *2-factor polytope*, $P^{2F}(G)$, is the convex hull of all incidence vectors of the 2-factors of G. In [10], Edmonds shows that $P^{2F}(G)$ is given by the following linear system:

$$x(\delta(v)) = 2 \qquad \text{for all } v \in V, \tag{1}$$

$$x(X) - x(\delta(S) \setminus X) \leq |X| - 1 \qquad \text{for all } S \subset V,\ X \subseteq \delta(S),\ X \text{ a matching, } |X| \text{ odd}, \tag{2}$$

$$0 \leq x_e \leq 1 \qquad \text{for all } e \in E. \tag{3}$$

Using this linear description and similar methods to those found in [20] and [26], we can strengthen Corollary 2.2 as follows.

Theorem 2.3. *Let $G = (V, E)$ be a bridgeless cubic graph with specified edge $e^* \in E$. Then there exists a 2-factor of G that covers all 3-edge cuts in G and does not contain e^*.*

Proof. Consider $x^* = \frac{2}{3}\chi^E$. It is easily verified (see [5]) that x^* satisfies constraints (1), (2) and (3), and hence lies in $P^{2F}(G)$. Thus x^* can be expressed as a convex combination of incidence vectors of 2-factors of G, i.e., there exists 2-factors F_i $(i = 1, 2, ..., k)$ of G and positive real numbers λ_i $(i = 1, 2, ..., k)$ such that $x^* = \sum_{i=1}^k \lambda_i \chi^{F_i}$ and $\sum_{i=1}^k \lambda_i = 1$.

Now consider any 3-edge cut $D = \delta(S)$ of G. (Note that any cut consisting of three edges in a bridgeless cubic graph is minimal, and hence a 3-edge cut.) We have that

$$2 = x^*(\delta(S)) = \sum_{i=1}^k \lambda_i |F_i \cap \delta(S)|. \tag{4}$$

Since any 2-factor F intersects the cut $\delta(S)$ in 2 or 0 edges, (4) implies that $|F_i \cap \delta(S)| = 2$ for all $i = 1, 2, ..., k$. Hence every 2-factor F_i in the convex combination covers all 3-edge cuts in G.

Moreover, $\sum_{i:\, e \in F_i} \lambda_i = 2/3$ for any $e \in E$, which implies that at least one of these 2-factors does not contain edge e^*. $\square$

Kaiser and Škrekovski [21] proved a still stronger theorem, for which we will give an algorithmic proof in § 4.

Theorem 2.4 (Kaiser and Škrekovski [21])**.** *Let G be a bridgeless cubic graph with specified edge e^*. Then there exists a 2-factor of G that covers all 3- and 4-edge cuts and does not contain e^*.*

3 Minimum-weight 2-factors covering the 3-edge cuts

In this section, let $G = (V, E)$ be a weighted bridgeless cubic graph with edge weights $c \in \mathbf{R}^E$. Recall that G may have multiple edges but no loops, and $|V| = n$. We present an $\mathrm{O}(n^3)$-time algorithm for finding a minimum-weight 2-factor that covers all the 3-edge cuts in G. We also give a complete linear description of the convex hull of the incidence vectors of 2-factors that cover all 3-edge cuts of G, as well as an analogous polyhedral result for perfect matchings of G that intersect every 3-edge cut in exactly one edge.

The set of edges incident to each vertex forms a 3-edge cut, which is covered by every 2-factor in G. To exclude this kind of trivially covered 3-edge cut, we call a 3-edge cut $\delta(S)$ *proper* if $2 \leq |S| \leq n-2$. Thus our goal is to find a minimum-weight 2-factor F of G that covers all proper 3-edge cuts in G.

The general approach involves choosing a proper 3-edge cut $D = \delta(S)$, finding a 2-factor F' of $G \times S$ covering all 3-edge cuts and a 2-factor F° of $G \times \bar{S}$ covering all 3-edge cuts such that F' and F° use the same two edges of D, and then "gluing" them together (this idea is an adaptation of one used in [8] and then [16] for different problems). The fact that the resulting 2-factor covers all 3-edge cuts in G is established by Lemma 3.1 below.

For the proof of this lemma, it is useful to note that for any graph $G = (V, E)$, the function $|\delta(.)|$ is *symmetric submodular*, i.e., for every two sets $A, B \subseteq V$ the following two properties hold:

$$|\delta(A)| + |\delta(B)| \geq |\delta(A \cup B)| + |\delta(A \cap B)|, \tag{5}$$

$$|\delta(A)| + |\delta(B)| \geq |\delta(A \setminus B)| + |\delta(B \setminus A)|. \tag{6}$$

For an edge subset $F \subseteq E$, we use $\delta_F(S)$ to denote $F \cap \delta(S)$. Note that $|\delta_F(.)|$ is also symmetric submodular.

Lemma 3.1. *Let $D = \delta(S)$ be a proper 3-edge cut of a bridgeless cubic graph G, F' be a 2-factor in $G \times S$ covering all 3-edge cuts and F° be a 2-factor in $G \times \bar{S}$ covering all 3-edge cuts such that F' and F° use the same two edges of D. Then $F = F' \cup F^\circ$ is a 2-factor in G covering all proper 3-edge cuts in G.*

Proof. Let $\delta(Z)$ be a proper 3-edge cut such that S and Z are crossing (i.e., $S \cap Z \neq \emptyset$, $S \cup Z \neq V$, $S \setminus Z \neq \emptyset$ and $Z \setminus S \neq \emptyset$). Since G is cubic, the fact that $|\delta(S)| = 3$ implies that $|S|$ is odd, and thus either $|S \cap Z|$ or $|S \setminus Z|$ must be odd. Assume without loss of generality that $|S \cap Z|$ is odd. Then $|S \cup Z|$ is also odd, and hence both $|\delta(S \cap Z)|$ and $|\delta(S \cup Z)|$ are odd. Since G is bridgeless, the submodular property (5) implies that $|\delta(S \cap Z)| = 3$ and $|\delta(S \cup Z)| = 3$. Hence $\delta(S \cap Z)$ is a 3-edge cut in $G \times \bar{S}$ and $\delta(S \cup Z)$ is a 3-edge cut in $G \times S$, and it follows by the definition of F that $|\delta_F(S \cap Z)| = 2$, $|\delta_F(S \cup Z)| = 2$ and $|\delta_F(S)| = 2$, which implies $|\delta_F(Z)| \geq 2$ by the submodular property (5) of $|\delta_F(.)|$. □

The algorithm is described as follows.

Algorithm W3CUT

Input. A bridgeless cubic graph $G = (V, E)$ with edge weights $c \in \mathbf{R}^E$.

Output. A minimum-weight 2-factor F in G covering all the 3-edge cuts in G.

Step 1. Find a proper 3-edge cut $D = \delta(S)$ of G such that no proper subset Y of S forms a proper 3-edge cut $\delta(Y)$ of G. If G does not contain any proper 3-edge cuts, then simply find a minimum-weight 2-factor F in G.

Step 2. Let $D = \{e_1, e_2, e_3\}$. For $i = 1, 2, 3$, find a minimum weight 2-factor F_i° in $G \times \bar{S}$ not containing edge e_i, the existence of which follows from Corollary 2.2.

Step 3. For each 2-factor F_i° found in Step 2, let $L_i = c(\gamma(S) \cap F_i^\circ)$. We assign an extra weight α_i to each edge e_i $(i = 1, 2, 3)$ in $G \times S$ such that $\alpha_1 + \alpha_2 = L_3$, $\alpha_2 + \alpha_3 = L_1$ and $\alpha_3 + \alpha_1 = L_2$. Define new edge weights c' for the edges of $G \times S$ by $c'_e = c_e$ for $e \in \gamma(\bar{S})$ and $c'_e = c_{e_i} + \alpha_i$ for $e = e_i$ $(i = 1, 2, 3)$.

Step 4. Now consider $G \times S$, which is also a bridgeless cubic graph of smaller size than G. Find a minimum-weight 2-factor F' covering all 3-edge cuts by recursively applying the algorithm to $G \times S$ with weights c'. Let e_{i^*} be the edge in $D \setminus F'$ and return $F = F' \cup F_{i^*}^\circ$.

The minimality of $c(F)$ in Step 4 is verified as follows. Let $\hat{F}$ be any 2-factor of G that covers all 3-edge cuts in G and $\hat{F}'$ be a 2-factor in $G \times S$ obtained from $\hat{F}$ by shrinking S. Then we have that $c(\hat{F}) \geq c'(\hat{F}') \geq c'(F') = c(F)$. Therefore the output F of Algorithm W3Cut has minimum weight among the 2-factors covering all 3-edge cuts in G.

The time complexity $T(n)$ of this algorithm is analyzed as follows. We have that

$$T(n) = T(n - l + 2) + T'(l) + f(n),$$

where $f(n)$ is the time complexity for finding a proper 3-edge cut $\delta(S)$ in G such that no proper subset of S provides a proper 3-edge cut and $T'(l)$ is that for solving the 2-factor problem in $G \times \bar{S}$ with l vertices. We obtain $f(n) = \mathrm{O}(n^2)$ as follows. We can assume that G is 3-edge-connected. (Otherwise, it suffices to find a 2-edge cut $\delta(U) = \{u_1v_1, u_2v_2\}$, where $u_1, u_2 \in U$, and then find a desired proper 3-edge cuts in two graphs, one is $G[U]$ plus edge u_1u_2 and the other $G[\bar{U}]$ plus v_1v_2.) Now construct a directed graph by replacing every edge with two oppositely directed edges and find three edge-disjoint r-arborescences for a fixed vertex $r \in V$. We have three paths from r to v for every vertex $v \in V \setminus \{r\}$, which correspond to a maximum r-v flow. Then we know all 3-edge cuts separating r and v by decomposing the residual graph into strongly connected components. Finding a 2-edge cut can be done in $\mathrm{O}(n \log n)$ time [13], finding three edge-disjoint r-arborescences in $\mathrm{O}(n \log n)$ time [13], decomposing the residual graphs for all vertices in $V \setminus \{r\}$ in $\mathrm{O}(n^2)$ time, and thus we obtain $f(n) = \mathrm{O}(n^2)$. Also, we have $T'(l) = \mathrm{O}(l^2 \log l)$ [12], and consequently $T(n) = \mathrm{O}(n^3)$.

Theorem 3.2. *Algorithm* W3Cut *finds a minimum-weight* 2*-factor in* G *covering all the* 3*-edge cuts in* $\mathrm{O}(n^3)$ *time.*

Since the complement of a 2-factor in G is a perfect matching, the following is an immediate consequence of Theorem 3.2:

Corollary 3.3. *Algorithm* W3Cut *finds a maximum-weight perfect matching in* G *which intersects every* 3*-edge cut in* G *in exactly one edge in* $\mathrm{O}(n^3)$ *time.*

We complete this section by giving a polyhedral description of the convex hull of all incidence vectors of the 2-factors of G covering all 3-edge cuts, denoted by $P^{2F3}(G)$, as well as an analogous polyhedral result for perfect matchings of G.

Theorem 3.4. *The polytope* $P^{2F3}(G)$ *is given by the system of constraints* (1)–(3) *for the* 2*-factor polytope* $P^{2F}(G)$*, with the addition of the following constraints:*

$$x(\delta(S)) = 2 \quad \text{for all } S \subset V,\ \delta(S) \text{ a proper 3-edge cut of } G. \tag{7}$$

Proof. Let P be the polytope defined by the system of constraints (1)–(3) and constraints (7). Clearly $P^{2F3}(G) \subseteq P$. To show $P \subseteq P^{2F3}(G)$ is also true (and thus complete the proof), we show that any $x^* \in P$ can be expressed as a convex combination of incidence vectors of 2-factors that cover all 3-edge cuts of G.

Since $x^* \in P$, we have that x^* is also in $P^{2F}(G)$, and thus can be expressed as a convex combination of incidence vectors of 2-factors, i.e., there exist 2-factors $F_1, \dots, F_k$ of G and positive real numbers $\lambda_1, \dots, \lambda_k$ such that

$$x^* = \sum_{i=1}^{k} \lambda_i \chi^{F_i} \text{ and } \sum_{i=1}^{k} \lambda_i = 1. \tag{8}$$

Now consider any proper 3-edge cut $\delta(S)$ of G. Clearly $\chi^{F_i}(\delta(S)) \leq 2$ for $i = 1, \dots, k$, since a 2-factor will use 0 or 2 edges of a 3-edge cut. Since we also have that $x^*(\delta(S)) = 2$, it follows from (8) that we must have $\chi^{F_i}(\delta(S)) = 2$ for $i = 1, \dots, k$. Consequently x^* is a convex combination of incidence vectors of 2-factors that cover all 3-edge cuts of G. □

Similarly, we can obtain a polyhedral description of the convex hull of the incidence vectors of perfect matchings of G that intersect every 3-edge cut in exactly one edge, which we denote by $P^{PM3}(G)$.

Given a graph $G = (V, E)$, the associated *perfect matching polytope* $P^{PM}(G)$ is the convex hull of all incidence vectors of the perfect matchings of G, and is given by the following linear system [10]:

$$x(\delta(v)) = 1 \quad \text{for all } v \in V, \tag{9}$$

$$x(\delta(S)) \geq 1 \quad \text{for all } S \subset V,\ |S| \text{ odd}, \tag{10}$$

$$x_e \geq 0 \quad \text{for all } e \in E. \tag{11}$$

Using this linear system, we have the following.

Theorem 3.5. *The polytope $P^{PM3}(G)$ is given by the system of constraints* (9)–(11) *for the perfect matching polytope $P^{PM}(G)$, with the addition of the following constraints:*

$$x(\delta(S)) = 1 \quad \textit{for all } S \subset V,\ \delta(S) \textit{ a proper } 3\textit{-edge cut of } G. \tag{12}$$

Proof. Let P be the polytope given by the constraints (9)–(12). Clearly $P^{PM3}(G) \subseteq P$. To show $P \subseteq P^{PM3}(G)$ is also true (and thus complete the proof), we show that any $x^* \in P$ can be expressed as a convex combination of incidence vectors of perfect matchings that intersect every 3-edge cut in exactly one edge.

Since $x^* \in P$, we have that x^* is also in $P^{PM}(G)$, and thus can be expressed as a convex combination of incidence vectors of perfect matchings of G, i.e., there exist perfect matchings $M_1, \dots, M_k$ of G and positive real numbers $\lambda_1, \dots, \lambda_k$ such that

$$x^* = \sum_{i=1}^{k} \lambda_i \chi^{M_i} \text{ and } \sum_{i=1}^{k} \lambda_i = 1. \tag{13}$$

Now consider any proper 3-edge cut $\delta(S)$ of G. Clearly $\chi^{M_i}(\delta(S)) \geq 1$ for $i = 1, \dots k$, since a perfect matching will use 1 or 3 edges of a 3-edge cut (note that $|W|$ is odd for such a cut). Since we also have that $x^*(\delta(S)) = 1$, it follows from (13) that we must have $\chi^{M_i}(\delta(S)) = 1$ for $i = 1, \dots k$. Consequently we have that x^* is a convex combination of incidence vectors of perfect matchings that intersect every 3-edge cut in exactly one edge. □

4 Finding 2-factors covering all the 3- and 4-edge cuts

In this section, let $G = (V, E)$ be a bridgeless cubic graph with n vertices. We present Algorithm 34Cut, an $O(n^3)$-time algorithm for finding a 2-factor that covers all the 3- and 4-edge cuts and does not contain a specified edge $e^* \in E$, which provides a constructive proof for

Theorem 2.4. We also obtain an analogous result for the perfect matching of G which is the complement of this 2-factor.

Once again, G may have multiple edges, but no loops. Observe that a 4-edge cut $D = \delta(S)$ with $|S| = 2$ is covered by every 2-factor. We call a 4-edge cut $D = \delta(S)$ *proper* if $3 \leq |S| \leq n-3$. Our goal is to find a 2-factor F that covers all the proper 3- and 4-edge cuts in G.

4.1 Covering 3-edge cuts

Our algorithm starts by finding a proper 3-edge cut $\delta(S)$. We will discuss later in § 4.2 what to do if G is free from proper 3-edge cuts. Once an appropriate S is found, the algorithm decomposes G into $G \times S$ and $G \times \bar{S}$. Note that both $G \times S$ and $G \times \bar{S}$ are bridgeless cubic graphs. Without loss of generality, suppose that $G \times S$ contains e^*. We then apply the algorithm recursively to $G \times S$ to obtain a 2-factor F' covering all the 3- and 4-edge cuts in $G \times S$ and excluding e^*. Identify the two edges $f, f' \in F'$ incident to v_S. Again apply the algorithm recursively to $G \times \bar{S}$ to obtain a 2-factor F° with $f, f' \in F^\circ$ covering all the 3- and 4-edge cuts in $G \times \bar{S}$. Then $F = F^\circ \cup F'$ forms a 2-factor in G. The algorithm terminates by returning F.

Lemma 4.1. *The output F of the above algorithm is a 2-factor in G covering all the 3- and 4-edge cuts in G.*

Proof. By Lemma 3.1, F covers all 3-edge cuts in G. So consider a 4-edge cut in G. Let $\delta(Z)$ be a proper 4-edge cut such that S and Z are crossing. Since $|\delta(S)|$ is odd and G is cubic, it follows that $|S|$ is odd, and thus either $|S \cap Z|$ or $|S \setminus Z|$ is even. Suppose without loss of generality that $|S \cap Z|$ is even and $|S \setminus Z|$ is odd, which implies $|\delta(S \cap Z)|$ is even and $|\delta(S \setminus Z)|$ is odd. Since G is bridgeless, we have $|\delta(S \cup Z)| \geq 2$, and thus $|\delta(S \cap Z)| \leq 5$ by the submodular property (5). Thus $|\delta(S \cap Z)| = 2$ or $|\delta(S \cap Z)| = 4$ holds.

Suppose $|\delta(S \cap Z)| = 2$. Consider $|\delta(Z \setminus S)|$. It must be even since $|Z \setminus S|$ is even, and thus $|\delta(Z \setminus S)|$ is either 2 or 4 by the submodular property (6). Since no proper subset of $\delta(Z)$ forms a cut, we have $|\delta(Z \setminus S)| + |\delta(S \cap Z)| > |\delta(Z)|$. Thus $|\delta(Z \setminus S)| = 4$, which implies $|\delta(S \setminus Z)| = 3$ by the submodular property (6). By the definition of F, we have $|\delta_F(Z \setminus S)| \geq 2$, $|\delta_F(S \setminus Z)| \geq 2$, and $|\delta_F(S)| = 2$. Thus we have $|\delta_F(Z)| \geq 2$ by the submodular property (6) for $|\delta_F(.)|$.

If $|\delta(S \cap Z)| = 4$, we have $|\delta(S \cup Z)| = 3$ by the fact that $|S \cup Z|$ is odd and the submodular property (5). By the definition of F, we have $|\delta_F(S \cap Z)| \geq 2$, $|\delta_F(S \cup Z)| = 2$, and $|\delta_F(S)| = 2$, we have $|\delta_F(Z)| \geq 2$ by the submodular property (5) for $|\delta_F(.)|$. □

4.2 Covering 4-edge cuts

In this subsection, suppose that G is free from proper 3-edge cuts. We now discuss how to find a 2-factor covering all the 4-edge cuts in G and containing two specified edges f and f' incident to a vertex $r \in V$. If G admits no proper 4-edge cut, then we find an arbitrary 2-factor containing f and f' by the algorithm in [4].

We now suppose that G has a proper 4-edge cut. Let $D = \delta(Y)$ be a proper 4-edge cut such that $r \notin Y$ and no proper subset Z of Y provides a proper 4-edge cut $\delta(Z)$. Suppose $D = \{e_1, e_2, e_3, e_4\}$ with $e_j = u_j v_j$, $u_j \in Y$, and $v_j \in \bar{Y}$ for $j = 1, 2, 3, 4$. Note that these end-vertices are all distinct, since D is a proper cut, no proper subset of D forms a cut, and G has no proper 3-edge cuts. Thus we cannot have both f and f' belong to D. Construct the graph $G \times \bar{Y}$. This graph is bridgeless and nearly cubic in the sense that every vertex has degree three except for $v_{\bar{Y}}$, which has degree four. It follows from Corollary 2.2 that such a graph also admits a 2-factor, as will be shown in the proof for Lemma 4.2 below. Let K be the family of pairs of edges in D contained in 2-factors in $G \times \bar{Y}$. Then, $H = (D, K)$ forms a graph with vertex set D and edge set K.

Lemma 4.2. *The graph $H = (D, K)$ contains a square as a subgraph.*

Proof. For each pair of distinct edges $e \in D$ and $e' \in D$, there exists a 2-factor that contains e and does not contain e' in $G \times \bar{Y}$. To see this, split the vertex $v_{\bar{Y}}$ into a pair of vertices t' and t° connected by an additional edge e° in such a way that e and e' are incident to t' and the remaining two edges in D are incident to t°. The resulting graph is a bridgeless cubic graph, which admits a 2-factor that does not contain e' by Corollary 2.2. Contracting the new edge e°, we obtain the desired 2-factor in $G \times \bar{Y}$. As a consequence, the degree of H is at least two at each $e \in D$. This implies that H contains a square as a subgraph. □

We now suppose without loss of generality that e_1e_2, e_2e_3, e_3e_4 and e_4e_1 form a square in H. Then construct a graph $G^* = (W, E^*)$ as follows. Construct $G \times Y$, and then split v_Y into a pair of vertices s_1 and s_2 connected by an additional edge e^* in such a way that e_1 and e_3 are incident to s_1 and e_2 and e_4 are incident to s_2. Thus the vertex set W and the edge set E^* of G^* are given by $W = (V \setminus Y) \cup \{s_1, s_2\}$ and $E^* = (E \setminus \gamma(Y)) \cup \{e^*\}$. Note that G^* is a bridgeless cubic graph which possibly contains a proper 3-edge cut, and smaller in size than G. Apply the algorithm recursively to obtain a 2-factor F^* covering all the 3- and 4-edge cuts in G^* and containing the two specified edges f and f'.

We then split $v_{\bar{Y}}$ in $G \times \bar{Y}$ into two vertices t_1 and t_2 connected by an additional edge $e^\bullet$ in such a way that e_1 and e_3 are incident to t_1 and e_2 and e_4 are incident to t_2 to obtain a bridgeless cubic graph $G^\bullet$. Find a 2-factor $F^\bullet$ such that $F^\bullet \cap D = F^* \cap D$ as follows: If $|F^* \cap D| = 4$, then it suffices to find a 2-factor in $G^\bullet$ excluding $e^\bullet$. If $|F^* \cap D| = 2$, then the existence of $F^\bullet$ follows from the construction of G^* and $G^\bullet$, and we have already obtained $F^\bullet$ in the construction of H. Then $F = F^* \cup F^\bullet$ forms a 2-factor in G. The following lemma establishes the correctness of our algorithm.

Lemma 4.3. *The output F is a 2-factor covering all the 4-edge cuts in G.*

Proof. Let $\delta(Z)$ be a proper 4-edge cut such that Y and Z are crossing.

Suppose $|\delta(Y \cap Z)| = 2$, which implies that $|Y \cap Z|$ is even. Since both $|Z|$ and $|Y \cap Z|$ are even, $|Z \setminus Y|$ is even, and so is $|\delta(Z \setminus Y)|$. Thus by the submodular property (6), $|\delta(Z \setminus Y)|$ is 2, 4 or 6. Since no proper subset of $\delta(Z)$ forms a cut, we have $|\delta(Z \setminus Y)| + |\delta(Y \cap Z)| > |\delta(Z)|$. Thus $|\delta(Z \setminus Y)| \neq 2$. If $|\delta(Z \setminus Y)| = 6$, then we have $|\delta(Y \setminus Z)| = 2$ by the submodular property (6), which is impossible since no proper subset of $\delta(Y)$ is a cut. Thus $|\delta(Z \setminus Y)| = 4$ holds, and there exists exactly one edge between $Y \cap Z$ and $Z \setminus Y$. Since $|\delta_F(Z \setminus Y)| \geq 2$ by the definition of F, at least one edge in F intersects $\delta(Z)$, which means Z is covered by F. Similarly, we may assert that $\delta(Z)$ is covered by F if $|\delta(Y \setminus Z)| = 2$.

Suppose $|\delta(Y \cap Z)| = 3$. Then $Y \cap Z$ must be a singleton, as G has no proper 3-edge cuts. Since Y and Z are proper 4-edge cuts, neither $Y \setminus Z$ nor $Z \setminus Y$ is a singleton. Note that $|Y \setminus Z|$ and $|Z \setminus Y|$ are odd, and so are $|\delta(Y \setminus Z)|$ and $|\delta(Z \setminus Y)|$. Therefore we have $|\delta(Y \setminus Z)| \geq 5$ and $|\delta(Z \setminus Y)| \geq 5$, which provide a contradiction to (6). Thus we obtain $|\delta(Y \cap Z)| \neq 3$. Similarly, we may assert that $|\delta(Y \setminus Z)| \neq 3$.

Finally, consider the remaining case of $|\delta(Y \cap Z)| = |\delta(Y \setminus Z)| = 4$. Since no proper subset of Y provides a proper 4-edge cut, we have $|Y \cap Z| = |Y \setminus Z| = 2$. Then the induced subgraph $G[Y] = (Y, \gamma(Y))$ forms a square with $\gamma(Y) = \{u_1u_2, u_2u_3, u_3u_4, u_4u_1\}$. If $|F^* \cap D| = 2$, then $F^\bullet$ contains three edges from $\gamma(Y)$, and at least one of the three intersects $\delta(Z)$. If $|F^* \cap D| = 4$, then e_1 and e_3 belongs to the same cycle in F^*, whereas u_1 and u_3 stay on different sides of $\delta(Z)$. Thus there must be an edge in F intersecting $\delta(Z)$. □

4.3 Algorithm description and complexity analysis

The algorithm is summarized as follows.

Algorithm 34Cut

Input. A bridgeless cubic graph $G = (V, E)$ and an edge $e^* \in E$.

Output. A 2-factor in G covering all the 3- and 4-edge cut in G and not containing e^*.

Step 1. Find a proper 3-edge cut $\delta(S)$. If G has no proper 3-edge cut, then go to Step 2. Otherwise, construct $G \times S$ and $G \times \bar{S}$, and let $G \times S$ contain e^*. Apply the algorithm recursively to obtain a 2-factor F' covering all the 3- and 4-edge cuts in $G \times S$ and excluding e^*. Let $f, f' \in F'$ be the edges incident to v_S. Then, apply the algorithm recursively to $G \times \bar{S}$ to obtain a 2-factor F° in $G \times \bar{S}$ covering all the 3- and 4-edge cuts in $G \times \bar{S}$ and excluding the unique edge in $\delta(S) \setminus \{f, f'\}$. Return $F' \cup F^\circ$.

Step 2. Find a proper 4-edge cut $D = \delta(Y)$ such that Y does not contain both endpoints of e^* and no proper subset Z of Y provides a proper 4-edge cut $\delta(Z)$. Then, go to Step 3.

Step 3. For each pair of distinct edges $e, e' \in D$, construct a graph $G(e, e')$ obtained from $G \times \bar{Y}$ by splitting $v_{\bar{Y}}$ into a pair of vertices t' and t° connected by an additional edge e° in such a way that e and e' are incident to t' and the remaining two edges in D are incident to t°, and find a 2-factor in $G(e, e')$ excluding e°. Then, go to Step 4.

Step 4. Construct G^* according to the result in Step 3. Apply the algorithm recursively to obtain a 2-factor F^* in G^* covering all the 3- and 4-edge cuts in G^* and excluding e^*, and then go to Step 5.

Step 5. Construct $G^\bullet$, find a 2-factor $F^\bullet$ in $G^\bullet$ such that $F^\bullet \cap D = F^* \cap D$, and then return $F^\bullet \cup F^*$.

We analyze the time complexity $T(n)$ of Algorithm 34Cut. The decomposition into $G \times S$ and $G \times \bar{S}$ for a proper 3-edge cut $\delta(S)$ implies that $T(n) = T(n_1) + T(n_2) + f(n)$, where $f(n)$ is the time complexity for finding a proper 3-edge cut $\delta(S)$ in G and $n_1 + n_2 = n$. Note that $f(n) = \mathrm{O}(n^2)$, as is shown in § 3. By repeated decomposition in Step 1, we obtain that

$$T(n) = \sum_{i=1}^{k} T'(n_i) + \mathrm{O}(n^3),$$

where $\sum_{i=1}^{k} n_i = n$ and $T'(n_i)$ is the time complexity for finding a 2-factor covering all the 4-edge cuts in a bridgeless cubic graph on n_i vertices without proper 3-edge cuts. Furthermore, for $T'(n)$, we have that

$$T'(n) = T(n - l) + g(l) + h(n),$$

where $g(l)$ is the time complexity for finding a 2-factor excluding a specified edge in a bridgeless cubic graph with l vertices, and $h(n)$ is that for finding a proper 4-edge cut $\delta(Y)$ such that no proper subset Z of Y provides a proper 4-edge cut in a bridgeless cubic graph on n vertices without a proper 3-edge cut. We have $g(l) = \mathrm{O}(l \log^4 l)$ [4], and we obtain $h(n) = \mathrm{O}(n^2)$ as follows. We can assume that the graph has neither proper 3-edge cuts nor 2-edge cuts. Choose a vertex u and denote the edges incident to u by e_1, e_2 and e_3. For each edge f not adjacent to e_1, contract e_1 and f to vertices v_{e_1} and v_f, respectively, and find four edge-disjoint paths connecting v_{e_1} and v_f, which correspond to a maximum v_{e_1}-v_f flow and can be found in $\mathrm{O}(n)$ time [22, 27]. Then we know all 4-edge cuts separating e_1 and f. In order to find 4-edge cuts including e_1, do the same procedures for e_2 and e_3. Consequently, we obtain $h(n) = \mathrm{O}(n^2)$, $T'(n) = \mathrm{O}(n^3)$ and $T(n) = \mathrm{O}(n^3)$.

Theorem 4.4. *Algorithm* 34Cut *finds a* 2*-factor in* G *covering all the* 3*- and* 4*-edge cuts and not containing a specified edge* $e^* \in E$ *in* $\mathrm{O}(n^3)$ *time.*

Again taking the complement of the obtained 2-factor, we have the following:

Corollary 4.5. *Algorithm* 34Cut *finds a perfect matching in* G *intersecting every* 3*-edge cut in exactly one edge and every* 4*-edge cut in* 0 *or* 2 *edges, and containing a specified edge* $e^* \in E$ *in* $\mathrm{O}(n^3)$ *time.*

We remark that Kaiser and Škrekovski's original proof [21] for Theorem 2.4 is not a constructive one, while they also build upon a similar argument of gluing 2-factors. In their proof, proper 4-edge cuts satisfying a specific property, not *interlaced* with any other proper 4-edge cut, in their terms, play a key role. Thus, in designing an algorithm based on their proof, one would need an efficient subroutine for finding a proper 4-edge cut satisfying that property.

5 Approximating a minimum 2-edge-connected subgraph

In this section, we describe Algorithm APX2EC, an $O(n^3)$-time algorithm for finding a 2-edge-connected spanning subgraph of a 3-edge-connected cubic graph $G = (V, E)$ with at most $6n/5$ edges, which uses Algorithm 34CUT as a preprocess. We then discuss the integrality gap $\alpha(2EC)$ for 2EC and prove that it is at most 6/5 for 3-edge-connected cubic graphs. For clarity, we remark here that a 3-edge-connected cubic graph has no multiple edges.

5.1 An approximation algorithm

A *path* is a sequence $(v_0, e_1, v_1, \ldots, e_k, v_k)$, where $e_i = v_{i-1}v_i$ $(i = 1, \ldots, k)$ and $v_1, \ldots, v_k$ are distinct. A path in a graph G is called a *Hamilton path* if it contains all vertices in G. A *cycle* is a sequence $(v_0, e_1, v_1, \ldots, e_k, v_k)$, where $e_i = v_{i-1}v_i$ $(i = 1, \ldots, k)$, $v_1, \ldots, v_{k-1}$ are distinct and $v_1 = v_k$. For a cycle C, an edge $e \in E[C] \setminus E(C)$ is a *chord* of C. The size of a cycle C is defined by the number of vertices in C and denoted by $|C|$. Where no confusion arises, we often identify a path or a cycle Q with the subgraph consisting of the vertices and edges in Q. For convenience, for a subgraph H of G, $\delta(H)$ denotes the set of edges connecting $V(H)$ and $V \setminus V(H)$. Also, we will use $G[H]$ to denote the subgraph induced by $V(H)$.

The following lemma is an easy observation, but plays a key role in our algorithm.

Lemma 5.1. *Let $G = (V, E)$ be a 2-edge-connected graph and C be a cycle in G with at most two chords. Let $V^* \subseteq V(C)$ be the set of vertices not incident to the chords. For any vertex $v^* \in V^*$, there is a Hamilton path in $G[C]$ starting at v^* and ending at some vertex $u^* \in V^*$.*

Proof. Denote the two vertices adjacent to v^* on C by i and j. We assume that $i \notin V^*$ and $j \notin V^*$, since the statement is obvious if $i \in V^*$ or $j \in V^*$.

Suppose that C has a chord ij. If ij is the unique chord of C, then the statement holds. Assume that C has another chord kl and the vertices in V^* appears in the order of i, k, l, j on C. If l and j are adjacent on C, then $G[C]$ has a Hamilton path starting at v^*, traversing j, ji, i, k, kl, l in this order and ending at a vertex between k and l. If there are vertices between l and j on C, then $G[C]$ has a Hamilton path starting at v^*, traversing j, ji, i, k, l in this order and ending at a vertex between j and l.

Suppose the chord ij does not exist. In this case, we have two chords, ik and jl. Assume that the vertices in V^* appears in the order of i, k, l, j on C. Then, $G[C]$ has a Hamilton path starting at v^*, traversing i, k, l, lj, j in this order and ending at a vertex between j and l.

Assume that the vertices in V^* appears in the order of i, l, k, j on C. If k and l are not adjacent on C, then $G[C]$ has a Hamilton path starting at v^*, traversing i, l, lj, j, k in this order and ending at a vertex between k and l. If k and l are adjacent on C, then there is at least one vertex between i and l or between j and k, since otherwise G has a 1-edge cut incident to v^*, which contradicts that G is 2-edge-connected. Suppose, without loss of generality, that i and l are not adjacent. Then, $G[C]$ has a Hamilton path starting at v^*, traversing i, ik, k, j, jl, l in this order and ending at a vertex between i and l. □

Note that the proof implies how to find a desired Hamilton path in $G[C]$.

To begin the algorithm, we apply Algorithm 34CUT to G to obtain a 2-factor F which covers all the 3- and 4-edge cuts. The family of cycles in F is denoted by $\mathcal{C}$. The edges in $E \setminus F$, which form a perfect matching in G, are called the *matching edges*. Observe that $|C| \geq 5$ for each

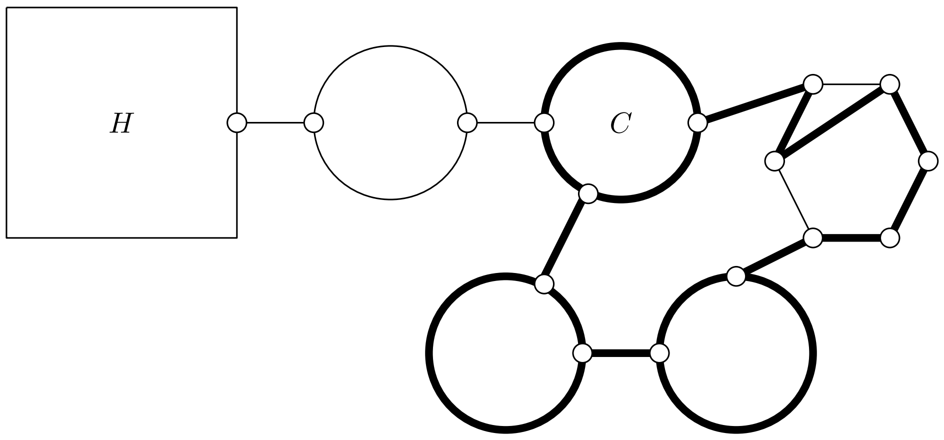

Figure 1: Construction of a lollipop L. The thick edges are the edges in L. The hexagon is a small cycle of size six, and the four circles indicate large cycles. (Some vertices and edges are omitted.)

cycle $C \in \mathcal{C}$. We say that a cycle $C \in \mathcal{C}$ is *large* if $|C| \geq 10$, and *small* if $|C| \leq 9$. Note that all small cycles C have at most two chords in $G[C]$, and thus Lemma 5.1 can be applied to such cycles.

In the algorithm, we maintain a subgraph H satisfying the following properties:

- H is 2-edge-connected;
- $V(H)$ is the union of the vertex set of a subfamily of the cycles in $\mathcal{C}$; and
- $|E(H)| \leq 6|V(H)|/5$.

The subgraph H is initially an arbitrary cycle in $\mathcal{C}$, and augmented by adding another subgraph H'. The algorithm terminates when $V(H) = V$.

In constructing H', first we choose an edge $e = uv \in \delta(H)$, where $u \in V(H)$ and $v \in V \setminus V(H)$, and let $H' := (\{u\}, \emptyset)$. Here, v belongs to a cycle $C \in \mathcal{C}$ which is not contained in H. If C is a large cycle, then we add e and C to H', and continue to grow H' from an arbitrary matching edge $f \in \delta(C) \setminus \{e\}$. If C is a small cycle, by Lemma 5.1 there exists a Hamilton path P of $G[C]$ starting at v and ending at some vertex w incident to a matching edge $f \in \delta(C) \setminus \{e\}$. We call v and w the *initial vertex* and *terminal vertex* of P, respectively. Here, we add e and P to H', and then continue to grow H' again from f, redefining $e := f$ and C as the cycle we reach at the other end of f.

The above two procedures are applied when C is not contained in H or H'. If we traverse $e = uv$ to reach a cycle $C \in \mathcal{C}$ which is in H, then we have completed the growth of the subgraph H', and we augment H with H'. If we traverse $e = uv$ to reach a cycle $C \in \mathcal{C}$ with $v \in C$ which has already been added to H', then we add e to H' and construct either a *lollipop* or a *tadpole*.

We construct a lollipop if C is a large cycle. A lollipop L is a subgraph consisting of the large cycle C, plus the elements of H' added after C was added to H'. See Figure 1 for an illustration. Then, we continue to grow H' from a matching edge $f \in \delta(L) \setminus E(H')$. Note that f always exists since G is 3-edge-connected.

We construct a tadpole if C is a small cycle. A tadpole T is a subgraph consisting of the Hamilton path P derived from C plus the elements of H' added after P was added to H'. For a tadpole T, we specify two subgraphs, the *tail* and the *head*. Let v_C and w_C be the initial and terminal vertices of P, respectively. The tail of T is a subgraph of P, the path connecting v_C and v. The head of T consists of the subgraph of P connecting v and w_C, and elements of

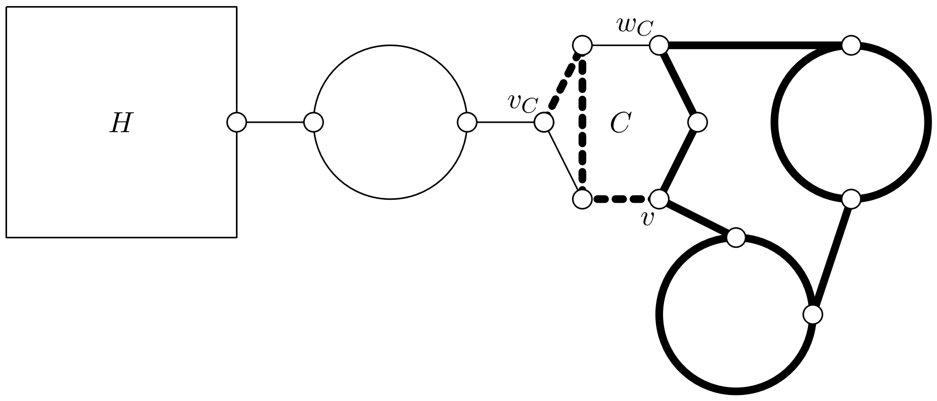

Figure 2: Construction of a tadpole T. The thick edges are the edges in T. The hexagon is a small cycle of size six, and the three circles indicate large cycles. (Some vertices and edges are omitted.) The dotted thick edges form the tail of T, and the solid thick edges form the head of T.

T added to H' after P was added to H'. See Figure 2 for an illustration. Then, we continue to grow H' from a matching edge connecting the head of T to $V \setminus V(T)$. The fact that there always exists such a matching edge will be proved in Lemma 5.2.

We call a cycle in $\mathcal{C}$ which does not belong to any lollipop or tadpole as an *independent cycle*. If C is neither in H nor independent, then we construct a larger lollipop or tadpole. If C is contained in a lollipop L, we construct a larger lollipop $\hat{L}$, which consists of L plus the elements of H' added after L was constructed. Then, continue to grow H' from a matching edge in $\delta(\hat{L}) \setminus E(H')$.

If C is contained in a tadpole T, then we construct a larger tadpole $\hat{T}$, which consists of T and elements of H' added after T was constructed. The tail and head of $\hat{T}$ are defined as follows:

- If the vertex $v \in V(C)$ at which we come back to C is in the tail of T, then the tail of $\hat{T}$ is a subgraph of the tail of T, a path connecting v and the initial vertex of the Hamilton path providing the tail of T. The head of $\hat{T}$ is a subgraph of H' consisting of edges in T not in the tail of $\hat{T}$, and edges in H' added after T was constructed.
- If v is in the head of T, then the tail of $\hat{T}$ is the same as that of T, and the head of $\hat{T}$ consists of the head of T and the elements of H' added after T was constructed.

Then, continue to grow H' from a matching edge connecting the head of $\hat{T}$ and $V \setminus V(\hat{T})$.

If a lollipop or a tadpole becomes contained in a larger lollipop or tadpole, then we remove the contained lollipops and tadpoles from our list of them. That is, we only handle inclusion-wise maximal lollipops and tadpoles.

Below is a full description of the algorithm.

Algorithm Apx2EC

Input. A 3-edge-connected cubic graph $G = (V, E)$.

Output. A 2-edge-connected subgraph H of G with at most $6|V|/5$ edges.

Step 1. Find a 2-factor F of G covering all the 3- and 4-edge cuts. Choose an arbitrary cycle C_0 in F, set $H := C_0$, and then go to Step 2.

Step 2. If $V(H) = V$, then return H. Otherwise, choose an arbitrary edge $e = uv \in \delta(H)$, where $u \in V(H)$ and $v \in V \setminus V(H)$, let H' be the graph consisting of the single vertex u, and go to Step 3.

Step 3. Denote by C the cycle in F containing v. If C is not contained in H, then go to Step 4. Otherwise, go to Step 5.

Step 4. Here, we have the following four cases.

- If C is a large cycle not contained in $H \cup H'$, then go to Step 4-1.
- If C is a small cycle not contained in $H \cup H'$, then go to Step 4-2.
- If C is an independent large cycle in H' or contained in a lollipop in H', then go to Step 4-3.
- If C is an independent small cycle in H' or contained in a tadpole in H', then go to Step 4-4.

Step 4-1. Add e and C to H'. That is, set $V(H') := V(H') \cup V(C)$ and $E(H') := E(H') \cup E(C) \cup \{e\}$. Then choose an arbitrary edge f from $\delta(C) \setminus \{e\}$, set the vertex that is the end of f not in C to v, set $e := f$, and go to Step 3.

Step 4-2. Let P be a Hamilton path of $G[C]$ starting at v (see Lemma 5.1), and denote the terminal vertex of P by w. Add e and P to H', that is, set $V(H') := V(H') \cup V(C)$ and $E(H') := E(H') \cup E(P) \cup \{e\}$. Then let f be the matching edge incident to w, set the vertex that is the other end of f to v, set $e := f$, and go to Step 3.

Step 4-3. Add e to H' and construct a lollipop L. Remove any lollipops or tadpoles contained in L from the list of lollipops and tadpoles. Then, choose an arbitrary edge f from $\delta(L) \setminus E(H')$, set the vertex that is the end of f not in L to v, set $e := f$, and go to Step 3.

Step 4-4. Add e to H' and construct a tadpole T. Remove any lollipops or tadpoles that are part of T from the list of lollipops and tadpoles. Then, choose an arbitrary edge f connecting the head of T to $V \setminus V(T)$, set the vertex that is the end of f not in T to v, set $e := f$, and go to Step 3.

Step 5. Augment H by H' and e. That is, set $V(H) := V(H) \cup V(H')$ and $E(H) := E(H) \cup E(H') \cup \{e\}$. Then, go to Step 2.

What remains to be proven is that we can continue to grow H' from the head of a tadpole in Step 4-4.

Lemma 5.2. *In Step 4-4, there exists an edge connecting the head of the tadpole T and $V \setminus V(T)$.*

Proof. Denote the small cycle providing the tail of T by C_T. Let $Q = V(T) \setminus V(C_T)$ and $R = V \setminus V(T)$. Note that all edges in $\delta(C_T)$, $\delta(Q)$ and $\delta(R)$ are matching edges.

Suppose that no edge connects the head of the tadpole T and $V \setminus V(T)$. Then, there are no edges between Q and R. Since G is 3-edge-connected and F covers all 3- and 4-edge cuts, there are at least five edges between $V(C_T)$ and Q, and at least five edges between $V(C_T)$ and R. It follows that $|C_T| \geq 10$, which contradicts that C_T is a small cycle. $\square$

Step 1 takes $\mathrm{O}(n^3)$ time (see Theorem 4.4). The other steps of the algorithm collectively takes $\mathrm{O}(n)$ time, and thus the total time complexity is $\mathrm{O}(n^3)$. The approximation ratio follows from the following lemma.

Lemma 5.3. *For an output H of Algorithm* APX2EC*, it holds that $|E(H)| \leq \max\{n, 6n/5-1\}$.*

Proof. For $n \leq 9$, the initial 2-factor which has all 3- and 4- edge cuts covered must consist of just one cycle, and thus $|E(H)| = n$.

For $n \geq 10$, partition the family $\mathcal{C}$ of cycles in F into two parts, $\mathcal{L}$ and $\mathcal{S}$, where $\mathcal{L}$ is the family of large cycles and $\mathcal{S}$ is that of small cycles. By construction, it is clear that H contains at most $2(|\mathcal{C}|-1)$ edges from the matching edges $E \setminus F$. Moreover, we use $|C|$ edges from $G[C]$

for a cycle $C \in \mathcal{L}$, and $|C| - 1$ edges from $G[C]$ for a cycle $C \in \mathcal{S}$, except possibly the initial cycle C_0—if it is a small cycle, then we use $|C|$ edges from it. Thus, it follows that

$$|E(H)| \leq n - (|\mathcal{S}| - 1) + 2(|\mathcal{C}| - 1) = n + |\mathcal{S}| + 2|\mathcal{L}| - 1 \leq 6n/5 - 1.$$

The last inequality follows from the definition of small and large cycles, which implies that $n \geq 5|\mathcal{S}| + 10|\mathcal{L}|$. □

As a consequence of the fact that n is a lower bound for 2EC we obtain the following theorem:

Theorem 5.4. *Algorithm* Apx2EC *finds a* $6/5$*-approximate solution for* 2EC *in* 3*-edge-connected cubic graphs in* $\mathrm{O}(n^3)$ *time.*

5.2 The integrality gap for 2EC

In the weighted version of 2EC, which we denote by W2EC, we are given a complete graph $K_n = (V, E)$ with edge weights $w \in \mathbf{R}^E$, and we wish to find a 2-edge-connected spanning subgraph of K_n of minimum weight. We can associate such a weighted problem with our 2EC problem for a given graph G by taking the *metric completion* of G, which is the complete weighted graph obtained by assigning weight 1 to all edges of G, and letting each other edge have weight equal to the length of a shortest path between its endpoints in G. We call this weighted version of 2EC *graph2EC*. Note that the minimum size of a solution for 2EC is equal to the minimum weight solution for its associated graph2EC problem (see [7]).

A related approach for finding approximated 2EC solutions is to study the *integrality gap* $\alpha(\text{2EC})$, which is the worst-case ratio between the optimal value for 2EC (i.e., its associated graph2EC) and the optimal value for the linear programming relaxation of the integer programming formulation of the associated graph2EC. The value $\alpha(\text{2EC})$ gives one measure of the quality of the lower bound provided by the linear programming relaxation for the problem. Moreover, a polynomial-time constructive proof for the value $\alpha(\text{2EC})$ would provide an $\alpha(\text{2EC})$-approximation algorithm for 2EC.

Since n can be shown to be a lower bound for the optimal value of the linear programming relaxation for graph2EC on cubic 3-edge-connected graphs, an immediate consequence of Algorithm Apx2EC is the following:

Theorem 5.5. *The value of* $\alpha(\text{2EC})$ *is at most* $6/5$ *for* 3*-edge-connected cubic graphs.*

This result is significant in that it has been conjectured that the integrality gap $\alpha(\text{W2EC})$ for general W2EC is 6/5, and moreover 6/5 is the worst lower bound known for $\alpha(\text{W2EC})$ (see [2]). Previously, the best upper bound known for $\alpha(\text{2EC})$ was 5/4, which is implied by Huh's algorithm in [18]. We remark that the worst lower bound we know for $\alpha(\text{2EC})$ for bridgeless cubic graphs is 7/6, and for 3-edge connected cubic graphs is 11/10.

6 A faster approximation algorithm

Let $G = (V, E)$ be a 3-edge-connected cubic graph with $|V| = n$. In this section, we describe an algorithm for finding a 2-edge-connected subgraph of G with at most $6n/5$ edges in $\mathrm{O}(n^2 \log^4 n)$ time.

In this algorithm, we begin with a 2-factor F which covers all the 3-edge cuts and consists of cycles of size at least five (see § 6.2 for details). Note that this is a relaxed condition for the initial 2-factor F in Algorithm Apx2EC. This relaxation improves the time complexity for finding F, which is the bottleneck part of the entire algorithm, but makes the following procedures more involved.

We inherit the terminology from Algorithm Apx2EC. The family of cycles in F is denoted by $\mathcal{C}$. The edges in $E \setminus F$, which form a perfect matching in G, are called the *matching edges.* We say that a cycle $C \in \mathcal{C}$ is *large* if $|C| \geq 10$, and *small* if $|C| \leq 9$.

6.1 Differences from Algorithm Apx2EC

We basically execute the same procedures as Algorithm Apx2EC. For the current initial 2-factor F, however, Lemma 5.2 fails due to the fact that F does not necessarily covers all 4-edge cuts. In this subsection, we exhibit tricks which ensures that a tadpole T has a matching edge connecting its head and $V \setminus V(T)$.

For a cycle C, let $G - C$ denote the subgraph of G obtained by deleting $G[C]$ and $\delta(C)$. If $G - C$ is disconnected, then C is called a *cut cycle*. Let T be a tadpole and let C denote the small cycle providing the tail of T. If C is a cut cycle, then we do not know whether there exists an edge connecting the head of T and $V \setminus V(T)$. In order to ensure the existence of such an edge, we impose two new rules in constructing H.

In preparation for describing the rules, let us analyze properties of small cut cycles.

Lemma 6.1. *A small cut cycle $C \in \mathcal{C}$ has the following properties*:

- *$G - C$ consists of exactly two components*;
- *$|C| = 8$ or $|C| = 9$; and*
- *C has no chords.*

Proof. Since G is 3-edge-connected and F covers all the 3-edge cuts, each component in $G - C$ is connected to C by at least four edges in G. As C is a small cycle, we have that $|\delta(C)| \leq 9$, which implies that the number of components in $G - C$ at most two. Thus, we have that $G - C$ consists of exactly two components and $|C|$ is eight or nine. Furthermore, we have that $|\delta(C)| \geq 8$ and $|C| \leq 9$, which implies that C has no chords. □

When we find the initial 2-factor F and choose the initial cycle C_0, for each small cut cycle C other than C_0, we label each edge in $\delta(C)$ as *backward* or *forward*. Denote the component in $G - C$ containing C_0 by H_0, and the other by H_1. We call the matching edges connecting H_0 and C *backward edges*, and those connecting H_1 and C *forward edges*. The lemma below immediately follows from Lemma 6.1.

Lemma 6.2. *For a small cut cycle C, the number of forward edges is four or five.*

One new rule is for the choice of a Hamilton path in $G[C]$ for a small cut cycle C. When we come to a small cycle C by traversing a matching edge $e = uv$ with $v \in V(C)$, we route a Hamilton path P in $G[C]$ and leave C from a matching edge $f \in \delta(C)$. We call this edge f as the *leaving edge* of P. Now, if C is a small cut cycle, Lemma 6.1 implies that $G[C]$ has two Hamilton paths starting at v, which end at a vertex adjacent to v on C. Here, we impose a rule on the choice between these two Hamilton paths.

Trick 1. If at least one of the Hamilton paths has a backward leaving edge, then choose it. Otherwise, choose a Hamilton path whose leaving edge is closer to a backward edge other than e on C than that of the other Hamilton path.

The other new rule is in construction of a tadpole. When we come back to a small cut cycle C, we construct a lollipop instead of a tadpole if a certain condition is satisfied.

Trick 2. Let $uv, ij, kl \in \delta(C)$ be matching edges used to come to C, leave C and come back to C, respectively, where $v, i, k \in V(C)$. If i and k are adjacent on the Hamilton path of $G[C]$ contained in H', then we construct a lollipop L instead of the tadpole T as follows. The edge ik is deleted from H' and vi is added to H'. The vertex set of L is the same as that of T, and the edge set of L is $E(L) = (E(T) \setminus \{ik\}) \setminus \{vi\}$. See Figure 3 for an illustration.

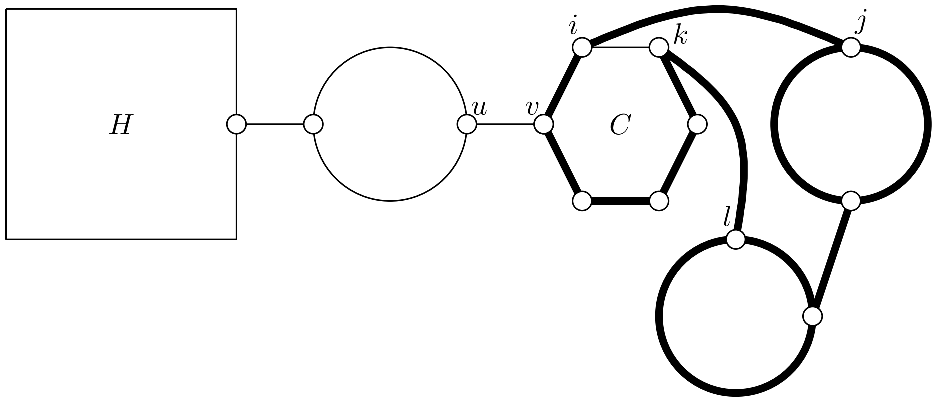

Figure 3: Construction of a lollipop L by Trick 2. The thick edges are the edges in L. The hexagon is the small cut cycle C of size six, and the three circles indicate large cycles. (Some vertices and edges are omitted.)

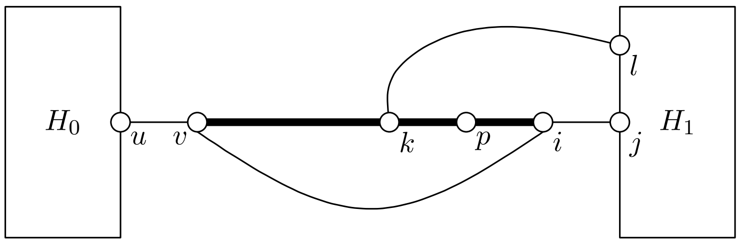

Figure 4: The path P, indicated by the thick line, and edge vi form the small cut cycle C_T. The subgraph of P connecting v and k is the tail of T.

Now, we prove that the algorithm does not get stuck in leaving a tadpole from its head.

Lemma 6.3. *For a tadpole T, there exists an edge connecting the head of the tadpole T and $V \setminus V(T)$.*

Proof. Suppose, to the contrary, that there exists no edge connecting the head of a tadpole T and $V \setminus V(T)$. Denote the small cycle providing the tail of T by C_T, the chosen Hamilton path of $G[C_T]$ by P, and the matching edge used to come to C_T at the first time by $m_T = uv$, where $v \in V(C_T)$. Here, we have that C_T is a small cut cycle. Recall that H_0 is the component in $G - C_T$ containing the initial cycle C_0 and H_1 is the other component. Observe that the rest of the tadpole forms H_1, that is, $V(T) \setminus V(C_T) = V(H_1)$. (See Figure 4 for an illustration.)

Denote the forward edge which is used to leave C_T by ij and the forward edge which is used to come back to C_T to form the tadpole T by kl, where $i, k \in V(C_T)$ and $j, l \in V(H_1)$. By Trick 2, there exists at least one vertex between i and k on P. Let p denote one of such vertices. Note that the matching edge incident to p is a forward edge, since p belongs to the head of T.

Thus, there exist at least two vertices on P between ij and a backward edge. By Trick 1, it implies that the matching edges incident to three vertices closest to v on P are forward edges. Therefore, C_T has at least six forward edges, which contradicts Lemma 6.2. □

6.2 Finding the initial 2-factor

In this subsection, we describe how to find a 2-factor which covers all the 3-edge cuts and consists of cycles of size at least five in a 3-edge-connected cubic graph efficiently. For simplicity, assume that G is not K_4.

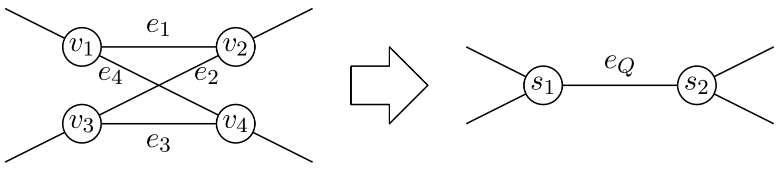

Figure 5: Shrinking a cycle of size four.

First, take a maximal family $\mathcal{Q}$ of vertex-disjoint cycles of size four. Then, we "shrink" each cycle $Q \in \mathcal{Q}$ in the following manner. Let $Q = (v_1, e_1, v_2, e_2, v_3, e_3, v_4, e_4, v_1)$. Delete all edges in $E(Q)$, merge v_1 and v_3 into a single vertex s_1, v_2 and v_4 into another vertex s_2, and connect s_1 and s_2 by a new edge e_Q. We refer to s_1, s_2 as *pseudo-vertices*, and e_Q as a *pseudo-edge*. See Figure 5 for illustration. Denote the graph obtained by shrinking every cycle in $\mathcal{Q}$ by $G' = (V', E')$. Note that G' is a 2-edge-connected cubic graph.

In G', we find a 2-factor $F' \subseteq E'$ which covers all the 3-edge cuts. More specifically, we describe a recursive algorithm to find a 2-factor F' covering all the proper 3-edge cuts and excluding a specified edge e^* in a 2-edge-connected cubic graph G'. If G' has no proper 3-edge cut, then it suffices to find an arbitrary 2-factor excluding e^*, which can be done in $\mathrm{O}(n \log^4 n)$ time [4]. Suppose G' has a proper 3-edge cut $D = \delta(S)$. We construct two graphs, $G' \times S$ and $G' \times \bar{S}$. Without loss of generality, suppose that $G' \times S$ contains e^*. Then, we apply the present algorithm recursively to $G' \times S$ to obtain a 2-factor F° covering all the 3-edge cuts and excluding e^* in $G' \times S$. Identify the unique edge e in $D \setminus F^\circ$. Then, we again apply the present algorithm to $G' \times \bar{S}$ recursively to obtain a 2-factor $F^\bullet$ in $G' \times \bar{S}$ convering all 3-edge cuts in $G' \times \bar{S}$ and excluding e. Now, by Lemma 3.1, $F' = F^\circ \cup F^\bullet$ is a 2-factor covering all the 3-edge cuts in G'.

In order to find a proper 3-edge cut D in G', perform the following procedure. Find an arbitrary 2-factor F' and then contract each cycle in F' into a vertex to obtain a new graph G''. Then find an arbitrary 3-edge cut in G''. Now, if a 3-edge cut in G'' is found, then it corresponds to a proper 3-edge cut in G'. If G'' has no 3-edge cut, then we can conclude that F' is a 2-factor covering all 3-edge cuts in G'.

Then, we unshrink all the pseudo-edges sequentially to recover G. In unshrinking a pseudo-edge e_Q, we extend F' by edges in $E(Q)$ in the following manner.

Case 1 ($e_Q \in F'$). Denote the cycle in F' containing e_Q by C. We choose the unique triple of edges in $E(Q)$ which provides a cycle C_Q of size $|C| + 2$.

Case 2 ($e_Q \notin F'$).

Case 2.1. If s_1 and s_2 belongs to an identical cycle C in F', then choose the pair of disjoint edges in $E(Q)$ which provides a cycle C_Q of size $|C| + 2$.

Case 2.2. If s_1 and s_2 belongs to distinct cycles C_1 and C_2 in F', then choose any of the pair of disjoint edges in $E(Q)$ to obtain a cycle C_Q of size $|C_1| + |C_2| + 2$.

See Figure 6 for an illustration. In the end, we obtain a desired 2-factor F in G.

Lemma 6.4. *The* 2*-factor F constructed as above covers all the* 3*-edge cuts and consists of cycles of size at least five.*

Proof. We first prove that F consists of cycles of size at least five. It is not difficult to see that F' does not contain a cycle of size less than five without pseudo-vertices. Let e_Q be a pseudo-edge in G' resulting from $Q \in \mathcal{Q}$, and let G'_Q and F'_Q be the graph and 2-factor obtained from G' and F' by unshrinking e_Q, respectively. In Cases 1 and 2.1, $|C_Q| < 5$ implies that $|C| = 2$. Then, G

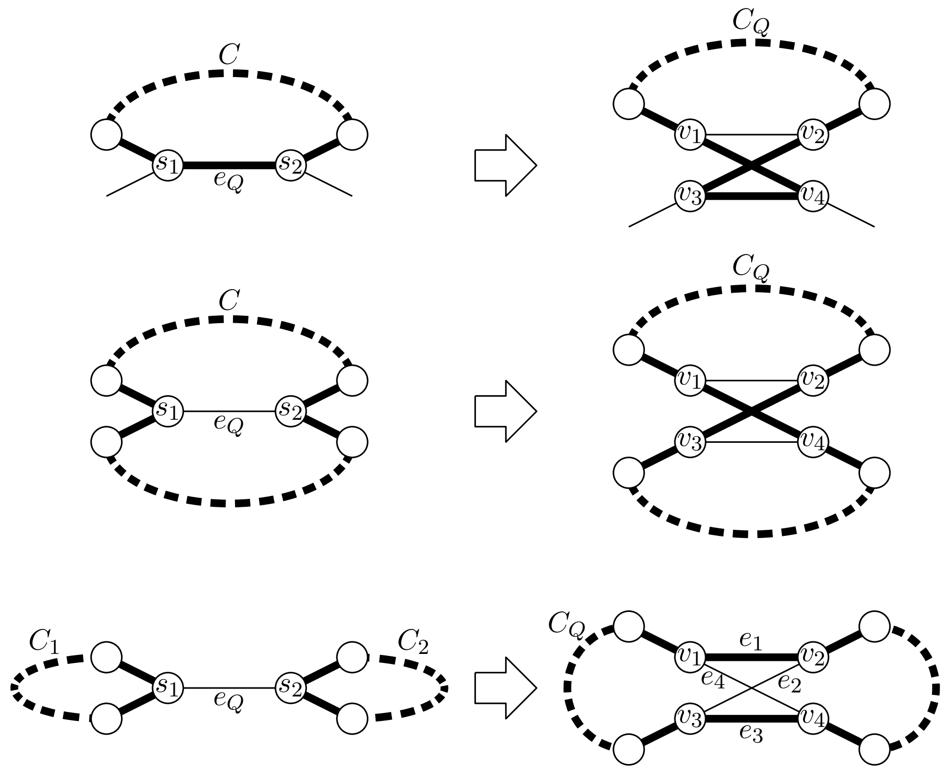

Figure 6: Unshrinking a pseudo-edge. Thick edges are in F' and F. The three patterns indicate Case 1, 2.1 and 2.2, respectively. In Case 2.2, it is also possible to add e_2 and e_4 to F' instead of e_1 and e_3.

has a 2-edge cut, which is a contradiction. In Case 2.2, if $|C_Q| < 5$, then $|C_1| = |C_2| = 1$, which implies that G is K_4.

We now prove is that F'_Q covers all the 3-edge in G'_Q. Since F' covers all the 3-edge cuts in G', it suffices to consider 3-edge cuts in G'_Q containing edges in $E(Q)$. In both Cases 1 and 2, such an edge cut is covered by C_Q. We inductively conclude that F covers all the 3-edge cuts in G. □

The procedures to find the initial 2-factor are summarized as follows.

Procedure Ini2F

Input. A 3-edge-connected cubic graph $G = (V, E)$.

Output. A 2-factor $F \subseteq E$ in G which covers all the 3-edge cuts and consists of cycles of size at least five.

Step 1. Let $\mathcal{Q}$ be a maximal family of vertex-disjoint cycles of size four. Shrink each cycle $Q \in \mathcal{Q}$ to a pseudo-edge e_Q to obtain a 2-edge-connected cubic graph G'.

Step 2. Apply Procedure 3Cut below to obtain a 2-factor F' covering all the 3-edge cuts in G'.

Procedure 3Cut

Step 2.1 Find an arbitrary 2-factor F' in G' and contract each cycle in F' into a single vertex to obtain a new graph G''.

Step 2.2. Find an arbitrary 3-edge cut in G''. If no 3-edge cut exists, return F'.

Step 2.3. Let $D = \delta(S)$ be a proper 3-edge cut in G' which corresponds to the 3-edge cut in G'' found in Step 2.2. Construct $G' \times S$ and $G' \times \bar{S}$. Apply Procedure 3Cut to $G' \times S$ recursively to obtain a 2-factor F° covering all the 3-edge cuts. Let e be the unique edge in $D \setminus F^\circ$ and again apply Procedure 3Cut to $G' \times \bar{S}$ to obtain a 2-factor $F^\bullet$ covering all the 3-edge cuts and excluding e. Return $F' = F^\circ \cup F^\bullet$.

Step 3. Unshrink all the pseudo-edges e_Q sequantially to obtain a desired 2-factor F.

Denote the time complexity of Procedure Ini2F by $T(n)$. Then, we have that

$$T(n) = T(n_1) + T(n_2) + f(n),$$

where n_1 and n_2 denote the number of vertices in $G \times S$ and $G \times \bar{S}$, respectively, and $f(n)$ denotes the time complexity for finding a proper 3-edge cut in a cubic graph with n vertices. Note that finding $\mathcal{Q}$ can be done in $\mathrm{O}(n)$ time. As stated above, finding a proper 3-edge cut can be done by finding a 2-factor and finding a 3-edge cut. The former can be done in $\mathrm{O}(n \log^4 n)$ time [4], and the latter in $\mathrm{O}(n \log n)$ [13] by trying to find four edge-disjoint arborescences in a digraph obtained by replacing each edge by two oppositely directed edges. Consequently, we have that $f(n) = \mathrm{O}(n \log^4 n)$ and $T(n) = \mathrm{O}(n^2 \log^4 n)$.

6.3 Algorithm description

We now show a precise description of Algorithm FastApx2EC.

Algorithm FastApx2EC

Input. A 3-edge-connected cubic graph $G = (V, E)$.

Output. A 2-edge-connected subgraph H of G with at most $6n/5$ edges.

Step 1. Apply Procedure Ini2F to obtain a 2-factor F of G covering all the 3-edge cuts and consisting of cycles of size at least five. Choose an arbitrary cycle C_0 in F, put $H := C_0$, and then go to Step 2.

Step 2. If $V(H) = V$, then return H and halt. Otherwise, choose an arbitrary edge $e = uv \in \delta(H)$, where $u \in V(H)$ and $v \in V \setminus V(H)$, let H' be a graph consisting of the single vertex u, and then go to Step 3.

Step 3. Let C denote the cycle in F containing v. If C is not contained in H, then go to Step 4. Otherwise, go to Step 5.

Step 4. Here, we have the following four cases.

- If C is a large cycle not contained in $H \cup H'$, then go to Step 4-1.
- If C is a small cycle not contained in $H \cup H'$, then go to Step 4-2.
- If C is an independent large cycle in H' or contained in a lollipop in H', then go to Step 4-3.
- If C is an independent small cycle in H' or contained in a tadpole in H', then go to Step 4-4.

Step 4-1. Add e and C to H'. That is,

$$V(H') := V(H') \cup V(C),$$
$$E(H') := E(H') \cup E(C) \cup \{e\}.$$

Then, choose an arbitrary edge f from $\delta(C) \setminus \{e\}$, set the vertex that is the end of f not in C to v, set $e := f$, and go to Step 3.

Step 4-2. Let P be a Hamilton path of $G[C]$ starting at v and ending at a vertex w not incident to a chord of C (see Lemma 5.1). Add e and P to H', that is,

$$V(H') := V(H') \cup V(C),$$
$$E(H') := E(H') \cup E(P) \cup \{e\}.$$

Then, let f be the matching edge incident to w, set the vertex that is the other end of f to v, set $e := f$, and go to Step 3.

Step 4-3. Add e to H' and construct a lollipop L. Remove any lollipops or tadpoles contained in L from the list of lollipops and tadpoles. Then, choose an arbitrary edge $f \in \delta(L) \setminus E(H')$, set the vertex that is the end of f not in L to v, set $e := f$, and go to Step 3.

Step 4-4. Let P denote the Hamilton path in $G[C]$ contained in H' and let i denote the terminal vertex of P. If v is adjacent to i on P, then go to Step 4-3. Otherwise, add e to H' and construct a tadpole T. Remove any lollipops or tadpoles that are contained in T from the list of lollipops and tadpoles. Then, choose an arbitrary edge f connecting the head of T to $V \setminus V(T)$, set the vertex that is the end of f not in T to v, set $e := f$, and go to Step 3.

Step 5. Add H' and e to H. That is,

$$V(H) := V(H) \cup V(H'),$$
$$E(H) := E(H) \cup E(H') \cup \{e\}.$$

Then, go to Step 2.

Above is a complete description of our algorithm. It is straightforward to see that the approximation ratio of Algorithm FastApx2EC is 6/5 and the running time is $O(n^2 \log^4 n)$.

Theorem 6.5. *Algorithm* FastApx2EC *finds a* 6/5*-approximate solution for* 2EC *in* 3*-edge-connected cubic graphs in* $O(n^2 \log^4 n)$ *time.*

References

[1] N. Aggarwal, N. Garg, and S. Gupta, *A 4/3-approximation for TSP on cubic 3-edge-connected graphs*, manuscript, 2011.

[2] A. Alexander, S. Boyd, and P. Elliott-Magwood, *On the integrality gap of the 2-edge connected subgraph problem*, Technical Report TR-2006-04, SITE, University of Ottawa, Ottawa, Canada, 2006.

[3] K. Bérczi and L. A. Végh, *Restricted b-matchings in degree-bounded graphs*, in: Proceedings of the Fourteenth International IPCO Conference, Lecture Notes Comput. Sci., 6080 (2010), pp. 43–56.

[4] T. C. Biedl, P. Bose, E. D. Demaine, and A. Lubiw, *Efficient algorithms for Petersen's matching theorem*, J. Algorithms, 38 (2001), pp. 110–134.

[5] S. Boyd and R. Carr, *A new bound for the ratio between the 2-matching problem and its linear programming relaxation*, Math. Program., Ser. A, 86 (1999), pp. 499–514.

[6] S. Boyd, R. Sitters, S. van der Ster, and L. Stougie, *TSP on cubic and subcubic graphs*, in: Proceedings of the Fifteenth International IPCO Conference, Lecture Notes Comput. Sci., 6655 (2011), pp. 65–77.

[7] J. Cheriyan, A. Sebő, and Z. Szigeti, *Improving on the* 1.5 *approximation of a smallest* 2*-edge connected spanning subgraph*, SIAM J. Discrete Math., 14 (2001), pp. 170–180.

[8] G. Cornuéjols, D. Naddef, and W. Pulleyblank, *The traveling salesman problem in graphs with* 3*-edge cutsets*, J. ACM, 32 (1985), pp. 383–410.

[9] B. Csaba, M. Karpinski, and P. Krysta, *Approximability of dense and sparse instances of minimum* 2*-connectivity, TSP and path problems*, in: Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2002, pp. 74–83.

[10] J. Edmonds, *Maximum matching and a polyhedron with* $0,1$*-vertices*, J. Res. Natl. Bur. Stand., Sect. B, 69 (1965), pp. 125–130.

[11] D. Fotakis and P. Spirakis, *Graph properties that facilitate travelling*, Electron. Colloq. Computat. Complexity, 31 (1998), pp. 1–18.

[12] H. N. Gabow, *Data structures for weighted matching and nearest common ancestors with linking*, in: Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms, 1990, pp. 434–443.

[13] H. N. Gabow, *A matroid approach to finding edge connectivity and packing arborescences*, J. Comput. Systems Sci., 50 (1995), pp. 259–273.

[14] D. Gamarnik, M. Lewenstein, and M. Sviridenko, *An improved upper bound for the TSP in cubic* 3*-edge-connected graphs*, Oper. Res. Lett., 33 (2005), pp. 467–474.

[15] D. Hartvigsen, *Extensions of Matching Theory*, Ph. D. Thesis, Carnegie Mellon University, Pittsburgh, PA, 1984.

[16] D. Hartvigsen and Y. Li, *Triangle-free simple* 2*-matchings in subcubic graphs*, in: Proceedings of the Twelfth International IPCO Conference, Lecture Notes Comput. Sci., 4513 (2007), pp. 43–52.

[17] D. Hartvigsen and Y. Li, *Maximum cardinality simple* 2*-matchings in subcubic graphs*, manuscript.

[18] W. T. Huh, *Finding* 2*-edge connected spanning subgraphs*, Oper. Res. Lett., 32 (2004), pp. 212–216.

[19] R. Jothi, B. Raghavachari, and S. Varadarajan, *A* $5/4$*-approximation algorithm for minimum* 2*-edge-connectivity*, in: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2003, pp. 725–734.

[20] T. Kaiser, D. Král', and S. Norine, *Unions of perfect matchings in cubic graphs*, Electron. Notes Discrete Math., 22 (2005), pp. 341–345.

[21] T. Kaiser and R. Škrekovski, *Cycles intersecting edge-cuts of prescribed sizes*, SIAM J. Discrete Math., 22 (2008), pp. 861–874.

[22] D. R. Karger and M. S. Levine, *Finding maximum flows in undirected graphs seems easier than bipartite matching*, in: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, 1998, pp. 69–78.

[23] Y. Kobayashi, *A simple algorithm for finding a maximum triangle-free* 2*-matching in subcubic graphs*, Discrete Optim. 7 (2010), pp. 197–202.

[24] P. Krysta and V. S. Anil Kumar, *Approximation algorithms for minimum size* 2*-connectivity problems*, in: Proceedings of the Eighteenth Annual Symposium on Theoretical Aspects of Computer Science, 2001, pp. 431–442.

[25] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, eds., *The Traveling Salesman Problem—A Guided Tour of Combinatorial Optimization*, Wiley, Chichester, 1985.

[26] D. Naddef and W. R. Pulleyblank, *Matchings in regular graphs*, Discrete Math., 34 (1981), pp. 283–291.

[27] H. Nagamochi and T. Ibaraki, *A linear-time algorithm for finding a sparse k-connected spanning subgraph on a k-connected graph*, Algorithmica, 7 (1992), pp. 583–596.

[28] C. Papadimitriou and M. Yannakakis, *The traveling salesman problem with distances one and two*, Math. Oper. Res., 18 (1993), pp. 1–11.

[29] J. Petersen, *Die Theorie der regulären graphen*, Acta Math., 15 (1891), pp. 193–220.

[30] T. Schönberger, *Ein Beweis des Petersenschen Graphensatzes*, Acta Litt. Sci. Szeged, 7 (1934–1935), pp. 51–57.

[31] S. Vempala and A. Vetta, *Factor* 4/3 *approximations for minimum* 2*-connected subgraphs*, in: Proceedings of the Third Workshop on Approximation Algorithms for Combinatorial Optimization, Lecture Notes Comput. Sci., 1913 (2000), pp. 262–273.

[32] O. Vornberger, *Easy and hard cycle covers*, preprint, Universität Paderborn, 1980.