FINDING 2-FACTORS CLOSER TO TSP TOURS IN CUBIC GRAPHS[*]

SYLVIA BOYD[†], SATORU IWATA[‡], AND KENJIRO TAKAZAWA[§]

Abstract. In this paper we are interested in algorithms for finding 2-factors that cover certain prescribed edge-cuts in bridgeless cubic graphs. Since a Hamilton cycle is a 2-factor covering all edge-cuts, imposing the constraint of covering those edge-cuts makes the obtained 2-factor closer to a Hamilton cycle. 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 spanning 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 on the subsequent processes.

1. Introduction. The classical theorem of Petersen [27] 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 section 3 we present a polyhedral description of 2-factors covering all 3-edge cuts, as well as that of perfect matchings intersecting all 3-edge cuts in exactly one edge, in bridgeless cubic graphs. In section 4 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, Note that this represents the first polynomial-time algorithm for this problem. In section 5 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 section 6 we demonstrate the usefulness of our results by using them to facilitate a new 6/5-approximation algo-

rithm 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 [17, 18].

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 [15] 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, unless G is K_4.

There exist polynomial-time algorithms for finding a minimum-weight triangle-free 2-factor in weighted subcubic graphs [16, 22, 31]. 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 [14]. Polynomial-time algorithms are known for finding a triangle- and square-free 2-factor in subcubic graphs [3, 15]. For general graphs, the complexity of this problem is also unknown. Moreover, Vornberger [31] 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, 10, 26]). 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 [9] that any bridgeless cubic graph has a 2-factor that covers all 3-edge cuts (see section 3), a fact which is shown in [19] and also shown and used in [5] to get the first 4/3-approximation for graph-TSP. Furthermore, Kaiser and Škrekovski [20] 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 section 5 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), or 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 [24]). 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 that provided by a minimum-weight 2-factor, or that 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 [13] 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 [5]. 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 of dealing with the cases of cycles of length five with chords.

In section 6 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 [8]) and has been widely investigated (see [18] for an overview). Krysta and Anil Kumar [23] designed a 21/16-approximation algorithm for cubic graphs, and Csaba, Karpinski, and Krysta [8] 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 [17]. For 2EC in general graphs, Vempala and Vetta [30] and Jothi, Raghavachari, and Varadarajan [18] claimed to have 4/3- and 5/4-approximation algorithms, respectively. A recent paper of Sebő and Vygen [29] contains, among other results, a 4/3-approximation algorithm for 2EC. Interestingly, Vempala and Vetta [30] mentioned 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 section 7). 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* $\alpha(2EC)$, 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 $\alpha(2EC)$ 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 $\alpha(2EC)$ would provide an $\alpha(2EC)$-approximation algorithm for 2EC. For the metric completion of 3-edge-connected cubic graphs, Huh's results [17] imply that $\alpha(2EC)$ is at most 5/4. In section 6 we improve upon this and show that $\alpha(2EC)$ for this class of graphs is at most 6/5. This is significant, in that it has been conjectured that $\alpha(2EC) = 6/5$ 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. In section 2 we provide some terminology and precise descriptions of the basic facts about 2-factors. A polyhedral description of 2-factors covering all 3-edge cuts in bridgeless cubic graphs is given in section 3. In section 4 we present Algorithm W3Cut, an algorithm for finding a minimum-weight 2-factor covering all 3-edge cuts in weighted bridgeless cubic graphs. In section 5 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 section 6 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 $\alpha(2\text{EC})$. Finally, in section 7 we give a faster $\text{O}(n^2 \log^4 n)$-time algorithm for the same problem with the same approximation ratio.

2. Preliminaries. In this section we provide precise definitions and notation and a formal description of basic results which motivated our work. 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 multiedges (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 $\bar{S}$. We denote $|\delta(S)|$ by $d(S)$. For each vertex $v \in V$, $\delta(\{v\})$ and $d(\{v\})$ are simply denoted by $\delta(v)$ and $d(v)$, respectively. A graph in which $d(v) = 3$ 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/S be the graph obtained by shrinking S into a single pseudovertex 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 $F \cap D \neq \emptyset$.

Petersen's theorem [27] states that every bridgeless cubic graph contains a perfect matching. Schönberger [28] proved the following strengthened form of Petersen's theorem.

THEOREM 2.1 (Schönberger [28]). *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 $\text{O}(n \log^4 n)$ 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^*.*

Kaiser and Škrekovski [20] proved a still stronger theorem, for which we will give an algorithmic proof in section 5.

THEOREM 2.3 (Kaiser and Škrekovski [20]). *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. Polyhedral results. For any edge set $F \subseteq E$, the *incidence vector of F* is the vector $\chi^F \in \mathbf{R}^E$ defined by $\chi_e^F = 1$ for $e \in F$ and $\chi_e^F = 0$ 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 [9], Edmonds shows that $P^{2F}(G)$ is given by the following linear system:

(3.1) $x(\delta(v)) = 2$ for all $v \in V$,

(3.2) $x(X) - x(\delta(S) \setminus X) \leq |X| - 1$ for all $S \subset V$,

                                                   $X \subseteq \delta(S)$, X a matching, $|X|$ odd,

(3.3) $0 \leq x_e \leq 1$ for all $e \in E$.

In the following theorem, we extend this result by giving a linear description of the convex hull of all incidence vectors of G covering all 3-edge cuts, denoted by $P^{2F3}(G)$.

THEOREM 3.1. *The polytope $P^{2F3}(G)$ is given by the system of constraints* (3.1)–(3.3) *for the 2-factor polytope $P^{2F}(G)$, with the addition of the following constraints:*

(3.4) $x(\delta(S)) = 2$ *for all $S \subset V$, $\delta(S)$ a proper 3-edge cut of G.*

Proof. Let P be the polytope defined by the system of constraints (3.1)–(3.3) and constraints (3.4). Clearly $P^{2F3}(G) \subseteq P$. To show that $P \subseteq P^{2F3}(G)$ 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 $F_1, \ldots, F_k$ of G and positive real numbers $\lambda_1, \ldots, \lambda_k$ such that

(3.5) $$x^* = \sum_{i=1}^{k} \lambda_i \chi^{F_i} \quad \text{and} \quad \sum_{i=1}^{k} \lambda_i = 1.$$

Now consider any proper 3-edge cut $\delta(S)$ of G. Clearly $\chi^{F_i}(\delta(S)) \leq 2$ for $i = 1, \ldots, k$ 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 (3.5) that we must have $\chi^{F_i}(\delta(S)) = 2$ for $i = 1, \ldots, k$. Consequently x^* is a convex combination of incidence vectors of 2-factors that cover all 3-edge cuts of G. $\square$

It is not difficult to see that $\frac{2}{3}\chi^E \in P^{2F3}(G)$; in particular, $P^{2F3}(G) \neq \emptyset$ for bridgeless cubic graphs. Thus, the following existence theorem, which is a weaker statement than Theorem 2.3, directly follows from Theorem 3.1.

COROLLARY 3.2. *Let $G = (V, E)$ be a bridgeless cubic graph with specified edge $e^* \in E$. Then there exists a 2-factor of G that covers all 3-edge cuts in G and does not contain e^*.*

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 [9]:

(3.6) $x(\delta(v)) = 1$ for all $v \in V$,

(3.7) $x(\delta(S)) \geq 1$ for all $S \subset V$, $|S|$ odd,

(3.8) $x_e \geq 0$ for all $e \in E$.

Using this linear system, we have the following theorem, which can be proved similarly to Theorem 3.1.

THEOREM 3.3. *The polytope $P^{PM3}(G)$ is given by the system of constraints* (3.6)–(3.8) *for the perfect matching polytope $P^{PM}(G)$, with the addition of the following constraints:*

(3.9) $x(\delta(S)) = 1$ *for all $S \subset V$, $\delta(S)$ a proper 3-edge cut of G.*

4. 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 $O(n^3)$-time algorithm for finding a minimum-weight 2-factor that covers all the 3-edge cuts in G. Note that such a 2-factor always exists by Corollary 3.2.

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 is an adaptation of one used in [7], and it involves choosing a proper 3-edge cut $D = \delta(S)$, finding a 2-factor F' in G/S covering all 3-edge cuts and a 2-factor F° in $G/\bar{S}$ covering all 3-edge cuts such that F' and F° use the same two edges of D, and then "gluing" them together. The fact that the resulting 2-factor covers all 3-edge cuts in G is established by Lemma 4.1 below.

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

$$(4.1) \qquad\qquad d(A) + d(B) \geq d(A \cup B) + d(A \cap B),$$
$$(4.2) \qquad\qquad d(A) + d(B) \geq d(A \setminus B) + d(B \setminus A).$$

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

LEMMA 4.1. *Let $D = \delta(S)$ be a proper 3-edge cut of a bridgeless cubic graph G, F' be a 2-factor in G/S covering all 3-edge cuts, and F° be a 2-factor in $G/\bar{S}$ 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 $d(S) = 3$ 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 $d(S \cap Z)$ and $d(S \cup Z)$ are odd. Since G is bridgeless, the submodular property (4.1) implies that $d(S \cap Z) = 3$ and $d(S \cup Z) = 3$. Hence $\delta(S \cap Z)$ is a 3-edge cut in $G/\bar{S}$ and $\delta(S \cup Z)$ is a 3-edge cut in G/S, and it follows by the definition of F that $d_F(S \cap Z) = 2$, $d_F(S \cup Z) = 2$, and $d_F(S) = 2$, which implies $d_F(Z) \geq 2$ by the submodular property (4.1) of $d_F(.)$. $\quad\square$

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 $G/\bar{S}$ not containing edge e_i. The existence of F_i° follows from Corollary 2.2, and F_i° can be found by applying a minimum-weight 2-factor algorithm to the graph obtained by deleting e_i from $G/\bar{S}$.

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/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/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/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/S with weights c'. Let e_{i*} be the edge in $D \setminus F'$, and return $F = F' \cup F^\circ_{i*}$.

The minimality of $c(F)$ in Step 4 is verified as follows. By the minimality of S, one can apply Lemma 4.1, and hence F is a 2-factor in G covering all 3-edge cuts in G. Let $\hat{F}$ be any 2-factor in G that covers all 3-edge cuts in G and $\hat{F}'$ be a 2-factor in G/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

$$T(n) = T(n - l + 2) + 3T'(l) + f(n),$$

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 minimum-weight 2-factor problem in $G/\bar{S}$ with l vertices. We obtain $f(n) = O(n^2)$ as follows. We can assume that G is 3-edge-connected. (Otherwise, it suffices to find a 2-edge cut $\delta(U) = \{u_1 v_1, u_2 v_2\}$, where $u_1, u_2 \in U$, and then find a desired proper 3-edge cut in two graphs: one is $G[U]$ plus edge $u_1 u_2$ and the other $G[\bar{U}]$ plus $v_1 v_2$.) 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 $O(n \log n)$ time [12], finding three edge-disjoint r-arborescences in $O(n \log n)$ time [12], and decomposing the residual graphs for all vertices in $V \setminus \{r\}$ in $O(n^2)$ time, and thus we obtain $f(n) = O(n^2)$. Also, we have $T'(l) = O(l^2 \log l)$ [11], and consequently $T(n) = O(n^3)$.

THEOREM 4.2. *Algorithm* W3CUT *finds a minimum-weight* 2-*factor in G covering all the* 3-*edge cuts in* $O(n^3)$ *time.*

Since the complement of a 2-factor in G is a perfect matching, the following is an immediate consequence of Theorem 4.2.

COROLLARY 4.3. *Algorithm* W3CUT *finds a maximum-weight perfect matching in G which intersects every* 3-*edge cut in G in exactly one edge in* $O(n^3)$ *time.*

5. 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 2.3. 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.

5.1. Covering 3-edge cuts. Our algorithm starts by finding a proper 3-edge cut $\delta(S)$. We will discuss later in section 5.2 what to do if G is free from proper 3-edge cuts. Once an appropriate S is found, the algorithm decomposes G into G/S

and $G/\bar{S}$. Note that both G/S and $G/\bar{S}$ are bridgeless cubic graphs. Without loss of generality, suppose that G/S contains e^*. We then apply the algorithm recursively to G/S to obtain a 2-factor F' covering all the 3- and 4-edge cuts in G/S and excluding e^*. Let f and f' be the two edges in $F' \cap \delta(v_S)$. Again apply the algorithm recursively to $G/\bar{S}$ to obtain a 2-factor F° with $f, f' \in F^\circ$ covering all the 3- and 4-edge cuts in $G/\bar{S}$. Then $F = F^\circ \cup F'$ forms a 2-factor in G. The algorithm terminates by returning F.

LEMMA 5.1. *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 4.1, 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 $d(S)$ 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 $d(S \cap Z)$ is even and $d(S \setminus Z)$ is odd. Since G is bridgeless, we have $d(S \cup Z) \geq 2$, and thus $d(S \cap Z) \leq 5$ by the submodular property (4.1). Thus $d(S \cap Z) = 2$ or $d(S \cap Z) = 4$ holds.

Suppose $d(S \cap Z) = 2$. Consider $d(Z \setminus S)$. It must be even since $|Z \setminus S|$ is even, and thus $d(Z \setminus S)$ is either 2 or 4 by the submodular property (4.2). Since no proper subset of $\delta(Z)$ forms a cut, we have $d(Z \setminus S) + d(S \cap Z) > d(Z)$. Thus $d(Z \setminus S) = 4$, which implies $d(S \setminus Z) = 3$ by the submodular property (4.2). By the definition of F, we have $d_F(Z \setminus S) \geq 2$, $d_F(S \setminus Z) \geq 2$, and $d_F(S) = 2$. Thus we have $d_F(Z) \geq 2$ by the submodular property (4.2) for $d_F(.)$.

If $d(S \cap Z) = 4$, we have $d(S \cup Z) = 3$ by the fact that $|S \cup Z|$ is odd and the submodular property (4.1). By the definition of F, we have $d_F(S \cap Z) \geq 2$, $d_F(S \cup Z) = 2$, and $d_F(S) = 2$, and we have $d_F(Z) \geq 2$ by the submodular property (4.1) for $d_F(.)$.   ∎

5.2. 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 $G/\bar{Y}$. 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 5.2 below. Let K be the family of pairs of edges in D contained in 2-factors in $G/\bar{Y}$. Then $H = (D, K)$ forms a graph with vertex set D and edge set K.

LEMMA 5.2. *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 $G/\bar{Y}$. 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 new edge e° does not form a 1-edge cut since no proper subset of D is a cut, and thus 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 $G/\bar{Y}$. 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 $e_1 e_2$, $e_2 e_3$, $e_3 e_4$, and $e_4 e_1$ form a square in H. Then construct a graph $G^* = (W, E^*)$ as follows. Construct G/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 is 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 $G/\bar{Y}$ 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 5.3. *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 $d(Y \cap Z) = 2$, 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 $d(Z \setminus Y)$. Thus by the submodular property (4.2), $d(Z \setminus Y)$ is 2, 4, or 6. Since no proper subset of $\delta(Z)$ forms a cut, we have $d(Z \setminus Y) + d(Y \cap Z) > d(Z)$. Thus $d(Z \setminus Y) \neq 2$. If $d(Z \setminus Y) = 6$, then we have $d(Y \setminus Z) = 2$ by the submodular property (4.2), which is impossible since no proper subset of $\delta(Y)$ is a cut. Thus $d(Z \setminus Y) = 4$ holds, and there exists exactly one edge between $Y \cap Z$ and $Z \setminus Y$. Since $d_F(Z \setminus Y) \geq 2$ 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 $d(Y \setminus Z) = 2$.

Suppose $d(Y \cap Z) = 3$. 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 $d(Y \setminus Z)$ and $d(Z \setminus Y)$. Therefore we have $d(Y \setminus Z) \geq 5$ and $d(Z \setminus Y) \geq 5$, which provide a contradiction to (4.2). Thus we obtain $d(Y \cap Z) \neq 3$. Similarly, we may assert that $d(Y \setminus Z) \neq 3$.

Finally, consider the remaining case of $d(Y \cap Z) = d(Y \setminus Z) = 4$. 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 $\gamma(Y) = \{u_1 u_2, u_2 u_3, u_3 u_4, u_4 u_1\}$. 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 belong 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 $\delta(Z)$. □

5.3. 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 cuts 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/S and $G/\bar{S}$, and let G/S contain e^*. Apply

the algorithm recursively to obtain a 2-factor F' covering all the 3- and 4-edge cuts in G/S and excluding e^*. Let $f, f' \in F'$ be the edges incident to v_S. Then apply the algorithm recursively to $G/\bar{S}$ to obtain a 2-factor F° in $G/\bar{S}$ covering all the 3- and 4-edge cuts in $G/\bar{S}$ 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 $G/\bar{Y}$ 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^* as in section 5.2, 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 $G^\bullet$ as in section 5.2, 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/S and $G/\bar{S}$ 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 $n_1 + n_2 = n + 2$. Note that $f(n) = \mathrm{O}(n^2)$, as shown in section 4. By repeated decomposition in Step 1, we obtain that

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

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

$$T'(l) = T(l_1 + 2) + g(l_2 + 2) + h(l),$$

where $l_1 + l_2 = l$, $g(l)$ is the time complexity for finding a 2-factor excluding a specified edge in a bridgeless cubic graph with l vertices, and $h(l)$ 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 $h(l) = \mathrm{O}(l^2)$ 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 $\mathrm{O}(l)$ time [21, 25]. 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 $h(l) = \mathrm{O}(l^2)$, $T'(l) = \mathrm{O}(l^3)$, and $T(n) = \mathrm{O}(n^3)$.

THEOREM 5.4. *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 5.5. *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 [20] for Theorem 2.3 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.

6. Approximating a minimum 2-edge-connected subgraph. In this section, we describe Algorithm APX2EC, a 6/5-approximation algorithm for the 2-edge-connected subgraph problem (2EC) for 3-edge-connected cubic graphs. This algorithm uses Algorithm 34CUT as a preprocess, finds a subgraph with at most $6n/5$ edges containing two specified adjacent edges $e_1, e_2 \in E$, and runs in $O(n^3)$ time. 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 since multiple edges with multiplicity two would provide a 2-edge cut, and those with multiplicity three imply that G has only those three edges.

6.1. 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 6.1. *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^* appear 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^* appear 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^* appear 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 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 $G[C]$ unless C contains all vertices in G, and thus Lemma 6.1 can be applied to such cycles.

In the algorithm, we maintain a subgraph H satisfying the following properties:
 1. H is 2-edge-connected;
 2. $V(H)$ is the union of the vertex set of a subfamily of the cycles in $\mathcal{C}$; and
 3. $|E(H)| \leq 6|V(H)|/5$.
The subgraph H is initially an arbitrary cycle in $\mathcal{C}$ and is 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 6.1 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 6.1 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 T added to H' after P was added to H'. See Figure 6.2 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 6.2.

We call a cycle in $\mathcal{C}$ which does not belong to any lollipop or tadpole 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 we continue to grow H' from a matching edge in $\delta(\hat{L}) \setminus E(H')$.
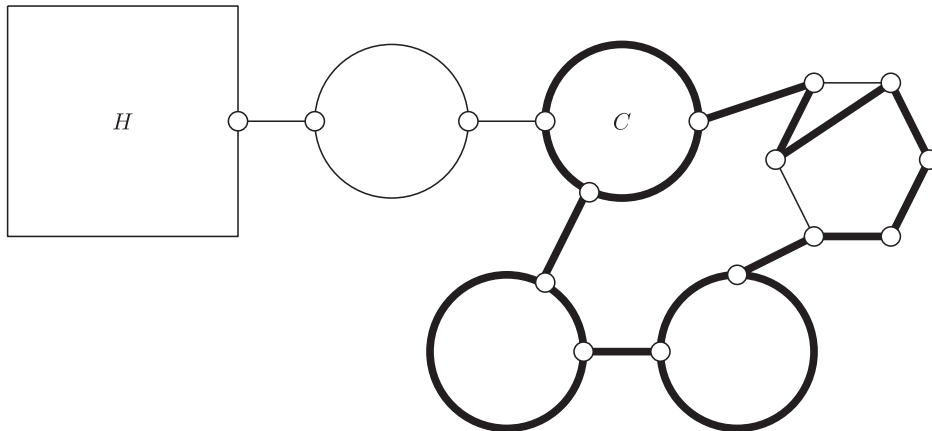
Fig. 6.1. *Construction of a lollipop L. The thick edges are the edges in L. The hexagon is a small cycle of size six, and the four circles indicate large cycles. (Some vertices and edges are omitted.)*
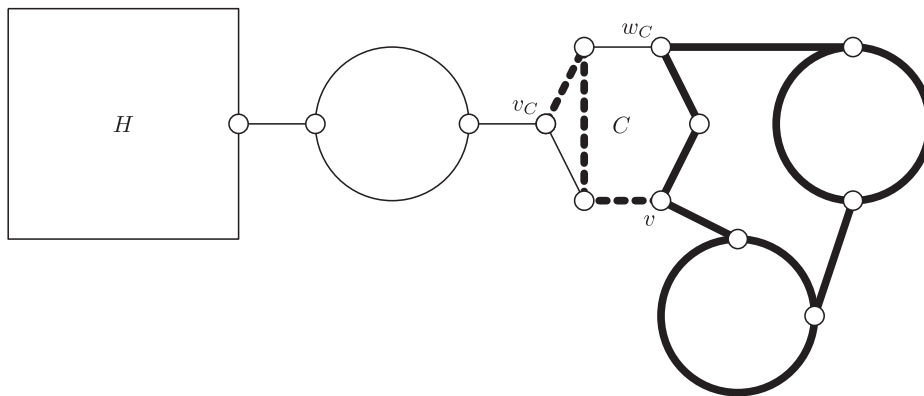

Fig. 6.2. *Construction of a tadpole T. The thick edges are the edges in T. The hexagon is a small cycle of size six, and the three circles indicate large cycles. (Some vertices and edges are omitted.) The dotted thick edges form the tail of T, and the solid thick edges form the head of T.*

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 we 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 handle only inclusionwise 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 spanning subgraph H of G with at most $6|V|/5$ edges.

Step 1. Apply Algorithm 34Cut to 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 is contained in a lollipop in H', then go to Step 4-3.
- If C is an independent small cycle in H' or is 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 6.1), 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 6.2. *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 $O(n^3)$ time (see Theorem 5.4). The other steps of the algorithm collectively take $O(n)$ time, and thus the total time complexity is $O(n^3)$. The approximation ratio follows from the following lemma.

LEMMA 6.3. *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 $n \geq 5|\mathcal{S}| + 10|\mathcal{L}|$. $\square$

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

THEOREM 6.4. *Algorithm* APX2EC *is a $6/5$-approximation algorithm for* 2EC *in 3-edge-connected cubic graphs. More precisely, for a 3-edge-connected cubic graph $G = (V, E)$, Algorithm* APX2EC *finds a 2-edge-connected subgraph H of G with $|E(H)| \leq 6n/5 - 1$ in $O(n^3)$ time.*

We finally remark that it is possible to make the output subgraph H in Algorithm APX2EC contain two specified adjacent edges.

COROLLARY 6.5. *For a 3-edge-connected cubic graph $G = (V, E)$ and two adjacent edges $e_1, e_2 \in E$, one can find a 2-edge-connected subgraph H of G with $|E(H)| \leq 6n/5 - 1$ and $\{e_1, e_2\} \subseteq E(H)$ in $O(n^3)$ time.*

Proof. Let $e_1, e_2 \in E$ be adjacent at $v \in V$, and let e_3 be the last edge in $\delta(v)$. In Step 1 of Algorithm APX2EC, we can apply Algorithm 34CUT so that the 2-factor F does not contain e_3. If e_1 and e_2 belong to a large cycle in F, then it is clear that they belong to the output subgraph. If they belong to a small cycle in F, then it suffices to choose that small cycle to be the initial cycle in H. $\square$

6.2. 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 [6]).

The *integrality gap* $\alpha(2\mathrm{EC})$ 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. Since

n is obviously a lower bound for the optimal value of the linear programming relaxation for graph2EC, we obtain the following theorem, which is an immediate consequence of Algorithm APX2EC.

THEOREM 6.6. *The value of $\alpha(2EC)$ 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 $\alpha(W2EC)$ for general W2EC is $6/5$, and, moreover, $6/5$ is the best lower bound known for $\alpha(W2EC)$ (see [2]). Previously, the best upper bound known for $\alpha(2EC)$ was $5/4$, which is implied by Huh's algorithm in [17]. We remark that the best lower bound we know for $\alpha(2EC)$ for bridgeless cubic graphs is $7/6$ and for 3-edge connected cubic graphs is $11/10$.

7. 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 spanning subgraph of G with at most $6n/5$ edges in $O(n^2 \log^4 n)$ 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 section 7.2 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 $|C| \leq 9$.

7.1. Differences from Algorithm APX2EC**.** We basically execute the same procedures as in Algorithm APX2EC. For the current initial 2-factor F, however, Lemma 6.2 fails due to the fact that F does not necessarily cover all 4-edge cuts. In this subsection, we exhibit tricks which ensure 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 the properties of small cut cycles.

LEMMA 7.1. *A small cut cycle $C \in \mathcal{C}$ has the following properties*:
1. *$G - C$ consists of exactly two components*;
2. *$|C| = 8$ or $|C| = 9$; and*
3. *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 $G - C$ is 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. $\square$

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 7.1.
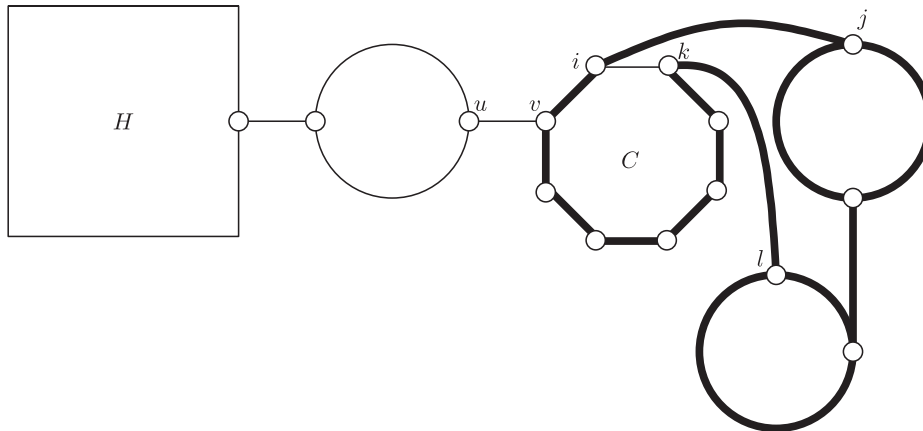
FIG. 7.1. *Construction of a lollipop L by Trick* 2. *The thick edges are the edges in L. The octagon is the small cut cycle C of size eight, and the three circles indicate large cycles. (Some vertices and edges are omitted.)*

LEMMA 7.2. *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 7.1 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 7.1 for an illustration.

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

LEMMA 7.3. *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 7.2 for an illustration.)
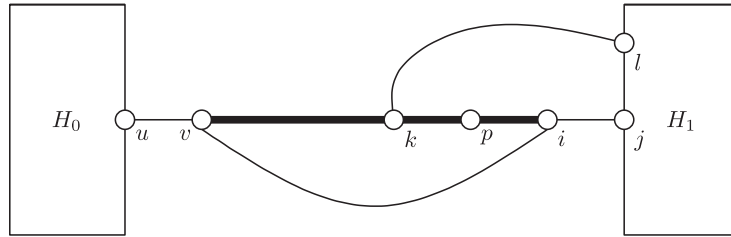
FIG. 7.2. *The path P, indicated by the thick line, and edge vi form the small cut cycle C_T. The subgraph of P connecting v and k is the tail of T.*
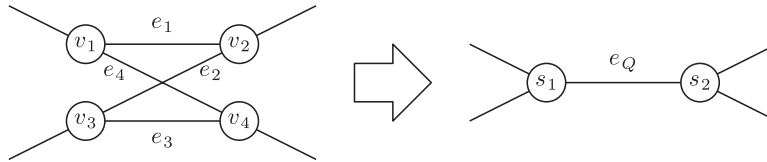

FIG. 7.3. *Shrinking a cycle of size four.*

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. Note that such a vertex p belongs to the head of T, and hence the matching edge incident to p is a forward edge. This implies that P has at least two vertices between i and a backward edge closest to i. By Trick 1, we have 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 7.2.   ☐

7.2. 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. If G is K_4, then any cycle of length four suffices. Thus, in what follows we assume that G is not K_4.

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 and 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 *pseudovertices* and e_Q as a *pseudoedge*. See Figure 7.3 for an 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. We also remark that a loop does not appear in this procedure: if G' has a loop attached to s_1, then $v_1v_3 \in E$, and thus either $V(Q)$ provides a 2-edge cut or $V(G) = V(Q)$ holds. The former case contradicts that G is 3-edge-connected and the latter that G is not a K_4.

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 $O(n \log^4 n)$ time [4]. Suppose G' has a proper 3-edge cut $D = \delta(S)$. We construct two graphs, G'/S and $G'/\bar{S}$. Without loss of generality,

suppose that G'/S contains e^*. Then we apply the present algorithm recursively to G'/S to obtain a 2-factor F° covering all the 3-edge cuts and excluding e^* in G'/S. Identify the unique edge e in $D \setminus F^\circ$. Then we again apply the present algorithm to $G'/\bar{S}$ recursively to obtain a 2-factor $F^\bullet$ in $G'/\bar{S}$ covering all 3-edge cuts in $G'/\bar{S}$ and excluding e. Now, by Lemma 4.1, $F' = F^\circ \cup F^\bullet$ 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 pseudoedges sequentially to recover G. In unshrinking a pseudoedge 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 belong to the same 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 belong to distinct cycles C_1 and C_2 in F', then choose any pair of disjoint edges in $E(Q)$ to obtain a cycle C_Q of size $|C_1| + |C_2| + 2$.

See Figure 7.4 for an illustration. In the end, we obtain a desired 2-factor F in G.

LEMMA 7.4. *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. By the definition of G', F' does not contain a cycle of size less than five without pseudovertices. Let e_Q be a pseudoedge 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 Case 1, $|C_Q| < 5$ implies that $|C| = 2$. Then we have that $V(Q)$ provides a 2-edge cut in G, which contradicts that G is 3-edge-connected. In Case 2.1, $|C_Q| < 5$ again implies that $|C| = 2$. Let $Q = (v_1, e_1, v_2, e_2, v_3, e_3, v_4, e_4, v_1)$. Then we have that $V(G) = \{v_1, v_2, v_3, v_4\}$ and $E(G) = \{e_1, e_2, e_3, e_4, e'_1, e'_3\}$, where e'_1 and e'_3 are parallel edges of e_1 and e_3, respectively. Now $\{v_1, v_2\}$ provides a 2-edge cut, which is a contradiction. In Case 2.2, $|C_Q| < 5$ implies that $|C_1| = |C_2| = 1$, which contradicts that G' has no loops.

We now prove that F'_Q covers all the 3-edge cuts 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 pseudoedge 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'.
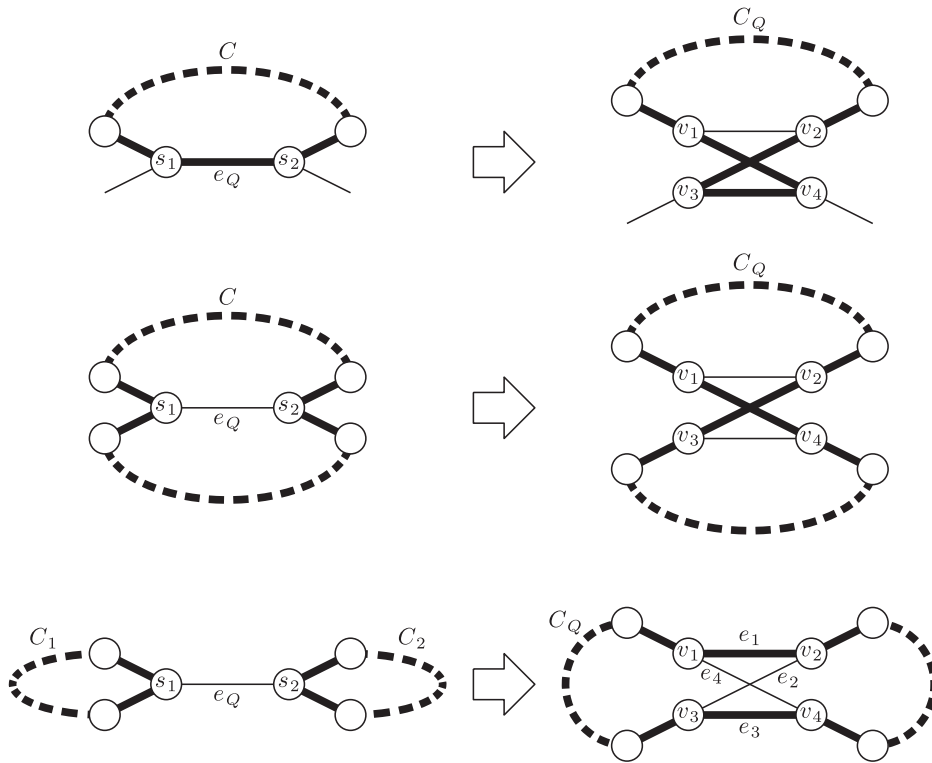
FIG. 7.4. *Unshrinking a pseudoedge. Thick edges are in F' and F. The three patterns indicate Cases 1, 2.1, and 2.2, respectively. In Case 2.2, it is also possible to add e_2 and e_4 to F' instead of e_1 and e_3.*

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 pseudoedges e_Q sequentially 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 D in a cubic graph with n vertices. Note that finding $\mathcal{Q}$ can be done in $O(n)$ time. As stated above, for finding a proper 3-edge cut D, it suffices to find a 2-factor in G' and a 3-edge cut in G''. The former can

be done in $O(n \log^4 n)$ time [4] and the latter in $O(n \log n)$ [12] 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 $f(n) = O(n \log^4 n)$ and $T(n) = O(n^2 \log^4 n)$.

By combining Procedure Ini2F and the argument in section 7.1, we obtain Algorithm FastApx2EC.

THEOREM 7.5. *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*, http://arxiv.org/abs/1101.5586 (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 Integer Programming and Combinatorial Optimization, Lecture Notes in Comput. Sci. 6080, Springer, Berlin, 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, R. SITTERS, S. VAN DER STER, AND L. STOUGIE, *TSP on cubic and subcubic graphs*, in Integer Programming and Combinatorial Optimization, Lecture Notes in Comput. Sci. 6655, Springer, Heidelberg, 2011, pp. 65–77.

[6] 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.

[7] G. CORNUÉJOLS, D. NADDEF, AND W. PULLEYBLANK, *The traveling salesman problem in graphs with* 3-*edge cutsets*, J. ACM, 32 (1985), pp. 383–410.

[8] 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.

[9] J. EDMONDS, *Maximum matching and a polyhedron with* 0,1-*vertices*, J. Res. Nat. Bur. Standards, Sect. B, 69 (1965), pp. 125–130.

[10] D. FOTAKIS AND P. SPIRAKIS, *Graph properties that facilitate travelling*, Electron. Colloq. Comput. Complex., 31 (1998), pp. 1–18.

[11] 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.

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

[13] 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.

[14] D. HARTVIGSEN, *Extensions of Matching Theory*, Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, 1984.

[15] D. HARTVIGSEN AND Y. LI, *Maximum cardinality simple* 2-*matchings in subcubic graphs*, SIAM J. Optim., 21 (2011), pp. 1027–1045.

[16] D. HARTVIGSEN AND Y. LI, *Polyhedron of triangle-free simple* 2-*matchings in subcubic graphs*, Math., Program., 138 (2013), pp. 43–82.

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

[18] 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.

[19] T. KAISER, D. KRÁL', AND S. NORINE, *Unions of perfect matchings in cubic graphs*, Electron. Notes Discrete Math., 22 (2005), pp. 341–345.

[20] T. KAISER AND R. ŠKREKOVSKI, *Cycles intersecting edge-cuts of prescribed sizes*, SIAM J. Discrete Math., 22 (2008), pp. 861–874.

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

[26] C. PAPADIMITRIOU AND M. YANNAKAKIS, *The traveling salesman problem with distances one and two*, Math. Oper. Res., 18 (1993), pp. 1–11.

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

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

[29] A. SEBŐ AND J. VYGEN, *Shorter Tours by Nicer Ears:* 7/5-*Approximation for Graphic TSP,* 3/2 *for the Path Version, and* 4/3 *for Two-Edge-Connected Subgraphs*, http://arxiv.org/abs/1201.1870 (2012).

[30] S. VEMPALA AND A. VETTA, *Factor* 4/3 *approximations for minimum* 2-*connected subgraphs*, in Approximation Algorithms for Combinatorial Optimization, Lecture Notes in Comput. Sci. 1913, Springer, Berlin, 2000, pp. 262–273.

[31] O. VORNBERGER, *Easy and Hard Cycle Covers*, preprint, Universität Paderborn, Paderborn, Germany, 1980.