An Efficient Algorithm for Cyclic Edge Connectivity of Regular Graphs

Dingjun Lou and Wei Wang
Department of Computer Science
Zhongshan University
Guangzhou 510275
People's Republic of China

Abstract

In this paper, we develop a polynomial time algorithm to determine the cyclic edge connectivity of a k-regular graph for $k \geq 3$. The time complexity of the algorithm is bounded by $O(k^{11}|V|^8)$, in particular, it is $O(|V|^8)$ for cubic graphs.

1 Introduction and terminology

Let G be an undirected, finite and connected graph. A cyclic edge cutset is an edge cutset whose deletion disconnects the graph such that two of the components contain a cycle respectively. The cyclic edge connectivity $c\lambda(G)$ is the minimum cardinality of all the cyclic edge cutsets of G. If no cyclic edge cutset exists, we say $c\lambda(G) = \infty$.

An edge cutset S can be written as $(T, V(G)\backslash T)$ that means that S contains all edges from T to $V(G)\backslash T$. Let C be a cycle of G. An edge e joining two vertices on C such that $e \notin E(C)$ is called a chord of C. Let C be a cycle without a chord. Then $S = (V(C), V(G)\backslash V(C))$ is called the *co-cycle* of C.

For the terminology and notation not defined in this paper, the reader is referred to [1].

The concept of cyclic edge connectivity was introduced by Tait [11] in the proof of the Four Colour Theorem. Plummer [10] studied the cyclic edge connectivity of planar graphs. In [3] and [5], Holton, Lou and Plummer showed the relation between cyclic edge connectivity and n-extendable graphs. In a paper of Peroche [9], several sorts of connectivity, including cyclic edge connectivity, and their relation are studied. For a summary of research in connectivity and edge connectivity, the reader is referred to [8].

It is a long standing unsolved problem whether the cyclic edge connectivity of a graph is a P-problem; even the case for regular graphs is unknown until now. In [6], Lou, Teng and Wu gave the first polynomial time algorithm to determine the cyclic edge connectivity of cubic graphs using the concept of removable edge (see [2]). However, there is an error in [6]. In this paper, we give an alternative polynomial time algorithm for cubic graphs to correct the first algorithm. Furthermore, we provide a polynomial time algorithm to determine the cyclic edge connectivity of k-regular graphs for $k \geq 3$. Although the time complexity of our algorithm is too large for practical use, this algorithm is the first polynomial time algorithm for the cyclic edge connectivity of regular graphs.

2 Prelininary results

In this section, we give two results on cyclic edge connectivity. The first lemma gives an upper bound of $c\lambda(G)$.

Lemma 1. ([9]): If $G = (V, E)$ is a simple graph with $|V| = \nu$ and $c\lambda(G) \neq \infty$, then $c\lambda(G) \leq 3(\nu - 3)$ for $\nu \geq 6$ and the bound is sharp. Equality holds when $G = K_\nu$.

Now we give a necessary and sufficient condition for $c\lambda(G) \neq \infty$ for all k-regular graphs.

Theorem 2: Let G be a connected k-regular graph. Then $c\lambda(G) \neq \infty$ if and only if $\nu(G) \geq 2g$, where g is the girth of G.

Proof. Suppose $\nu(G) \geq 2g$. We shall prove that the co-cycle of a minimum cycle C of G is a cyclic edge cutset, then $c\lambda(G) \leq (k-2)g$, hence $c\lambda(G) \neq \infty$.

Now we prove that $G - V(C)$ has a cycle. Suppose not. Then $G - V(C)$ is a forest. Then $k(\nu - g) - 2(\nu - g - 1) \leq (k-2)g$. We have $\nu \leq 2g - 2/(k-2)$. But ν is an integer. So $\nu \leq 2g - 1$, which contradicts the assumption $\nu(G) \geq 2g$.

Suppose $\nu(G) < 2g$ and $c\lambda \neq \infty$. Then G has a cyclic edge cutset S such that $G - S$ has two components C_1 and C_2 that both C_1 and C_2 have a cycle. So $|V(C_1)| \geq g$ and $|V(C_2)| \geq g$, which contradicts the assumption that $\nu(G) < 2g$. □

3 Algorithm for the cyclic edge connectivity of cubic graphs

In this section, we give an efficient algorithm to determine the cyclic edge connectivity of cubic graphs first. Then we give the proof of correctness of the algorithm and analyse the time complexity of the algorithm.

Algorithm 1:

```
1. Use a breadth first search strategy to find a shortest cycle containing $v$ for each vertex $v$ in $G$, then we can find the girth $g$ of $G$; // $O(|V|^2)$
2. If $\nu(G) < 2g$, then $c\lambda(G) = \infty$ and is returned; // $O(1)$
3. Use a breadth first search strategy to find all minimal cycles $C$ containing edge $e$ for each edge $e \in E(G)$ such that $|V(C)| \leq 4(\log_2 \nu + 1)$. Let $C_e$ be the set of all such cycles containing $e$ and let $F := \cup_{e \in E(G)} C_e$; // $O(|V|^3)$
4. $s := g$; // $O(1)$
5. For any two different cycles $C_1$ and $C_2$ in $F$ do // $O(|V|^6)$
BEGIN
    6. If $V(C_1) \cap V(C_2) = \emptyset$, then we construct a new graph $G'$ such that $V(G') = V(G) \cup \{x, y\}$, where $x, y \notin V(G)$ and $E(G') = E(G) \cup \{xu \mid u \in V(C_1)\} \cup \{yv \mid v \in V(C_2)\}$; // $O(|V|)$
    7. Use the algorithm of [7] to find a minimum edge cutset $S_{xy}$ which separates $x$ and $y$; // $O(|V|^2)$
    8. $s := \min\{s, |S_{xy}|\}$; // $O(1)$
END;
9. Then $c\lambda(G) = s$ and is returned. // $O(1)$
```

In Step 3, the minimal cycle C containing e is a cycle without chord. Now we prove the correctness of Algorithm 1.

Theorem 3: Algorithm 1 can find a minimum cyclic edge cutset and hence can determine $c\lambda(G)$.

Proof. By Theorem 2, if $g > \nu/2$, then Algorithm 1 will return $c\lambda(G) = \infty$ in Step 2.

Let C be a shortest cycle in G such that $|V(C)| = g \leq \nu/2$. By Theorem 2, the co-cycle S of C is a cyclic edge cutset. So $c\lambda(G) \leq |S| = g$. If

$c\lambda(G) = g$, by Steps 4, 8 and 9, Algorithm 1 will determine that $c\lambda(G) = g$.

Now suppose that $c\lambda(G) \leq g-1$. Assume $S = (V(D_1), V(D_2))$ is a minimum cyclic edge cutset, where $D_2 = G - V(D_1)$. Then D_1 and D_2 have shortest cycles C_1 and C_2 respectively. Let $C_1 = a_0a_1\ldots a_{c-1}a_0$ be a shortest cycle in D_1. Notice that C_1 does not have any chord. Let $N_0(a_i) = \{a_i\}$, $N_1(a_i)\backslash C_1 = \{u \mid u \in V(G),\ ua_i \in E(G) \text{ and } u \notin V(C_1)\}$, $N_r(a_i)\backslash C_1 = \{u \mid u \in V(G)\backslash V(C_1),\ d(a_i,u) = r, \text{ and } \exists x \in N_{r-1}(a_i)\backslash C_1,\ xu \in E(G)\}$ $(r \geq 2,\ i = 0,1,\ldots,c-1)$. Let $M_r(a_i) = N_0(a_i) \cup_{j=1}^{r} N_j(a_i)\backslash C_1$.

Suppose $(N_1(a_i)\backslash C_1) \cap (N_1(a_j)\backslash C_1) \neq \emptyset$ (for some $i \neq j$). Then either we have a cycle C' in D_1 of length less than $|V(C_1)|$, contradicting the assumption that C_1 is a shortest cycle in D_1, or $|V(C_1)| \leq 4 \leq 4(\log_2 \nu + 1)$.

Now suppose that $|V(C_1)| > 4$. Let $|V(C_1)| = c$. Let $G_i = G[M_{c/4-1}(a_i)]$. Since C_1 is a shortest cycle in D_1, $(G_i \cap D_1) \cap (G_j \cap D_1) = \emptyset$ $(i \neq j)$. Since $|S| \leq g-1$ and $|V(C)| \geq g$, the edges of S lie at most $g-1$ of G_i $(i = 0,1,\ldots,c-1)$. So there is an a_j such that G_j does not contain any edges in S. This means that $G_j \subseteq D_1$. But G_j is a tree as C_1 is a shortest cycle in D_1. $\nu(G_j) = 2^0 + 2^0 + 2^1 + 2^2 + \ldots + 2^{c/4-2} = 2^{c/4-1}$. So $2^{c/4-1} = \nu(G_j) \leq \nu(D_1) \leq \nu(G)$. Hence $c \leq 4(\log_2 \nu + 1)$.

By the same reason, the shortest cycle in D_2 has length at most $4(\log_2 \nu + 1)$.

In Step 3 of Algorithm 1, we find all minimal cycles of length at most $4(\log_2 \nu + 1)$ in G. In Steps 5, 6, and 7, we find a minimum edge cutset which separates C_1 in D_1 and C_2 in D_2. Hence we can find a cyclic edge cutset of size $|S|$. By Step 8, we can determine the cyclic edge connectivity of G. □

Now we analyse the time complexity of Algorithm 1.

Theorem 4: The time complexity of Algorithm 1 is bounded by $O(|V|^8)$.

Proof. First we notice that, since G is a cubic graph, $|E| = 3|V|/2$, so $O(|E|)$ is equal to $O(|V|)$.

In Step 1, it takes $O(|E|)$ time to find a shortest cycle containing a given vertex v using breadth first search strategy. Step 1 takes totally $O(|V||E|) = O(|V|^2)$ time for all vertices v in $V(G)$.

Now we analyse Step 3. Let $e = xy$ be an edge in $E(G)$.

We define $N_0(x) = \{x\}$, $N_0(y) = \{y\}$; $N_1(x) = \{u \mid \exists v \in N_0(x),\ uv \in E(G),\ u \notin N_0(y), u \neq x\}$, $N_1(y) = \{u \mid \exists v \in N_0(y),\ uv \in E(G),$

$u \notin N_0(x) \cup N_1(x), u \neq y\}$; $N_r(x) = \{u \mid \exists v \in N_{r-1}(x),\ uv \in E(G),\ u \notin \cup_{i=0}^{r-1} N_i(y), u \neq x\}$, $N_r(y) = \{u \mid \exists v \in N_{r-1}(y),\ uv \in E(G),\ u \notin \cup_{i=0}^{r} N_i(x), u \neq y\}$ $(r \geq 2)$. Then $|N_0(x)| = 2^0$, $|N_0(y)| = 2^0$; $|N_1(x)| = 2^1$, $|N_r(x)| \leq 2^r$, $|N_r(y)| \leq 2^r$ $(r \geq 2)$.

Since, in Step 3, we find all minimal cycles C containing e for the given edge e such that $|V(C)| \leq 4(\log_2 \nu + 1) = c$, notice that C does not contain any chord and every edge from $\cup_{i=1}^{c/2-1} N_i(x)$ to $\cup_{i=0}^{c/2-1} N_i(y)$ corresponds to a different cycle containing e, also notice that, if there is an edge from $N_r(x)$ to $\cup_{i=0}^{c/2-1} N_i(y)$, then $|N_{r+1}(x)|$ is less than 2^{r+1} by one, since G is a cubic graph, the number of such cycles is at most $2 \times 2^{c/2-1} = 2^{c/2} = 2^{2(\log_2 \nu+1)} = 4\nu^2 = O(|V|^2)$. For each $e \in E(G)$, there are at most $O(|V|^2)$ minimal cycles of length at most $4(\log_2 \nu + 1)$ containing e. There are at most $O(|V|^2|E|) = O(|V|^3)$ such cycles in F for all edges in $E(G)$.

In Step 5, the FOR loop is for each combination of two different cycles in F. So the loop repeats $O(|V|^6)$ times. Step 6 takes $O(|V|)$ time to test $V(C_1) \cap V(C_2) = \emptyset$ and to construct G'.

By [7], Step 7 takes $O(|V||E|) = O(|V|^2)$ time to find a minimum edge cutset S_{xy}. So the loop of Steps 5, 6, 7, and 8 takes totally $O(|V|^8)$ time.

Hence the whole algorithm takes $O(|V|^8)$ time. □

4 Algorithm for cyclic edge connectivity of k-regular graphs

In the following, we give an efficient algorithm to determine the cyclic edge connectivity of k-regular graphs. The idea is the same as Algorithm 1. We write these two algorithms separately for the ease of understanding.

Algorithm 2:

```
1. Use a breadth first search strategy to find a shortest cycle containing $v$ for each vertex $v$ in $G$, then we can find the girth $g$ of $G$; // $O(k|V|^2)$
2. If $\nu(G) < 2g$, then $c\lambda(G) = \infty$ and is returned; // $O(1)$
3. Use a breadth first search strategy to find all minimal cycles $C$ containing edge $e$ for each edge $e \in E(G)$ such that $|V(C)| \leq 4(\log_{k-1} \nu + 2)$. Let $C_e$ be the set of all such cycles containing $e$ and let $F = \cup_{e \in E(G)} C_e$; // $O(k^5|V|^3)$
4. $s := (k-2)g$; // $O(1)$
5. For any two different cycles $C_1$ and $C_2$ in $F$ do // $O(k^{10}|V|^6)$
BEGIN
```

6. If $V(C_1) \cap V(C_2) = \emptyset$, then we construct a new graph G' such that $V(G') = V(G) \cup \{x, y\}$, where $x, y \notin V(G)$, and $E(G')$ contains all edges in $E(G)$, for each vertex u on C_1, we put $(k-2)$ multiple edges between x and u, and for each vertex v on C_2, we put $(k-2)$ multiple edges between y and v; // $O(|V|)$
7. Use the algorithm of [7] to find a minimum edge cutset S_{xy} which separates x and y; // $O(k|V|^2)$
8. $s := \min\{s, |S_{xy}|\}$; // $O(1)$
END;
9. Then $c\lambda(G) = s$ and is returned; // $O(1)$

Now we prove the correctness of Algorithm 2.

Theorem 5: Algorithm 2 can find a minimum cyclic edge cutset and hence can determine $c\lambda(G)$.

Proof. By Theorem 2, if $g > \nu/2$, then Algorithm 2 will give answer that $c\lambda = \infty$ in Step 2.

Now suppose $g \le \nu/2$. Let C be a shortest cycle in G such that $|V(C)| = g \le \nu/2$. By Theorem 2, the co-cycle S of C is a cyclic edge cutset. So $c\lambda(G) \le |S| = (k-2)g$. If $c\lambda(G) = (k-2)g$, by Steps 4, 8, and 9, Algorithm 2 will determine that $c\lambda(G) = (k-2)g$.

Now suppose that $c\lambda(G) \le (k-2)g - 1$. Assume $S = (V(D_1), V(D_2))$ is a minimum cyclic edge cutset, where $D_2 = G - V(D_1)$. Then D_1 and D_2 have shortest cycles C_1 and C_2 respectively. Let $C_1 = a_0 a_1 \ldots a_{c-1} a_0$ be a shortest cycle in D_1. Notice that C_1 does not have any chord. Let $N_0(a_i) = \{a_i\}$ and let $a_{i1}, a_{i2}, \ldots, a_{i,k-2}$ be the vertices in $N(a_i)\backslash V(C_1)$. Let $N_1(a_{ij})\backslash C_1 = \{a_{ij}\}$, $N_r(a_{ij})\backslash C_1 = \{u \mid u \in V(G)\backslash V(C_1), d(a_{ij}, u) = r-1, \text{ and } \exists x \in N_{r-1}(a_{ij})\backslash C_1, xu \in E(G)\}$ $(r \ge 2)$. Let $M_r(a_{ij}) = N_0(a_i) \cup_{k=1}^{r} N_k(a_{ij})\backslash C_1$.

Suppose $(N_1(a_{ij})\backslash C_1) \cap (N_1(a_{pq})\backslash C_1) \neq \emptyset$ for some $i \neq p$ or $j \neq q$. Then either we have a cycle C' in D_1 of length less than $|V(C_1)|$, contradicting the assumption that C_1 is a shortest cycle in D_1, or $|V(C_1)| \le 4 \le 4(\log_{k-1} \nu + 2)$.

Now suppose that $|V(C_1)| > 4$. Let $|V(C_1)| = c$ and $G_{ij} = G[M_{c/4-1}(a_{ij})]$. Since C_1 is a shortest cycle in D_1, $(G_{ij} \cap D_1) \cap (G_{pq} \cap D_1) = \emptyset$ $(i \neq p$ or $j \neq q)$. Since $|S| \le (k-2)g - 1$ and $|V(C)| \ge g$, the edges of S lie in at most $(k-2)g - 1$ of G_{ij} $(i = 1, 2, \ldots, c-1; j = 1, 2, \ldots, k-2)$. So there is an a_{ij} such that $G_{ij} = G[M_{c/4-1}(a_{ij})]$ does not contain any edge in S. This means that $G_{ij} \subseteq D_1$. But G_{ij} is a tree as C_1 is a shortest cycle in D_1. So $\nu(G_{ij}) = 1 + (k-1)^0 + (k-1)^1 + \ldots + (k-1)^{c/4-2} = 1 + [(k-1)^{c/4-1} - 1]/(k-2)$.

Then $\nu(G) \geq \nu(D_1) \geq \nu(G_{ij}) = 1 + [(k-1)^{c/4-1} - 1]/(k-2)$. So $c \leq 4[\log_{k-1}((k-2)(\nu-1)+1)+1] \leq 4[\log_{k-1}\nu + 2)$.

In Step 3 of Algorithm 2, we find all minimal cycles of length at most $4[\log_{k-1}\nu + 2)$ in G. In Steps 5, 6, and 7, we find a minimum edge cutset which separates C_1 in D_1 and C_2 in D_2. Hence we can find a cyclic edge cutset of size $|S|$. By Step 8, we can determine the cyclic edge connectivity of G. □

Now we analyse the time complexity of Algorithm 2.

Theorem 6: The time complexity of Algorithm 2 is bounded by $O(k^{11}|V|^8)$.

Proof. In Step 1, we use breadth first search strategy to find a shortest cycle containing v for a given vertex v takes $O(|E|)$ time. For all vertices v in $V(G)$, Step 1 takes $O(|V||E|)$ time. Since G is a k-regular graph, $|E| = k|V|/2$. Hence $O(|V||E|) = O(k|V|^2)$.

Step 2 takes $O(1)$ time.

Now we analyse Step 3. Let $e = xy$ be an edge in $E(G)$. We define $N_0(x) = \{x\}$, $N_0(y) = \{y\}$, $N_1(x) = \{u \mid \exists v \in N_0(x), uv \in E(G), u \notin N_0(y), u \neq x\}$, $N_1(y) = \{u \mid \exists v \in N_0(y), uv \in E(G), u \notin N_0(x) \cup N_1(x), u \neq y\}$, $N_r(x) = \{u \mid \exists v \in N_{r-1}(x), uv \in E(G), u \notin \cup_{i=0}^{r-1} N_i(y), u \neq x\}$, $N_r(y) = \{u \mid \exists v \in N_{r-1}(y), uv \in E(G), u \notin \cup_{i=0}^{r} N_i(x), u \neq y\}$ $(r \geq 2)$. Then $|N_0(x)| = (k-1)^0$, $|N_0(y)| = (k-1)^0$, $|N_1(x)| = (k-1)^1$, $|N_1(y)| \leq (k-1)^1$, $|N_r(x)| \leq (k-1)^r$, $|N_r(y)| \leq (k-1)^r$.

Since, in Step 3, we find all minimal cycles C containing e for the given edge e such that $|V(C)| \leq 4(\log_{k-1}\nu + 2) = c$, notice that C does not contain any chord and every edge from $\cup_{i=1}^{c/2-1} N_i(x)$ to $\cup_{i=0}^{c/2-1} N_i(y)$ corresponds to a different cycle containing e, also notice that, if there is an edge from $N_r(x)$ to $\cup_{i=0}^{c/2-1} N_i(y)$, then $|N_{r+1}(x)|$ is less than $(k-1)^{r+1}$ by one, since G is k-regular, the number of such cycles is at most $(k-1)(k-1)^{c/2-1} = (k-1)^{c/2} = (k-1)^{2(\log_{k-1}\nu+2)} = (k-1)^4\nu^2 = O(k^4|V|^2)$. For all edges in $E(G)$, there are at most $O(k^4|V|^2|E|) = O(k^5|V|^3)$ such cycles in F.

In Step 5, the FOR loop is for each combination of two different cycles in F. So the loop repeats $O(k^{10}|V|^6)$ times. Step 6 takes $O(|V|)$ time to test $V(C_1) \cap V(C_2) = \emptyset$ and construct G'.

By [7], Step 7 takes $O(|V||E|) = O(k|V|^2)$ time to find a minimum edge cutset S_{xy}. So the loop of Steps 5, 6, 7, and 8 takes totally $O(k^{11}|V|^8)$ time.

Hence the algorithm takes $O(k^{11}|V|^8)$ time. □

Acknowledgement. The work of this paper was supported by the National Science Foundation of China.

References

[1] J.A. Bondy and U.S.R. Murty, Graph theory with applications, MacMillan Press, London (1976).

[2] D.A. Holton, B. Jackson, A. Saito and N.C. Wormald, Removable edges in 3-connected graphs, *J. Graph Theory* **14** (1990), 465–473.

[3] D.A. Holton, Dingjun Lou and M.D. Plummer, On the 2-extendability of planar graphs, *Discrete Math.* **96** (1991), 81–99.

[4] J.E. Hopcroft and R.E. Tarjan, Dividing a graph into triconnected components, *SIAM J. Computing* **2:3** (1973), 135–158.

[5] Dingjun Lou and D.A. Holton, Lower bound of cyclic edge connectivity for n-extendability of regular graphs, *Discrete Math.* **112** (1993), 139–150.

[6] Dingjun Lou, Lihua Teng and Xiangjun Wu, A polynomial algorithm for cyclic edge connectivity of cubic graphs, *Australas. J. Combin.* **24** (2001), 247–259.

[7] H. Nagamochi and T. Ibaraki, Computing edge-connectivity in multigraphs and capacitated graphs, *SIAM J. Discrete Math.* **5** (1992), 54–66.

[8] O.R. Oellermann, Connectivity and edge-connectivity in graphs: A survey, *Congress. Numer.* **116** (1996), 231–252.

[9] B. Peroche, On several sorts of connectivity, *Discrete Math.* **46** (1983), 267–277.

[10] M.D. Plummer, On the cyclic connectivity of planar graphs, in: Y. Alavi, D.R. Lick and A.T. White eds., Graph Theory and Applications, Springer-Verlag, Berlin (1972), 235–242.

[11] R.G. Tait, Remarks on the colouring of maps, *Proc. Roy. Soc. Edinburg* **10** (1880), 501–503.