Computing Eternal Vertex Cover Number of Maximal Outerplanar Graphs in Linear Time

Jasine Babu[1,‡], K. Murali Krishnan[2,†], Veena Prabhakaran[1,*,†] and Nandini J. Warrier[2,†]

[1]Indian Institute of Technology Palakkad, India
[2]National Institute of Technology Calicut, India

Abstract

Eternal vertex cover problem is a variant of the classical vertex cover problem modeled as a two player attacker-defender game. Computing eternal vertex cover number of graphs is known to be NP-hard in general and even for bipartite graphs. There is a quadratic complexity algorithm known for this problem for chordal graphs. Maximal outerplanar graphs forms a subclass of chordal graphs, for which no algorithm of sub-quadratic time complexity is known. In this paper, we obtain a linear time recursive algorithm for computing eternal vertex cover number of maximal outerplanar graphs.

Keywords

Eternal Vertex Cover, Maximal Outerplanar Graph, Linear Time Algorithm

1. Introduction

A set S of vertices in a graph G forms a vertex cover of G if every edge in G has at least one end point in S. The size of a minimum vertex cover in G, called the vertex cover number of G, will be denoted by $\mathrm{mvc}(G)$. The eternal vertex cover problem of graphs is motivated by the following dynamic network security / fault-tolerance model. A network can be modeled by a graph $G(V, E)$, where the nodes of the network are represented by the vertices of G and the links by the edges of G. The problem is to deploy a minimum set of guards at the nodes, so that if there is an attack (or fault) on a single link at any time, a guard is available at the end of the link, who can move across the link to defend (or repair) the attack (or fault). Simultaneously, the remaining guards need to reconfigure themselves, possibly by repositioning themselves to one of their adjacent nodes, so that any attack (or fault) on a single edge at any future instant of time can also be protected in the same manner. Thus, the model requires guaranteeing protection against single link attacks/failures ad-infinitum. It is immediate that lowest number of guards needed to achieve this goal in a graph G is at least $\mathrm{mvc}(G)$. If k guards are sufficient to achieve the goal, then we say that G is k-defendable.

Formally, guards are initially placed on a vertex cover C_0 of G, forming an initial configuration.

Proceedings of the 23rd Italian Conference on Theoretical Computer Science, Rome, Italy, September 7-9, 2022
*Corresponding author.
†These authors contributed equally.
‡Jasine Babu acknowledges support from SERB Power Grant SPG/2021/003250
jasine@iitpkd.ac.in (J. Babu); kmurali@nitc.ac.in (K. Murali Krishnan); veenaprabhakaran7@gmail.com (V. Prabhakaran); nandini.wj@gmail.com (N. J. Warrier)

Suppose at instant i, guards are placed in a configuration C_i and an attack occurs on an arbitrary edge $uv \in E(G)$, then a guard at either u or v (or both) must move across the edge. Other guards may or not move across one of their neighboring edges simultaneously. At the end of these movements, the next configuration C_{i+1} is reached. To ensure that all edges remain guarded at the instant $i+1$, we require C_{i+1} to be a vertex cover of G. The minimum number of guards necessary for a graph G to be protected against any infinite sequence of single edge attacks is called the eternal vertex cover number of G, denoted by $\mathrm{evc}(G)$. The eternal vertex cover problem has two models. The first one allows only at most one guard on a vertex in any configuration, while in the second model this constraint is absent. The results in this paper work in both the models.

Computing $\mathrm{evc}(G)$ for an arbitrary graph G is NP-hard, though in PSPACE [1], and is fixed parameter tractable with $\mathrm{evc}(G)$ as parameter [1]. Recently, it was shown that the problem is NP-hard even for bipartite graphs [2]. It is also known that the problem is NP-complete for biconnected internally triangulated planar graphs [3]. Polynomial time algorithms for computing $\mathrm{evc}(G)$ exactly were known only for very elementary graph classes such as an $O(n)$ algorithm for trees [4], a polynomial time algorithm for a tree-like graph class [5] and a linear time algorithm for cactus graphs [6]. A recent structure theorem developed in [7] has resulted in a quadratic time algorithm for chordal graphs.

An outerplanar graph is a planar graph that admits a planar embedding with all its vertices lying on the exterior face and it is maximal outerplanar if addition of any more edges between existing vertices will make the graph not outerplanar. Since maximal outerplanar graphs are chordal, it follows from [7] that its evc number can be computed in quadratic time. We improve the complexity and show that the evc number of maximal outerplanar graphs can be computed in linear time. The techniques used here are fundamentally different from that used in the algorithm known for cactus graphs [6], which is based on the fact that no edge of a cactus graph lies on more than one cycle.

Our algorithm takes a maximal outerplanar graph G on at least three vertices and an edge uv on the outer face of G and recursively computes $\mathrm{evc}(G)$, $\mathrm{mvc}(G)$ along with some related parameters. If both u and v are of degree greater than two, then the recursion is applied on the two induced subgraphs G_u and G_v of G as shown in Figure 1. Otherwise, the recursion works on the graph obtained by deleting the degree two end point of the edge uv from G. Since $\mathrm{evc}(G)$ happens to be not merely a function of the eternal vertex cover number and vertex cover number of these associated subgraphs, the algorithm has to recursively compute this larger set of parameters. The linear time complexity of the algorithm is derived from Theorems 1 to 4, which assert that $\mathrm{evc}(G)$ can be determined in constant time from this larger set of parameters of the associated subgraphs. Apart from the algorithmic result, these theorems serve to demonstrate the structural connection between the minimum vertex cover problem and the eternal vertex cover problem for maximal outerplanar graphs.

2. Preliminaries

Let $G(V, E)$ be a graph with $S \subseteq V$. The parameter $\mathrm{evc}_S(G)$ denotes the minimum number k such that G is k-defendable with all vertices of S occupied in all configurations. Similarly,

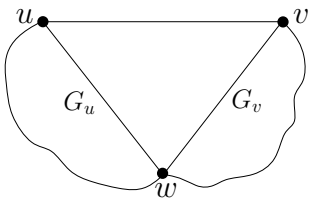

Figure 1: G_u is the uv segment of this graph that contains u and G_v is the uv segment that contains v.

$\mathrm{mvc}_S(G)$ denote the size of the smallest cardinality vertex cover of G containing all vertices of S. When $S = \{v_1 \dots v_i\}$ for $1 \leq i \leq 3$, we shorten the notation $\mathrm{evc}_{\{v_1 \dots v_i\}}(G)$ and $\mathrm{mvc}_{\{v_1 \dots v_i\}}(G)$ as $\mathrm{evc}_{v_1 \dots v_i}(G)$ and $\mathrm{mvc}_{v_1 \dots v_i}(G)$ respectively.

Proofs omitted from the main content due to space constraints are available in the arXiv preprint [8]. Proposition 1 given below is an easy adaptation of a result given in [3].

Proposition 1. *[3] Let $G(V, E)$ be a maximal outerplanar graph with at least two vertices and $S \subseteq V$. If for every vertex $v \in V \setminus S$, $\mathrm{mvc}_{S\cup\{v\}}(G) = \mathrm{mvc}_S(G)$, then $\mathrm{evc}_S(G) = \mathrm{mvc}_S(G)$. Otherwise, $\mathrm{evc}_S(G) = \mathrm{mvc}_S(G) + 1$. Hence, $\mathrm{evc}_S(G) = \max_{v\in V(G)} \mathrm{mvc}_{S\cup\{v\}}(G)$.*

The following proposition is a direct consequence of Proposition 1.

Proposition 2. *Let G be a maximal outerplanar graph with an edge uv. Then, $\mathrm{evc}_v(G) \leq \mathrm{evc}_{uv}(G) \leq \mathrm{evc}(G) + 1$.*

From now on, whenever we say a graph is maximal outerplanar, we assume a fixed outerplanar embedding of the graph. For a maximal outerplanar graph G on at least three vertices and any edge uv on the outer face of G, we use the notation $\Delta(uv)$ to denote the unique common neighbor of u and v.

Definition 1 (uv-segments)**.** *Let G be a maximal outerplanar graph with at least three vertices. Let uv be an edge on the outer face of G. Let $w = \Delta(uv)$. We define the graph $G_u(uv)$ (respectively, $G_v(uv)$) to be the maximal biconnected outerplanar subgraph of G that satisfies the following two properties (see Figure 1):*

1. *The edge uw (respectively, vw) is on the outer face of $G_u(uv)$ (respectively, $G_v(uv)$).*
2. *$G_u(uv)$ (respectively, $G_v(uv)$) does not contain the vertex v (respectively u).*

$G_u(uv)$ and $G_v(uv)$ will be called the uv segments of G.

Note 1. *We will write G_u and G_v instead of $G_u(uv)$ and $G_v(uv)$ when there is no scope for confusion. Observe that G_u and G_v will be single edge graphs if G is a triangle.*

Definition 2 (mvc and evc parameters)**.** *For a maximal outerplanar graph G and an edge uv, the set of parameters $\mathscr{M}(G, uv) = \{\mathrm{mvc}(G), \mathrm{mvc}_u(G), \mathrm{mvc}_v(G), \mathrm{mvc}_{uv}(G)\}$ is called the (set of) mvc parameters of G with respect to uv and the set $\mathscr{E}(G, uv) = \{\mathrm{evc}(G), \mathrm{evc}_u(G), \mathrm{evc}_v(G), \mathrm{evc}_{uv}(G)\}$ is called the (set of) evc parameters of G with respect to uv.*

The following proposition that gives the mvc and evc parameters of a triangle is the base case of the recursive computation of these parameters for larger graphs described in subsequent sections.

Proposition 3. *Suppose G is a triangle and uv an edge of it. Then,*

1. $\mathrm{mvc}(G) = \mathrm{mvc}_u(G) = \mathrm{mvc}_v(G) = \mathrm{mvc}_{uv}(G) = 2$
2. $\mathrm{evc}(G) = \mathrm{evc}_u(G) = \mathrm{evc}_v(G) = 2$ *and* $\mathrm{evc}_{uv}(G) = 3$

3. Computation of the mvc Parameters

Throughout this section, we assume that G is a maximal outerplanar graph of at least four vertices, uv is an edge on the outer face of G and $w = \Delta(uv)$. The results in this section show that the mvc parameters of G with respect to the edge uv - viz., $\mathscr{M}(G, uv)$, can be computed in constant time if the mvc parameters of G_u with respect to uw - viz., $\mathscr{M}(G_u, uw)$ and the mvc parameters of G_v with respect to vw - viz., $\mathscr{M}(G_v, vw)$ are given.

The following observation is easy to see.

Observation 1. *If $deg_G(u) = 2$ and $deg_G(v) > 2$, then,*

1. $\mathrm{mvc}(G) = \mathrm{mvc}_{vw}(G \setminus u)$
2. $\mathrm{mvc}_u(G) = \mathrm{mvc}(G \setminus u) + 1$
3. $\mathrm{mvc}_v(G) = \mathrm{mvc}(G)$
4. $\mathrm{mvc}_{uv}(G) = \mathrm{mvc}_v(G \setminus u) + 1$

The following theorem is a consequence of Observation 1.

Theorem 1. *Let G be a maximal outerplanar graph and uv be an edge on the outer face of G such that $deg_G(u) = 2$. Let $w = \Delta(uv)$.*

1. *Given $\mathscr{M}(G \setminus u, vw)$, it is possible to compute $\mathrm{mvc}(G)$ in constant time.*
2. *Given $\mathrm{mvc}(G)$ and $\mathscr{M}(G \setminus u, vw)$, it is possible to compute the remaining mvc parameters of G with respect to uv in constant time.*

Now, we look at the computation of $\mathscr{M}(G, uv)$ when $deg_G(u) > 2$ and $deg_G(v) > 2$. The following proposition is easy to obtain.

Proposition 4. *If $deg_G(u) > 2$ and $deg_G(v) > 2$, then*
$\mathrm{mvc}(G) \in \{\mathrm{mvc}(G_u) + \mathrm{mvc}(G_v) - 1, \mathrm{mvc}(G_u) + \mathrm{mvc}(G_v)\}$.

The following lemma gives a method to compute $\mathrm{mvc}(G)$ using $\mathscr{M}(G_u, uw)$ and $\mathscr{M}(G_v, vw)$.

Lemma 1. *If $deg_G(u) > 2$ and $deg_G(v) > 2$, then* $\mathrm{mvc}(G) = \min\{\mathrm{mvc}_w(G_u) + \mathrm{mvc}_{vw}(G_v) - 1, \mathrm{mvc}_{uw}(G_u) + \mathrm{mvc}_w(G_v) - 1, \mathrm{mvc}(G_u) + \mathrm{mvc}(G_v)\}$.

The next lemma shows that given $\mathrm{mvc}(G)$, $\mathscr{M}(G_u, uw)$ and $\mathscr{M}(G_v, vw)$, the values of $\mathrm{mvc}_u(G)$, $\mathrm{mvc}_v(G)$ and $\mathrm{mvc}_{uv}(G)$ are computable in constant time.

Lemma 2. *If $deg_G(u) > 2$ and $deg_G(v) > 2$, then:*

1. *If* $\mathrm{mvc}(G) = \mathrm{mvc}(G_u) + \mathrm{mvc}(G_v) - 1$, *then* $\mathrm{mvc}_u(G) = \mathrm{mvc}_{uw}(G_u) + \mathrm{mvc}(G_v) - 1$, $\mathrm{mvc}_v(G) = \mathrm{mvc}(G_u) + \mathrm{mvc}_{vw}(G_v) - 1$ *and* $\mathrm{mvc}_{uv}(G) = \mathrm{mvc}_{uw}(G_u) + \mathrm{mvc}_{vw}(G_v) - 1$.
2. *If* $\mathrm{mvc}(G) = \mathrm{mvc}(G_u) + \mathrm{mvc}(G_v)$, *then* $\mathrm{mvc}_u(G) = \min\{\mathrm{mvc}_u(G_u) + \mathrm{mvc}(G_v), \mathrm{mvc}(G_u) + \mathrm{mvc}_w(G_v), \mathrm{mvc}(G) + 1\}$, $\mathrm{mvc}_v(G) = \min\{\mathrm{mvc}(G_u) + \mathrm{mvc}_v(G_v), \mathrm{mvc}_w(G_u) + \mathrm{mvc}(G_v), \mathrm{mvc}(G) + 1\}$ *and* $\mathrm{mvc}_{uv}(G) = \min\{\mathrm{mvc}_u(G_u) + \mathrm{mvc}_v(G_v), \mathrm{mvc}(G) + 1\}$.

From Lemma 1 and Lemma 2, the following theorem is immediate.

Theorem 2. *Let G be a maximal outerplanar graph and uv be an edge on the outer face of G such that $deg_G(u) > 2$ and $deg_G(v) > 2$. Let $w = \Delta(uv)$.*

1. *Given $\mathscr{M}(G_u, uw)$ and $\mathscr{M}(G_v, vw)$, it is possible to compute $\mathrm{mvc}(G)$ in constant time.*
2. *Given $\mathrm{mvc}(G)$, $\mathscr{M}(G_u, uw)$ and $\mathscr{M}(G_v, vw)$, it is possible to compute the remaining mvc parameters of G with respect to uv in constant time.*

4. Bounds on the evc Parameters

In this section, we will derive some bounds on the evc parameters of a maximal outerplanar graph. These bounds will be used in the next section for the recursive computation of the evc parameters. Throughout this section, we consider G to be a maximal outerplanar graph on at least four vertices, uv an edge on its outer face and $w = \Delta(uv)$. First, we derive some bounds for $\mathscr{E}(G, uv)$ when $deg_G(u) = 2$.

The proof of the lemma below is easy to obtain by repeated applications of Proposition 1.

Lemma 3. *If u is a degree-2 vertex in G, then:*

1. $\mathrm{evc}(G) \leq \mathrm{evc}_u(G) = \mathrm{evc}(G \setminus u) + 1$
2. $\mathrm{evc}_{uv}(G) = \mathrm{evc}_v(G \setminus u) + 1$
3. $\mathrm{evc}(G) \geq max\{\mathrm{mvc}(G \setminus u) + 1, \mathrm{evc}_{vw}(G \setminus u)\}$
4. $\mathrm{evc}_v(G) = max\{\mathrm{mvc}_{uv}(G), \mathrm{evc}(G)\}$

Now, we derive some upper bounds for $\mathscr{E}(G, uv)$ when $deg_G(u) > 2$ and $deg_G(v) > 2$. The following proposition is a direct consequence of Proposition 1.

Proposition 5.

1. $\max_{x \in V(G_v)} \mathrm{mvc}_x(G) \leq \min\{\mathrm{mvc}_w(G_u) + \mathrm{evc}_{vw}(G_v) - 1, \mathrm{mvc}_{uw}(G_u) + \mathrm{evc}_w(G_v) - 1, \mathrm{mvc}(G_u) + \mathrm{evc}(G_v)\}$
2. $\max_{x \in V(G_u)} \mathrm{mvc}_x(G) \leq \min\{\mathrm{evc}_{uw}(G_u) + \mathrm{mvc}_w(G_v) - 1, \mathrm{evc}_w(G_u) + \mathrm{mvc}_{vw}(G_v) - 1, \mathrm{evc}(G_u) + \mathrm{mvc}(G_v)\}$

The proof of Lemma 4 follows easily from propositions 1 and 5.

Lemma 4. *If $deg_G(u) > 2$ and $deg_G(v) > 2$, then:*

1. $\mathrm{evc}(G) \leq \min\{\mathrm{mvc}(G)+1, \mathrm{evc}_w(G_u)+\mathrm{evc}_w(G_v)-1\}$
2. $\mathrm{evc}(G) \leq \max\{U, V\}$, *where* $U = \min\{\mathrm{evc}_{uw}(G_u)+\mathrm{mvc}_w(G_v)-1, \mathrm{evc}_w(G_u)+\mathrm{mvc}_{vw}(G_v)-1, \mathrm{evc}(G_u)+\mathrm{mvc}(G_v)\}$ *and* $V = \min\{\mathrm{mvc}_w(G_u)+\mathrm{evc}_{vw}(G_v)-1, \mathrm{mvc}_{uw}(G_u)+\mathrm{evc}_w(G_v)-1, \mathrm{mvc}(G_u)+\mathrm{evc}(G_v)\}$

In a similar way, we can also prove Lemma 5 and Lemma 6, which give upper bounds on $\mathrm{evc}_v(G)$ and $\mathrm{evc}_{uv}(G)$ respectively.

Lemma 5. *If* $deg_G(u) > 2$ *and* $deg_G(v) > 2$*, then:*

1. $\mathrm{evc}_v(G) \leq \min\{\mathrm{evc}(G)+1, \mathrm{mvc}_v(G)+1, \mathrm{evc}_w(G_u)+\mathrm{evc}_{vw}(G_v)-1\}$
2. $\mathrm{evc}_v(G) \leq \max\{U, V\}$, *where* $U = \min\{\mathrm{evc}(G_u)+\mathrm{mvc}_v(G_v), \mathrm{evc}_w(G_u)+\mathrm{mvc}_{vw}(G_v)-1\}$ *and* $V = \min\{\mathrm{mvc}(G_u)+\mathrm{evc}_v(G_v), \mathrm{mvc}_w(G_u)+\mathrm{evc}_{vw}(G_v)-1\}$

Lemma 6. *If* $deg_G(u) > 2$ *and* $deg_G(v) > 2$*, then:*

1. $\mathrm{evc}_{uv}(G) \leq \min\{\mathrm{evc}(G)+1, \mathrm{mvc}_{uv}(G)+1, \max\{P_1, P_2\}\}$, *where* $P_1 = min\{\mathrm{evc}_{uw}(G_u)+\mathrm{mvc}_{vw}(G_v)-1, \mathrm{evc}_u(G_u)+\mathrm{mvc}_v(G_v)\}$ *and* $P_2 = min\{\mathrm{mvc}_{uw}(G_u)+\mathrm{evc}_{vw}(G_v)-1, \mathrm{mvc}_u(G_u)+\mathrm{evc}_v(G_v)\}$
2. $\mathrm{evc}_{uv}(G) \leq \min\{Q, \mathrm{evc}_{uw}(G_u)+\mathrm{evc}_{vw}(G_v)-1, \mathrm{evc}_u(G_u)+\mathrm{evc}(G_v), \mathrm{evc}_v(G_v)+\mathrm{evc}(G_u)\}$, *where* $Q = \max\{\mathrm{evc}_u(G_u)+\mathrm{evc}_{vw}(G_v)-1, \mathrm{evc}_u(G_u)+\mathrm{mvc}_v(G_v)\}$

The next lemma gives some lower bounds for $\mathscr{E}(G, uv)$ when $deg_G(u) > 2$ and $deg_G(v) > 2$.

Lemma 7. *If* $deg_G(u) > 2$ *and* $deg_G(v) > 2$*, then:*

1. $\mathrm{evc}(G) \geq \max\{\mathrm{evc}_w(G_u)+\mathrm{mvc}(G_v)-1, \mathrm{mvc}(G_u)+\mathrm{evc}_w(G_v)-1\}$
2. $\mathrm{evc}_u(G) \geq \max\{\mathrm{evc}(G), \mathrm{mvc}_{uw}(G_u)+\mathrm{evc}(G_v)-1, \mathrm{evc}_{uw}(G_u)+\mathrm{mvc}(G_v)-1, \mathrm{evc}_u(G_u)+\mathrm{mvc}_w(G_v)-1\}$
3. $\mathrm{evc}_{uv}(G) \geq \max\{\mathrm{evc}(G), \mathrm{evc}_u(G), \mathrm{evc}_v(G), \mathrm{mvc}_{uw}(G_u)+\mathrm{evc}_v(G_v)-1, \mathrm{evc}_u(G_u)+\mathrm{mvc}_{vw}(G_v)-1, \mathrm{evc}_{uw}(G_u)+\mathrm{mvc}_v(G_v)-1, \mathrm{mvc}_u(G_u)+\mathrm{evc}_{vw}(G_v)-1\}$

5. Computation of the evc Parameters

Let G be a maximal outerplanar graph with at least four vertices, uv be an edge on its outer face and $w = \Delta(uv)$. In this section, we describe the method of computing $\mathscr{E}(G, uv)$ using the bounds obtained in Section 4.

5.1. Computing $\mathscr{E}(G, uv)$ when $deg_G(u) = 2$

In this subsection, we consider the case when u is a degree-2 vertex in G. Bounds obtained in Proposition 2 and Lemma 3 together with Proposition 1 give the following.

Lemma 8. $\mathrm{evc}(G) = max\{\mathrm{mvc}(G \setminus u)+1, \mathrm{evc}_{vw}(G \setminus u)\}$.

From Lemma 3 and Lemma 8, the following theorem is immediate.

Theorem 3. *Let* G *be a maximal outerplanar graph and* u *be a degree-2 vertex in* G *with neighbors* v *and* w*.*

1. *Given* $\mathrm{mvc}(G \setminus u)$ *and* $\mathscr{E}(G \setminus u, vw)$*, it is possible to compute* $\mathrm{evc}(G)$ *in constant time.*
2. *Given* $\mathrm{evc}(G)$*,* $\mathscr{M}(G, uv)$ *and* $\mathscr{E}(G \setminus u, vw)$*, it is possible to compute the remaining evc parameters of* G *with respect to* uv *in constant time.*

5.2. Computing $\mathscr{E}(G, uv)$ when $deg_G(u) > 2$ and $deg_G(v) > 2$

Now, we will see how $\mathscr{E}(G, uv)$ can be computed when $deg_G(u) > 2$ and $deg_G(v) > 2$ using $\mathrm{mvc}(G)$, $\mathscr{M}(G_u, uw)$, $\mathscr{M}(G_v, vw)$, $\mathscr{E}(G_u, uw)$ and $\mathscr{E}(G_v, vw)$. For this subsection, we will assume that $deg_G(u) > 2$ and $deg_G(v) > 2$. We will also assume that $\mathrm{mvc}(G_u) = k_u$ and $\mathrm{mvc}(G_v) = k_v$.

Lemma 9 handles the computation of $\mathrm{evc}(G)$. The proof of this lemma uses the bounds obtained in Lemma 4 and Lemma 7.

Lemma 9.

1. *If* $\mathrm{evc}(G_u) = k_u$ *and* $\mathrm{evc}(G_v) = k_v$, *then*
   - $\mathrm{evc}(G) = \max\{\mathrm{evc}_w(G_u) + \mathrm{mvc}(G_v) - 1, \mathrm{mvc}(G_u) + \mathrm{evc}_w(G_v) - 1\}$.
2. *If* $\mathrm{evc}(G_u) = k_u$ *and* $\mathrm{evc}(G_v) = k_v + 1$, *then*
   - $\mathrm{evc}(G) = \min\{\mathrm{mvc}(G)+1, \mathrm{evc}(G_u)+\mathrm{evc}_{vw}(G_v)-1, \mathrm{mvc}_{uw}(G_u)+\mathrm{evc}_w(G_v)-1\}$.
3. *If* $\mathrm{evc}(G_u) = k_u + 1$ *and* $\mathrm{evc}(G_v) = k_v + 1$, *then*
   - $\mathrm{evc}(G) = \min\{\mathrm{mvc}(G) + 1, \max\{\mathrm{evc}_{uw}(G_u) + \mathrm{mvc}_w(G_v) - 1, \mathrm{mvc}_w(G_u) + \mathrm{evc}_{vw}(G_v) - 1\}\}$.

Lemmas 10-12 deal with the computation of the remaining parameters in $\mathscr{E}(G, uv)$ using $\mathrm{evc}(G)$, $\mathscr{M}(G, uv)$, $\mathscr{M}(G_u, uw)$, $\mathscr{M}(G_v, vw)$, $\mathscr{E}(G_u, uw)$ and $\mathscr{E}(G_v, vw)$. The proofs of these lemmas critically use the bounds stated in lemmas 5-7.

Lemma 10. *Suppose* $\mathrm{evc}(G_u) = k_u$ *and* $\mathrm{evc}(G_v) = k_v$.

1. *If* $\mathrm{evc}(G) = k_u + k_v - 1$, *then*
   - $\mathrm{evc}_u(G) = \mathrm{evc}_{uw}(G_u) + \mathrm{mvc}(G_v) - 1$.
   - $\mathrm{evc}_v(G) = \mathrm{evc}_{vw}(G_v) + \mathrm{mvc}(G_u) - 1$.
   - $\mathrm{evc}_{uv}(G) = min\{\mathrm{mvc}_{uv}(G) + 1, \mathrm{evc}_{uw}(G_u) + \mathrm{evc}_{vw}(G_v) - 1\}$.
2. *If* $\mathrm{evc}(G) = k_u + k_v$, *then*
   - $\mathrm{evc}_u(G) = \mathrm{evc}_v(G) = \mathrm{evc}(G)$.
   - $\mathrm{evc}_{uv}(G) = \min\{\mathrm{evc}_u(G_u) + \mathrm{mvc}(G_v), \mathrm{evc}_v(G_v) + \mathrm{mvc}(G_u), \mathrm{mvc}_{uv}(G) + 1\}$.

Lemma 11. *Suppose* $\mathrm{evc}(G_u) = k_u$ *and* $\mathrm{evc}(G_v) = k_v + 1$.

1. *If* $\mathrm{mvc}(G) = k_u + k_v - 1$, *then*
   - $\mathrm{evc}_u(G) = \mathrm{mvc}_{uw}(G_u) + \mathrm{evc}(G_v) - 1$.
   - $\mathrm{evc}_v(G) = \min\{\mathrm{mvc}_v(G) + 1, \mathrm{evc}_w(G_u) + \mathrm{evc}_{vw}(G_v) - 1, \max\{\mathrm{mvc}_w(G_u) + \mathrm{evc}_{vw}(G_v) - 1, \mathrm{evc}(G_u) + \mathrm{mvc}_v(G_v)\}\}$.
   - $\mathrm{evc}_{uv}(G) = \min\{\mathrm{mvc}_{uv}(G) + 1, \mathrm{evc}_{uw}(G_u) + \mathrm{evc}_{vw}(G_v) - 1, \mathrm{evc}_u(G_u) + \max\{\mathrm{evc}_{vw}(G_v) - 1, \mathrm{mvc}_v(G_v)\}\}$.
2. *If* $\mathrm{mvc}(G) = evc(G) = k_u + k_v$, *then*
   - $\mathrm{evc}_u(G) = \mathrm{evc}_u(G_u) + \mathrm{evc}_w(G_v) - 1$

- $\text{evc}_v(G) = \min\{\text{mvc}_v(G)+1, \text{evc}_w(G_u)+\text{evc}_{vw}(G_v)-1, \max\{\text{mvc}_w(G_u)+\text{evc}_{vw}(G_v)-1, \text{evc}(G_u)+\text{mvc}_v(G_v)\}\}$
- $\text{evc}_{uv}(G) = \min\{\text{evc}(G)+1, \max\{\text{evc}_u(G_u)+\text{mvc}_v(G_v), \text{evc}_u(G_u)+\text{evc}_{vw}(G_v)-1\}\}$.

3. *If* $\text{evc}(G) = k_u + k_v + 1$, *then*
   - $\text{evc}_u(G) = \text{evc}_v(G) = \text{evc}(G)$
   - $\text{evc}_{uv}(G) = \min\{\text{mvc}_{uv}(G)+1, \text{evc}(G_u)+\text{evc}_v(G_v)\}$.

Lemma 12. *Suppose* $\text{evc}(G_u) = k_u + 1$ *and* $\text{evc}(G_v) = k_v + 1$. *Then,*

1. $\text{evc}_u(G) = \min\{\text{mvc}_u(G)+1, \text{evc}_u(G_u)+\text{evc}_w(G_v)-1\}$
2. $\text{evc}_v(G) = \min\{\text{mvc}_v(G)+1, \text{evc}_v(G_v)+\text{evc}_w(G_u)-1\}$
3. - *If* $\text{mvc}_{uv}(G) \leq k_u + k_v$ *then* $\text{evc}_{uv}(G) = \text{mvc}_{uv}(G)+1$.
   - *Otherwise,* $\text{evc}_{uv}(G) = min(\text{mvc}_{uv}(G)+1, max(Q_1, Q_2))$, *where* $Q_1 = min(\text{evc}_{uw}(G_u)+\text{mvc}_{vw}(G_v)-1, \text{evc}_u(G_u)+\text{mvc}_v(G_v))$ *and* $Q_2 = min(\text{evc}_{vw}(G_v)+\text{mvc}_{uw}(G_u)-1, \text{evc}_v(G_v)+\text{mvc}_u(G_u))$.

From Lemma 9-12, the following theorem is immediate.

Theorem 4. *Let* G *be a maximal outerplanar graph and* uv *be an edge on the outer face of* G *such that* $deg_G(u) > 2$ *and* $deg_G(v) > 2$. *Let* $w = \Delta(uv)$.

1. *Given* $\mathscr{M}(G_u, uw)$, $\mathscr{E}(G_u, uw)$, $\mathscr{M}(G_v, vw)$ *and* $\mathscr{E}(G_v, vw)$, *it is possible to compute* $\text{evc}(G)$ *in constant time.*
2. *Given* $\text{evc}(G)$, $\mathscr{M}(G, uv)$, $\mathscr{M}(G_u, uw)$, $\mathscr{E}(G_u, uw)$, $\mathscr{M}(G_v, vw)$ *and* $\mathscr{E}(G_v, vw)$, *it is possible to compute the remaining evc parameters of* G *with respect to* uv *in constant time.*

6. A Linear Time Algorithm to Compute evc Number

In this section, we formulate a divide and conquer algorithm that takes a pair (G, uv) as input, where G is a maximal outerplanar graph and uv is an edge on its outer face, and recursively computes $\mathscr{M}(G, uv)$ and $\mathscr{E}(G, uv)$. The case when G is a triangle forms the base case of the recursion and it is handled using Proposition 3. Let $w = \Delta(uv)$. If $deg_G(u) > 2$ and $deg_G(v) > 2$, then the recursion works on (G_u, uw) and (G_v, vw) using Theorem 2 and Theorem 4. Otherwise, suppose u' is the degree-2 vertex among u and v and v' is the other one. In this case, the recursion works on $(G \setminus u', v'w)$ using Theorem 1 and Theorem 3.

A high level overview of our method is given in Algorithm 1. To obtain the linear time guarantee, we need to ensure that the time spent in steps other than the recursive calls is $O(1)$. Theorems 1-4 guarantee that lines $8-11$ and lines $17-20$ work in constant time, forming the most crucial part of our algorithm. However, we will have to avoid the explicit computation of the subgraphs and passing them as explicit parameters in the recursive calls using a refinement of Algorithm 1.

Algorithm 1 To compute $\mathscr{M}(G, uv)$ and $\mathscr{E}(G, uv)$ of a maximal outerplanar graph G with uv an edge on its outer face. [A high level overview]

Inputs: A maximal outerplanar graph G with at least three vertices, an edge uv on the outer face of G.
Outputs: $\mathscr{M}(G, uv)$ and $\mathscr{E}(G, uv)$.

procedure EVC_Parameters(G, uv)
  $w \leftarrow \Delta(uv)$.
  if G is a triangle **then**
    $\mathrm{mvc}(G)$=$\mathrm{evc}(G)$=$\mathrm{mvc}_u(G)$=$\mathrm{mvc}_v(G)$=$\mathrm{mvc}_{uv}(G)$=2, $\mathrm{evc}_u(G)$=$\mathrm{evc}_v(G)$=2, $\mathrm{evc}_{uv}(G)$=3.
  else if One end point of uv is degree-2 **then**
    u'=degree-2 end point of uv, v'=higher degree end point of uv.
    Recursively call EVC_Parameters($G \setminus u', v'w$).
    Compute $\mathrm{mvc}(G)$ using Theorem 1 (Part 1).
    Compute $\mathrm{mvc}_u(G)$, $\mathrm{mvc}_v(G)$ and $\mathrm{mvc}_{uv}(G)$ using Theorem 1 (Part 2).
    Compute $\mathrm{evc}(G)$ using Theorem 3 (Part 1).
    Compute $\mathrm{evc}_{uv}(G)$ in constant using Theorem 3 (Part 2).
  else
    $G_u \leftarrow uv$-segment of G containing the vertex u
    $G_v \leftarrow uv$-segment of G containing the vertex v
    Recursively call EVC_Parameters(G_u, uw).
    Recursively call EVC_Parameters(G_v, vw).
    Compute $\mathrm{mvc}(G)$ using Theorem 2 (Part 1).
    Compute $\mathrm{mvc}_u(G)$, $\mathrm{mvc}_v(G)$, $\mathrm{mvc}_{uv}(G)$ using Theorem 2 (Part 2).
    Compute $\mathrm{evc}(G)$ using Theorem 4 (Part 1).
    Compute $\mathrm{evc}_u(G)$, $\mathrm{evc}_v(G)$, $\mathrm{evc}_{uv}(G)$ using Theorem 4 (Part 2).
  end if
  return ($\mathscr{M}(G, uv)$ and $\mathscr{E}(G, uv)$)
end procedure

For the purpose of a refined algorithm, we maintain a Vertex Edge Face Adjacency List data structure using which the following operations are possible:

- For any edge e, in constant time we can traverse through the internal faces on which the edge e lies.
- For any internal face f, in constant time we can traverse through the edges on the boundary of f.

The details of the data structure is given in Figure 2. For any edge v_iv_j, there are links between the edge node v_iv_j and the edge node v_jv_i. For each face f in G, the face node representing f contains links to the edge nodes corresponding to the bounding edges of f and vice versa. Remember that each edge of G has two edge nodes corresponding to it. Hence, for each face node f, there are six pointers to edge nodes. For each face node, there is a visit field which is initialized to *unvisited*. Each vertex node has a mark field, which is initialized to *unmarked*, the

cur-edge field which is initialized to *null* and a pointer to the beginning of the adjacency list of that vertex.

```
struct vertex
{
  int vnum;
  int degree
  boolean mark;
  edgenode*head;
  edgenode *cur-edge;
}
struct vertex Varray[size]
```

```
struct edgenode
{
int vertex1;
int vertex2;
facenode *f1;
facenode *f2 ;
edgenode *pair;
edgenode *next;
}
```

```
struct facenode
{
edgenode *e1;
edgenode *pair-e1;
edgenode *e2;
edgenode *pair-e2;
edgenode *e3;
edgenode *pair-e3;
boolean visit;
}
```

Figure 2: Details of the vertex edge face adjacency list data structure.

Algorithm 2 gives a linear time refinement of Algorithm 1. To compute the evc number of a maximal outerplanar graph, we pass an edge uv on the outer face of the graph as input parameter to the algorithm. We assume that the vertex face adjacency list of this graph is maintained as a global data structure. It may be noted that the Vertex Edge Face Adjacency List of a maximal outerplanar graph can be produced in linear time by modifying the algorithm suggested by N. Chiba and T. Nishizeki [9] for listing all the triangles of a planar graph.

Using the *visit* field of faces in the data structure, we avoid passing the subgraph as an explicit parameter in recursive calls. Initially, the visit field of all (internal) faces are set to *unvisited*. Note that, the edge uv is present in exactly one unvisited face uvw initially. In subsequent recursive calls, the following invariants will be maintained:

- the edge e passed as parameter to the recursion is present in exactly one unvisited face f.
- The e-segment of G that contains the unvisited face f containing e is precisely the subgraph on which the recursive call in Algorithm 1 would have been made.

Now we explain how the correspondence between the computational steps of Algorithm 1 and Algorithm 2 are maintained.

In line number 2 of Algorithm 1, we find the unique common neighbor w of u and v in the input graph. Instead of this step, line number 2 of Algorithm 2 identifies w as the third vertex in the unique unvisited face containing the edge uv. After this, the face uvw is marked as visited in line number 4 of Algorithm 2 and subsequently the edges uw and vw are in at most one unvisited face each.

Algorithm 1 is divided into following three cases:

1. The input graph is a triangle (degrees of both the end vertices of the edge uv are 2). Refer to line number 3 of Algorithm 1.
2. The input graph is not a triangle and exactly one end vertex of the input edge uv is of degree 2. Refer to line number 5 of Algorithm 1.
3. The input graph is not a triangle and both the end vertices of the input edge uv has degree greater than 2. Refer to line number 12 of Algorithm 1.

The three cases of Algorithm 2 corresponding to the three cases of Algorithm 1 are as follows:

1. Both edges uw and vw lie in no unvisited face and thereby uvw is a triangle. Refer to line number 5 of Algorithm 2.
2. Exactly one among the edges uw and vw lie in an unvisited face. Refer to line number 8 of Algorithm 2.
3. Both uw and vw lie in one unvisited face each. Refer to line number 16 of Algorithm 2.

Algorithm 2 For computing $\mathscr{M}(G, uv)$ and $\mathscr{E}(G, uv)$ of a maximal outerplanar graph G with uv an edge on its outer face in linear time

---

Inputs: An edge uv on the outer face of the maximal outerplanar graph G that has at least three vertices.
Output: $\mathscr{M}(G, uv)$ and $\mathscr{E}(G, uv)$.
Assumption: The vertex edge face adjacency list is maintained globally.

1: **procedure** EVC_Parameters
2: Identify the unvisited face uvw in which the edge uv lie.
3: ▷ Possible in constant time by vertex edge face adjacency list
4: Set the visit field of the face uvw to visited.
5: **if** neither edge uw nor edge vw is in any unvisited faces **then**
6: ▷ Possible to check in constant time using vertex edge face adjacency list
7: $\mathrm{mvc}(G)$=$\mathrm{evc}(G)$=$\mathrm{mvc}_u(G)$=$\mathrm{mvc}_v(G)$=$\mathrm{mvc}_{uv}(G)$=2, $\mathrm{evc}_u(G)$=$\mathrm{evc}_v(G)$=2, $\mathrm{evc}_{uv}(G)$=3.
8: **else if** exactly one among uw and vw lie in an unvisited face **then**
9: ▷ Possible to check in constant time using the vertex edge face adjacency list
10: e be the edge among uw and vw that lie in an unvisited face.
11: $t \longleftarrow$ EVC_Parameters(e).
12: Compute $\mathrm{mvc}(G)$ in constant time using Theorem 1 (part 1).
13: Compute $\mathrm{mvc}_u(G)$, $\mathrm{mvc}_v(G)$ and $\mathrm{mvc}_{uv}(G)$ in constant time using Theorem 1 (part 2).
14: Compute $\mathrm{evc}(G)$ in constant time using Theorem 3 (part 1).
15: Compute $\mathrm{evc}_{uv}(G)$ in constant using Theorem 3 (part 2).
16: **else**
17: $t_1 \longleftarrow$ EVC_Parameters(uw).
18: $t_2 \longleftarrow$ EVC_Parameters(vw).
19: Compute $\mathrm{mvc}(G)$ in constant time using Theorem 2 (part 1).
20: Compute $\mathrm{mvc}_u(G)$, $\mathrm{mvc}_v(G)$, $\mathrm{mvc}_{uv}(G)$ in constant time using Theorem 2 (part 2).
21: Compute $\mathrm{evc}(G)$ in constant time using Theorem 4 (part 1).
22: Compute $\mathrm{evc}_u(G)$, $\mathrm{evc}_v(G)$, $\mathrm{evc}_{uv}(G)$ in constant time using Theorem 4 (part 2).
23: **end if**
24: **return** ($\mathscr{M}(G, uv)$ and $\mathscr{E}(G, uv)$)
25: **end procedure**

Now, we show that line number 3 of Algorithm 1 is equivalent to line number 5 of Algorithm 2. In line number 3 of Algorithm 1, we check whether the input graph is a triangle, which is the

base case. Equivalently, by the invariants stated above, in Algorithm 2, it is enough to check if the uv-segment of G containing the face uvw is a triangle. In line 5 of Algorithm 2, we do the following in constant time using vertex edge face adjacency list: We check if uw or vw lie in any unvisited face. If neither uw nor vw lie in an unvisited face, then we can infer that uv-segment of G containing the face uvw is a triangle.

In line 5 of Algorithm 1, we check whether exactly one end vertex of the edge uv is of degree 2. In line number 8 of Algorithm 2, if uw (respectively, vw) is in any univisited face, then we can infer that the degree of the vertex u (respectively, v) is greater than 2 in the uv-segment of G that contains the face uvw. Otherwise, the degree of u (respectively, v) in the uv-segment of G that contains the face uvw is equal to 2.

In line number 7 of Algorithm 1, we recursively call the function EVC_Parameters with two input parameters: (1) the graph obtained by deleting the degree-2 endpoint u' of the edge uv from the input graph and (2) the edge $v'w$ bounding the face uvw, where degree of v' is greater than 2 in the input graph. Since in Algorithm 2, uvw is already marked as visited in line number 4, it is enough to invoke the recursive call with the edge e as parameter, where e is the bounding edge of the face uvw with one unvisited face adjacent to it. This is achieved in line number 12 of Algorithm 2, in which we recursively call the function EVC_Parameters with the edge e as input, where e is that edge among the edges uw and vw which lies in an unvisited face. Note that the invariants of the Algorithm 2 are maintained.

The equivalence between line number 12 of Algorithm 1 and line number 16 of Algorithm 2 follows from our arguments so far. In line number 15 (respectively, 16) of Algorithm 1, a recursive call to the algorithm is made with input parameters: (1) G_u (respectively, G_v), the uw-segment (respectively, vw-segment) of G that does not contain the edge vw (respectively, uw) and (2) the edge uw (respectively, vw). Note that the edges bounding the face uvw, other than uv, are used as parameters in these two calls. Equivalently, in line number 17 and 18 of Algorithm 2, recursive calls are made respectively with the edges uw and vw as input parameters. Since the face uvw is marked as visited already, the invariants of the algorithm are maintained here as well.

Thus, we can see that Algorithm 2 is a linear time implementation of Algorithm 1.

7. Conclusion

This paper presents a linear time algorithm for computing the eternal vertex cover number of maximal outerplanar graphs, lowering the best known upper bound to the complexity of the problem from quadratic [7] time to linear. The techniques presented in the paper make crucial use of the planarity of the underlying graph to yield a divide and conquer algorithm for computing the evc number of a maximal outerplanar graph. Attempts to generalize our techniques to maximal planar graphs may not be successful due to the known NP-hardness result on the evc computation of biconnected internally triangulated planar graphs [3]. The complexity status of the problem of computing the evc number of outerplanar graphs is open, and may be attempted using some techniques developed in this work. However, the observation that for a chordal graph, any vertex cover containing all its cut vertices is connected [3], which yields Proposition 1 that is extensively used in this work, does not hold for outerplanar graphs.

References

[1] F. V. Fomin, S. Gaspers, P. A. Golovach, D. Kratsch, S. Saurabh, Parameterized algorithm for eternal vertex cover, Information Processing Letters 110 (2010) 702 – 706.

[2] N. Misra, S. Nanoti, Eternal vertex cover on bipartite and co-bipartite graphs, CoRR abs/2201.03820 (2022). URL: https://arxiv.org/abs/2201.03820. arXiv:2201.03820.

[3] J. Babu, L. S. Chandran, M. Francis, V. Prabhakaran, D. Rajendraprasad, N. J. Warrier, On graphs whose eternal vertex cover number and vertex cover number coincide, Discrete Applied Mathematics 319 (2022) 171–182. doi:https://doi.org/10.1016/j.dam.2021.02.004.

[4] W. Klostermeyer, C. Mynhardt, Edge protection in graphs, Australasian Journal of Combinatorics 45 (2009) 235 – 250.

[5] H. Araki, T. Fujito, S. Inoue, On the eternal vertex cover numbers of generalized trees, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E98.A (2015) 1153–1160. doi:10.1587/transfun.E98.A.1153.

[6] J. Babu, V. Prabhakaran, A. Sharma, A substructure based lower bound for eternal vertex cover number, Theoretical Computer Science (2021). doi:https://doi.org/10.1016/j.tcs.2021.08.018.

[7] J. Babu, V. Prabhakaran, A new lower bound for the eternal vertex cover number of graphs, Journal of Combinatorial Optimization (2021) 1–17.

[8] J. Babu, K. M. Krishnan, V. Prabhakaran, N. J. Warrier, Eternal vertex cover number of maximal outerplanar graphs, 2022. URL: https://arxiv.org/abs/2201.06577.

[9] N. Chiba, T. Nishizeki, Arboricity and subgraph listing algorithms, SIAM Journal on computing 14 (1985) 210–223.