Eternal vertex cover number of maximal outerplanar graphs

Jasine Babu¹, K. Murali Krishnan², Veena Prabhakaran¹, and Nandini J. Warrier²

¹ Indian Institute of Technology Palakkad, India jasine@iitpkd.ac.in, veenaprabhakaran7@gmail.com
² National Institute of Technology Calicut, India kmurali@nitc.ac.in, nandini.wj@gmail.com

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 the complexity status of the problem for bipartite graphs is open. 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 recursive algorithm of linear time 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 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 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 C₀ of G, forming an initial configuration. Suppose at instant i, guards are placed in a configuration Cᵢ and an attack occurs on an arbitrary edge uv ∈ 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 Cᵢ₊₁ is reached. To ensure that all edges remain guarded at the instant i + 1, we require Cᵢ₊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 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 $\text{evc}(G)$ for an arbitrary graph $G$ is NP-hard, though in PSPACE \cite{1}, and is fixed parameter tractable with $\text{evc}(G)$ as parameter$^{[1]}$. The problem appears significantly harder than mvc computation. For instance, mvc computation is polynomial time for bipartite graphs, but the complexity status of the problem is open for bipartite graphs$^{[1]}$. Consequent to the hardness results, work on solving the problem on various graph classes have been attempted in the literature. It is known that the problem is NP-complete even for biconnected internally triangulated planar graphs$^{[2]}$. Polynomial time algorithms for computing $\text{evc}(G)$ were known exactly only for very elementary graph classes such as an $O(n)$ algorithm for trees$^{[3]}$, a polynomial time algorithm for a tree-like graph class$^{[4]}$ and a linear time algorithm for cactus graphs$^{[5]}$. A recent structure theorem developed in$^{[6]}$ has resulted in a quadratic time algorithm for chordal graphs. Similar graph protection problems defined on parameters like dominating sets$^{[7,8,9,10,11,12]}$ and independent sets$^{[13,14]}$ are also studied in literature.

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$^{[6]}$ 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.

Our algorithm takes a maximal outerplanar graph on $G$ on at least three vertices and an edge $uv$ on the outer face of $G$ and recursively computes $\text{evc}(G)$, $\text{mvc}(G)$ 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 Fig. 1. Otherwise, the recursion works on the graph obtained by deleting the degree two end point of the edge $uv$ from $G$. Since $\text{evc}(G)$ 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$^{[2]}$ to$^{[4]}$ which assert that $\text{evc}(G)$ 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 $\text{MVC}$ problem and the $\text{EVC}$ problem for maximal outerplanar graphs.

2 Preliminaries

Let $G$ be a graph with $S \subseteq V$. The parameter $\text{evc}_S(G)$ denotes the minimum number $k$ such that $G$ is $k$-defendable with all vertices of $S$ occupied in all configurations. Similarly, $\text{mvc}_S(G)$ denote the size of the smallest cardinality vertex cover of $G$ containing all vertices of $S$. When $S = \{v_1, \ldots, v_i\}$ for $1 \leq i \leq 3$, we shorten the notation $\text{evc}_{\{v_1, \ldots, v_i\}}(G)$ and $\text{mvc}_{\{v_1, \ldots, v_i\}}(G)$ as $\text{evc}_{v_1, \ldots, v_i}(G)$ and $\text{mvc}_{v_1, \ldots, v_i}(G)$ respectively.

Proposition$^{[1]}$ is an easy adaptation of a result from known literature.

**Proposition 1.**$^{[2]}$ Let $G(V, E)$ be a maximal outerplanar graph with at least two vertices and $S \subseteq V$. If for every vertex $v \in V \setminus S$, $\text{mvc}_{S \cup \{v\}}(G) = \text{mvc}_S(G)$, then $\text{evc}_S(G) = \text{mvc}_S(G)$. Otherwise, $\text{evc}_S(G) = \text{mvc}_S(G) + 1$. Hence, $\text{evc}_S(G) = \max_{v \in V(G)} \text{mvc}_{S \cup \{v\}}(G)$.

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

**Proof.** Let $\text{evc}(G) = k$. The first part of the inequality is obvious. So, we prove the second part. By Proposition$^{[1]}$ for each vertex $x \in V(G)$, $\text{mvc}_x(G) \leq k$. Since $uv \in E(G)$, this implies that
mvcc(G) ≤ k or mvccv(G) ≤ k. Therefore, for any vertex x ∈ V(G), we have mvccuv(G) ≤ k + 1. Hence by Proposition 1, evcc(G) ≤ k + 1 = 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 ∆(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 = ∆(uv). We define the graph Gu(uv) (respectively, Gv(uv)) to be the maximal biconnected outerplanar subgraph of G that satisfies the following two properties (see Fig. 1):

1. The edge uv (respectively, vw) is on the outer face of Gu(uv) (respectively, Gv(uv)).
2. Gu(uv) (respectively, Gv(uv)) does not contain the vertex v (respectively u).

Gu(uv) and Gv(uv) will be called the uv segments of G.

**Note 1.** We will write Gu and Gv instead of Gu(uv) and Gv(uv) when there is no scope for confusion.

Observe that Gu (respectively, Gv) will be a single edge, if degG(u) = 2 (respectively, degG(v) = 2).

![Fig. 1.](image)

(a) degG(u) = 2 and degG(v) > 2
(b) degG(u) > 2 and degG(v) > 2

**Fig. 1.** Gu is the uv segment of the graph that contains u and Gv is the uv segment that contains v. In (a), Gu is a single edge.

**Definition 2 (mvc and evc parameters).** For a maximal outerplanar graph G and an edge uv, the set of parameters Μ(G, uv) = {mvc(G), mvca(G), mvcc(G), mvccuv(G)} is called the (set of) mvc parameters of G with respect to uv and the set E(G, uv) = {evc(G), evca(G), evcc(G), evccuv(G)} is called the (set of) evc parameters of G with respect to uv.

The following observation, which 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.

**Observation 1** Let G be a triangle uvw. Consider the edge uv. Then, (1) mvc(G) = mvca(G) = mvcc(G) = mvccuv(G) = 2, (2) evc(G) = evca(G) = evcc(G) = 2 and evccuv(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., $\mathcal{M}(G, uv)$, can be computed in constant time if the mvc parameters of $G_w$ with respect to $uv$ - viz., $\mathcal{M}(G_u, uv)$ and the mvc parameters of $G_v$ with respect to $vw$ - viz., $\mathcal{M}(G_v, vw)$ are given.

The following theorem is a consequence of Observation 2.

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 $\mathcal{M}(G_u, uv)$, it is possible to compute $\mvc(G)$ in constant time.
2. Given $\mvc(G)$ and $\mathcal{M}(G_u, uv)$, 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 $\mathcal{M}(G, uv)$ when $\deg_G(u) > 2$ and $\deg_G(v) > 2$.

The following observation is easy to see.

Observation 2 If $\deg_G(u) = 2$ and $\deg_G(v) > 2$, then, (1) $\mvc(G) = \mvc_{uv}(G_v)$ (2) $\mvc_u(G) = \mvc(G_u) + 1$, (3) $\mvc_v(G) = \mvc(G)$ and (4) $\mvc_{uv}(G) = \mvc(G_u) + 1$.

The following theorem is a consequence of Observation 2.

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 $\mathcal{M}(G_u, uv)$, it is possible to compute $\mvc(G)$ in constant time.
2. Given $\mvc(G)$ and $\mathcal{M}(G_u, uv)$, 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 $\mathcal{M}(G, uv)$ when $\deg_G(u) > 2$ and $\deg_G(v) > 2$.

The following observation is easy to obtain.

Observation 3 If $\deg_G(u) > 2$ and $\deg_G(v) > 2$, then

$\mvc(G) \in \{\mvc(G_u) + \mvc(G_v) - 1, \mvc(G_u) + \mvc(G_v)\}$.

The next lemma gives a method to compute $\mvc(G)$ using $\mathcal{M}(G_u, uv)$ and $\mathcal{M}(G_v, vw)$.

Lemma 1. If $\deg_G(u) > 2$ and $\deg_G(v) > 2$, then

$\mvc(G) = \min\{\mvc_{uw}(G_u) + \mvc_{uv}(G_v) - 1, \mvc_{uw}(G_u) + \mvc_{uw}(G_v) - 1, \mvc(G_u) + \mvc(G_v)\}$.

Proof. It can be easily seen that $\mvc(G) \leq \min\{\mvc_u(G_u) + \mvc_{uv}(G_v) - 1, \mvc_{uw}(G_u) + \mvc_{uw}(G_v) - 1, \mvc(G_u) + \mvc(G_v)\}$. By Observation 3, we have the following two cases to consider.

If $\mvc(G) = \mvc(G_u) + \mvc(G_v)$, then both $G_u$ and $G_v$ have a minimum vertex cover with $w$. Further, either $\mvc_{uw}(G_u) = \mvc(G_u)$ or $\mvc_{uw}(G_v) = \mvc(G_v)$. Hence, it is clear that $\mvc(G) = \mvc(G_u) + \mvc(G_v) - 1 = \min\{\mvc_u(G_u) + \mvc_{uv}(G_v) - 1, \mvc_{uw}(G_u) + \mvc_{uw}(G_v) - 1\}$.

From this, the result follows.

If $\mvc(G) = \mvc(G_u) + \mvc(G_v)$, then by the upper bound shown in the beginning of this proof, $\mvc(G_u) + \mvc(G_v) = \mvc(G) \leq \min\{\mvc_u(G_u) + \mvc_{uv}(G_v) - 1, \mvc_{uw}(G_u) + \mvc_u(G_v) - 1\}$.

From this, the result follows.

The next lemma shows that given $\mvc(G)$, $\mathcal{M}(G_u, uw)$ and $\mathcal{M}(G_v, vw)$, the values of $\mvc_u(G)$, $\mvc_v(G)$ and $\mvc_{uw}(G)$ are computable in constant time.

Lemma 2. Let $\deg_G(u) > 2$ and $\deg_G(v) > 2$. If $\mvc(G) = \mvc(G_u) + \mvc(G_v) - 1$, then

$\mvc_u(G) = \mvc_{uw}(G_u) + \mvc(G_v) - 1$, 
$\mvc_v(G) = \mvc(G_u) + \mvc_{vw}(G_v) - 1$ and 
$\mvc_{uw}(G) = \mvc_{uw}(G_u) + \mvc_{vw}(G_v) - 1$. 

Proof. Let $\text{mvc}(G_u) = k_u$ and $\text{mvc}(G_v) = k_v$. Since $\text{mvc}(G) = k_u + k_v - 1$, both $G_u$ and $G_v$ have a minimum vertex cover with $w$. Let $k = \text{mvc}_w(G)$. Since $\text{mvc}_w(G_v) = k_v$, we can observe that $G$ has a vertex cover $C$ of size $k$ such that $u, w \in C$. Now, it can be easily seen that $C$ is the union of a vertex cover of $G_u$ containing both $u$ and $w$ and a minimum vertex cover of $G_v$ with $w$. Therefore, $\text{mvc}_u(G) = \text{mvc}_w(G_u) + \text{mvc}(G_v) - 1$. Similarly, it is easy to see that
\[ \text{mvc}(G) = \text{mvc}(G_u) + \text{mvc}_w(G_v) - 1 \]
and
\[ \text{mvc}_w(G) = \text{mvc}_w(G_u) + \text{mvc}(G_v) - 1. \]

Lemma 3. Let $\deg_G(u) > 2$ and $\deg_G(v) > 2$. If $\text{mvc}(G) = \text{mvc}(G_u) + \text{mvc}(G_v)$, then
\[ \text{mvc}_u(G) = \min\{\text{mvc}_u(G_u) + \text{mvc}(G_v), \text{mvc}(G_u) + \text{mvc}_w(G_v), \text{mvc}(G) + 1\}, \]
\[ \text{mvc}_v(G) = \min\{\text{mvc}(G_u) + \text{mvc}_v(G_v), \text{mvc}_w(G_u) + \text{mvc}(G_v), \text{mvc}(G) + 1\} \]
and
\[ \text{mvc}_{uw}(G) = \min\{\text{mvc}_u(G_u) + \text{mvc}_v(G_v), \text{mvc}(G) + 1\}. \]

Proof. Let $\text{mvc}(G_u) = k_u$ and $\text{mvc}(G_v) = k_v$. First we prove that $\text{mvc}_u(G) = \min\{\text{mvc}_u(G_u) + \text{mvc}(G_v), \text{mvc}(G_u) + \text{mvc}_w(G_v), \text{mvc}(G) + 1\}$. It can be easily seen that $\text{mvc}_u(G) \leq \min\{\text{mvc}_u(G_u) + \text{mvc}(G_v), \text{mvc}(G_u) + \text{mvc}_w(G_v), \text{mvc}(G) + 1\}$. Now, we need to show that $\text{mvc}_u(G) \leq \min\{\text{mvc}_u(G_u) + \text{mvc}(G_v), \text{mvc}(G_u) + \text{mvc}_w(G_v), \text{mvc}(G) + 1\}$. We know that $\text{mvc}_u(G) \in \{\text{mvc}(G), \text{mvc}(G) + 1\}$. If $\text{mvc}_u(G) = \text{mvc}(G) + 1$, then the proof is immediate, because $\text{mvc}_u(G)$ is equal to one of the three quantities listed and so it is greater than or equal to their minimum. Otherwise, we have $\text{mvc}_u(G) = \text{mvc}(G) = \text{mvc}(G_u) + \text{mvc}(G_v)$. In this case, it is enough to show that among the three parameters, the first or the second will be equal to $\text{mvc}(G_u) + \text{mvc}(G_v)$. For contradiction, suppose this was not the case. Then, $\text{mvc}_u(G_u) = k_u + 1$ and $\text{mvc}_v(G_v) = k_v + 1$. Consider any minimum vertex cover $S$ of $G$ such that $u \in S$. If $w \in S$, then $|S| \geq \text{mvc}_u(G_u) + \text{mvc}_w(G_v) - 1 \geq k_u + k_v + 1$, a contradiction. If $w \notin S$, again $|S| \geq \text{mvc}_u(G_u) + \text{mvc}_v(G_v) \geq k_u + k_v + 1$, a contradiction. Hence, we are done. The proof for $\text{mvc}_v(G)$ is symmetric to the previous case.

Now, we show that $\text{mvc}_{uw}(G) = \min\{\text{mvc}_u(G_u) + \text{mvc}_v(G_v), \text{mvc}(G) + 1\}$. It is easy to see that $\text{mvc}_{uw}(G) \leq \min\{\text{mvc}_u(G_u) + \text{mvc}_v(G_v), \text{mvc}(G) + 1\}$. We know that $\text{mvc}_{uw}(G) \in \{\text{mvc}(G), \text{mvc}(G) + 1\}$. Suppose $\text{mvc}_{uw}(G) = \text{mvc}(G) = k_u + k_v$. In this case, it suffices to show that $\text{mvc}_u(G_u) = k_u$ and $\text{mvc}_v(G_v) = k_v$. For contradiction, assume $\text{mvc}_u(G_u) = k_u + 1$. Then, $\text{mvc}_w(G_u) = k_u$. Since, $\text{mvc}_{uw}(G) = k_u + k_v$, $\text{mvc}_w(G_v) = k_v$. Then, $\text{mvc}(G) = k_u + k_v - 1$, which is a contradiction. Hence, $\text{mvc}_u(G_u) \leq k_u$. Similarly, $\text{mvc}_v(G_v) \leq k_v$ as required. Now, suppose $\text{mvc}_{uw}(G) = \text{mvc}(G) + 1 = k_u + k_v + 1$. In this case, it suffices to show that $\text{mvc}_u(G_u) + \text{mvc}_v(G_v) \geq \text{mvc}_{uw}(G) = \text{mvc}(G) + 1$, which is straightforward to obtain. □

From Lemma 1, Lemma 2 and Lemma 3, 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 $\mathcal{M}(G_u, uv)$ and $\mathcal{M}(G_v, vw)$, it is possible to compute $\text{mvc}(G)$ in constant time.
2. Given $\text{mvc}(G)$, $\mathcal{M}(G_u, uv)$ and $\mathcal{M}(G_v, vw)$, it is possible to compute the remaining $\text{mvc}$ parameters of $G$ with respect to $uv$ in constant time.

4 Bound 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 $\delta(G, uv)$ when $\deg_G(u) = 2$.

**Lemma 4.** If $u$ is a degree-2 vertex in $G$, then:

1. $\text{evc}(G) \leq \text{evc}_u(G) = \text{evc}(G_v) + 1$
2. $\text{evc}_{uv}(G) = \text{evc}_v(G_v) + 1$
3. $\text{evc}(G) \geq \max\{\text{mvc}(G_v) + 1, \text{evc}_{uw}(G_v)\}$
4. $\text{evc}_v(G) = \max\{\text{mvc}_{uw}(G), \text{evc}(G)\}$

**Proof.** Let $\text{evc}(G) = k$.

1. First inequality is straightforward. So, it suffices to show that $\text{evc}_u(G) = \text{evc}(G_v) + 1$. From Proposition 1, $\max_{x \in V(G_v)} \text{mvc}_x(G_v) = \text{evc}(G_v)$. Hence, $\max_{x \in V(G_v)} \text{mvc}_{ev}(G_v) = \text{evc}(G_v) + 1$. Hence, by Proposition 1, we get $\text{evc}_u(G) = \text{evc}(G_v) + 1$.

2. Similar to the previous case, by Proposition 1, $\max_{x \in V(G_v)} \text{mvc}_x(G_v) = \text{evc}(G_v)$. Hence, $\max_{x \in V(G_v)} \text{mvc}_{uw}(G_v) \leq \text{evc}(G_v) + 1$. Hence, by Proposition 1, we get $\text{evc}_{uv}(G) \leq \text{evc}(G_v) + 1$. By Proposition 2 and Part 1 of this lemma, $\text{evc}_{uv}(G) \geq \text{evc}_v(G) = \text{evc}(G_v) + 1$. Therefore, $\text{evc}_u(G) = \text{evc}(G_v) + 1$.

3. By Observation 2, we get $\text{evc}_u(G) = \text{mvc}(G_v) + 1$. By Proposition 1, since $\text{evc}(G) \geq \text{evc}_u(G)$, we have $\text{evc}(G) \geq \text{mvc}(G_v) + 1$.

   By Proposition 1, for any vertex $x \in V(G)$, $\text{mvc}_x(G) \leq k$. Further, any vertex cover of $G$ contain at least two vertices among $u$, $v$, and $w$. Let $x$ be an arbitrary vertex in $G_v$ and $C$ be a vertex cover of $G$ of size $k$ containing $x$. If $C$ contains both $v$ and $w$, the vertex cover obtained by restricting $C$ to $V(G_v)$ gives a vertex cover of $G_v$ of size at most $k$. If $G$ does not contain $v$ (respectively, $w$), then $C' = C \setminus \{u\} \cup \{v\}$ (respectively, $C' = C \setminus \{u\} \cup \{w\}$) is a vertex cover of $G_v$ of size $k$ that contains $x$, $v$, and $w$. Therefore, for any vertex $x \in V(G_v)$, $\text{mvc}_{uw}(G_v) \leq k$ containing $x$ along with both $v$ and $w$. By Proposition 1, $\text{evc}_{uw}(G_v) \leq k = \text{evc}(G_v)$.

4. By Proposition 1, $\text{evc}_v(G) \geq \text{mvc}(G_v)$ and it is easy to see that $\text{evc}_v(G) \geq \text{evc}(G)$. Hence, $\text{evc}_v(G) \geq \max\{\text{mvc}(G_v), \text{evc}(G)\}$. Now, we need to show that $\text{evc}_v(G) \geq \max\{\text{mvc}_{uw}(G), \text{evc}(G)\}$. By Proposition 2, $\text{evc}_v(G) \leq \text{evc}(G) + 1$. If $\text{evc}_v(G) = \text{evc}(G)$, then we are done. Hence, suppose $\text{evc}_v(G) = \text{evc}(G) + 1$. In this case, it is enough to show that $\text{mvc}_{uw}(G) \geq \text{evc}(G) + 1$. By Proposition 1, we know that $\text{evc}(G) = \max_{x \in V(G)} \text{mvc}_x(G)$. Since degree of $u$ is 2, this implies that for any vertex $x \in V(G) \setminus \{u\}$, $\text{mvc}_{uw}(G) \leq \text{evc}(G)$. If $\text{mvc}_{uw}(G) \leq \text{evc}(G)$, then by Proposition 1, $\text{evc}(G) = \text{evc}(G)$ which is a contradiction. Hence, $\text{mvc}_{uw}(G) \geq \text{evc}(G) + 1$ as required.

Now, we derive some upper bounds for $\delta(G, uv)$ when $\deg_G(u) > 2$ and $\deg_G(v) > 2$.

**Proposition 3.** 1. $\max_{x \in V(G_v)} \text{mvc}_x(G) \leq \min\{\text{mvc}_{uw}(G_u) + \text{evc}_{uw}(G_v) - 1, \text{mvc}_{uw}(G_u) + \text{evc}_{uw}(G_v)\}$
2. $\max_{x \in V(G_v)} \text{mvc}_x(G) \leq \min\{\text{evc}_{uw}(G_u) + \text{mvc}_{uw}(G_v) - 1, \text{evc}_{uw}(G_u) + \text{mvc}_{uw}(G_v)\}$

**Proof.** Since Parts 1 and 2 are symmetric, we only prove Part 1. Consider a vertex $x \in V(G_v)$. It can be easily seen that $\text{mvc}_x(G) \leq \min\{\text{mvc}_{uw}(G_u) + \text{evc}_{uw}(G_v) - 1, \text{mvc}_{uw}(G_u) + \text{evc}_{uw}(G_v)\}$. By Proposition 1, we get $\text{mvc}_x(G) \leq \min\{\text{mvc}_{uw}(G_u) + \text{evc}_{uw}(G_v) - 1, \text{mvc}_{uw}(G_u) + \text{evc}_{uw}(G_v)\}$. Now, it remains to prove that $\text{mvc}_x(G) \leq \text{evc}(G_v) + \text{evc}(G_v)$. If $\text{mvc}_x(G_v) = \text{evc}(G_v)$, then $\text{mvc}_x(G) \leq \text{mvc}(G_v) + \text{evc}(G_v) \leq \text{evc}(G_v) + \text{evc}(G_v)$ by Proposition 1. Otherwise, it
is the case that $\text{mvc}_x(G_u) = \text{mvc}_x(G_v)$. This implies $\text{mvc}_x(G) \leq \text{mvc}_u(G_u) + \text{mvc}_w(G_v) - 1$. Since, $\text{mvc}_u(G_u) \leq \text{mvc}(G_u) + 1$, we get $\text{mvc}_x(G) \leq \text{mvc}(G_u) + \text{mvc}_w(G_v) = \text{mvc}(G_u) + \text{mvc}_x(G_v) \leq \text{mvc}(G_u) + \text{evc}(G_v)$ by Proposition 4. □

Lemma 5 gives an upper bound for $\text{evc}(G)$ when $\text{deg}_G(u) > 2$ and $\text{deg}_G(v) > 2$.

**Lemma 5.** If $\text{deg}_G(u) > 2$ and $\text{deg}_G(v) > 2$, then:

1. $\text{evc}(G) \leq \min\{\text{mvc}(G) + 1, \text{mvc}_u(G_u) + \text{evc}_w(G_v) - 1\}$
2. $\text{evc}(G) \leq \max\{U, V\}$ where
   - $U = \min\{\text{mvc}_u(G_u) + \text{mvc}_w(G_v) - 1, \text{mvc}_u(G_u) + \text{mvc}_w(G_v) - 1, \text{mvc}(G_u) + \text{mvc}(G_v)\}$ and
   - $V = \min\{\text{mvc}_u(G_u) + \text{evc}_w(G_v) - 1, \text{mvc}_u(G_u) + \text{evc}_w(G_v) - 1, \text{mvc}(G_u) + \text{evc}(G_v)\}$

**Proof.**
1. By Proposition 1 it is easy to see that $\text{evc}(G) \leq \min\{\text{mvc}(G) + 1, \text{mvc}_u(G_u) + \text{evc}_w(G_v) - 1\}$.
2. By Proposition 3 we have $\max_{x \in V(G)} \text{mvc}_x(G) = \max(U, V)$. Hence, by Proposition 1 we get $\text{evc}(G) \leq \max(U, V)$.

**Proposition 4.**

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

**Proof.**
1. It is easy to see that for any vertex $x \in V(G_v)$, $\text{mvc}_x(G_v) \leq \text{mvc}_w(G_u) + \text{mvc}_xw(G_v) - 1$ and $\text{mvc}_x(G) \leq \text{mvc}(G_u) + \text{mvc}_x(G_v)$. By Proposition 1 we get the result.
2. It is easy to see that for any vertex $x \in V(G_u)$, $\text{mvc}_x(G) \leq \text{mvc}_x(G_u) + \text{mvc}_v(G_v)$ and $\text{mvc}_x(G) \leq \text{mvc}_x(G_u) + \text{mvc}_x(G_v) - 1$. By Proposition 1 we get the result. □

Lemma 6 gives an upper bound for $\text{evc}_v(G)$ when $\text{deg}_G(u) > 2$ and $\text{deg}_G(v) > 2$.

**Lemma 6.** If $\text{deg}_G(u) > 2$ and $\text{deg}_G(v) > 2$, then:

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

**Proof.**
1. By Proposition 2 we know that $\text{evc}_v(G) \leq \text{evc}(G) + 1$. By Proposition 4 $\max_{x \in V(G_u)} \text{mvc}_x(G) \leq \text{evc}_v(G_u) + \text{evc}_w(G_v) - 1 \leq \text{evc}(G_u) + \text{evc}_v(G_v) - 1$ and $\max_{x \in V(G_v)} \text{mvc}_x(G) \leq \text{mvc}(G_u) + \text{evc}_w(G_v) - 1 \leq \text{evc}(G_u) + \text{evc}_w(G_v) - 1$. By Proposition 1 we get $\text{evc}_v(G) \leq \text{evc}_w(G_u) + \text{evc}_w(G_v) - 1$. By Proposition 1 we also get $\text{evc}_v(G) \leq \text{mvc}_v(G_u) + 1$. Hence, we are done.
2. By Proposition 4 we have $\max_{x \in V(G_u)} \text{mvc}_x(G) \leq U$ and $\max_{x \in V(G_v)} \text{mvc}_x(G) \leq V$. Therefore, by Proposition 1 we get $\text{evc}_v(G) \leq \max_{x \in V(G)} \text{mvc}_x(G) \leq \max\{U, V\}$.

Proposition 5 and Lemma 7 are symmetric to Proposition 4 and Lemma 6 respectively.

**Proposition 5.**

1. $\max_{x \in V(G_u)} \text{mvc}_x(G) \leq \min\{\text{mvc}_u(G_u) + \text{evc}(G_v), \text{mvc}_u(G_u) + \text{evc}_w(G_v) - 1\}$
2. $\max_{x \in V(G_v)} \text{mvc}_x(G) \leq \min\{\text{mvc}_v(G_u) + \text{evc}(G_v), \text{mvc}_w(G_u) + \text{evc}_w(G_v) - 1\}$

**Lemma 7.** If $\text{deg}_G(u) > 2$ and $\text{deg}_G(v) > 2$, then:

1. $\text{evc}_u(G) \leq \min\{\text{evc}(G) + 1, \text{mvc}_u(G_u) + \text{evc}_w(G_v) - 1\}$
2. \( evc_u(G) \leq \max\{U, V\} \), where \( U = \min\{evc_u(G_u) + mvc(G_v), evc_u(G_u) + evc(G_v) - 1\} \) and \( V = \min\{mvc_u(G_u) + evc(G_v), mvc_u(G_u) + evc(G_v) - 1\} \)

**Proposition 6.**
1. \( \max_{x \in V(G_u)} mvc_{xuv}(G) \leq \min\{mvc_{uw}(G_u) + mvc(G_v) - 1, evc(G_u) + mvc(G_v)\} \)
2. \( \max_{x \in V(G_v)} mvc_{xuv}(G) \leq \min\{mvc_u(G_u) + evc(G_v) - 1, mvc_u(G_u) + evc(G_v)\} \)

**Proof.**
1. Consider any vertex \( x \in V(G_u) \). By Proposition 1, we know that \( mvc_{xuv}(G_u) \leq evc(G_u) \).

   This implies \( mvc_{xuv}(G) \leq evc(G_u) + mvc(G_v) - 1 \). Similarly, by Proposition 1, we have \( mvc_{xuv}(G_v) \leq evc(G_u) + mvc(G_v) \). Thus, for any \( x \in V(G_u) \), we have \( mvc_{xuv}(G) \leq \min\{mvc_{uw}(G_u) + mvc(G_v) - 1, evc(G_u) + mvc(G_v)\} \).

2. This case is symmetric to the previous one.

**Lemma 8.** If \( deg_G(u) > 2 \) and \( deg_G(v) > 2 \), then:

1. \( evc_{uw}(G) \leq \min\{evc(G) + 1, mvc_{uw}(G) + 1, \max\{P_1, P_2\}\} \), where \( P_1 = \min\{evc_u(G_u) + mvc_u(G_v) - 1, evc_u(G_u) + mvc_u(G_v)\} \) and \( P_2 = \min\{mvc_{uw}(G_u) + evc_u(G_v) - 1, mvc_u(G_u) + evc(G_v)\} \)

2. \( evc_{uw}(G) \leq \min\{Q, evc_u(G_u) + mvc_u(G_v) - 1, evc_u(G_u) + evc(G_v), evc(G_v) + evc_u(G_u)\} \), where \( Q = \max\{evc_u(G_u) + evc_u(G_v) - 1, evc_u(G_u) + mvc_u(G_v)\} \)

**Proof.**
1. By Proposition 2, we know that \( evc_u(G_u) \leq evc(G) + 1 \). By Proposition 1, we know that \( evc_{uw}(G) \leq mvc(G) + 1 \). By Proposition 1 and Proposition 6, we get \( evc_{uw}(G) = \max_{x \in V(G)} mvc_{xuv}(G) \leq \max(P_1, P_2) \).

2. It can be easily seen that \( evc_{uw}(G) \leq evc_u(G_u) + mvc_u(G_v) - 1 \). Now, we show that \( evc_{uw}(G) \leq Q \). By Proposition 6, we get \( \max_{x \in V(G)} mvc_{xuv}(G) \leq \max\{mvc_u(G_u) + evc_u(G_v) - 1, evc_u(G_u) + mvc_u(G_v)\} \). Hence, by Proposition 1, we get \( evc_{uw}(G) \leq Q \).

Next, we show that \( evc_u(G_u) \leq evc_u(G_u) + evc(G_v) \). By Proposition 1 and Proposition 6, we get \( \max_{x \in V(G_u)} mvc_{xuv}(G_u) \leq evc_u(G_u) + mvc_u(G_v) \leq evc_u(G_u) + evc(G_v) \). Since \( mvc_u(G_u) \leq evc_u(G_u) \) by Proposition 1 and \( evc_u(G_u) \leq evc_u(G_u) + evc(G_v) \) by Proposition 1 and \( evc_u(G_u) \leq evc_u(G_u) + evc(G_v) + 1 \) by Proposition 6, we get \( \max_{x \in V(G_u)} mvc_{xuv}(G_u) \leq evc_u(G_u) + evc_u(G_v) + 1 \). Hence, by Proposition 1, we get \( evc_{uw}(G) \leq evc_u(G_u) + evc(G_v) \). Symmetrically, we can prove that \( evc_{uw}(G) \leq evc_u(G_u) + evc(G_v) \).

The following lemma gives some lower bounds for \( \varepsilon(G, uv) \) when \( deg_G(u) > 2 \) and \( deg_G(v) > 2 \).

**Lemma 9.** If \( deg_G(u) > 2 \) and \( deg_G(v) > 2 \), then:

1. \( evc(G) \geq \max\{mvc_u(G_u) + mvc(G_v) - 1, mvc_u(G_u) + evc(G_v) - 1\} \)
2. \( evc_u(G) \geq \max\{evc(G), evc_u(G_u) + evc(G_v) - 1, evc_u(G_u) + mvc(G_v) - 1\} \)
3. \( evc(G) \geq \max\{evc(G), mvc_{uw}(G_u) + evc(G) - 1, mvc_{uw}(G_u) + evc(G_v) - 1, mvc_{uw}(G_u) + evc(G) - 1\} \)

**Proof.**
1. For any vertex \( x \in V(G_u) \), consider a minimum vertex cover \( S_x \) of \( G \) of size \( mvc_{x}(G) \) containing \( x \). It is easy to see that \( |S_x \cap V(G_u)| \geq mvc(G_v) \). If \( w \in S_x \), then \( |S_x \cap V(G_u)| \geq mvc_{ux}(G_u) \) and hence, \( mvc_{x}(G) \geq mvc_{ux}(G_u) + mvc(G_v) - 1 \). If \( w \notin S_x \), then \( |S_x \cap V(G_u)| \geq mvc_{x}(G) \). Therefore, \( mvc_{x}(G) \geq mvc_{x}(G_u) + mvc(G_v) \). Since \( mvc_{ux}(G_u) \leq mvc_u(G_u) + 1 \), in this case also we get \( mvc_{x}(G) \geq mvc_{ux}(G_u) + mvc(G_v) - 1 \). By Proposition 1, we get \( evc(G) \geq \max_{x \in V(G_u)} mvc_{x}(G) \geq mvc_{ux}(G_u) + mvc(G_v) - 1 \). Symmetrically, we get \( evc(G) \geq mvc(G_u) + mvc(G_v) - 1 \). From this, the result follows.
2. By Proposition 1, it can be easily seen that evc\(_u(G)\) \geq evc(G). By Proposition 1 there exists a vertex \(x \in V(G)\) such that mvc\(_{zu}(G)\) = evc\(_u(G)\). Let \(S_{zu}\) be a vertex cover of \(G\) of size mvc\(_{zu}(G)\) containing \(x\) and \(u\). If \(w \in S_{zu}\), then it can be seen that \(|S_{zu}| \geq mvc_{uw}(G_u) + mvc_x(G_v) - 1 = mvc_{uw}(G_u) + evc(G_v) - 1\). Otherwise, we get \(|S_{zu}| \geq mvc_u(G_u) + evc(G_v) \geq mvc_{uw}(G_u) - 1 + evc(G_v)\). Hence, by Proposition 1 we get evc\(_u(G)\) \geq mvc_{uw}(G_u) + evc(G_v) - 1.

Next, we to show that evc\(_u(G)\) \geq evc\(_u(G_u)\) + mvc\(_v(G_v)\) - 1. For every vertex \(x \in V(G_u)\), mvc\(_{zu}(G)\) \geq mvc\(_{xu}(G_u)\) + mvc\(_v(G_v)\) - 1. By Proposition 1, it can be seen that evc\(_u(G)\) \geq \max_{v \in V(G_u)} mvc\(_{zu}(G)\) \geq \max_{x \in V(G_u)} mvc\(_{xu}(G_u)\) + mvc\(_v(G_v)\) - 1 = evc\(_u(G_u)\) + mvc\(_v(G_v)\) - 1. Now, it remains to show that evc\(_u(G)\) \geq evc\(_u(G_u)\) + evc\(_v(G_v)\) - 1. By Proposition 1 there exists a vertex \(x \in V(G_u)\) such that mvc\(_{zu}(G_u)\) = evc\(_u(G_u)\). Let \(S_{zu}\) be a vertex cover of \(G\) of size mvc\(_{zu}(G)\) containing \(x\) and \(u\). If \(w \in S_{zu}\), then it can be seen that \(|S_{zu}| \geq mvc_{uw}(G_u) + mvc_{uw}(G_u) - 1 = evc\(_u(G_u)\) + evc\(_v(G_v)\) - 1\). Otherwise, we get \(|S_{zu}| \geq mvc\(_u(G_u)\) + mvc\(_v(G_v)\) \geq evc\(_u(G_u)\) + evc\(_v(G_v)\) - 1. Hence, by Proposition 1 we get evc\(_u(G)\) \geq evc\(_u(G_u)\) + evc\(_v(G_v)\) - 1.

3. By Proposition 1 it can be easily seen that evc\(_v(G)\) \geq \max\{evc(G), evc\(_u(G), evc\(_v(G)\)\}. Similar to the proof of evc\(_u(G)\) \geq mvc\(_uw(G_u)\) + evc\(_v(G_v)\) in the previous case, we get evc\(_u(G)\) \geq mvc\(_uw(G_u)\) + evc\(_v(G_v)\) - 1 and evc\(_v(G)\) \geq evc\(_u(G_u)\) + mvc\(_v(G_v)\) - 1. Similar to the proof of evc\(_u(G)\) \geq mvc\(_uw(G_u)\) + evc\(_v(G_v)\) - 1 in the previous case, we get evc\(_u(G)\) \geq evc\(_uw(G_u)\) + mvc\(_v(G_v)\) - 1 and evc\(_v(G)\) \geq mvc\(_u(G_u)\) + evc\(_v(G_v)\) - 1.

\[\n\]

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 \(w = \Delta(u)\). In this section, we describe the method of computing \(\delta(G, uv)\) using the bounds obtained in Section 4.

5.1 Computing \(\delta(G, uv)\) when deg\(_G(u) = 2\)

In this subsection, we consider the case when \(u\) is a degree-2 vertex in \(G\).

**Lemma 10.** Let mvc\(_G(G)\) \neq evc\(_G(G)\). Then, evc\(_G(G) = evc\(_G(G)\) if and only if evc\(_uw\)(G\(_u\)) = evc\(_G(G)\).

**Proof.** Let mvc\(_G(G) < evc\(_G(G) \leq evc\(_uw\)(G\(_u\)). Then, by Proposition 1 evc\(_G(G) = mvc\(_G(G) + 1.**

Lemma 3 Part 3, evc\(_G(G) \geq evc\(_uw\)(G\(_u\)).

To prove the forward direction, assume evc\(_G(G) = evc\(_G(G). This gives evc\(_G(G) = evc\(_G(G) \geq evc\(_uw\)(G\(_u\)). This implies evc\(_G(G) = mvc\(_uw\)(G\(_u\)). Now, to prove the reverse direction, assume evc\(_uw\)(G\(_u\)) = evc\(_G(G). It is enough to show that evc\(_G(G) \leq evc\(_uw\)(G\(_u\)). By Proposition 1 for any vertex \(x \in V(G\(_v\)), we get mvc\(_xu\)(G\(_u\)) \leq evc\(_G(G) which implies mvc\(_x\)(G\(_v\)) \leq evc\(_G(G). Moreover, since mvc\(_u(G) \leq mvc\(_G(G) + 1 = evc\(_G(G), by Proposition 1 evc\(_G(G) \leq evc\(_uw\)(G\(_u\).**

**Lemma 11.** evc\(_G(G) = max\{mvc\(_G(G) + 1, evc\(_uw\)(G\(_u\))\}.

**Proof.** We prove this lemma by splitting into following two cases.

1. Suppose mvc\(_G(G) = evc\(_G(G). In this case, by Lemma 4 Part 1 and Part 3, we have evc\(_G(G) = evc\(_G(G) + 1 = mvc\(_G(G) + 1. Further by Part 3 of Lemma 4 evc\(_uw\)(G\(_u\)) \leq evc\(_G(G) Thus, we have evc\(_G(G) = max\{mvc\(_G(G) + 1, evc\(_uw\)(G\(_u\))\}.

\[\]
2. Suppose \( \text{mvc}(G_v) < \text{evc}(G_v) \). If \( \text{evc}_{uw}(G_v) = \text{evc}(G_v) \), then by Lemma 10 we have \( \text{evc}(G) = \text{evc}(G_v) = \text{evc}_{uw}(G_v) = \text{mvc}(G_v) + 1 \). Else if \( \text{evc}_{uw}(G_v) \neq \text{evc}(G_v) \), then by Lemma 10 and Proposition 2 we get \( \text{evc}(G) = \text{evc}(G_v) + 1 = \text{evc}_{uw}(G_v) \). In both the cases, we can observe that \( \text{evc}(G) = \max\{\text{mvc}(G_v) + 1, \text{evc}_{uw}(G_v)\} \). \( \square \)

From Lemma 4 and Lemma 11 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 \( \text{mvc}(G_v) \) and \( \text{evc}(G_v) \), it is possible to compute \( \text{evc}(G) \) in constant time.
2. Given \( \text{evc}(G), \mathcal{M}(G, uv) \) and \( \mathcal{E}(G, uv) \), it is possible to compute the remaining evc parameters of \( G \) with respect to \( uv \) in constant time.

### 5.2 Computing \( \mathcal{E}(G, uv) \) when \( deg_G(u) > 2 \) and \( deg_G(v) > 2 \)

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

Lemmas 12 and 14 handle the computation of \( \text{evc}(G) \).

**Lemma 12.** If \( \text{evc}(G_u) = k_u \) and \( \text{evc}(G_v) = k_v \), then

\[
\text{evc}(G) = \max\{\text{evc}_{uw}(G_u) + \text{mvc}(G_v) - 1, \text{mvc}(G_u) + \text{evc}_{uw}(G_v) - 1\}.
\]

**Proof.** First, suppose \( \text{evc}_{uw}(G_u) = k_u \) and \( \text{evc}(G_v) = k_v \). By Lemma 3 Part 1, we have \( \text{evc}(G) \leq k_u + k_v - 1 \). By Observation 3 and Proposition 1, we get \( \text{evc}(G) = k_u + k_v - 1 \). Note that in this case \( \text{evc}_{uw}(G_u) + \text{evc}(G_v) - 1 = \text{evc}_{uw}(G_u) + \text{evc}(G_v) - 1 = k_u + k_v - 1 \) as well.

Now, by Proposition 2 it remains to consider the case when \( \text{evc}_{uw}(G_u) = k_u + 1 \) and \( \text{evc}(G_v) = k_v \). In this case, \( \text{evc}_{uw}(G_u) + \text{evc}(G_v) = k_u + k_v \). We will show that \( \text{evc}(G) = k_u + k_v \). From Lemma 5 Part 2, we get \( \text{evc}(G) \geq \max\{\text{evc}_{uw}(G_u) + \text{evc}(G_v), \text{evc}_{uw}(G_u) + \text{evc}(G_v)\} = k_u + k_v \). By Lemma 9 Part 1, we get \( \text{evc}(G) \geq \max\{\text{evc}_{uw}(G_u) + \text{evc}(G_v) - 1, \text{evc}_{uw}(G_u) + \text{evc}(G_v) - 1\} \). Hence, we are done. \( \square \)

**Lemma 13.** If \( \text{evc}(G_u) = k_u \) and \( \text{evc}(G_v) = k_v + 1 \), then

\[
\text{evc}(G) = \min\{\text{evc}(G) + 1, \text{evc}(G_u) + \text{evc}(G_v) - 1, \text{mvc}_{uw}(G_u) + \text{evc}(G_v) - 1\}.
\]

**Proof.** Since \( \text{evc}(G_v) = k_v + 1 \), by Lemma 9 Part 1, we get \( \text{evc}(G) \geq k_u + k_v \). By Observation 3 and Proposition 1, we get \( \text{evc}(G) \leq k_u + k_v + 1 \). By Proposition 2 it is easy to see that \( \text{evc}(G) \leq \min\{\text{evc}(G) + 1, \text{evc}(G_u) + \text{evc}(G_v) - 1, \text{mvc}_{uw}(G_u) + \text{evc}(G_v) - 1\} \). By Observation 4, we get \( \text{evc}(G) \leq \min\{\text{evc}(G) + 1, \text{evc}(G_u) + \text{evc}(G_v) - 1, \text{mvc}_{uw}(G_u) + \text{evc}(G_v) - 1\} \).

Hence, it suffices to show that when \( \text{evc}(G) = k_u + k_v \), then \( \text{evc}(G) \leq \min\{\text{evc}(G) + 1, \text{evc}(G_u) + \text{evc}(G_v) - 1, \text{mvc}_{uw}(G_u) + \text{evc}(G_v) - 1\} = k_u + k_v \). For this, suppose \( \text{evc}(G) = k_u + k_v \). By Lemma 9 Part 1, we have \( \text{evc}_{uw}(G_u) \leq k_u + 1 \) and \( \text{evc}(G_v) \leq k_v + 1 \). If \( \text{evc}_{uw}(G_u) = k_u \), we get \( \min\{\text{evc}(G_u), \text{evc}(G_v) - 1, \text{mvc}_{uw}(G_u) + \text{evc}(G_v) - 1\} \leq k_u + k_v = \text{evc}(G) \) as required. Therefore, let us assume that \( \text{evc}_{uw}(G_u) = k_u + 1 \). In this case, it is enough to show that \( \text{evc}_{uw}(G_v) = k_v + 1 \). By Proposition 1 there exists a vertex \( x \in V(G_v) \) such that \( \text{mvc}_{x}(G_v) = k_v + 1 \). Consider such
Therefore, it has to be the case that \( mvc \) and by Lemma 9 Part 2, we get \( evc \) then \( evc \) Hence, \( evc \) then \( evc \).

\[ mvc \leq \min \{ evc(G) + 1, max \{ evc(uw(G_u) + evc(G_v) - 1, evc(uv(G_v)) + evc(wv(G_v)) - 1 \} \}. \]

Hence, it is enough to show that if \( evc(G) = k_u + k_v \), then \( min \{ evc(G) + 1, max \{ evc(uw(G_u) + evc(G_v) - 1, evc(uv(G_v)) + evc(wv(G_v)) - 1 \} \} = k_u + k_v \).

For this, suppose \( evc(G) = k_u + k_v \). Since \( evc(G) = k_u + k_v \), \( evc(G) = k_u + 1 \) and \( evc(G) = k_v + 1 \), by Lemma 9 Part 1, we get \( evc(G_u) = k_u + 1 \) and \( evc(G_v) = k_v + 1 \). Now, it is enough to show that \( evc_u(G_u) = k_u \) and \( evc_v(G_v) = k_v \). Since \( evc(G_u) = k_u + 1 \), by Proposition 1 there exists a vertex \( x \in V(G_u) \) such that \( mvc(G_u) = k_u + 1 \). By Proposition 1 we know \( k_u + k_v = evc(G) \geq mvc(G) \). Therefore, it has to be the case that \( mvc(G_u) = k_u \). Symmetrically, we get \( mvc(G_v) = k_v \).

\[ evc(uw(G_u)) = evc(wv(G_v)) = 1. \]

\[ evc(G) = evc(G_u) + evc(G_v) = evc(uw(G_u)) + evc(wv(G_v)) = evc_u(G_u) + evc_v(G_v) = 1. \]

By Lemma 9 Part 1, we know \( evc(G) \geq max \{ evc_u(G_u) + evc_v(G_v) - 1, evc(G_u) + evc(G_v) - 1 \} \). Therefore, we get \( evc(G_u) = k_u \) and \( evc(G_v) = k_v \). By Proposition 1 we know \( mvc(G_u) = k_u \) and \( mvc(G_v) = k_v \). Hence, we get \( mvc(G) = k_u + k_v - 1 \).

First, we will prove the expression for \( evc(G) \). By Lemma 7 \( evc_u(G) \leq evc_u(G_u) + evc(G_v) - 1 = evc_u(G_u) + evc(G_v) - 1 \). By Lemma 9 Part 2, we get \( evc(G) \geq evc_u(G_u) + evc(G_v) - 1 \). Hence, \( evc_u(G) = evc_u(G_u) + evc(G_v) - 1 \). Similarly, we get \( evc_v(G) = evc_v(G_v) + evc(G_u) - 1 \). Now, we show that \( evc(G) = evc(G_u) + evc(G_v) - 1 \). By Lemma 9 we have \( evc(G) \leq evc(G_u) + evc(G_v) - 1 \). Now, it is enough to show that if \( evc(G) = evc_u(G_u) + evc_v(G_v) - 1 \), then \( evc(G) = evc(G_u) + evc(G_v) - 1 = mvc(G) \). Recall that \( mvc(G_u) = k_u + k_v - 1 \). Suppose \( evc_u(G_u) = evc(vw(G_v)) = k_u + k_v - 1 \). Then, by Lemma 9 Part 3, we get \( evc(G_u) = k_u \) and \( evc_v(G_v) = k_v \). Therefore, \( evc(G_u) + evc_v(G_v) - 1 = k_u + k_v - 1 \).
Now, we show that $\text{evc}_{uv}(G) = \min\{\text{evc}_{u}(G_u) + \text{mvc}(G_v), \text{mvc}(G_u) + \text{evc}_{v}(G_v), \text{mvc}(G_u) + \text{evc}_{uv}(G_v) + 1\}$. Since by Proposition 1 $\text{mvc}_{u}(G_u) = k_u$ and $\text{mvc}_{v}(G_v) = k_v$, by Lemma 8 Part 1, we have $\text{evc}_{uv}(G) \leq \min\{\text{evc}_{u}(G_u) + \text{mvc}(G_v), \text{mvc}(G_u) + \text{evc}_{v}(G_v), \text{mvc}(G_u) + \text{evc}_{uv}(G_v) + 1\}$. We know that $\text{evc}_{uv}(G) \in \{\text{mvc}_{uv}(G), \text{mvc}_{uv}(G) + 1\}$. So, it is now enough to show that when $\text{evc}_{uv}(G) = \text{mvc}_{uv}(G)$, then $\min\{\text{evc}_{u}(G_u) + \text{mvc}(G_v), \text{mvc}(G_u) + \text{evc}_{v}(G_v)\} = \text{evc}_{uv}(G)$.

Consider the case when $\text{evc}_{uv}(G) = \text{mvc}_{uv}(G)$. In this case, we will show that $\text{mvc}_{uv}(G) = k_u + k_v$.

Since we have already proved that $\text{evc}_{u}(G) = \text{evc}(G) = k_u + k_v$, by Proposition 1 we know that $\text{mvc}_{u}(G_u) = k_u$ and $\text{mvc}_{v}(G_v) = k_v$. Moreover, since $\text{evc}_{uv}(G) \geq \text{evc}(G) = k_u + k_v$, we get $\text{evc}_{uv}(G) = \text{mvc}_{uv}(G) = k_u + k_v$. It remains to show that $\min\{\text{evc}_{u}(G_u) + \text{mvc}(G_v), \text{mvc}(G_u) + \text{evc}_{v}(G_v)\} = k_u + k_v$. Since $\text{evc}_{uv}(G) = k_u + k_v$, it can be easily seen that either $\text{mvc}_{uv}(G_u) = k_u + 1$ or $\text{mvc}_{uv}(G_u) = k_u$. Without loss of generality, assume that $\text{mvc}_{uv}(G_u) = k_u + 1$. In this case, since $\text{evc}_{uv}(G) = k_u + k_v$, by Lemma 9 Part 3, we get $\text{evc}_{u}(G_v) = k_v$. Hence, we get $\min\{\text{evc}_{u}(G_u) + \text{mvc}(G_v), \text{mvc}(G_u) + \text{evc}_{v}(G_v)\} = k_u + k_v$.

**Lemma 17.** If $\text{evc}(G_u) = k_u$, $\text{evc}(G_v) = k_v + 1$ and $\text{mvc}(G) = k_u + k_v - 1$, then $\text{evc}(G) = \text{mvc}_{uv}(G_u) + \text{evc}(G_v) - 1$.



**Proof.** Since $\text{evc}(G_v) = k_v + 1$, by Lemma 13 it can be seen that $\text{evc}(G) = k_u + k_v$.

First, we will look at $\text{evc}(G)$. By Lemma 9 Part 2, we get $\text{evc}(G) \geq \text{mvc}_{uv}(G_u) + \text{evc}(G_v) - 1$. Now, it is enough to show that $\text{evc}(G) \leq \text{mvc}_{uv}(G_u) + \text{evc}(G_v) - 1$. We know that $\text{mvc}_{uv}(G_u) \in \{\text{mvc}(G_u), \text{mvc}(G_u) + 1\}$. First, suppose $\text{mvc}(G_u) = \text{evc}(G_u)$. In this case, it can be seen that $\text{mvc}_{uv}(G_u) = k_u$ and $\text{evc}(G_v) - 1 = k_u + k_v$. Moreover, by Proposition 1 we know $\text{evc}(G_u) \leq \text{mvc}_{uv}(G_u) + 1 = k_u + k_v$. Hence, we are done. Next, suppose $\text{mvc}(G_u) = \text{mvc}(G_u) + 1$. In this case, by Lemma 2 we get $\text{mvc}_{uv}(G_u) = k_u + 1$. Hence, we get $\text{mvc}_{uv}(G_u) + \text{evc}(G_v) - 1 = k_u + k_v + 1$.

By Lemma 7 we have $\text{evc}(G) \leq \text{evc}(G) + 1 = k_u + k_v + 1$. Hence, we are done.

Now, we will look at $\text{evc}(G)$. Let $Q = \min\{\text{mvc}_{uv}(G_u) + \text{evc}(G_v) - 1, \text{evc}(G_u) + \text{mvc}(G_v)\}$, the expression for $\text{evc}(G)$ to be proved. By Lemma 6 we have $\text{evc}(G) \leq Q$. Since $\text{evc}(G) \in \{\text{mvc}(G_u), \text{mvc}(G_u) + 1\}$, by Proposition 1 it is enough to show that if $\text{evc}(G) = \text{evc}(G) = \text{mvc}_{uv}(G_u)$, then $Q \leq \text{evc}(G)$. Assume $\text{evc}(G) = \text{mvc}_{uv}(G_u)$. Since $\text{evc}(G) \geq \text{evc}(G) = k_u + k_v$, we have $\text{evc}(G) = \text{evc}(G) = k_u + k_v$. By Lemma 7 Part 2, we get $\text{evc}(G_v) \leq k_v + 1$. If $\text{evc}(G_u) = k_u$, then we get $Q \leq k_u + k_v$ as required. If this was not the case, then there exists a vertex $x \in V(G_u)$ with $\text{mvc}_{uv}(G_u) = k_u + 1$. If $\text{mvc}_{uv}(G_u) = k_u$, then we get $Q \leq k_u + k_v$ immediately. Hence, we assume $\text{mvc}_{uv}(G_u) = k_u + 1$. Then, for the vertex $x \in V(G_v)$ for which $\text{mvc}_{uv}(G_u) = k_u + 1$, we get $\text{mvc}_{uv}(G_v) > k_u + 1$. Since $\text{evc}(G) \geq \text{mvc}(G_v)$ by Proposition 1 this is a contradiction. Thus, $Q \leq k_u + k_v$.

Now, we will consider $\text{evc}_{uv}(G)$. Let $P = \min\{\text{mvc}_{uw}(G_u) + \text{mvc}_{uw}(G_v) + \text{evc}_{uv}(G_v) - 1, \text{evc}(G_u) + \text{mvc}(G_v)\}$, the expression for $\text{evc}_{uv}(G)$ to be proved. By Lemma 8 we get $\text{evc}_{uv}(G) \leq P$. Now, we will show that $\text{evc}_{uv}(G) \geq P$. Since $\text{evc}_{uv}(G) \in \{\text{mvc}_{uw}(G_u), \text{mvc}_{uw}(G_v) + 1\}$ by Proposition 1 it is enough to show that if $\text{evc}_{uv}(G) = \text{mvc}_{uw}(G_u)$, then $P \leq \text{evc}_{uv}(G)$. Assume $\text{evc}_{uv}(G) = \text{mvc}_{uw}(G_u)$. Since $\text{evc}_{uv}(G) \geq \text{evc}(G) = k_u + k_v$, we have $\text{evc}_{uv}(G) = \text{evc}(G) = k_u + k_v$. By Lemma 9 Part 3, we get $\text{evc}_{uv}(G_v) = k_v + 1$. If $\text{evc}_{uv}(G_u) = k_u$, then it can be seen that $P \leq k_u + k_v$. Hence, assume $\text{evc}_{uv}(G_u) = k_u + 1$. Then, by Lemma 9 Part 3, we get $\text{evc}_{uv}(G_v) = k_v$. Further, by Lemma 9 Part 3, it can be seen that $\text{mvc}_{uw}(G_u) = k_u$. This
implies $\text{mvc}_{uv}(G_v) = k_v + 1$ by Lemma\ref{lem:mvcuv}. Hence, by Lemma\ref{lem:evcvk} we get $\text{evc}_u(G_u) = k_u$. Therefore, we get $P \leq k_u + k_v = \text{evc}_{uv}(G)$.

**Lemma 18.** If $\text{evc}(G_u) = k_u$, $\text{evc}(G_v) = k_v + 1$ and $\text{mvc}(G) = \text{evc}(G) = k_u + k_v$, then:

$$
\text{evc}_u(G) = \text{evc}_u(G_u) + \text{evc}_v(G_v) - 1,
$$

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

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

**Proof.** First, we will show that $\text{evc}_u(G) = \text{evc}_u(G_u) + \text{evc}_v(G_v) - 1$. Since $\text{evc}(G) = k_u + k_v$, by Lemma\ref{lem:evc} Part 1, $\text{evc}(G_v) \leq k_v + 1$, which implies $\text{evc}_v(G_v) = k_v + 1$. Further, by Proposition\ref{prop:evcvk} $\text{evc}_u(G) \in \{k_u + k_v, k_u + k_v + 1\}$. Now, it is enough to show that if $\text{evc}_u(G) = \text{evc}(G) = k_u + k_v$, then $\text{evc}_u(G_u) = k_u$ and if $\text{evc}_u(G) = \text{evc}(G) + 1 = k_u + k_v + 1$, then $\text{evc}_u(G_u) = k_u + 1$. First, suppose $\text{evc}_u(G) = k_u + k_v$. By Lemma\ref{lem:evc} Part 2, we get $\text{mvc}_{uv}(G_u) = k_u$ and by Lemma\ref{lem:evc} we get $\text{mvc}_{uv}(G_v) \neq k_v$. Hence, we have $\text{mvc}_v(G_v) = k_v$. For contradiction, suppose $\text{evc}_u(G_u) = k_u + 1$. Then, by Proposition\ref{prop:evc} there exists a vertex $x \in V(G_u)$, such that $\text{mvc}_{xv}(G_u) = k_u + 1$. Since $\text{mvc}_{uv}(G_u) = k_u + 1$, this implies $\text{mvc}_{uv}(G_u) \geq k_u + k_v + 1$. By Proposition\ref{prop:evc} this contradicts the value of $\text{evc}_u(G)$. Hence, we get $\text{evc}_u(G_u) = k_u$ as required. Next, suppose $\text{evc}_u(G) = k_u + k_v + 1$. For contradiction, suppose $\text{evc}_u(G_u) = k_u$. Then, by Proposition\ref{prop:evc} we get $\text{mvc}_{uv}(G_u) = k_u$. By Lemma\ref{lem:evc} we have $\text{evc}_u(G) \leq \max\{\text{evc}_u(G_u) + \text{mvc}_v(G_v), \text{mvc}_{uv}(G_u) + \text{evc}(G_v) - 1\} = k_u + k_v$, which is a contradiction. Hence, $\text{evc}_u(G_u) = k_u + 1$ and we are done.

The proof of the expression for $\text{evc}_v(G)$ is exactly the same as that in the proof of Lemma\ref{lem:evc}.

Now we will look at $\text{evc}_{uv}(G)$. Let $P = \max\{\text{evc}_u(G_u) + \text{mvc}_v(G_v), \text{evc}_u(G_u) + \text{evc}_{uv}(G_v)\}$. We need to show that $\text{evc}_{uv}(G) = \min\{\text{evc}(G) + 1, P\}$. By Lemma\ref{lem:evc} we get $\text{evc}_{uv}(G) \leq \min\{\text{evc}(G) + 1, P\}$. Now, we will show that $\text{evc}_{uv}(G) \geq \min\{\text{evc}(G) + 1, P\}$. If $\text{evc}_{uv}(G) = \text{evc}(G) + 1$, we are done. Therefore, consider the case when $\text{evc}_{uv}(G) = \text{evc}(G) = k_u + k_v$. In this case, we show that $P \leq k_u + k_v$. Since $\text{mvc}_v(G_v) = k_u + k_v$ and $\text{mvc}_{uv}(G_u) = k_v$ by Proposition\ref{prop:evc} we get $\text{mvc}_{uv}(G_v) = k_v + 1$ by Lemma\ref{lem:evc}. Hence, by Lemma\ref{lem:evc} Part 3, we get $\text{evc}_u(G_u) = k_u$. Moreover, we also get $\text{evc}_{uv}(G_v) = k_v + 1$ and $\text{mvc}_{uv}(G_u) = k_v$ by Lemma\ref{lem:evc} Part 3. Therefore, since $\text{mvc}(G) = k_u + k_v$, we know $\text{mvc}_{uv}(G_v) \neq k_v$. Thus, we get $P \leq k_u + k_v$.

**Lemma 19.** If $\text{evc}(G_u) = k_u$, $\text{evc}(G_v) = k_v + 1$ and $\text{evc}(G) = k_u + k_v + 1$, then

$$
\text{evc}_u(G) = \text{evc}_v(G) = \text{evc}(G) = k_u + k_v + 1,
$$

$$
\text{evc}_v(G) = \text{evc}(G) = k_u + k_v + 1,
$$

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

**Proof.** First, we will show that $\text{evc}_u(G) = \text{evc}(G)$. Since we know that $\text{evc}_u(G) \geq \text{evc}(G)$ by Lemma\ref{lem:evc} Part 2, it is enough to show that $\text{evc}_u(G) \leq \text{evc}(G)$. For any vertex $x \in V(G_u)$, we have $\text{mvc}_{xv}(G_u) = k_u$ by Proposition\ref{prop:evcvk} and hence $\text{mvc}_{xv}(G_u) \leq k_u + 1$. This implies $\text{mvc}_{uv}(G_u) \leq k_u + k_v = \text{evc}(G)$. Similarly, for any vertex $y \in V(G_v)$, we have $\text{mvc}_{yu}(G_v) \leq k_v + 1$. Hence, $\text{mvc}_{yu}(G_v) \leq k_u + k_v + 1$. Therefore, by Proposition\ref{prop:evcvk} $\text{evc}_{uv}(G) \leq k_u + k_v + 1$. Hence, $\text{evc}_u(G) = \text{evc}(G)$.

Next, we will show that $\text{evc}_v(G) = \text{evc}(G)$. Since we know that $\text{evc}_v(G) \geq \text{evc}(G)$ by Lemma\ref{lem:evc} Part 2, it is enough to show that $\text{evc}_v(G) \leq \text{evc}(G)$. For any vertex $x \in V(G_u)$, we have $\text{mvc}_{xv}(G_u) = k_v$ by Proposition\ref{prop:evcvk} and hence $\text{mvc}_{xv}(G_u) \leq k_u + k_v + 1 = \text{evc}(G)$. Similarly, for any vertex $y \in V(G_v)$, $\text{mvc}_{yu}(G_v) \leq k_v + 2$. Since $\text{mvc}_{uv}(G_u) = k_u$, this gives $\text{mvc}_{yu}(G_v) \leq k_u + k_v + 1$. Therefore, by Proposition\ref{prop:evcvk} $\text{evc}_v(G) \leq k_u + k_v + 1$. Hence, $\text{evc}_v(G) = \text{evc}(G)$.

Now, we will show that $\text{evc}_{uv}(G) = \min\{\text{evc}(G) + 1, \text{evc}(G_u) + \text{evc}_v(G_v)\}$. By Lemma\ref{lem:evc} we get $\text{evc}_{uv}(G) \leq \min\{\text{evc}(G) + 1, \text{evc}(G_u) + \text{evc}_v(G_v)\}$. Next, we show that $\text{evc}_{uv}(G) \geq \min\{\text{evc}(G) + 1, \text{evc}(G_u) + \text{evc}_v(G_v)\}$. If $\text{mvc}_{uv}(G_u) = k_u + k_v$ or $\text{evc}_v(G_v) = k_v + 1$, this is trivial. So, we are left with the case when $\text{mvc}_{uv}(G_u) = k_u + k_v + 1$ and $\text{evc}_v(G_v) = k_v + 2$. For
Lemma 21. It can be verified that \( mvc(x) \) any vertex \( mvc(x) \) pose evc to
Hence, by Proposition 1, we get evc. Since evc has a contradiction, suppose evc was not true. By Lemma 9 Part 1, we get evc(P) = evc(G) + evc(G) - 1. Therefore it is enough to show that evc(G) = evc(G) + evc(G) - 1. Hence, by Proposition 1 we get evc(G) ≤ evc(G) + evc(G) - 1.

Next, we need to show that evc(G) = mvc(G) + evc(G) - 1. By Proposition 1, we get evc(G) = mvc(G) + evc(G) - 1. We have evc(G) = evc(G) = evc(G) + evc(G) - 1. Suppose evc(G) = mvc(G). Since mvc(G) ≤ evc(G) ≤ evc(G), we have mvc(G) = evc(G). By Observation 3 and Lemma 14 we have evc(G) = \( k_u + k_v + 1 \). First, suppose evc(G) = mvc(G). Then, evc(G) ≥ evc(G) + evc(G) - 1.

Next, suppose evc(G) = mvc(G) + evc(G) - 1. In this case, it is enough to show that evc(G) = mvc(G) + evc(G) - 1. Therefore, evc(G) = mvc(G) + evc(G) - 1. If mvc(G) = evc(G) = evc(G) + evc(G) - 1, it can be verified that evc(G) = mvc(G) + evc(G) - 1. Hence, by Proposition 1 we get evc(G) = mvc(G) + evc(G) - 1.

Lemma 20. If evc(G) = k_u + 1 and evc(G) = k_v + 1, then:
1. If mvc(G) ≤ k_u + k_v then evc(G) = mvc(G) + 1.
2. Otherwise, evc(G) = \( min\{mvc(G) + 1, max(P_1, P_2)\} \), where
   \( P_1 = min\{evc(G) + mvc(G) - 1, evc(G) + evc(G)\} \) and \( P_2 = min\{mvc(G) + evc(G) - 1, mvc(G) + evc(G)\} \).

Proof. 1. Suppose mvc(G) ≤ k_u + k_v. By Proposition 1 evc(G) ∈ \{mvc(G), mvc(G) + 1\}. Therefore it is enough to show that evc(G) ≠ mvc(G). For contradiction, suppose this was not true. By Lemma 9 Part 1, we get evc(G) ≥ k_u + k_v. Therefore, evc(G) = mvc(G) + evc(G) - 1. Hence, we may assume that mvc(G) = evc(G) = k_u + k_v. By Lemma 9 Part 3, we get mvc(G) = k_u and mvc(G) = k_v. This gives evc(G) = k_u + k_v - 1, a contradiction. Hence, evc(G) = mvc(G) + 1.

2. Suppose mvc(G) = k_u + k_v + 1. By Lemma 8 we know evc(G) ≤ mvc(G) + 1, max(P_1, P_2). Now, we will show that evc(G) ≥ mvc(G) + 1, max(P_1, P_2). Recall that we are in a case when mvc(G) = k_u + k_v + 1 and by Proposition 1 evc(G) ∈ \{mvc(G), mvc(G) + 1\}. If evc(G) = mvc(G) + 1, we are done. Therefore we are left with the case where evc(G) = mvc(G) = k_u + k_v + 1. In this case mvc(G) = k_u + 1 and
Algorithm 3 in the appendix. The details of the data structure is given in Fig. 2 in the appendix. An example for Vertex Edge Face Adjacency List is given in Appendix. Vertex Edge Face Adjacency List of an outerplanar graph can be produced by modifying the algorithm suggested by N. Chiba and T. Nishizeki [15] for listing all the triangles of a planar graph in linear time. An algorithm for producing the Vertex Edge Face Adjacency List is given in Algorithm 3 in the appendix.

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 \(\mathcal{M}(G, uv)\) and \(\mathcal{E}(G, uv)\). The case when \(G\) is a triangle forms the base case of the recursion and it is handled using Observation 1. Let \(w = \Delta(uv)\). If \(\text{deg}_G(u) > 2\) and \(\text{deg}_G(v) > 2\), 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 \((G_{u'}, v'w)\) 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. For this refinement, we maintain a Vertex Edge Face Adjacency List data structure using which the following are possible:

1. For any edge \(e\), we can traverse through the internal faces on which the edge lies in \(O(1)\) time.
2. For any internal face \(f\), we can traverse through the edges on the boundary of \(f\) in \(O(1)\) time.

The details of the data structure is given in Fig. 2 in the appendix. An example for Vertex Edge Face Adjacency List of a maximal outerplanar graph on four vertices is given in Fig. 3 in the appendix. Vertex Edge Face Adjacency List of an outerplanar graph can be produced by modifying the algorithm suggested by N. Chiba and T. Nishizeki [15] for listing all the triangles of a planar graph in linear time. An algorithm for producing the Vertex Edge Face Adjacency List is given in Algorithm 3 in the appendix.

By using the visit field of faces in the Vertex Edge Face Adjacency List data structure, we avoid passing the subgraph as an explicit parameter in recursive calls, maintaining the following invariants:

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

Further, line numbers 2, 3, 5 and 6 of Algorithm 1 can also be implemented in linear time using the new data structure.

Algorithm 1  To compute $M(G, uv)$ and $E(G, uv)$ of a maximal outerplanar graph $G$ with $uv$ an edge on its outer face.

**Inputs:** A maximal outerplanar graph $G$ with at least three vertices, an edge $uv$ on the outer face of $G$.

**Outputs:** $M(G, uv)$ and $E(G, uv)$.

1: **procedure** EVC$\_\$PARAMETERS($G$, $uv$)
2: $w \leftarrow \Delta(uv)$. 
3: if $G$ is a triangle then
4:  $\text{mvc}({G}) = \text{evc}({G}) = \text{mvc}_u({G}) = \text{mvc}_v({G}) = 2$, $\text{evc}_w({G}) = 2$, $\text{evc}_{uv}({G}) = 3$.
5: else if One end point of $uv$ is degree-2 then
6:  $u' =$ degree-2 end point of $uv$, $v' =$ higher degree end point of $uv$.
7:  Recursively call EVC$\_\$PARAMETERS($G_{v'}$, $v'w$).
8:  Compute $\text{mvc}({G})$ using Theorem 1 (Part 1).
9:  Compute $\text{mvc}_u({G})$, $\text{mvc}_v({G})$ and $\text{mvc}_{uv}({G})$ using Theorem 1 (Part 2).
10: Compute $\text{evc}({G})$ using Theorem 3 (Part 1).
11: Compute $\text{evc}_{uv}({G})$ in constant using Theorem 3 (Part 2).
12: else
13:  $G_u \leftarrow uv$-segment of $G$ containing the vertex $u$
14:  $G_v \leftarrow uv$-segment of $G$ containing the vertex $v$
15:  Recursively call EVC$\_\$PARAMETERS($G_u$, $uw$).
16:  Recursively call EVC$\_\$PARAMETERS($G_v$, $vw$).
17:  Compute $\text{mvc}({G})$ using Theorem 2 (Part 1).
18:  Compute $\text{mvc}_u({G})$, $\text{mvc}_v({G})$, $\text{mvc}_{uv}({G})$ using Theorem 2 (Part 2).
19:  Compute $\text{evc}({G})$ using Theorem 4 (Part 1).
20:  Compute $\text{evc}_u({G})$, $\text{evc}_v({G})$, $\text{evc}_{uv}({G})$ using Theorem 4 (Part 2).
21: end if
22: **return** ($M(G, uv)$ and $E(G, uv)$)
23: end procedure

Algorithm 2 (in the appendix) gives a linear time refinement of Algorithm 1. The details of the correspondence between the two algorithms are given in the appendix.

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 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 the 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 [2]. However, the complexity status of the problem...
of computing the evc number of outerplanar graphs is open and may be attempted using the techniques developed in this work.

References

1. Fomin, F.V., Gaspers, S., Golovach, P.A., Kratsch, D., Saurabh, S.: Parameterized algorithm for eternal vertex cover. Information Processing Letters 110(16) (2010) 702 – 706
2. Babu, J., Chandran, L.S., Francis, M., Prabhakaran, V., Rajendraprasad, D., Warrier, N.J.: On graphs whose eternal vertex cover number and vertex cover number coincide. Discrete Applied Mathematics (2021)
3. Klostermeyer, W., Mynhardt, C.: Edge protection in graphs. Australasian Journal of Combinatorics 45 (2009) 235 – 250
4. Araki, H., Fujito, T., Inoue, S.: On the eternal vertex cover numbers of generalized trees. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E98.A (06 2015) 1153–1160
5. Babu, J., Prabhakaran, V., Sharma, A.: A substructure based lower bound for eternal vertex cover number. Theoretical Computer Science (2021)
6. Babu, J., Prabhakaran, V.: A new lower bound for the eternal vertex cover number of graphs. Journal of Combinatorial Optimization (2021) 1–17
7. Rinemberg, M., Soulignac, F.J.: The eternal dominating set problem for interval graphs. Information Processing Letters 146 (2019) 27–29
8. Goldwasser, J.L., Klostermeyer, W.F.: Tight bounds for eternal dominating sets in graphs. Discrete Mathematics 308(12) (2008) 2589–2593
9. Klostermeyer, W.F., Mynhardt, C.: Vertex covers and eternal dominating sets. Discrete Applied Mathematics 160(7) (2012) 1183–1190
10. Goddard, W., Hedetniemi, S.M., Hedetniemi, S.T.: Eternal security in graphs. J. Combin. Math. Combin. Comput 52 (2005) 169–180
11. Klostermeyer, W.F., Mynhardt, C.M.: Graphs with equal eternal vertex cover and eternal domination numbers. Discrete Mathematics 311 (2011) 1371 – 1379
12. Anderson, M., Carrington, J.R., Brigham, R.C., D.Dutton, R., Vitray, R.P.: Graphs simultaneously achieving three vertex cover numbers. Journal of Combinatorial Mathematics and Combinatorial Computing 91 (2014) 275 – 290
13. Hartnell, B., Mynhardt, C.: Independent protection in graphs. Discrete Mathematics 335 (2014) 100 – 109
14. Caro, Y., Klostermeyer, W.: Eternal independent sets in graphs. Volume 3. (2016)
15. Chiba, N., Nishiziki, T.: Arboricity and subgraph listing algorithms. SIAM Journal on computing 14(1) (1985) 210–223
8 Appendix - Refinement of Algorithm 1 to Work in Linear Time

8.1 Vertex Edge Face Adjacency List

To make Algorithm 1 work in linear time, we should avoid the explicit passing of the subgraphs $G_u$ and $G_v$ as parameters for the recursive calls. Instead, we do the following.

Using an algorithm suggested by N. Chiba and T. Nishizeki [15], in linear time all the triangles of a planar graph can be listed/printed. Let $G$ be a maximal outerplanar graph with a fixed outerplanar embedding. When $G$ is given as input to the algorithm of Chiba and Nishizeki, it will precisely list all the internal faces of the graph. By modifying the algorithm proposed in [15], we produce a global data structure named **Vertex Edge Face Adjacency List**. The details of the data structure is given in Fig. 2. The algorithm for producing this data structure is given in Algorithm 3 and will be discussed later.

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

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

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

Fig. 2. Details of the Vertex Edge Face Adjacency List data structure.

In a Vertex Edge Face Adjacency List data structure, for any edge $v_i v_j$, there are links between the edge node $v_i v_j$ and the edge node $v_j v_i$. For each face node representing a face $f$ in $G$, there are links to the edge nodes corresponding to the bounding edges of $f$ and vice versa. Remember that each edge of $G$ has 2 edge nodes corresponding to it. Hence, for each face node $f$, there are 6 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.

The Vertex Edge Face Adjacency List of a maximal outerplanar graph on four vertices is given in Fig. 3.

8.2 Refined algorithm and its correspondence with Algorithm 1

We will first assume that Vertex Edge Face Adjacency List of the graph has been already produced and give the remaining details of the linear time implementation. Algorithm 2 gives a linear time version 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 to the algorithm. Initially, the visit field of all faces are set to unvisited. Note that, the edge $uv$ is present in exactly one unvisited face $uvw$ initially. On entering the algorithm, the face $uvw$ is marked as visited. Once this marking is done, the edges $uw$
and \( vw \) can be in at most one unvisited face each. Note that if \( uw \) is not in any unvisited face, then \( \deg_G(u) = 2 \). Otherwise \( \deg_G(u) > 2 \). Similarly we can check if \( \deg_G(v) = 2 \) or not from the number of unvisited faces containing the edge \( vw \).

**Algorithm 2** For computing \( \mathcal{M}(G,uv) \) and \( \mathcal{E}(G,uv) \) 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 \( G \).

**Output:** \( \mathcal{M}(G,uv) \) and \( \mathcal{E}(G,uv) \).

**Assumption:** The Vertex Edge Face Adjacency List is maintained globally.

1: procedure EVC
2: Identify the unvisited face \( uvw \) in which the edge \( uv \) lie.
3: \( \triangledown \) 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: \( \triangledown \) Possible to check in constant time using Vertex Edge Face Adjacency List
7: \( mvc(G)=evc(G)=mvc_u(G)=mvc_v(G)=2, evc_u(G)=evc_v(G)=2, evc_{uv}(G)=3. \)
8: else if exactly one among \( uw \) and \( vw \) lie in an unvisited face then
9: \( \triangledown \) 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: \( t \leftarrow \) EVC\_Parameters\((e)\).
12: Compute \( mvc(G) \) in constant time using Theorem 1 (part 1).
13: Compute \( mvc_u(G) \), \( mvc_v(G) \) and \( mvc_{uv}(G) \) in constant time using Theorem 1 (part 2).
14: Compute \( evc(G) \) in constant time using Theorem 3 (part 1).
15: Compute \( evc_{uv}(G) \) in constant using Theorem 3 (part 2).
16: else
17: \( t_1 \leftarrow \) EVC\_Parameters\((uw)\).
18: \( t_2 \leftarrow \) EVC\_Parameters\((vw)\).
19: Compute \( mvc(G) \) in constant time using Theorem 2 (part 1).
20: Compute \( mvc_u(G) \), \( mvc_v(G) \) in constant time using Theorem 2 (part 2).
21: Compute \( evc(G) \) in constant time using Theorem 4 (part 1).
22: Compute \( evc_u(G) \), \( evc_v(G) \), \( evc_{uv}(G) \) in constant time using Theorem 4 (part 2).
23: end if
24: return \( (\mathcal{M}(G,uv) \) and \( \mathcal{E}(G,uv)) \)
25: end procedure

In subsequent recursive calls, the following invariants will be maintained:

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

By maintaining the visit field of faces, we avoid passing the subgraph as an explicit parameter in recursive calls in Algorithm 2.

From these observations, we show that Algorithm 2 is a linear time implementation of Algorithm 1. The inputs of the function EVC\_Parameters of Algorithm 1 are a maximal outerplanar graph \( G \) and an edge \( uv \) on the outer face. The algorithm 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 input to the function EVC_Parameters of Algorithm 2 is an edge \(uv\) on the outer face of the input graph. For the input edge \(uv\), let \(uvw\) be the unvisited face in which \(u\) lies. 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

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.

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 \(uw\)-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 \(uw\)-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 unvisited face, then we can infer that the degree of the vertex \(u\) (respectively, \(v\)) is greater than 2 in the \(uw\)-segment of \(G\) that contains the face \(uvw\). Otherwise, the degree of \(u\) (respectively, \(v\)) in the \(uw\)-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 \(uw\) 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 \(uw\), 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 \( uv \) 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 linear time implementation of Algorithm 1.

8.3 Building the Vertex Edge Face Adjacency List

We can use the algorithm given below to populate the Vertex Edge Face Adjacency List of a maximal outerplanar graph.

**Algorithm 3**

*For populating the Vertex Edge Face Adjacency List*

1: **procedure** **Populate Vertex Edge Face Adjacency List**
2: **Inputs**: Bare vertex edge face adjacency list \( L \) of a maximal outerplanar graph \( G \), in which face nodes are absent and links to face nodes from edge nodes are initialized to NULL.
3: **Outputs**: Modified vertex edge face adjacency list \( L \) with all relevant details filled in.
4: Create a copy \( L' \) of \( L \).
5: For any edge node \( v_i, v_j \) in \( L' \), keep a pointer to and from the corresponding edge node in \( L \).
6: Sort the vertices of \( G \) in decreasing order of vertex degree. (using bucket sort in linear time)
7: for \( i=1 \) to \( n \) do
8: Let \( v_i = \text{next highest degree vertex} \).
9: \( p = s = Varray[v_i].\text{head} \) in \( L' \).  
10: while \( (s \neq \text{NULL}) \) do \( \triangleright \text{Parse the edge list of } v_i \).
11: \( x = s.\text{vertex} \_2 \)
12: \( \text{Varray}[x].\text{mark} = \text{Marked} \)
13: \( \text{Varray}[x].\text{cur}-\text{edge} = s \)
14: \( s = s.\text{next} \)
15: end while
16: while \( (p \neq \text{NULL}) \) do
17: \( v_j = p.\text{vertex} \_2 \)
18: \( q = \text{Varray}[v_j].\text{head} \) \( \triangleright \text{Beginning of adjacency list of } v_j \)
19: while \( (q \neq \text{NULL}) \) do
20: \( v_k = q.\text{vertex} \_2 \)
21: if \( (\text{Varray}[v_k].\text{status} = \text{Marked}) \) then
22: \( r = \text{Varray}[v_k].\text{cur}-\text{edge} \)
23: \( p' = \text{Copy}(p) \) in \( L \), \( q' = \text{Copy}(q) \) in \( L \) and \( r' = \text{Copy}(r) \) in \( L \).
24: In \( L \), create face node \( f \) with links to edge nodes \( p', q' \) and \( r' \) and reverse.
25: Simultaneously insert links from \( f \) to pair nodes of \( p', q' \) and \( r' \) and reverse.
26: end if
27: \( q = q.\text{next} \);
28: end while
29: \( v_j.\text{mark} = \text{Unmarked} \)
30: \( p = p.\text{next} \);
31: end while
32: In \( L' \), \( \text{Varray}[v_i].\text{mark} = \text{Deleted} \) and delete the edge list of \( v_i \).
33: Whenever edge \( v_i, v_k \) is deleted, delete \( v_k, v_i \) also.
34: end for
35: end procedure
We assume a bare Vertex Edge Face Adjacency List $L$ as the input graph representation in which the face nodes and the links to and from face nodes are absent. The Algorithm 2 begins by creating a copy $L'$ of the bare Vertex Edge Face Adjacency List $L$ and links from an edge node in $L'$ to its copy in $L$. (For simplicity, in the data structure given in Fig. 2, we have avoided a field for this link.) During the running of Algorithm 2, we traverse through $L'$ and each time a new face node is created corresponding to a face $f_i$ in $G$, we assign links from the newly created face node to the edge nodes (bounding edges of $f_i$) in $L$ and also the reverse links. This step can be done in constant time by utilizing the links from each edge nodes of $L'$ to its copy in $L$. During the running of Algorithm 2, some edge nodes of $L'$ gets deleted while their copies in $L$ are retained. Note that, this way of deletion of edge nodes of $L'$ is necessary to obtain linear time complexity, as it was done in [15]. Hence, the linear time complexity of the construction follows from the analysis in [15].

![Fig. 3. The Vertex Edge Face Adjacency List of a maximal outerplanar graph $G$ on four vertices and two faces. $F_1$ and $F_2$ denote the face nodes.](image-url)