Parameterized lower bound and improved kernel for Diamond-free Edge Deletion

R.B. Sandeep¹, Naveen Sivadasan

Department of Computer Science & Engineering
Indian Institute of Technology Hyderabad, INDIA
{cs12p0001, nsivadasan}@iith.ac.in

Abstract

A diamond is a graph obtained by removing an edge from a complete graph on four vertices. A graph is diamond-free if it does not contain an induced diamond. The Diamond-free Edge Deletion problem asks to find whether there exist at most k edges in the input graph whose deletion results in a diamond-free graph. The problem is known to be fixed-parameter tractable and a polynomial kernel of $O(k^3)$ vertices is found in [4].

In this paper, we give an improved kernel of $O(k^3)$ vertices for Diamond-free Edge Deletion. We complement the result by proving that the problem is NP-complete and cannot be solved in time $2^{o(k)} \cdot n^{O(1)}$, unless Exponential Time Hypothesis fails.

Keywords:
polynomial kernel, edge deletion, parameterized lowerbound, diamond

1. Introduction

For a graph property $Π$, the $Π$ Edge Deletion problem asks whether there exist at most $k$ edges whose deletion from the input graph results in a graph with property $Π$. Edge deletion problems have been studied for the last four decades on various fronts: hardness, polynomial time algorithms, approximability, fixed parameter tractability, polynomial kernelization and incompressibility. The problem has found applications in physical mapping of DNA [10]. Unlike the corresponding vertex deletion problems, edge deletion problems did not yield any general hardness result. It has been proved in [2] that $Π$ Edge Deletion problem is fixed parameter tractable if $Π$ can be characterized by a finite set of forbidden induced subgraphs. Polynomial kernelization and incompressibility of edge deletion problems were subjected to rigorous studies in the recent past. It has been proved that there exist no polynomial kernel for $Π$ Edge Deletion where $Π$ is the property ‘$H$-free’ where $H$ is any 3-connected graph other than a

¹Supported by TCS Research Scholarship
complete graph \( H \), unless \( \text{coNP} \subseteq \text{NP/poly} \). In the same paper, under the same assumption, it is proved that, if \( H \) is a path or a cycle, then \( H\text{-free Edge Deletion} \) is incompressible if and only if \( H \) has at least four vertices. It is proved in \cite{1} that \( H\text{-free Edge Deletion} \) admits polynomial kernelization on bounded degree graphs if \( H \) is a finite set of connected forbidden induced subgraphs. Though, polynomial kernelization results have been found for many \( \Pi \text{ Edge Deletion} \) problems, \( \text{Claw-free Edge Deletion} \) withstood the test of time and yielded neither an incompressibility result nor a polynomial kernel. Some progress has been made recently for this problem such as a polynomial kernel for \( \text{Claw-free Edge Deletion} \) on \( K_t \)-free input graphs \cite{1} and a polynomial kernel for \( \{ \text{Claw, Diamond}\}\text{-free Edge Deletion} \) \cite{5}.

In this paper, we study the polynomial kernelization and parameterized lower bound of \( \text{Diamond-free Edge Deletion} \). It is proved in \cite{4} that \( \text{Diamond-free Edge Deletion} \) admits a kernel of \( O(k^5) \) vertices. We improve this result by giving a kernel of \( O(k^3) \) vertices. Our analysis of the bound on the kernel size is simpler compared to that in \cite{4}. In \cite{4}, the author used a propagation graph which has the information on how new diamonds are generated while deleting edges. Though the concept of propagation graph may be useful in similar contexts, it complicates the analysis of the kernelization. Instead, we use vertex modulator technique, which was used recently in \cite{6} to give a polynomial kernel for \( \text{Trivially Perfect Editing} \) and in \cite{5} to obtain a polynomial kernel for \( \{ \text{Claw, Diamond}\}\text{-free Edge Deletion} \). We introduce a novel rule named as Vertex-Split which splits a vertex into a set of independent vertices where each vertex in the set corresponds to a component in the neighborhood of the vertex. We believe that this rule may have further applications in similar settings.

To complement our kernelization result, we prove that \( \text{Diamond-free Edge Deletion} \) is \( \text{NP-complete} \). Though, there is a hardness result in \cite{7}, for \( \Pi \text{ Edge Deletion} \) problem where \( \Pi \) is defined by diamond as a forbidden homeomorph or minor, to the best of our knowledge, there is no hardness result for \( \text{Diamond-free Edge Deletion} \) in the literature. Our reduction is from the \( \text{Vertex Cover} \) on cubic graphs and is a linear parameterized reduction. This enables us to prove that, unless Exponential Time Hypothesis (ETH) fails, there exists no parameterized subexponential time algorithm (an algorithm which runs in time \( 2^{o(k)} \cdot n^{O(1)} \)) for \( \text{Diamond-free Edge Deletion} \).

### 1.1. Preliminaries

The problem we consider in this paper is \( \text{Diamond-free Edge Deletion} \): whether there exist at most \( k \) edges whose deletion from the input graph results in a graph without any induced diamond. In the parameterized version, the parameter is \( k \).

**Graphs:** Every graph considered here is simple, finite and undirected. For a graph \( G \), \( V(G) \) and \( E(G) \) denote the vertex set and the edge set of \( G \) respectively. \( N_G(v) \) denotes the (open) neighborhood of a vertex \( v \in V(G) \), which is the set of vertices adjacent to \( v \) in \( G \). The closed neighborhood of \( v \) is denoted by \( N_G[v] \) and is defined by \( N_G(v) \cup \{v\} \). We remove the subscript when there
is no ambiguity about the underlying graph $G$. A graph $G' : (V', E')$ is called an induced subgraph of a graph $G$ if $V' \subseteq V(G)$, $E' \subseteq E(G)$ and an edge $\{x, y\} \in E(G)$ is in $E'$ if and only if $\{x, y\} \subseteq V'$. For a vertex set $V' \subseteq V(G)$, $G[V']$ denotes the induced subgraph with a vertex set $V'$ of $G$. A component $G'$ of a graph $G$ is a connected induced subgraph of $G$ such that there is no edge between $V(G')$ and $V(G) \setminus V(G')$. For a set of vertices $V' \subseteq V(G)$, $G - V'$ denotes the graph obtained by removing the vertices in $V'$ and all its incident edges from $G$. For an edge set $E' \subseteq E(G)$, $G - E'$ denotes the graph obtained by deleting all edges in $E'$ from $G$. If $V' \ (E')$ is a singleton set $\{v\} \ (\{e\})$, we denote the graph $G - V' \ (G - E')$ by $G - v \ (G - e)$. For an edge set $E' \subseteq E(G)$, $V_{E'}(G)$ denotes the vertices in $G$ incident to the edges in $E'$. A matching (non-matching) is a set of edges (non-edges) such that every vertex in the graph is incident to at most one edge (non-edge) in the matching (non-matching). Diamond is a graph obtained by deleting an edge from a complete graph on four vertices. A graph $G$ is called diamond-free, if $G$ does not contain any diamond as an induced subgraph. Whenever we mention that $\{a, b, c, d\} \subseteq V(G)$ induces a diamond in $G$, $a$ and $b$ are degree-3 vertices and $c$ and $d$ are degree-2 vertices. In a diamond, we call the edge between the degree-3 vertices as middle edge.

**Parameterized complexity:** A parameterized problem is fixed parameter tractable, if there is an algorithm to solve it in time $f(k) \cdot n^{O(1)}$, where $f$ is any computable function and $n$ is the size of the input. Polynomial kernelization is an algorithm which takes as input $(G, k)$, an instance of a parameterized problem, runs in time $(|G| + k)^{O(1)}$ and returns an instance $(G', k')$ of the same problem such that $|G'|, k' \leq p(k)$, where $p$ is any polynomial function. A rule for kernelization is safe if $(G, k)$ is a yes-instance if and only if $(G', k')$ is a yes-instance where $(G, k)$ and $(G', k')$ are the input and output of the kernelization. Linear parameterized reduction from a parameterized problem $A$ to another $B$ is a polynomial time reduction such that $k' = O(k)$ where $k$ and $k'$ are the parameters of the instances of $A$ and $B$ respectively. A subexponential time algorithm for a parameterized problem is an algorithm which runs in time $2^{o(k)} \cdot n^{O(1)}$ where $n$ is the size of the problem instance.

2. Polynomial Kernel

In this section, we give a kernel with $O(k^3)$ vertices for DIAMOND-FREE EDGE DELETION. To start with, we introduce two properties of graphs and two rule based on those properties.

**Property 1** (Core-Member). A vertex or an edge of a graph $G$ is a Core-Member of $G$ if it is a part of some induced diamond or $K_4$ in $G$. $G$ has Core-Member property if every vertex and every edge of $G$ is a Core-Member.

**Rule 1** (Core-Member-Selection). Let $(G, k)$ be an input to the rule. If there is an edge $e \in E(G)$ which is not a Core-Member of $G$, then delete $e$ from $G$.

**Lemma 2.1.** Core-Member-Selection rule is safe and can be applied in polynomial time.
Proof. Let \((G, k)\) be an instance of **Diamond-free Edge Deletion**. Let \(G'\) be obtained by applying Rule 1. We claim that \((G, k)\) is a yes-instance if and only if \((G', k)\) is a yes-instance. Let \(S\) be a solution of size at most \(k\) of \((G, k)\). For a contradiction, assume that \(G - S\) has an induced diamond with a vertex set \(D\). Since \(D\) does not induce a diamond in \(G - S\), the edge \(e\) deleted by Rule 1 has both the end points in \(D\). Then \(D\) induces a \(K_4\) in \(G\) and \(e\) is a part of that \(K_4\). This is a contradiction. Conversely, let \(S'\) be a solution of size at most \(k\) of \((G', k)\). Assume that \(G - S'\) has an induced diamond with vertex set \(D\). The edge deleted by Rule 1 has both the end points in \(D\). This is a contradiction as \(D\) induces either a diamond or a \(K_4\) in \(G\) and \(e\) is a part of it.

Since, in polynomial time, we can verify whether an edge is part of an induced diamond of \(K_4\), the rule can be applied in polynomial time. 

**Property 2** (Connected-Neighborhood). For a graph \(G\) and a vertex \(v \in V(G)\), \(v\) has Connected-Neighborhood property if \(G[N(v)]\) is connected. \(G\) has Connected-Neighborhood property if every vertex in \(G\) has Connected-Neighborhood property.

**Rule 2** (Vertex-Split). Let \(v \in V(G)\) and \(v\) does not have Connected-Neighborhood property in \(G\). Let there be \(t > 1\) components in \(G[N(v)]\) with vertex sets \(V_1, V_2 \ldots V_t\). Introduce \(t\) new vertices \(v_1, v_2 \ldots v_t\) and make \(v_i\) adjacent to all vertices in \(V_i\) for \(1 \leq i \leq t\). Delete \(v\).

![An example of the application of Vertex-Split rule](image)

An example of the application of Vertex-Split rule is depicted in Figure 1. We denote the set of vertices created by splitting \(v\) by \(V_v\). For convenience, we identify an edge \((v, u)\) in \(G\) with an edge \((v_j, u)\) in \(G'\) where \(u\) is in the \(j^{th}\) component of \(G[N(v)]\), so that for every set of edges \(S\) in \(G\), there is a corresponding set of edges in \(G'\) and vice versa. We identify a set of vertices \(V' \subseteq V(G) \setminus \{v\}\) with the corresponding vertices in \(G'\). Similarly, we identify \(V' \subseteq V(G') \setminus V_v\) with the corresponding vertex set in \(G\). Before proving the safety of the rule, we prove two simple observations.

**Observation 2.2.** Let Vertex-Split rule be applied on \(G\) to obtain \(G'\). Let \(v \in V(G)\) be the vertex being split. Then:

(i). For any pair of vertices \(\{v_i, v_j\} \subseteq V_v\), the distance between \(v_i\) and \(v_j\) is at least 4.
(ii). Let \( u \in V(G) \setminus \{v\} \) and \( u \) has Connected-Neighborhood property in \( G \). Then \( u \) has Connected-Neighborhood property in \( G' \). Furthermore, every new vertex \( v_i \) introduced in \( G' \) has Connected-Neighborhood property.

**Proof.**

\[ \text{(i). let } \{v_i, v_j\} \subseteq V_u. \text{ Clearly, } v_i \text{ and } v_j \text{ are non-adjacent. Consider any two vertices } u_i \in N(v_i) \text{ and } u_j \in N(v_j). \text{ If } u_i = u_j \text{ or } u_i \text{ and } u_j \text{ are adjacent in } G', \text{ there would be only one vertex generated for the component containing } u_i \text{ and } u_j \text{ in } G[N(v)] \text{ by splitting } v, \text{ which is a contradiction. It follows that the distance between } v_i \text{ and } v_j \text{ is at least 4.}
\]

\[ \text{(ii). If } v \notin N_G(u), \text{ then the neighborhood of } u \text{ is same in both } G \text{ and } G' \text{ and hence } u \text{ has Connected-Neighborhood property in } G'. \text{ Let } v \in N(u). \text{ Since } G[N_G(u)] \text{ is connected, to prove that } G'[N_{G'}(u)] \text{ is connected, it is enough to get an isomorphism between } G[N_G(u)] \text{ and } G'[N_{G'}(u)]. \text{ Let } V' \text{ be the set of all vertices in } N_G[u] \text{ to which } v \text{ is adjacent to. We note that } u \in V'. \text{ Let } v_i \text{ be the vertex generated by splitting } v \text{ for the component in } G[N(v)] \text{ containing } u. \text{ Since, there is only one new vertex introduced for a component of } G[N(v)], \text{ no other new vertex is adjacent to } u \text{ in } G'. \text{ Now, let } V'' \text{ be the set of all vertices in } N_{G'}[u] \text{ to which } v_i \text{ is adjacent to. Proving } V' = V'' \text{ will establish an isomorphism between } G[N_G(u)] \text{ and } G'[N_{G'}(u)]. \text{ In order to prove that } V' = V'' \text{ it is enough to prove that } G[V'] \text{ is connected. This is true since } u \in V' \text{ and } u \text{ is adjacent to all other vertices in } V'. \text{ Since a new vertex is made adjacent to a component in the neighborhood of } v, \text{ every new vertex } v_j \text{ in } G' \text{ has Connected-Neighborhood property.} \]

**Lemma 2.3.** Vertex-Split rule is safe and can be applied in polynomial time.

**Proof.** Let \((G', k)\) be obtained by applying Vertex-Split rule on \((G, k)\). Let \( v \) be the vertex being split. We need to prove that \((G, k)\) is a yes-instance if and only if \((G', k)\) is a yes-instance. Let \( S \) be a solution of size at most \( k \) of \((G, k)\). For a contradiction, assume that \( G' - S \) has an induced diamond with a vertex set \( D' = \{a, b, c, d\} \). At least one vertex in \( D' \) is created by splitting \( v \), otherwise \( D' \) would induce a diamond in \( G - S \). By Observation 2.2, the distance between any two vertices in \( V_v \) is at least 4. Since, the maximum distance between any two vertices in \( D' \) in \( G' \) is at most 2, there can be only at most one vertex from \( D' \) in \( V_v \). Hence \(|V_v \cap D'| = 1 \). Thus, there are two cases and in both the cases we obtain a contradiction.

(i). \( V_v \cap D' = \{a \text{ (or b)}\} \): As \( a \) is adjacent to \( b, c \) and \( d \) in \( G' \), \( v \) is adjacent to \( b, c \) and \( d \) in \( G \). Hence, \( \{v, b, c, d\} \) induces a diamond in \( G - S \).

(ii). \( V_v \cap D' = \{c \text{ (or d)}\} \): As \( c \) is adjacent to \( a \) and \( b \) in \( G' \), \( v \) must be adjacent to \( a \) and \( b \) in \( G \). Similarly, \( \{c, d\} \) is adjacent in \( G' \) if and only if \( \{v, d\} \) is adjacent in \( G \). Hence, \( \{a, b, v, d\} \) induces a diamond in \( G - S \).

Conversely, let \( S' \) be a solution of size at most \( k \) of \((G', k)\). For a contradiction, assume that \( G - S' \) has an induced diamond with a vertex set \( D = \{a, b, c, d\} \). If \( v \notin D \), then \( D \) induces a diamond in \( G' - S \). Now, there are two cases:
(i). \( v = a \) (or \( b \)): Let \( a' \) be the vertex introduced by splitting \( a \) for the component containing \( b, c \) and \( d \) in \( G[N(a)] \). Then \( \{a', b, c, d\} \) induces a diamond in \( G' - S' \).

(ii). \( v = c \) (or \( d \)): Let \( c' \) be the vertex introduced by splitting \( c \) for the component containing \( a \) and \( b \) in \( G[N(c)] \). Irrespective of whether \( c \) is adjacent to \( d \) or not in \( G \), \( \{a, b, c', d\} \) induces a diamond in \( G' - S' \).

Since the components induced by the neighborhood of a vertex can be computed in polynomial time, the rule can be applied in polynomial time.

The next rule deletes an edge \( e \), if \( e \) is the middle edge of \( k + 1 \) otherwise edge-disjoint diamonds. This rule is found in [4].

**Rule 3 (Sunflower).** Let \((G, k)\) be an input to the rule. If there is an edge \( e = \{x, y\} \in E(G) \) such that \( G[N(x) \cap N(y)] \) has a non-matching of size at least \( k + 1 \), then delete \( e \) from \( G \) and decrease \( k \) by 1.

**Lemma 2.4.** Sunflower rule is safe and can be applied in polynomial time.

**Proof.** Let \((G, k)\) be an instance of DIAMOND-FREE EDGE DELETION. Let \( e = \{x, y\} \in E(G) \) and \( V' = N(x) \cap N(y) \). Assume that \( G[V'] \) has a non-matching \( M' \) of size at least \( k + 1 \). Let Sunflower rule is applied on \((G, k)\) to obtain \((G - e, k - 1)\). We need to prove that \((G, k)\) is a yes-instance if and only if \((G - e, k - 1)\) is a yes-instance. Let \( S \) be a solution of size at most \( k \) of \((G, k)\). Every non-edge \( \{a, b\} \) in \( M' \) corresponds to an induced diamond \( \{x, y, a, b\} \) in \( G \). The diamonds corresponds to any two different non-edges in \( M' \) share only one edge \( \{x, y\} \). Since at least one edge from every induced diamond is in \( S \), \( e \) must be in \( S \). Therefore, \( S - e \) is a solution of size at most \( k - 1 \) of \((G - e, k - 1)\). Conversely, let \((G - e, k - 1)\) be a yes-instance. Let \( S' \) be a solution of size at most \( k - 1 \) of \((G - e, k - 1)\). Then, clearly, \( S' \cup \{e\} \) is a solution of size at most \( k \) in \((G, k)\). The rule can be applied in polynomial time as maximum non-matching can be found in polynomial-time.

Now, we give a trivial rule.

**Rule 4 (Irrelevant-Component).** Let \((G, k)\) be an input to the rule. If a component of \( G \) is diamond-free, then delete the component from \( G \).

**Lemma 2.5.** Irrelevant-Component rule is safe and can be applied in polynomial time.

**Proof.** It is straight forward to verify that if a component has no induced diamond then the component can be removed safely. Since, both finding components of a graph and checking whether a component has induced diamond or not, can be done in polynomial time, the rule can be applied in polynomial time.

Now, we are ready with the Phase 1 of the kernelization.
Phase 1
Let \((G, k)\) be an input to Phase 1.
Step 1: Exhaustively apply Core-Member-Selection rule on \((G, k)\) to obtain \((G_1, k)\).
Step 2: Exhaustively apply Vertex-Split rule on \((G_1, k)\) to obtain \((G_2, k)\).
Step 3: Exhaustively apply Sunflower rule on \((G_2, k)\) to obtain \((G_3, k')\).
Step 4: Exhaustively apply Irrelevant-Component rule on \((G_3, k)\) to obtain \((G', k')\).
Step 5: If none of the steps (Step 1 to Step 4) is applicable on \((G', k')\), then return \((G', k')\). Otherwise apply Phase 1 on \((G', k')\).

Lemma 2.6. Let \((G', k')\) be obtained by applying Phase 1 on \((G, k)\). Then:

(i). \(G'\) has Core-Member property.
(ii). \(G'\) has Connected-Neighborhood property.
(iii). Every component in \(G'\) has an induced diamond.
(iv). Let \((G, k)\) be an input to Step 1. After executing steps 1 to 4, let \((G', k')\)
be the resultant instance. Then \(|E(G')| \leq |E(G)|\) and \(|V(G')| \leq 2|E(G)|\).

Proof. \([\text{iii}]\). If \(G'\) does not have Core-Member property, then there exist an edge or a vertex which is not a Core-Member of \(G'\). Let \(e \in E(G')\) be not a Core-
Member of \(G'\). Then, Core-Member-Selection rule is applicable on \(G'\), which is a contradiction. Let \(v \in V(G')\) is not a Core-Member of \(G'\). If \(v\) is an isolated vertex, then Irrelevant-Component rule is applicable to \(G'\). If there is an edge incident to \(v\) in \(G'\), since every edge is a Core-Member of \(G'\), \(v\) is a Core-Member of \(G'\).

\([\text{ii}]\). Let \(G'\) does not have Connected-Neighborhood property. Then there is a vertex \(v\) in \(G'\) such that \(v\) does not have Connected-Neighborhood property. Then Vertex-Split rule is applicable to \(G'\).

\([\text{iii}]\). Assume that there is a component in \(G\) which does not have an induced diamond. Then, Irrelevant-Component rule is applicable to \(G'\).

\([\text{iv}]\). None of the steps from 1 to 4 increases the number of edges. Hence \(|E(G')| \leq |E(G)|\). Since, Step 4 removes all isolated vertices, \(|V(G')| \leq 2|E(G')| \leq 2|E(G)|\).

Lemma 2.7. Applying Phase 1 is safe and Phase 1 runs in polynomial time.

Proof. The safety of Phase 1 follows from safety of the rules - Core-Member-
Selection, Vertex-Split, Sunflower and Irrelevant-Component, which we have already proved.
Let \((G, k)\) be an original input (used for the first invocation) to Phase 1. The only step at which the size of the instance increases is Step 2, where, there can be a quadratic blow up in the number of vertices. By Lemma 2.6(iv), Step 4 ensures that, before a subsequent execution of Step 2, the number of vertices is at most twice the number of edges.

By Lemma 2.1, an application of Core-Member-Selection rule runs in polynomial time. Every application of the rule decreases the number of edges. Hence the rule applies at most \(|E(G)|\) times at Step 1. Hence, Step 1 runs in polynomial time.

By Lemma 2.3, an application of Vertex-Split rule runs in polynomial time. By Observation 2.2(ii), every application of the rule decreases the number of vertices not having Connected-Neighborhood property. Hence the rule is applied at most \(|V(G_1)| = |V(G)|\) times at Step 2. Hence Step 2 runs in polynomial time. Since every vertex is split into at most \(|V(G_1)| - 1\) vertices, \(G_2\) has \(O(|V(G_1)|^2) = O(|V(G)|^2)\) vertices.

By Lemma 2.4, an application of Sunflower rule runs in polynomial time. Every application of the rule decreases the number of edges. Hence the rule applies at most \(|E(G_2)| \leq |E(G)|\) times at Step 3. Hence Step 3 runs in polynomial time.

By Lemma 2.5, an application of Irrelevant-Component rule runs in polynomial time. Every application of the decreases the number of vertices. Hence the rule applies at most \(|V(G_3)| = O(|V(G_1)|^2) = O(|V(G)|^2)\) times at Step 4. Hence Step 4 runs in polynomial time.

None of the steps causes an increase in the number of edges. Steps 1 and 3 decreases the number of edges. Step 4 may also delete edges. The total number of times, steps which delete edges execute is at most \(|E(G)|\). Since, Step 4 deletes a component, it does not increase the number of vertices not having Connected-Neighborhood property. Hence, between two execution of Step 2 there must be a step which deletes edges. Hence the total number of times Step 2 executes is at most \(|E(G)| + 1\). Since, no step can execute consecutively, Step 4 executes at most \(|E(G)| + |E(G)| + 1 + 1 = 2|E(G)| + 2\) times. Lemma 2.6(iv) ensures that the size of the input to every step is polynomial in the size of the original input. Hence Phase 1 runs in polynomial time.

We define a vertex modulator for DIAMOND-FREE EDGE DELETION similar to that defined for TRIVIALLY PERFECT EDITING in [6].

**Definition 2.8 (D-modulator).** Let \((G, k)\) be an instance of DIAMOND-FREE EDGE DELETION. Let \(V' \subseteq V(G)\) be such that \(G[V \setminus V']\) is diamond-free. Then, \(V'\) is called a D-modulator.

**Lemma 2.9 ([8]).** A graph \(G\) is diamond-free if and only if every edge in \(G\) is a part of exactly one maximal clique.

For a diamond-free graph \(G\), since every edge is in exactly one maximal clique, there is a unique way of partitioning the edges into maximal cliques. For convenience, we call the set of subsets of vertices, where each subset is the
vertex set of a maximal clique, as a maximal clique partitioning. We note that, 
one vertex may be a part of many sets in the partitioning.

**Lemma 2.10.** Let \((G, k)\) be an instance of **Diamond-free Edge Deletion**. 
Then, in polynomial time, the edge set \(X\) of a maximal set of edge-disjoint diamonds, a D-modulator \(V_X\) of size at most \(4k\) and a maximal clique partitioning \(C\) of \(G[V(G) \setminus V_X]\) can be obtained or it can be declared that \((G, k)\) is a no-instance.

**Proof.** Let \(X = \emptyset\). Include edges of any induced diamond of \(G\) in \(X\). Then, iteratively include edges of any induced diamond of \(G \setminus X\) in \(X\) until \(k + 1\) iterations are completed or no more induced diamond is found in \(G \setminus X\). If \(k + 1\) iterations are completed, then we can declare that the instance is a no-instance as every solution must have at least one edge from every induced diamonds. If the number of iterations is less than \(k + 1\) such that there is no induced diamond in \(G - X\), then \(|X| \leq 5k\), as every diamond has five edges. Let \(V_X\) be the set of vertices incident to the edges in \(X\). Then \(|V_X| \leq 4k\), as every diamond has four vertices. Since \(G[V \setminus V_X]\) has no induced diamond, \(V_X\) is a D-modulator. Since, there are only at most \(k + 1\) iterations and each iteration takes polynomial time, this can be done in polynomial time. Since \(G[V(G) \setminus V_X]\) is diamond-free, by Lemma 2.9 every edge in it is part of exactly one maximal clique. Now, the maximal clique partitioning \(C\) of \(G[V(G) \setminus V_X]\) where each \(C \in C\) is a set of vertices of a maximal clique, can be found by greedily obtaining the maximal cliques, which can be done in polynomial time. \(\square\)

Let \((G, k)\) be an output of Phase 1. Here onward, we assume that \(X\) is an edge set of the maximal set of edge-disjoint diamonds, \(V_X\) is a D-modulator, which is the set of vertices incident to \(X\) and \(C\) is the maximal clique partitioning of \(G[V(G) \setminus V_X]\). Observation 2.11 directly follows from the maximality of \(X\). Observation 2.12 is found in Lemma 3.1 of [5]. It was proved there, if \(G\) is \{claw, diamond\}-free, but is also applicable if \(G\) is diamond-free.

**Observation 2.11.** Every induced diamond in \(G\) has an edge \(\{a, b\}\) such that \(\{a, b\} \in X\).

**Observation 2.12.** Let \(C, C' \in C\). Then:

(i) \(|C \cap C'| \leq 1|.

(ii) If \(v \in C \cap C'\), then there is no edge between \(C \setminus \{v\}\) and \(C' \setminus \{v\}\).

**Proof.** \([\Box]\). Assume that \(x, y \in C \cap C'\). Then the edge \(\{x, y\}\) is part of two maximal cliques, which is a contradiction by Lemma 2.9.

\([\Box]\). Let \(x \in C \setminus \{v\}\) and \(y \in C' \setminus \{v\}\). Let \(x\) and \(y\) be adjacent. Then \(\{x, v\}\) is part of two maximal cliques, which is a contradiction. \(\square\)

**Definition 2.13 (Local Vertex).** Let \(G\) be a graph and \(C \subseteq V(G)\) induces a clique in \(G\). A vertex \(v\) in \(C\) is called local to \(C\) in \(G\), if \(N(v) \subseteq C\).
Lemma 2.14. Let \((G, k)\) be an instance of Diamond-free Edge Deletion. Let \(C\) be a clique with at least \(2k + 2\) vertices in \(G\).

(i). Every solution \(S\) of size at most \(k\) of \((G, k)\) does not contain any edge \(e\) where both the end points of \(e\) is in \(C\).

(ii). Let \(C' \subseteq C\) be such that every vertex \(v \in C'\) is local to \(C\) in \(G\). Every induced diamond with vertex set \(D\) in \(G\) can contain at most one vertex in \(C'\).

(iii). Let \(C' \subseteq C\) be such that every vertex \(v \in C'\) is local to \(C\) in \(G\). Then, it is safe to delete \(\min\{|C'|-1,|C|- (2k+2)\}\) vertices of \(C'\) in \(G\).

Proof. \(\Box\). Let \(e = \{x, y\}\) be an edge in \(G\) such that \(x, y \in C\). Let \(S\) be a solution of size at most \(k\) of \((G, k)\) such that \(e \in S\). Consider any two vertices \(a, b \in C \setminus \{x, y\}\) (assuming \(k\) is at least 1). Clearly, \(\{a, b, x, y\}\) induces a diamond in \(G - e\). Consider a maximum matching \(M\) of \(G[C \setminus \{x, y\}]\). Since \(C \setminus \{x, y\}\) induces a clique of size at least \(2k\) in \(G\), \(|M| \geq k\). For any two edges \(\{a, b\}, \{a', b'\} \in M\), the diamonds induced by \(\{a, b, x, y\}\) and \(\{a', b', x, y\}\) are edge-disjoint. \(S\) must contain one edge from the diamonds corresponds each edge in \(M\). Since \(e \in S\), \(|S| \geq k + 1\), which is a contradiction.

(ii). For a contradiction, assume that \(D\) induces a diamond in \(G\) and \(D\) contains two vertices \(\{x, y\}\) of \(C'\). Let \(a\) and \(b\) be the other two vertices in \(D\). Since \(x\) and \(y\) are local to \(C\) in \(G\), any other vertex in \(G\), is either adjacent to or non-adjacent to both \(x\) and \(y\). Hence \(\{x, y\}\) must be the middle edge of the diamond induced by \(D\). Since the common neighbors of \(x\) and \(y\) form a clique, \(a\) and \(b\) are adjacent in \(G\). Hence \(D\) does not induce a diamond.

(iii). Let \(G'\) be obtained by deleting a set \(C''\) of \(t\) vertices of \(C'\) from \(G\) such that \(t = \min\{|C'|-1,|C|- (2k+2)\}\). We need to prove that \((G', k)\) is a yes-instance if and only if \((G', k)\) is a yes-instance. Let \(S\) be a solution of size at most \(k\) of \((G, k)\). Since \(G' - S\) is an induced subgraph of \(G - S\), and \(G - S\) is diamond-free, we obtain that \(G' - S\) is diamond-free. Conversely, let \(S'\) be a solution of size at most \(k\) of \((G', k)\). We claim that \(S'\) is a solution of \((G, k)\). Assume not. Let \(G' - S'\) has an induced diamond with a vertex set \(D\). Since \(|C| - |C''| \geq 2k+2\), by (ii), \(S'\) does not contain any edge in the clique induced by \(C \setminus C''\) in \(G'\). Now there are three cases:

(a). \(C'' \cap D = \emptyset\): In this case \(D\) induces a diamond in \(G' - S'\), which is a contradiction.

(b). \(C'' \cap D = \{v\}\): We observe that we retained at least one vertex \(u\) of \(C'\) in \(G'\). By (ii), \(D\) does not contain any other vertex from \(C'\). Then, \(D \cup \{u\} \setminus \{v\}\) induces a diamond in \(G' - S'\).

(c). \(\{v, v'\} \subseteq C'' \cap D\): This case is not possible by (ii). \(\Box\)
We partition $C$ into three - $C_1, C_2$ and $C_{\geq 3}$, the sets of vertices of maximal cliques with one, two and three or more vertices respectively. The first in the following observation has been proved in Lemma 3.2 in [5] in the context where $G - V_X$ is \{diamond,claw\}-free. Here we prove it in the context where $G - V_X$ is diamond-free.

**Observation 2.15.** Let $C \in C_{\geq 3}$. Then:

(i). If there is a vertex $v \in V_X$ such that $v$ is adjacent to at least two vertices in $C$. Then $v$ is adjacent to all vertices in $C$.

(ii). A vertex in $V(G) \setminus (V_X \cup C)$ is adjacent to at most one vertex in $C$.

**Proof.**

(i). Let $v$ is adjacent to two vertices in $x, y$ in $C$ but not adjacent to $z \in C$. Then $\{x, y, v, z\}$ induces a diamond such that none of the edges of the diamond is in $X$.

(ii). Assume that a vertex $u \in V(G) \setminus (V_X \cup C)$ is adjacent to all vertices in $C$. This contradicts the fact that $C$ induces a maximal clique in $G - V_X$. Let $u$ be adjacent to at least two vertices $\{a, b\}$ in $C$ and non-adjacent to at least one vertex $v \in C$. Then $\{a, b, u, v\}$ induces a diamond where none of the edges of the diamond is in $X$. \hfill \Box

Consider $C \in C$. We define three sets of vertices in $G$ based on $C$.

$$A_C = \{v \in V_X : v \text{ is adjacent to all vertices in } C\}$$

$$B_C = \{v \in V(G) \setminus (V_X \cup C) : v \text{ is adjacent to exactly one vertex in } C\}$$

$$D_C = \{v \in V_X : v \text{ is adjacent to exactly one vertex in } C\}$$

For a vertex $v \in C$, let $B_v$ denote the set of all vertices in $B_C$ adjacent to $v$. Similarly let $D_v$ denote the set of all vertices in $D_C$ adjacent to $v$.

**Observation 2.16.** Let $C \in C$. Then,

(i). The set of vertices in $V(G) \setminus C$ adjacent to at least one vertex in $C$ is $A_C \cup B_C \cup D_C$.

(ii). If $|C| > 1$, then $A_C$ induces a clique in $G$.

(iii). For two vertices $u, v \in C$, $B_u \cap B_v = \emptyset$ and $D_u \cap D_v = \emptyset$.

**Proof.**

(i). [Directly follows from Observation 2.15]

(ii). Assume not. Let $a$ and $b$ be two non-adjacent vertices in $A_C$. By Observation 2.15, both $a$ and $b$ are adjacent to all vertices in $C$. Consider any two vertices $x, y \in C$. $\{x, y, a, b\}$ induces a diamond with no edge in $X$, which is a contradiction.

(iii). [Directly follows from the definition of $B_C$ and $D_C$]. \hfill \Box

**Lemma 2.17.** Let $v \in C_{\geq 3}$. If $B_v$ is non-empty then $D_v$ is non-empty.
Proof. By Connected-Neighborhood property, \( G[N(v)] \) is connected. We observe that \( N(v) = A_C \cup B_v \cup D_v \cup (C \setminus \{v\}) \). Assume \( B_v \) is non-empty. There is no edge between the sets \( B_v \) and \( C \setminus \{v\} \). Consider a vertex \( v_b \in B_v \) adjacent to \( A_C \cup D_v \). Assume \( v_b \) is not adjacent to \( D_v \). Then \( v_b \) must be adjacent to a vertex \( v_a \in A_C \). Let \( v' \) be any other vertex in \( C \). Then \( \{v_a, v, v', v_b\} \) induces a diamond which has no edge intersection with \( X \). Therefore \( v_b \) must be adjacent to a vertex in \( D_v \).

**Observation 2.18.** Let \( C \in C \). Then there are two adjacent vertices \( x \) and \( y \) such that \( x \in A_C \) and \( y \in A_C \cup D_C \).

**Proof.**

**Case 1:** \( C = \{v\} \in C_1 \). Since \( \{v\} \in C_1 \), \( v \) is not adjacent to any vertex in \( V(G) \setminus V_X \). Since \( v \) is a Core-Member, \( v \) is part of an induced diamond or \( K_4 \) in \( G \). Hence there exist two adjacent vertices \( x, y \in A_C \).

**Case 2:** \( C = \{u, v\} \in C_2 \). Since the edge \( \{u, v\} \) is a Core-Member, it is part of some induced diamond or \( K_4 \) in \( G \). Let \( a, b \) be the other two vertices in an induced diamond or \( K_4 \) in which \( \{u, v\} \) is a part. If both \( a, b \in V(G) \setminus V_X \), then it contradicts with either the maximality of \( X \) (if \( a, b, u \) and \( v \) induce a diamond) or with the fact that \( \{x, y\} \) is part of exactly one maximal clique (if \( a, b, u \) and \( v \) induce a \( K_4 \)). Let \( a \in V_X \) and \( b \in V(G) \setminus V_X \). Then, if \( a, b, u \) and \( v \) induces a diamond, then it contradicts with the maximality of \( X \). If \( a, b, u \) and \( v \) induces a \( K_4 \), then \( u, v \) and \( b \) induce a \( K_3 \) which contradicts with the fact that \( \{u, v\} \) is a part of exactly one maximal clique. Hence \( a, b \in V_X \). Since \( a, b, u, v \) induce a diamond or a \( K_4 \), one of \( a, b \) must be adjacent to both \( u \) and \( v \) and the other vertex must be adjacent to at least one of \( u \) and \( v \).

**Case 3:** \( C \in C_{\geq 3} \). Assume that \(|A_C| = 0\). If \( B_C \cup D_C = \emptyset \), then by Observation 2.16, the clique \( C \) is a component in \( G \). Then, Irrelevant-Component rule is applicable. Hence \( B_C \cup D_C \) is non-empty. Consider a vertex \( v \in C \) such that \( B_v \cup D_v \) is non-empty. Consider \( N(v) \). \( G[N(v)] \) has at least two components, one from \( B_v \cup D_v \) and the other from \( C \), which contradicts with the Connected-Neighborhood property of \( v \). Hence, \(|A_C| > 0\). Assume \(|A_C| = \{x\} = 1\). For a contradiction, assume that \( D_C = \emptyset \). Then Lemma 2.17 implies that \( B_C \) is empty. Then \( x \) violates Connected-Neighborhood property. Hence, \( D_C \) is non-empty. If \(|A_C| \geq 2\), we are done.

**Lemma 2.19.** Let \( C \in C_{\geq 3} \). Then, the number of vertices in \( C \) which are adjacent to at least one vertex in \( B_C \cup D_C \) is at most \( 4k - 1 \).

**Proof.** By Observation 2.18, \(|A_C| \geq 1\). Since \(|V_X| \leq 4k\), \(|D_C| \leq 4k - 1\). Let \( C' \) be the set of vertices in \( C \) which are adjacent to \( B_C \cup D_C \). For every vertex \( v \in C' \), by Lemma 2.17, if \( B_v \) is non-empty, then \( D_v \) is non-empty. Since \( v \in C' \), if \( B_v \) is empty, then also \( D_v \) is non-empty. For any two vertices \( v, u \in C' \), by Observation 2.16, \( D_u \cap D_v = \emptyset \). Therefore \(|C'| \leq |D_C| \leq 4k - 1\).

Now, we state the last rule of the kernelization.

**Rule 5 (Clique-Reduction).** Let \( C \in C_{\geq 3} \) such that \(|C| > 4k\). Let \( C' \) be \( C \cup A_C \). Let \( C'' \) be the set of vertices in \( C \) which are local to \( C' \). Then, delete any \(|C''| - 1\) vertices from \( C'' \).
Observation 2.20. After the application of Clique-Reduction rule, the number of vertices retained in $C$ is at most $4k$.

Proof. By Lemma 2.19, the number of vertices in $C$ which are not local to $C'$ is at most $4k - 1$. Hence, the rest of the vertices in $C$ are local to $C'$ in $G$. If $|C| > 4k$, Clique-Reduction rule retains only one local vertex and deletes all other vertices in $C$ local to $C'$.

Lemma 2.21. Clique-Reduction rule is safe and can be applied in polynomial time.

Proof. The safety of the rule follows directly from Lemma 2.14(iii). Once the $X, V_X$ and $C$ are given, the rule finds a maximal clique $C'$ with more than $4k$ vertices and deletes all except one vertex in $C$ local to $C'$ in $G$. It is straightforward to verify that this can be done in polynomial time.

Now we give the kernelization algorithm.

| Kernelization of DIAMOND-FREE EDGE DELETION |
|-------------------------------------------|
| Let $(G, k)$ be the input.                |
| Step 1: Apply Phase 1 on $(G, k)$ to obtain $(G_1, k_1)$. |
| Step 2: Find $X, V_X$ and $C$ of $G_1$. Apply Clique-Reduction rule on $(G_1, k_1)$ to obtain $(G', k_1)$. |
| Step 3: If neither Step 1 nor Step 2 is applicable on $(G', k_1)$, then return $(G', k_1)$. Otherwise apply the kernelization on $(G', k_1)$. |

Lemma 2.22. The kernelization algorithm is safe and can be applied in polynomial time.

Proof. The safety of the kernelization follows directly from the safety of Phase 1 and Clique-Reduction rule. By Lemma 2.7, Phase 1 runs in polynomial time. By Lemma 2.10 and Lemma 2.21, Step 2 runs in polynomial time. Every application of Clique-Reduce rule decreases the number of edges. Hence Step 2 runs at most $|E(G)|$ times. Two execution of Phase 1 cannot be done consecutively. Hence the kernelization runs in polynomial time.

2.1. Bounding the Kernel Size

In this subsection, we bound the number of vertices in the kernel obtained by the kernelization.

Let $(G, k)$ be an instance of DIAMOND-FREE EDGE DELETION and $(G', k')$ is obtained by the kernelization. Consider an $X, V_X$ and $C$ of $(G', k')$.

Lemma 2.23. $\sum_{C \in C_1} |C| = O(k^3)$.
Proof. Since, Phase 1 is not applicable on \((G', k')\), by Lemma 2.16, every vertex is a Core-Member of \(G'\). Let \(\{v\} \subseteq C_1\). By Observation 2.18, \(v\) must be adjacent to two vertices \(x, y \in V_X\) such that \(x\) and \(y\) are adjacent. Now consider the edge \(\{x, y\}\). In the common neighborhood of \(\{x, y\}\) there can be at most \(2k + 1\) vertices \(v\) with the property that \(\{v\} \subseteq C_1\) (otherwise Sunflower rule applies). Since there are at most \(O(k^2)\) edges in \(G'[V_X]\), we obtain that the total number of vertices in the singleton sets of \(C\) is \(O(k^3)\). \(\Box\)

Lemma 2.24. (i) Consider any two vertices \(x, y \in V_X\). Let \(C' \subseteq C_2 \cup C_{\geq 3}\) such that for any \(C \in C'\), \(x, y \in A_C\). If \(\{x, y\} \subseteq X\) then \(|C'| \leq 2k + 1\). If \(\{x, y\} \not\subseteq X\), then \(|C'| \leq 1\).

(ii) Consider any ordered pair of vertices \((x, y)\) in \(V_X\) such that \(x\) and \(y\) are adjacent in \(G\). Let \(C' \subseteq C_2 \cup C_{\geq 3}\) such that for any \(C \in C'\), \(x \in A_C\) and \(y \in D_C\). If \(\{x, y\} \subseteq X\) then \(|C'| \leq 2k + 1\). If \(\{x, y\} \not\subseteq X\), then \(|C'| = 0\).

Proof. (i) Let \(C_a, C_b \subseteq C'\). By Observation 2.12(i), \(|C_a \cap C_b| \leq 1\). If \(v \in C_a \cup C_b\), then by Observation 2.12(ii), there is no edge between \(C_a \setminus \{v\}\) and \(C_b \setminus \{v\}\). Hence, \(\{x, v, a, b\}\) induces a diamond where \(a \in C_a \setminus \{v\}\) and \(b \in C_b \setminus \{v\}\), which is edge disjoint with \(X\), a contradiction. Hence \(C_a \cap C_b = \emptyset\). Now, consider any two vertices \(a \in C_a\) and \(b \in C_b\). Clearly, \(\{x, y, a, b\}\) induces a diamond. Hence, \(\{x, y\}\) must be an edge in \(X\), otherwise the diamond is edge disjoint with \(X\), a contradiction. Therefore, if \(\{x, y\} \not\subseteq X\), \(|C'| \leq 1\). Now we consider the case in which \(\{x, y\} \subseteq X\). If \(|C'| \geq 2k + 2\), we get at least \(k + 1\) diamonds where every two diamond has the only edge intersection \(\{x, y\}\). Then Sunflower rule applies, which is a contradiction.

(ii) Let \(C'\) be the set of all \(C \in C_2 \cup C_{\geq 3}\) such that \(x \in A_C\) and \(y \in D_C\). Consider any two of them - \(C_a\) and \(C_b\). By Observation 2.12(ii), \(|C_a \cap C_b| \leq 1\). If \(v \in C_a \cup C_b\), then by Observation 2.12(ii), there is no edge between \(C_a \setminus \{v\}\) and \(C_b \setminus \{v\}\). Let \(a \in C_a \setminus \{v\}\) and \(b \in C_b \setminus \{v\}\). Then \(\{x, v, a, b\}\) induces a diamond which is edge disjoint with \(X\), a contradiction. Hence \(C_a \cap C_b = \emptyset\). Let \(a, a' \in C_a\) such that \(a\) is adjacent to \(y\). Then, if \(\{x, y\} \not\subseteq X\), \(\{x, a, a', y\}\) induces a diamond, which is edge disjoint with \(X\). Therefore, if \(\{x, y\} \not\subseteq X\), then \(|C'| = 0\). Now we consider the case in which \(\{x, y\} \subseteq X\). If \(|C'| \geq 2k + 2\), we get at least \(k + 1\) diamonds where every two diamond has the only edge intersection \(\{x, y\}\). Then Sunflower rule applies, which is a contradiction. \(\Box\)

Lemma 2.25. \(\sum_{C \in C_2 \cup C_{\geq 3}} |C| = O(k^3)\).

Proof. Consider any two adjacent vertices \(x, y \in V_X\). Let \(C_{xy} \subseteq C_2 \cup C_{\geq 3}\) be such that \(x, y \in A_C\). Then by Lemma 2.24(ii), if \(\{x, y\} \subseteq X\), then \(|C_{xy}'| \leq 2k + 1\) and if \(\{x, y\} \not\subseteq X\), then \(|C_{xy}'| \leq 1\). Since there are at most \(5k\) edges in \(X\) and \(O(k^2)\) edges in \(G[V_X] \setminus X\), \(\bigcup_{(x, y) \in E(G[V_X])} C_{xy}'\) has at most \(O(k) \cdot (2k + 1) + O(k^2) = O(k^2)\) maximal cliques. Since every maximal clique has at most \(4k\) vertices (by Observation 2.20), the total number of vertices in those cliques is \(O(k^3)\).

Now, let \(C_{xy} \subseteq C_2 \cup C_{\geq 3}\) be such that \(x \in A_C\) and \(y \in D_C\). Then by Lemma 2.24(ii), if \(\{x, y\} \subseteq X\), then \(|C_{xy}'| \leq 2k + 1\) and if \(\{x, y\} \not\subseteq X\), then
\[ |C'_{xy}| = 0. \] Since there are at most \(2 \cdot 5k = 10k\) ordered adjacent pairs of vertices in \(X\), \(\bigcup_{(x,y) \in E(G[V_X])} C'_{xy}\) has at most \(O(k) \cdot (2k + 1)\) maximal cliques. Since every maximal clique has at most \(4k\) vertices (by Observation 2.20), the total number of vertices in those cliques is \(O(k^3)\). Since, by Observation 2.18, for every \(C \in \mathcal{C}\), there exist two vertices \(x \in A_C\) and \(y \in A_C \cup D_C\), we have counted every \(C \in \mathcal{C}_2 \cup \mathcal{C} \geq 3\). Hence \(\sum_{C \in \mathcal{C}_2 \cup \mathcal{C} \geq 3} |C| = O(k^3)\). \qed

**Theorem 2.26.** Given an instance \((G, k)\) of **Diamond-free Edge Deletion**, the kernelization gives an instance \((G', k')\) such that \(|V(G')| = O(k^3)\) and \(k' \leq k\) or declares that the instance is a no-instance.

**Proof.** None of the rules increases the parameter \(k\). Then, the theorem follows from Lemma 2.23 and Lemma 2.25 and the fact that \(|V_X| \leq 4k\).

3. **Parameterized Lower Bound**

In this section, we give a polynomial time reduction from **Vertex Cover** on cubic (i.e., every vertex has degree 3) graphs to **Diamond-free Edge Deletion**. **Vertex Cover** on cubic graph is proved to be **NP-complete** in [9]².

Exponential Time Hypothesis (ETH) (along with Sparsification Lemma [11]) is an assumption that there is no algorithm which solves 3-SAT in time \(2^{o(n+m)}(n+m)^{O(1)}\), where \(n\) is the number of variables and \(m\) is the number of clauses. We can use linear parameterized reduction from 3-SAT (with parameter \(n+m\)) to another parameterized problem to show that the latter does not have a subexponential parameterized algorithm, unless ETH fails.

The reduction from 3-SAT to **Vertex Cover** on cubic graphs in [9] is a linear parameterized reduction and hence there is no algorithm which solves **Vertex Cover** on cubic graphs in time \(2^o(k) \cdot |V(G)|^{O(1)}\), where \(k\) is the solution size parameter. The reduction that we give here is also a linear parameterized reduction and hence it proves that **Diamond-free Edge Deletion** is **NP-hard** and there exists no subexponential parameterized algorithm to solve it, unless ETH fails.

**Reduction:** Let \((G, k)\) be an instance of **Vertex Cover** and let \(G\) be a cubic graph. We replace each edge \(uv\) of \(G\) by a path of length 3. For every edge \(uv\), we denote the newly introduced vertices as \(s_{uv}\) and \(s_{vu}\) where \(s_{uv}\) is adjacent to \(u\) and \(s_{vu}\) is adjacent to \(v\). Let \(S\) be the set of all new vertices. For every \(u \in V(G)\), \(S_u\) denotes the three vertices in \(S\) adjacent to \(u\). Make every pair

---

²In [9], it is proved that the **Vertex Cover** on graphs in which every vertex has degree at most 3 is **NP-complete** such that the input 3-SAT instance with \(n\) variables and \(m\) clauses is satisfiable if and only if the output graph has a vertex cover of size at most \(5m\). The output graph has exactly \(3m\) vertices with degree 2 and \(6m\) vertices with degree 3. In order to make sure that every vertex has degree 3, we can use a technique used in [12] to convert a degree 2 vertex by a simple structure so that the 3-SAT instance is satisfiable if and only if the resultant graph has a vertex cover of size at most \(11m\).
of vertices in $S_u$ adjacent to each other such that the vertices in $S_u$ form a triangle. We introduce a universal vertex $w$ which is adjacent to all the vertices in $V(G) \cup S$. For every edge $uv$ in $G$, we make sure that the edge $s_{uv}s_{vu}$ is un-deletable by making it part of a large clique such that deleting $s_{uv}s_{vu}$ will create an unmanageable number of diamonds. For this purpose we introduce a set $C_{(u,v)}$ of $6k$ vertices each of them are adjacent to each other and to both $s_{uv}$ and $s_{vu}$. This completes the reduction. Let the resultant graph be $G'$.

For every vertex $u \in V(G)$, by $G'_u$ we denote a subgraph of $G'$ induced by $S_u \cup \{u, w\}$. $G'$ when $G$ is a $K_4$ is given in Figure 2. We will prove that $(G, k)$ is a yes-instance if and only if $(G', 3k)$ is a yes-instance. Before proving this, we observe some properties of $(G', 3k)$.

![Figure 2: Graph $G'$ when $G$ is a $K_4$. $w$ is adjacent to all visible vertices. A thick edge $s_{uv}s_{vu}$ denotes a clique of size $6k + 2$ with the vertices $C_{(u,v)} \cup \{s_{uv}, s_{vu}\}$. $s_{uv}$ and $s_{vu}$ retain its adjacency as shown in the figure, whereas $C_{(u,v)}$ vertices are adjacent to only the vertices in $C_{(u,v)} \cup \{s_{uv}, s_{vu}\}$.

Observation 3.1. i. Let $E'$ be a solution of size at most $3k$ of $(G', 3k)$. Then $E'$ does not contain any edge from the graph induced by $C_{(u,v)} \cup \{s_{uv}, s_{vu}\}$.

ii. Every induced diamond in $G'$ contain the vertex $w$.

Proof. [1]. Let $C'$ be $C_{(u,v)} \cup \{s_{uv}, s_{vu}\}$. Let $x, y \in C'$ be such that $e = \{x, y\} \in E'$. Consider any pair of vertices $x', y' \in C' \setminus \{x, y\}$. Clearly $\{x', y', x, y\}$ induces a diamond in $G' - e$. Any other pair of vertices $x'', y'' \in C' \setminus \{x, y\}$ such that $\{x', y'\} \cap \{x'', y''\} = \emptyset$ induces a diamond $\{x'', y'', x, y\}$ which is edge disjoint with that induced by $\{x', y', x, y\}$. There should be at least one edge in $E'$ from every such diamond. Since there are $6k$ vertices in $C' \setminus \{x, y\}$, $|E'| \geq 3k + 1$, which is a contradiction.

[2]. Let $H$ be $G' - w$. We claim that $H$ is diamond-free. For every $u \in V(G)$, we observe that $S_u \cup \{u\}$ forms a maximal clique of $H$. For every pair of adjacent
vertices \( \{u, v\} \) in \( G \), \( C_{\{u,v\}} \cup \{s_{uv}, s_{vu}\} \) forms a maximal clique of \( H \). Now, every edge in \( H \) is in one of these maximal cliques. Hence, by Lemma 2.9, \( H \) is diamond-free. \( \square \)

**Theorem 3.2. Diamond-free Edge Deletion is NP-complete.**

**Proof.** Diamond-free Edge Deletion is trivially in \( \text{NP} \). Let \((G, k)\) be an instance of Vertex Cover on cubic graphs and we apply the reduction described to obtain \((G', 3k)\), an instance of Diamond-free Edge Deletion. We need to prove that \((G, k)\) is a yes-instance of Vertex Cover if and only if \((G', 3k)\) is a yes-instance of Diamond-free Edge Deletion.

Let \( U \) be a vertex cover of size at most \( k \) of \( G \). Let \( D = \{s_{uv}w : u \in U, uv \in E(G)\} \), i.e., \( D \) is the set of edges between \( w \) and \( S_u \) for all \( u \in U \). We claim that \( G - D \) is diamond-free. To prove this, we give a maximal clique partitioning of \( G' - D \). For every vertex \( u \in U \), \( S_u \cup \{u\} \) is a maximal clique in \( G' - D \). For every vertex \( v \in V(G) \setminus U \), \( G'_v \) is a maximal clique in \( G' - D \). For every edge \( \{u, v\} \) in \( G \), \( C_{\{u,v\}} \cup \{s_{uv}, s_{vu}\} \) is a maximal clique in \( G' - D \). Now, we observe that every edge in \( G' - D \) is part of some maximal cliques obtained above. Since \( G \) is cubic, \(|D| \leq 3k \).

Conversely, assume that \( D \) is the set of edges in \( G' \) such that \( G' - D \) is diamond-free and \(|D| \leq 3k \). For an edge \( \{u, v\} \) in \( G \), \( \{s_{uv}, s_{vu}, w, c\} \), where \( c \) is any vertex in \( C_{\{u,v\}} \) induces a diamond in \( G' \). Since the only deletable edges in this diamond are \( s_{uv}w \) and \( s_{vu}w \), either of them, say \( s_{uv}w \) must be in \( D \). In that case, we observe that at least 2 more edges have to be deleted from \( G'_u \). This implies that, if at all a single edge is deleted from \( G'_u \), then at least 3 edges must be deleted from \( G'_u \). Hence for every edge \( uv \in E(G) \) at least 3 edges from \( G'_u \) or 3 edges from \( G'_v \) must be in \( D \). Now let \( U = \{u : D \) has an edge from \( G'_u\} \). Clearly, \( U \) is a vertex cover of size at most \( k \). \( \square \)

**Corollary 3.3.** There exist no algorithm to solve Diamond-free Edge Deletion which runs in time \( 2^{o(k)} \cdot |V(G)|^{O(1)} \), unless ETH fails.

**4. Concluding Remarks**

We obtained an \( O(k^3) \) kernel for Diamond-free Edge Deletion which is an improvement over the previously known kernel. We proved that Diamond-free Edge Deletion is NP-complete and does not admit a subexponential time algorithm unless ETH fails. We believe that the Vertex-Split rule introduced in this paper will be useful in similar settings. One way of extending our result is to give a polynomial kernel for \( \mathcal{H} \)-free Edge Deletion where \( \mathcal{H} \) is a finite set of graphs containing diamond. We conclude with an open problem: Does Paw-free Edge Deletion admit a polynomial kernel?
References

[1] NR Aravind, RB Sandeep, and Naveen Sivadasan. On polynomial kernelization of $H$-free edge deletion. In Parameterized and Exact Computation, pages 28–38. Springer, 2014.

[2] Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. Information Processing Letters, 58(4):171–176, 1996.

[3] Leizhen Cai and Yufei Cai. Incompressibility of $H$-free edge modification. In Parameterized and Exact Computation, pages 84–96. Springer, 2013.

[4] Yufei Cai. Polynomial kernelisation of $H$-free edge modification problems. Master’s thesis, Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong SAR, China, 2012.

[5] Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, Erik Jan van Leeuwen, and Marcin Wrochna. Polynomial kernelization for removing induced claws and diamonds. to appear in WG, 2015.

[6] Pål Grønås Drange and Michał Pilipczuk. A polynomial kernel for trivially perfect editing. to appear in ESA, 2015.

[7] Ehab S El-Mallah and Charles J Colbourn. The complexity of some edge deletion problems. Circuits and Systems, IEEE Transactions on, 35(3):354–362, 1988.

[8] A Farrugia. Clique-helly graphs and hereditary clique-helly graphs, a mini-survey. Algorithmic graph theory (CS 762)–2002–Project, Dept. of Combinatorics, University of Waterloo, 2002.

[9] Michael R Garey, David S Johnson, and Larry Stockmeyer. Some simplified NP-complete problems. In Proceedings of the sixth annual ACM symposium on Theory of computing, pages 47–63. ACM, 1974.

[10] Paul W Goldberg, Martin C Golumbic, Haim Kaplan, and Ron Shamir. Four strikes against physical mapping of dna. Journal of Computational Biology, 2(1):139–152, 1995.

[11] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? Journal of Computer and System Sciences, 63(4):512 – 530, 2001.

[12] Bojan Mohar. Face covers and the genus problem for apex graphs. Journal of Combinatorial Theory, Series B, 82(1):102–117, 2001.