# A Refined Exact Algorithm for Edge Dominating Set

Mingyu Xiao

School of Computer Science and Engineering, University of Electronic Science and Technology of China, China,
myxiao@gmail.com

Hiroshi Nagamochi

Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Japan,
nag@amp.i.kyoto-u.ac.jp

**Abstract**

We present an $O^*(1.3160^n)$-time algorithm for the edge dominating set problem in an $n$-vertex graph, which improves previous exact algorithms for this problem. The algorithm is analyzed by using the "Measure and Conquer method." We design new branching rules based on conceptually simple local structures, called "clique-producing vertices/cycles," which significantly simplify the algorithm and its running time analysis, attaining an improved time bound at the same time.

**Key words.** Exact Algorithm, Edge Dominating Set, Measure and Conquer

## 1 Introduction

An *edge dominating set* of a graph $G = (V, E)$ is a subset $M$ of edges such that each edge in $E - M$ is adjacent to at least one edge in $M$. The *edge dominating set* problem (EDS), to find an edge dominating set of minimum size, is a basic NP-hard graph problem and has been extensively studied in the line of research on worst-case analysis of exact exponential algorithms for NP-hard optimization problems. Yannakakis and Gavril [16] showed that EDS is NP-hard even when the graph is restricted to planar or bipartite graphs of maximal degree three. Randerath and Schiermeyer [10] designed the first nontrivial algorithm for EDS, which runs in $O^*(1.4423^{|E|})$ time.[1] The upper bound on the running time was improved frequently later. Raman *et al.* [9] showed an $O^*(1.4423^n)$-time algorithm, where $n = |V|$ is the number of vertices. Fomin *et al.* [4] further improved this result to $O^*(1.4082^n)$ by considering the treewidth of the graph. By using the Measure and Conquer method, Rooij and Bodlaender [11] designed a simple $O^*(1.3323^n)$-time algorithm, and further improved the running time bound to $O^*(1.3226^n)$ by checking a number of local structures. When the graph is restricted to graphs of maximal degree three, the result can be improved to $O^*(1.2721^n)$ [14]. There are also a numerous contributions to the parameterized algorithms for EDS with parameter $k$ being the size of the edge dominating set [3, 4, 1, 15, 13]. Currently, the best result is the $O^*(2.3147^k)$-time algorithm introduced in [13]. In this paper, we will use the Measure and Conquer method to design an improved exact algorithm for EDS, which can also be used to derive faster algorithms for some related problems, such as the weighted edge dominating set problem, the minimum maximal matching problem, the matrix domination problem and so on.

The Measure and Conquer method, first introduced by Fomin *et al.* [5], is a powerful method used to analyze the running time of branching algorithms. Most of the currently best known exact algorithms to particular NP-hard problems are obtained by using this method. The idea behind the Measure and Conquer method is to focus on the choice of the measure, instead of creating algorithms with more and more branching and reduction rules. In this method, coefficients involved in the measure, typically called weights, need to be fixed so as to minimize the established running time. To establish the best value of the weights, usually we need to solve a quasiconvex program. Although the algorithm may be simple, in the analysis, we need to consider many cases and get a

[1] Technical report 2011-014, September 22, 2011.

[1] The $O^*$-notation suppresses polynomial factors.

large number of constraints in the quasiconvex program. Sometimes it is hard to check the cases by hand. In [11], a simple algorithm for EDS is presented by using the Measure and Conquer method, and further improvements are claimed by means of additional branching rules based on a list of local structures. However, a large number of cases arisen and some detailed analysis are omitted in the proof in their extended abstract. In this paper, we will also use the Measure and Conquer method to analyze the algorithm. We identify conceptually simple local structures, called "clique-producing vertices/cycles" to design new branching rules, which makes the algorithm much simpler, attaining an improved time bound at the same time. Finally we can clearly list out the constraints in our quasiconvex program and point out the bottlenecks for our solutions.

Our algorithm for EDS is based on enumeration vertex covers. The idea of this method is introduced in Sec. 2. The branching rules used in the algorithm are given in Sec. 3. Then the algorithm and the analysis are presented in Sec. 4 and Sec. 5 respectively. Finally some concluding remarks are given in Sec. 6.

## 2 Enumeration-Based Algorithms

Given a graph $G$, a subset $C$ of vertices of $G$ is called a *vertex cover*, if any edge in $G$ is incident on at least one vertex in $C$. A subset $I$ of vertices of $G$ is called an *independent set*, if there is no edge between any two vertices in $I$. EDS is an important problem studied from the view of enumeration vertex covers [3, 11, 4, 14]. Note that the vertex set of an edge dominating set is a vertex cover. Conversely, given a minimal vertex cover $C$, a minimal edge dominating set that contains $C$ in the set of its end points can be computed in polynomial time by computing a maximum matching in the induced graph $G[C]$ and adding an edge for each unmatched vertex in $C$. This observation reduces the problem to that of finding such a minimal vertex cover $C$. However, it is not easy to find a "right" vertex cover. The idea is to enumerate all minimal vertex covers to find it. All minimal vertex covers can be enumerated in $O(1.4423^n)$ time [6, 8], which immediately yields an $O^*(1.4423^n)$-time algorithm for EDS. We will use a branch-and-reduce method to find vertex covers. We fix some part of a minimal vertex cover and then try to extend it.

For a set $F$ of edges, let $V(F)$ denote the set of all end points of edges in $F$. For a subset $C \subseteq V$ and an independent set $I \subseteq V - C$ in $G$, an edge dominating set $M$ is called a $(C, I)$-*eds* if

$$C \subseteq V(M) \quad \text{and} \quad I \cap V(M) = \emptyset.$$

In the search for the vertex cover $V(M)$ of a minimum $(C, I)$-eds $M$, we keep track of a partition of the vertices of $G$ in four sets: $C$, $I$, $U_1$ and $U_2$. Initially $C = I = U_1 = \emptyset$ and $U_2 = V$. The following conditions are kept invariant: (i) $I$ is an independent set in $G$, and (ii) each component of $G[U_1]$ is a clique component (a connected component that is a clique) of $G[V \setminus (C \cup I)]$. The vertices in $U_1 \cup U_2$ are called *undecided* vertices. We use a five-tuple

$$(G, C, I, U_1, U_2)$$

to denote the state described above. Rooij and Bodlaender [11] noticed that for each clique component $Q_i$ of $G[V \setminus (C \cup I)]$, at least $|Q_i| - 1$ vertices will be in the vertex set of any edge dominating set. If we create a new graph by introducing a new vertex $v'$ that is adjacent to all vertices in $Q_i$, then in the new graph we can move all the vertices in $Q_i$ together with $v'$ into $C$ to compute edge dominating sets. A minimum $(C, I)$-eds for the original graph can be derived in linear time from a minimum $(C, I)$-eds for the new graph. Then Rooij and Bodlaender [11] showed that

**Lemma 1** *If $U_2 = \emptyset$ then a minimum $(C, I)$-eds $M$ of $G$ can be found in polynomial time.*

When there are no undecided vertices in the graph, we can easily find a minimum $(C, I)$-eds. Lemma 1 tells us that clique components in the *undecided graph* $G[V \setminus (C \cup I)]$ do not cause any trouble in finding a minimum $(C, I)$-eds. We use some branching rules to deal with vertices in $U_2$ and move any newly created clique components into $U_1$.

## 3 Branching Rules and Some Structural Properties

We introduce the branching rules used to move vertices in $U_2$ out of this set. Note that each vertex is either in the vertex set of a minimum edge dominating set or not. Let $v$ be an arbitrary vertex in $U_2$. Then we can branch on $v$ by either including it into $C$ or $I$. When $v$ is included into $I$, we also include all neighbors of $v$ into $C$. This is the simplest branching procedure in our algorithm, called *branching on a vertex* $v$. In our algorithm, once a clique component $Q$ is newly created in $G[V \setminus (C \cup I)]$, we move $V(Q)$ into $U_1$. We use another branching procedure "branching on a 4-cycle." We say that *abcd* is a 4-*cycle*, if there exist four edges *ab*, *bc*, *cd* and *da* in $G[U_2]$. *Branching on a* 4-*cycle abcd* means that we branch by including either $\{a, c\}$ or $\{b, d\}$ into $C$. The correctness of this rule follows from the observation: for a 4-cycle *abcd*, any vertex cover in the graph contains either $\{a, c\}$ or $\{b, d\}$ [12].

When we execute two branching procedures, we choose vertices or 4-cycles carefully to reduce the time complexity. We introduce several rules to choose vertices or 4-cycles for branching in our algorithm. For a vertex $v \in U_2$ in graph $G[U_2]$, let $d(v)$ be the degree of $v$ in $G[U_2]$, $N(v)$ the set of all neighbors of $v$ in $G[U_2]$, $N[v] = N(v) \cup \{v\}$ the set of vertices with distance at most 1 from $v$, $N_2(v)$ the set of vertices with distance exactly 2 from $v$ in $G[U_2]$, and $N_2[v] = N_2(v) \cup N[v]$. For a subset $X \subseteq U_2$ of vertices, let $N(X)$ denote the neighbors of $X$, i.e., $N(X) = \cup_{v \in X} N(v) - X$, and $d(X)$ denote the number of edges between $X$ and $U_2 - X$.

A vertex $v \in U_2$ is called a *clique-producing* vertex (*cp-vertex* for short) if at least one clique component will be generated by removing $v$ from $G[U_2]$. Note that any degree-1 vertex is adjacent to a cp-vertex. A 4-cycle *abcd* is called a *clique-producing* cycle (*cp-cycle* for short) if removing $\{a, c\}$ or $\{b, d\}$ generates at least one clique component. When $G[U_2]$ contains a cp-vertex, we branch on an optimal cp-vertex, where a cp-vertex $v$ is *optimal* if removing $v$ generates the largest total size of cliques. Otherwise we will branch on a cp-cycle or a vertex of maximum degree. We branch on cp-cycles when the maximum degree is $\leq 3$. For branching on a vertex of maximum degree $d$, we may choose an "optimal degree-$d$ vertex." A degree-$d$ vertex $v \in U_2$ is called an *optimal degree-d vertex* if (i) the degree sequence $d(u_1) \leq d(u_2) \leq \cdots \leq d(u_d)$ of the $d$ neighbors $u_1, u_2, \ldots, u_d$ of $v$ is lexicographically minimum over all degree-$d$ vertices; and (ii) $d(N[v])$ (the number of edges between $N[v]$ and $N_2(v)$) is maximized among such vertices $v$ satisfying (i). The following properties about optimal degree-$d$ vertices are used in our algorithm.

**Lemma 2** *Let $v$ be an optimal degree-$d$ vertex in a connected $d$-regular graph $H$ that is not a clique. The $d(N[v]) \geq 4$ for $d = 3$. If $H$ contains more than 6 vertices, then $d(N[v]) \geq 6$ for $d = 4$.*

*Proof.* Since the graph is a $d$-regular graph, we can see that $d(N[v])$ can only be an even integer. If $d(N[v]) = 0$, then the graph is a clique. Hence $d(N[v]) > 0$.

First, we prove the case of $d = 3$. If $d(N[v]) = 2$, we assume that there are two edges $u_0u_1$ and $u_0u_2$, where $u_0$, $u_1$ and $u_2$ are the three neighbors of $v$. Then $u_1$ and $u_2$ are adjacent to some vertex in $N_2(v)$. It is easy to see that $d(N[u_1]) = d(N[u_2]) = 4 > 2$, a contradiction. Therefore, $d(N[v]) \geq 4$ for $d = 3$.

Second, we prove the case of $d = 4$. Note that $d(N[u]) \geq 8$ for any vertex $u$ such that $G[N(u)]$ contains at most two edges. If $d(N[v]) = 2$, then exactly two neighbors $u_1$ and $u_2$ of $v$ are adjacent to $N_2(v)$, and each $u_i$ satisfies $d(N[u_i]) = 6$, a contradiction to the maximality of $d(N[v])$. Let $d(N[v]) = 4$. Now there are exactly 4 edges in $G[N(v)]$. No vertex in $N(v)$ can be adjacent to

three or more vertices in $N_2(v)$. If a vertex $w \in N(v)$ is adjacent to two vertices in $N_2(v)$, then $w$ is adjacent to $w' \in N(v)$ such that $w'$ is not adjacent to any vertex in $N_2(v)$, otherwise there are less than 4 edges in $G[N(v)]$. For this case, $G[N(w)]$ contains at most two edges, and $d(N[w]) \geq 8$ holds, a contradiction. Now each vertex in $G[N(v)]$ is adjacent to exactly one vertex in $N_2(v)$ and then $G[N(v)]$ is a 4-cycle. Since $H$ has at least 8 vertices, it must hold $|N_2(v)| \geq 2$. If a vertex $u \in N_2(v)$ is adjacent to exactly two or three vertices in $N(v)$, then $G[N(u)]$ contains at most two edges, and $d(N[u]) \geq 8$ would hold. Hence each vertex $u \in N_2(v)$ is adjacent to exactly one vertex $u' \in N(v)$. Thus, $|N_2(v)| = 4$, and now for any vertex $u' \in N(v)$, $G[N(u')]$ contains at most two edges, and $d(N[u']) \geq 8$ holds, a contradiction. Therefore, $d(N[v]) \geq 6$. ∎

**Lemma 3** *Let $v$ be an optimal degree-4 vertex in a graph with maximum degree 4 which has no cp-vertices. Assume that $v$ has one degree-3 neighbor and three degree-4 neighbors. If $d(N[v]) < 5$, then $d(N[v]) = 3$ and $N_2(v)$ contains a vertex of degree $\leq 3$.*

*Proof.* Let $u_i$, $i = 0, 1, 2, 3$ be the four neighbors of $v$, where $d(u_0) \leq d(u_1) \leq d(u_2) \leq d(u_3)$. Since $d(N[v]) + d(v) + \sum_{0 \leq i \leq 3} d(u_i)$ is an even integer, $d(v) = 4$ and $\sum_{0 \leq i \leq 3} d(u_i) = 15$, we know that $d(N[v])$ is an odd integer. Let $u_1 \in N(v) - \{u_0\}$ be a neighbor not adjacent to $u_0$. When $d(N[v]) = 1$, only $u_1$ can be adjacent to a vertex in $N_2(v)$, and the vertices $N[v] - \{u_1\}$ induce a clique, implying that $u_1$ is a cp-vertex, a contradiction to the assumption. Let $d(N[v]) = 3$, and $u_j$ ($j \in \{2, 3\}$) be a neighbor of $v$ which is adjacent to $u_0$ (such $u_j$ exists, since otherwise $d(u_1) = d(u_2) = d(u_3) = 4$ cannot be attained). Since $v$ is an optimal vertex, it must hold that $d(N[u_j]) \leq 3 = d(N[v])$. If $u_j$ is adjacent to a vertex in $N_2(v)$, then we see that $d(N[u_j]) \geq 5$, a contradiction. Hence by $d(N[v]) = 3$, there is exactly one such $u_j$ in $N(v) - \{u_0\}$, and hence $u_0$ is adjacent to a vertex $z \in N_2(v)$. If $d(z) = 4$, then we easily see that $d(N[z]) \geq 5$, which contradicts the optimality of $v$. Therefore, $d(z) \leq 3$, as required. ∎

## 4 The Algorithm

Our algorithm for EDS is described in Fig. 1.

## 5 The Analysis

We will use the Measure and Conquer method to analyze the running time bound of our algorithm. We set a vertex weight function $W : \mathbb{N}^* \to \mathbb{R}^*$ in the graph according to the degree of the vertex (where $\mathbb{N}^*$ and $\mathbb{R}^*$ denote the sets of nonnegative integers and nonnegative reals, respectively). We denote by $w_i$ the weight $W(v)$ of a vertex $v$ of degree $i \geq 0$ in $G[U_2]$. Then we adopt $w = \sum_{v \in U_2} W(v) = \sum_i w_i n_i$ as the measure of the graph, where $n_i$ is the number of degree-$i$ vertices in $U_2$. In our algorithm, when $w = 0$, the problem can be solved in polynomial time directly. We also require that $w_i \leq 1$ for all $i$'s. Then the measure $w$ is not greater than the number $n$ of vertices. If we can get a running time bound related to measure $w$, then we can also get a running time bound related to $n$.

Let $C(w)$ denote the worst-case running time $\alpha^w$ to find a solution in graphs that have measure at most $w$ in our algorithm, where $\alpha > 1$ is a constant. To get a running time bound of the algorithm, we show how much the measure $w$ can be reduced in each branching step in the algorithm and then determine an upper bound on $\alpha$.

To simplify case analysis of our algorithm, we set $w_0 = 0$ and $w_i = 1$ for $i \geq 4$. We only need to decide the best values of $w_1, w_2$ and $w_3$. We use $\Delta w_i$ to denote $w_i - w_{i-1}$ for $i \geq 1$, where $\Delta w_i = 0$, $i \geq 5$. Furthermore we let the weight $w$ meet the conditions:

**Input**: A graph $G = (V, E)$ and a partition of $V$ into sets $C$, $I$, $U_1$ and $U_2$. Initially $C = I = U_1 = \emptyset$ and $U_2 = V$.
**Output**: A minimum $(C, I)$-eds.

1. **If** {There is a clique component $Q$ in $G[U_2]$}, move $V(Q)$ into $U_1$.
2. **Elseif**{There are some cp-vertices in $G[U_2]$}, branch on an optimal cp-vertex $v$ by including it into either $C$ or $I$.
3. **Elseif** {There is a vertex $v$ of degree $\geq 5$ in $G[U_2]$}, branch on $v$ by including it into either $C$ or $I$.
4. **Elseif**{There are some degree-4 vertices in $G[U_2]$}, branch on an optimal degree-4 vertex by including it into either $C$ or $I$.
5. **Elseif**{There is a cp-cycle $abcd$ in $G[U_2]$}, branch on it by including either $\{a, c\}$ or $\{b, d\}$ into $C$.
6. **Elseif**{There are some degree-3 vertices in $G[U_2]$}, branch on an optimal degree-3 vertex by including it into either $C$ or $I$.
7. **Elseif** {$G[U_2]$ contains only components of cycles of length $\geq 5$}, branch on any degree-2 vertex by including it into either $C$ or $I$.
8. **Else** (Now $U_2 = \emptyset$ and $C \cup I \cup U_1 = V$ hold.) Compute the candidate $(C, I)$-eds and return the smallest one.

Figure 1: Algorithm $EDS(G)$

$$\begin{array}{c} \Delta w_1 \geq \Delta w_2 \geq \Delta w_3 \geq \Delta w_4 \geq 0, \quad \text{and} \\ w_3 + \Delta w_3 \geq w_4 + 3\Delta w_4. \end{array} \tag{1}$$

### 5.1 Preliminaries

Next, we give the framework of the analysis of branching on a vertex in $G[U_2]$.

Let $v \in U_2$ be a vertex of maximum degree $d$ in $G[U_2]$. Assume that $v$ has $d_i$ neighbors of degree $i \in [0, n]$. Then $d = \sum_{i=1}^{n} d_i = \sum_{i=1}^{d} d_i$. Assume that the algorithm will branch on $v$ by including it into $C$ or $I$. Now we analyze how much we can reduce $w$ in each branch. For a subset $X \subseteq U_2$, we use $\delta(X)$ to denote the amount of $w$ being reduced by removing the vertices in $X$ out the current $U_2$, where $\delta(X)$ depends on $X$ itself, the neighbors of $X$ and any possible cliques resulting from the removal of $X$. More precisely, $\delta(X)$ in the current graph $G[U_2]$ is given as follows

$$\delta(X) = \delta'(X) + \delta''(X) + \delta'''(X)$$

such that $\delta'(X) = \sum_{v \in X} w_{d(v)}$, $\delta''(X) = \sum_{u \in N(X)} (w_{d(u)} - w_{d'(u)})$, and $\delta'''(X) = \sum_{i \geq 1} k_i i w_{i-1}$, where $d'(u)$ and $k_i$ denote the degree of a vertex $u \in U_2 - X$ and the number of cliques with size $i$ newly created by the removal of $X$ in the graph $G[U_2 - X]$.

Now, we consider branching on a single vertex $v$. In the branch where $v$ is moved into $C$, we can delete a degree-$d$ vertex $v$ and reduce the degree of all neighbors of $v$ by 1 in $G[U_2]$. We have $\delta(v) \geq \delta'(v) + \delta''(v)$ with

$$\delta'(v) = w_d \quad \text{and} \quad \delta''(v) = \sum_{i=1}^{d} d_i \Delta w_i.$$

In the branch where $v$ is included into $I$, we also move $N(v)$ into $C$. Then we will delete $N[v]$ from $G[U_2]$, and reduce the degree of the vertices in $N_2(v)$. We get $\delta(N[v]) \geq \delta'(N[v]) + \delta''(N[v])$ with

$$\delta'(N[v]) = w_d + \sum_{i=1}^{d} d_i w_i \quad \text{and} \quad \delta''(N[v]) = \sum_{u \in N_2(v)} (w_{d(u)} - w_{|N(u)-N(v)|}).$$

By (1), a lower bound on $\delta''(N[v])$ can be obtained as follows:

$$\delta''(N[v]) \geq d(N[v])\Delta w_{d'},$$

where $d'$ is the maximum degree of a vertex in $N_2(v)$.

We introduce some lemmas to reduce the case analysis for the running time of our algorithm.

**Lemma 4** *For any positive value $x \geq y \geq t' \geq t \geq 0$, it holds $C(w-y) + C(w-x) \leq C(w-(y-t')) + C(w-(x+t))$.*

*Proof.* It is easy to see that $\alpha^{w-y} + \alpha^{w-x} \leq \alpha^{w-(y-t)} + \alpha^{w-(x+t)}$ holds for any $\alpha > 1$. Then we have $C(w-y) + C(w-x) \leq C(w-(y+t)) + C(w-(x-t))$. By the definition of $C(t)$, we know that $C(t) \leq C(t')$. Then we get $C(w-(y+t)) + C(w-(x-t)) \leq C(w-(y-t')) + C(w-(x+t))$. ∎

In our analysis, we often need to evaluate an upper bound on the formula $C(w-(\delta'(v)+\delta''(v))) + C(w-(\delta'(N[v])+\rho))$ for branching on a vertex $v$, where $\rho \geq 0$ is a lower bound on $\delta''(N[v])$.

**Lemma 5** *For indices $j < k$, let $d'$ be defined by $d'_j = d_j - 1$, $d'_k = d_k + 1$ and $d'_i = d_i$ $(i \neq j, k)$. Then it holds*

$$\begin{aligned} &C(w - (w_d + \textstyle\sum_{i=1}^{n} d_i \Delta w_i)) + C(w - (w_d + \textstyle\sum_{i=1}^{n} d_i w_i + \rho)) \\ &\quad \leq C(w - (w_d + \textstyle\sum_{i=1}^{n} d'_i \Delta w_i)) + C(w - (w_d + \textstyle\sum_{i=1}^{n} d'_i w_i + \rho')) \end{aligned}$$

*if $\rho' \geq \rho \geq 0$ and $w_j + \Delta w_j \geq w_k + \Delta w_k + (\rho' - \rho)$.*

*Proof.* Let $x = w_d + \sum_{i=1}^{d} d_i w_i + \rho$, $y = w_d + \sum_{i=1}^{d} d_i \Delta w_i$, $t = \sum_{i=1}^{n} d'_i w_i - \sum_{i=1}^{d} d_i w_i$, and $t' = \sum_{i=1}^{d} d_i \Delta w_i - \sum_{i=1}^{n} d'_i \Delta w_i + (\rho - \rho')$. By Lemma 4, we obtain the lemma if $x \geq y \geq t' \geq t \geq 0$ holds. We claim that $x \geq y \geq t' \geq t \geq 0$ holds. We easily see that $x \geq y$ since $\rho \geq 0$ and $w_i \geq \Delta w_i$. Also $y \geq \sum_{i=1}^{d} d_i \Delta w_i \geq t'$ holds. Finally we show that $t' \geq t \geq 0$. Note that

$$t = \sum_{i=1}^{n} d'_i w_i - \sum_{i=1}^{n} d_i w_i = w_k - w_j + (\rho' - \rho) \geq 0 \quad \text{(by } k > j \text{ and } \rho' \geq \rho)$$

and

$$t' = \sum_{i=1}^{n} d_i \Delta w_i - \sum_{i=1}^{n} d'_i \Delta w_i = -\Delta w_k + \Delta w_j.$$

By assumption $w_j + \Delta w_j \geq w_k + \Delta w_k + (\rho' - \rho)$ on $j < k$, we see that $t' - t = -\Delta w_k + \Delta w_j - w_k + w_j - (\rho' - \rho) \geq 0$. ∎

Lemma 5 can be applied to any indices $k > j \geq 3$.

### 5.2 Step 2

In Step 2, we branch on an optimal cp-vertex $v$. Let $H$ be the component of $G[U_2]$ containing $v$. After Step 1 is applied, $H$ is not clique. Removing $v$ generates at least one new clique from $H$. In both branches of including $v$ into $C$ and $I$, all the vertices in the cliques will be removed from $U_2$. Let $k$ be the total size of the generated cliques. If $k \geq 2$, then at least three vertices will be removed from $U_2$ by removing each of $v$ and $N[v]$, and one of the removed vertices is of degree $\geq 2$, and we have recurrence

$$C(w) \leq 2C(w - (w_2 + 2w_1)). \tag{2}$$

For $k = 1$, $v$ has exactly one degree-1 neighbor and $d(v) \geq 3$ (otherwise $k \geq 2$ due the optimality of $v$). Hence we have $\delta(v) \geq w_3 + w_1$ and $\delta(N[v]) \geq w_3 + 2w_2 + w_1$, which implies recurrence

$$C(w) \leq C(w - (w_3 + w_1)) + C(w - (w_3 + 2w_2 + w_1)). \tag{3}$$

### 5.3 Step 3

In Step 3, the algorithm will select a vertex $v$ of maximum degree $d \geq 5$ in $G[U_2]$ to branch on. Notice that after Step 2, there is no degree-1 vertex in $U_2$. We get the recurrence $C(w) \leq C(w - \delta(v)) + C(w - \delta(N[v])) \leq C(w - \delta(v)) + C(w - \delta'(N[v]))$. Since we can apply Lemma 5 to indices $k > j \geq 3$ and $\rho' = \rho = 0$, we only need to analyze the case where each neighbor of $v$ is of degree 2 or $\geq 5$. Let $q$ be the number of degree-2 neighbors of $v$. Then it holds

$$\begin{array}{rcl} C(w) & \leq & C(w - \delta(v)) + C(w - \delta'(N[v])) \\ & = & C(w - (w_d + \sum_{i=1}^{d} d_i \Delta w_i)) + C(w - (w_d + \sum_{i=1}^{d} d_i w_i)) \\ & \leq & C(w - (1 + q\Delta w_2)) + C(w - (d + 1 - q + q w_2)), \end{array}$$

and we get recurrences

$$C(w) \leq C(w - (1 + q\Delta w_2)) + C(w - (6 - q + q w_2)), \quad q = 0, 1, \ldots, 5. \tag{4}$$

### 5.4 Step 4

In this step, we will branch on an optimal degree-4 vertex $v$ in $G[U_2]$. Let $d_i$ denote the number of degree-$i$ neighbors of $v$. After Step 2, it holds $d_1 = 0$.

**Case 1.** $d_2 = 4$ and $d_3 + d_4 = 0$: In this case, no two neighbors of $v$ are adjacent since otherwise $v$ would be a cp-vertex. Hence $\delta''(N[v]) \geq 4\Delta w_4$, and we have recurrence

$$\begin{array}{rcl} C(w) & \leq & C(w - (w_4 + 4\Delta w_2)) + C(w - (w_4 + 4w_2 + 4\Delta w_4)) \\ & = & C(w - (1 + 4w_2 - 4w_1)) + C(w - (5 - 4w_3 + 4w_2)). \end{array} \tag{5}$$

**Case 2.** $d_2 = 3$ and $d_3 + d_4 = 1$: Since $w_3 + \Delta w_3 \geq w_4 + \Delta w_4$ by (1), we can apply Lemma 5 with $j = 3$, $k = 4$ and $\rho' = \rho = 0$ to evaluate the recurrence $C(w) \leq C(w - \delta(v)) + C(w - \delta'(N[v]))$. Hence we only need to consider the case of $d_2 = 3$ and $d_4 = 1$, and we get

$$\begin{array}{rcl} C(w) & \leq & C(w - (w_4 + 3\Delta w_2 + \Delta w_4)) + C(w - (2w_4 + 3w_2)) \\ & = & C(w - (2 - w_3 + 3w_2 - 3w_1)) + C(w - (2 + 3w_2)). \end{array} \tag{6}$$

**Case 3.** $d_2 = d_3 + d_4 = 2$: Analogously with Case 2, we only need to consider the case of $d_2 = d_4 = 2$ to evaluate the recurrence $C(w) \leq C(w - \delta(v)) + C(w - \delta'(N[v]))$ by Lemma 5. Hence we get

$$\begin{array}{rcl} C(w) & \leq & C(w - (w_4 + 2\Delta w_4 + 2\Delta w_3)) + C(w - (w_4 + 2w_4 + 2w_3)) \\ & = & C(w - (3 - 2w_2)) + C(w - (3 + 2w_3)). \end{array} \tag{7}$$

**Case 4.** $d_2 = 1$ and $d_3 + d_4 = 3$: We first consider the case of $d_2 = 1$ and $d_4 = 3$. In this case, there are at least two edges between $N(v)$ and $N_2(v)$, and $\delta''(N[v]) \geq \rho' = 2\Delta w_4$. Then we get the recurrence

$$\begin{aligned} C(w) &\leq C(w - \delta(v)) + C(w - (\delta'(N[v]) + \rho')) \\ &\leq C(w - (w_4 + 3\Delta w_4 + \Delta w_2)) + C(w - (4w_4 + w_2 + 2\Delta w_4)) \\ &= C(w - (4 - 3w_3 + w_2 - w_1)) + C(w - (6 - 2w_3 + w_2)). \end{aligned} \tag{8}$$

By Lemma 5, the other cases will be covered by the case of $d_2 = 1$ and $d_4 = 3$, since for $\rho = 0$ and $\rho' = 2\Delta w_4$, it holds that $w_3 + \Delta w_3 \geq w_4 + \Delta w_4 + 2\Delta w_4$ by (1).

**Case 5.** $d_2 = 0$ and $d_3 + d_4 = 4$: We first consider the case of $d_4 = 4$. In this case, all vertices in $G[U_2]$ are of degree 4, and we will select an optimal degree-4 vertex $v$. Let $H$ be the 4-reguler graph containing $v$. If $H$ has only 6 vertices, then all vertices in $H$ will be removed from $U_2$ when $v$ is included into $I$, and branching on $v$ gives a recurrence

$$\begin{aligned} C(w) &\leq C(w - \delta(v)) + C(w - \delta(N[v])) \\ &\leq C(w - (w_4 + 4\Delta w_4)) + C(w - 6) = C(w - (5 - 4w_3)) + C(w - 6). \end{aligned} \tag{9}$$

Let $H$ contains more than 6 vertices. Then $d(N[v]) \geq 6$ by Lemma 2, and $\delta''(N[v]) \geq \rho' = 6\Delta w_4$. We get the recurrence

$$\begin{aligned} C(w) &\leq C(w - \delta(v)) + C(w - (\delta'(N[v]) + \rho')) \\ &\leq C(w - (w_4 + 4\Delta w_4)) + C(w - (w_4 + 4w_4 + 6\Delta w_4)) \\ &= C(w - (5 - 4w_3)) + C(w - (11 - 6w_3)). \end{aligned} \tag{10}$$

For the other case, we get the recurrence $C(w) \leq C(w - \delta(v)) + C(w - (\delta'(N[v]) + \rho))$, where $\rho$ is a lower bound on $\delta''(N[v])$. We will show that the other cases will be covered by the case of $d_4 = 4$ using Lemma 5.

First we consider the case of $d_3 = 1$ and $d_4 = 3$. To show that the recurrence in this case is covered by (10) by Lemma 5 with $j = 3$ and $k = 4$, it suffices to prove that $w_3 + \Delta w_3 \geq w_4 + \Delta w_4 + (\rho' - \rho)$ for a lower bound $\rho$ on $\delta''(N[v])$. Lemma 3 tells that we have $\delta''(N[v]) \geq 2\Delta w_4 + \Delta w_3$ or $\delta''(N[v]) \geq 5\Delta w_4$. By setting $\rho = \min\{2\Delta w_4 + \Delta w_3, 5\Delta w_4, \rho'\}$, we have $\delta''(N[v]) \geq \rho$ and $(\rho' - \rho) \leq 2\Delta w_4$ (by $\Delta w_3 \geq 3\Delta w_4$ in (1)), which implies that $w_3 + \Delta w_3 \geq w_4 + \Delta w_4 + 2\Delta w_4 \geq w_4 + \Delta w_4 + (\rho' - \rho)$ by (1), as required.

Next we consider the case of $d_3 = 2$ and $d_4 = 2$. It holds $d(N[v]) \geq 2$, since otherwise $(d(N[v]) = 0)$ a degree-3 neighbor of $v$ would be a cp-vertex. Hence $\delta''(N[v]) \geq \rho = 2\Delta w_4$. Note that $v$ has two degree-3 neighbors when $d_3 = 3$. By applying Lemma 5 with $j = 3$ and $k = 4$ twice, where $\frac{1}{2}(\rho' - \rho) = 2\Delta w_4$, we see that the recurrence for $d_3 = 2$ is covered by (10), since $w_3 + \Delta w_3 \geq w_4 + \Delta w_4 + 2\Delta w_4 \geq w_4 + \Delta w_4 + \frac{1}{2}(\rho' - \rho)$ by (1).

Finally consider the case of $d_3 = 3$ and $d_4 = 1$. Since the sum of degrees of the neighbors of $v$ is odd, we have $d(N[v]) \geq 1$ and $\delta''(N[v]) \geq \rho = \Delta w_4$. Note that $v$ has three degree-3 neighbors when $d_3 = 3$. By applying Lemma 5 with $j = 3$ and $k = 4$ three times, where $\frac{1}{3}(\rho' - \rho) = \frac{5}{3}\Delta w_4 < 2\Delta w_4$, we see that the recurrence for $d_3 = 3$ is covered by (10), since $w_3 + \Delta w_3 \geq w_4 + \Delta w_4 + 2\Delta w_4 \geq w_4 + \Delta w_4 + \frac{1}{3}(\rho' - \rho)$ by (1).

### 5.5 Step 5

In this step, we will branch on a cp-cycle $abcd$. Note that $G[U_2]$ has none of clique components, cp-vertices and vertices of degree $> 3$ after applying Steps 1-4. In the branch where $a$ and $c$ (resp., $b$ and $d$) are included into the 4-cycle, we will delete $\{a, c\}$ (resp., $\{b, d\}$) from $G[U_2]$ and reduce the degree of $b$ and $d$ (resp., $a$ and $c$) by 2.

If removing each of $\{a, c\}$ and $\{b, d\}$ generates a clique, then we get recurrence $C(w) \leq 2C(w - 3w_2)$, which is covered by (2).

If removing one of $\{a, c\}$ and $\{b, d\}$ generates cliques whose total size is at least 2, then we get recurrence $C(w) \leq C(w - 2w_2) + C(w - 4w_2)$, which is covered by (3) by $\Delta w_2 \geq \Delta w_3$.

The remaining case is that removing one of $\{a, c\}$ and $\{b, d\}$ generates a clique with size 1 and removing the other one does not generate any clique. Thus, only one vertex, say $a$, in the 4-cycle is of degree 2 and the other three are of degree 3. Then we have $\delta(\{a, c\}) \geq w_2+w_3+2(w_3-w_1)+\Delta w_3 = 4w_3 - 2w_1$, and $\delta(\{b, d\}) \geq 2w_3 + w_2 + (w_3 - w_1) + 2\Delta w_3 = 5w_3 - w_2 - w_1$. Therefore, we get

$$C(w) \leq C(w - (4w_3 - 2w_1)) + C(w - (5w_3 - w_2 - w_1)). \tag{11}$$

### 5.6 Step 6

In this step, we will branch on optimal degree-3 vertices in $G[U_2]$, where every vertex is of degree 2 or 3. Let $N(v) = \{a, b, u\}$, where $d(a) \geq d(b) \geq d(u) \geq 2$ is assumed without loss of generality. We distinguish two cases.

**Case 1.** Vertex $u$ is of degree 2: We first show that $\delta(v) \geq 3w_3 - w_2 - w_1$ and $\delta(N[v]) \geq 4w_3$. Since $\Delta w_3 \leq \Delta w_2$, we observe that $\delta(v) \geq w_3 + 2\Delta w_3 + \Delta w_2 = 3w_3 - w_2 - w_1$. Vertex $u$ is not adjacent to $a$ or $b$ if $a$ and $b$ are adjacent, otherwise there would be a cp-cycle which must have been eliminated by Step 5. Also no two degree-2 neighbors of $v$ are adjacent (otherwise $v$ would be a cp-vertex). For $d(a) = d(b) = d(u) = 2$, it holds $d(N[v]) \geq 3$ and we have $\delta(N[v]) \geq w_3+3w_2+3\Delta w_3 = 4w_3$. For $d(a) = 3$ and $d(b) = d(u) = 2$, $d(N[v]) = 0$ would imply that a degree-2 neighbor is a cp-vertex, and it holds $d(N[v]) \geq 2$ by the parity condition of degrees, indicating that $\delta(N[v]) \geq 2w_3 + 2w_2 + 2\Delta w_3 = 4w_3$. Finally for $d(a) = d(b) = 3$ and $d(u) = 2$, $d(N[v]) = 1$ would imply that $N(v)$ contains a cp-vertex, and it holds $d(N[v]) \geq 3$ by the parity condition of degrees, from which we have $\delta(N[v]) \geq 3w_3+w_2+3\Delta w_3 \geq w_3+3w_2+3\Delta w_3 = 4w_3$. Then we get recurrence

$$C(w) \quad \leq C(w - (3w_3 - w_2 - w_1)) + C(w - 4w_3). \tag{12}$$

We here further show that the algorithm will branch on a cp-vertex after including $v$ into $C$. By combining the 'child' recurrence together with the 'parent' recurrence, we derive a better recurrence than (12).

Note that in the branch where $v$ is including into $C$, vertex $u$ will become a degree-1 vertex. Then $G[U_2 - \{v\}]$ contains some cp-vertices and the algorithm will further branch on an optimal cp-vertex $v^*$ in the next step. We also note that any degree-2 vertex can not be in a 4-cycle in $G[U_2]$, since there is no cp-cycle. Then in $G[U_2 - \{v\}]$, each vertex is adjacent to at most one degree-1 vertex.

If the removal of $v^*$ generates cliques of total size at least 2, we have the following arguments. Since $v^*$ is adjacent to at most one degree-1 vertex in $G[U_2 - \{v\}]$, we see that $\delta(v^*) \geq w_3 + w_2 + w_1$ and $\delta(N[v^*]) \geq w_3 + 2w_2 + w_1$. By combining this and (12), we get a recurrence

$$\begin{aligned} C(w) \quad \leq \quad & C(w - (3w_3 - w_2 - w_1) - (w_3 + w_2 + w_1)) \\ & +C(w - (3w_3 - w_2 - w_1) - (w_3 + 3w_2 + w_1)) + C(w - 4w_3) \\ = \quad & C(w - 4w_3) + C(w - (3w_3 + 2w_2)) + C(w - 4w_3). \end{aligned} \tag{13}$$

Otherwise, the removal of $v^*$ generates a clique of size 1. Then $v^*$ can only be a degree-3 vertex in $G[U_2 - \{v\}]$. Without loss of generality, we assume the three neighbors of $v^*$ are $u'$, $a'$ and $b'$, where $d(u') = 1$, $d(a') \geq 2$ and $d(b') \geq 2$. We here show that $\delta(v^*) \geq 3w_3 - 2w_2 + w_1$ and $\delta(N[v^*]) \geq 3w_3 + w_1$.

Since $\Delta w_3 \leq \Delta w_2$, we observe that $\delta(v^*) \geq w_3 + 2\Delta w_3 + w_1 = 3w_3 - 2w_2 + w_1$. For $d(a') = d(b') = 2$, $a'$ and $b'$ are not adjacent (otherwise $v^*$ would be a cp-vertex the removal of which generates a clique $a'b'$ with size 2), and there are two edges between $N(v^*)$ and $N_2(v^*)$, implying that $\delta(N[v^*]) \geq w_3 + 2w_2 + w_1 + 2\Delta w_3 = 3w_3 + w_1$. For $d(a') = 3$ and $d(b') = 2$, there is at least one edge between $N(v^*)$ and $N_2(v^*)$, and we have $\delta(N[v^*]) \geq w_3 + w_1 + w_2 + w_3 + \Delta w_3 = 3w_3 + w_1$.

Finally for $d(a') = d(b') = 3$, there are at least two edges between $N(v^*)$ and $N_2(v^*)$, and we obtain $\delta(N[v^*]) \geq 3w_3 + w_1 + 2\Delta w_3 \geq w_3 + 2w_2 + w_1 + 2\Delta w_3 = 3w_3 + w_1$. Then for branching on $v^*$ we get the recurrence

$$C(w) \leq C(w - (3w_3 - 2w_2 + w_1)) + C(w - (3w_3 + w_1)). \tag{14}$$

Recurrence (12) is followed by (14). This gives the recurrence

$$\begin{array}{rcl} C(w) & \leq & C(w - (3w_3 - w_2 - w_1) - (3w_3 - 2w_2 + w_1)) \\ & & +C(w - (3w_3 - w_2 - w_1) - (3w_3 + w_1)) + C(w - 4w_3) \\ & = & C(w - (6w_3 - 3w_2)) + C(w - (6w_3 - w_2)) + C(w - 4w_3). \end{array} \tag{15}$$

**Case 2.** Vertex $u$ is of degree 3: Note that for this case, the component is a 3-regular graph. By Lemma 2, we know that $d(N[v]) \geq 4$. If $d(N[v]) = 6$, then $\delta''(N[v]) \geq 6\Delta w_3$ and we branch on $v$ with the following recurrence

$$C(w) \leq C(w - (4w_3 - 3w_2)) + C(w - (10w_3 - 6w_2)). \tag{16}$$

Assume that $d(N[v]) = 4$. Then By branching on $v$, we have recurrence

$$C(w) \leq C(w - (4w_3 - 3w_2)) + C(w - (8w_3 - 4w_2)). \tag{17}$$

To get better recurrences, we further analyze how our algorithm continues branching operations after including $v$ into $C$. Note that for this case, each degree-3 vertex in the regular graph in $G[U_2]$ is contained in a triangle, and no two triangles share an edge (otherwise it would form a cp-cycle). Then after removing $v$ from $G[U_2]$, the graph has not a degree-1 vertex nor a triangle containing two degree-2 vertices. Then there are no cp-vertices in $G[U_2 - \{v\}]$. It is still possible that $G[U_2 - \{v\}]$ contains a cp-cycle (see the following Case 2.1). If $G[U_2 - \{v\}]$ contains no cp-cycle, the algorithm will select an optimal degree-3 vertex to branch on, which will be adjacent to a degree-2 vertex (see the following Case 2.2). Let the three neighbors of $v$ be $a, b$ and $c$, where $a$ and $b$ are adjacent.
Case 2.1. Vertices $a$ and $b$ are in a 4-cycle *abut* in $G[U_2]$: Since each degree-3 vertex in the regular graph in $G[U_2]$ is contained in a triangle, the 4-cycle *abut* must share edges with two triangles, say *abv* and *uty*. Then after branching on the degree-3 vertex $v$ with (17) and the algorithm will further branch on the cp-cycle *abut* in the first branch which includes $v$ into $C$ (note that no other cp-vertex or cp-cycle can be created in this branch). Since only two adjacent vertices are degree-3 vertices in the cp-cycle, Then each branch of removing $\{a, u\}$ and $\{b, t\}$ reduces $w$ by $w_3 + w_2 + (w_3 - w_1) + w_2 + \Delta w_3 = 3w_3 + w_2 - w_1$. By combining this and (17), we get

$$\begin{array}{rcl} C(w) & \leq & 2C(w - (4w_3 - 3w_2) - (3w_3 + w_2 - w_1)) + C(w - (8w_3 - 4w_2)) \\ & = & 2C(w - (7w_3 - 2w_2 - w_1)) + C(w - (8w_3 - 4w_2)). \end{array} \tag{18}$$

Case 2.2. Vertices $a$ and $b$ are not in any 4-cycle in $G[U_2]$: We still branch on $v$ with (17). In the branch where $v$ is included into $C$, we will get three degree-2 vertices $a, b$ and $c$, where $a$ and $b$ are adjacent, and $c$ is in a triangle, say $ca'b'$.

In this branch, the algorithm will select a vertex $v'$ adjacent to a degree-2 vertex (one of $\{a, b, c\}$) to branch on with recurrence (12). In the subbranch where $v'$ is included into $C$, there is a cp-vertex $v''$ (by denoting $u_1$ and $t_1$ (resp., $u_2$ and $t_1$) be the two degree-3 vertices adjacent to $a$ and $b$ (resp., $a'$ and $b'$), if $v' = u_i$, then $v'' = t_i$; if $v' = t_i$, then $v'' = u_i$ ($i \in \{1, 2\}$)). Then algorithm will further branch on $v''$ with the following recurrence

$$\begin{array}{rcl} C(w) & \leq & C(w - (w_3 + 2\Delta w_3 + w_2 + w_1)) + C(w - (3w_3 + 2\Delta w_3 + w_2 + w_1)) \\ & = & C(w - (3w_3 - w_2 + w_1)) + C(w - (3w_3 - w_2 + w_1)). \end{array}$$

Considering all these together, we will get

$$\begin{aligned} C(w) &\leq C(w-(4w_3-3w_2)-(3w_3-w_2-w_1)-(3w_3-w_2+w_1)) \\ &\quad +C(w-(4w_3-3w_2)-(3w_3-w_2-w_1)-(3w_3-w_2+w_1)) \\ &\quad +C(w-(4w_3-3w_2)-(4w_3))+C(w-(8w_3-4w_2)) \\ &= 2C(w-(10w_3-5w_2))+C(w-(12w_3-5w_2))+C(w-(8w_3-4w_2)). \end{aligned} \tag{19}$$

### 5.7 Step 7

In this step, the maximum degree of $G[U_2]$ is 2. Note that a component of a path will contain a cp-vertex and a component of a 4-cycle is a cp-cycle. We know that $G[U_2]$ contains only components of cycles of length $\geq 5$. The algorithm will branch on any vertex in the cycle and then branch on optimal cp-vertices in paths created. Lemma 1 in [11] implies that

**Lemma 6** *Algorithm $EDS(G)$ branches on a component of cycle of length $l \geq 5$ in $G[U_2]$ into at most $2^{l/3}$ subbranches, in which all the $l$ vertices are removed from $U_2$.*

*Proof.* Let $C(l)$ (resp., $P(l)$) be the number of subbranches created by Algorithm $EDS(G)$ when branching on a cycle (resp., path) of $l$ edges to remove all vertices in the cycle (resp. path) out of $U_2$. According to our branching rules, we have that

$$C(l) \leq P(l-2)+P(l-4) \quad \text{and} \quad P(l) \leq P(l-3)+P(l-4).$$

First, we prove that $P(l) \leq 7^{l/9}$ for $l \geq 6$. Straightforward computation gives that $P(6) = 4 \leq 7^{6/9}$, $P(7) = 4 \leq 7^{7/9}$, $P(8) = 5 \leq 7^{8/9}$, and $P(9) = 7 = 7^{9/9}$. For $l \geq 10$, by induction on $l$, we get $P(l) \leq P(l-3)+P(l-4) \leq 7^{l/9}(7^{-3/9}+7^{-4/9}) < 7^{l/9}$. Second, we prove that $C(l) \leq 2^{l/3}$ for $l \geq 5$. Straightforward computation gives that $C(5) = 3 < 2^{5/3}$, $C(6) = 4 = 2^{6/3}$, $C(7) = 5 < 2^{7/3}$, $C(8) = 6 < 2^{8/3}$, and $C(9) = 7 < 2^{9/3}$. For $l \geq 10$, we get $C(l) \leq P(l-2)+P(l-4) \leq 7^{(l-2)/9}+7^{(l-4)/9} = 7^{(l-10)/9}(7^{8/9}+7^{4/9}) < 2^{(l-10)/3} \times 2^{10/3} = 2^{l/3}$. ∎

By Lemma 6, we can get the following recurrence $C(w) \leq 2^{l/3}C(w-lw_2)$, which is equivalent to

$$C(w) \leq 2^{1/3}C(w-w_2). \tag{20}$$

### 5.8 Putting All Together

Each of the worst recurrences derived in the above will generate a constraint in our quasiconvex program to solve a best value for $w_1$, $w_2$ and $w_3$.

By solving Recurrence $(i)$ above, we will get that $C(w) \leq (\alpha_i(w_1,w_2,w_3))^w$, where $2 \leq i \leq 20$ $(i \neq 12, 14, 17)$. We choose a value of $w_1, w_2$ and $w_3$ that minimizes $\max_{2\leq i\leq 20,\ i\neq 12,14,17}\{\alpha_i(w_1,w_2,w_3)\}$ under the constraint (1). By solving this quasiconvex program according to the method introduced in [2], we get a running time bound of $O(1.3160^w)$ by setting $w_1 = 0.8120$, $w_2 = 0.9006$ and $w_3 = 0.9893$ for our problem. Now the tight constraints are $\Delta w_2 \geq \Delta w_3$ in (1), (2), (10) and (16).

**Theorem 7** *Algorithm $EDS(G)$ can find a minimum edge dominating set in an $n$-vertex graph $G$ in $O^*(1.3160^n)$ time.*

## 6 Concluding Remarks

In this paper, we have presented an improved exact algorithm for the edge dominating set problem. There are stranded polynomial reductions from the minimum maximal matching problem to EDS and from the matrix domination problem to EDS [16, 3]. Our algorithm implies improved exact algorithms for these two problems. For the *weighted edge dominating set* problem (given a graph together with a nonnegative edge weight, and asked to find an edge dominating set of minimum

weight), Algorithm $EDS(G)$ can also solve it in the same time. However, the last step of $EDS(G)$ to compute candidate $(C, I)$-eds's is different. We can not simply construct an edge dominating set from a maximum matching of $G[C]$. Instead, we need to solve a *generalized edge cover* problem (to find a minimum edge set $M$ in an edge-weighted graph such that each vertex in a specified subset $C$ of vertices is an endpoint of at least one edge in $M$), which can be solved in cubic time [7]. More details about the extensions from EDS to the weighted edge dominating set problem and to the weighted minimum maximal matching problem have been discussed in [11, 3]. Our algorithm also implies improved exact algorithms for the weighted versions of the problems.

Technically, our algorithm adopts a branch-and-search method to enumerate vertex covers for the vertex set of a minimum edge dominating set, and uses the Measure and Conquer method to analyze the size of the search tree. Different from most previous algorithms based on the Measure and Conquer method, our algorithm uses a set of effective branching rules based on local structures clique-producing vertices and cycles, which are conceptually easy to recognize. This enables us to avoid checking numerous local structures and a possibly huge number of case analyses to get improved time bounds. Finally we can clearly list out all the constraints in the quasiconvex program and point out the tight ones to solve the the best weight of the vertices of our problem.

## References


[1] Binkele-Raible, D. and H. Fernau, Enumerate and measure: improving parameter budget management. In *Proceedings IPEC*, LNCS 6478 (2010), pp. 38–49.

[2] Eppstein D.: Quasiconvex analysis of backtracking algorithms. In: SODA, ACM Press (2004), pp. 781–790.

[3] Fernau, H., Edge dominating set: Efficient enumeration-based exact algorithms, In *Proceedings IWPEC*, Springer-Verlag LNCS 4169 (2006), pp. 142–153.

[4] Fomin, F., Gaspers, S., Saurabh, S., Stepanov, A. On two techniques of combining branching and treewidth, *Algorithmica* **54**(2) (2009), pp. 181–207.

[5] Fomin, F., Grandoni, F., Kratsch, D. Measure and Conquer: Domination - A Case Study, *ICALP* Springer-Verlag LNCS 3580 (2005), pp. 191–203.

[6] Johnson, D., Yannakakis, M., Papadimitriou, C. On generating all maximal independent sets, *Information Processing Letters* **27**(3) (1988), pp. 119–123.

[7] Plesnik, J. Constrained weighted matchings and edge coverings in graphs. *Disc. Appl. Math.* **92** (1999), pp. 229–241.

[8] Moon, J.W. and Moser, L. On cliques in graphs, *Israel J. Math.* **3** (1965), pp. 23–28.

[9] Raman, V., Saurabh, S., Sikdar, S. Efficient exact algorithms through enumerating maximal independent sets and other techniques, *Theory of Computing Systems* **42**(3) (2007), pp. 563–587.

[10] Randerath, B., Schiermeyer, I. Exact algorithms for minimum dominating set. Technical Report zaik 2005-501, Universität zu Köln, Germany, 2005.

[11] van Rooij, J. M., Bodlaender, H. L. Exact algorithms for edge domination, In *Proceedings IWPEC*, Springer-Verlag LNCS 5018 (2008), pp. 214–225.

[12] Xiao, M., A simple and fast algorithm for maximum independent set in 3-degree graphs, In *Proceedings WALCOM*, LNCS 5942 (2010), pp. 281–292.

[13] Xiao, M., Kloks, T., Poon, S-H. New Parameterized Algorithms for the Edge Dominating Set Problem, In F. Murlak and P. Sankowski (Eds.):*Proceedings MFCS* 2011, LNCS 6907 (2011), pp. 604–615.

[14] Xiao, M., Nagamochi, H. Exact algorithms for annotated edge dominating set in cubic graphs. TR 2011-009. Kyoto University. 2011.

A preliminary version appeared as: Xiao, M. Exact and parameterized algorithms for edge dominating set in 3-degree graphs, In W. Wu, O. Daescu, (Eds.), *Proceedings COCOA*, Springer-Verlag LNCS 6509 (2010), pp. 387–400.

[15] Xiao, M., Nagamochi, H. Parameterized edge dominating set in cubic graphs, *Proceedings FAW-AAIM*, Springer-Verlag LNCS 6681 (2011), pp. 100–112.

[16] Yannakakis, M., Gavril, F. Edge dominating sets in graphs, *SIAM J. Appl. Math.* **38**(3) (1980), pp. 364–372.