

ALGORITHMIC LOWER BOUNDS FOR PROBLEMS PARAMETERIZED BY CLIQUE-WIDTH

Fedor V. Fomin* Petr A. Golovach Daniel Lokshтанov
Saket Saurabh

Department of Informatics, University of Bergen, N-5020 Bergen, Norway

{fedor.fomin|petr.golovach|daniello|saket.saurabh}@ii.uib.no

Abstract

Many NP-hard problems can be solved efficiently when the input is restricted to graphs of bounded tree-width or clique-width. In particular, by the celebrated result of Courcelle, every decision problem expressible in monadic second order logic is fixed parameter tractable when parameterized by the tree-width of the input graph. On the other hand if we restrict ourselves to graphs of clique-width at most t , then there are many natural problems for which the running time of the best known algorithms is of the form $n^{f(t)}$, where n is the input length and f is some function. It was an open question whether natural problems like GRAPH COLORING, MAX-CUT, EDGE DOMINATING SET, and HAMILTONIAN PATH are fixed parameter tractable when parameterized by the clique-width of the input graph. As a first step toward obtaining lower bounds for clique-width parameterizations, in [SODA 2009], we showed that unless $\text{FPT} \neq \text{W}[1]$, there is no algorithm with run time $O(g(t) \cdot n^c)$, for some function g and a constant c not depending on t , for GRAPH COLORING, EDGE DOMINATING SET and HAMILTONIAN PATH. But the lower bounds obtained in [SODA 2009] are weak when compared to the upper bounds on the time complexity of the known algorithms for these problems when parameterized by the clique-width.

In this paper, we obtain the asymptotically tight bounds for MAX-CUT and EDGE DOMINATING SET by showing that both problems

- cannot be solved in time $f(t)n^{o(t)}$, unless Exponential Time Hypothesis (ETH) collapses; and
- can be solved in time $n^{O(t)}$,

where f is an arbitrary function of t , on input of size n and clique-width at most t .

We obtain our lower bounds by giving non-trivial structure-preserving “linear FPT reductions”.

1 Introduction

Tree-width is one of the most fundamental parameters in Graph Algorithms. Graphs of bounded tree-width enjoy good algorithmic properties similar to trees and this is why many problems which are hard on general graphs can be solved efficiently when the input is restricted to graphs of bounded tree-width. On the other hand, many hard problems also become tractable when restricted to graphs “similar to complete graphs”.

Courcelle and Olariu [6] introduced the notion of clique-width which captures nice algorithmic properties of both extremes.

Since 2000, the research on algorithmic and structural aspects of clique-width is an active direction in Graph Algorithms, Logic, and Complexity. Cornél, Habib, Lanlignel, Reed, and Rotics [4] show that graphs of clique-width at most 3 can be recognized in polynomial time. Fellows, Rosamond, Rotics, and Szeider [10] settled a long standing open problem by showing that computing clique-width is NP-hard. Oum and Seymour [26] describe an algorithm that, for any fixed t , runs in time $O(|V(G)|^9 \log |V(G)|)$ and computes $(2^{3t+2} - 1)$ -expressions for a graph G of clique-width at most t . Recently, Hliněný and Oum obtained an algorithm running in time $O(|V(G)|^3)$ and computing $(2^{t+1} - 1)$ -expressions for a graph G of clique-width at most t [18]. We refer to the recent survey [19] for further information on different width parameters beyond tree-width.

There was an intensive study on the algorithmic perspective of graphs of bounded clique-width. There is a meta-theorem of Courcelle, Makowsky, and Rotics [5] that all problems expressible in MS_1 -logic are fixed parameter tractable when parameterized by the clique-width of a graph. For many other problems, that are not expressible in this logic, like MAX-CUT, EDGE DOMINATING SET, GRAPH COLORING, or HAMILTONIAN CYCLE, there is a significant amount of the literature devoted to algorithms for these problems and their generalizations on graphs of bounded clique-width [9, 14, 15, 16, 21, 22, 23, 27, 28, 29]. The running time of all these algorithms on an n -vertex graph of clique-width at most t is $O(n^{f(t)})$, where f is some function of t .

One of the central questions in the area is whether the bound of $O(n^{f(t)})$ on the running time of all these algorithms is asymptotically optimal. Even the existence of fixed parameter tractable algorithms (with clique-width being the parameter) for all these prob-

*Partially supported by the Norwegian Research Council

lems (or their generalizations) was open until very recently [14, 21, 22, 23, 16]. As the first step toward obtaining lower bounds for clique-width parameterizations, we have shown in [12] that unless $\text{FPT} \neq \text{W}[1]$, there is no algorithm with run time $O(g(t) \cdot n^c)$, for some function g and a constant c not depending on t , for GRAPH COLORING, EDGE DOMINATING SET and HAMILTONIAN PATH.

Even though our results in [12] resolve the parameterized complexity of these problems, the conclusion that unless $\text{FPT} \neq \text{W}[1]$, there is no algorithm with run time $O(g(t) \cdot n^c)$, for some function g and a constant c not depending on t , is weak to compare the known algorithmic upper bounds. In this paper, we provide asymptotically tight optimal lower bounds for MAX-CUT and EDGE DOMINATING SET. In particular, we show that unless ETH fails, there is no $f(t)n^{o(t)}$ -time algorithm for these problems, where f is an arbitrary function of k , on input of size n and clique-width at most k . While known algorithms for these problems run in times $n^{O(t^2)}$ [21, 22, 9, 29], we give new algorithmic upper bounds of the form $n^{O(t)}$. These two results together, lower and upper bounds, give asymptotically tight algorithmic bounds for MAX-CUT and EDGE DOMINATING SET.

To obtain our lower bounds we construct “linear FPT-reductions”. These type of reductions are much more stringent and delicate than the usual FPT reductions. This is the reason why this research direction is still in a nascent stage and not so many asymptotically tight results are known in the literature. Chen et al. [2, 3] initiated this area of strong computational lower bounds and showed that there is no algorithm for k -CLIQUE (finding a clique of size k) running in time $f(k)n^{o(k)}$ unless there exists an algorithm for solving 3-SAT running in time $2^{o(n)}$ on a formula with n -variables. The assumption that there does not exist an algorithm for solving 3-SAT running in time $2^{o(n)}$ is known as EXPONENTIAL TIME HYPOTHESIS (ETH) [20] and it is equivalent to the parameterized complexity conjecture that $\text{FPT} \neq \text{M}[1]$ [7, 11]. The lower bound on k -CLIQUE can be extended to some other parameterized problems via a linear FPT-reductions [2, 3]. This kind of investigation has also been useful in obtaining tight algorithmic lower bounds for polynomial time approximation schemes [25] and for constraint satisfaction problems when parameterized by the tree-width of the “primal graph” [24]. We further extend the utility of this approach by obtaining asymptotically tight algorithmic bounds for clique-width parameterizations.

2 Definitions and Preliminary results

Parameterized Complexity: Parameterized complexity is a two dimensional framework for studying the computational complexity of a problem. One dimension is the input size n and another one is a *parameter* k . We refer to the books of Downey and Fellows [8] and Flum and Grohe [11] for a detailed treatment to parameterized complexity. Now we define the notion of parameterized (linear) reduction which is the main tool for establishing of our results.

DEFINITION 1. Let A, B be parameterized problems. We say that A is (uniformly many:1) **FPT-reducible** to B if there exist functions $f, g : \mathbb{N} \rightarrow \mathbb{N}$, a constant $\alpha \in \mathbb{N}$ and an algorithm Φ which transforms an instance (x, k) of A into an instance $(x', g(k))$ of B in time $f(k)|x|^\alpha$ so that $(x, k) \in A$ if and only if $(x', g(k)) \in B$. The reduction is called linear if $g(k) = O(k)$.

Graphs: We only consider finite undirected graphs without loops or multiple edges. The vertex set of a graph G is denoted by $V(G)$ and its edge set by $E(G)$. A set $S \subseteq V(G)$ of pairwise adjacent vertices is called a *clique*. For $v \in V(G)$, by $E_G(v)$ we mean the set of edges incident to v . For a vertex v , we denote by $N_G(v)$ its (*open*) *neighborhood*, that is, the set of vertices which are adjacent to v . The *closed neighborhood* of v , that is, the set $N_G(v) \cup \{v\}$, is denoted by $N_G[v]$. The *degree* of a vertex v is denoted by $d_G(v)$. For a graph G , the incidence graph of G is the bipartite graph $I(G)$ with the vertex set $V(G) \cup E(G)$ such that $v \in V(G)$ and $e \in E(G)$ are adjacent if and only if v is incident to e in G .

Tree-width: A *tree decomposition* of a graph G is a pair (X, T) where T is a tree whose vertices we will call *nodes* and $X = (\{X_i \mid i \in V(T)\})$ is a collection of subsets of $V(G)$ such that

1. $\bigcup_{i \in V(T)} X_i = V(G)$,
2. for each edge $vw \in E(G)$, there is an $i \in V(T)$ such that $v, w \in X_i$, and
3. for each $v \in V(G)$ the set of nodes $\{i \mid v \in X_i\}$ forms a subtree of T .

The *width* of a tree decomposition $(\{X_i \mid i \in V(T)\}, T)$ equals $\max_{i \in V(T)} \{|X_i| - 1\}$. The *tree-width* of a graph G is the minimum width over all tree decompositions of G . We use notation $\text{tw}(G)$ to denote the tree-width of a graph G .

Clique-width: Let G be a graph, and t be a positive integer. A t -graph is a graph whose vertices are labeled by integers from $\{1, 2, \dots, t\}$. We call the t -graph consisting of exactly one vertex labeled by some integer

from $\{1, 2, \dots, t\}$ an initial t -graph. The *clique-width* $\mathbf{cwd}(G)$ is the smallest integer t such that G can be constructed by means of repeated application of the following four operations: (1) *introduce*: construction of an initial t -graph labeled by i (denoted by $i(v)$), (2) *disjoint union* (denoted by \oplus), (3) *relabel*: changing all labels i to j (denoted by $\rho_{i \rightarrow j}$) and (4) *join*: connecting all vertices labeled by i with all vertices labeled by j by edges (denoted by $\eta_{i,j}$).

An *expression tree* of a graph G is a rooted tree T of the following form:

- The nodes of T are of four types i , \oplus , η and ρ .
- Introduce nodes $i(v)$ are leaves of T , corresponding to initial t -graphs with vertices v , which are labeled i .
- A union node \oplus stands for a disjoint union of graphs associated with its children.
- A relabel node $\rho_{i \rightarrow j}$ has one child and is associated with the t -graph, which is the result of relabeling operation for the graph corresponding to the child.
- A join node $\eta_{i,j}$ has one child and is associated with the t -graph, which is the result of join operation for the graph corresponding to the child.
- The graph G is isomorphic to the graph associated with the root of T (with all labels removed).

The *width* of the tree T is the number of different labels appearing in T . If a graph G has $\mathbf{cwd}(G) \leq t$ then it is possible to construct a rooted expression tree T with *width* t of G . Given a node X of an expression tree, the graph G_X represents the graph formed by the subtree of the expression tree rooted at X .

A well-known fact is that if the tree-width of a graph is bounded then its clique-width also is bounded. On the other hand, complete graphs have clique-width 2 and unbounded tree-width. But for sparse graphs the tree-width and clique-width are linearly related. Particularly, Gurski and Wanke [17] proved that if a graph G has no subgraph isomorphic to $K_{r,r}$, then $\mathbf{tw}(G) \leq 3(r-1)\mathbf{cwd}(G) - 1$. Linear upper bounds of the clique-width by the tree-width for sparse graphs were established by Fomin et al. [13]. We use the following proposition.

PROPOSITION 2.1. ([13]) *If G is a planar graph, then $\mathbf{cwd}(G) \leq 12(\mathbf{tw}(G) + 1)$.*

Moreover, the proof is constructive and an expression tree for G of width at most $12(\mathbf{tw}(G) + 1)$ can be constructed in FPT time (with tree-width being the parameter) from the tree decomposition of a planar graph G .

Capacitated Domination – Preliminary Results.

A *capacitated graph* is a pair (G, c) , where G is a graph and $c: V(G) \rightarrow \mathbb{N}$ is a *capacity* function such that $1 \leq c(v) \leq d_G(v)$ for every vertex $v \in V(G)$ (sometimes we simply say that G is a capacitated graph if the capacity function is clear from the context). A set $S \subseteq V(G)$ is called a *capacitated dominating set* if there is a *domination mapping* $f: V(G) \setminus S \rightarrow S$ which maps every vertex in $V(G) \setminus S$ to one of its neighbors such that the total number of vertices mapped by f to any vertex $v \in S$ does not exceed its capacity $c(v)$. We say that for a vertex $u \in S$, vertices in the set $f^{-1}(u)$ are *dominated by* u . In the CAPACITATED DOMINATING SET (or CDS) problem, we are given a capacitated graph (G, c) and a positive integer k as an input and the question is whether there exists a capacitated dominating set S for G containing at most k vertices.

We also consider a special variant of CDS problem which we call EXACT SATURATED CAPACITATED DOMINATING SET (or EXACT SATURATED CDS). Given a capacitated dominating set S , a vertex $v \in S$ is called *saturated* if the corresponding domination mapping f maps $c(v)$ vertices to v , that is, $|f^{-1}(v)| = c(v)$. A capacitated dominating set $S \subseteq V(G)$ is called *saturated* if there is a domination mapping f which saturates all vertices of S . In the EXACT SATURATED CAPACITATED DOMINATING SET problem, a capacitated graph (G, c) and a positive integer k are given as an input and the question is whether G has a saturated capacitated dominating set S with exactly k vertices.

A *red-blue capacitated graph* is a pair (G, c) , where G is a bipartite graph with the vertex bipartition R and B and $c: R \rightarrow \mathbb{N}$ is a *capacity* function such that $1 \leq c(v) \leq d_G(v)$ for every vertex $v \in R$. The vertices of the set R are called *red* and the vertices of B are called *blue*. A set $S \subseteq R$ is called a *capacitated dominating set* if there is a *domination mapping* $f: B \rightarrow S$ which maps every vertex in B to one of its neighbors such that the total number of vertices mapped by f to any vertex $v \in S$ does not exceed its capacity $c(v)$. The RED-BLUE CAPACITATED DOMINATING SET (or RED-BLUE CDS) problem for a given red-blue capacitated graph (G, c) and a positive integer k , asks whether there exists a capacitated dominating set S for G containing at most k vertices. A capacitated dominating set $S \subseteq R$ is called *saturated* if there is a domination mapping f which saturates all vertices of S , that is, $|f^{-1}(v)| = c(v)$ for each $v \in S$. The RED-BLUE EXACT SATURATED DOMINATING SET problem (RED-BLUE EXACT SATURATED CDS) takes a red-blue capacitated graph (G, c) and a positive integer k as an input and asks whether there exists a saturated capacitated dominating set with exactly k vertices.

If the input graph G is restricted to be *planar* we call these problems PLANAR CDS, EXACT SATURATED PLANAR CDS, RED-BLUE PLANAR CDS and RED-BLUE EXACT SATURATED PLANAR CDS respectively. The following proposition can be deduced from the constructions presented in [1].

PROPOSITION 2.2. PLANAR CDS, EXACT SATURATED PLANAR CDS, RED-BLUE PLANAR CDS and RED-BLUE EXACT SATURATED PLANAR CDS can not be solved in time $f(t)n^{o(t)}$, where n is the number of vertices and t is the tree-width of the input graph, unless ETH fails.

The basic schema of all the proofs to come is following. The reduction in Proposition 2.2 is from k -CLIQUE to the above mentioned problems and the graphs obtained after the reduction are essentially $k \times k^2$ grid. This immediately implies that the tree-width t of these instances is $O(k)$. Now using the result of Chen et al. [2, 3] about k -CLIQUE, we conclude that PLANAR CDS, EXACT SATURATED PLANAR CDS, RED-BLUE PLANAR CDS and RED-BLUE EXACT SATURATED PLANAR CDS can not be solved in time $f(t)n^{o(t)}$, where t is the tree-width of the input graph, unless ETH fails. Now from Proposition 2.1 we know that the tree-width and the clique-width are linearly related in planar graph. This allows us to conclude that PLANAR CDS, EXACT SATURATED PLANAR CDS, RED-BLUE PLANAR CDS and RED-BLUE EXACT SATURATED PLANAR CDS can not be solved in time $f(t)n^{o(t)}$, where t is the *clique-width* of the input graph, unless ETH fails. This result is the starting point for our reductions to obtain the desired algorithmic lower bounds for MAX-CUT and EDGE DOMINATING SET when parameterized by the clique-width. Furthermore, our reductions are constrained to not blow up the clique-width in the resulting instances. That is, the clique-width of the input instance and the clique-width of the instance obtained after the reduction must be linearly related.

3 Max-Cut and related problems

In this section we consider the MAX-CUT problem and a few other problems that are closely related to it. A *cut set* of a graph G is the set of edges $C \subseteq E(G)$ such that the graph G' with the vertex set $V(G)$ and the edge set C is a bipartite graph. The size of a maximum cut set in G is denoted by $\mathbf{mcut}(G)$. For a partition V_1, V_2 of $V(G)$, the cut set is defined as $C_G(V_1, V_2) = \{uv \in E(G) : u \in V_1, v \in V_2\}$. It is well known that there is one to one correspondence between cut sets and partitions of the vertex set. In the MAX-CUT problem, we are given a graph G and a positive integer k , and the objective is to check whether there

exists a cut set $C \subseteq E(G)$ such that $|C| \geq k$. Our main theorem in this section is following.

THEOREM 3.1. The MAX-CUT problem can not be solved in time $f(t)n^{o(t)}$ unless ETH fails, where n is the number of vertices and t is the clique-width of the input graph. Moreover, the MAX-CUT problem can be solved in time $n^{O(t)}$ if an expression tree of width t is given.

We prove this theorem in two parts. We first show the lower bound and then complement this result with the corresponding upper bound.

3.1 Lower Bounds. To prove our result we give a reduction from the RED-BLUE PLANAR CDS problem to the MAX-CUT problem. The proof is organized as follows: we first give a construction, then prove its correctness and finally argue on the clique-width of the transformed instance.

Construction: Let (G, c) be an instance of RED-BLUE PLANAR CDS with $R = \{u_1, \dots, u_n\}$ being the set of red vertices and $B = \{v_1, \dots, v_r\}$ being the set of blue vertices. We also assume that G has m edges and k is a positive integer. Now we describe the auxiliary gadgets.

Auxiliary gadgets $F(x, y)$ and $F'(x, y)$: Let x, y be two vertices. We construct $F(x, y)$ by joining x and y by $4m + 1$ paths of length two. The graph $F'(x, y)$ is constructed by joining x and y by $4m + 1$ paths of length three. The properties of $F(x, y)$ and $F'(x, y)$ which is required for our proof is summarized in the following simple lemma.

LEMMA 3.1. For a pair of vertices x and y , $\mathbf{mcut}(F(x, y)) = 8m + 2$, $\mathbf{mcut}(F'(x, y)) = 12m + 3$. For any partition V_1, V_2 of the set of vertices in the gadget $F(x, y)$ such that $x \in V_1$ and $y \in V_2$, $|C_{F(x, y)}(V_1, V_2)| \leq \mathbf{mcut}(F(x, y)) - 4m - 1$, and for any partition V_1, V_2 of the set of vertices in the gadget $F'(x, y)$ such that $x, y \in V_1$, $|C_{F'(x, y)}(V_1, V_2)| \leq \mathbf{mcut}(F'(x, y)) - 4m - 1$.

We are going to attach gadgets $F(x, y)$ and $F'(x, y)$ to other part of our construction through the vertices x and y . Notice that we can always assume that the vertices of $V(F(x, y)) \setminus \{x, y\}$ are included in exactly one side of an optimal partition of the vertex set leading to the maximum sized cut. Similarly, we can assume that the vertices of $N_{F'(x, y)}(x)$ ($N_{F'(x, y)}(y)$ respectively) also included in exactly one side of an optimal partition of the vertex set.

Auxiliary gadgets $H_{s,t}(x_1, \dots, x_s, y)$: Let $l = \max\{n, r\}$. We first construct a graph H

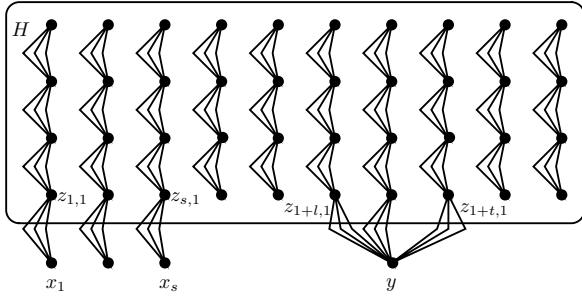


Figure 1: Graph $H_{s,t}(x_1, \dots, x_s, y)$

with the vertex set $\{z_{i,j} : 1 \leq i \leq 2l, 1 \leq j \leq 4m + 1\}$. Any vertices $z_{i,j}$ and $z_{i',j'}$ are joined by an edge for $1 \leq i < i' \leq 2l$. That is, we get a complete $2l$ partite graph with the $2l$ -partition Z_1, \dots, Z_{2l} , where $Z_i = \{z_{i,1}, \dots, z_{i,4m+1}\}$. Then we add graphs $F(z_{i,1}, z_{i,2}), \dots, F(z_{i,4m}, z_{i,4m+1})$ for each $i \in \{1, \dots, 2l\}$. Let $h = l(4m + 1)(4ml + 16m + 1)$. One can easily see that a partition V_1, V_2 corresponding to $\mathbf{mcut}(H)$ is following. Let V_1 consist of Z_1, \dots, Z_l and all the vertices of gadgets $F(z_{i,1}, z_{i,2}), \dots, F(z_{i,4m}, z_{i,4m+1})$, $i \in \{l + 1, \dots, 2l\}$, except those vertices of these gadgets which are contained in Z_{l+1}, \dots, Z_{2l} and let V_2 be the remaining vertices. Using this partition V_1, V_2 corresponding to $\mathbf{mcut}(H)$ and Lemma 3.1, we get the following.

LEMMA 3.2. *For any partition V_1, V_2 of the set of vertices of H such that if V_1 (or V_2) does not contain exactly l sets from Z_1, \dots, Z_{2l} , then $|C_H(V_1, V_2)| \leq \mathbf{mcut}(H) - 4m - 1$. Furthermore, $\mathbf{mcut}(H) = h$.*

Let s and t be two positive integers such that $s, t \leq l$. We construct the graph $H_{s,t}(x_1, \dots, x_s, y)$ from H by adding vertices x_1, \dots, x_s and y , and then joining them with H by the following gadgets, $F(x_1, z_{1,1}), \dots, F(x_s, z_{s,1})$ and $F(y, z_{l+1,1}), \dots, F(y, z_{l+t,1})$ (see Fig 1). Let $h_{s,t} = h + (8m + 2)(s + t)$. Lemmata 3.1 and 3.2 imply the following properties of this graph.

LEMMA 3.3. *The following properties holds for the graph $H_{s,t}(x_1, \dots, x_s, y)$.*

- The $\mathbf{mcut}(H_{s,t}(x_1, \dots, x_s, y)) = h_{s,t}$.
- Let V_1, V_2 be an optimal partition of $V(H_{s,t}(x_1, \dots, x_s, y))$, that is, $\mathbf{mcut}(H_{s,t}(x_1, \dots, x_s, y)) = |C_{H_{s,t}(x_1, \dots, x_s, y)}(V_1, V_2)|$, and $y \in V_1$. Then at most $l - t$ vertices among x_1, \dots, x_s are included in V_1 .

- Furthermore, there is an optimal partition V_1, V_2 such that $y \in V_1$ and for any $0 \leq p \leq l - t$, exactly p vertices among x_1, \dots, x_s are included in V_1 .
- Moreover, for any non optimal partition V_1, V_2 of $V(H_{s,t}(x_1, \dots, x_s, y))$ such that (a) for any gadget $F(z_{i,j}, z_{i,j+1})$, $1 \leq i \leq 2l$ and $1 \leq j \leq 4m$, we have that $V(F(z_{i,j}, z_{i,j+1})) \setminus \{z_{i,j}, z_{i,j+1}\}$ is either contained in V_1 or V_2 ; (b) for any gadget $F(x_i, z_{i,1})$, $1 \leq i \leq s$ and $F(y, z_{l+j,1})$, $1 \leq j \leq t$, we have that $V(F(x_i, z_{i,1})) \setminus \{x_i, z_{i,1}\}$ and $V(F(y, z_{l+j,1})) \setminus \{y, z_{l+j,1}\}$ is either contained in V_1 and V_2 ; then $|C_{H_{s,t}(x_1, \dots, x_s, y)}(V_1, V_2)| \leq \mathbf{mcut}(H_{s,t}(x_1, \dots, x_s, y)) - 4m - 1$.

Final Reduction: Now we describe our reduction. Each edge $e = u_i v_j$ of G is replaced by two vertices a_e and b_e and joined by edges to u_i and v_j . We create two vertices w_1 and w_2 and construct a copy of $F'(w_1, w_2)$. For each vertex $v_j \in B$, a copy of $F(v_j, w_1)$ is created. In the next step, we introduce a copy of $H_{n,l-k}(u_1, \dots, u_n, w_1)$. By G' we denote the graph obtained until now. Finally, for each vertex $u_i \in R$, a copy of $H_{d_G(u_i), l-c(u_i)}(a_{e_1}, \dots, a_{e_{d_G(u_i)}}, w_2)$, where $\{e_1, \dots, e_{d_G(u_i)}\} = E_G(u_i)$ is constructed, and for each vertex $v_j \in B$, a copy of $H_{d_G(v_j), l-1}(a_{e_1}, \dots, a_{e_{d_G(v_j)}}, w_2)$, where $\{e_1, \dots, e_{d_G(v_j)}\} = E_G(v_j)$ is added. Let the resulting graph be Q . Let $\mu = (4m + 1)(2r + 3) + h_{n,l-k} + \sum_{i=1}^n h_{d_G(u_i), l-c(u_i)} + \sum_{j=1}^r h_{d_G(v_j), l-1} + 2(m + r)$.

LEMMA 3.4. *The graph G has a capacitated dominating set of the size at most k if and only if Q has a cut set with at least μ edges.*

Proof. Let S be a capacitated dominating set of the size at most k in G and f be a corresponding domination mapping. We construct a partition V_1, V_2 of the vertex set of Q which corresponds to the cut set of size at least μ as follows. The vertex w_1 is included in V_1 , the vertex w_2 is included in V_2 , all vertices v_1, \dots, v_r are included in V_1 , all the vertices in S are included in V_1 and vertices in $R \setminus S$ are included in V_2 . We also include all the vertices b_e in V_2 . For each edge $e = u_i v_j \in E(G)$ such that $f(v_j) = u_i$, that is, e is being used for domination, the corresponding vertex a_e is included in V_2 and all other vertices a_e , whose corresponding edge is not used for domination are included in V_1 . Finally, we extend our partition to an optimal partition of all gadgets $F(x, y)$, $F'(x, y)$ and $H_{s,t}(x_1, \dots, x_s, y)$ used in the construction of Q . The desired extensions of these gadgets to an optimal partition can be done by applications of Lemmata 3.1, 3.2 and 3.3. By construction of our partitions V_1 and V_2 , the contribution

of the gadgets $F(x, y)$, $F'(x, y)$ and $H_{s,t}(x_1, \dots, x_s, y)$ to the cut $C_Q(V_1, V_2)$ is $\mathbf{mcut}(F(x, y))$, $\mathbf{mcut}(F'(x, y))$ and $\mathbf{mcut}(H_{s,t}(x_1, \dots, x_s, y))$ respectively. Hence, we have already accounted for $(4m + 1)(2r + 3) + h_{n,l-k} + \sum_{i=1}^n h_{d_G(u_i), l-c(u_i)} + \sum_{j=1}^r h_{d_G(v_j), l-1}$ edges in the cut $C_Q(V_1, V_2)$. The remaining $2(m + r)$ edges in the cut $C_Q(V_1, V_2)$ come from the edges incident on the vertices a_e and b_e for some e . Look at an edge e , then we have two cases, either it is an edge used for domination or not. In the first case when $e = uv$ is used for dominating then ua_e , $a_e v$, ub_e and $b_e v$ are part of the cut. In the second case, for an edge when $e = uv$ exactly two of the edges among ua_e , $a_e v$, ub_e and $b_e v$ are part of the cut. In any case for every e at least two edges among ua_e , $a_e v$, ub_e and $b_e v$ are part of the cut and hence edges incident to the vertices a_e and b_e contribute at least $2(m - r) + 4r = 2(m + r)$ to the cut $C_Q(V_1, V_2)$. This completes the forward direction of the proof.

Assume now that Q has a cut set C of size at least μ , and V_1, V_2 be the corresponding partition of the vertex set of Q . Let Q' be the graph obtained by the union of the edge sets of auxiliary gadgets $F(x, y)$, $F'(x, y)$ and $H_{s,t}(x_1, \dots, x_s, y)$. Then there exists a partition A and B of $V(Q')$ such that $C_{Q'}(A, B) = \mu'$, where $\mu' = (4m + 1)(2r + 3) + h_{n,l-k} + \sum_{i=1}^n h_{d_G(u_i), l-c(u_i)} + \sum_{j=1}^r h_{d_G(v_j), l-1}$. Suppose that at least for one of our auxiliary gadgets $F(x, y)$, $F'(x, y)$ or $H_{s,t}(x_1, \dots, x_s, y)$, say $F(x, y)$, the partition V'_1 and V'_2 of $V(F(x, y))$ obtained by restricting the partition V_1 and V_2 to $V(F(x, y))$ is not optimal. That is, $|C_{F(x,y)}(V'_1, V'_2)| < \mathbf{mcut}(F(x, y))$. Then because of Lemmata 3.1, 3.2 and 3.3, $|C| \leq \mu' - (4m + 1) + 4m < \mu$. By choosing a non-optimal partition of auxiliary gadgets we at least lose $4m + 1$ edges while we can only gain $4m$ new edges by cutting $4m$ edges of Q which do not belong to these gadgets. This implies that C restricted to all these gadgets is an optimal cut in Q' . By Lemma 3.1, w_1 and w_2 belong to different sets of the bipartition V_1, V_2 . Assume that $w_1 \in V_1$ and $w_2 \in V_2$. Then Lemma 3.1 implies that $v_1, \dots, v_r \in V_1$. Thus, using Lemma 3.3 we conclude that at most k vertices of the set $R = \{u_1, \dots, u_n\}$ belong to V_1 . We set $S = R \cap V_1$ and prove that S is a capacitated dominating set in G . Notice that by Lemma 3.3, at most one vertex a_e in the neighborhood of each vertex v_j is included in V_2 . Suppose that there is a vertex v_j such that its neighborhood in Q has no vertices $a_e \in V_2$. Then $|C| \leq \mu' + 2m + 2(r - 1) < \mu$, a contradiction. So, for each vertex v_j , there is an edge $e = u_i v_j$ such that $a_e \in V_2$. Now we argue that $u_i \in S$. This follows from the fact that if $u_i \notin S$ then $u_i \in V_2$ and hence $|C| \leq \mu' + 2m + 2r - 2 < \mu$. We define the domination mapping $f(v_j) = u_i$. Since by Lemma 3.3, at most $c(u_i)$

vertices in the set $N_Q(u_i) \cap \{a_e \mid e \in E(G)\}$ are included in V_2 , $|f^{-1}(u_i)| \leq c(u_i)$. This concludes the proof.

Now we upper bound the clique-width of Q by a linear function of the tree-width of G .

LEMMA 3.5. *Let $t \geq 2$. If $\mathbf{tw}(G) \leq t$ then $\mathbf{cwd}(Q) \leq 96t + 106$.*

Proof. Since $t \geq 2$, $\mathbf{tw}(I(G)) = \mathbf{tw}(G)$, and by Proposition 2.1 we have that $\mathbf{cwd}(I(G)) \leq c = 12t + 12$. We construct an expression tree for Q in two stages and use $8c + 10$ labels. At the first stage we construct an expression tree for G' using $4c + 10$ labels, and at the second stage we describe how it can be modified to get an expression tree for Q using $4c$ additional labels.

Construction of an expression tree for G' : Suppose that the expression tree for $I(G)$ uses c labels $\{\alpha_1, \dots, \alpha_c\}$. To construct the expression tree for G' we use the following additional labels.

- Labels β_1, \dots, β_c for the vertices v_1, \dots, v_r .
- Labels $\gamma_1, \dots, \gamma_c$ for the vertices $\{a_e \mid e \in E(G)\}$.
- Labels $\delta_1, \dots, \delta_c$ for the vertices $\{b_e \mid e \in E(G)\}$.
- Labels ζ_1, ζ_2 for the vertices w_1, w_2 .
- Label η for the vertices $z_{i,j}$ in $H_{n,l-k}(u_1, \dots, u_n, w_1)$.
- Working labels $\lambda_1, \lambda_2, \lambda_3$ and $\xi_1, \xi_2, \xi_3, \xi_4$.

We construct the required expression tree for G' by going over the expression tree for $I(G)$ and making necessary changes in it. When a vertex $u_i \in R$ labeled by α_p is introduced, we perform following set of operations. We first introduce the vertex u_i labeled α_p and a vertex (which is essentially $z_{i,1}$) labeled by ξ_3 . Then $4m + 1$ vertices labeled with ξ_2 are introduced and joined with vertices labeled α_p and ξ_3 . Then the vertices labeled ξ_2 are relabeled λ_1 . Now we repeat the following operations $4m$ times: (a) introduce a vertex labeled ξ_1 and $4m + 1$ vertices labeled ξ_2 ; (b) join vertices labeled ξ_2 with vertices labeled ξ_1 and ξ_3 ; (c) relabel vertices labeled ξ_2 by λ_1 , the vertex labeled ξ_3 by η , and the vertex labeled ξ_1 by ξ_3 ; (d) finally, the vertex labeled ξ_3 is relabeled η . We omit the union operations from our descriptions here and henceforth in any similar descriptions and assume that if some vertex is introduced then union is always performed.

When a vertex $x \in V(I(G))$ which corresponds to an edge $e \in E(G)$ labeled α_p is introduced, we introduce the vertices a_e and b_e and label it with γ_p and δ_p , respectively. Now we move toward introduction of vertices from the set B . When a vertex $v_j \in B$ labeled α_p is introduced, we introduce the vertex v_j with label

β_p . Then $4m + 1$ vertices labeled ξ_1 are introduced, joined with the vertex labeled β_p and relabeled λ_2 . We are labeling these vertices with λ_2 to finally join them with the vertex w_1 , when it gets introduced.

For each union operation in the expression tree for $I(G)$, we do as follows. If both graphs contain vertices labeled η , then (a) vertices labeled η in one of the graphs are relabeled ξ_1 ; (b) we perform the union operation; (c) the vertices labeled η and ξ_1 are joined; and (d) the vertices labeled ξ_1 are relabeled η . If only one graph contains vertices labeled η then we just do the union operation.

If in the expression tree of $I(G)$, we have join operation between two labels say α_p and α_q then we simulate this by applying join operations between following: (i) α_p and γ_q ; (ii) α_p and δ_q ; (iii) β_p and γ_q ; (iv) β_p and δ_q ; (v) α_q and γ_p ; (vi) α_q and δ_p ; (vii) β_q and γ_p ; and (viii) β_q and δ_p .

Finally, the relabel operation in the expression tree of G , that is, relabel α_p to α_q is replaced by following relabeling process: (a) α_p to α_q ; (b) β_p to β_q ; (c) γ_p to γ_q ; and (d) δ_p to δ_q .

After we have completed the scanning of the expression tree for $I(G)$, the vertices w_1 and w_2 labeled by ζ_1 and ζ_2 respectively, are introduced. Then we repeat the following operations $4m + 1$ times: (a) introduce two vertices labeled ξ_1 and ξ_2 ; (b) join vertices labeled ζ_1 and ξ_1 , ξ_1 and ξ_2 , ξ_2 and ζ_2 ; (c) and relabel vertices labeled ζ_1 and ζ_2 by λ_1 . After that the vertex w_1 labeled ζ_1 is joined with vertices labeled λ_2 .

Now we show how to complete the construction of $H_{n,l-k}(u_1, \dots, u_n, w_1)$. We start of by repeating the following $l - n + k$ times. A vertex labeled ξ_3 is introduced. Now we repeat the following operations $4m$ times: (a) introduce a vertex labeled ξ_1 and $4m + 1$ vertices labeled ξ_2 ; (b) join vertices labeled ξ_2 and vertices labeled with ξ_1 and ξ_3 ; (c) relabel vertices labeled ξ_2 by λ_1 , the vertex labeled ξ_3 by ξ_4 , and the vertex labeled ξ_1 by ξ_3 . Finally, the vertex labeled ξ_3 is relabeled ξ_4 , the vertices labeled ξ_4 are joined with vertices labeled η and then relabeled by η .

Now, we do the following $l - k$ times. A vertex labeled ξ_3 and $4m + 1$ vertices labeled ξ_1 are introduced. The vertices labeled ξ_1 are joined with vertices labeled ζ_1 and ξ_2 , and relabeled λ_1 . After this we repeat the following operations $4m$ times: (a) introduce a vertex labeled ξ_1 and $4m + 1$ vertices labeled ξ_2 ; (b) join vertices labeled ξ_2 and vertices labeled ξ_1 and ξ_3 ; (c) relabel vertices labeled ξ_2 by λ_1 , the vertex labeled ξ_3 by ξ_4 , and the vertex labeled ξ_1 by ξ_3 . Finally, the vertex labeled ξ_3 is relabeled ξ_4 , the vertices labeled ξ_4 are joined with vertices labeled η and then relabeled η .

Construction of an expression tree for Q : We

now show how to modify the expression tree for G' to add gadgets $H_{d_G(u_i), l-c(u_i)}(a_{e_1}, \dots, a_{e_{d_G(u_i)}}, w_2)$ where $\{e_1, \dots, e_{d_G(u_i)}\} = E_G(u_i)$ for $u_i \in R$ using $2c$ additional labels. The gadgets $H_{d_G(v_j), l-1}(a_{e_1}, \dots, a_{e_{d_G(v_j)}}, w_2)$ where $\{e_1, \dots, e_{d_G(v_j)}\} = E_G(v_j)$ for vertices $v_j \in B$ can be added in the same way by using additional $2c$ labels.

To add gadgets $H_{d_G(u_i), l-c(u_i)}(a_{e_1}, \dots, a_{e_{d_G(u_i)}}, w_2)$ where $\{e_1, \dots, e_{d_G(u_i)}\} = E_G(u_i)$ for $u_i \in R$, we use following additional labels $\alpha'_1, \dots, \alpha'_c$ and $\beta'_1, \dots, \beta'_c$. We scan the expression tree for G' and iteratively change it for each u_i , $i \in \{1, \dots, n\}$ to add the corresponding gadgets. Let $E_G(u_i) = \{e_1, \dots, e_{d_G(u_i)}\}$. Let A denote the set of vertices $\{a_{e_1}, \dots, a_{e_{d_G(u_i)}}\}$ and let $U = A \cup \{u_i\}$. Let X be a node of the expression tree for G' and G'_X be the subgraph of G' corresponding to this node. If $V(G'_X) \cap U \neq \emptyset$ but $G'[U]$ is not a subgraph of G'_X then we can observe the following:

- If $u_i \in V[G'_X]$ then u_i is labeled by a label which is different from labels of other vertices of G'_X .
- If $u_i \notin V[G'_X]$ then vertices of $U \cap V(G'_X)$ are labeled by labels which are different from labels of other vertices of G'_X .

We use these observations to construct the graph $H_{d_G(u_i), l-c(u_i)}(a_{e_1}, \dots, a_{e_{d_G(u_i)}}, w_2)$ in such a way that all vertices $z_{i,j}$ of this gadget constructed for the node X are labeled by same labels, if $u_i \in V[G'_X]$ and this vertex is labeled by λ_p then we label it by α'_p , and if $u_i \notin V[G'_X]$ and the labels $\gamma_{p_1}, \dots, \gamma_{p_h}$ are used for vertices $U \cap V(G'_X)$ then all vertices $z_{i,j}$ are labeled by one of the labels $\gamma'_{p_1}, \dots, \gamma'_{p_h}$. The construction of the gadget $H_{d_G(u_i), l-c(u_i)}(a_{e_1}, \dots, a_{e_{d_G(u_i)}}, w_2)$ is completed when after some union operation all vertices of U are included in the graph G'_X . We finally relabel vertices of G'_X by the label λ_1 which is not used for any join operation.

When a vertex $u_i \in R$ labeled by α_p is introduced, we perform following set of operations. First, we introduce the vertex u_i labeled by α_p . Then we repeat the following operations $l + c(u_i) - d_G(u_i)$ times. A vertex labeled ξ_3 is introduced and then the following operations are repeated $4m$ times: (a) introduce a vertex labeled ξ_1 and $4m + 1$ vertices labeled ξ_2 , (b) join vertices labeled ξ_2 and vertices labeled ξ_1 and ξ_3 , (c) relabel vertices labeled ξ_2 by λ_1 , the vertex labeled ξ_3 by ξ_4 , and the vertex labeled ξ_1 by ξ_3 . Finally, the vertex labeled ξ_3 is relabeled ξ_4 , the vertices labeled ξ_4 are joined with vertices labeled α'_p (if they exist) and then relabeled α'_p . Next we perform the following $l - c(u_i)$ times: (a) a vertex labeled ξ_3 and $4m + 1$ vertices labeled ξ_1 are introduced; (b) the vertices labeled ξ_1 are joined

with vertices labeled ξ_2 , and relabeled λ_3 . Then we repeat the following operations $4m$ times: (i) introduce a vertex labeled ξ_1 and $4m + 1$ vertices labeled ξ_2 , (ii) join vertices labeled ξ_2 and vertices labeled ξ_1 and ξ_3 , (iii) relabel vertices labeled ξ_2 by λ_1 , the vertex labeled ξ_3 by ξ_4 , and the vertex labeled ξ_1 by ξ_3 . Finally, the vertex labeled ξ_3 is relabeled ξ_4 , the vertices labeled ξ_4 are joined with vertices labeled α'_p and then relabeled α'_p .

When a vertex a_e such that $u_i a_e \in E(Q)$ labeled by γ_q is introduced, we perform following set of operations. First, we introduce the vertex a_e labeled γ_q and a vertex labeled by ξ_3 . Then $4m + 1$ vertices labeled ξ_2 are introduced and joined with vertices labeled γ_p and ξ_3 . Then the vertices labeled ξ_2 are relabeled λ_1 . Now we repeat the following operations $4m$ times: (a) introduce a vertex labeled ξ_1 and $4m + 1$ vertices labeled ξ_2 , (b) join vertices labeled ξ_2 and vertices labeled ξ_1 and ξ_3 , (c) relabel vertices labeled ξ_2 by λ_1 , the vertex labeled ξ_3 by γ'_q , and the vertex labeled ξ_1 by ξ_3 . Finally, the vertex labeled ξ_3 is relabeled γ'_q .

Having dealt with introduction nodes, next we consider union operations. Let X be a union node of the expression tree for G' . Denote by X and Y two children of this node and let G'_Y and G'_Z be subgraphs of G' which correspond to these nodes. If one of these graphs do not contain vertices of U then we just perform the same operation. Otherwise we have two cases.

- $u_i \in V(G'_Y) \cup V(G'_Z)$. Suppose that $u_i \in V(G'_Y)$ and is labeled α_p . Then vertices $z_{i,j}$ of $H_{d_G(u_i), l-c(u_i)}(a_{e_1}, \dots, a_{e_{d_G(u_i)}}), w_2)$ which are constructed for the node Y are labeled α'_p . The graph G'_Z includes vertices of A labeled by some labels $\gamma_{p_1}, \dots, \gamma_{p_h}$, and all vertices $z_{i,j}$ of $H_{d_G(u_i), l-c(u_i)}(a_{e_1}, \dots, a_{e_{d_G(u_i)}}), w_2)$ corresponding to this node are labeled by same label γ'_{p_j} . We do the union operation as before, then join the vertices labeled α'_p and γ'_{p_j} , relabel the vertices labeled γ'_{p_j} by α'_p . If the graph G'_X corresponding to X contains all vertices of U , then the vertices labeled α'_p are relabeled λ_1 .
- $u_i \notin V(G'_Y) \cup V(G'_Z)$. Then G'_Y includes vertices of A that are labeled by $\gamma_{p_1}, \dots, \gamma_{p_h}$, and all vertices $z_{i,j}$ of $H_{d_G(u_i), l-c(u_i)}(a_{e_1}, \dots, a_{e_{d_G(u_i)}}), w_2)$ corresponding to this node and are labeled by same label γ'_{p_i} . Similarly, G'_Z includes vertices of A labeled by some labels $\gamma_{q_1}, \dots, \gamma_{q_f}$, and all the vertices $z_{i,j}$ of $H_{d_G(u_i), l-c(u_i)}(a_{e_1}, \dots, a_{e_{d_G(u_i)}}), w_2)$ corresponding to this node and are labeled by same label γ'_{q_j} . We relabel vertices labeled γ'_{p_i} by ξ_2 in the first graph, then perform the union operation, join the vertices labeled ξ_2 and γ'_{q_j} , relabel the

vertices labeled ξ_2 by γ'_{q_j} .

The join operations in the expression tree for G' are done in the new tree in exactly the same way. The relabel operation in the expression tree of G' , that is, relabel α_p to α_q and relabel γ_p to γ_q , are replaced by relabel α_p to α_q , α'_p to α'_q , and γ_p to γ_q , γ'_p to γ'_q , respectively.

When we have completed the scan of the expression tree for G' , the only thing which remains is to join vertices labeled λ_3 and the vertex labeled ζ_2 (the vertex w_2).

To conclude the first part of the proof of the Theorem 3.1, we observe that the number of vertices of Q is polynomial in $n + r$, and therefore if we could solve the MAX-CUT in time $f(t)|V(Q)|^{o(t)}$ where $t = \mathbf{c wd}(Q)$ then the RED-BLUE PLANAR CDS could be solved in time $f(t)|V(G)|^{o(t')} = f(t)|V(G)|^{o(t)}$ where $t' = \mathbf{t w}(G) = O(\mathbf{c wd}(Q)) = O(t)$.

3.2 Algorithmic upper bounds for Max-Cut.

Now we outline an algorithm for solving MAX-CUT in time $n^{O(t)}$ on graphs of clique-width at most t . The algorithm is based on dynamic programming over the expression tree of the input graph. We first describe what we store in the tables corresponding to the nodes in the expression tree.

Let G be a graph with n vertices and m edges, and let T be an expression tree for G of width t . For a node X of T , denote by G_X the t -graph associated with this node, and let $U_1(X), \dots, U_t(X)$ be the sets of vertices of G_X labeled $1, \dots, t$ respectively. The table of data for the node X stores vectors (s_1, \dots, s_t, r) of integers such that $0 \leq s_i \leq |U_i(X)|$ for $1 \leq i \leq t$, and $0 \leq r \leq |E(G_X)|$, for which there is a partition V_1, V_2 of $V(G_X)$ such that $|V_1 \cap U_i(G_X)| = s_i$ and $|C_{G_X}(V_1, V_2)| \geq r$. Notice that this table contains at most $(n+1)^t \cdot m$ vectors. If X is the root node of T (that is, $G = G_X$) then $\mathbf{mcut}(G)$ is equal to the maximum value of r for which the table for X contains an entry with this value.

Now we give the details of how we make our tables and how do we update it.

Introduce Node: Tables for introduce nodes of T are constructed in a straightforward manner.

Relabel Node: Suppose that X is a relabel node $\rho_{i \rightarrow j}$, and let Y be the child of X . Then the table for X contains a vector (s_1, \dots, s_t, r) if and only if $s_i = 0$ and the table for Y contains the entry (s'_1, \dots, s'_t, r) such that $s'_p = s_p$ for $1 \leq p \leq t$, $t \neq i, j$, and $s_j = s'_i + s'_j$.

Union Node: Let X be a union node with children Y and Z . In this case the table for X contains a vector (s_1, \dots, s_t, r) if and only if the tables for Y and Z have vectors (s'_1, \dots, s'_t, r') and $(s''_1, \dots, s''_t, r'')$ respectively, such that $s'_i + s''_i = s_i$ for $1 \leq i \leq t$, and $r' + r'' \geq r$.

Join Node: Finally, suppose that X is a join node $\eta_{i,j}$ with the child Y . It can be noted that the table for X has a vector (s_1, \dots, s_t, r) if and only if the table for Y includes a vector (s_1, \dots, s_t, r') such that $r' + s_i(|U_j(Y)| - s_j) + s_j(|U_i(Y)| - s_i) \geq r$.

Correctness of the algorithm follows from the description of the procedure, and it runs in time $O(t^{O(1)}n^{2t+O(1)})$. This proves that MAX-CUT can be solved in time $n^{O(t)}$ on graphs of clique-width at most t .

3.3 Bipartization by Edge Removal and Maximum (Minimum) Bisection. Theorem 3.1 have several interesting corollaries for similar problems like BIPARTIZATION BY EDGE REMOVAL and MAXIMUM (MINIMUM) BISECTION.

In the BIPARTIZATION BY EDGE REMOVAL problem, we are given a graph G and a positive integer k , and the question is whether there is a set of edges X such that $|X| \leq k$ and the graph G' with the vertex set $V(G)$ and the edge set $E(G) \setminus X$ is bipartite. Since this problem is dual to the MAXIMUM CUT problem, we immediately have the following corollary.

COROLLARY 3.1. *The BIPARTIZATION BY EDGE REMOVAL problem can not be solved in time $f(t)n^{o(t)}$ unless ETH fails, where n is the number of vertices and t is the clique-width of the input graph. Moreover, the BIPARTIZATION BY EDGE REMOVAL problem can be solved in time $n^{O(t)}$ if an expression tree of width t is given.*

In the MAXIMUM (MINIMUM) BISECTION problem, we are given a graph G with an even number of vertices and a positive integer k , and the objective is to check whether there is a partition of $V(G)$ into two sets V_1 and V_2 of equal size such that $|C_G(V_1, V_2)| \geq k$ ($|C_G(V_1, V_2)| \leq k$).

COROLLARY 3.2. *The MAXIMUM (MINIMUM) BISECTION problem can not be solved in time $f(t)n^{o(t)}$ unless ETH fails, where n is the number of vertices and t is the clique-width of the input graph. Moreover, the MAXIMUM (MINIMUM) BISECTION problem can be solved in time $n^{O(t)}$ if an expression tree of width t is given.*

Proof. The algorithmic upper bound for the MAXIMUM BISECTION follows from the observation that the algorithm for the MAX-CUT described in 3.2 can be modified for this problem. The lower bound can be obtained

from the fact that the MAX-CUT problem for a graph G can be reduced to the MAXIMUM BISECTION by adding $|V(G)|$ isolated vertices. The claim about the MINIMUM BISECTION follows from the observation that the MAXIMUM BISECTION problem for a graph G can be reduced to the MINIMUM BISECTION problem in the complement \overline{G} , and the fact that $\text{cwd}(\overline{G}) \leq 2\text{cwd}(G)$ (see [29, 6]).

4 Edge Dominating Set

In this section, we consider the EDGE DOMINATING SET problem. In the EDGE DOMINATING SET problem, we are given a graph G and a positive integer k , and the objective is to determine whether there is a set of edges $X \subseteq E(G)$ such that $|X| \leq k$ and every edge of G is either included in X , or it is adjacent to at least one edge of X (which *dominates* it). The set X is called an *edge dominating set* of G . We prove the following result for EDGE DOMINATING SET.

THEOREM 4.1. *The EDGE DOMINATING SET problem can not be solved in time $f(t)n^{o(t)}$ unless ETH fails, where n is the number of vertices and t is the clique-width of the input graph. Moreover, the EDGE DOMINATING SET problem can be solved in time $n^{O(t)}$ if an expression tree of width t is given.*

The proof of Theorem 4.1 is omitted here and will appear in the journal version of our paper.

5 Conclusion and Further Directions

In this paper, we obtained the first asymptotically tight bounds for problems parameterized by the clique-width of the input graph. In particular, we showed that MAX-CUT and EDGE DOMINATING SET cannot be solved in time $f(t)n^{o(t)}$, unless ETH collapses; while there do exist algorithms with running time $n^{O(t)}$ for both these problems, where t is the clique-width of the input graph. We believe that our results opens a new direction in the algorithmic study around clique-width. Our reduction to obtain a tight lower bound for MAX-CUT is also an FPT-reduction, thus resolving an open problem about the parameterized complexity of MAX-CUT.

We conclude with an open problem related to HAMILTONIAN CYCLE. In the HAMILTONIAN CYCLE problem, we are given a graph G and the objective is to check whether there exists a cycle passing through every vertex of G . Similar to MAX-CUT and EDGE DOMINATING SET we can obtain the following algorithmic lower bound for the HAMILTONIAN CYCLE problem when parameterized by the clique-width of the input graph.

THEOREM 5.1. *The HAMILTONIAN CYCLE problem can not be solved in time $f(t)n^{o(t)}$, where n is the number of vertices and t is the clique-width of the input graph, unless ETH fails.*

However, all the algorithms we know for HAMILTONIAN CYCLE run in time $n^{O(t^2)}$ if an expression tree of width t is given. We leave it open to find either an improved lower bound or an improved upper bound for the HAMILTONIAN CYCLE problem.

References

- [1] H. BODLAENDER, D. LOKSHTANOV, AND E. PENNINKX, *Planar capacitated dominating set is $W[1]$ -hard*, in IWPEC'09, 2009.
- [2] J. CHEN, X. HUANG, I. A. KANJ, AND G. XIA, *On the computational hardness based on linear FPT-reductions*, *J. Comb. Optim.*, 11 (2006), pp. 231–247.
- [3] ———, *Strong computational lower bounds via parameterized complexity*, *J. Comput. Syst. Sci.*, 72 (2006), pp. 1346–1367.
- [4] D. G. CORNEIL, M. HABIB, J.-M. LANLIGNEL, B. A. REED, AND U. ROTICS, *Polynomial time recognition of clique-width ≤ 3 graphs (extended abstract)*, in LATIN'00, vol. 1776 of LNCS, Springer, 2000, pp. 126–134.
- [5] B. COURCELLE, J. A. MAKOWSKY, AND U. ROTICS, *Linear time solvable optimization problems on graphs of bounded clique-width*, *Theory Comput. Syst.*, 33 (2000), pp. 125–150.
- [6] B. COURCELLE AND S. OLARIU, *Upper bounds to the clique width of graphs*, *Discrete Appl. Math.*, 101 (2000), pp. 77–114.
- [7] R. G. DOWNEY, V. ESTIVILL-CASTRO, M. R. FELLOWS, E. PRIETO, AND F. A. ROSAMOND, *Cutting up is hard to do: the parameterized complexity of k -cut and related problems*, *Electr. Notes Theor. Comput. Sci.*, 78 (2003).
- [8] R. G. DOWNEY AND M. R. FELLOWS, *Parameterized complexity*, *Monographs in Computer Science*, Springer-Verlag, New York, 1999.
- [9] W. ESPELAGE, F. GURSKI, AND E. WANKE, *How to solve NP-hard graph problems on clique-width bounded graphs in polynomial time*, in WG'01, vol. 2204 of LNCS, Springer, 2001, pp. 117–128.
- [10] M. R. FELLOWS, F. A. ROSAMOND, U. ROTICS, AND S. SZEIDER, *Clique-width minimization is NP-hard (extended abstract)*, in STOC'06, ACM, 2006, pp. 354–362.
- [11] J. FLUM AND M. GROHE, *Parameterized complexity theory*, *Texts in Theoretical Computer Science. An EATCS Series*, Springer-Verlag, Berlin, 2006.
- [12] F. V. FOMIN, P. A. GOLOVACH, D. LOKSHTANOV, AND S. SAURABH, *Clique-width: on the price of generality*, in SODA, 2009, pp. 825–834.
- [13] F. V. FOMIN, S.-IL OUM, AND D. M. THILIKOS, *Rank-width and tree-width of H -minor-free graphs*, *Manuscript*, (2008).
- [14] M. U. GERBER AND D. KOBLEK, *Algorithms for vertex-partitioning problems on graphs with fixed clique-width*, *Theoret. Comput. Sci.*, 299 (2003), pp. 719–734.
- [15] O. GIMÉNEZ, P. HLINEŃY, AND M. NOY, *Computing the tutte polynomial on graphs of bounded clique-width*, *SIAM J. Discret. Math.*, 20 (2006).
- [16] B. GODLIN, T. KOTEK, AND J. A. MAKOWSKY, *Evaluations of graph polynomials*, in WG, vol. 5344 of *Lecture Notes in Computer Science*, 2008, pp. 183–194.
- [17] F. GURSKI AND E. WANKE, *The tree-width of clique-width bounded graphs without $k_{n,n}$* , in WG'00, LNCS, Springer, 2000, pp. 196–205.
- [18] P. HLINEŃY AND S. IL OUM, *Finding branch-decompositions and rank-decompositions*, *SIAM J. Comput.*, 38 (2008), pp. 1012–1032.
- [19] P. HLINEŃY, S.-IL OUM, D. SEESE, AND G. GOTTLÖB, *Width parameters beyond tree-width and their applications*, *Comput. J.*, 51 (2008), pp. 326–362.
- [20] R. IMPAGLIAZZO, R. PATURI, AND F. ZANE, *Which problems have strongly exponential complexity?*, *J. Comput. Syst. Sci.*, 63 (2001), pp. 512–530.
- [21] D. KOBLEK AND U. ROTICS, *Polynomial algorithms for partitioning problems on graphs with fixed clique-width (extended abstract)*, in SODA'01, ACM-SIAM, 2001, pp. 468–476.
- [22] ———, *Edge dominating set and colorings on graphs with fixed clique-width*, *Discrete Appl. Math.*, 126 (2003), pp. 197–221.
- [23] J. MAKOWSKY, U. ROTICS, I. AVERBOUCH, AND B. GODLIN, *Computing graph polynomials on graphs of bounded clique-width*, in WG'06, LNCS, Springer, 2006, pp. 191–204.
- [24] D. MARX, *Can you beat treewidth?*, in FOCS, 2007, pp. 169–179.
- [25] ———, *On the optimality of planar and geometric approximation schemes*, in FOCS, 2007, pp. 338–348.
- [26] S.-IL OUM AND P. SEYMOUR, *Approximating clique-width and branch-width*, *J. Combin. Theory Ser. B*, 96 (2006), pp. 514–528.
- [27] M. RAO, *MSOL partitioning problems on graphs of bounded treewidth and clique-width*, *Theoret. Comput. Sci.*, 377 (2007), pp. 260–267.
- [28] K. SUCHAN AND I. TODINCA, *On powers of graphs of bounded NLC-width (clique-width)*, *Discrete Appl. Math.*, 155 (2007), pp. 1885–1893.
- [29] E. WANKE, *k -NLC graphs and polynomial algorithms*, *Discrete Appl. Math.*, 54 (1994), pp. 251–266.