Capacitated Domination and Covering: A Parameterized Perspective

Michael Dom[*] Daniel Lokshtanov[†] Saket Saurabh [†] Yngve Villanger [†]

Abstract

Capacitated versions of Dominating Set and Vertex Cover have been studied intensively in terms of polynomial time approximation algorithms. Although the problems Dominating Set and Vertex Cover have been subjected to considerable scrutiny in the parameterized complexity world, this is not true for the capacitated versions. Here we make an attempt to understand the behavior of the problems Capacitated Dominating Set and Capacitated Vertex Cover from the perspective of parameterized complexity.

The original versions of these problems, Vertex Cover and Dominating Set, are known to be fixed parameter tractable when parameterized by a structure of the graph called the *treewidth* (tw). In this paper we show that the capacitated versions of these problems behave differently. Our results are:

- Capacitated Dominating Set is W[1]-hard when parameterized by treewidth. In fact, Capacitated Dominating Set is W[1]-hard when parameterized by both treewidth and solution size k of the capacitated dominating set.
- Capacitated Vertex Cover is W[1]-hard when parameterized by treewidth.
- Capacitated Vertex Cover can be solved in time $2^{O(\mathrm{tw}\log k)}n^{O(1)}$ where tw is the treewidth of the input graph and k is the solution size. As a corollary, we show that the weighted version of Capacitated Vertex Cover in general graphs can be solved in time $2^{O(k\log k)}n^{O(1)}$. This improves the earlier algorithm of Guo et al. [15] running in time $O(1.2^{k^2}+n^2)$.

We would also like to point out that our W[1]-hardness result for Capacitated Vertex Cover, when parameterized by treewidth, makes it (to the best of our knowledge) the first known "subset problem" which has turned out to be fixed parameter tractable when parameterized by solution size but W[1]-hard when parameterized by treewidth.

1 Introduction

Dominating Set (or more generally Set Cover) and Vertex Cover are problems representative for domination and covering, respectively. Given a graph G and an integer k, Vertex Cover asks for a size-k set of vertices that cover all edges of the graph, while Dominating Set asks for a size-k set of vertices such that every vertex in the graph either belongs to this set or has a neighbor which does. These fundamental problems in algorithms and complexity have been studied extensively and find applications in various domains [3, 4, 5, 8, 9, 12, 13, 15, 16, 18, 22].

[*]Institut für Informatik, Friedrich-Schiller-Universität Jena, Ernst-Abbe-Platz 2, D-07743 Jena, Germany. Email: dom@minet.uni-jena.de

[†]Department of Informatics, University of Bergen, POB 7803, 5020 Bergen, Norway. Email: {daniello,saket,yngvev}@ii.uib.no

VERTEX COVER and DOMINATING SET have a special place in parameterized complexity [7, 10, 21]. VERTEX COVER was one of the earliest problems that was shown to be fixed parameter tractable (FPT) [7]. On the other hand, DOMINATING SET turned out to be intractable in the realm of parameterized complexity—specifically, it was shown to be W[2]-complete [7]. VERTEX COVER has been put to intense scrutiny, and many papers have been written on the problem. After a long race, the currently best algorithm for VERTEX COVER runs in time $O(1.2738^k + kn))$ [4]. VERTEX COVER has also been used as a testbed for developing new techniques for showing that a problem is FPT [7, 10, 21]. Though DOMINATING SET is a fundamentally hard problem in the parameterized W-hierarchy, it has been used as a benchmark problem for developing *sub-exponential time* parameterized algorithms [1, 6, 11] and also for obtaining a *linear kernels* in planar graphs [2, 14, 10, 21], and more generally, in graphs that exclude a fixed graph H as a minor.

Different applications of VERTEX COVER and DOMINATING SET (or SET COVER) have initiated studies of different generalizations and variations of these problems. These include CONNECTED VERTEX COVER, CONNECTED DOMINATING SET, PARTIAL VERTEX COVER, PARTIAL SET COVER , CAPACITATED VERTEX COVER and CAPACITATED DOMINATING SET, to name a few. All these problems have been investigated extensively and are well understood in the context of polynomial time approximation [5, 12, 13, 16]. However, these problems hold a lot of promise and remain hitherto unexplored in the light of parameterized complexity; with exceptions that are few and far between [3, 15, 19, 22, 23].

Problems Considered: Here we consider two problems, CAPACITATED VERTEX COVER (CVC) and CAPACITATED DOMINATING SET (CDS). To define these problems, we need to introduce the notions of *capacitated graphs*, *vertex covers*, and *dominating sets.* A capacitated graph is a graph $G = (V, E)$ together with a capacity function $c : V \rightarrow \mathbb{N}$ such that $1 \leq c(v) \leq d(v)$, where $d(v)$ is the degree of the vertex v. Now let $G = (V, E)$ be a capacitated graph, C be a vertex cover of G and D be a dominating set of G.

Definition 1 *We call $C \subseteq V$ a capacitated vertex cover if there exists a mapping $f : E \rightarrow C$ which maps every edge in E to one of its two endpoints such that the total number of edges mapped by f to any vertex $v \in C$ does not exceed $c(v)$.*

Definition 2 *We call $D \subseteq V$ a capacitated dominating set if there exists a mapping $f : (V \setminus D) \rightarrow D$ which maps every vertex in $(V \setminus D)$ to one of its neighbors such that the total number of vertices mapped by f to any vertex $v \in D$ does not exceed $c(v)$.*

Now we are ready to define CAPACITATED VERTEX COVER and CAPACITATED DOMINATING SET.

> CAPACITATED VERTEX COVER (CVC): Given a capacitated graph $G = (V, E)$ and a positive integer k, determine whether there exists a capacitated vertex cover C for G containing at most k vertices.
>
> CAPACITATED DOMINATING SET (CDS): Given a capacitated graph $G = (V, E)$ and a positive integer k, determine whether there exists a capacitated dominating set D for G containing at most k vertices.

Our Results: To describe our results we first need to define the *treewidth (*tw*)* of a graph.

Let $V(U)$ be the set of vertices of a graph U. A tree decomposition of an (undirected) graph $G = (V, E)$ is a pair (X, U) where U is a tree whose vertices we will call *nodes* and $X = \{X_i \mid i \in V(U)\}$ is a collection of subsets of V such that

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

The *width* of a tree decomposition $(\{X_i | i \in V(U)\}, U)$ equals $\max_{i \in V(U)}\{|X_i| - 1\}$. The *treewidth* of a graph G is the minimum width over all tree decompositions of G.

There is a tendency to think that most combinatorial problems, especially "subset problems", are tractable for graphs of bounded treewidth (tw) when parameterized by tw. In fact, the non-capacitated versions of the problems considered here, namely Vertex Cover and Dominating Set, are known to be fixed parameter tractable when parameterized by the treewidth of the input graph. The algorithms for Vertex Cover and Dominating Set run in time $O(2^{\text{tw}}n)$ [21] and time $O(4^{\text{tw}}n)$ [1], respectively. In contrast, the capacitated versions of these problems behave differently. More precisely, we show the following:

- Capacitated Dominating Set is W[1]-hard when parameterized by treewidth. In fact, CDS is W[1]-hard when parameterized by both treewidth and solution size k of the capacitated dominating set.
- Capacitated Vertex Cover is W[1]-hard when parameterized by treewidth.
- Capacitated Vertex Cover can be solved in time $2^{O(\text{tw} \log k)} n^{O(1)}$ where tw is the treewidth of the input graph and k is the solution size. As a corollary of the last result we obtain an improved algorithm for the weighted version of Capacitated Vertex Cover in general graphs. Here, every vertex of the input graph has, in addition to the capacity, a weight, and the question is if there is a capacitated vertex cover whose weight is at most k. Our algorithm running in time $O(2^{O(k \log k)} n^{O(1)})$ improves the earlier algorithm of Guo et al. [15] running in time $O(1.2^{k^2} + n^2)$.

The so-called "subset problems" are known to go either way, that is, FPT or W[i]-hard ($i \geq 1$) when parameterized by solution size. However, when parameterized by treewidth they have invariably been FPT. Examples favoring this claim include, but are not limited to, Independent Set, Dominating Set, Partial Vertex Cover. Contrary to these observed patterns, our hardness result for CVC when parameterized by treewidth makes it possibly the first known "subset problem" which has turned out to be FPT when parameterized by solution size, but W[1]-hard when parameterized by treewidth.

2 Preliminaries

We assume that all our graphs are simple and undirected. Given a graph $G = (V, E)$, the number of its vertices is represented by n and the number of its edges by m. For a subset $V' \subseteq V$, by $G[V']$ we mean the subgraph of G induced by V'. With $N(u)$ we denote all vertices that are adjacent to u, and with $N[u]$, we refer to $N(u) \cup \{u\}$. Similarly, for a subset $D \subseteq V$, we define $N[D] = \cup_{v \in D} N[v]$ and $N(D) = N[D] \setminus D$. Let f be the function associated with a capacitated dominating set D. Given $u \in D$ and $v \in V \setminus D$, we say that u *dominates* v if $f(v) = u$; moreover, every vertex $u \in D$ dominates itself. Note that the capacity of a vertex v only limits the number of neighbors that v can dominate, that is, a vertex $v \in D$ can dominate $c(v)$ of its neighbors plus v itself.

Parameterized complexity is a two-dimensional framework for studying the computational complexity of problems [7, 10, 21]. One dimension is the input size n and the other one the *parameter*. A problem is called *fixed-parameter tractable (FPT)* if it can be solved in time $f(k) \cdot n^{O(1)}$, where f is a computable function only depending on k. Now we define the notion of parameterized reduction.

Definition 3 *Let A, B be parameterized problems. We say that A is (uniformly many:1) reducible to B if there is an algorithm Φ which transforms (x, k) into $(x', g(k))$ in time $f(k) \cdot |x|^\alpha$, where $f, g : \mathbb{N} \to \mathbb{N}$ are arbitrary functions and α is a constant independent of $|x|$ and k, so that $(x, k) \in A$ if and only if $(x', g(k)) \in B$.*

3 Parameterized Intractability – Hardness Results

3.1 CDS is W[1]-hard parameterized by treewidth and solution size

In this section we show that Capacitated Dominating Set is $W[1]$-hard when parameterized by treewidth and solution size. We reduce from k-Multicolor Clique, a restriction of the k-Clique problem.

> Multicolor Clique: Given an integer k and a connected undirected graph $G = (V[1] \cup V[2] \cdots \cup C[k], E)$ such that for every i the vertices of $V[i]$ induce an independent set, is there a k-clique C in G?

In fact, we will reduce to a slightly modified version of Capacitated Dominating Set, Marked Capacitated Dominating Set where we *mark* some vertices and demand that all marked vertices must be in the dominating set. We can then reduce from Marked Capacitated Dominating Set to Capacitated Dominating Set by attaching $k+1$ leaves to each marked vertex and increasing the capacity of each marked vertex by $k+1$. It is easy to see that the new instance has a k-capacitated dominating set if and only if the original one had a k-capacitated dominating set that contained all marked vertices, and that this operation does not increase the treewidth of the graph. Thus, to prove that Capacitated Dominating Set is $W[1]$-hard when parameterized by treewidth and solution size, it is sufficient to prove that Marked Capacitated Dominating Set is.

We will show how given an instance (G, k) of Multicolor Clique, we can build an instance (H, c, k') of Marked Capacitated Dominating Set such that

- $k' = 7k(k-1) + 2k$,
- G has a clique of size k if and only if H has a capacitated dominating set of size k', and
- the treewidth of H is $O(k^4)$.

For a pair of distinct integers i, j, let $E[i, j]$ be the set of edges with one endpoint in $V[i]$ and the other in $V[j]$. Without loss of generality, we will assume that $|V[i]| = N$ and $|E[i, j]| = M$ for all i, j, $i \neq j$. To each vertex v we assign a unique identification number v^{up} between $N+1$ and $2N$, and we set $v^{down} = 2N - v^{up}$. For two vertices u and v, by adding an (A, B)-*arrow* from u to v we will mean adding A subdivided edges between u and v and attaching B leaves to v (see Fig. 1). Now we describe how to build the graph H for a given instance $(G = (V[1] \cup V[2] \cdots \cup V[k], E), k)$ of Multicolor Clique.

For every integer i between 1 and k we add a marked vertex $\hat{x}_i$ that has a neighbor $\overline{v}$ for every vertex v in $V[i]$. For every $j \neq i$, we add a marked vertex $\hat{y}_{ij}$ and a marked vertex $\hat{z}_{ij}$. Now, for every vertex $v \in V[i]$ and every integer $j \neq i$ we add a (v^{up}, v^{down})-arrow from $\overline{v}$ to $\hat{y}_{ij}$ and a (v^{down}, v^{up})-arrow from $\overline{v}$ to $\hat{z}_{ij}$. Finally we add a set S_i of $k'+1$ vertices and make every vertex in S_i adjacent to every vertex $\overline{v}$ with $v \in V[i]$. See Fig. 2 for an illustration.

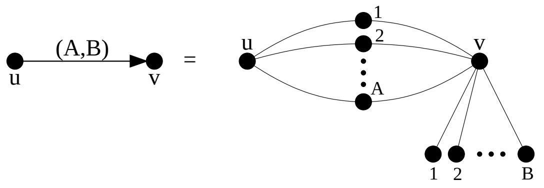

Figure 1: Adding an (A, B)-arrow from u to v.

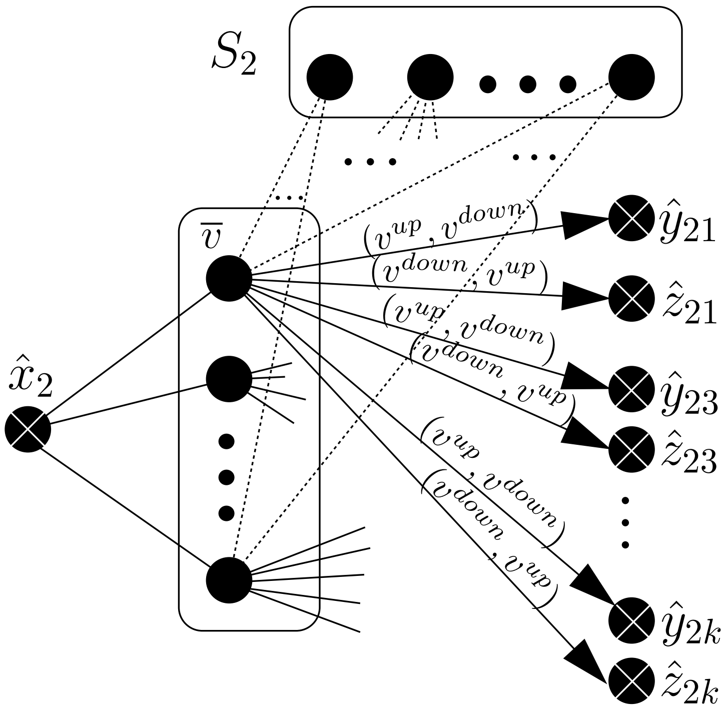

Figure 2: Vertex selection for color class 2.

Similarly, for every pair of integers i, j with $i < j$, we add a marked vertex $\hat{x}_{ij}$ with a neighbor $\overline{e}$ for every edge e in $E[i, j]$. Moreover, we add four new marked vertices $\hat{p}_{ij}, \hat{p}_{ji}, \hat{q}_{ij}$, and $\hat{q}_{ji}$. For every edge $e = \{u, v\}$ in $E[i, j]$ with $u \in V[i]$ and $v \in V[j]$, we add a (u^{down}, u^{up})-arrow from $\overline{e}$ to $\hat{p}_{ij}$, a (u^{up}, u^{down})-arrow from $\overline{e}$ to $\hat{q}_{ij}$, a (v^{down}, v^{up})-arrow from $\overline{e}$ to $\hat{p}_{ji}$ and a (v^{up}, v^{down})-arrow from $\overline{e}$ to $\hat{p}_{ji}$. We also add a set S_{ij} of $k' + 1$ vertices and make every vertex in S_{ij} adjacent to every vertex $\overline{e}$ with $e \in E[i, j]$. See Fig. 3 for an illustration.

Finally, we add a marked vertex $\hat{r}_{ij}$ and a marked vertex $\hat{s}_{ij}$ for every $i \neq j$. For every $i \neq j$, we add $(2N, 0)$-arrows from $\hat{y}_{ij}$ to $\hat{r_{ij}}$, from $\hat{p}_{ij}$ to $\hat{r_{ij}}$, from $\hat{z}_{ij}$ to $\hat{s_{ij}}$, and from $\hat{q}_{ij}$ to $\hat{s_{ij}}$ (see Fig. 4). This concludes the description of the graph H.

We now describe the capacities of the vertices. For every $i \neq j$, the vertex $\hat{x}_i$ has capacity $N - 1$, the vertex $\hat{x}_{ij}$ has capacity $M - 1$, the vertices $\hat{y}_{ij}$ and $\hat{z}_{ij}$ both have capacity $2N^2$, the vertices $\hat{p}_{ij}$ and $\hat{q}_{ij}$ have capacity $2NM$, and both $\hat{r}_{ij}$ and $\hat{s}_{ij}$ have capacity $2N$. For all other vertices, their capacity is equal to their degree in H.

Observation 1 *The treewidth of H is $O(k^4)$.*

Proof: If we remove all marked vertices ($\bigcup_{i=1}^{k} S_i$ and $\bigcup_{i \neq j} S_{ij}$), a total of $O(k^4)$ vertices, from H, we obtain a forest. As deleting a vertex reduces the treewidth by at most one, this

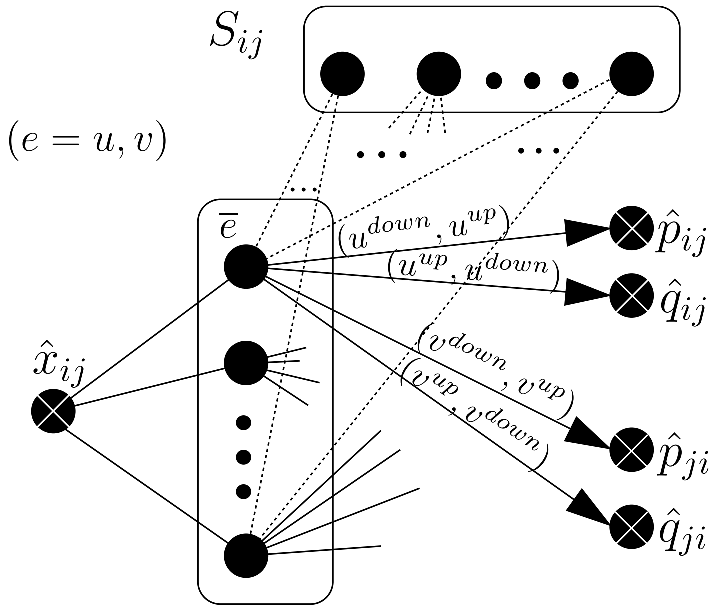

Figure 3: Edge selection

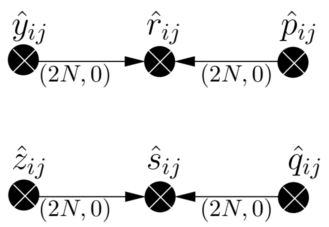

Figure 4: Vertex-Edge incidence gadget

concludes the proof. □

Lemma 1 *If G has a multicolor clique $C = \{v_1, v_2, \dots, v_k\}$ then H has a capacitated dominating set D of size k' containing all marked vertices.*

Proof: For every $i < j$ let e_{ij} be the edge from v_i to v_j in G. In addition to all the marked vertices, let D contain $\overline{v_i}$ and $\overline{e_{ij}}$ for every $i < j$. Clearly D contains exactly k' vertices, so it remains to prove that D is indeed a capacitated dominating set.

For every $i < j$, let $\hat{x}_i$ and $\hat{x}_{ij}$ dominate all their neighbors except for $\overline{v_i}$ and $\overline{e_{ij}}$ respectively. The vertices $\overline{v_i}$ and $\overline{e_{ij}}$ can dominate all their neighbors, since their capacity is equal to their degree. Let $\hat{r}_{ij}$ dominate v_i^{down} of the vertices in the $(2N, 0)$-arrow from $\hat{y}_{ij}$, and v_i^{up} of the vertices of the $(2N, 0)$-arrow from $\hat{p}_{ij}$. Similarly let $\hat{s}_{ij}$ dominate v_i^{up} of the vertices in the $(2N, 0)$-arrow from $\hat{z}_{ij}$, and v_i^{down} of the vertices of the $(2N, 0)$-arrow from $\hat{q}_{ij}$. Finally, for every $i \neq j$ we let $\hat{y}_{ij}$, $\hat{z}_{ij}$, $\hat{p}_{ij}$ and $\hat{q}_{ij}$ dominate all their neighbors that have not been dominated yet. One can easily check that every vertex of H will either be a dominator or dominated in this manner, and that no dominator dominates more vertices than its capacity. □

Lemma 2 *If H has a capacitated dominating set D of size k' containing all marked vertices, then G has a multicolor clique of size k.*

Proof: Observe that for every integer $1 \le i \le k$, there must be a $v_i \in V[i]$ such that $\overline{v}_i \in D$. Otherwise we have that $S_i \subset D$ and, since $|S_i| > k'$, we obtain a contradiction. Similarly, for every pair of integers i, j with $i < j$ there must be an edge $e_{ij} \in E[i,j]$ such that $\overline{e}_{ij} \in D$. We let $e_{ji} = e_{ij}$. Since $|D| \le k'$ it follows that these are the only unmarked vertices in D. Since all the unmarked vertices in D have capacity equal to their degree, we can assume that each such vertex dominates all its neighbors. We now proceed with proving that for every pair of integers i,j with $i \ne j$, $e_{ij} = uv$ is incident to v_i. We prove this by showing that if $u \in V[i]$ then $v_i^{up} + u^{down} = 2N$.

Suppose for a contradiction that $v_i^{up} + u^{down} < 2N$. Observe that each vertex of $T = (N(\hat{y}_{ij}) \cup N(\hat{r}_{ij}) \cup N(\hat{p}_{ij})) \setminus (N(\overline{v}_i) \cup N(\overline{e}_{ij}))$ must be dominated by either $\hat{y}_{ij}$, $\hat{r}_{ij}$, or $\hat{p}_{ij}$. However, by our assumption that $v_i^{up} + u^{down} < 2N$, it follows that $|T| = 2N^2 + 4N + 2MN - (v_i^{up} + u^{down}) > 2N^2 + 2N + 2MN$. The sum of the capacities of $\hat{y}_{ij}$, $\hat{r}_{ij}$, and $\hat{p}_{ij}$ is exactly $2N^2 + 2N + 2MN$. Thus it is impossible that every vertex of T is dominated by one of $\hat{y}_{ij}$, $\hat{r}_{ij}$, and $\hat{p}_{ij}$, a contradiction. If $v_i^{up} + u^{down} > 2N$ then $v_i^{down} + u^{up} < 2N$, and we can apply an identical argument for $\hat{z}_{ij}$, $\hat{s}_{ij}$, and $\hat{q}_{ij}$.

Thus, it follows that for every $i \ne j$ there is an edge e_{ij} incident both to v_i and to v_j. Thus $\{v_1, v_2, \ldots, v_k\}$ forms a clique in G. As any k-clique in G is a multicolor clique this completes the proof. $\square$

Observation 1, Lemma 1 and Lemma 2 immediately imply the following theorem.

Theorem 1 CDS *parameterized by treewidth and solution size is* $W[1]$*-hard.*

3.2 CVC parameterized by treewidth is W[1]-hard

Usually vertex cover problems can be seen as restrictions of domination problems, and therefore it is natural to expect Capacitated Vertex Cover to be somewhat easier than Capacitated Dominating Set. In this section, we give a result similar to the hardness result for Capacitated Dominating Set, but weaker in the sense that we only show that Capacitated Vertex Cover is hard when parameterized by the treewidth, while we have seen in the previous section that Capacitated Dominating Set is hard when parameterized by the treewidth and the solution size.

To obtain our result we reduce from Multicolor Clique, as in the previous section. Again, we reduce to a marked version of Capacitated Vertex Cover, where we search for a size k' capacitated vertex cover that contains all the marked vertices. The reduction from Marked Capacitated Vertex Cover to Capacitated Vertex Cover is identical to the reduction from Marked Capacitated Dominating Set to Capacitated Dominating Set described in the previous section. Notice also that in Marked Capacitated Vertex Cover it makes sense to have marked vertices with capacity zero, as they will get non-zero capacity after the reduction to Capacitated Vertex Cover.

We reduce by building for an instance (G, k) of Multicolor Clique an instance (H, c, k') of Marked Capacitated Vertex Cover. In fact, we construct the graph H from G in almost the same manner as in the reduction to Marked Capacitated Dominating Set. The only differences are:

- We do not add the vertex sets S_i and S_{ij} for every i, j.
- When we add an (A, B)-arrow from u to v, the A vertices on the subdivided edges are marked and have capacity 1, while the B leaves attached to v are also marked but have capacity 0.

- $k' = 7k(k-1) + 2k + (2k^2N + (2M+4) \cdot k \cdot (k-1)) \cdot 2N$.

The new term in the value of k' is simply a correction for all the extra marked vertices in the (A, B)-arrows. Notice that in this case the value of k' is not a function of k alone, and that therefore this reduction does not imply that Capacitated Vertex Cover is $W[1]$-hard parameterized by treewidth and solution size. However, by applying arguments almost identical to the ones in the previous section, we can prove the following claims. The verification of these claims is similar to the ones we made in the last section and hence the details are omitted.

Claim 1 *The treewidth of H is $O(k^3)$.*

Claim 2 *If G has a multicolor clique $C = \{v_1, v_2, \ldots, v_k\}$ then H has a capacitated vertex cover S of size k' containing all marked vertices.*

Claim 3 *If H has a capacitated vertex cover S of size k' containing all marked vertices, then G has a multicolor clique of size k.*

Together, the claims imply that Capacitated Vertex Cover parameterized by treewidth is $W[1]$-hard.

Theorem 2 CVC *parameterized by treewidth is $W[1]$-hard.*

4 FPT Algorithm for CVC on Graphs of Bounded Treewidth

In the last section we showed that Capacitated Vertex Cover, when parameterized only by treewidth (tw) of the input graph, is W[1]-hard. However, for Capacitated Dominating Set we were able to show that the problem remains W[1]-hard even when we parameterize by both tw and k (the solution size). We complement the hardness result about Capacitated Vertex Cover of the last section by giving a time $2^{O(\text{tw} \log k)} n^{O(1)}$ algorithm on graphs of bounded treewidth with respect to the combined parameters tw and k, a result which was sketched independently by Hannes Moser [20]. Furthermore, using this $2^{O(\text{tw} \log k)} n^{O(1)}$ time algorithm for CVC on graphs of bounded treewidth, we give an improved algorithm for the weighted version of Capacitated Vertex Cover in general graphs. Our algorithm, running in time $O(2^{O(k \log k)} n^{O(1)})$, improves the earlier algorithm of Guo et al. [15], that runs in time $O(1.2^{k^2} + n^2)$.

To this end, we give a dynamic programming algorithm working on a so-called *nice tree decomposition* of the input graph G: A tree decomposition (X, U) is a nice tree decomposition if one can root U in such a way that the root and every inner node of U is either an *insert node*, a *forget node*, or a *join node*. Thereby, a node i of U is an insert node if i has exactly one child j, and X_i consists of all vertices of X_j plus one additional vertex; it is a forget node if i has exactly one child j, and X_i consists of all but one vertices of X_j; and it is a join node if i has exactly two children j_1, j_2, and $X_i = X_{j_1} = X_{j_2}$. Given a tree decomposition of width tw, a nice tree decomposition of the same width can be found in linear time [17]. In what follows, we assume that the nice tree decomposition (X, U) that we are using has the additional property that the bag associated with the root of U is empty (such a decomposition can easily be constructed by taking an arbitrary nice tree decomposition and adding some forget nodes "above" the original root). Similarly, we assume that every bag associated with a leaf node different from the root of U contains exactly one vertex. For a node i in the tree U of a tree decomposition (X, U), let

$$\begin{aligned} Y_i &:= \bigcup\{v \in X_j \mid j \text{ is a node in the subtree of } U \text{ whose root is } i\}, \\ Z_i &:= Y_i \setminus X_i, \text{ and} \\ E_i &:= \{\{v, w\} \in E \mid v \in Z_i \vee w \in Z_i\}. \end{aligned}$$

Starting at the leaf nodes of U that are different from the root, our dynamic programming algorithm assigns to every node i of U a table A_i that has

- a column ℓ with $\ell \leq k$,
- for every vertex $v \in X_i$ a column $\mathrm{vc}(v)$ with $\mathrm{vc}(v) \in \{\mathit{true}, \mathit{false}\}$, and
- for every vertex $v \in X_i$ a column $\mathrm{s}(v)$ with $\mathrm{s}(v) \in \{\mathit{null}, 0, 1, \ldots, k-1\}$.

Every row of such a table A_i corresponds to a solution (f, C) for CVC on the subgraph of G that consists of all vertices in Y_i and all edges in E having at least one endpoint in Z_i. More exactly, for every row of a table A_i there is a vertex set $C \subseteq Y_i$ and mapping $f : E_i \to C$ with the following properties:

- C is a capacitated vertex cover for $G_i = (Y_i, E_i)$.
- $|C| \leq \ell$.
- C contains all vertices $v \in X_i$ with $\mathrm{vc}(v) = \mathit{true}$ and no vertex $v \in X_i$ with $\mathrm{vc}(v) = \mathit{false}$.
- For every vertex $v \in X_i \cap C$, we have

$$|\{\{v, w\} \in E_i \mid f(\{v, w\} = w\}| = \mathrm{s}(v),$$

  and for every vertex $v \in X_i \setminus C$, we have $\mathrm{s}(v) = \mathit{null}$.

Intuitively speaking, for a vertex $v \in C$, the variable $\mathrm{s}(v)$ contains the number of edges incident to v that are covered by vertices in Z_i and, therefore, do not have to be covered by v. The simple observation that $s(v)$ can be at most $k-1$ (because C can contain at most $k-1$ neighbors of v) is crucial for the running time of the algorithm.

Clearly, if the table associated with the root of U is nonempty, the given instance of CVC is a *yes*-instance.

We will now describe the computation of the table A_i for a node i in U, depending on if i is a leaf node different from the root, an insert node, a forget node, or a join node. If necessary, we write ℓ_i, $\mathrm{vc}_i(v)$, and $\mathrm{s}_i(v)$ in order to make clear that a value ℓ, $\mathrm{vc}(v)$, and $\mathrm{s}(v)$, respectively, stems from a row of a table A_i.

The node i is a leaf node different from the root. Let $X_i = \{v\}$. Then we add one row to the table A_i for the case that v is not part of C and one row for the case that v is part of C, provided that $k > 0$. Because i has no child and, hence, no neighbor of v belongs to Z_i, the variable $\mathrm{s}(v)$ is set to 0 in the case that v is part of C:

```
if $k > 0$: {
    add a new row to $A_i$;
    update the new row in $A_i$ and set $\mathrm{vc}(v) := \mathit{true}$;   $\mathrm{s}(v) := 0$;   $\ell := 1$; }
add a new row to $A_i$;
update the new row in $A_i$ and set $\mathrm{vc}(v) := \mathit{false}$;   $\mathrm{s}(v) := \mathit{null}$;   $\ell := 0$;
```

The node i is an insert node. Let j be the child of i in U, and let $X_i = X_j \cup \{v\}$. Here we extend the table A_j by adding the variables $\mathrm{vc}(v)$ and $\mathrm{s}(v)$. For every row of A_j, we add one row to the table A_i for the case that v is not part of C and one row for the case that v is part of C, provided that $\ell_j < k$. Because no neighbor of v can belong to Z_i, the variable $\mathrm{s}(v)$ is set to 0 in the case that v is part of C:

```
for every row $r$ of $A_j$: {
    if $\ell_j < k$: {
        copy the row $r$ from $A_j$ into $A_i$;
        update the new row in $A_i$ and set $\mathrm{vc}(v) :=$ *true*;   $\mathrm{s}(v) := 0$;   $\ell := \ell + 1$; }
    copy the row $r$ from $A_j$ into $A_i$;
    update the new row in $A_i$ and set $\mathrm{vc}(v) :=$ *false*;   $\mathrm{s}(v) :=$ *null*; }
```

The node i is a forget node. Let j be the child of i in U, and let $X_i = X_j \setminus \{v\}$. Clearly, all neighbors of v belong to Y_j due to the definition of a tree decomposition. What has to be done is to consider the edges $\{v, w\}$ with $w \in X_i$, to decide which of them shall be covered by v, and to set the value of $s_j(v)$ accordingly. Note that this approach ensures that for all edges $\{v, w\}$ with $w \in Z_j$ we have already decided in a previous step which of these edges are covered by v. More exactly, for every row of A_j, we perform the following steps. If $\mathrm{vc}_j(v) = \mathit{true}$, then we try all possibilities for which edges between v and vertices $w \in X_i$ can be covered by v and add rows to A_i accordingly. If $\mathrm{vc}_j(v) = \mathit{false}$, then, of course, no edge between v and vertices $w \in X_j$ can be covered by v, and we add one row to A_i. In both cases, we have to check that for every edge $\{v, w\}$ with $w \in X_i$ that is not covered by v it holds that $\mathrm{vc}_j(w) = \mathit{true}$ and the remaining capacity of w, which can be computed from $s(w)$ and the number of w's neighbors in Z_j, is big enough to cover $\{v, w\}$:

```
$N' := N(v) \cap X_i$;
for every row $r$ of $A_j$: {
    if $\mathrm{vc}_j(v) =$ *true*: {
        for every subset $N''$ of $N'$ with $|N''| = \min\{|N'|, \mathrm{cap}(v) - (|N(v) \cap Z_j| - \mathrm{s}_j(v))\}$: {
            if $\forall w \in N' \setminus N'' : \mathrm{vc}_j(w) \wedge \mathrm{cap}(w) > |N(w) \cap Z_j| - \mathrm{s}_j(w)$: {
                copy the row $r$ from $A_j$ into $A_i$;
                for every vertex $w \in N''$ with $\mathrm{vc}(w) =$ *true*: {
                    update the new row in $A_i$ and set $\mathrm{s}(w) := \mathrm{s}(w) + 1$; }}}
    else: { // $\mathrm{vc}_j(v) =$ *false*
        if $\forall w \in N' : \mathrm{vc}(w) =$ *true* $\wedge\, \mathrm{cap}(w) > |N(w) \cap Z_j| - \mathrm{s}_j(w)$: {
            copy the row $r$ from $A_j$ into $A_i$; }}}
```

The node i is a join node. Let j_1 and j_2 be the children of i in U. Here we consider every pair r_1, r_2 of rows where r_1 is from A_{j_1} and r_2 is from A_{j_2}. We say that two rows r_1 and r_2 are *compatible* if for every vertex v in X_i it holds that $\mathrm{vc}_{j_1}(v) = \mathrm{vc}_{j_2}(v)$. If they are compatible, then we check whether for every vertex $v \in X_i$ with $\mathrm{vc}_{j_1}(v) = \mathrm{vc}_{j_2}(v) = \mathit{true}$ the number of edges $\{v, w\}$ covered by v with $w \in Z_{j_1}$ plus the number of edges $\{v, w\}$ covered by v with $w \in Z_{j_2}$ is at most $\mathrm{cap}(v)$. If this is the case, a new row is added to A_i:

```
for every compatible pair $r_1, r_2$ of rows where $r_1$ is from $A_{j_1}$ and $r_2$ is from $A_{j_2}$: {
    if $\forall v \in X_i : \mathrm{vc}_{j_1}(v) =$ *false* $\vee\, \mathrm{cap}(v) \geq |N(v) \cap Z_{j_1}| - \mathrm{s}_{j_1}(w) + |N(v) \cap Z_{j_2}| - \mathrm{s}_{j_2}(w)$: {
        add a new row to $A_i$;
        update the new row in $A_i$ and set $\ell := \ell_{j_1} + \ell_{j_2} - |\{v \in X_i \mid \mathrm{vc}_{j_1}(v) =$ *true*$\}|$;
        for every vertex $v \in X_i$: {
            update the new row in $A_i$ and set $\mathrm{vc}(v) = \mathrm{vc}_{j_1}(v)$;   $\mathrm{s}(v) = \mathrm{s}_{j_1}(v) + \mathrm{s}_{j_2}(v)$; }}}
```

In all four cases (i is a leaf node different from the root, an insert node, a forget node, or a join node), after inserting a row to A_i, we delete *dominated* rows from A_i. A row r_1 is dominated by a row r_2 if r_1 and r_2 are compatible, the value of ℓ in r_1 is equal or greater than the value of ℓ in r_2, and for every vertex $v \in X_i$ with $\mathrm{vc}(v) = \mathit{true}$ the value of $\mathrm{s}(v)$ in r_1 is equal or less than the value of $\mathrm{s}(v)$ in r_2. The correctness of this data reduction is obvious: If the solution corresponding to r_1 can be extended to a solution for the whole graph, then this is also possible with the solution corresponding to r_2 instead. Clearly, due to this data reduction, the table can never contain more than k^{tw} rows, which leads to the following theorem.

Theorem 3 CVC *on graphs of treewidth* tw *can be solved in* $k^{3\cdot \mathrm{tw}} \cdot n^{O(1)}$ *time.*

Proof: The correctness of the algorithm follows from the above description. The running time for computing one table A_i associated with a tree node i is bounded from above by $k^{3\cdot \mathrm{tw}} \cdot n^{O(1)}$, due to the fact that every table contains at most k^{tw} rows and that the tree decomposition has $O(n)$ tree nodes [17]. □

We mention in passing that with usual backtracking techniques it is possible to construct the mapping f and the set C after running the dynamic programming algorithm.

CVC in General Undirected Graphs: The algorithm described above can also be used for solving CVC on general graphs with the following two observations. Firstly, the treewidth of graphs that have a vertex cover of size k is bounded above by k, and a corresponding tree decomposition of width k can be found in $O(1.2738^k + kn)$ time [4]. (For a graph $G = (V, E)$ that has a vertex cover C with $|C| = k$, a tree decomposition of width k can be constructed as follows: Let U be a path of length $|V \setminus C|$, and assign to every node i of U a bag X_i that contains C and one vertex from $V \setminus C$. The vertex cover of size k can be found in time $O(1.2738^k + kn)$ [4].) Secondly, Theorem 3 can easily be adapted to the *weighted* version of CVC, where every vertex of the input graph has, in addition to the capacity, a weight, and the question is if there is a capacitated vertex cover whose weight is at most k. With these observations, we get the following corollary.

Corollary 1 *The weighted version of* CVC *on general graphs can be solved in* $k^{3k} \cdot n^{O(1)} = 2^{O(k \log k)} \cdot n^{O(1)}$ *time.*

5 Conclusions and Discussions

In this paper we studied Capacitated Vertex Cover and Capacitated Dominating Set, generalizations of Vertex Cover and Dominating Set, respectively, from the parameterized perspective. In particular, we focused on the parameterized complexity when parameterized by (a) the treewidth of the input graph and (b) the treewidth of the input graph and the solution size k. While the original version of these problems were known to be FPT when parameterized by treewidth we found their behavior in the capacitated context to be surprising. In particular, CDS turned out to be W[1]-hard and CVC to be FPT when parameterized by treewidth and k. An improved algorithm for CVC in general undirected graphs was obtained by using the FPT algorithm for CVC (when parameterized by treewidth and k) as a subroutine. We also observed that CVC is possibly the first known "subset problem" which has been shown to be FPT when parameterized by solution size but W[1]-hard when parameterized by treewidth.

It is easy to observe that if a planar graph has a CDS of size at most k then the treewidth of the input graph is at most $O(\sqrt{k})$ [1, 6, 11]. Hence, in order to show that CDS is FPT for planar graphs, it is sufficient to obtain a dynamic programming algorithm for it on *planar graphs* of bounded treewidth. The following question in this direction remains unanswered:

- Is CDS in planar graphs parameterized by solution size fixed parameter tractable?

References

[1] J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, and R. Niedermeier. Fixed parameter algorithms for DOMINATING SET and related problems on planar graphs. *Algorithmica*, 33(4):461–493, 2002.

[2] J. Alber, M. R. Fellows, and R. Niedermeier. Polynomial-time data reduction for dominating set. *J. ACM*, 51(3):363–384, 2004.

[3] M. Bläser. Computing small partial coverings. *Inf. Process. Lett.*, 85(6):327–331, 2003.

[4] J. Chen, I. A. Kanj, and G. Xia. Improved parameterized upper bounds for Vertex Cover. In *Proc. 31st MFCS*, volume 4162 of *LNCS*, pages 238–249. Springer, 2006.

[5] J. Chuzhoy and J. Naor. Covering problems with hard capacities. *SIAM J. Comput.*, 36(2):498–515, 2006.

[6] E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *J. ACM*, 52(6):866–893, 2005.

[7] R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer, 1999.

[8] R. Duh and M. Fürer. Approximation of k-Set Cover by semi-local optimization. In *Proc. 29th STOC*, pages 256–264. ACM Press, 1997.

[9] U. Feige. A threshold of ln n for approximating Set Cover. *J. ACM*, 45(4):634–652, 1998.

[10] J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer, 2006.

[11] F. V. Fomin and D. M. Thilikos. Dominating sets in planar graphs: Branch-width and exponential speed-up. *SIAM J. Comput.*, 36(2):281–309, 2006.

[12] R. Gandhi, E. Halperin, S. Khuller, G. Kortsarz, and A. Srinivasan. An improved approximation algorithm for Vertex Cover with hard capacities. *J. Comput. System Sci.*, 72(1):16–33, 2006.

[13] S. Guha, R. Hassin, S. Khuller, and E. Or. Capacitated vertex covering. *J. Algorithms*, 48(1):257–270, 2003.

[14] J. Guo and R. Niedermeier. Linear problem kernels for NP-hard problems on planar graphs. In *Proc. 34th ICALP*, volume 4596 of *LNCS*, pages 375–386. Springer, 2007.

[15] J. Guo, R. Niedermeier, and S. Wernicke. Parameterized complexity of Vertex Cover variants. *Theory Comput. Syst.*, 41(3):501–520, 2007.

[16] E. Halperin and A. Srinivasan. Improved approximation algorithms for the Partial Vertex Cover problem. In *Proc. 5th APPROX*, volume 2462 of *LNCS*, pages 161–174. Springer, 2002.

[17] T. Kloks. *Treewidth. Computations and Approximations*, volume 842 of *LNCS.* Springer, 1994.

[18] L. Lovàsz. On the ratio of optimal fractional and integral covers. *Discrete Math.*, 13:383–390, 1975.

[19] D. Mölle, S. Richter, and P. Rossmanith. Enumerate and expand: Improved algorithms for Connected Vertex Cover and Tree Cover. In *Proc. CSR*, volume 3967 of *LNCS*, pages 270–280. Springer, 2006.

[20] H. Moser. Exact algorithms for generalizations of Vertex Cover. Diploma thesis, Institut für Informatik, Friedrich-Schiller Universität Jena, 2005.

[21] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms.* Oxford University Press, 2006.

[22] N. Nishimura, P. Ragde, and D. M. Thilikos. Fast fixed-parameter tractable algorithms for nontrivial generalizations of Vertex Cover. *Discrete Appl. Math.*, 152(1–3):229–245, 2005.

[23] V. Raman and S. Saurabh. Short cycles make W-hard problems hard: FPT algorithms for W-hard problems in graphs with no short cycles. *Algorithmica.* To appear.