Parameterized Complexity of Vertex Cover Variants*

Jiong Guo† Rolf Niedermeier Sebastian Wernicke‡
Institut für Informatik, Friedrich-Schiller-Universität Jena,
Ernst-Abbe-Platz 2, D-07743 Jena, Germany.
{guo,niedermr,wernicke}@minet.uni-jena.de.

February 28, 2006

Abstract

Important variants of the Vertex Cover problem (among others, Connected Vertex Cover, Capacitated Vertex Cover, and Maximum Partial Vertex Cover) have been intensively studied in terms of polynomial-time approximability. By way of contrast, their parameterized complexity has so far been completely open. We close this gap here by showing that, with the size of the desired vertex cover as parameter, Connected Vertex Cover and Capacitated Vertex Cover are both fixed-parameter tractable while Maximum Partial Vertex Cover is W[1]-complete. This answers two open questions from the literature. The results extend to several closely related problems. Interestingly, although the considered variants of Vertex Cover behave very similar in terms of constant-factor approximability, they display a wide range of different characteristics when investigating their parameterized complexities.

1 Introduction

Given an undirected graph $G = (V, E)$, the NP-complete Vertex Cover problem is to find a set $C \subseteq V$ with $|C| \leq k$ such that each edge in E has at least

*An extended abstract of this work appears under the title "Parameterized Complexity of Generalized Vertex Cover Problems" in the proceedings of the Ninth Workshop on Algorithms and Data Structures (WADS'05), Springer, LNCS 3608, pages 36–48, held in Waterloo, Canada, August 15–17, 2005.

†Supported by the Deutsche Forschungsgemeinschaft, Emmy Noether research group PIAF (fixed-parameter algorithms), NI 369/4.

‡Supported by the Deutsche Telekom Stiftung and the Studienstiftung des deutschen Volkes.

one endpoint in C. In some sense, Vertex Cover could be considered the *Drosophila* of fixed-parameter algorithmics [22, 37]:

1. There is a long list of continuous improvements on the combinatorial explosion with respect to the parameter k when solving the problem exactly. The currently best exponential bound is below 1.28^k [9, 38, 16, 40, 14, 17].

2. Vertex Cover has been a benchmark for developing sophisticated data reduction and problem kernelization techniques [1, 24].

3. It was the first parameterized problem where the usefulness of interleaving depth-bounded search trees with problem kernelization was proven [39].

4. Restricted to planar graphs, it was—besides Dominating Set—one of the central problems for the development of "subexponential" fixed-parameter algorithms and the corresponding theory of relative lower bounds [3, 5, 13].

5. Vertex Cover served as a testbed for algorithm engineering in the realm of fixed-parameter algorithms [1, 2, 4, 15].

6. Studies of Vertex Cover led to new research directions within parameterized complexity such as counting [8], enumerating [25, 21], parallel processing [2, 15], and using "vertex cover structure" as a general strategy to solve parameterized problems [42].

This probably incomplete list gives an impression of how important Vertex Cover was and continues to be for the whole field of parameterized complexity. However, research to date in this field appears to have neglected a closer investigation of recent significant generalizations and variants of Vertex Cover which so far only been studied in the context of their polynomial-time approximability. We close this gap here by providing several first-time parameterized complexity results, which also answer two open questions from the literature.

We are only aware of two papers that perform somewhat related research. First, Nishimura, Ragde, and Thilikos [41] also study variants of Vertex Cover. However, they follow a completely different route: Whereas we study concrete problems such as Capacitated Vertex Cover or Maximum Partial Vertex Cover on general graphs, their interest lies in recognizing general classes of graphs with a very special case of interest being the class of graphs with bounded vertex covers (refer to [41] for details). Second, Bläser [10] shows that some partial covering problems are fixed-parameter tractable when the parameter is the number of objects covered instead of the size of the covering set.

This work deals with a whole list of vertex covering problems, all of them possessing constant-factor (mostly factor-2) polynomial-time approximation algorithms. Deferring their formal definitions to the next section, we now informally describe the studied problems, known results, and our new results which are also

Problem	Result	
Connected Vertex Cover	$6^k n + 4^k n^2 + n^2 \log n + nm$	Theorem 1
Tree Cover	$6^k k^2 n + 4^k k^2 n^2 + kn^3$	Theorem 3
Tour Cover	$2^k k^{2k} m$	Theorem 5
Capacitated Vertex Cover	$1.2^{k^2} + n^2$	Theorem 7
Soft Capacitated Vertex Cover	$1.2^{k^2} + n^2$	Corollary 11
Hard Capacitated Vertex Cover	$1.2^{k^2} + n^2$	Corollary 11
Maximum Partial Vertex Cover	W[1]-complete	Theorem 14
Minimum Partial Vertex Cover	W[1]-complete	Theorem 15

Table 1: New parameterized complexity results for several NP-complete variants of Vertex Cover treated in this work. The parameter k is the size of the desired vertex cover, m denotes the number of edges, and n denotes the number of vertices.

surveyed in Table 1. In our presentation, n denotes the number of vertices and m denotes the number of edges of the input graph. The parameter k always denotes the size of the vertex cover.

1. For Connected Vertex Cover one demands that the vertex cover set be connected. This problem is known to have a factor-2 approximation [44]. We show that it can be solved in $O(6^k n + 4^k n^2 + n^2 \log n + nm)$ time. In addition, we derive results for the closely related variants Tree Cover and Tour Cover.[1]

2. For Capacitated Vertex Cover, the "covering capacity" of each graph vertex is limited in that it may not cover all of its incident edges. This problem has a factor-2 approximation [30]. Addressing an open problem from [30], we show that Capacitated Vertex Cover can be solved in $O(1.2^{k^2} + n^2)$ time using an enumerative approach. We also provide a problem kernelization. Altogether, we thus show that Capacitated Vertex Cover—including two variants with "hard" and "soft" capacities—is fixed-parameter tractable.

3. For Maximum Partial Vertex Cover, one only wants to cover a specified number of edges (that is, not necessarily all of them) by at most k vertices. This problem is known to have a factor-2 approximation [11]. Answering an open question from [6], we show that this problem appears to be fixed-parameter intractable—it is W[1]-complete. The same is proven

[1] Improving upon our $6^k \cdot n^{O(1)}$ bounds for Connected Vertex Cover and Tree Cover, Mölle et al. [36] very recently reported $3.24^k \cdot n^{O(1)}$ algorithms for these two problems.

for its minimization version.[2]

Interestingly, although all problems we consider in this work behave in more or less the same way from the viewpoint of polynomial-time approximability—all have factor-2 approximations—the picture becomes completely different from a parameterized complexity point of view: Maximum Partial Vertex Cover appears to be intractable and Capacitated Vertex Cover appears to be significantly harder than Connected Vertex Cover.

Summarizing, we emphasize that our main focus is on deciding between fixed-parameter tractability and W[1]-hardness for all of the considered problems. It is conceivable that run-time improvements on our initial algorithms may come up in future research.

2 Preliminaries and Previous Work

We consider three types of Vertex Cover (VC) variants, namely demanding that the vertices of the cover must be *connected* (Section 3), introducing *capacities* for the vertices (Section 4), and relaxing the condition that *all* edges in the graph must be covered (Section 5). In this section, we first give a brief overview about parameterized complexity and parameterized reductions. This is followed by the formal definitions for the considered problems and an overview of some known results (mainly in the area of polynomial-time approximation). An overview of our results is provided in Table 1.

Parameterized complexity is a two-dimensional framework for studying the computational complexity of problems.[3] One dimension is the input size n (as in classical complexity theory) and the other one the *parameter* k (usually a positive integer). A problem is called *fixed-parameter tractable* (fpt) if it can be solved in $f(k) \cdot n^{O(1)}$ time, where f is a computable function only depending on k. A core tool in the development of fixed-parameter algorithms is polynomial-time preprocessing by *data reduction rules*, often yielding a *reduction to a problem kernel*. Here the goal is, given any problem instance x with parameter k, to transform it into a new instance x' with parameter k' such that the size of x' is bounded by some function only depending on k, the instance (x, k) has a solution iff (x', k') has a solution, and $k' \leq k$.

A formal framework to show *fixed-parameter intractability* was developed by Downey and Fellows [22] who introduced the concept of *parameterized reductions*. A parameterized reduction from a parameterized language L to another parameterized language L' is a function that, given an instance (x, k), computes in time $f(k) \cdot n^{O(1)}$ an instance (x', k') (with k' only depending on k) such that

[2]Parameterized hardness results for the two Partial Vertex Cover problems have also been independently announced by Cai [12].

[3]For more detailed introductions see [22, 26, 37].

$(x, k) \in L \Leftrightarrow (x', k') \in L'$. The basic complexity class for fixed-parameter intractability is W[1] as there is good reason to believe that W[1]-hard problems are not fixed-parameter tractable [22].

As mentioned above, we consider three variants of Vertex Cover (VC) on undirected graphs. The first variant demands connectedness of the cover and is dealt with in Section 3.

> Connected Vertex Cover: Given a graph $G = (V, E)$ and an integer $k \geq 0$, determine whether there exists a vertex cover C for G containing at most k vertices such that the subgraph of G induced by C is connected.

This problem is NP-complete and polynomial-time approximable within a factor of two [7]. Two variants are derived by introducing a weight function $w : E \to \mathbb{R}^+$ on the edges and requiring that the cover induce a subgraph with a certain structure and minimum weight.

> Tree Cover: Given a graph $G = (V, E)$ where each edge is weighted by a positive real number, an integer $k \geq 0$, and a real number $W > 0$, determine whether there exists a tree $T = (V', E')$ with $V' \subseteq V$, $|V'| \leq k$, $E' \subseteq E$, and $\sum_{e \in E'} w(e) \leq W$ such that V' is a vertex cover for G.

Note that Tree Cover is equivalent to Connected Vertex Cover for unweighted graphs. The closely related problem Tour Cover differs from Tree Cover only in that the edges in G' should form a closed walk (which may contain repeated vertices and edges) instead of a tree. Both Tree Cover and Tour Cover were introduced in [7] where it is shown that Tree Cover is polynomial-time approximable within factor 3.55 and Tour Cover within factor 5.5. Könemann et al. [34] improved both approximation factors to 3.

Section 4 considers the Capacitated Vertex Cover (CVC) problem and related variants. Here, the graph is *capacitated*, meaning that each vertex $v \in V$ is assigned an integer *capacity* $c(v) \geq 1$ that limits the number of edges it can cover when being part of the vertex cover.

Definition 1. *Given a capacitated graph $G = (V, E)$ and a vertex cover C for G, we call C* capacitated vertex cover *if there exists a mapping $f : E \to C$ which maps each 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 $c(v)$.*

> Capacitated Vertex Cover: Given a vertex-weighted (with positive real numbers) and capacitated graph G, an integer $k \geq 0$, and a real number $W \geq 0$, determine whether there exists a capacitated vertex cover C for G containing at most k vertices such that $\sum_{v \in C} w(v) \leq W$.

Note that solving CVC becomes equivalent to solving VC if every vertex has a capacity that is at least its degree. Two special flavors of CVC exist in the literature that arise by allowing "copies" of a vertex to be in the capacitated vertex cover [19, 27, 30]. In that context, taking a vertex x-times into the capacitated vertex cover causes the vertex to have x-times its original capacity. The number of such copies is unlimited in the Soft Capacitated Vertex Cover (Soft CVC) problem while it may be restricted for each vertex individually in the Hard Capacitated Vertex Cover (Hard CVC) problem. Soft CVC was introduced by Guha et al. [30] who also give a factor-2 polynomial-time approximation algorithm. For unweighted Hard CVC, the best known polynomial-time approximation algorithm also achieves a factor of 2 [27]. The weighted version Hard CVC is at least as hard to approximate as Set Cover [19], so no constant-factor polynomial-time approximation algorithm can be expected for it.[4]

Section 5 considers a third type of VC variant besides demanding connectedness and introducing capacities—we relax the condition that *all* edges must be covered.

> Maximum Partial Vertex Cover: Given a graph $G = (V, E)$ and two integers $k \geq 0$ and $t \geq 0$, determine whether there exists a vertex subset $V' \subseteq V$ of size at most k such that V' covers at least t edges.

This problem was introduced by Bshouty and Burroughs [11] who showed it to be polynomial-time approximable within a factor of two. Depending on the maximum degree of the input graph, this approximation factor can be somewhat improved [31]. Note that Maximum Partial Vertex Cover is fixed-parameter tractable with respect to the parameter t [10]. We also consider the corresponding minimization problem Minimum Partial Vertex Cover where we are asked for a vertex subset with *at least* k vertices covering *at most* t edges.

3 Connected Vertex Cover and Variants

In this section we show that Connected Vertex Cover is fixed-parameter tractable with respect to the size of the connected vertex cover. More precisely, it can be solved by an algorithm running in $O(6^k n + 4^k n^2 + n^2 \log n + nm)$ time where n and m denote the number of vertices and edges in the input graph and k denotes the size of the connected vertex cover. We then modify this algorithm to also show the fixed-parameter tractability of the closely related variants Tree Cover and Tour Cover.

[4] Grandoni et al. [29] present a distributed approximation algorithm for weighted Hard CVC which achieves an approximation factor of $(2 + \epsilon)$ for any $\epsilon > 0$. However, this algorithm does not necessarily run in polynomial time since its running time depends on the ratio of the maximum to the minimum vertex weight in the graph.

For solving Connected Vertex Cover, we use the Dreyfus-Wagner algorithm for Steiner Minimum Tree as a subprocedure [23, 43].

> Steiner Minimum Tree (Steiner Tree): Given a graph $G = (V, E)$ where each edge is weighted by a positive real number, a real number $W \geq 0$, and a subset $K \subseteq V$, determine whether there exists a tree $T = (V', E')$ with $V' \subseteq V$, $E' \subseteq E$, and $K \subseteq V'$ such that $\sum_{e \in E'} w(e) \leq W$.

The vertices in K are called the *terminals* of T and every subtree of G that contains all vertices from K is called a *Steiner tree* for K in G. A *Steiner minimum tree* for K in G is a Steiner tree T such that the weight of edges contained in T is minimum. Steiner Tree is NP-complete [33]. The Dreyfus-Wagner algorithm computes a Steiner minimum tree in $O(3^{|K|}n + 2^{|K|}n^2 + n^2 \log n + nm)$ time [23, 43].[5]

Our algorithm for Connected Vertex Cover consists of two steps:

1. Enumerate all minimal vertex covers with at most k vertices. If one of the enumerated minimal vertex covers is connected, output it and terminate.

2. Otherwise, for each of the enumerated minimal vertex covers C, use the Dreyfus-Wagner algorithm to compute a Steiner minimum tree with C as the set of terminals. If one minimal vertex cover has a Steiner minimum tree T with at most $k - 1$ edges (i.e., weight at most $k - 1$ as the graph is unweighted), then return the vertex set of T as output; otherwise, there is no connected vertex cover with at most k vertices.

Theorem 1. Connected Vertex Cover *can be solved in* $O(6^k n + 4^k n^2 + n^2 \log n + nm)$ *time.*

Proof. The first step of the algorithm is correct since each connected vertex cover (covc) contains at least one minimal vertex cover as a subset. For a given graph, there are at most 2^k minimal vertex covers with at most k vertices. We can enumerate all such minimal vertex covers in $O(2^k n)$ time, i.e., the running time of the first step is $O(2^k n)$.

The correctness of the second step follows directly from the following easy to prove observation: For a set of vertices C, there exists a connected subgraph of G with at most k vertices which contains all vertices in C iff there exists

[5] Mölle et al. [35] state an improvement of the running time to $O((2 + \epsilon)^{|K|} \cdot n^{O(1)})$ for arbitrary $\epsilon > 0$. This directly translates into an algorithm that solves Connected Vertex Cover in $O((4 + \epsilon)^k \cdot n^{O(1)})$ time by Theorem 1. Despite this, we use the Dreyfus-Wagner algorithm for two reasons: First, the algorithm in [35] is infeasible in practice since it has huge constants hidden in the O-notation. Second, the recursion of the Dreyfus-Wagner algorithm can easily be adapted to also solve Tree Cover, giving a more unified presentation here.

a Steiner tree in G with C as the terminal set and at most $k-1$ edges. By applying the Dreyfus-Wagner algorithm to G with C as the terminal set, we can easily find out whether there are $k-|C|$ vertices from $V \setminus C$ connecting C and, hence, whether there is a covc with at most k vertices that contains C. Since $|C| < k$, the second step can be done in $O(2^k \cdot (3^k n + 2^k n^2) + n^2 \log n + nm) = O(6^k n + 4^k n^2 + n^2 \log n + nm)$ time. (The $O(n^2 \log n + nm)$ part in the running time of the Dreyfus-Wagner algorithm is due to a calculation of the distance matrix of the input graph [43], which only needs to be carried out once since G is not modified.) □

The algorithm for Connected Vertex Cover can be modified to solve Tree Cover. While the basic idea is the same (enumerate all vertex covers and—if necessary—add some vertices to form a tree cover), recall that the input graph for Tree Cover is *weighted* and hence it is not sufficient to simply find a Steiner minimum tree with a given vertex cover C as the set of terminal vertices: There is the additional constraint that the tree may not contain more than k vertices. Hence, we formulate the following modified Steiner Tree problem:

> Steiner Minimum k-Tree (Steiner k-Tree): Given an instance (G, W, K) of Steiner Tree and a positive integer $k \geq |K|$, determine whether there exists a Steiner tree for K in G of weight at most W which contains at most k vertices.

Lemma 2. Steiner k-Tree *can be solved in* $O(3^{|K|}k^2 n + 2^{|K|}k^2 n^2 + kn^3)$ *time.*

Proof. To show the lemma, we adapt the dynamic programming recursions of the Dreyfus-Wagner algorithm (closely following the presentation of Prömel and Steger in [43, Lemma 5.5]) to include restrictions on the overall tree size.

For a given instance (G, W, K, k) of Steiner k-Tree, a nonempty subset $X \subseteq K$, and $v \in V \setminus X$, let $s_j(X \cup \{v\})$ denote the weight of a Steiner minimum tree for $X \cup \{v\}$ that has at most j vertices. Furthermore, for two vertices $v, w \in V$, let $p_j(v, w)$ denote the weight of the shortest path between v and w that contains at most j vertices.[6]

We claim that the value s_j can be computed recursively, i.e.,

$$s_j(X \cup \{v\}) = \min_{0 \leq i < j} \Big(\min \big\{ \min_{w \in X} \{p_{i+1}(v, w) + s_{j-i}(X)\}, \min_{w \in V \setminus X} \{p_{i+1}(v, w) + s_{j-i,w}(X \cup \{w\})\} \big\} \Big), \tag{1}$$

where $s_{j,v}(X \cup \{v\})$ is defined as

$$s_{j,v}(X \cup \{v\}) := \min_{\substack{\emptyset \neq X' \subset X \\ 0 \leq i < j}} \{s_{i+1}(X' \cup \{v\}) + s_{j-i}((X \setminus X') \cup \{v\})\}. \tag{2}$$

[6] We let s_j, $s_{j,v}$, and p_j take a value of ∞ in the case that no corresponding tree (or path, respectively) exists.

In order to justify (1), assume that we have a Steiner tree T for $X \cup \{v\}$ which contains at most j vertices. On the one hand, if v is an internal vertex of T, then $s_j(X \cup \{v\}) = s_{j,v}(X \cup \{v\})$. This case is included in the recursion since we allow $v = w$. On the other hand, if v is a leaf of T then two cases need to be distinguished: Let w be the end of the longest path in T which starts at v and where all internal vertices have degree two. In the first case, we have $w \in X$, i.e., there exists $0 \leq i < j$ such that T can be decomposed into a Steiner minimum tree for X with at most $(j - i)$ vertices and a shortest path from w to v which contains at most $i+1$ vertices. In the second case—which is captured by (2)—we have $w \notin X$ which directly implies that w has degree at least three. Hence, there exists $0 \leq i < j$ such that T can be decomposed into a Steiner minimum tree for X with at most $(j - i)$ vertices (in which w has degree at least two) and a shortest path from w to v which contains at most $i + 1$ vertices.

To see the overall running time as claimed, note that $p_j(v, w)$ for all $1 \leq j \leq k$ and $v, w \in V$ can be computed in $k \cdot n^3$ time by successive matrix-multiplication (see [20, Section 25.1] for details). As to the start of the recursion, we have for all $u, v \in V$ and $0 < j < k$ that $s_j(\{u, v\}) = p_j(u, v)$. It remains to evaluate (1) and (2) for all $X \subseteq K$, $j \leq k$ and $v \in V \setminus X$. It is easy to verify that this upper-bounds the number of additions and comparisons in (2) by $3^{|K|}k^2n$ and that the number of tuples for which we have to evaluate (1) is upper-bounded by $2^{|K|}k^2n$. □

Theorem 3. Tree Cover *can be solved in* $O(6^k k^2 n + 4^k k^2 n^2 + kn^3)$ *time.*

Proof. With one exception to note, the algorithm is analogous to the algorithm for Connected Vertex Cover: We invoke the adapted Dreyfus-Wagner algorithm from Lemma 2 for every size-at-most-k vertex cover C enumerated in Step 1. (This is even done when C is connected since with weighted edges, taking additional vertices into a connected minimal vertex cover could result in a tree cover with a smaller total edge weight.) Overall, the running time is bounded by $O(2^k m + 2^k(3^k k^2 n + 2^k k^2 n^2) + kn^3) = O(6^k k^2 n + 4^k k^2 n^2 + kn^3)$. □

The algorithm solving Tour Cover consists also of two steps: enumerating all minimal vertex covers C with at most k vertices and adding at most $k - |C|$ vertices from $V \setminus C$ to C to form a minimum tour. Unfortunately, there appears to be no easily conceivable adaption of the Dreyfus-Wagner algorithm that we could make use of in the second step.[7] For this reason, the next theorem (Theorem 5) relies on an enumerative approach to establish the fixed-parameter tractability of Tour Cover. To this end, we make use of the following lemma, a generalization of a result by Goodman and Hedetniemi [28] to edge-weighted graphs.

[7] One reason for this is that the algorithm for Steiner k-Tree crucially relies on optimum partial solutions being vertex-disjoint except for one designated vertex v. This is clearly not true for a closed walk, where vertices can be used multiple times.

Lemma 4. *Given an n-vertex graph $G = (V, E)$ and a weight function $w : E \to \mathbb{R}^+$, every minimum-weight closed walk visiting all vertices in G traverses at most $2(n-1)$ edges.*

Proof. Instead of directly dealing with a walk W, we consider the *traversal mapping* $f_W : E \to \mathbb{N}_0$ which represents the number of times that an edge $e \in E$ is used by W. We argue that a mapping f_W corresponds to a closed walk if and only if the following two properties are satisfied:

1. The edges for which f_W is nonzero induce a connected subgraph in G.
2. For every vertex, the sum of f_W over all its adjacent edges is even.

Clearly, every closed walk satisfies these properties. To see that the converse also holds true, consider the induced subgraph as described in Property 1 and replicate every edge $e \in E$ exactly $f_W(e)$ times. Then, the resulting graph is an Eulerian multigraph and the corresponding Euler tour provides a closed walk in G [32].

Properties 1 and 2 directly imply that a *minimum-weight* closed walk in G traverses every edge at most twice. If this were not the case for some edge $e \in E$, then we could decrease the value of the traversal mapping for e by two without violating Property 1 or 2, thus obtaining a closed walk with less weight.

Consider a minimum-weight closed walk H visiting all vertices in G and let f_H be the corresponding traversal mapping. Clearly, H contains a spanning tree $T = (V, E_T)$ of G. Partition the edges used by H into four disjoint sets $E_T^1 := \{e \in E_T : f_H(e) = 1\}$, $E_T^2 := \{e \in E_T : f_H(e) = 2\}$, $E_{H\setminus T}^1 := \{e \in (E \setminus E_T) : f_H(e) = 1\}$, and $E_{H\setminus T}^2 := \{e \in (E \setminus E_T) : f_H(e) = 2\}$. Firstly, note that $E_{H\setminus T}^2$ must be empty if H is to have minimum weight because setting $f_H(e) = 0$ for all edges in this set still satisfies Properties 1 and 2 while yielding a smaller-weight closed walk. Secondly, note that the graph induced by $E_T^2 \cup E_{H\setminus T}^1$ does not contain any cycle; if it were to contain a cycle C, then decreasing f_H by one for each edge in C would still satisfy Properties 1 and 2, yielding a smaller-weight closed walk. This implies that $|E_T^2 \cup E_{H\setminus T}^1| \leq n - 1$ and thus the number of edge traversals that H makes is upper-bounded by

$$|E_T^1| + 2 \cdot |E_T^2| + |E_{H\setminus T}^1| = |E_T^1 \cup E_T^2| + |E_T^2 \cup E_{H\setminus T}^1| \leq 2(n-1). \qquad \square$$

Theorem 5. Tour Cover *can be solved in $O(2^k k^{2k} m)$ time.*

Proof. Our algorithm for Tour Cover is analogous to the one for Tree Cover which was presented in the last theorem, replacing only the step that solves Steiner k-Tree. More precisely, for a given vertex cover $C = \{v_1, \ldots, v_K\}$ (where $K \leq k$) of the input graph $G = (V, E)$, we now face the task of finding a set $\hat{C} = \{v_{K+1}, \ldots, v_k\}$ (which can be empty) such that $C \cup \hat{C}$ induces a tour of

minimum weight in G. We now show how a brute-force approach can solve this task in $O(k^{2k}m)$ time, which then yields the overall running time as claimed.

From Lemma 4 it is easy to see that a minimum weight tour that visits k vertices in a weighted graph traverses at most $2(k-1)$ edges. Hence, the minimum tour that we seek for solving Tour Cover can be coded as a *tour word* of length $2k-1$ over the alphabet $\{v_1, \ldots, v_K\} \cup \{v_{K+1}, \ldots, v_k\}$. Given one of these at most k^{2k-1} different tour words, it is straightforward to check in $O(m)$ time whether it represents a valid tour in the input graph and to assign vertices from $V \setminus C$ to $\{v_{K+1}, \ldots, v_k\}$ in such a way that the weight of the resulting tour is minimized:

1. We can w.l.o.g. assume that the tour word starts and stops with v_1 (since this vertex must be visited by the represented tour anyway). Also, we assume the tour word to be "cut short" after the point where its represented tour has visited all vertices in C and returned to v_1.

2. If two vertices from $\{v_1, \ldots, v_K\}$ are adjacent in the tour word, we can check in $O(m)$ time whether they are connected by an edge in G and determine the weight of that edge.

3. No two vertices from $\{v_{K+1}, \ldots, v_k\}$ must be adjacent in the tour word since the vertices in $V \setminus C$ are an independent set in G. Hence, for every vertex $\hat{v} \in \{v_{K+1}, \ldots, v_k\}$, the given tour word explicitly defines a set of edges from $\hat{v}$ to vertices in C that are traversed in the represented tour. It is possible in $O(m)$ time to check whether there exists a vertex in $V \setminus C$ that has the required neighborhood for $\hat{v}$. If more than one such vertex exists, we choose the one that minimizes the cost of the represented tour.

Using a brute-force approach to enumerate all possible tour words for at most 2^k vertex covers of the input graph, we can thus upper-bound the overall running time needed to solve Tour Cover by $O(2^k k^{2k} m)$. □

4 Capacitated Vertex Cover and Variants

In this section we present fixed-parameter algorithms for the Capacitated Vertex Cover (CVC) problem and its variants Hard CVC and Soft CVC. In the case of CVC, the easiest way to show its fixed-parameter tractability is to give a reduction to a problem kernel. This is what we begin with here, afterwards complementing this result with an enumerative approach for further improving the overall time complexity.

Theorem 6. *Given an n-vertex graph $G = (V, E)$ and an integer $k \geq 0$ as part of an input instance for* CVC*, it is possible to construct an $O(4^k \cdot k^2)$-vertex graph $\tilde{G}$ such that G has a size-k solution for* CVC *iff $\tilde{G}$ has a size-k solution for*

CVC. *In the special case of uniform vertex weights, $\tilde{G}$ has only $O(4^k \cdot k)$ vertices. The construction of $\tilde{G}$ can be performed in $O(n^2)$ time.*

Proof. We first assume uniform vertex weights, generalizing the approach to weighted graphs at the end of the proof.

Let $u, v \in V$, $u \neq v$, and $\{u, v\} \notin E$. The simple observation that lies at the heart of the data reduction rule needed for the kernelization is that if the open neighborhoods of u and v coincide (i.e., $N(u) = N(v)$) and $c(u) < c(v)$, then u is part of a minimum capacitated vertex cover only if v is as well. We can generalize this finding to a data reduction rule: Let the vertices $\{v_1, v_2, \ldots, v_{k+1}\} \subseteq V$ form an independent set in G such that $N(v_1) = N(v_2) = \cdots = N(v_{k+1})$. Call this the *neighbor set*. Then delete from G a vertex $v_i \in \{v_1, v_2, \ldots, v_{k+1}\}$ which has minimum capacity. This rule is correct because any size-k capacitated vertex cover C containing v_i can be modified by replacing v_i with a vertex from $\{v_1, v_2, \ldots, v_{k+1}\}$ which is not in C.

Based on this data reduction rule, $\tilde{G}$ can be computed from G as claimed by the following two steps:

1. Use the straightforward linear-time factor-2 approximation algorithm[8] to find a vertex cover S for G of size at most $2k'$ (where k' is the size of a minimum (not necessarily capacitated) vertex cover for G and hence $k' \leq k$). If $|S| > 2k$ we can stop, because then no size-k (capacitated) vertex cover can be found. Note that $V \setminus S$ induces an edgeless subgraph of G.

2. Examining $V \setminus S$, check whether there is a subset of $k + 1$ vertices that fulfill the premises of the above rule. Repeatedly apply the data reduction rule until it is no longer applicable. Note that this process continuously shrinks $V \setminus S$.

The above computation is clearly correct. The number of all possible neighbor sets can be at most 2^{2k} (the number of different subsets of S). For each neighbor set, there can be at most k neighboring vertices in $V \setminus S$; otherwise, the reduction rule would apply. Hence we can have at most $2^{2k} \cdot k$ vertices in the remaining graph $\tilde{G}$.

The generalization to non-uniform vertex weights works as follows: Again, we compute a vertex cover S for the input graph of size at most $2k$. Then, the vertices in $V \setminus S$ may have maximum vertex degree $2k$ and a capacity of a vertex in $V \setminus S$ greater than $2k$ can—without any harm—be replaced by a capacity of $2k$. Therefore, without loss of generality, one may assume that the maximum capacity of vertices in $V \setminus S$ is $2k$. We then have to modify the data reduction rule as follows. If there are vertices $v_1, v_2, \ldots, v_{2k^2+1} \in V$ with

[8]Note that the approximation factor refers to the cardinality and not to the weight of the cover.

$N(v_1) = N(v_2) = \cdots = N(v_{2k^2+1})$, partition them into subsets of vertices with equal capacity. There are at most $2k$ such sets; if one of these contains more than k vertices, delete a vertex with maximum weight. Altogether, we thus end up with a problem kernel of $2^{2k} \cdot 2k^2 = O(4^k \cdot k^2)$ vertices.

It remains to justify the polynomial running time. First, note that the trivial factor-2 approximation algorithm runs in $O(|E|) = O(n^2)$ time. Second, examining the common neighborhoods can be done in $O(n^2)$ time by successively partitioning the vertices in $V \setminus S$ according to their neighborhoods. □

Clearly, a simple brute-force search within the reduced instance (with a size of only $O(4^k \cdot k^2)$ vertices) already yields the fixed-parameter tractability of CVC. However, this would require the exploration of $O((4^k \cdot k^2)^k) = O(4^{k^2}k^{2k})$ possible solutions. As the next theorem shows, we can do much better concerning the running time.

Theorem 7. Capacitated Vertex Cover *can be solved in* $O(1.2^{k^2} + n^2)$ *time.*

The theorem is proven by first giving an algorithm to solve Capacitated Vertex Cover and then analyzing its running time. The basic idea behind the algorithm is as follows: Enumerate all minimal vertex covers $C = \{c_1, \ldots, c_i\} \subseteq V$ for the input graph $G = (V, E)$. Due to a possible lack of capacities, a minimal vertex cover C is not necessarily a *capacitated* vertex cover for G. In the case that C is *not* a capacitated vertex cover, we need to add some additional vertices from $V \setminus C$ to C in order to provide additional capacities. More precisely, since for each vertex $v \in (V \setminus C)$ all of its neighbors are in C, adding v can be seen as "freeing" exactly one unit of capacity for as many as $c(v)$ neighbors of v. Our algorithm uses an exhaustive search approach based on this observation: It enumerates all possible patterns of capacity-freeing and, for each of these patterns, computes the cheapest set of vertices from $V \setminus C$ (if one exists) that matches it.

Definition 2. *Given a graph* $G = (V, E)$ *and a vertex cover* $C = \{c_1, \ldots, c_i\} \subseteq V$ *for* G*, a* capacity profile *of length* i *is a binary string* $s = s[1] \cdots s[i] \in \{0, 1\}^i$*. A vertex* $w \in V \setminus C$ *is said to* match *a capacity profile* s *if it is incident to each vertex* $c_j \in C$ *with* $s[j] = 1$ *and its capacity is at least the number of ones in* s*.*

The concept of capacity profiles is illustrated in Figure 1. Using Definition 2, the pseudocode shown in Figure 2 gives an algorithm for CVC.

Lemma 8. *The algorithm for* Capacitated Vertex Cover *given in Figure 2 is correct.*

Proof. Preprocessing the graph in line *01* is correct according to Theorem 6. Since a capacitated vertex cover for a graph $G = (V, E)$ is also a vertex cover, its vertices can be partitioned into two sets C and $\hat{C}$ such that C is a minimal

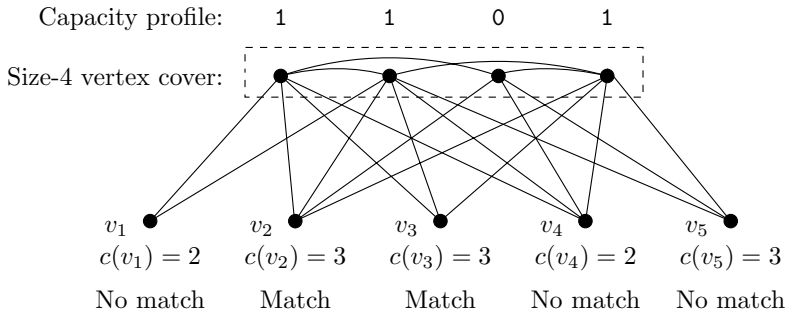

Figure 1: An illustration for the concept of capacity profiles: In the above graph, only the vertices v_2 and v_3 match the capacity profile `1101`. All other vertices lack either the neighborhood or the capacity to match this profile.

vertex cover for G. Assume the vertices in C to be ordered (arbitrarily but fixed). Each vertex in $\hat{C}$ gives additional capacity to a subset of the vertices in C, i.e., for every $\hat{c} \in \hat{C}$, we can construct a capacity profile $s_{\hat{c}}$ where $s_{\hat{c}}[j] = 1$ if and only if $\hat{c}$ uses its capacity to cover the edge to the j-th vertex in C. The correctness of the algorithm follows from its exhaustive nature: It tries all minimal vertex covers, all possible combinations of capacity profiles and for each combination it determines the cheapest possible set $\hat{C}$ such that $C \cup \hat{C}$ is a capacitated vertex cover for G. □

Lemma 9. *The algorithm for* Capacitated Vertex Cover *given in Figure 2 runs in* $O(1.2^{k^2} + n^2)$ *time.*

Proof. The preprocessing in line *01* can be carried out in $O(n^2)$ time according to Theorem 6. This leads to a new graph containing at most $\tilde{n} := O(4^k \cdot k^2)$ vertices. Line *02* of the algorithm causes the subsequent lines *03–07* to be called at most 2^k times (and causes only polynomial delay between two such calls). Due to [19], we can decide in $\tilde{n}^{O(1)}$ time whether a given vertex cover is also a capacitated vertex cover (lines *03* and *07*). For line *04*, note that for a given $0 \leq i < k$ there exist 2^i different capacity profiles of that length. Furthermore, it is well-known that given a set A with $|A| = a$, there exist exactly $\binom{a+b-1}{b}$ b-element multisets with elements drawn from A. Hence, line *04* causes lines *05–07* to be executed $\binom{2^i+(k-i)-1}{k-i}$ times. The delay between enumeration of two multisets can be kept constant. As it will be shown in Lemma 10, line *06* takes $\tilde{n}^{O(1)}$ time. Overall, the running time T_{CVC}

Algorithm: CAPACITATED VERTEX COVER
Input: A capacitated and vertex-weighted graph $G = (V, E)$, $k \in \mathbb{N}^+$, $W \in \mathbb{R}^+$
Output: "YES" if G has a capacitated vertex cover of size at most k with weight $\leq W$; "NO" otherwise

01 Perform the kernelization from Theorem 6 on G
02 **for** every minimal vertex cover C of G with size $i \leq k$ **do**
03 **if** C is a cap. vertex cover with weight $\leq W$ **then return** "YES"
04 **for** each multiset M consisting of $(k-i)$ length-i capacity profiles **do**
05 remove the all-zero profiles from M
06 find the cheapest size-$(k-i)$ set $\hat{C} \subseteq (V \setminus C)$ so that there exists a bijective mapping $f : \hat{C} \to M$ where each $\hat{c} \in \hat{C}$ matches the capacity profile $f(\hat{c})$. Set $\hat{C} \leftarrow \emptyset$ if no such set exists
07 **if** $\hat{C} \neq \emptyset$, the weight of $C \cup \hat{C}$ is $\leq W$, and $C \cup \hat{C}$ is a capacitated vertex cover for G **then return** "YES"
08 **return** "NO"

Figure 2: An algorithm that solves CAPACITATED VERTEX COVER in $O(1.2^{k^2} + n^2)$ time.

of the algorithm is thus bounded from above by

$$\begin{aligned}
T_{\text{CVC}} &\leq O(n^2) + 2^k \cdot \tilde{n}^{O(1)} \cdot \max_{1 \leq i < k} \left(\binom{2^i + (k-i) - 1}{k-i} \cdot \tilde{n}^{O(1)} \right) \\
&= O(n^2) + 2^k \cdot \max_{1 \leq i < k} \left(\frac{(2^i + (k-i) - 1)!}{(2^i - 1)!(k-i)!} \right) \cdot \tilde{n}^{O(1)} \\
&= O(n^2) + 2^k \cdot \max_{1 \leq i < k} \left(\frac{2^i + (k-i) - 1}{k-i} \cdot \ldots \cdot \frac{2^i}{1} \right) \cdot \tilde{n}^{O(1)} \\
&\overset{(*)}{<} O(n^2) + 2^k \cdot \max_{1 \leq i < k} \left((2^i)^{k-i} \right) \cdot \tilde{n}^{O(1)} \\
&\leq O(n^2) + 2^k \cdot \left(2^{(k^2-1)/4} \right) \cdot \tilde{n}^{O(1)} \\
&= O(n^2) + 2^k \cdot O(1.189^{k^2}) \cdot (4^k \cdot k^2)^{O(1)} \\
&= O(n^2 + 1.2^{k^2})
\end{aligned}$$

where $(*)$ is due to the fact that for all $j \geq 1$, we have $\frac{2^i + j - 1}{j} < 2^i$. □

It remains to show the running time for line *06* of the algorithm.

Lemma 10. *Given a weighted and capacitated n-vertex graph $G = (V, E)$, a vertex cover C for G of size $i \leq k$, and a multiset M of $k - i$ capacity profiles of*

length i, it takes $n^{O(1)}$ time to find the cheapest size-$(k-i)$ subset $\hat{C} \subseteq (V \setminus C)$ (or determine that no such set $\hat{C}$ exists) such that there exists a bijective mapping $f : \hat{C} \to M$ where each $\hat{c} \in \hat{C}$ matches the capacity profile $f(\hat{c})$.

Proof. Finding $\hat{C}$ is equivalent to finding a minimum weight maximum bipartite matching (that is, a bipartite matching of minimum weight among those of maximum cardinality) on the bipartite graph $G' = (V_1', V_2', E')$ where each vertex in V_1' represents a capacity profile from M, $V_2' := V \setminus C$, and two vertices $v \in V_1', u \in V_2'$ are connected by an edge in E' if and only if the vertex represented by u matches the profile represented by v (the weight of the edge is $w(u)$). Finding such a matching is well-known to be solvable in polynomial time [20]. □

It is possible to solve Soft CVC and Hard CVC by adapting the CVC algorithm: Observe that if we choose multiple copies of a vertex into the cover, each of these copies will have its own individual capacity profile. Thus, only line *06* of the CVC algorithm has to be modified to solve Soft CVC and Hard CVC.

Corollary 11. Soft CVC *and* Hard CVC *are solvable in $O(1.2^{k^2} + n^2)$ time.*

Proof. First, observe that the kernelization for Capacitated Vertex Cover (Theorem 6) also works for Soft CVC and Hard CVC since the crucial observation "if $N(u) = N(v)$ and $c(u) < c(v)$, then u is part of a minimum capacitated vertex cover only if v is as well" still holds for these two problems. Hence, it only remains to show how to change line *06* of the CVC algorithm.
For Soft CVC, the algorithm for line *06* as given in Lemma 10 can be replaced by a simple greedy-strategy that always takes the cheapest candidate for each profile. (In this way, the algorithm becomes faster than the original CVC algorithm because we do not have to compute a bipartite matching.)
For Hard CVC, the same bipartite matching algorithm as in Lemma 10 can be employed, with the modification that the vertex set V_2' in the bipartite graph G' contains as many representatives of each vertex $v \in V \setminus C$ as we are allowed to make copies of it. □

5 Maximum and Minimum Partial Vertex Cover

All Vertex Cover variants we studied in the previous two sections are known to have a polynomial-time constant-factor approximation (mostly factor-2) and all of them were shown to be fixed-parameter tractable. By way of contrast, we now present a result where a variant that has a polynomial-time factor-2 approximation is shown to be fixed-parameter intractable. More precisely, we show that Maximum Partial Vertex Cover (MaxPVC) is W[1]-complete with respect to the size k of the partial vertex cover by showing containment in W[1] and giving a parameterized reduction from the W[1]-complete Independent Set problem [22]. With the solution size as parameter, we also show the

W[1]-completeness of its minimization version MinPVC, employing a reduction from Clique.

In order to show that MaxPVC and MinPVC both are in W[1], we use the following "machine characterization" of W[1] by Chen, Flum, and Grohe [18].

Theorem 12 (Adapted from Definition 4 and Theorem 16 in [18]). *For a parameterized problem* P*, we have* $\mathrm{P} \in \mathrm{W}[1]$ *if and only if there exist a computable function* f*, a polynomial* $p(n)$*, and a nondeterministic RAM program deciding* P *such that for every run of the program on an instance* (x, k) *(where* $|x| = n$*),*

1. *it performs at most* $f(k) \cdot p(n)$ *steps;*
2. *at most the first* $f(k) \cdot p(n)$ *registers are used;*
3. *at every point of the computation, no register contains numbers strictly greater than* $f(k) \cdot p(n)$*;*
4. *all nondeterministic steps are among the last* $f(k)$ *steps.*

For more details on the computational model employed, we refer to [18].

Lemma 13. *Both* MaxPVC *and* MinPVC *are contained in* W[1].

Proof. Given a graph $G = (V, E)$ and integers k, t as input, a nondeterministic RAM program can decide if the corresponding instance of MaxPVC (MinPVC) has a solution with respect to the parameter k in three steps:

1. Preprocess the graph by storing its adjacency matrix in n^2 registers and storing the degree of each vertex in n registers. This takes $O(n^2)$ time.
2. Guess k vertices to put into the cover. This takes $O(k)$ time.
3. Let t' be the sum of the degrees of all guessed vertices. This sum can be calculated in $O(k)$ time. For each possible (unordered) pair of the guessed vertices, check whether they are connected by an edge. For each pair that is connected, subtract 1 from t'. This check can be performed in $O(k^2)$ time because in Step 1 we have stored the adjacency matrix in the registers of the RAM and we can thus check for edges in constant time.[9]

[9] Actually, keeping the time constant for an edge-check requires a little more care: Assume we have stored the adjacency matrix of $G = (V, E)$ in n^2 subsequent registers called *adjacency registers*. Let $V = \{v_1, \ldots, v_n\}$. Then, testing whether two vertices v_i and v_j are connected by an edge in E requires indirect addressing of the $(i \cdot n + j)$-th adjacency register. While indirect addressing is allowed in the RAM model used in [18], this is problematic from a run-time perspective because the RAM model does *not* allow for multiplication in constant time but only addition, subtraction, and rounded division by two. Thus, in order to quickly perform the indirect addressing of adjacency registers in Step 3, we need to pre-calculate the value $i \cdot n$ for every vertex v_i in Step 1 of the program. Clearly, this is doable in polynomial time.

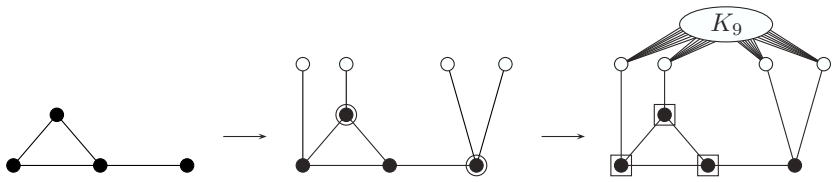

Figure 3: An illustration for the reductions given in the proofs of Theorems 14 and 15. For the given graph G, we wish to find an independent set of size two and a clique of size three. By adding the white vertices to G, we obtain G' as a MaxPVC instance. The two circled vertices are a size-2 solution of MaxPVC with $t = 6$ and correspond to a size-2 independent set in G. Graph G'' is the MinPVC instance (K_9 denotes a clique of nine vertices), the three boxed vertices are a size-3 solution of MinPVC with $t := 6$ and correspond to a size-3 clique in G.

In the case of MaxPVC, it remains to output "yes" if $t' \geq t$ and "no" otherwise. For MinPVC, we output "yes" if $t' \leq t$ and "no" otherwise. Clearly, the described program meets the requirements of Theorem 12 and hence we have established that MaxPVC $\in$ W[1] and MinPVC $\in$ W[1]. □

It remains to show the W[1]-hardness for MaxPVC and MinPVC. For MaxPVC, we employ a parameterized reduction from the W[1]-complete Independent Set problem [22].

> Independent Set: Given a graph $G = (V, E)$ and an integer $k \geq 0$, determine whether there is a vertex subset $I \subseteq V$ with at least k vertices such that the subgraph of G induced by I contains no edge.

With *independent set*, we denote a set of mutually nonadjacent vertices in a given graph.

Theorem 14. Maximum Partial Vertex Cover *is* W[1]*-complete with respect to the size of the cover.*

Proof. Containment in W[1] has been shown in Lemma 13. To show W[1]-hardness, we give a parameterized reduction from Independent Set to MaxPVC. Given an input instance $(G = (V, E), k)$ of Independent Set, let $\deg(v)$ denote the degree of a vertex v in G. We construct a new graph $G' = (V', E')$ in the following way: For each vertex $v \in V$ we insert $|V| - 1 - \deg(v)$ new vertices into G and connect each of these new vertices with v. Thus, all vertices in V have degree $|V| - 1$ in G'. Figure 3 gives an example for this construction. In the following, we show that a size-k independent set in G one-to-one corresponds to a size-k partial vertex cover in G' which covers at least $t := k \cdot (|V| - 1)$ edges.

First, a size-k independent set in G also forms a size-k independent set in G'. Moreover, each of these k vertices has exactly $|V| - 1$ incident edges. Then, these k vertices form a partial vertex cover that covers $k \cdot (|V| - 1)$ edges.

Second, if we have a size-k partial vertex cover in G' which covers $k \cdot (|V| - 1)$ edges, then we know that none of the newly inserted vertices in G' can be in this cover. Hence, this cover contains k vertices from V. Moreover, a vertex in G' can cover at most $(|V| - 1)$ edges and two adjacent vertices can cover no more than $2|V| - 3$ edges. Therefore, no two vertices in the partial vertex cover can be adjacent, which implies that it forms a size-k independent set in G. □

In the case of Minimum Partial Vertex Cover (MinPVC), we wish to choose *at least* k vertices such that *at most* t edges are covered. Through a parameterized reduction from the W[1]-complete Clique problem [22], it is possible to show—analogously to MaxPVC—that MinPVC is also W[1]-hard.

> Clique: Given a graph $G = (V, E)$ and an integer $k \geq 0$, determine whether there is a vertex subset $K \subseteq V$ with at least k vertices such that the subgraph of G induced by K forms a complete graph.

A complete graph is called a *clique*. The reduction works in a similar way as the reduction in the proof above.

Theorem 15. Minimum Partial Vertex Cover *is* W[1]*-complete with respect to the size of the cover.*

Proof. Containment in W[1] has been shown in Lemma 13. To show W[1]-hardness, we give a parameterized reduction from Clique to MinPVC. Given a Clique-instance $(G = (V, E), k)$, we construct a graph $G' = (V', E')$ as described in the proof of Theorem 14. In addition, we add a clique consisting of $k \cdot (|V| - 1)$ new vertices into G', with V'' denoting the vertex set of this clique. Then, we insert edges between all pairs of vertices in $V' \setminus V$ and V''. The resulting graph is denoted by G'', Figure 3 gives an example for the construction. We now show that a size-k clique in G one-to-one corresponds to a size-k partial vertex cover in G'' which covers at most $t := k \cdot (|V| - 1) - k \cdot (k - 1)/2$ edges.

On the one hand, it is easy to verify that each size-k clique C of G covers $k \cdot (|V| - 1) - k \cdot (k - 1)/2$ edges in G''. On the other hand, if there exists a partial vertex cover C with k vertices that covers $k \cdot (|V| - 1) - k \cdot (k - 1)/2$ edges in G'', then none of the vertices in C can be in $V'' \cup (V' \setminus V)$ since all vertices in $V'' \cup (V' \setminus V)$ have at least degree $k \cdot (|V| - 1)$ in G''. Thus, all vertices in C are from V. Since each vertex in V has exactly $|V| - 1$ incident edges, the number of edges in the subgraph of G induced by C is $k \cdot (k - 1)/2$. Hence, the vertices in C form a size-k clique of G. □

6 Conclusion

We extended the parameterized complexity picture for natural variants and generalizations of Vertex Cover. Table 1 in Section 2 summarizes our new results. Notably, whereas the fixed-parameter tractability of Vertex Cover immediately follows from a simple search tree strategy, this does not appear to be the case for all of the problems studied here.

Our fixed-parameter tractability results clearly generalize to cases where the vertices have real weights $\geq c$ for some given constant $c > 0$ and the parameter becomes the weight of the desired vertex cover (see [40] for corresponding studies for weighted Vertex Cover).

It is a task for future research to significantly improve on the presented worst-case running times (exponential factors in parameter k).[10] In particular, it would be interesting to learn more about the amenability of the considered problems to problem kernelization by (more) efficient data reduction techniques.

Besides the significant interest in the studied problems on their own, we want to mention one more feature of our work that lies a little aside. Adding our results to the already known large arsenal of facts about Vertex Cover, this problem can be even better used and understood as a seed problem for parameterized complexity as a whole: New aspects now directly related to vertex covering by means of our results are issues such as enumerative techniques, dynamic programming, or parameterized intractability. This might be of particular use when learning or teaching parameterized complexity through basically one natural and easy to grasp problem—Vertex Cover—and its "straightforward" variants.

Acknowledgments. We are grateful to an anonymous referee of *WADS 2005* for spotting a flaw in a previous proof of the fixed-parameter tractability of Connected Vertex Cover. An anonymous referee greatly helped us in simplifying the proof of Lemma 4. We thank Leizhen Cai for informing us about his independent hardness results for the Partial Vertex Cover problems. We greatly profited from constructive feedback during the Dagstuhl Seminar 05301 "Exact Algorithms and Fixed-Parameter Tractability" organized by Rod Downey, Martin Grohe, Michael Hallett, and Gerhard Woeginger in July 2005 where Jörg Flum, Catherine McCartin, and Gerhard Woeginger pointed us to the containment of MaxPVC and MinPVC in W[1].

References

[1] F. N. Abu-Khzam, R. L. Collins, M. R. Fellows, M. A. Langston, W. H. Suters, and C. T. Symons. Kernelization algorithms for the Vertex Cover

[10] First results in this direction concerning Connected Vertex Cover and Tree Cover appear in work of Mölle et al. [36].

problem: theory and experiments. In *Proceedings of the 6th Workshop on Algorithm Engineering and Experiments (ALENEX'04)*, pages 62–69. ACM/SIAM, 2004.

[2] F. N. Abu-Khzam, M. A. Langston, P. Shanbhag, and C. T. Symons. Scalable parallel algorithms for FPT problems. To appear in *Algorithmica*, 2006.

[3] 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.

[4] J. Alber, F. Dorn, and R. Niedermeier. Experimental evaluation of a tree decomposition based algorithm for Vertex Cover on planar graphs. *Discrete Applied Mathematics*, 145(2):219–231, 2005.

[5] J. Alber, H. Fernau, and R. Niedermeier. Parameterized complexity: exponential speed-up for planar graph problems. *Journal of Algorithms*, 52:26–56, 2004.

[6] J. Alber, J. Gramm, and R. Niedermeier. Faster exact algorithms for hard problems: a parameterized point of view. *Discrete Mathematics*, 229:3–27, 2001.

[7] E. M. Arkin, M. M. Halldórsson, and R. Hassin. Approximating the tree and tour covers of a graph. *Information Processing Letters*, 47(6):275–282, 1993.

[8] V. Arvind and V. Raman. Approximation algorithms for some parameterized counting problems. In *Proceedings of the 13th International Symposium on Algorithms and Computation (ISAAC'02)*, volume 2518 of *LNCS*, pages 453–464. Springer, 2002.

[9] R. Balasubramanian, M. R. Fellows, and V. Raman. An improved fixed-parameter algorithm for Vertex Cover. *Information Processing Letters*, 65(3):163–168, 1998.

[10] M. Bläser. Computing small partial coverings. *Information Processing Letters*, 85(6):327–331, 2003.

[11] N. H. Bshouty and L. Burroughs. Massaging a linear programming solution to give a 2-approximation for a generalization of the Vertex Cover problem. In *Proceedings of the 15th Symposium on Theoretical Aspects of Computer Science (STACS'98)*, volume 1373 of *LNCS*, pages 298–308. Springer, 1998.

[12] L. Cai. Parameterized complexity of cardinality constrained optimization problems. Manuscript, May 2005.

[13] L. Cai and D. Juedes. On the existence of subexponential parameterized algorithms. *Journal of Computer and System Sciences*, 67(4):789–807, 2003.

[14] L. S. Chandran and F. Grandoni. Refined memorisation for Vertex Cover. *Information Processing Letters*, 93(3):125–131, 2005.

[15] J. Cheetham, F. Dehne, A. Rau-Chaplin, U. Stege, and P. J. Taillon. Solving large FPT problems on coarse-grained parallel machines. *Journal of Computer and System Sciences*, 67(4):691–706, 2003.

[16] J. Chen, I. A. Kanj, and W. Jia. Vertex Cover: further observations and further improvements. *Journal of Algorithms*, 41:280–301, 2001.

[17] J. Chen, I. A. Kanj, and G. Xia. Simplicity is beauty: improved upper bounds for Vertex Cover. Technical Report 05-008, DePaul University, Chicago, 2005.

[18] Y. Chen, J. Flum, and M. Grohe. Machine-based methods in parameterized complexity theory. *Theoretical Computer Science*, 339(2-3):167–199, 2005.

[19] J. Chuzhoy and J. S. Naor. Covering problems with hard capacities. To appear in *SIAM Journal on Computing*, 2006.

[20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, 2nd Edition*. MIT Press, 2001.

[21] P. Damascheke. Parameterized enumeration, transversals, and imperfect phylogeny reconstruction. *Theoretical Computer Science*, 351(3):337–350, 2006.

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

[23] S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1:195–207, 1972.

[24] M. R. Fellows. New directions and new challenges in algorithm design and complexity, parameterized. In *Proceedings of the 8th Workshop on Algorithms and Data Structures (WADS'03)*, volume 2748 of *LNCS*, pages 505–520. Springer, 2003.

[25] H. Fernau. On parameterized enumeration. In *Proceedings of the 8th Annual International Computing and Combinatorics Conference (COCOON'02)*, volume 2383 of *LNCS*, pages 564–573. Springer, 2002.

[26] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer-Verlag, 2006.

[27] R. Gandhi, E. Halperin, S. Khuller, G. Kortsarz, and A. Srinivasan. An improved approximation algorithm for Vertex Cover with hard capacities. *Journal of Computer and System Sciences*, 72:16–33, 2006.

[28] S. E. Goodman and S. T. Hedetniemi. On Hamiltonian walks in graphs. *SIAM Journal on Computing*, 3(3):214–221, 1974.

[29] F. Grandoni, J. Könemann, A. Panconesi, and M. Sozio. Primal-dual based distributed algorithms for Vertex Cover with semi-hard capacities. In *Proceedings of the 24th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC'05)*, pages 118–125, 2005.

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

[31] E. Halperin and A. Srinivasan. Improved approximation algorithm for the Partial Vertex Cover problem. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX'02)*, volume 2462 of *LNCS*, pages 161–174. Springer, 2002.

[32] C. Hierholzer. Über die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren. *Mathematische Annalen*, 6:30–32, 1873.

[33] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[34] J. Könemann, G. Konjevod, O. Parekh, and A. Sinha. Improved approximations for tour and tree covers. *Algorithmica*, 38(3):441–449, 2004.

[35] D. Mölle, S. Richter, and P. Rossmanith. A faster algorithm for the Steiner Tree problem. In *Proceedings of the 23rd Symposium on Theoretical Aspects of Computer Science (STACS'06)*, volume 3884 of *LNCS*, pages 561–570. Springer, 2006.

[36] D. Mölle, S. Richter, and P. Rossmanith. Enumerate and expand: Improved algorithms for Connected Vertex Cover and Tree Cover, To appear in *Proceedings of the 1st International Computer Science Symposium in Russia (CSR'06)*, LNCS, Springer, 2006.

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

[38] R. Niedermeier and P. Rossmanith. Upper bounds for Vertex Cover further improved. In *Proceedings of the 16th Symposium on Theoretical Aspects*

of Computer Science (STACS'99), volume 1563 of *LNCS*, pages 561–570. Springer, 1999.

[39] R. Niedermeier and P. Rossmanith. A general method to speed up fixed-parameter-tractable algorithms. *Information Processing Letters*, 73:125–129, 2000.

[40] R. Niedermeier and P. Rossmanith. On efficient fixed-parameter algorithms for Weighted Vertex Cover. *Journal of Algorithms*, 47(2):63–77, 2003.

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

[42] E. Prieto and C. Sloper. Either/or: using vertex cover structure in designing FPT-algorithms—the case of k-Internal Spanning Tree. *Nordic Journal of Computing*, 12(3):308–318, 2005.

[43] H. J. Prömel and A. Steger. *The Steiner Tree Problem*. Vieweg-Verlag, 2002.

[44] C. D. Savage. Depth-first search and the vertex cover problem. *Information Processing Letters*, 14(5):233–237, 1982.