An $O(pn + 1.151^p)$-Algorithm for p-Profit Cover and its Practical Implications for Vertex Cover*

Ulrike Stege[1], Iris van Rooij[2], Alex Hertel[1], and Philipp Hertel[1]

[1] Dept. of Comp. Sc., Univ. of Victoria, Victoria B.C.
stege@cs.uvic.ca, awkkh@shaw.ca
[2] Dept. of Psych., Univ. of Victoria, Victoria B.C., irisvr@uvic.ca

Abstract. We introduce the problem PROFIT COVER which finds application in, among other areas, psychology of decision-making. A common assumption is that *net value* is a major determinant of human choice. PROFIT COVER incorporates the notion of net value in its definition. For a given graph $G = (V, E)$ and an integer $p > 0$, the goal is to determine $PC \subseteq V$ such that the profit, $|E'| - |PC|$, is at least p, where E' are the by PC covered edges. We show that p-PROFIT COVER is a parameterization of VERTEX COVER. We present a fixed-parameter-tractable (fpt) algorithm for p-PROFIT COVER that runs in $O(p|V| + 1.150964^p)$. The algorithm generalizes to an fpt-algorithm of the same time complexity solving the problem p-EDGE WEIGHTED PROFIT COVER, where each edge $e \in E$ has an integer weight $w(e) > 0$, and the profit is determined by $\sum_{e \in E'} w(e) - |PC|$. We combine our algorithm for p-PROFIT COVER with an fpt-algorithm for k-VERTEX COVER. We show that this results in a more efficient implementation to solve MINIMUM VERTEX COVER than each of the algorithms independently.

1 Introduction

We introduce the *profit problem* PROFIT COVER and study its classical and parameterized complexity. PROFIT COVER is an adaptation of the graph problem VERTEX COVER. We define the optimization version of this well known graph problem. For a given graph[3] $G = (V, E)$, a subset $V' \subseteq V$ is called a *vertex cover* for G if for each edge $(u, v) \in E$, $u \in V'$ or $v \in V'$.

MINIMUM VERTEX COVER (MVC)
Input: A graph $G = (V, E)$.
Output: A *minimum* vertex cover $V' \subseteq V$ for G (i.e., a vertex cover V' for G where $|V'|$ is minimized over all possible vertex covers for G).

VERTEX COVER (VC) is known to be NP-complete [9]. Also its natural parameterization k-VERTEX COVER (k-VC) is known to be fixed-parameter tractable [4, 7] (here k denotes the size of the vertex cover to be determined). The fastest

* This research is supported by a UVic research grant and by NSERC grant 54194.
[3] All graphs considered in this article are simple and undirected.

known fixed-parameter-tractable algorithm that solves k-VC has a running time of $O(|V|k + 1.285^k)$ [3].

To illustrate the characteristic nature of a profit problem, we consider a problem with two goals: (1) Find a set of vertices that covers as many edges as possible and (2) find a set of vertices that contains as few vertices as possible. Satisfying either goal is trivial. However, if we want to *satisfy both goals at the same time* we are confronted with the trade-off between the number of edges to cover and the number of vertices needed to do that. MVC is a special case of the problem described above. In MVC, goal (1) is given priority over goal (2), such that the final goal is to cover *all* edges with as *few* vertices as possible. When we consider MAXIMUM PROFIT COVER then the same two goals are given, but neither takes precedence over the other. The goal is to determine a subset $V' \subseteq V$ such that $|E'| - |V'|$ is maximized, where $E' \subseteq E$ are the edges covered by the vertices in V'. $|E'| - |V'|$ is called the *profit* of V' for G.

MAXIMUM PROFIT COVER (MPC)

Input: A graph $G = (V, E)$.

Question: A *maximum profit cover* $V' \subseteq V$ (i.e., a subset $V' \subseteq V$ where $\text{profit}_{\text{PC},G}(V') = |\text{E}_{\text{PC},G}(V')| - |V'|$ is maximized over all possible subsets of $V' \subseteq V$. Here, $(u, v) \in \text{E}_{\text{PC},G}(V')$ if $(u, v) \in E$ and ($u \in V'$ or $v \in V'$)).

The decision version of this problem is as follows.

PROFIT COVER (PC)

Input: A graph $G = (V, E)$, and an integer $p > 0$.

Question: Does there exist a subset $V' \subseteq V$ with $\text{profit}_{\text{PC},G}(V') \geq p$?

Even though the questions asked in VC and PC are different, the problems are closely related and thus the complexity of PC is of interest to fields in which VC finds its application (e.g., data cleaning for multiple sequence alignments [4, 21] and phylogeny compatibility in computational biology [19]). We also describe a new field of research in which VC, and PC in particular, find a natural application, viz. the psychology of decision-making (see [12] for a review). In this field, EDGE WEIGHTED PROFIT COVER instantiates a useful generalization of PC.

EDGE WEIGHTED PROFIT COVER (EWPC)

Input: A graph $G = (V, E)$, where each edge $e \in E$ is associated a integer weight $\text{w}(e) > 0$, integer $p > 0$.

Question: Does there exist a subset $V' \subseteq V$ with $\text{profit}_{\text{EWPC},G}(V') \geq p$? Here, $\text{profit}_{\text{EWPC},G}(V') = \sum_{e \in \text{E}_{\text{PC},G}(V')} \text{w}(e) - |V'|$?[4]

1.1 Graph Problems in Human Decision Making

VC, PC and EWPC model choice situations that humans may encounter in their everyday live. We discuss a scheduling problem as an illustration. Suppose a student (DM) is planning a schedule for the upcoming year. There are a number of activities that DM wants to undertake (e.g., attend a conference, take a course), but for some pairs of these activities there exists a conflict in DM's world (e.g.,

[4] As in PC, in EWPC $(u, v) \in \text{E}_{\text{PC},G}(V')$ if (1) $(u, v) \in E$ and (2) $u \in V'$ or $v \in V'$. For simplicity, for the weight of an edge (u, v) we write $\text{w}(u, v)$ instead of $\text{w}((u, v))$.

a conference overlaps with a course). This situation can be modeled as follows. Let $G = (V, E)$ be a graph such that the vertices represent the activities and each edge (u, v) represents a conflict between u and v. Assume that DM wants to exclude as few activities as possible from his or her schedule. Also, DM wants to reduce the amount of conflict as much as possible. Since both goals cannot be satisfied at the same time, DM has to formulate a goal that can be satisfied. Let DM set as a goal to find *at most* k activities to exclude, such that there is *no conflict* in the schedule. Then DM's choice situation is modeled by VC. Alternatively, DM can be interested in excluding a set of activities such that DM makes at least p *profit*, where the profit is considered to be the number of conflicts resolved *minus* the number of activities excluded. This choice situation is modeled by PC. In many real-world situations the value of activities and the degree of conflict between pairs of activities may vary. Consider the situation where the degree of conflict differs for different pairs of activities, but the value of each activity is the same. If the goal is to establish a satisfactory profit by excluding activities, then EWPC models the choice situation.

We have introduced and motivated the problems PC and EWPC. Complexity analyses of such everyday human decision problems are relevant for the important topic of human rationality [13, 14]. Specifically, complexity theory can aid psychology in understanding the limitations of human rationality [18, 20].

1.2 Overview

We show NP-completeness of the problems PC and EWPC (Section 3). We further show that p-Profit Cover is a parameterization for VC. In Section 4, we show both PC and EWPC have linear problem kernels for parameter p (of size $2p$ for connected graphs and of size $3p-3$ for general graphs) and thus both problems are in FPT. We then extend the algorithm by a bounded search-tree technique and re-kernelization. The resulting running time is $O(p|V|+1.150964^p)$ (Section 4.2). Section 5 discusses how to combine fpt-algorithms for different parameterizations of a problem. As an example we discuss how the combination of an algorithm for k-VC and our p-PC algorithm results in a practical and efficient implementation solving the problems MVC and MPC.

2 Notation and Terminology

We assume basics in graph theory, algorithms, and complexity theory [9, 11]. Let $G = (V, E)$ be a graph with vertex set V and edge set E. We define the *neighborhood* $\mathrm{N}_G(v)$ of a vertex $v \in V$ as the set of all vertices $u \in V$ with $(u, v) \in E$. The *degree* of a vertex $v \in V$ is denoted by $\deg_G(v)$, where $\deg_G(v) = |\mathrm{N}_G(v)|$. The *set difference* of two sets V and W is denoted by $V \setminus W = \{v \in V | v \notin W\}$. For a graph $G = (V, E)$ and a set V', $G - V'$ denotes the graph $G^* = (V^*, E^*)$, such that $V^* = V \setminus V'$ and $E^* = E \setminus E'$ with $E' = \{(u, v) \in E | u \in V' \text{ or } v \in V'\}$. If $V' = \{v\}$ we simply write $G - v$ instead of $G - V'$. A graph $S = (V_S, E_S)$ is a *subgraph* of a graph $G = (V, E)$ if $V_S \subseteq V$ and $E_S \subseteq E$. Let $S = (V_S, E_S)$ be a subgraph of $G = (V, E)$. Then $G - S$ denotes the graph $G - V_S$. For a rooted tree $T = (V_T, E_T)$ that consists of more than 2 vertices we assume that root r

Branching vector	Estimation for $\|r\|$
[3,8]	1.146139
[4,6]	1.150964
[4,7]	1.138182
[5,6]	1.134724

Table 1. Branching vectors and their corresponding $|r|$.

is of degree $\deg_T(r) \geq 2$. Let $v, w \in V_T$ such that w is parent of v. Then T_v denotes the by v *induced subtree* of T if T_v is the connected component of $T - (v, w)$ that contains vertex v.

Parameterized complexity was introduced by Downey and Fellows [4]. For introductory surveys see [1, 5, 6, 8]. We denote instances of *parameterized decision problems* with (I, k) where I is the non-parameterized input and k denotes the problem parameter. Furthermore, (I, k) is called a *yes-instance* if the answer to the question asked in the problem is *yes*. If the answer is *no*, we call (I, k) a *no-instance*. A parameterized decision problem is called *fixed-parameter tractable* (in short *fpt*), if for every instance (I, k) it can be solved in time $\text{pol}(|I|) + f(k)$ where pol is a polynomial function and f is a function depending on k only. The complexity class containing all the fpt decision problems is denoted by FPT.[5]

A *problem kernel* for an instance (I, k) is $|I'|$ where (I', k') is the result of applying a polynomial-time algorithm on (I, k) such that $|I'| = f(k)$ for a function f [4]. Such a technique is called *kernelization*. It is known that a parameterized decision problem is in FPT if and only if it is kernelizable [5].

In this paper we also make use of the *bounded search-tree technique*. The goal is to maintain a bounded search tree for the different possible solutions of a given problem instance (I, k). A bounded search tree is a rooted tree bounded in the size by a function $f(k)$. We call the vertices in a search tree *nodes*. The nodes of the search tree are labeled by k-solution candidate sets. Since the size of the search tree depends only on the parameter, the search tree becomes constant size for fixed k. To estimate the size of such a search-tree we use linear recurrence relations with constant coefficients. For an fpt-algorithm and a search tree, we call $[b_1, b_2, \ldots, b_s]$ a *branching vector*[6] of *length* s, if the algorithm recursively makes calls for parameters of sizes $k - b_1$, $k - b_2, \ldots$, and $k - b_s$. This corresponds to the recurrence $t_k = t_{k-b_1} + t_{k-b_2} + \ldots + t_{k-b_s}$ with the characteristic polynomial $x^b = x^{b-b_1} + x^{b-b_2} + \ldots + x^{b-b_s}$ (here b is the maximum value of all b_i, $i \in \{1, \ldots, s\}$). If the characteristic polynomial has a single root r, then we can estimate the search-tree size with $O(|r|^k)$ [10]. For a list of branching vectors and their corresponding estimated search-tree size used in this article see Table 1.

The combination of kernelization and bounded search-tree techniques leads to a natural running time of $O(\text{pol}(|I|) + f(k))$. In the case of an $O(|r|^k)$ search-tree size, a problem kernel of size $O(k^\alpha)$ and a kernelization time of $O(kn)$, a typical running time might be $O(kn + |r|^k k^\alpha)$. The technique *re-kernelization*, described first in [22], repeats the kernelization step after each branching in the search tree.

[5] Parameterized decision problems that are not in FPT (unless W[1] = FPT) are called W[1]-hard, where W[1] $\supseteq$ FPT [4].

[6] The term branching vector was first introduced in [16]. For a first fpt-algorithm using this technique to estimate search trees see [2].

Niedermeier and Rossmanith presented a better time-complexity analysis for a search-tree algorithm using re-kernelization [17]. Using their analysis the running time improves to $O(kn + |r|^k)$.

3 NP-completeness of PC and EWPC

We show NP-completeness for PC via a reduction from VC. As a consequence, the more general problem EWPC is also NP-complete.[7] Before proving NP-completeness of PROFIT COVER we observe that for a given graph $G = (V, E)$ for each subset $PC \subseteq V$ with profit p, there exists a vertex cover $V' \supseteq PC$, $V' \subseteq V$, with profit at least p.

Observation 1. *Let $G = (V, E)$ be a graph and $V' \subseteq V$ with $\text{profit}_{\text{EWPC},G}(V') = p$. If there exists $(u, v) \in E$ with $u, v \notin V'$, then $\text{profit}_{\text{EWPC},G}(V' \cup \{u\}) \geq p$.*

Theorem 1. PROFIT COVER *is NP-complete.*

Proof. We reduce from VC. An instance for VC is given by a graph $G = (V, E)$ and an integer $k > 0$. We show, G has a vertex cover $V' \subseteq V$ of size k for G if and only if there is a subset $PC \subseteq V$ with $\text{profit}_{\text{PC},G}(PC) = |E| - k$. Let $V' \subseteq V$ be a vertex cover for G of size k. Then $\text{E}_{\text{PC},G}(V') = E$ and therefore $\text{profit}_{\text{PC},G}(V') = |\text{E}_{\text{PC},G}(V')| - |V'| = |E| - k$. Conversely, let $PC \subseteq V$ with $\text{profit}_{\text{PC},G}(PC) = p$. We distinguish two cases: (1) Let $\text{E}_{\text{PC},G}(PC) = E$. We rewrite $p = |E| - k$. We can conclude that $|PC| = k$ and thus PC is a vertex cover of size k. (2) Let $\text{E}_{\text{PC},G}(PC) \neq E$. We extend PC to a vertex cover V' by applying Observation 1 as long as there exist uncovered edges in G. Thus, $\text{profit}_{\text{PC},G}(V') \geq \text{profit}_{\text{PC},G}(PC) \geq p$. Then $\text{E}_{\text{PC},G}(V') = E$ and the vertex-cover size is $k = |V'| \leq |E| - \text{profit}_{\text{PC},G}(V') \leq |E| - p$.

Corollary 1. EDGE WEIGHTED PROFIT COVER *is NP-complete.*

The algorithm described in the proof above computes, in linear time, a vertex cover of size $k = |E| - p$ for a given set with profit p in G; especially we receive a minimum vertex cover for a given maximum profit cover.

4 Fast fpt-algorithms for p-PC and p-EWPC

In this section we present an fpt-algorithm for p-EWPC, the naturally parameterized version of EWPC. Note that PC is a special case of EWPC. Thus the algorithm applies also for p-PROFIT COVER (p-PC).

p-EDGE WEIGHTED PROFIT COVER (p-EWPC)
Input: A graph $G = (V, E)$, each edge $e \in E$ has a positive weight $\text{w}(e) \in \mathbb{N}\backslash\{0\}$, integer $p > 0$.
Parameter: p
Question: Does there exist a subset $V' \subseteq V$ with $\text{profit}_{\text{EWPC},G}(V') \geq p$?

We first show that p-EWPC has a linear problem kernel and then present a bounded search-tree algorithm for p-EWPC which results in a time complexity of $O(p|V| + 1.150964^p)$.

[7] It is easily verified that both PC and EWPC are members of the class NP.

4.1 A linear problem kernel for p-EWPC

We present a linear-sized problem kernel for p-EWPC that we can construct in linear time.[8] We start by stating the Subgraph Lemma, the key property used to show the existence of a problem kernel ($|V| \leq 2p$) for connected graphs. We then present the six reduction rules that, together with the Component Lemma (cf. page 8), allow us to conclude a problem kernel for general graphs ($|V| \leq 3p-3$).

Lemma 1. (Subgraph Lemma) *Let $G = (V, E)$ be an edge-weighted graph and $S = (V_S, E_S)$ be a subgraph of G. If there exists a set V', $V' \subseteq V_S$, with $\text{profit}_{\text{PC},S}(V') = p$ then $\text{profit}_{\text{EWPC},G}(V') \geq p$.*

(K 1) If $v \in V$ with $\deg_G(v) = 0$ in an edge-weighted graph $G = (V, E)$ then (G, p) is a yes-instance for p-EWPC if and only if $(G - v, p)$ is a yes-instance for p-EWPC.

(K 2) Let $G = (V, E)$ be an edge-weighted graph and let $v \in V$ with $\deg_G(v) = 1$ and $\text{N}_G(v) = \{w\}$. Then (G, p) is a yes-instance for p-EWPC if and only if $(G - \{v, w\}, p - \text{profit}_{\text{EWPC},G}(\{w\}))$ is a yes-instance instance for p-EWPC.

(K 3) Let $G = (V, E)$ be an edge-weighted graph where neither (K 1) nor (K 2) apply. Let $v \in V$ with $\text{N}_G(v) = \{u, w\}$ and $(u, w) \in E$. Then (G, p) is a yes-instance for p-EWPC if and only if $(G - \{u, v, w\}, p - \text{profit}_{\text{EWPC},G}(\{u, w\}))$ is a yes-instance for p-EWPC.

(K 4) Let $G = (V, E)$ be an edge-weighted graph where the rules (K 1) to (K 3) do not apply. Let $v \in V$ with $\text{N}_G(v) = \{u, w\}$ and $(u, w) \notin E$. We define an edge-weighted graph $G^* = (V^*, E^*)$ with weight function $\text{w}^*(.)$ as follows. Let F_1 and F_2 be edge sets with

- $F_1 = \{(x, u) | (x, u) \in E \text{ and } (x, w) \in E\} \setminus \{(u, v)\}$ (i.e., $F_1 \subseteq E$) and
- F_2 consists of *new* edges, namely $F_2 = \{(x, u) | (x, u) \notin E \text{ and } (x, w) \in E\}$ (i.e., $F_2 \cap E = \emptyset$).

Let $V^* = V \setminus \{v, w\}$ and let $E^* = (E \setminus (\{(x, w) | x \in \text{N}_G(w)\} \cup \{(u, v)\})) \cup F_2$. We define $\text{w}^*(.)$ for G^* such that

- for all edges $e \in E^* \setminus (F_1 \cup F_2)$ $\text{w}^*(e) = \text{w}(e)$.
- for every $e \in F_1$, $e = (u, x)$. Then $\text{w}^*(u, x) = \text{w}(u, x) + \text{w}(w, x)$.
- for every $e \in F_2$, $e = (u, x)$. Then $\text{w}^*(u, x) = \text{w}(w, x)$.

Then (G, p) is a yes-instance for p-EWPC if and only if $(G^*, p - \text{w}(v, u) - \text{w}(v, w) + 1)$ is a yes-instance for p-EWPC.[9]

Proof. (*Sketch*) Let $V' \subseteq V$ with $\text{profit}_{\text{EWPC},G}(V') = p$. We show there exists $V'' \subseteq V^*$ with $\text{profit}_{\text{EWPC},G}(V'') \geq p - \text{w}(v, u) - \text{w}(v, w) + 1$. We distinguish the cases (1) $v \in V'$ and $u, w \notin V'$ and (2) $u, w \in V'$ and $v \notin V'$. (To see

[8] We remark that the kernelization algorithm described here is much simpler than the algorithm that creates a linear kernel for k-VC by Chen *et al* [3]. Their algorithm requires after a $O(k|V|)$ preprocessing step another $O(k^3)$ procedure applying a theorem by Nemhauser and Trotter [15].

[9] A similar reduction for vertex cover was first described in [22]. Chen *et al.* call this reduction *vertex folding* [3].

that these are the only cases, assume w.l.o.g. $v, u \in V'$ and $w \notin V'$. But then $\text{profit}_{\text{EWPC},G}((V' \setminus \{v\}) \cup \{w\}) \geq p$ and we can consider case (2) instead.) Because of Observation 1, we can assume if $v \in V'$ and $u, w \notin V'$ that $\text{N}_G(u) \subseteq V'$ and $\text{N}_G(w) \subseteq V'$. Then $\text{profit}_{\text{EWPC},G^*}(V' \setminus \{v\}) = p - \text{profit}_{\text{EWPC},G}(\{v\}) = p - \text{w}(v,u) - \text{w}(v,w) + 1$. If $v \notin V'$ and $u, w \in V'$, then $\text{profit}_{\text{EWPC},G^*}(V' \setminus \{w\}) = p - \text{w}(v,u) - \text{w}(v,w) + 1$.

Conversely, assume there exists $V' \subseteq V^*$ with $\text{profit}_{\text{EWPC},G^*}(V') \geq p^*$. Because of Observation 1, we can assume that every edge in G^* is covered by V'. Therefore either $u \in V'$ or $\text{N}_{G^*}(u) \subseteq V'$. If $u \in V'$ then $\text{profit}_{\text{EWPC},G}(V' \cup \{w\}) \geq p^* + \text{w}(v,u) + \text{w}(v,w) - 1$. If $u \notin V'$ then $\text{profit}_{\text{EWPC},G}(V' \cup \{v\}) \geq p^* + \text{w}(v,u) + \text{w}(v,w) - 1$.

(K 5) Let $G = (V, E)$ be an edge-weighted graph and let $x \in V$ with $\text{N}_G(x) = \{u, v, w\}$. Assume that the rules (K 1) to (K 4) do not apply. If the subgraph of G induced by the vertices x, u, v, and w is a clique, then (G, p) is a yes-instance for p-EWPC if and only if $(G - \{x, u, v, w\}, p - \text{profit}_{\text{EWPC},G}(\{u, v, w\}))$ is a yes-instance for p-EWPC.

(K 6) Let $G = (V, E)$ be an edge-weighted graph and let $x \in V$ with $\text{N}_G(x) = \{u, v, w\}$. Assume that the rules (K 1) to (K 5) do not apply to G. Assume further that the subgraph of G induced by the vertices x, u, v, and w contains exactly 5 edges, say $(u, w) \notin E$. Then (G, p) is a yes-instance for p-EWPC if and only $(G - v, p - \text{profit}_{\text{EWPC},G}(\{v\}))$ is a yes-instance for p-EWPC.

Proof. (*Sketch*) Let $V' \subseteq V$ with $\text{profit}_{\text{EWPC},G}(V') \geq p$. Because of Observation 1 we can assume that V' covers every edge in G. If $x \notin V'$ then u, v, and w are included in V'. If $x \in V'$ then either v or 2 vertices of $\{u, v, w\}$ are included in V'. Say $u, w \in V'$. But then $\text{profit}_{\text{EWPC},G}((V' \setminus \{x\}) \cup \{v\}) \geq p$. Therefore, in all cases we can assume that $v \in V'$. Thus, $\text{profit}_{\text{EWPC},G-v}(V' \setminus \{v\}) \geq p - \text{profit}_{\text{EWPC},G}(\{v\})$. On the other hand, assume there exists $V^* \subseteq V \setminus \{v\}$ with $\text{profit}_{\text{EWPC},G-v}(V^*) = p^*$ then $\text{profit}_{\text{EWPC},G}(V^* \cup \{v\}) \geq p^* + \text{profit}_{\text{EWPC},G}(\{v\})$.

Note that after each application of (K 6) to G (on a vertex x) we can apply rule (K 4) (to vertex x).

Definition 1. *We call an edge-weighted graph $G = (V, E)$* reduced *if (K 1) to (K 6) do not apply to G.*

Observation 2. *Let $G = (V, E)$ be a reduced and connected edge-weighted graph. Then $|V| \geq 5$.*

Theorem 2 shows the existence of a problem kernel for connected graphs. Note that this theorem also applies for non-reduced graphs with $|V| \geq 3$.

Theorem 2. *Let $G = (V, E)$ be a connected edge-weighted graph. If $|V| \geq 2p+1$ then (G, p) is a yes-instance for p-EWPC.*

To prove Theorem 2, we remark that each connected graph has a spanning subtree. In Lemma 2 we show that each tree consisting of at least $2p+1$ vertices

has profit p. From the Subgraph Lemma (Lemma 1) it then follows that every connected graph consisting of at least $2p+1$ vertices has profit p.

Lemma 2. *Let $T = (V_T, E_T)$ be a tree with $|V_T| \geq 2p+1$. Then (T, p) is a yes-instance for p-EWPC.*

Proof. Let $T = (V_T, E_T)$ be a rooted tree with $|V_T| \geq 2p+1$. Since $|V_T| \geq 3$, we can pick a vertex $w \in V_T$ with $\deg_T(w) \geq 2$ and w is parent of leaves only. Let $T_w = (V_w, E_w)$ be the by w induced subtree of T. Then $|V_w| = \deg_T(w)$ and $\text{profit}_{\text{EWPC},T}(\{w\}) = \deg_T(w) - 1$. Consider $T - T_w$. Then $|V_T \setminus V_w| \geq 2p + 1 - \deg_T(w)$ and $T - T_w$ has profit $p' = p - (\deg_T(w) - 1)$. To prove the claim it is enough to show that $|V_T \setminus V_w| \geq 2p'+1$. Since $p = p' + (\deg_T(w) - 1)$ it follows that $|V_T \setminus V_w| \geq 2p + 1 - \deg_T(w) = 2(p' + \deg_T(w) - 1) + 1 - \deg_T(w) = 2p' + \deg_T(w) - 1$. We know that $\deg_T(w) \geq 2$ and thus $|V_T \setminus V_w| \geq 2p' + 1$.

With Theorem 2 above we determined a lower bound for the maximum profit in connected graphs, i.e. for each graph $G = (V, E)$ there is a subset $V' \subseteq V$ with profit at least $\lfloor \frac{|V|-1}{2} \rfloor$. It follows from (K 2) that the linear-time algorithm implied by the proof above solves MAXIMUM EWPC for trees. We generalize the kernelization result to disconnected graphs using the following lemma.

Lemma 3. (Component Lemma) *Let $G = (V, E)$ be a reduced edge-weighted graph consisting of at least $\frac{p}{2}$ connected components. Then (G, p) is a yes-instance for p-EWPC.*

Theorem 3. *Let $G = (V, E)$ be a reduced edge-weighted graph. Assume further that the Component Lemma does not apply. If $|V| \geq 3p - 2$ then (G, p) is a yes-instance for p-EWPC.*

Proof. Let $G_i = (V_i, E_i)$, $i = 1, \ldots, q$ $(q < \frac{p}{2})$, be the connected components of G. Let $|V| \geq 3p - 2$. Then $p \leq \frac{1}{2}|V| - \frac{p}{2} + 1$. (G, p) is a yes-instance because G has profit $\sum_{i=1}^{q} \lfloor \frac{|V_i|-1}{2} \rfloor \geq \sum_{i=1}^{q} \frac{|V_i|-1}{2} - \frac{q}{2} = \frac{1}{2}\sum_{i=1}^{q} |V_i| - q = \frac{1}{2}|V| - q \geq \frac{1}{2}|V| - \frac{p}{2} + 1$.

Definition 2. *For an instance (G, p) for p-EWPC, we say $G = (V, E)$ is* kernelized *if (K 1) to (K 6) and the Component Lemma do not apply, and $|V| \leq 3p-3$.*

Corollary 2. *p-PC and p-EWPC are in FPT.*

4.2 A Bounded Search Tree

In the remainder of this section, let (G, p) be a kernelized instance for p-EWPC. From Observation 1 we can conclude the following branching step.

Basic branching step. Let (G, p) be an instance for p-EWPC. Then for a vertex v of G we can branch into instances (G_1, p_1) and (G_2, p_2) with $G_1 = G - v$, $p_1 = p - \text{profit}_{\text{EWPC},G}(\{v\})$, $G_2 = G - (\text{N}_G(v) \cup \{v\})$, and $p_2 = p - \text{profit}_{\text{EWPC},G}(\text{N}_G(v))$.

This branching step is the only branching step applied. To branch efficiently we branch on a highest degree vertex. After each branching step we re-kernelize before branching is repeated.

Theorem 4. *Let $G = (V, E)$ be a kernelized edge-weighted graph and let (G, p) be an instance for p-EWPC. Then the branching vector at the node labeled by (G, p) is $[a, b]$ where either $a \geq 4$, $b \geq 6$ or $a \geq 3$, $b \geq 8$.*

Proof. (*Sketch*) Let $G = (V, E)$ be a kernelized edge-weighted graph. We know that for all $x \in V$, $\deg_G(x) \geq 3$. We distinguish five cases. For every case we assume that the preceding cases do not apply.

Case 1. Let $x \in V$ with $\deg_G(x) \geq 6$. Branching on x results in the instances (G_1, p_1) and (G_2, p_2), $G_i = (V_i, E_i)$ $(i = 1, 2)$, with $G_1 = G - x$ and $G_2 = G - (\mathrm{N}_G(x) \cup \{x\})$. Then we ask if (G_1, p_1) or (G_2, p_2) is a yes-instance for p-EWPC with $p_1 = p - \mathrm{profit}_{\mathrm{EWPC},G}(\{x\})$ and $p_2 = p - \mathrm{profit}_{\mathrm{EWPC},G}(\mathrm{N}_G(x))$. We know that $\mathrm{profit}_{\mathrm{EWPC},G}(\{x\}) \geq 5$. Because each vertex in $\mathrm{N}_G(x)$ has a degree of at least 3 in G, there are at least 12 edges incident to the vertices in $\mathrm{N}_G(x)$. Then $\mathrm{profit}_{\mathrm{EWPC},G}(\mathrm{N}_G(x)) \geq 12 - 6 = 6$. Thus, the worst branching vector resulting out of this case is [5,6].

Case 2. Let $x \in V$ with $\deg_G(x) = 5$. Because each vertex in $\mathrm{N}_G(x)$ has a degree of at least 3 and the graph is reduced, there are at least 11 edges incident to $\mathrm{N}_G(x)$. We know that $\mathrm{profit}_{\mathrm{EWPC},G}(\{x\}) = 4$ and $\mathrm{profit}_{\mathrm{EWPC},G}(\mathrm{N}_G(x)) \geq 6$. Thus, the worst branching vector is [4,6].

Case 3. Let $\deg_G(x) = 4$. Because each vertex in $v \in V$ has $3 \leq \deg_G(v) \leq 4$ and G is kernelized, there are at least 10 edges incident to $\mathrm{N}_G(x)$. We distinguish three cases. *Case 3a.* There are at least 12 edges incident to the vertices in $\mathrm{N}_G(x)$. Since $\mathrm{profit}_{\mathrm{EWPC},G}(\{x\}) = 3$ and $\mathrm{profit}_{\mathrm{EWPC},G}(\mathrm{N}_G(x)) \geq 8$, the worst branching vector is [3,8]. *Case 3b.* There are exactly 10 edges incident to the vertices in $\mathrm{N}_G(x)$. Let $N(x) = \{a, b, c, d\}$. Then w.l.o.g. the edges incident to $\mathrm{N}_G(x)$ are $(a, b), (c, d)$, (a, v_a), (b, v_b), (c, v_c), and (d, v_d), with $\{v_a, v_b, v_c, v_d\} \subseteq V$ but $v_a, v_b, v_c, v_d \notin \mathrm{N}_G(x)$ (here v_a, v_b, v_c, v_d are not necessarily distinct). Branching on x results in (G_1, p_1) and (G_2, p_2) with $G_1 = G - x$ and $G_2 = G - (\mathrm{N}_G(x) \cup \{x\})$. Since $\mathrm{profit}_{\mathrm{EWPC},G}(\{x\}) = 3$ and $\mathrm{profit}_{\mathrm{EWPC},G}(\mathrm{N}_G(x)) \geq 6$, the worst branching vector is [3,6]. Since $\deg_{G_1}(a) = 2$ we can always reduce (G_1, p_1) further by applying (K 3) or (K 4) and improve the profit by at least 1. Thus, the worst case branching vector improves to [4,6]. *Case 3c.* Assume there are exactly 11 edges incident to $\mathrm{N}_G(x)$. Branching on x results in (G_1, p_1) and (G_2, p_2) with $G_1 = G - x$ and $G_2 = G - (\mathrm{N}_G(x) \cup \{x\})$. Since $\mathrm{profit}_{\mathrm{EWPC},G}(\{x\}) = 3$ and $\mathrm{profit}_{\mathrm{EWPC},G}(\mathrm{N}_G(v)) \geq 7$, the worst branching vector is [3,7]. Because there exists $v \in \mathrm{N}_{G_1}(x)$ with $\deg_{G_1}(v) = 2$ we can always reduce (G_1, p_1) further by applying (K 3) or (K 4) and improve the profit by at least 1. Thus the worst case branching vector improves to [4,7].

Case 4. Let $\deg_G(x) = 3$, $\mathrm{N}_G(x) = \{a, b, c\}$. Since G is reduced, we only have to consider two cases. *Case 4a.* Assume $(b, c), (a, c) \notin E$ but $(a, b) \in E$. Since G is kernelized and 3-regular, a and b have x as the only common neighbor. Branching on x results in (G_1, p_1) and (G_2, p_2) with $G_1 = G - x$ and $G_2 = G - (\mathrm{N}_G(x) \cup \{x\})$. Since $\mathrm{profit}_{\mathrm{EWPC},G}(\{x\}) = 2$ and $\mathrm{profit}_{\mathrm{EWPC},G}(\mathrm{N}_G(x)) \geq 5$, the worst branching vector so far is [2,5]. Because $\deg_{G_1}(a) = 2$, we can apply (K 4) to a (note that, since a and c are not adjacent, this does not affect the degree of c) and apply (K 3) or (K 4) to c. Thus the branching vector improves to [4,5]. Consider G_2.

Since G_2 has at least one degree-2 vertex we can apply (K 3) or (K 4). The final worst case branching vector is [4,6]. *Case 4b.* Assume $(a,b),(a,c),(b,c) \notin E$. We know that G is 3-regular. Branching on x results in (G_1, p_1) and (G_2, p_2) with $G_1 = G - x$ and $G_2 = G - (\mathrm{N}_G(x) \cup \{x\})$. Since $\mathrm{profit}_{\mathrm{EWPC},G}(\{x\}) = 2$ and $\mathrm{profit}_{\mathrm{EWPC},G}(\mathrm{N}_G(x)) \geq 6$, the worst branching vector is [2,6]. We know $\deg_{G_1}(a) = \deg_{G_1}(b) = \deg_{G_1}(c) = 2$. Since a and b have no neighbor in common we can apply (K 3) or (K 4) to a and b and thus the branching vector improves to [4,6].

From Theorem 4 and Table 1 on page 4 we receive the estimated search-tree size for the algorithm described above.

Corollary 3. *The bounded search tree has a worst-case size of* $O(1.150964^p)$.

The kernelization step of the described fpt-algorithm can be realized in time $O(p|V|)$. Then the bounded-search-tree algorithm with the integrated re-kernelization after each branching step leads to the running time of $O(p|V| + 1.150964^p)$ (see Section 2 for a more detailed explanation of this complexity analysis).

Corollary 4. *The fpt-algorithm above has a running time of* $O(p|V| + 1.150964^p)$.

5 Efficient implementations via combining tractable parameterizations

We conclude from the NP-completeness proof for PC that there is a 1:1 correspondence between the yes-instances (G,p) for p-PC and the yes-instances $(G, |E| - p)$ for k-VC (Theorem 1). Thus we can consider p-PC and k-VC as two different parameterizations of VC.[10] With our fpt-algorithm for p-PC we presented an alternative strategy to attack VC. In this section we demonstrate how different fpt-parameterizations of a given decision problem can be combined to an algorithm that incorporates the advantages of all these algorithms. Therefore, the combined algorithm yields a more efficient implementation than any of the algorithms if implemented independently. As an example, we sketch how we can combine the presented fpt-algorithm for p-PC with any fpt-algorithm for k-VC.

The general flavor of common fpt-algorithms for k-VC [3, 5, 16] is analogous to our fpt-algorithm for p-PC (i.e., the algorithms consist of a combination of kernelization, bounded-search-tree technique, and often re-kernelization, as described in the last paragraph in Section 2). The last step in a kernelization for an instance (G,k) for k-VC is a check of the number of vertices in the reduced graph. If the number exceeds the problem-kernel size, then (G,k) is a no-instance. At each node in the search tree at least one vertex is picked and thus k is reduced by at least 1. Therefore, the bounded search tree has branches of length at most k. The computation of a branch in the search tree is finished as soon as k vertices are picked or a vertex cover is found. On the other hand, our p-PC algorithm

[10] For another parameterization for VC consider the problem INDEPENDENT SET. For a given graph $G = (V,E)$ there exists a vertex cover of size k if and only if there exists an independent set of size $|V| - k$ [9]. INDEPENDENT SET, parameterized by the size of the independent set to be determined, is known to be W[1]-complete [4].

returns as an answer yes for an instance (G, p) if, after reducing, the graph is larger than the problem kernel. The branch length in the search tree is bounded by p; the computation of a branch can be ended as soon as enough profit is allocated and therefore its length does not exceed $\frac{p}{3}$ (Theorem 4).

We construct a combined algorithm by (1) keeping track of both parameters during every step of the algorithm and (2) checking the possible decisions for both parameters at each stage of the implementation. First, we take advantage of the problem kernels for the different parameterizations. The problem kernel for k-VC allows no-instances to be identified early whereas the problem kernel for p-PC allows yes-instances to be identified early. If, for a given positive integer p, the problem kernel for p-PC is smaller than the problem kernel for k-VC (with $k = |E| - p$) then there are more instances for which an early decision can be made (thanks to the p-PC kernel) than if we solely use the problem kernel for k-VC. Conversely, if the kernel for p-PC is larger than the problem kernel for k-VC (with $k = |E| - p$) then there are more instances for which an early decision can be made (thanks to the k-VC kernel) than if we solely use the problem kernel for p-PC. Secondly, in the search tree, whenever a decision in terms of k or p $(= |E| - k)$ can be made after a branching step, the computation of this particular branch terminates. Note that a yes-decision in terms of p might be possible even before a vertex cover is found. Conversely, since a no-decision in terms of k can be made even before a vertex cover is found, we can decide early that it is impossible to allocate $|E| - k$ profit.

To solve for example MVC an fpt-algorithm has to be repeated for a given graph until the optimum value for the fixed parameter is found. Using a combined implementation as described above will speed up the process of early decisions and thus avoid a big part of the otherwise necessary exponential search via branching. We implemented the algorithm suggested above and the results reflect the expected speed-up when compared with an implemenation of a k-VC algorithm as published in [5].

6 Conclusions

We presented a new fpt-algorithm for VERTEX COVER. We first introduced a parameterization, p-PROFIT COVER, and showed that it is solvable in time $O(p|V| + 1.150964^p)$. We then showed how our fpt-algorithm for p-PROFIT COVER can be used to speed up existing algorithms for k-VERTEX COVER. This new approach of combining fpt-algorithms for different parameterizations of a problem provides an additional tool for applying parameterized complexity theory in practice.

Besides its useful relationship to VC, the problem PC is also of theoretical and practical interest in itself. We have demonstrated that in some contexts PC is a better model than VC (Section 1.1). For example, in certain problems in human decision making VC may pose unnecessary constraints on the set of acceptable solutions. Further, in many real-world situations conflicts are a matter of degree. Here EDGE WEIGHTED PROFIT COVER provides a natural and useful adaptation of VC to model those problem situations.

We have shown that both p-PC and p-EWPC have linear problem kernels of

size $2p$ for connected graphs and size $3p-3$ for disconnected graphs. The fast fpt-algorithm we presented for p-PC also solves the more general problem p-EWPC. Despite the small constant $c = 1.150964$ in the running time of $O(p|V| + c^p)$, this fpt-algorithm is much simpler to implement than the sophisticated k-VC algorithm presented in [3].

Acknowledgements We thank Hausi A. Müller and Frank Ruskey for their feedback.

References

[1] J. Alber, J. Gramm, R. Niedermeier, "Faster exact algorithms for hard problems: A parameterized point of view," *Discr. Mathematics* (2001) 229, 3–27.

[2] R. Balasubramanian, M.R. Fellows, and V. Raman, "An improved fixed-parameter algorithm for Vertex Cover". *Inform. Proc. Letters* (1998) 65, 163–168.

[3] J. Chen, I.A. Kanj, W. Jia, "Vertex Cover: Further observations and further improvements," *J. Algorithms* (2001), 41, 280–301.

[4] R.G. Downey and M.R. Fellows, *Parameterized Complexity* (1999), Springer.

[5] R.G. Downey, M.R. Fellows, and U. Stege, "Parameterized Complexity: A Framework for Systematically Confronting Computational Intractability," *AMS-DIMACS Proc. Series* (1999) 49, 49–99.

[6] R.G. Downey, M.R. Fellows, and U. Stege, "Computational Tractability: The View From Mars," *Bulletin of the EATCS* (1999).

[7] M.R. Fellows, "On the complexity of vertex set problems," *Tech. Rep.* (1988), Computer Science Department, University of New Mexico.

[8] M.R. Fellows, "Parameterized Complexity: The Main Ideas and Connections To Practical Computing," *1st Dagstuhl Workshop on Exp. Algorithms* (2001).

[9] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (1979), Freeman.

[10] R.L. Graham, D.E. Knuth, and O. Patashnik, *Concrete Mathematics*, Addison-Wesley (1994).

[11] J. Gross and J. Yellen, *Graph theory and its applications* (1999), CRC Press.

[12] R. Hastie, "Problems for judgment and decision making," *Ann. Review of Psychology* (2001) 52, 653–683.

[13] K.I. Manktelow and D.E. Over (Eds.), *Rationality: Psychological and philosophical perspectives* (1993), London, Routledge.

[14] P.K. Moser (Ed.), *Rationality in action: Contemporary approaches* (1990), Cambridge University Press.

[15] G.L. Nemhauser and L.E. Trotter, "Vertex packing: structural properties and algorithms," *Mathematical Programming* (1975), 8, 232–248.

[16] R. Niedermeier and P. Rossmanith, "Upper Bounds for Vertex Cover Further Improved," In *Proc. of 16th STACS* (1999). LNCS 1563 , 561–570.

[17] R. Niedermeier, P. Rossmanith, "A general method to speed up fixed-parameter-tractable algorithms," *Inf. Proc. Letters* (2000) 73, 125–129.

[18] M. Oaksford and N. Chater, "Reasoning theories and bounded rationality," In K.I. Manktelow & D.E. Over (Eds.) *Rationality: Psychological and philosophical perspectives* (1993), 31–60, Routledge.

[19] J. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology* (1997), PWS Publ. Comp.

[20] H.A. Simon, "Invariants of human behavior". *Ann. Rev. Psych.* (1990) 41(1), 1–19.

[21] U. Stege, *Resolving Conflicts from Problems in Computational Biology* (2000) Ph.D. thesis, No.13364, ETH Zürich.

[22] U. Stege and M.R. Fellows, "An Improved Fixed-Parameter-Tractable Algorithm for Vertex Cover," (1999) Tech. Rep. 318, Dept. of Comp. Science, ETH Zürich.