Algorithmica manuscript No.
(will be inserted by the editor)

Maximizing a Submodular Function with Viability Constraints

Wolfgang Dvořák · Monika Henzinger · David P. Williamson

Received: date / Accepted: date

Abstract We study the problem of maximizing a monotone submodular function with viability constraints. This problem originates from computational biology, where we are given a phylogenetic tree over a set of species and a directed graph, the so-called food web, encoding viability constraints between these species. These food webs usually have constant depth. The goal is to select a subset of k species that satisfies the viability constraints and has maximal phylogenetic diversity. As this problem is known to be NP-hard, we investigate approximation algorithms. We present the first constant factor approximation algorithm if the depth is constant. Its approximation ratio is $(1 - \frac{1}{\sqrt{e}})$. This algorithm not only applies to phylogenetic trees with viability constraints but for arbitrary monotone submodular set functions with viability constraints. Second, we show that there is no $(1 - 1/e + \epsilon)$-approximation algorithm for our problem setting (even for additive functions) and that there is no approximation algorithm for a slight extension of this setting.

1 Introduction

We consider the problem of maximizing a monotone submodular set function f over subsets of a ground set X, subject to a restriction on what subsets

W. Dvořák
Universität Wien, Fakultät für Informatik, Währingerstraße 29, A-1090 Vienna, Austria
Tel.: +43 1 4277 783 21
Fax: +43 1 4277 8 783 21
E-mail: wolfgang.dvorak@univie.ac.at

M. Henzinger
Universität Wien, Fakultät für Informatik, Währingerstraße 29, A-1090 Vienna, Austria

D. P. Williamson
School of Operations Research and Information Engineering, Cornell University, Ithaca, New York, 14853, USA

are allowed. As discussed below, this problem has been well-studied with constraints on the allowed sets that are *downward-closed*; that is, if S is allowed subset, then so is any $S' \subset S$. Here we study the problem of maximizing such a function with a constraint that is not downward-closed. Specifically, we assume that there exists a directed acyclic graph D with the elements of X as nodes in the graph and only consider so-called *viable* sets of a certain size. A set S is viable if each element either has no outgoing edges in D or it has a path P to such an element with $P \subseteq S$. Such viability constraints are a natural way to model dependencies between elements, where an element can only contribute to the function if it appears together with specific other elements.

We are motivated by a problem arising in conservation biology. The problem is given as a rooted phylogenetic tree $\mathcal{T}$ with nonnegative weights on the edges, where the leaves of the tree represent species, and the weights represent genetic distance. Given a conservation limit k, we would like to select k species so as to maximize the overall phylogenetic diversity of the set, which is equivalent to maximizing the weight of the induced subtree on the k selected leaves plus the root. This problem, known as *Noah's Ark problem* [20], can be solved in polynomial time via a greedy algorithm [5, 14, 17].

When it comes the real-world instances the above formalization of Noah's Ark problem has been criticized for not considering that the survival of a species might depend on the survival of other species (see e.g. [18]). If one does not respect these dependencies a species might become extinct even if it is selected for preservation, which indeed would result in suboptimal solutions. Moulton, Semple, and Steel [12] introduced an extension of Noah's Ark problem which takes into account the dependence of various species on one another in a food web. In this food web, an arc is directed from species a towards species b if a's survival depends on species b. Moulton et al. now consider selecting viable subsets given by the food web of size k, i.e. a species is viable if also at least one of its successors in the food web is preserved. Note that in real life the *depth* of the food web, i.e., the longest path between any node in D and a node with no out-edge, is rather small (usually no larger than 30) (see e.g. [3]).

Faller et al. [6] show that the problem of maximizing phylogenetic diversity with viability constraints is NP-hard, even in simple special cases with constant depth (e.g. the food web is a directed tree of constant depth).

Since phylogenetic diversity induces a monotone, submodular function on a set of species, this problem is a special case of the problem of maximizing a submodular function with viability constraints. There exists a long line of research on approximately maximizing monotone submodular functions with constraints. This line of work was initiated by Nemhauser et al. [13] in 1978; they give a greedy $(1 - \frac{1}{e})$-approximation algorithm for maximizing a monotone submodular function subject to a cardinality constraint. Fisher et al. [8] introduced approximation algorithms for maximizing a monotone submodular function subject to matroid constraints (in which the set S must be an independent set in a single or multiple matroids). In recent work other types of constraints have been studied, as well as nonmonotone submodular functions; see the surveys by Vondrak [19] and Goundan and Schulz [9].

In our case, the viability constraints are *not downward-closed* while most of the prior work studies downward-closed constraints. One notable exception, where not downward-closed constraints are considered, are matroid base constraints [11]. The viability constraint could be extended to be downward-closed by simply defining every subset of a viable set to be allowable. However, this extension violates the exchange property of matroids and thus viability constraints also differ from matroid base constraints. Hence we consider a new type of constraint in submodular function maximization. We show how variants on the standard greedy algorithm can be used to derive approximation algorithms for maximizing a monotone, submodular function with viability constraints; thus we show that a new type of constraint can be handled in submodular function maximization.

Specifically we first present a scheme of $(1 - \frac{1}{e^{p/(p+d-1)}})/2$ - approximation algorithms for monotone submodular set functions with viability constraints, where d is the minimum of the depth of the food web and k, and p is a parameter of the algorithm. Let n be the number of species and m the number of edges in the food web then above algorithm's running time is in $\mathcal{O}\left(k \cdot (3^p n^{p+2} + n^{p+1} m)\right)$, i.e., the running time is exponential in p but is polynomial for any fixed p. For instance if we set $p = d$ we achieve a $(1 - \frac{1}{\sqrt{e}})/2$ - approximation algorithm.

We further present a variant of these algorithms which are $(1 - \frac{1}{e^{p/(p+d-1)}})$ - approximations, but whose running time is $\mathcal{O}\left(k \cdot (3^p n^{4p+3d-1} + n^{4p+3d-2} m)\right)$, i.e., exponential in both d and p. For fixed $d{=}p$ this is polynomial and provides an $(1 - \frac{1}{\sqrt{e}})$ - approximation algorithm. However, as the running time heavily depends on d and p this is only feasible for small values of d. For larger values of d one has to find the right balance between running-time and approximation guarantee.

Next by a reduction from the maximum coverage problem, we show that there is no $(1 - 1/e + \epsilon)$-approximation algorithm for the phylogenetic diversity problem with viability constraints (unless $\mathsf{P} = \mathsf{NP}$). However, the reduction from maximum coverage introduces a food web of linear depth. Thus, we further give a reduction from Max Vertex Cover that shows APX-hardness even for instances with constant depth food webs. Finally we consider a generalization of our problem where we additionally allow AND-constraints such as "species a is viable only if we preserve *both* species b and species c" and show that this generalization has no approximation algorithm (assuming $\mathsf{P} \neq \mathsf{NP}$) by a reduction from 3-SAT.

The remainder of the paper is structured as follows. We define the problem more precisely in Section 2, introduce our algorithms in Section 3, and give the hardness results in Section 4.

2 Phylogenetic Diversity with Viability Constraints

We first give a formal definition of the problem.

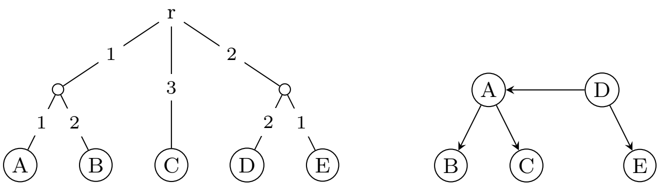
Definition 1 A (rooted) *phylogenetic tree* $\mathcal{T} = (T, E_{\mathcal{T}})$ is a rooted tree with root r and each non-leaf node having at least 2 child-nodes together with a weight function w assigning non-negative integer weights to the edges. Let $X_{\mathcal{T}}$ denote the set of leaf nodes of $\mathcal{T}$ also called species. For any set $A \subseteq X_{\mathcal{T}}$ the operator $\mathcal{T}(A)$ yields the spanning tree of the set $A \cup \{r\}$, and by $\mathcal{T}_E(A)$ we denote the edges of this spanning tree. Then for any set $S \subseteq X_{\mathcal{T}}$ the *phylogenetic diversity* is defined as

$$\mathcal{PD}(S) = \sum_{e : e \in \mathcal{T}_E(S)} w(e)$$

A *food web* D for the phylogenetic tree $\mathcal{T} = (T, E_{\mathcal{T}})$ is an acyclic directed graph $(X_{\mathcal{T}}, E)$. A set $S \subseteq X_{\mathcal{T}}$ is called viable if each $s \in S$ is either a sink (a node with out-degree 0) in D or there is a $s' \in S$ such that $(s, s') \in E$.

Below we exemplify these concepts.

Example 1 Consider the phylogenetic tree given in Figure 1. The root r is at the top and the set of species $X = \{A, B, C, D, E\}$ at the bottom. We omitted names for the two inner nodes. Now we have that $\mathcal{PD}(\{A\}) = 2$, $\mathcal{PD}(\{B\}) = 3$ and $\mathcal{PD}(\{A, B\}) = 4$. Finally considering the concept of viable sets. Considering the given food web we have that $\{A, B\}$, $\{A, B, D\}$ are viable as B is a sink and for A, D one of the successors is included in the set. On the other hand side the sets $\{A\}$, $\{A, D, E\}$ are not viable as none of the successors of A is included.



(a) Phylogenetic Tree

(b) Food Web

Fig. 1 An illustration of Example 1.

Now our problem of interest is defined as follows.

Definition 2 (OptPDVC) The *Optimizing Phylogenetic Diversity with Viability Constraints* (OptPDVC) problem is defined as follows. You are given a phylogenetic tree $\mathcal{T}$ and a food web $D = (X_{\mathcal{T}}, E)$, and a positive integer k. Find a viable subset $S \subseteq X_{\mathcal{T}}$ of size (at most) k maximizing $\mathcal{PD}(S)$.

OptPDVC is known to be NP-hard [6], even for restricted classes of phylogenetic trees and dependency graphs.

First we study fundamental properties of the function $\mathcal{PD}$.

Definition 3 The set function $\mathcal{PD}(.|.) : 2^X \times 2^X \mapsto \mathbb{N}_0$ for each $A, B \subseteq X$ is defined as $\mathcal{PD}(A|B) = \mathcal{PD}(A \cup B) - \mathcal{PD}(B)$.

The intuitive meaning of $\mathcal{PD}(A|B)$ is the gain of diversity we get by adding the set A to the already selected species B.

We next recall the definition of submodular set functions. We call a set function *submodular* if

$$\forall A, B, C \subseteq \Omega : A \subseteq B \Rightarrow f(A \cup C) - f(A) \geq f(B \cup C) - f(B).$$

It turns out that $\mathcal{PD}$ as defined above happens to be a submodular function.

Proposition 1 *$\mathcal{PD}$ is a non-negative monotone submodular function [2].*

Now consider the function $\mathcal{PD}(.|.)$. As $\mathcal{PD}(.)$ is monotone also $\mathcal{PD}(.|.)$ is monotone in the first argument and because of the submodularity of $\mathcal{PD}(.)$ the function $\mathcal{PD}(.|.)$ is anti-monotone in the second argument.

In the remainder of the paper we will not refer to the actual definition of the functions $\mathcal{PD}(.)$, $\mathcal{PD}(.|.)$, but only exploit monotonicity, submodularity and the fact that these functions can be efficiently computed.

Moreover we will consider a function VE (*viable extension*) which, given a set of species S, returns a viable set S' of minimum size containing S. In the simplest case where S consist of just one species it computes a shortest path to any sink node in the food web. Given a food web D and the set $\mathcal{P}_D$ of paths that end in leaf node of D, we define the *truncated depth* [1] d of D as

$$d(D) = \min(\max_{P \in \mathcal{P}_D} |P|, k)$$

i.e., as the minimum of the cardinality of the longest path that ends in a node without outgoing edges and k. If the food web is clear from the context we just write d instead of $d(D)$. Note that the problem remains NP-hard for instances with $d = 2$, even if $\mathcal{PD}$ is additive [6]. Finally we define the *costs* $c(A|B)$ of adding a set of species A to a set B

$$c(A|B) = |\,\mathrm{VE}(B \cup A)| - |B|.$$

Notice that although B typically will be a viable set this is not required in the definition.

[1] Notice that we use a slightly different definition for d than in [4].

Algorithm 1 Greedy, Faller et al.

```
$S \leftarrow \emptyset$
while $|S| < k$ do
    $s \leftarrow \underset{c(s|S)=1}{\operatorname{argmax}} v(\{s\}|S)$
    $S \leftarrow S \cup \{s\}$
end while
```

3 Approximation Algorithms

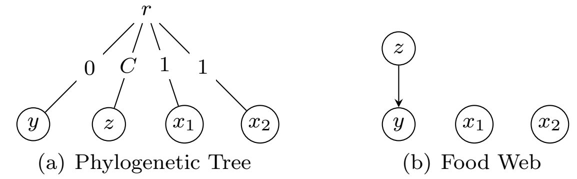
In this section we assume that a non-negative, monotone submodular function $\mathcal{PD}(.)$ is given as an oracle and we want to maximize $\mathcal{PD}(.)$ under viability constraints together with a cardinality constraint. We first review the greedy algorithm given by Faller et al. [6] presented in Algorithm 1. The idea is that, in each step, one considers only species which either have no successors in the food web or for which one of the successors has already been selected (adding one of these species will keep the set viable). Then one adds the species that gives the largest gain of phylogenetic diversity. By the restriction on the considered species the constructed set is always viable, but we might miss highly valuable species which is illustrated by the following example.

Example 2 Consider the set of species $X = \{y, z, x_1, x_2\}$ the phylogenetic tree $\mathcal{T} = (\{r\} \cup X, \{(r, y), (r, z), (r, x_1), (r, x_2)\}$, weights $w(r, x_i) = 1$, $w(r, y) = 0$, $w(r, z) = C$ with $C > 2$ and the food web $(X, \{(z, y)\})$ (see Fig. 2). Assume a budget $k = 2$. As the species y has weight 0, Algorithm 1 would pick x_1, and x_2. Hence Algorithm 1 results in a viable set with diversity 2. But the set $\{z, y\}$ is viable and has diversity C, which can be made arbitrarily large.

This example shows that the solution computed by the greedy algorithm can perform arbitrarily bad, because it ignores highly weighted species if they are "on the top of" less valuable species. Hence one can not give any approximation guarantee for Algorithm 1.

3.1 A Greedy Approximation Algorithm

By the above observations, to get an approximation guarantee, we have to consider all subsets of species up to a certain size p which can be made viable



(a) Phylogenetic Tree (b) Food Web

Fig. 2 An illustration of Example 2.

Algorithm 2

```
1: Select a set $S$ with $|S| \leq p$, $c(S|\emptyset) \leq k$ and maximal $\mathcal{PD}(S)$
2: $\mathcal{G} = \mathrm{VE}(S)$
3: $G \leftarrow \emptyset$
4: while $|G| < k$ do
5:     Select a set $S$ with $|S| \leq p$, $c(S|G) \leq k - |G|$ and maximal ratio $\frac{\mathcal{PD}(S|G)}{c(S|G)}$
6:     $G = \mathrm{VE}(G \cup S)$
7: end while
8: if $\mathcal{PD}(G) > \mathcal{PD}(\mathcal{G})$ then
9:     $\mathcal{G} \leftarrow G$
10: end if
```

and pick the most valuable subset. Algorithm 2 exploits this observation. It generalizes concepts from Bordewich and Semple [1], which itself builds on Khuller et al. [10]. In the algorithm G denotes the current set of selected species; and $\mathcal{G}$ denotes the best viable set we have found so far. Lines 3 - 7 of the algorithm implements a greedy algorithm that in each step selects among the sets of size $\leq p$ with costs that are within the remaining budget the most "cost efficient" subset of species, i.e. the subset S of species that maximizes the ratio of the increase in PD over the cost of adding S, and adds it to the solution. But Algorithm 2 does not solely run the greedy algorithm, it first computes the set with maximal $\mathcal{PD}$ among all sets of size $\leq p$ that can be made viable. In certain cases this set is better than the viable set obtained by the greedy algorithm, a fact that we exploit in the proof of Theorem 1.

The next theorem will analyze the approximation ratio of Algorithm 2.

Theorem 1 *For all integers $p \geq 1$ Algorithm 2 gives a $(1 - \frac{1}{e^{p/(p+d-1)}})/2$ approximation.*

As for $p \geq k$ the algorithm enumerates all possible solutions and thus is optimal in the following analysis we will assume that $p \leq k$. To prove Theorem 1, we introduce some notation. First let $O \subseteq X$ denote the optimal solution. We will consider a decomposition $\mathcal{D}_O$ of O into sets $O_1, \ldots, O_{\lceil k/p \rceil}$ of size $\leq p$. By decomposition we mean that (i) $\bigcup_{i=1}^{\lceil k/p \rceil} O_i = O$ and (ii) $O_i \cap O_j = \emptyset$ if $i \neq j$. Moreover we require that $|\mathrm{VE}(O_i)| \leq p+d-1$ and $\sum_i |\mathrm{VE}(O_i)| \leq \frac{k}{p}(p+d-1)$. Next we show that such a decomposition $\mathcal{D}_O$ always exists.

Lemma 1 *There exist $\lceil \frac{k}{p} \rceil$ many pairs $(O_1, B_1), \ldots, (O_{\lceil \frac{k}{p} \rceil}, B_{\lceil \frac{k}{p} \rceil})$ such that $O = \bigcup_{1 \leq i \leq \lceil \frac{k}{p} \rceil} O_i$, $O_i \cup B_i$ is viable, $|O_i| \leq p$, $|B_i| \leq d-1$ and $\sum_i |O_i \cup B_i| \leq \frac{k}{p}(p+d-1)$.*

Proof The optimal solution O is a viable subset of size at most k. Consider the reverse graph G of D projected on the set O i.e. $G = (O, E^{-} \cap (O \times O))$, and add an artificial root τ that has an edge to all roots of G (the sinks of D). By the definition of $d(D)$ we obtain that in G the longest path starting in τ is at most of size $d+1$.

Start a depth-first-search in τ and initialize O_1, B_1 with empty sets. Whenever the DFS removes a node from the stack, we add this node to the current set O_i, $i \geq 1$. When $|O_i| = p$ then we add the nodes on the stack, except τ, to the set B_i, but do not change the stack itself. Then we continue the DFS with the next pair (O_{i+1}, B_{i+1}), again initialized by empty sets. Eventually the DFS stops, then the stack is empty and thus $(O_{\lceil \frac{k}{p} \rceil}, \emptyset)$ is the last pair. Notice since the longest path starting in τ is at most of size $d+1$, there are at most d nodes on the stack, one being the root τ and hence $|B_i| \leq d-1$. Since the DFS removes each node exactly once from the stack, all the sets $O_i \subseteq O$ are disjoint and all except the last one are of size p. Hence the DFS produces $\lceil \frac{k}{p} \rceil$ many sets O_i satisfying $O_i \cup B_i$ is viable, $|O_i| \leq p$ and $|B_i| \leq d-1$. Finally as, by construction, $B_{\lceil \frac{k}{p} \rceil} = \emptyset$ and $|O_{\lceil \frac{k}{p} \rceil}| = k \mod p$ we obtain that $\sum_{i=1}^{\lceil \frac{k}{p} \rceil} |O_i \cup B_i| = \sum_{i=1}^{\lfloor \frac{k}{p} \rfloor} |O_i \cup B_i| + (k \mod p) \leq \lfloor \frac{k}{p} \rfloor (p+d-1) + (k \mod p) \leq \frac{k}{p}(p+d-1)$. □

First, we consider the greedy algorithm and the value l where the l-th iteration is the first iteration such that after executing the loop body, $\max_{O_j \in \mathcal{D}_O} \frac{\mathcal{PD}(O_j|G)}{c(O_j|G)} > \max_{|S| \leq p, c(S|G) \leq k-|G|} \frac{\mathcal{PD}(S|G)}{c(S|G)}$. If the greedy solution is different from the optimal one the inequality holds at least for the last iteration of the loop where $S = \emptyset$. We define $O'_{l+1} = \underset{O_j \in \mathcal{D}_O}{\operatorname{argmax}} \frac{\mathcal{PD}(O_j|G)}{c(O_j|G)}$, i.e. O'_{l+1} is in the optimal viable set and would be a better choice than the selection of the algorithm, but the greedy algorithm cannot make $G \cup O'_{l+1}$ viable without violating the cardinality constraint.

Let S_i denote the set S added to G in iteration i of the while loop in Line 4. Moreover, for $i \leq l$ we denote the set G after the i-th iteration by G_i, with $G_0 = \emptyset$, the set $G \cup S$ from Line 6 as $G_i^* = G_{i-1} \cup S_i$ and the "costs" of adding set S_i by $c_i = c(S_i|G_{i-1}) = c(G_i|G_{i-1})$. With a slight abuse of notation we will use G_{l+1} to denote the viable set $\mathrm{VE}(G_l \cup O'_{l+1})$, c_{l+1} to denote $c(O'_{l+1}|G_l)$ and G^*_{l+1} to denote the set $G_l \cup O'_{l+1}$ (G_{l+1} is not a feasible solution as $|\mathrm{VE}(G_{l+1})| > k$). Notice that while the sets G_i^* are not necessarily viable sets, all the $G_i, i \geq 0$ are viable sets and moreover $\mathcal{PD}(\mathcal{G}) \geq \mathcal{PD}(G_i), i \leq l$.

First we show that in each iteration of the algorithm the set S_i gives a certain approximation of the missing part of the optimal solution.

Lemma 2 *For all integers $1 \leq p \leq k$ and $1 \leq i \leq l+1$:*

$$\frac{\mathcal{PD}(G_i^*|G_{i-1})}{c_i} \geq \frac{p}{(p+d-1)k} \cdot \mathcal{PD}(O|G_{i-1})$$

Proof By definition of S_i and O'_{l+1} for each $O_j \in \mathcal{D}_O$ the following holds:

$$\frac{\mathcal{PD}(O_j|G_{i-1})}{c(O_j|G_{i-1})} \leq \frac{\mathcal{PD}(S_i|G_{i-1})}{c(S_i|G_{i-1})} \text{ for } i \leq l \qquad \frac{\mathcal{PD}(O_j|G_l)}{c(O_j|G_l)} \leq \frac{\mathcal{PD}(O'_{l+1}|G_l)}{c(O'_{l+1}|G_l)}$$

Combining the two inequalities we have

$$\frac{\mathcal{PD}(O_j|G_{i-1})}{c(O_j|G_{i-1})} \leq \frac{\mathcal{PD}(G_i^*|G_{i-1})}{c(G_i|G_{i-1})} \quad \text{for } i \leq l+1$$

Next we use the monotonicity and submodularity of $\mathcal{PD}$ (for the first inequality) and the inequality from above (for the second inequality).

$$\begin{aligned}\mathcal{PD}(O|G_{i-1}) &\leq \sum_{O_j \in \mathcal{D}_O} \mathcal{PD}(O_j|G_{i-1}) = \sum_{O_j \in \mathcal{D}_O} \frac{\mathcal{PD}(O_j|G_{i-1})}{c(O_j|G_{i-1})}\, c(O_j|G_{i-1}) \\ &\leq \sum_{O_j \in \mathcal{D}_O} \frac{\mathcal{PD}(G_i^*|G_{i-1})}{c_i}\, c(O_j|G_{i-1}) \leq \frac{\mathcal{PD}(G_i^*|G_{i-1})}{c_i}\, \frac{p+d-1}{p} \cdot k\end{aligned}$$

The last step exploits that by Lemma 1 $\sum_{O_j \in \mathcal{D}_O} c(O_j|G_{i-1}) \leq \frac{k}{p} \cdot (p+d-1)$. □

Lemma 3 *For* $1 \leq i \leq l+1$*:*

$$\mathcal{PD}(G_i^*) \geq \left[1 - \prod_{j=1}^{i} \left(1 - \frac{p \cdot c_j}{(d+p-1)\cdot k}\right)\right] \mathcal{PD}(O)$$

Proof The proof is by induction on i. The base case for $i = 1$ is by Lemma 2. For the induction step we show that if the claim holds for all $i' < i$ then it must also hold for i. For convenience we define $C_i = \frac{p \cdot c_i}{(d+p-1)\cdot k}$.

$$\begin{aligned}\mathcal{PD}(G_i^*) &= \mathcal{PD}(G_{i-1}) + \mathcal{PD}(G_i^*|G_{i-1}) \geq \mathcal{PD}(G_{i-1}) + C_i \cdot \mathcal{PD}(O|G_{i-1}) \\ &= \mathcal{PD}(G_{i-1}) + C_i \cdot (\mathcal{PD}(O \cup G_{i-1}) - \mathcal{PD}(G_{i-1})) \\ &\geq (1 - C_i) \cdot \mathcal{PD}(G_{i-1}) + C_i \cdot \mathcal{PD}(O) \\ &\geq (1 - C_i) \cdot \mathcal{PD}(G_{i-1}^*) + C_i \cdot \mathcal{PD}(O) \\ &\geq (1 - C_i) \left[1 - \prod_{j=1}^{i-1} (1 - C_j)\right] \mathcal{PD}(O) + C_i \cdot \mathcal{PD}(O) \\ &= \left[1 - \prod_{j=1}^{i} (1 - C_j)\right] \mathcal{PD}(O)\end{aligned}$$

□

Proof (Theorem 1) We first give a bound for G_{l+1}^*. To this end consider $\sum_{m=1}^{l+1} c_m$. As G_{l+1} exceeds the cardinality constraint $\sum_{m=1}^{l+1} c_m > k$ it follows that:

$$\begin{aligned}1 - \prod_{j=1}^{l+1} \left(1 - \frac{p \cdot c_j}{(d+p-1)\cdot (k)}\right) &\geq 1 - \prod_{j=1}^{l+1} \left(1 - \frac{p \cdot c_j}{(d+p-1)\cdot \sum_{m=1}^{l+1} c_m}\right) \\ \geq 1 - \left(1 - \frac{p}{(d+p-1)\cdot(l+1)}\right)^{l+1} &\geq 1 - \frac{1}{e^{p/(d+p-1)}}\end{aligned}$$

To obtain second inequality we used the fact that the term $\prod_{j=1}^{l+1} \left(1 - \frac{c_j'}{C}\right)$ with constant C and the constraint $\sum_{j=1}^{l+1} c_j' = 1$ has its maximum at $c_j' = 1/(l+1)$.

By Lemma 3 we obtain $\mathcal{PD}(G_{l+1}^*) \geq \left(1 - \frac{1}{e^{p/(d+p-1)}}\right) \cdot \mathcal{PD}(O)$, thus it only remains to relate $\mathcal{PD}(G_{l+1}^*)$ to $\mathcal{PD}(G_l)$. To this end we consider the optimal set of size p selected in Line 2 and denote it by S_o. If the greedy solution has higher $\mathcal{PD}$ than S_o the algorithm returns a superset of G_l^* otherwise a superset of S_o. Hence, $\mathcal{PD}(G)$ is larger or equal to the maximum of $\mathcal{PD}(G_l^*)$ and $\mathcal{PD}(S_o)$. From the definitions of G_l^* and S_o it follows that

$$\mathcal{PD}(G_{l+1}^*) \leq \mathcal{PD}(G_l^*) + \mathcal{PD}(O'_{l+1}) \leq \mathcal{PD}(G_l^*) + \mathcal{PD}(S_o).$$

Now we have that either $\mathcal{PD}(G_l^*) \geq \mathcal{PD}(G_{l+1}^*)/2$ or $\mathcal{PD}(S_o) \geq \mathcal{PD}(G_{l+1}^*)/2$ and thus obtain the following.

$$\mathcal{PD}(G) \geq \max(\mathcal{PD}(G_l^*), \mathcal{PD}(S_o)) \geq \left(1 - \frac{1}{e^{p/(d+p-1)}}\right) \cdot \frac{\mathcal{PD}(O)}{2}$$

Hence Algorithm 2 provides an $\left(1 - \frac{1}{e^{p/(d+p-1)}}\right)/2$ - approximation. □

Theorem 2 *Algorithm 2 runs in time* $\mathcal{O}\left(k \cdot (3^p n^{p+2} + n^{p+1} m)\right)$ *where* n *is the number of species and* m *the number of edges in the food web.*

Proof First notice that computing the function VE can be reduced to a Steiner tree problem as follows. To compute VE(S) first modify the graph D by (i) taking all the species in S that are already connected (via nodes in S) to a sink node in S, and merging these nodes into a single terminal node t, and (ii) connecting the remaining sink nodes in D to t. For the starting nodes in the Steiner tree problem we use the species in S that are not viable. In Algorithm 2 when computing VE($G \cup S$), there are at most p starting nodes as the set G is made viable after each iteration and thus the viable set G is contracted to the single vertex t. The Steiner tree problem on acyclic directed graphs can be solved in time $\mathcal{O}(3^j n^2 + nm)$ [16], where j is the number of starting and terminal nodes. In Line 1 we have to consider $\mathcal{O}(n^p)$ sets S and for each of them we solve a Steiner tree problem. with at most p starting nodes. So this first loop can be done in time $\mathcal{O}(3^p n^{p+2} + n^{p+1} m)$. The number of iterations of the while loop is bounded by k and in each iteration, in Line 4, we have to solve $\mathcal{O}(n^p)$ Steiner tree problems with at most p starting nodes. Now as each iteration takes time $\mathcal{O}(3^p n^{p+2} + n^{p+1} m)$ we get a total running time of $\mathcal{O}(k \cdot (3^p n^{p+2} + n^{p+1} m))$. □

3.2 An Approximation Algorithm using the Enumeration Technique

In this Section we use a modification of the enumeration technique as described in [10], to get rid of the factor 1/2 in the approximation ratio. The idea is to consider all (viable) sets of a certain size and for each of them to run the greedy algorithm starting with this set. Finally one chooses the best of the produced solutions. These sets typically have to contain three objects of interest, in the case of the maximum coverage problem [10] (cf. Def. 4 below) just three sets from the collection $\mathcal{S}$ and thus the running time there is only increased by a cubic factor. However, in our setting such an object of interest is a pair (O_i, B_i), i.e. a set O_i of size $\leq p$ and a set B_i of size $< d$ making O_i viable. Thus these three objects result in a set of size of $3p + 3d - 3$, increasing the

Algorithm 3

```
1: $\mathcal{G} \leftarrow \emptyset$
2: for each $G \subseteq X$, $G$ viable, $|G| \leq \min(3p+3d-3, k)$ do
3:     while $|G| < k$ do
4:         Select a set $S$ with $|S| \leq p$, $c(S|G) \leq k - |G|$ and maximal ratio $\frac{\mathcal{PD}(S|G)}{c(S|G)}$
5:         $G = \mathrm{VE}(G \cup S)$
6:     end while
7:     if $\mathcal{PD}(G) > \mathcal{PD}(\mathcal{G})$ then
8:         $\mathcal{G} \leftarrow G$
9:     end if
10: end for
```

running time by a factor of $n^{3p+3d-3}$. Algorithm 3 gives a precise formulation of the modified algorithm.

Theorem 3 *Algorithm 3 is an* $\left(1 - \frac{1}{e^{p/(d+p-1)}}\right)$ *- approximation algorithm for OptPDVC which runs in time* $\mathcal{O}\left(k \cdot (3^p n^{4p+3d-1} + n^{4p+3d-2} m)\right)$, *where* n *is the number of species,* m *the number of edges in the food web and* $p \geq 1$ *the parameter used in Algorithm 3.*

The proof of the above theorem is similar to the above analysis for Algorithm 2. Again let O be the optimal viable set and $\mathcal{D}_O$ a decomposition of O, given by Lemma 1. We consider the set $G_0^* = O_1^* \cup O_2^* \cup O_3^*$ with $\{O_1^*, O_2^*, O_3^*\} \subseteq \mathcal{D}_O$ such that $\{O_1^*, O_2^*\} = \underset{\{O_i, O_j\} \subseteq \mathcal{D}_O}{\operatorname{argmax}} \mathcal{PD}(O_i \cup O_j)$ and $O_3^* = \underset{O_i \in \mathcal{D}_O}{\operatorname{argmax}} \mathcal{PD}(O_1^* \cup O_2^* \cup O_i)$ and the viable extension $G_0 = G_0^* \cup B_1^* \cup B_2^* \cup B_3^*$. At some point Algorithm 3 will consider G_0. We consider this iteration of the for loop in Line 2 and consider the greedy algorithm. To this end we use the same notation as in the proof of Theorem 1, the only difference being the definition of the set G_0 above. Recall that the value l is defined such that the l-th iteration is the first iteration such that after executing the loop body, $\max_{O_j \in \mathcal{D}_O} \frac{\mathcal{PD}(O_j|G)}{c(O_j|G)} > \max_{|S| \leq p, c(S|G) \leq k - |G|} \frac{\mathcal{PD}(B|G)}{c(B|G)}$.

If $|G_0| = k$ then Line 2 of the algorithm enumerates all possible solutions and will eventually find the optimal solution. So in this case there is no need to study the greedy algorithm. Hence in the remainder of this section we will assume that $|G_0| < k$.

Lemma 4 *For* $1 \leq i \leq l+1$, $p \in \{1, \ldots, k\}$*:*

$$\frac{\mathcal{PD}(G_i^*|G_{i-1})}{c_i} \geq \frac{p}{(p+d-1)(k-|G_0|)} \cdot \mathcal{PD}(O|G_{i-1})$$

Proof By definition of S_i and O'_{l+1} for each $O_j \in \mathcal{D}_O$ the following holds:

$$\frac{\mathcal{PD}(O_j|G_{i-1})}{c(O_j|G_{i-1})} \leq \frac{\mathcal{PD}(S_i|G_{i-1})}{c(S_i|G_{i-1})} \text{ for } i \leq l \qquad \frac{\mathcal{PD}(O_j|G_l)}{c(O_j|G_l)} \leq \frac{\mathcal{PD}(O'_{l+1}|G_l)}{c(O'_{l+1}|G_l)}$$

Combining the two inequalities we have

$$\frac{\mathcal{PD}(O_j|G_{i-1})}{c(O_j|G_{i-1})} \leq \frac{\mathcal{PD}(G_i^*|G_{i-1})}{c(G_i|G_{i-1})} \quad \text{for } i \leq l+1$$

Next we use the monotonicity and submodularity of $\mathcal{PD}$ (for the first inequality), the inequality from above (for the second inequality). We use $\mathcal{D}'_O$ to denote $\mathcal{D}_O \setminus \{O_j \subseteq G_{i-1}\}$.

$$\begin{aligned}
\mathcal{PD}(O|G_{i-1}) &\leq \sum_{O_j \in \mathcal{D}'_O} \mathcal{PD}(O_j|G_{i-1}) = \sum_{O_j \in \mathcal{D}'_O} \frac{\mathcal{PD}(O_j|G_{i-1})}{c(O_j|G_{i-1})}\, c(O_j|G_{i-1}) \\
&\leq \sum_{O_j \in \mathcal{D}'_O} \frac{\mathcal{PD}(G_i^*|G_{i-1})}{c_i}\, c(O_j|G_{i-1}) = \frac{\mathcal{PD}(G_i^*|G_{i-1})}{c_i} \sum_{O_j \in \mathcal{D}'_O} c(O_j|G_{i-1}) \\
&\leq \frac{\mathcal{PD}(G_i^*|G_{i-1})}{c_i}\, \frac{p+d-1}{p} \cdot (k - |G_0|)
\end{aligned}$$

For the last step consider the modified food web graph where (i) all nodes in G_0 are deleted and (ii) for each node that had an edge to a node in G_0 all outgoing edges are deleted, i.e. the node is considered as a species that does not depend on any other species. Then we can apply Lemma 1 to this graph and the set $O' = O \setminus G_0$ to obtain a decomposition satisfying $\sum_{O_j \in \mathcal{D}'_O} c(O_j|G_{i-1}) \leq \frac{k-|G_0|}{p} \cdot (p+d-1)$. □

Lemma 5 *For* $1 \leq i \leq l+1$*:*

$$\mathcal{PD}(G_i^*|G_0) \geq \left[1 - \prod_{j=1}^{i} \left(1 - \frac{p \cdot c_j}{(d+p-1)\cdot(k-|G_0|)}\right)\right] \mathcal{PD}(O|G_0)$$

Proof The proof is by induction on i. The base case for $i=1$ is by Lemma 4. For the induction step we show that if the claim holds for all $i' < i$ then it must also hold for i. For convenience we define $C_i = \frac{p \cdot c_i}{(d+p-1)\cdot(k-|G_0|)}$.

$$\begin{aligned}
\mathcal{PD}(G_i^*|G_0) &= \mathcal{PD}(G_{i-1}|G_0) + \mathcal{PD}(G_i^*|G_{i-1}) \\
&\geq \mathcal{PD}(G_{i-1}|G_0) + C_i \cdot \mathcal{PD}(O|G_{i-1}) \\
&= \mathcal{PD}(G_{i-1}|G_0) + C_i \cdot (\mathcal{PD}(O \cup G_{i-1}) - \mathcal{PD}(G_{i-1})) \\
&= \mathcal{PD}(G_{i-1}|G_0) + C_i \cdot (\mathcal{PD}(O \cup G_{i-1}) - \mathcal{PD}(G_0) - (\mathcal{PD}(G_{i-1}) - \mathcal{PD}(G_0))) \\
&= \mathcal{PD}(G_{i-1}|G_0) + C_i \cdot (\mathcal{PD}(O \cup G_{i-1}|G_0) - \mathcal{PD}(G_{i-1}|G_0)) \\
&\geq (1 - C_i) \cdot \mathcal{PD}(G_{i-1}|G_0) + C_i \cdot \mathcal{PD}(O|G_0) \\
&\geq (1 - C_i) \cdot \mathcal{PD}(G_{i-1}^*|G_0) + C_i \cdot \mathcal{PD}(O|G_0) \\
&\geq (1 - C_i) \left[1 - \prod_{j=1}^{i-1} (1 - C_j)\right] \mathcal{PD}(O|G_0) + C_i \cdot \mathcal{PD}(O|G_0) \\
&= \left[1 - \prod_{j=1}^{i} (1 - C_j)\right] \mathcal{PD}(O|G_0)
\end{aligned}$$

□

We are now prepared to prove the claimed approximation ratio.

Proof (Theorem 3 approximation ratio) We first give a bound for G^*_{l+1}. To this end consider $\sum_{m=1}^{l+1} c_m$. As G^*_{l+1} exceeds the cardinality constraint $\sum_{m=1}^{l+1} c_m > k - |G_0|$ and hence:

$$\begin{aligned}
&1-\prod_{j=1}^{l+1}\left(1-\frac{p\cdot c_j}{(d+p-1)\cdot(k-|G_0|)}\right) \geq 1-\prod_{j=1}^{l+1}\left(1-\frac{p\cdot c_j}{(d+p-1)\cdot\sum_{m=1}^{l+1} c_m}\right)\\
&\geq 1-\left(1-\frac{p}{(d+p-1)\cdot(l+1)}\right)^{l+1} \geq 1-\frac{1}{e^{p/(d+p-1)}}
\end{aligned}$$

To obtain the second inequality we used that the term $\prod_{j=1}^{l+1}\left(1-\frac{c'_j}{C}\right)$ with constant C and the constraint $\sum_{j=1}^{l+1} c'_j = 1$ has its maximum at $c'_j = 1/(l+1)$.

By Lemma 5 we obtain $\mathcal{PD}(G^*_{l+1}|G_0) \geq \left(1-\frac{1}{e^{p/(d+p-1)}}\right)\ \mathcal{PD}(O|G_0)$ and thus it only remains to relate G^*_{l+1} to G_l. First as $G_0 = O^*_1 \cup O^*_2 \cup O^*_3$ and by the definition of O^*_1, O^*_2 we get $\mathcal{PD}(O^*_3|O^*_1 \cup O^*_2) \leq \mathcal{PD}(G_0)/3$. Next consider

$$\begin{aligned}
\mathcal{PD}(G^*_{l+1}) - \mathcal{PD}(G_l) &= \mathcal{PD}(O'_{l+1}|G_l) \leq \mathcal{PD}(O'_{l+1}|O^*_1 \cup O^*_2)\\
&\leq \mathcal{PD}(O^*_3|O^*_1 \cup O^*_2) \leq \mathcal{PD}(G_0)/3
\end{aligned}$$

Finally, we can combine our results to obtain the claim:

$$\begin{aligned}
\mathcal{PD}(G) &\geq \mathcal{PD}(G_l) \geq \mathcal{PD}(G^*_{l+1}) - \mathcal{PD}(G_0)/3\\
&= \mathcal{PD}(G^*_{l+1}|G_0) + \mathcal{PD}(G_0) - \mathcal{PD}(G_0)/3\\
&\geq \left(1-\frac{1}{e^{p/(p+d-1)}}\right)\ (\mathcal{PD}(O) - \mathcal{PD}(G_0)) + 2/3\cdot\mathcal{PD}(G_0)\\
&\geq \left(1-\frac{1}{e^{p/(p+d-1)}}\right)\cdot\mathcal{PD}(O)
\end{aligned}$$

The last inequality is by the fact that $1-\frac{1}{e^{p/(p+d-1)}} \leq 2/3$ for all $p, d \geq 1$. □

To conclude the proof of Theorem 3 we finally consider the running time of Algorithm 3.

Proof (Theorem 3 - running time) As discussed in the proof of Theorem 2 computing the function VE is basically a Steiner tree problem and can be solved in time $\tilde{\mathcal{O}}(3^j n^2 + nm)$, where j is the number of starting nodes. In the algorithm we have to consider $\mathcal{O}(n^{3p+3d-3})$ sets S and for each of them we start the greedy algorithm. The number of iterations of the while loop is bounded by k and in each iteration, in Line 4, we have to solve $\mathcal{O}(n^p)$ Steiner tree problems with at most p starting nodes. As each iteration takes time $\mathcal{O}(3^p n^{p+2} + n^{p+1}m)$ we get a total running time of $\mathcal{O}\left(n^{3p+3d-3}\cdot k\cdot(3^p n^{p+2} + n^{p+1}m)\right) = \mathcal{O}\left(k\cdot(3^p n^{4p+3d-1} + n^{4p+3d-2}m)\right)$. □

4 Impossibility Results

In this section we investigate upper bounds for the approximation guarantees one can achieve with approximation algorithms for OptPDVC. If we allow arbitrary monotone submodular functions in OptPDVC it is easy to see that no $1-\frac{1}{e}+\epsilon$-approximation algorithm exists (unless $\mathsf{P}=\mathsf{NP}$). This is immediate by the corresponding result for Max Coverage (with cardinality constraints). In the following we investigate impossibility results that even hold if one considers phylogenetic diversity functions.

In Section 4.1 we show that when considering viability constraints there is no $1-\frac{1}{e}+\epsilon$-approximation algorithm for OptPDVC even if $\mathcal{PD}$ is additive/modular. However, the hardness proof requires food webs of linear depth.

Thus in Section 4.2 we investigate OptPDVC instances with constant depth food webs and show that maximizing the phylogenetic diversity is APX-hard.

Finally, in Section 4.3 we consider a straightforward generalization of viability constraints, where we additionally allow AND-constraints such as "species a is viable only if we preserve *both* species b and species c", and show the inapproximability of the phylogenetic diversity under these constraints.

4.1 $(1-1/e)$-Hardness for Additive Functions with Linear Depth Food Webs

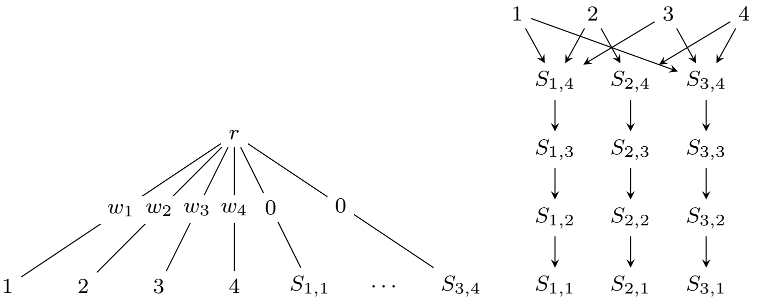
Towards our hardness result for additive functions, we first give a formal definition of the Max Coverage problem and recall the corresponding hardness result from the literature.

Definition 4 The input to the *Max Coverage* problem is a set of domain elements $D=\{1,2,\ldots,n\}$, together with non-negative integer weights $w_1,\ldots w_n$, a collection $\mathcal{S}=\{S_1,\ldots S_m\}$ of subsets of D and a positive integer k. The goal is to find a set $\mathcal{S}'\subseteq\mathcal{S}$ of cardinality k maximizing $\sum_{i\in\bigcup_{S\in\mathcal{S}'}S} w_i$.

Proposition 2 *There is no α-approximation algorithm for Max Coverage with $\alpha>1-\frac{1}{e}$ (unless $\mathsf{P}=\mathsf{NP}$) [7, 10].*

Below we give a reduction from the Max Coverage problem to OptPDVC, and then we show that it is approximation ratio preserving. The idea is to first model an additive function via phylogenetic tree as follows: We consider each element $i\in D$ as species which has an edge to the root with weight w_i. All the other species which we will use to build the appropriate food web are also connected to the root but with weight 0, i.e. the do not contribute to the value of the function. In the food web we have to encode that we may only pick an element $i\in D$ if we also pick one of the set S_j containing i. This is done by introducing species $S_{j,1}$ to $S_{j,n}$ for each set S_j and connecting i to the nodes $S_{j,n}$ for each S_j with $i\in S_j$. Finally, we guarantee that only k of the nodes $S_{j,n}$ are selected for viable sets by putting each $S_{j,n}$ as the top element of a chain of n species and setting the budget to $(k+1)\cdot n$. [2]

[2] While it is in principle possible to select $k+1$ nodes $S_{j,n}$, these sets cannot select any $i\in D$ and thus have value 0 and can be neglected.


r
w_1 w_2 w_3 w_4 0 0
1 2 3 4 $S_{1,1}$ ⋯ $S_{3,4}$
1 2 3 4
$S_{1,4}$ $S_{2,4}$ $S_{3,4}$
$S_{1,3}$ $S_{2,3}$ $S_{3,3}$
$S_{1,2}$ $S_{2,2}$ $S_{3,2}$
$S_{1,1}$ $S_{2,1}$ $S_{3,1}$

(a) Phylogenetic Tree $(T, E_{\mathcal{T}})$ with weights w_i (b) Food Web (X, E)

Fig. 3 An illustration of Reduction 1, applied to $D = \{1, 2, 3, 4\}$, $\mathcal{S} = \{S_1, S_2, S_3\}$, $S_1 = \{1, 2, 3\}$, $S_2 = \{2, 4\}$, $S_3 = \{1, 3, 4\}$.

Reduction 1 *Given an instance $(D, \mathcal{S}, k)$ of the Max Coverage problem, we build an instance of OptPDVC as follows (cf. Fig. 3)*

$$
\begin{aligned}
X &= D \cup \{S_{i,j} \mid S_i \in \mathcal{S}, 1 \le j \le n\}\\
E &= \{(j, S_{i,n}) \mid j \in S_i\} \cup \{(S_{i,j+1}, S_{i,j}) \mid 1 \le i \le m, 1 \le j < n\}\\
\mathcal{T} &= (\{r\} \cup X, \{(r, s) \mid s \in X\})\\
w_e &= \begin{cases} w_i & \text{if } e = (r, i), i \in D\\ 0 & \text{otherwise} \end{cases}\\
k' &= (k + 1) \cdot n
\end{aligned}
$$

Lemma 6 *Let $(D, \mathcal{S}, k)$ be an instance of the Max Coverage problem and let $(\mathcal{T}, (X, E), k')$ be the instance of OptPDVC given by Reduction 1. Let $W > 0$. Then there exists a cover $C \subseteq \mathcal{S}$ of size k with $w(C) \ge W$ for $(D, \mathcal{S}, k)$ iff there exists a viable set A of size $k' = (k + 1) \cdot n$ with $\mathcal{PD}(A) \ge W$.*

Proof ⇒: First assume that there is a cover C of size k with $w(C) = W$. We construct a viable set A of size k with $\mathcal{PD}(A) \ge W$. The set $A' = \{S_{i,j} \mid S_i \in C, 1 \le j \le n\} \cup \bigcup_{S_i \in C} S_i$ is a viable set of size $\le k \cdot n + n$. Clearly $\mathcal{PD}(A') = W$ and if $|A'| = k'$ we set $A = A'$ and are done. If $|A'| < k'$ we can construct a viable set A of size k' with $\mathcal{PD}(A) \ge W$ by adding arbitrary viable species.

⇐: Assume there is a viable set A of size $(k + 1) \cdot n$ with $\mathcal{PD}(A) = W$. We construct a cover C of size k with $w(C) \ge W$. There are at most $k + 1$ elements $S_i \in \mathcal{S}$ such that $S_{i,n} \in A$. This is by the fact that if $S_{i,n} \in A$ then also $S_{i,1}, \ldots, S_{i,n-1} \in A$. Now consider the case where there are exactly $k + 1$ such elements. Then we already have $(k + 1) \cdot n$ species in A and thus no $x \in D$ is contained in A. But then $\mathcal{PD}(A) = 0$ as only the edges (r, x) with $x \in D$ have non-zero weight. Assuming $W > 0$ we thus have at most k elements $S_i \in E$ such that $S_{i,n} \in A$ and further as A is viable for each $x \in A$ there is an $S_{i,n} \in A$ such that $x \in S_i$. Hence $C' = \{S_i \mid S_{i,n} \in A\}$ is of size at most k and covers all $x \in A \cap D$, i.e. $w(C') = W$. Now by adding arbitrary $S_i \in \mathcal{S}$ we can construct a cover C of size k with $w(C) \ge W$. □

Theorem 4 *There is no α-approximation algorithm for OptPDVC with $\alpha > 1 - \frac{1}{e}$ (unless $\mathsf{P} = \mathsf{NP}$), even if $\mathcal{PD}$ is an additive function.*

Proof Immediate by Proposition 2, Lemma 6 and the fact that Reduction 1 can be performed in polynomial time. □

Notice that in the above reduction the depth d of the food web graph is not bounded by a constant and in fact is linear in $|X|$.

4.2 APX-Hardness for OptPDVC with Constant Depth Food Webs

Using a result for Max Vertex Cover on bounded degree graphs, we can show APX-hardness of OptPDVC with constant depth food webs. [3]

Definition 5 The input to the *Max Vertex Cover* is a graph $G = (V, E)$ (with bounded degree) together with a a positive integer k. The goal is to find a set $S \subseteq V$ of cardinality k that covers a maximum number of edges, where an edge is covered if it is incident to at least one node in S.

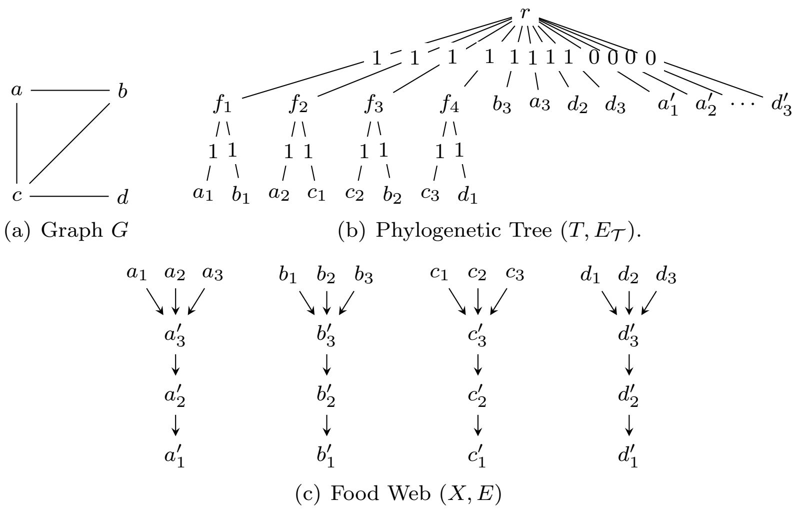
Proposition 3 *Max Vertex Cover is* APX*-hard for bounded degree graphs. In particular there is no PTAS unless* $\mathsf{P} = \mathsf{NP}$ *[15].*

Below we give an reduction of Max Vertex Cover to our OptPDVC problem. The main ideas are as follows (see also Fig. 4). Given a graph $G = (V_G, E_G)$ with max. degree Γ, for each node v of the graph we make Γ many copies $v_1, v_2, \ldots, v_\Gamma$. Then we build a phylogenetic tree with a root node r, inner nodes that correspond to the edges E_G and the copies of nodes $v \in V_G$ as leaf nodes. Each of the inner nodes is connected to the root via an edge of weight 1 and as child-nodes it has one of the copies of each of the two nodes incident to the corresponding edge in G. Again the child-nodes are connected via an edge of weight 1. Moreover, each of the nodes $v_1, v_2, \ldots, v_\Gamma$ is connected to at most one of the inner nodes and those not connected to an inner node are connected to the root via an edge of weight 1.

Then we build the food web graph of depth $\Gamma + 1$ such that for each v an optimal solution either picks all copies $v_1, v_2, \ldots, v_\Gamma$ or none of them. To achieve this we have to introduce additional nodes, which we can add to the phylogenetic tree such that they do not contribute to the phylogenetic diversity themselves, by setting the corresponding edge weights to 0.

Reduction 2 *Given an instance $(G = (V_G, E_G), k)$ of the Max Vertex Cover, with Γ being the maximum degree of G. For each node we assume an arbitrary*

[3] The authors are grateful to an anonymous reviewer who pointed them to the Max Vertex Cover problem.


r
1 1 1 1 1 1 1 1 0 0 0 0
f_1 f_2 f_3 f_4 b_3 a_3 d_2 d_3 a'_1 a'_2 $\cdots$ d'_3
1 1 1 1 1 1 1 1
a_1 b_1 a_2 c_1 c_2 b_2 c_3 d_1
a b c d
a_1 a_2 a_3 a'_3 a'_2 a'_1
b_1 b_2 b_3 b'_3 b'_2 b'_1
c_1 c_2 c_3 c'_3 c'_2 c'_1
d_1 d_2 d_3 d'_3 d'_2 d'_1

(a) Graph G

(b) Phylogenetic Tree $(T, E_{\mathcal{T}})$.

(c) Food Web (X, E)

Fig. 4 An illustration of Reduction 2, applied to the Graph G given in (a) with $\Delta = 3$.

order on the incident edges. We build an instance of OptPDVC as follows

$$
\begin{aligned}
X &= \{v_i, v'_i \mid v \in V_G, 1 \leq i \leq \Gamma\} \\
E &= \{(v_i, v'_\Gamma) \mid v \in V_G, 1 \leq i \leq \Gamma\} \cup \{(v'_i, v'_{i-1}) \mid v \in V_G, 2 \leq i \leq \Gamma\} \\
\mathcal{T} &= (\{r\} \cup E_V \cup X, E_{\mathcal{T}}) \\
E_{\mathcal{T}} &= \{(r, f) \mid f \in E_G\} \cup \{(r, v'_i) \mid v \in V_G, 1 \leq i \leq \Gamma\} \cup \\
&\quad \{(f, v_i) \mid f \textit{ is the } i\textit{-th incident edge of } v\} \cup \\
&\quad \{(r, v_i) \mid v \textit{ has degree smaller than } i\} \\
w_e &= \begin{cases} 1 & \textit{if } e = (r, v_i), v \in V_G, 1 \leq i \leq \Gamma \\ 1 & \textit{if } e = (f, v_i), f \in E, v \in V_G, 1 \leq i \leq \Gamma \\ 1 & \textit{if } e = (r, f), f \in E \\ 0 & \textit{otherwise} \end{cases} \\
k' &= 2k \cdot \Gamma
\end{aligned}
$$

Lemma 7 *Let $(G = (V_G, E_G), k)$ be an instance of Max Vertex Cover with maximal degree Γ, and let $(\mathcal{T}, (X, E), k')$ be the instance of OptPDVC given by Reduction 2. There exists a vertex cover of size k that covers W many edges iff there exists a viable set A of size k' with $\mathcal{PD}(A) \geq W + k \cdot \Gamma$.*

Proof ⇒: First assume that there is a vertex cover S of size k with W many incident edges. Now we can define a viable set $A = \{v_i, v'_i \mid v \in S, 1 \leq i \leq \Gamma\}$. It is easy to verify that A is indeed viable and $|A| = 2k \cdot \Gamma = k'$. For $\mathcal{PD}(A)$ we first have the contributions of the edge in $\mathcal{T}$ that are incident to the nodes in A which is 1 for nodes v_i and 0 for nodes v'_i, in total $k \cdot \Gamma$. Second, for each edge (u, v) in G that is covered by S one of the corresponding nodes $u_i, v_{i'}$ is in A. Thus in $\mathcal{T}$ the inner node f correspond to (u, v) has a child node in A and the edge (r, f) contributes 1 to $\mathcal{PD}(A)$. Hence, $\mathcal{PD}(A) = W + k \cdot \Gamma$.

$\Leftarrow$: Assume there is an optimal viable set A of size k' with $\mathcal{PD}(A) = W + k \cdot \Gamma$. We first prove the following claim.

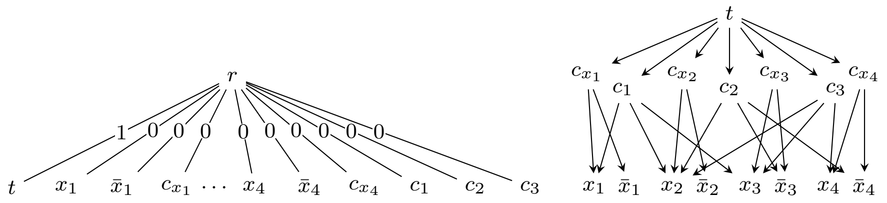
Claim: For each $v \in V_G$ either all $\{v_i, v'_i \mid 1 \leq i \leq \Gamma\}$ are contained in A or none of them.

Consider the sets $A_v = A \cap \{v_i, v'_i \mid 1 \leq i \leq \Gamma\}$ of nodes corresponding to v and included in A. As A is a viable set, as soon as we have one v_i in A all v'_i are in A. Moreover, as A is optimal also a species v'_i is only in A if at least one of the corresponding v_i is in A. Thus if A_v is non-empty then $\Gamma + 1 \leq |A_v| \leq 2\Gamma$. We have that (i) $\mathcal{PD}(A_v) \leq 2(|A_v| - \Gamma)$, as each v_i can at most cover two edges of $\mathcal{T}$ (both with weight 1) and the v'_i do not contribute to $\mathcal{PD}$ at all. Towards a contradiction let us assume there is an $v \in A$ with (ii) $|A_v| < 2\Gamma$. As, by definition of k', we can always find a viable set such that all $A_u, u \in V_G$ are either of size 2Γ or empty, there must be a set $U \subset V_G$ of nodes such that (a) $A_u \neq \emptyset$ for all $u \in U$ and (b) in total at least $|A_v|$ many nodes of $\bigcup_{u \in U}\{u_i \mid 1 \leq i \leq \Gamma\}$ are not included in $\bigcup_{u \in U} A_u$ (but all nodes $\bigcup_{u \in U}\{u'_i \mid 1 \leq i \leq \Gamma\}$ are included). That is we can consider $A \setminus A_v$ and add $|A_v|$ many of these uncovered nodes while maintaining viability. While by excluding A_v the diversity drops by less than $|A_v|$ (by (ii) $|A_v| < 2\Gamma$ and thus by (i) $\mathcal{PD}(A_v) < |A_v|$) for each node added we increase the diversity by at least 1. In total we increased the diversity, while maintaining viability, a contradiction to the optimality of A which concludes the proof of the claim.

Given the claim we can define a set cover $S = \{v \mid v \in V_G, v_1 \in A\}$ that is of size k. Now consider $\mathcal{PD}(A)$, which is composed of two parts. First, the contribution from the edges (in $\mathcal{T}$) that are incident with nodes in A. By the structure of A (given in the Claim) there are $\Gamma \cdot k$ many nodes v_i in A and each of them contributes 1. Thus their total contribution is $\Gamma \cdot k$. Second, the contribution by the edges (in $\mathcal{T}$) from the root to some inner node which has a child in A. As $\mathcal{PD}(A) = W + k \cdot \Gamma$ these inner nodes contribute W and as each of them contributes 1 there are W many such nodes. We next show that S covers at least W edges. Consider an inner node f and the corresponding edge in G. We know that in $\mathcal{T}$ one child u_i of f is included in A but then $u \in S$ and by construction u is incident to the edge f, i.e. the edge f is covered by S. Hence, S covers at least W many edges. □

Theorem 5 *OptPDVC is* APX*-hard for constant depth food webs. In particular there is no PTAS unless* P = NP.

Proof Immediate by Proposition 3, Lemma 7 and the fact that Reduction 2 can be performed in polynomial time. Also notice that in the condition $\mathcal{PD}(A) \geq W + k \cdot \Gamma$ in Lemma 7 (i) Γ is a constant as we consider bounded degree graphs for Max Vertex Cover and (ii) that for an optimal vertex cover/viable set W is at least k. i.e., we can cover at least k edges. Thus, each $(1 - \epsilon)$ approximation (resp. each PTAS) for OptPDVC would give a $(1 - \Gamma \cdot \epsilon)$ approximation (resp. a PTAS) for Max Vertex Cover. □



(a) Phylogenetic Tree $(T, E_{\mathcal{T}})$ with weights w_i

(b) Food Web (X, E)

Fig. 5 An illustration of Reduction 3, applied to the propositional formula $\varphi = (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee \neg x_4) \wedge (x_2 \vee x_3 \vee x_4)$.

4.3 Inapproximability of OptPDVC with Generalized Viability Constraints

Finally let us consider a straightforward generalization of viability constraints. So far we assumed that a species is viable iff at least one of its successors survives, but one can also imagine cases where one node needs several or even all of its successors to survive to be viable. In the following we consider food webs where we allow two types of nodes: (i) nodes that are viable if at least one successors survives and (ii) nodes that are viable only if all successors survive.

We will show that in this setting no approximation algorithm is possible using a reduction from the NP-hard problem of deciding whether a propositional formula in 3-CNF is satisfiable. A 3-CNF formula is a propositional formula which is the conjunction of clauses, and each clause is the disjunction of exactly three literals, e.g. $\varphi = (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee \neg x_4) \wedge (x_2 \vee x_3 \vee x_4)$. The main idea behind Reduction 3 is that we can build a food web such that a specific species is in a viable set iff the given formula is satisfiable.

Reduction 3 *Given a propositional formula φ in 3-CNF over propositional variables $\mathcal{V} = \{x_1, \ldots, x_n\}$ with clauses $c_1, \ldots, c_m$ build the following instance $(T, E_{\mathcal{T}})$, (X, E) and weight w_e (cf. Fig. 5) :*

$$\begin{aligned}
X &= \{c_1, \ldots, c_m\} \cup \{x, \bar{x}, c_x \mid x \in \mathcal{V}\} \cup \{t\} \\
\mathcal{T} &= (\{r\} \cup X, \{(r, s) \mid s \in X\}) \\
w_e &= \begin{cases} 1 & e = (r, t) \\ 0 & \textit{otherwise} \end{cases} \\
E &= \{(c_x, x), (c_x, \bar{x}) \mid x \in \mathcal{V}\} \cup \{(c_i, x) \mid x \in c_i\} \cup \{(c_i, \bar{x}) \mid \neg x \in c_i\} \\
&\quad \cup \{(t, c_i), (t, c_x) \mid 1 \leq i \leq m, x \in \mathcal{V}\} \\
k &= 2 \cdot |\mathcal{V}| + m + 1
\end{aligned}$$

The species $\{c_1, \ldots c_m\} \cup \{x, \bar{x}, c_x \mid x \in \mathcal{V}\}$ are viable in the traditional sense and t is viable iff all its successors survive. More formally, a set $S \subseteq X$ is viable if (i) for each $s \in S$ either s is a sink or there is a $s' \in S$ with $(s, s') \in E$ and (ii) if $t \in S$ it holds for all s' with $(t, s') \in E$ that $s' \in S$.

Lemma 8 *Given a propositional formula φ and the instance $(\mathsf{TC}, (X, E), k)$ of OptPDVC given by Reduction 3. Then φ is satisfiable iff there exists a viable set A of size $\leq k$ with $\mathcal{PD}(A) > 0$.*

Proof $\Rightarrow$: Let α be a truth assignment satisfying φ, i.e. $\alpha(\varphi) = 1$. Then it is easy to verify that $A = \{x \mid x \in \mathcal{V}, \alpha(x) = 1\} \cup \{\bar{x} \mid x \in \mathcal{V}, \alpha(x) = 0\} \cup \{c_1, \dots c_m\} \cup \{c_x \mid x \in \mathcal{V}\} \cup \{t\}$ is a viable set of size $k = 2 \cdot |\mathcal{V}| + m + 1$ with $\mathcal{PD}(A) = 1$.

$\Leftarrow$: If there is a viable subset A' with $\mathcal{PD}(A') > 0$ there is also a viable set $A \supset A'$ of size $k = 2 \cdot |\mathcal{V}| + m + 1$ and $\mathcal{PD}(A) > 0$, because $|X|$ is of size $3 \cdot |\mathcal{V}| + m + 1$. We show that the truth-assignment α setting each $s \in A \cap \mathcal{V}$ to 1 and each $s \in \mathcal{V} \setminus A$ to 0 satisfies φ. As $\mathcal{PD}(A) > 0$ we clearly have that $t \in A$. Now as A is viable and we have an AND constraint on t also $\{c_1, \dots c_m\} \cup \{c_x \mid x \in \mathcal{V}\} \subseteq A$. By $c_x \in A$ we obtain that for each $x \in \mathcal{V}$ either $x \in A$ or $\bar{x} \in A$, but not both of them (as the budget only allows $|\mathcal{V}|$ further species). Finally as $c_i \in A$ we have that for each clause there is either an $x \in C$ with $\alpha(x) = 1$ or a $\neg x \in C$ with $\alpha(x) = 0$. Thus each clause c_i is satisfied by α, i.e. $\alpha(c_i) = 1$, and hence also $\alpha(\varphi) = 1$. □

Now assuming that there is an approximation algorithm for OptPDVC with generalized viability constraints we would immediately get a procedure deciding 3-CNF formulas: apply Reduction 3 to the formula, compute $\mathcal{PD}$ using the α-approximation algorithm, and return satisfiable if $\mathcal{PD}$ is positive.

Theorem 6 *It is* NP*-hard to decide whether an instance of OptPDVC with generalized viability constraints has a viable set S with $\mathcal{PD}(S) > 0$. Thus no approximation algorithm for the problem can exist unless* $\mathsf{P} = \mathsf{NP}$.

Proof Immediate by Lemma 8, and the fact that Reduction 3 can be performed in polynomial time. □

Acknowledgments

A preliminary version of this paper has been presented at the 21st European Symposium on Algorithms (ESA'13) [4].

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement no. 340506.

References

1. Magnus Bordewich and Charles Semple. Nature reserve selection problem: A tight approximation algorithm. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 5(2):275–280, 2008.
2. Magnus Bordewich and Charles Semple. Budgeted nature reserve selection with diversity feature loss and arbitrary split systems. *Journal of mathematical biology*, 64(1-2):69–85, 2012.

3. Olga Chernomor, Bui Quang Minh, Flix Forest, Steffen Klaere, Travis Ingram, Monika Henzinger, and Arndt von Haeseler. Split diversity in constrained conservation prioritization using integer linear programming. *Methods in Ecology and Evolution*, 6(1):83–91, 2015.
4. Wolfgang Dvořák, Monika Henzinger, and David P. Williamson. Maximizing a submodular function with viability constraints. In Hans L. Bodlaender and Giuseppe F. Italiano, editors, *Algorithms - ESA 2013 - 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, volume 8125 of *Lecture Notes in Computer Science*, pages 409–420. Springer, 2013.
5. Daniel P. Faith. Conservation evaluation and phylogenetic diversity. *Biological Conservation*, 61(1):1–10, 1992.
6. Beáta Faller, Charles Semple, and Dominic Welsh. Optimizing Phylogenetic Diversity with Ecological Constraints. *Annals of Combinatorics*, 15:255–266, 2011.
7. Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.
8. M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for maximizing submodular set functions – II. *Mathematical Programming Study*, 8:73–87, 1978.
9. Pranava R. Goundan and Andreas S. Schulz. Revisiting the greedy approach to submodular set function maximization. Working Paper, Massachusetts Institute of Technology, 2007. Available at `http://www.optimization-online.org/DB_HTML/2007/08/1740.html`.
10. Samir Khuller, Anna Moss, and Joseph Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.
11. Jon Lee, Vahab S. Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 323–332. ACM, 2009.
12. Vincent Moulton, Charles Semple, and Mike Steel. Optimizing phylogenetic diversity under constraints. *Journal of Theoretical Biology*, 246(1):186 – 194, 2007.
13. G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions — I. *Mathematical Programming*, 14:265–294, 1978.
14. Fabio Pardi and Nick Goldman. Species choice for comparative genomics: being greedy works. *PLoS Genetics*, 1(6):e71, 2005.
15. Erez Petrank. The hardness of approximation: Gap location. *Computational Complexity*, 4:133–157, 1994.
16. Tsan sheng Hsu, Kuo-Hui Tsai, Da-Wei Wang, and D. T. Lee. Two variations of the minimum steiner problem. *Journal of Combinatorial Optimization*, 9(1):101–120, 2005.
17. Mike Steel. Phylogenetic diversity and the greedy algorithm. *Systematic Biology*, 54(4):527–529, 2005.
18. C. Martijn van der Heide, Jeroen van den Bergh, and Ekko van Ierland. Extending weitzman's economic ranking of biodiversity protection: combining ecological and genetic considerations. *Ecological Economics*, 55(2):218–223, 2005.
19. Jan Vondrák. Submodular functions and their applications. SODA 2013 plenary talk. Slides available at `http://theory.stanford.edu/~jvondrak/data/SODA-plenary-talk.pdf`.
20. Martin L. Weitzman. The Noah's ark problem. *Econometricay*, 66:1279 – 1298, 1998.