PaNDA: Efficient Optimization of Phylogenetic Diversity in Networks

Niels Holtgrefe* Leo van Iersel† Ruben Meuwese‡ Yukihiro Murakami§
Jannik Schestag¶

Delft University of Technology, Delft, The Netherlands
{n.a.l.holtgrefe, l.j.j.vanIersel, r.h.meuwese, y.murakami, j.t.schestag}@tudelft.nl

Abstract

Phylogenetic diversity plays an important role in biodiversity, conservation, and evolutionary studies by measuring the diversity of a set of taxa based on their phylogenetic relationships. In phylogenetic trees, a subset of k taxa with maximum phylogenetic diversity can be found by a simple and efficient greedy algorithm. However, this algorithmic tractability is lost when considering phylogenetic networks, which incorporate reticulate evolutionary events such as hybridization and horizontal gene transfer. To address this challenge, we introduce `PaNDA` (Phylogenetic Network Diversity Algorithms), the first software package and interactive graphical user-interface for exploring, visualizing and maximizing diversity in phylogenetic networks. `PaNDA` includes a novel algorithm to find a subset of k taxa with maximum diversity, running in polynomial time for networks of bounded *scanwidth*, a measure of tree-likeness of a network that grows slower than the well-known *level* measure. This algorithm considers the variant of phylogenetic diversity on networks in which the branch lengths of all paths from the root to the selected taxa contribute towards their diversity. We demonstrate the scalability of this algorithm on simulated networks, successfully analyzing level-15 networks with up to 200 taxa in seconds. We also provide a proof-of-concept analysis using a phylogenetic network on *Xiphophorus* species, illustrating how the tool can support diversity studies based on real genomic data. The software is easily installable and freely available at https://github.com/nholtgrefe/panda. Additionally, we extend the definition of phylogenetic diversity to semi-directed phylogenetic networks, which are mixed graphs increasingly used in phylogenetic analysis to model uncertainty of the root location. We prove that finding a subset of k taxa with maximum diversity remains NP-hard on semi-directed networks, but do present a polynomial-time algorithm for networks with bounded level.

Keywords. Phylogenetic Diversity, Phylogenetic Network, Semi-Directed Network, Scanwidth, Software

*ORCID: 0009-0001-6162-9668. Supported by grant OCENW.M.21.306 from the Dutch Research Council (NWO).
†ORCID: 0000-0001-7142-4706. Supported by grant OCENW.M.21.306 from NWO.
‡ORCID: 0000-0002-6884-8222. Supported by grant OCENW.GROOT.2019.015 from NWO.
§ORCID: 0000-0003-1355-5884.
¶ORCID: 0000-0001-7767-2970. Supported by grant OCENW.GROOT.2019.015 from NWO.

1 Introduction

A prominent measure of biodiversity of a set of species, or taxa, is their *phylogenetic diversity* [12], which has found widespread application in conservation [15, 16, 21], biogeography [33], and human health [4, 29]. Defined as the total branch length of the subtree induced by the considered taxa in a phylogenetic tree, phylogenetic diversity serves as a key indicator of heterogeneity and evolutionary remoteness [8]. In part, phylogenetic diversity has earned its popularity because the associated optimization problem, in which a subset of k taxa with maximum phylogenetic diversity is determined, can be solved efficiently by a greedy algorithm [31, 38].

Phylogenetic networks, which generalize phylogenetic trees by accounting for reticulate evolutionary events such as hybridization and horizontal gene transfer, offer a more realistic view of the evolutionary history of taxa in many cases [3, 27]. With the rise of methods that infer phylogenetic networks [27], the natural question has been raised: How should phylogenetic diversity be measured in phylogenetic networks? One intuitive measure is *all-paths phylogenetic diversity* [42], which generalizes the definition on trees by considering the total length of branches on all paths from the root to the considered set of taxa. However, accommodating reticulate events comes at a computational cost: the associated maximization problem Maximize-All-Paths-PD (MapPD) (see Section 2.1 for definitions) is NP-hard [7].

Although some algorithms have been presented for MapPD [7, 22, 23], these are of a theoretical nature and have not been implemented. Also, other variants of phylogenetic diversity on networks have recently received much interest from a theoretical point of view [7, 22, 23, 39, 40], but practical work has been restricted to [42], and its brute-force approach makes it only applicable to very small data sets. Related practical work was also presented in [9, 41], but these consider different types of input: trees with feature presence-absence data [9] or unrooted data-display networks [41]. None of these includes an easy-to-use and scalable software tool.

To address this gap, we introduce `PaNDA` (Phylogenetic Network Diversity Algorithms): an interactive software package for diversity optimization, visualization, and exploration in phylogenetic networks. `PaNDA` contains a novel algorithm that optimally solves MapPD in polynomial time on networks of bounded *scanwidth* [6], a measure of tree-likeness that is particularly suitable for designing practical algorithms for phylogenetic networks. For example, networks with small scanwidth may still have large *level* [20], a well-known measure of network tree-likeness defined as the maximum number of edges that needs to be deleted per *blob* (reticulated or biconnected component) to turn the network into a tree. The algorithm has running time $\mathcal{O}(2^{\mathrm{sw}}\,\mathrm{sw}\,k^2 m)$, with sw the scanwidth, m the number of edges and k the number of taxa to select. This excludes the time necessary to compute the scanwidth, which can be done efficiently in practice [20] and was fast enough for all considered instances.

To make diversity analysis in networks accessible and user-friendly, `PaNDA` is supplied with a graphical user-interface (GUI) that facilitates both optimization and exploration, see Figure 1. Through this interface, users can use the underlying MapPD-algorithm to find subsets of taxa of maximum phylogenetic diversity, interactively explore how diversity changes across different combinations of taxa, and visualize networks. Moreover, `PaNDA` provides an excellent framework to which other variants can be added in future work, which will make it easier to compare and combine these variants in benchmark studies and biological applications.

We evaluate the performance of our algorithm through extensive simulations and show that it scales well, even on networks far beyond the size and complexity currently supported by existing network inference methods [27]. Specifically, optimal solutions on level-15 phylogenetic networks with up to 200 leaves can be found in seconds. To demonstrate its real-world applicability, we use `PaNDA` to study a phylogenetic network on a set of *Xiphophorus* species, showcasing how it can be used to meaningfully analyze empirical data and support biologically relevant insights.

In addition to this practical contribution, we introduce a new variant of the MapPD problem where the input network is a *semi-directed* phylogenetic network which contains both directed and undirected edges. These networks have gained attention recently as they account for uncertainty in root placement, addressing unidentifiability issues that arise under several evolutionary models [2, 14]. We show that MapPD on semi-directed networks remains NP-hard and we present an algorithm solving it in $\mathcal{O}(2^{\ell_v}\ell_v^2\,nmk^2)$ time, with n the total number of taxa and ℓ_v the *visible vertex level*, a parameter that can be much smaller than the level. The main advantage of this algorithm is that it can be applied directly to semi-directed networks produced by statistically consistent inference methods [1, 17, 36]. Both presented algorithms are applicable to binary as well as to nonbinary networks.

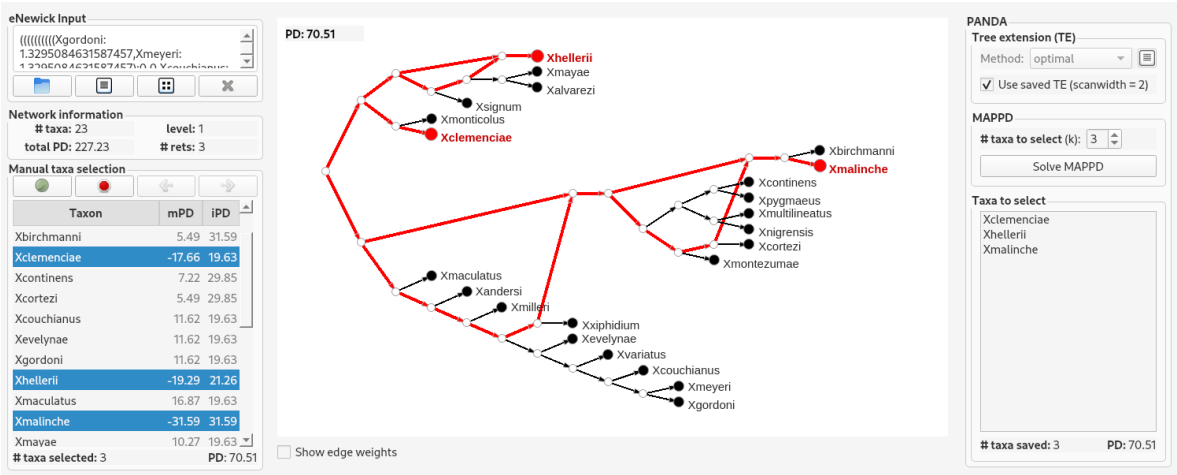

Figure 1: The PaNDA graphical user-interface with the network presented in Figure 5 as input.

2 Methods

2.1 Preliminaries

For $k \in \mathbb{N}_{>0}$, we write $[k] = \{1, \ldots, k\}$ and $[k]_0 = \{0, 1, \ldots, k\}$.

Graphs. Throughout this paper, we consider only simple mixed graphs, i.e., graphs that may contain both directed and undirected edges but no multiple edges between the same pair of vertices. Given a mixed graph G, for a vertex $v \in V(G)$, we use $\Delta^+_G(v)$ to denote its set of out-neighbours and $\Delta^-_G(v)$ for its set of in-neighbours, where we omit the subscript if G is clear from context. For $U \subseteq V(G)$, we denote by $G[U]$ the *subgraph induced by* U.

We consider four types of paths in a mixed graph. An *up-down path* P between two vertices u and v, called *endpoints*, is a vertex-disjoint path containing both undirected and directed edges where no two directed edges are oriented towards each other. P is also *semi-directed path* if all its directed edges are oriented in the same direction. Moreover, P is an *edge path* if it contains no directed edges, and it is a *directed path* if it contains only directed edges.

Phylogenetic networks. A *directed (phylogenetic) network* $\mathcal{N}^+$ on a set of at least two taxa X is a directed acyclic graph such that (i) there is a unique vertex with in-degree zero (the *root*) and its out-degree is at least two; (ii) the vertices with out-degree zero (the *leaves*) have in-degree one and they are bijectively labeled by elements of X; (iii) all other vertices either have in-degree one and out-degree at least two (*tree vertices*), or in-degree two and out-degree one (*reticulation vertices*). See Figure 2a(i) for an example. The edges entering a reticulation vertex are called *reticulation edges*.

A *semi-directed (phylogenetic) network* $\mathcal{N}^-$ on a set of at least two taxa X is a mixed graph which can be obtained from a directed network $\mathcal{N}^+$ on X by replacing every edge with an undirected edge except for the reticulation edges, and subsequently suppressing degree two vertices and identifying parallel edges. See Figure 2a(ii). Note that we may still refer to the leaves, tree vertices, reticulation vertices and reticulation edges of a semi-directed network. The reticulation edges are precisely the directed edges. We say that a vertex u is *above* a non-reticulation vertex v if there exists a semi-directed path from u to v. We say that a vertex u is *above* a reticulation vertex v if there exists a semi-directed path that is not an edge path from u to v.

Throughout this paper, we assume that every (directed or semi-directed) network $\mathcal{N}$ is equipped with an *edge weight* function $\omega : E(\mathcal{N}) \to \mathbb{R}_{\geq 0}$, which may, e.g., represent branch lengths. We call $\mathcal{N}$ *binary* if the degree of each vertex is either one or three, with the exception of a possible root which has degree two. A *blob* B of $\mathcal{N}$ is a maximal subgraph without any cut-edges, and it is *non-trivial* if it contains at least one reticulation vertex. A reticulation r is *visible* if there is a leaf x such that for every vertex u above r, every semi-directed path from u to x passes through r. The *visible vertex level* of $\mathcal{N}$ is the maximum number of visible reticulations in any of its blobs, and the *vertex level* of $\mathcal{N}$ is the maximum number of reticulations in any of its blobs. For binary networks, the vertex level equals the standard notion of the *level* of a network: the maximum number of reticulation edges to be deleted in any of its

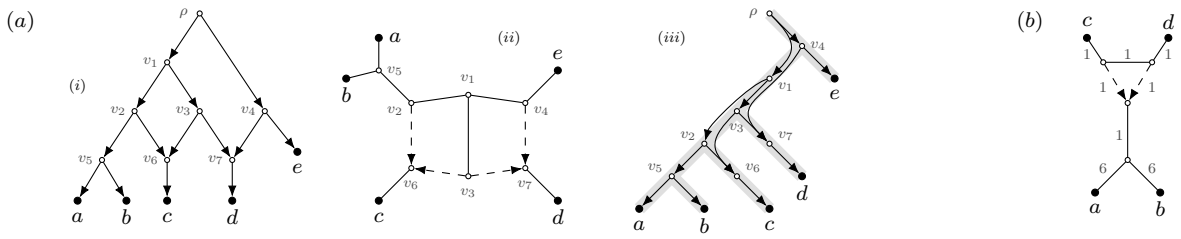

Figure 2: (a): A directed phylogenetic network on $X = \{a, b, c, d, e\}$ (i), its associated semi-directed network (ii), and an optimal tree-extension of the directed network with a scanwidth of 3 (iii). (b): A semi-directed network on $\{a, b, c, d\}$ with edge-weights.

blobs to turn the network into a tree. We reserve the letters n and m for the number of leaves and edges in a network, respectively.

Phylogenetic diversity. A *root-leaf path* in a directed network $\mathcal{N}^+$ on X is a directed path from the root of $\mathcal{N}^+$ to a leaf in X. A leaf $x \in X$ is in the *offspring* $\text{off}(e)$ of an edge e, if there is a root-leaf path to x that traverses e. Given a set of leaves $Y \subseteq X$, we use the notation $\text{RP}(Y)$ to denote the set of all edges on root-leaf paths towards leaves in Y. The *all-paths phylogenetic diversity* $\text{PD}_{\mathcal{N}^+}(Y)$ *of* Y is then

$$\text{PD}_{\mathcal{N}^+}(Y) := \sum_{e \in \text{RP}(Y)} \omega(e). \tag{1}$$

We now formally define MapPD, introduced and proved to be NP-hard even for binary networks in [7].

Maximize-All-Paths-PD (MapPD)
Input: A directed network $\mathcal{N}^+$ on X with edge-weight function ω; integers k, D.
Question: Is there a set $Y \subseteq X$ of size at most k, such that $\text{PD}_{\mathcal{N}^+}(Y) \geq D$?

Since semi-directed networks have no roots by definition, we define a new phylogenetic diversity score for a semi-directed network $\mathcal{N}^-$ on X based on up-down paths. Given a set of vertices Y, we use the notation $\text{UP}(Y)$ to denote the set of all edges on up-down paths between any pair of vertices in Y. Let $Y \subseteq X$ be a set of leaves. The *all-paths semi-directed phylogenetic diversity* $\text{SPD}_{\mathcal{N}^-}(Y)$ *of* Y is then

$$\text{SPD}_{\mathcal{N}^-}(Y) := \sum_{e \in \text{UP}(Y)} \omega(e), \tag{2}$$

where we say that an edge uv *contributes* to $\text{SPD}_{\mathcal{N}^-}(Y)$ if $uv \in \text{UP}(Y)$. For example, letting $\mathcal{N}^-$ be the network in Figure 2(b), $\text{SPD}_{\mathcal{N}^-}(\{a, b\}) = 12$ and $\text{SPD}_{\mathcal{N}^-}(\{a, c\}) = 11$. Observe that SPD has the attractive property that it is a generalization of the standard definition of phylogenetic diversity for unrooted phylogenetic trees [12]. With this new notion, we can define the new problem MapSPD.

Maximize-All-Paths-SPD (MapSPD)
Input: A semi-directed network $\mathcal{N}^-$ on X with edge-weight function ω; integers k, D.
Question: Is there a set $Y \subseteq X$ of size at most k, such that $\text{SPD}_{\mathcal{N}^-}(Y) \geq D$?

MapPD can be reduced to MapSPD by adding a high-weight edge from the root to a new leaf and then making the network semi-directed. This results in Theorem 1.

Theorem 1. MapSPD *is* NP*-hard, even for binary semi-directed networks.*

Note that the reverse direction is not as obvious as it may seem: naively iterating over all possible roots of a semi-directed network and solving MapPD does not solve MapSPD. Consider, for example, the network in Figure 2(b) with $k = 2$: no rooting of the network has an optimal MapPD solution corresponding to an optimal MapSPD solution in the semi-directed network.

Algorithm 1: Solving MAPPD parameterized by the scanwidth

Data: An instance $\mathcal{I} = (\mathcal{N}, X, \omega, k, D)$ of MAPPD, a tree extension $T_{\mathcal{N}}$ of $\mathcal{N}$.
Result: `yes`, if $\mathcal{I}$ is a `yes`-instance of MAPPD; otherwise, `no`.

1 **if** $\mathcal{N}$ *is nonbinary* **then**
2 Replace $\mathcal{N}$ by a binary resolution and change $\mathcal{T}_{\mathcal{N}}$ accordingly, as outlined in the proof of Lemma 2
3 **foreach** $v \in V(\mathcal{T}_{\mathcal{N}})$ **do** `// using a reverse BFS-traversal`
4 **foreach** $\Phi \subseteq \mathrm{GW}(v)$ *and* $t \in [k]_0$ **do**
5 **if** $\Phi = \emptyset$ *and* $t = 0$ **then**
6 $\mathrm{DP}[v, \Phi, t] \leftarrow 0$
7 **else if** $\Phi \neq \emptyset$ *and* $t = 0$ **then**
8 $\mathrm{DP}[v, \Phi, t] \leftarrow -\infty$
9 **else if** $v \in X$ *and* $t \geq 1$ **then**
10 $\mathrm{DP}[v, \Phi, t] \leftarrow \omega(\Phi)$
11 **else if** v *has one child* u *in* $\mathcal{T}_{\mathcal{N}}$ **then**
12 $\mathrm{DP}[v, \Phi, t] \leftarrow \max_{\Psi \subseteq \Delta^+(v)} \mathrm{DP}[u, (\Phi \cup \Psi) \cap \mathrm{GW}(u), t] + \omega(\Phi \cap \Delta^-(v))$
13 **else if** v *has two children* u, w *in* $\mathcal{T}_{\mathcal{N}}$ **then**
14 $\mathrm{DP}[v, \Phi, t] \leftarrow \max_{t' \in [t]_0, \Psi \subseteq \Delta^+(v)} \mathrm{DP}[u, (\Phi \cup \Psi) \cap \mathrm{GW}(u), t']$
15 $+ \mathrm{DP}[w, (\Phi \cup \Psi) \cap \mathrm{GW}(w), t - t'] + \omega(\Phi \cap \Delta^-(v))$
16 **if** $\mathrm{DP}[\rho, \emptyset, k] < D$ *where* ρ *is the root of* $\mathcal{T}_{\mathcal{N}}$ **then**
17 **return** `no`
18 **return** `yes`

Scanwidth. Given a directed acyclic graph G, we define a relation $\succ_G$ on $V(G)$ where $u \succ_G v$ if there exists a directed path from u to v in G. We extend this relation to $\succeq_G$, where $u \succeq_G v$ if either $u \succ_G v$ or $u = v$. A *tree extension* of a directed acylic graph G is a rooted tree $\mathcal{T}_G$ with $V(\mathcal{T}_G) = V(G)$ and such that $u \succ_G v$ implies $u \succ_{\mathcal{T}_G} v$ for all $u, v \in V(G)$. For every vertex $v \in V(\mathcal{T}_G)$, we let $\mathrm{GW}_{\mathcal{T}_G}(v) = \{(x, y) \in E(G) : x \succeq_{\mathcal{T}_G} v \succ_{\mathcal{T}_G} y\}$, where we often omit subscripts if G and $\mathcal{T}_G$ are clear from context. The *scanwidth* $\mathrm{sw}_{\mathcal{T}_G}$ of the tree extension $\mathcal{T}_G$ is $\max_{v \in V(\mathcal{T}_G)} |\mathrm{GW}_{\mathcal{T}_G}(v)|$ and we call $\mathcal{T}_G$ *optimal* if it has the smallest scanwidth among all tree extensions of G. The *scanwidth* sw_G of a directed acyclic graph G, first introduced in [6], is the scanwidth of an optimal tree extension of G. See Figure 2a(iii) for an example. The subtree of $\mathcal{T}_{\mathcal{N}}$ rooted at v is denoted by $\mathcal{T}_{\mathcal{N}}^{(v)}$.

2.2 Algorithm for directed networks

We present Algorithm 1, which solves an instance $(\mathcal{N}, X, \omega, k, D)$ of MAPPD in polynomial time for directed networks of bounded scanwidth. The algorithm considers a tree-extension $\mathcal{T}_{\mathcal{N}}$ in a bottom-up fashion and uses a dynamic programming table to store intermediate solutions. We utilize a *binary resolution* of the network to handle cases where the input network is nonbinary. This is outlined in Lemma 2. The time complexity and correctness of the algorithm are proved in Theorem 2.

Theorem 2. *Algorithm 1 solves instances* $(\mathcal{N}, X, \omega, k, D)$ *of* MAPPD *in* $\mathcal{O}(2^{\mathrm{sw}}\,\mathrm{sw} \cdot k^2 m)$ *time, if a tree-extension* $\mathcal{T}_{\mathcal{N}}$ *of* $\mathcal{N}$ *of scanwidth* sw *is given.*

The intuition of the table DP at the core of Algorithm 1 is as follows. For a vertex $v \in V(\mathcal{N})$, a set of edges $\Phi \subseteq \mathrm{GW}(v)$, and an integer $t \in [k]_0$, the table entry $\mathrm{DP}[v, \Phi, t]$ stores $\omega(\Phi)$ plus the maximum phylogenetic diversity of a set $A \subseteq X$ of size t in $\mathcal{N}[V(\mathcal{T}_{\mathcal{N}}^{(v)})]$ where $\mathrm{off}(e) \cap A \neq \emptyset$ for each $e \in \Phi$. If such a set does not exist, it stores $-\infty$, which, in practice, can be replaced by a large negative value. When v is the root ρ of the tree-extension, Φ is empty, and t equals k, the table therefore stores the maximum phylogenetic diversity in $\mathcal{N}$ of a set of k leaves, thereby solving MAPPD. Lines 6, 8 and 10 of the algorithm cover the three base cases of the table, whereas Lines 12 and 14 contain recurrence relations that compute table entries based on entries corresponding to vertices further down in the tree.

We note that $\mathcal{N}[V(\mathcal{T}_{\mathcal{N}}^{(v)})]$ may not necessarily be a directed network, and hence we naturally extend the definition of $\mathrm{PD}_{\mathcal{N}}$ to general directed acyclic graphs.

2.3 Algorithm for semi-directed networks

In this section, we study a generalization of the all-paths semi-directed phylogenetic diversity $\mathrm{SPD}_{\mathcal{N}}$. Let an integer k, a semi-directed network $\mathcal{N}$ and functions $\sigma : X \to [k]_0$ and $f : X \times [k]_0 \to \mathbb{R}_{\geq 0} \cup \{-\infty\}$ be given. Denote by X_σ the set of leaves x with $\sigma(x) > 0$. We define the *generalized phylogenetic diversity* $\mathrm{GPD}_{\mathcal{N}}(\sigma)$ as

$$\mathrm{GPD}_{\mathcal{N}}(\sigma) := \mathrm{SPD}_{\mathcal{N}}(X_\sigma) + \sum_{x \in X} f(x, \sigma(x)), \tag{3}$$

where we say that *a budget of h is spent on a set of leaves L* if $\sum_{x \in L} \sigma(x) \leq h$ and $\sigma(x) > 0$ for each $x \in L$. In a sense, σ models how much of the 'budget' k is 'invested' in each taxon, and f models additional diversity to be acquired, dependent on the choice of σ. We study the following problem, where one can replace $-\infty$ with a large negative value in practice.

Generalized-Maximize-All-Paths-SPD (G-MapSPD)	
Input:	A semi-directed network $\mathcal{N}$ on X, integers $k, D \in \mathbb{N}_{>0}$, and functions $f : X \times [k]_0 \to \mathbb{R}_{\geq 0} \cup \{-\infty\}$ and $\omega : E(\mathcal{N}) \to \mathbb{R}_{\geq 0}$.
Question:	Is there a function $\sigma : X \to [k]_0$ such that $\sum_{x \in X} \sigma(x) \leq k$ and $\mathrm{GPD}_{\mathcal{N}}(\sigma) \geq D$?

Observe that MapSPD is a special case of G-MapSPD, where $f(x, t) = 0$ for each $x \in X$, and $t \in [k]_0$. Thus, G-MapSPD is NP-hard by Theorem 1. Budgeted Max-PD [32] is also a special case of G-MapSPD.[1]

We present Algorithm 2, which solves instances $(\mathcal{N}, X, f, \omega, k, D)$ of G-MapSPD recursively in polynomial time for networks with bounded visible vertex level. Below, we explain the steps and notation in the algorithm. The correctness and running time of the algorithm are proven in Theorem 3, with Corollary 1 following by using the reduction in the proof of Theorem 1.

Theorem 3. *Algorithm 2 solves instances $(\mathcal{N}, X, f, \omega, k, D)$ of* G-MapSPD *in $\mathcal{O}(2^{\ell_v} \ell_v^2 \cdot nmk^2)$ time, where ℓ_v is the visible vertex level, n is the number of vertices, and m is the number of edges in $\mathcal{N}$.*

Corollary 1. *An instance $(\mathcal{N}, X, \omega, k, D)$ of* MapPD *can be solved in $\mathcal{O}(2^{\ell_v} \ell_v^2 \cdot nmk^2)$ time, where ℓ_v is the visible vertex level of $\mathcal{N}$.*

The algorithm first uses Rule 0 to reduce all *2-blobs*: blobs with exactly 2 incident cut-edges.

Reduction Rule 0. *Let B be a 2-blob with incident cut-edges $\{u, v\}$ and $\{w, z\}$ such that $v, w \in V(B)$. Remove B, $\{u, v\}$ and $\{w, z\}$ and add the edge $\{u, z\}$ with weight $\sum_{e \in E(B)} \omega(e) + \omega(\{u, v\}) + \omega(\{w, z\})$.*

The algorithm relies on three further reduction rules that reduce trees to single vertices. Rule 1 reduces *cherries*—two leaves connected to the same vertex—to a single leaf. Rule 2 reduces degree-two vertices that may arise from applying Rule 1. Exhaustively applying these two rules to a tree creates a single edge, which can be reduced to a single vertex with Rule 3. For integers i, j, k, we use $\delta_{i<j<k}$ to denote the *Kronecker delta*, i.e., $\delta_{i<j<k}$ equals 1 if $i < j < k$, and 0 otherwise.

Reduction Rule 1. *For given edges $\{v, x\}$ and $\{v, y\}$ of leaves x, y, add a new leaf z with edge $\{v, z\}$ of weight 0 and set $f'(z, i)$ to $\max_{j=0}^{i} \varphi(v, x, y, j, i)$ with $\varphi(v, x, y, j, i) := f(x, j) + f(y, i - j) + \delta_{0<j<k} \cdot \omega(\{v, x\}) + \delta_{0<i-j<k} \cdot \omega(\{v, y\})$ for each $i \in [k]_0$. Then, remove x and y with incident edges.*

Reduction Rule 2. *Let v be a degree-2-vertex with edges $\{u, v\}$ and $\{v, w\}$. Add an edge $\{u, w\}$ of weight $\omega(\{u, v\}) + \omega(\{v, w\})$ and remove v with its incident edges.*

Reduction Rule 3. *Let $\{x, y\}$ be the single edge of a network. Introduce a vertex v. For each $i \in [k]_0$, set $f(v, i)$ to $\max_{j=0}^{i} f(x, j) + f(y, i - j) + \delta_{0<j<k} \cdot \omega(\{x, y\})$. Remove x, y and $\{x, y\}$.*

Before we continue, we define a non-trivial blob B to be a *lowest* blob with a *designated vertex* $v \in V(B)$, if

- the neighborhood of $V(B) \setminus \{v\}$ consists of vertices from $V(B)$ and a subset of the leaves. We call these vertices which are not in $V(B)$ the *leaf neighbors of B*;

[1] Budgeted Max-PD [32] is a variant of Max-PD on unrooted phylogenetic trees in which each taxon x has a cost c_x of survival. It is a special case of G-MapSPD, if C—the biggest cost of a taxon—is polynomial in the input size. That is, by setting $f(x, h)$ to $-\infty$ for $h < c_x$, and 0 otherwise. Budgeted Max-PD can be solved in $\mathcal{O}(k^2 n)$ time [32], or in $\mathcal{O}(C^2 n^3)$ time [25]. However, Budgeted Max-PD is NP-hard, when taxa can have arbitrarily big costs. In that case, defining f is not possible in polynomial time.

Algorithm 2: Solving G-MapSPD parameterized by the visible vertex level

Data: An instance $\mathcal{I} = (\mathcal{N}, X, f, \omega, k, D)$ of G-MapSPD.
Result: `yes` if $\mathcal{I}$ is a `yes`-instance of G-MapSPD; otherwise, `no`.

1 Apply Rules 0 to 3 exhaustively, updating $(\mathcal{N}, X, f, \omega)$ at every step;
2 **if** *$\mathcal{N}$ is a network on a single vertex v* **then**
3 **if** $f(v, k) \geq D$ **then**
4 **return** `yes`
5 **return** `no`
6 Find a lowest blob B;
7 $R_V \leftarrow \{$visible reticulations in $B\}$;
8 **foreach** $(P, \alpha) \in \mathcal{P}(R_V) \times \{0, 1\} \setminus \{(\emptyset, 0)\}$ **do**
9 Apply Rules 1 to 3 exhaustively on $T_{(P,\alpha)}$, and call the remaining vertex $u_{(P,\alpha)}$;
10 **if** $\alpha \in \{0, 1\}$ **and** $f(u_{(P,\alpha)}, k) + H_{(P,\alpha)} + \omega(E_{(P,\alpha)}) - m_{(P,\alpha)} \cdot M \geq D$ **then**
11 **return** `yes`
12 $h_{(v)} \leftarrow -\infty$ for each $h \in [k]_0$;
13 **foreach** $(P, \alpha) \in \mathcal{P}(R_V) \times \{2\}$ **do**
14 **foreach** $h \in [k-1]_0$ **do**
15 Apply Rules 1 to 3 exhaustively on $T_{(P,\alpha)}$, and call the remaining vertex $u_{(P,\alpha)}$;
16 $h_{(v)} \leftarrow \max\{h_{(v)}, f(u_{(P,\alpha)}, h+2) + H_{(P,\alpha)} + \omega(E_{(P,\alpha)}) - m_{(P,\alpha)} \cdot M\}$
17 $\mathcal{N}' \leftarrow \mathcal{N}$ with $B \cup B_X$ replaced by u;
18 $X' \leftarrow X \setminus B_X \cup \{u\}$;
19 $f' \leftarrow f$ for all inputs in $X \setminus B_X \times [k]_0$;
20 $f'(u, h) \leftarrow h_{(v)}$ for $h \in [k]_0$;
21 $\omega'(e) \leftarrow \omega(e)$ for all $e \in E(\mathcal{N}')$;
22 **return** Algorithm 2 applied to $(\mathcal{N}', X', f', \omega', k, D)$;

- v is above every vertex of B; and
- v has at most one non-leaf neighbor outside of B.

For networks containing exactly one blob, v can be ambiguous, but in a network of two blobs, v is unique.

If the network is not a tree, the algorithm recursively processes lowest blobs until a tree remains. Specifically, for a semi-directed network $\mathcal{N}$ with vertex set V, the algorithm identifies a lowest blob B with designated vertex v on Line 6. Such a blob exists and can be found efficiently (see Lemma 8). We let B_X be the leaf neighbors of B. The algorithm reduces $B \cup B_X$ to a single leaf u on Lines 17 and 18. On Line 20, for $h \in [k]_0$, a value $h_{(v)}$ is assigned to $f'(u, h)$. This value—computed in the for-loop on Line 14 which is discussed below—stores the maximum generalized diversity that edges in B contribute when a budget of h is spent in the leaves attached to B. Observe that then a budget of $k - h$ is spent in the part above v (the graph induced by $V \setminus \{B \cup B_X\}$).

Letting R_V denote the visible reticulations in B, the for-loop on Line 14 iterates over every set P in the powerset $\mathcal{P}(R_V)$, and over a flag $\alpha \in \{0, 1, 2\}$. With P we give a set of visible reticulations such that for each $p \in P$, some budget is forced to be spent on some leaves in $\mathcal{C}_p$: the set of vertices of $B \cup B_X$ that can be reached with an edge-path from p. The flag α indicates one of three—not necessarily distinct—options, where $\alpha = 0$ stands for not necessarily spending budget on taxa above elements in P, $\alpha = 1$ for definitely spending $h \geq 1$ in $\mathcal{C}_v$, and $\alpha = 2$ stands for definitely spending $h \geq 1$ on some taxa of $X \setminus B_X$. For each combination of P and α (ignoring the case $P = \emptyset$ and $\alpha = 0$), we then construct a tree $T_{(P,\alpha)}$ capturing the maximum generalized diversity that edges in B contribute under the scenario outlined above. On Line 9, these trees are reduced to single leaves $u_{(P,\alpha)}$ with Rules 1 to 3. The remaining lines in the for-loop amalgamate this information to compute the correct values of $h_{(v)}$.

It remains to present the construction of the trees $T_{(P,\alpha)}$, which requires more definitions. Fix a vertex w of $\mathcal{C}_v$ with outgoing edge wr, where r is a reticulation with a semi-directed path to a reticulation in P. If $P = \emptyset$, we set $w := v$. We define sets $P_0 := P$, $P_1 := P \cup \{w\}$, and $P_2 := P \cup \{v\}$. Let $V_{(P,\alpha)}$ be the vertices and $E_{(P,\alpha)}$ be the edges that are on up-down paths between two vertices in P_α. In the special case that $|P| = 1$, we define $V_{(P,0)}$ as P and $E_{(P,0)}$ as $\emptyset$. In Lemma 9 we show that $V_{(P,1)}$ and $E_{(P,1)}$ are well-defined, regardless of the vertex w that is chosen. Let $M > \omega(E(\mathcal{N})) = \sum_{e \in E(\mathcal{N})} \omega(e)$ be a fixed integer. For a set U of vertices, denote by hidden(U) the set of vertices that can *not* be reached from U

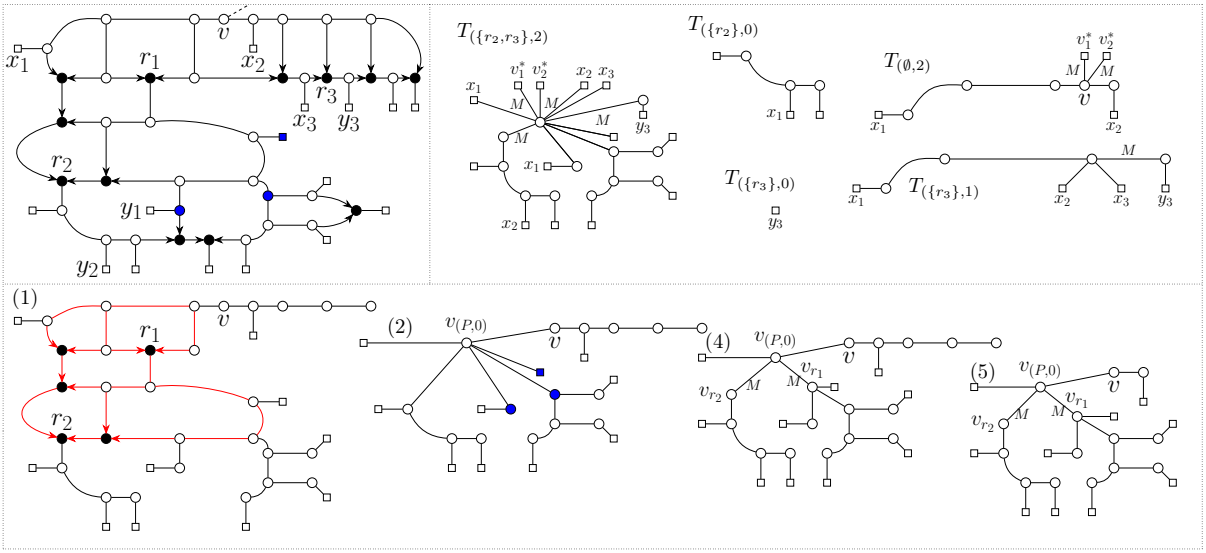

Figure 3: A graphic example of Algorithm 2. Observe, that reticulation r_i is visible, for example, with respect to y_i, for $i \in \{1, 2, 3\}$. Not all vertices are labeled. Edge weights smaller than M are omitted. **Top left**: An example lowest blob. **Bottom**: In (i): The construction of $T_{(P,0)}$ after Step i for $P = \{r_1, r_2\}$. In (1), the edges in $E_{(P,0)}$—which are contracted in Step 2—are marked in red. In (2) and the top left, vertices in $\mathcal{C}_{r_1} \setminus V_{(P,0)}$ are marked in blue. **Top right**: Five other examples of constructed trees.

with an edge-path.

We define trees $T_{(P,\alpha)}$ by taking the graph $G[B \cup B_X]$ and

1. removing hidden$(V_{(P,\alpha)})$ and the incident edges; let $H_{(P,\alpha)}$ be a the sum of $f(u, 0)$ for each leaf in hidden$(V_{(P,\alpha)})$—note that $H_{(P,\alpha)} = -\infty$, if $f(x, 0) = -\infty$ for some leaf in hidden$(V_{(P,\alpha)})$;
2. contracting the edges $E_{(P,\alpha)}$ to a single vertex $v_{(P,\alpha)}$;
3. if $\alpha = 2$, then adding two leaf-neighbors v_1^*, v_2^* to v (which is identified with $v_{(P,\alpha)}$ if $P \neq \emptyset$), setting the weight of $\{v, v_i^*\}$ to M, and $f(v_i^*, h)$ to 0 for each $h \in [k]_0$ and each $i \in \{1, 2\}$;
4. for each $r \in P$: (a) adding a vertex v_r and an edge $\{v_{(P,\alpha)}, v_r\}$ of weight M, and (b) replacing every edge $\{v_{(P,\alpha)}, w\}$ with $w \in \mathcal{C}_r$ by an edge $\{v_r, w\}$ of the same weight; and
5. exhaustively removing leaves not in X.

We define $m_{(P,\alpha)}$ to be the number of edges in $T_{(P,\alpha)}$ with a weight of M. In Figure 3, an example of the construction is given.

3 Software and Experiments

Implementation. Algorithm 1 has been implemented in our `Python` package `PaNDA` in a serial manner. The algorithm takes as input a directed phylogenetic network in `eNewick` format [10] and a tree extension, which by default is computed using software originally developed in [20]. The algorithm returns—for a given integer k—which k species maximize the all-paths phylogenetic diversity and what their diversity score is. To optimize memory usage, the solutions are computed through backtracking. Additionally, we provide a graphical user interface (GUI), allowing the user to interactively explore how diversity scores vary across different sets of taxa and to visualize phylogenetic networks (see Figure 1). The package and GUI are freely available at `https://github.com/nholtgrefe/panda`.

Simulations. We analyze the computational efficiency of our implementation through a simulation study, exploring how various network parameters influence its running time. Utilizing the R package `SiPhyNetwork` [24], we generated a dataset consisting of 6400 phylogenetic networks. Specifically, we exhaustively simulated networks under a birth-death-hybridization model until our data set consisted of 100 n-leaf level-ℓ networks for each combination of $n \in \{20, 50, 100, 200\}$ and $\ell \in \{0, \ldots, 15\}$. Then, we applied our implementation of Algorithm 1 to compute the optimal phylogenetic diversity of the networks

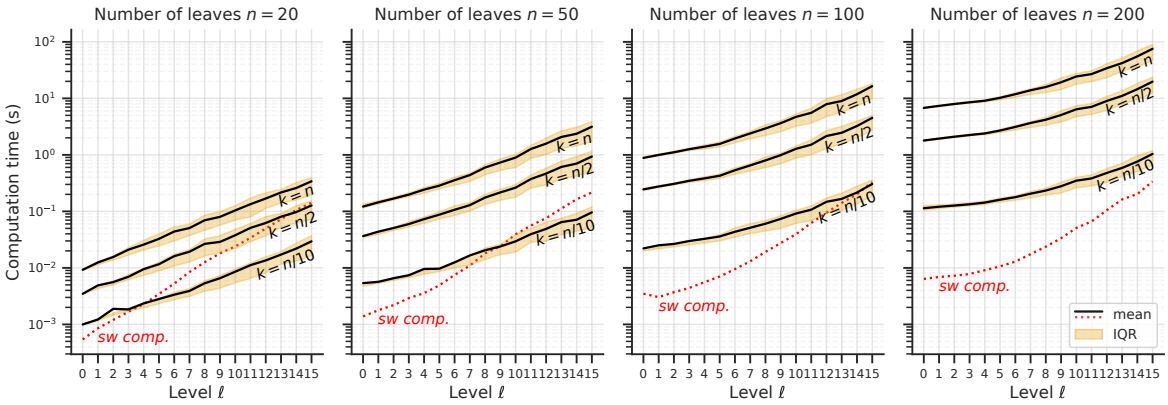

Figure 4: Computation times of Algorithm 1 for sets of 100 n-leaf level-ℓ networks, using k-values of $\frac{n}{10}$, $\frac{n}{2}$ and n. Both the mean and interquartile range (IQR)—the middle 50% of the data—are depicted. The dotted line annotated 'sw comp.' shows the mean computation time for computing an optimal tree-extension using the software from [20]; this time is not included in the other measurements.

for $k \in \{\frac{n}{10}, \frac{n}{2}, n\}$.[2] Figure 4 depicts the computation times of these experiments on a logarithmic scale. For clarity, we plot the network level, which is polynomial-time computable and a more familiar measure of tree-likeness than scanwidth within the phylogenetic network community.

Our implementation shows great scalability, solving our largest instances in the order of seconds. Importantly, the computation of an optimal tree-extension does not pose a bottleneck for these instances, consistently completing in under one second. However, for extremely large networks, it may be advantageous to employ a heuristic tree-extension [20], which can reduce preprocessing time at the expense of increased runtime for Algorithm 1. This trade-off is particularly relevant for small values of k, where the tree-extension computation gets close to dominating the total runtime of Algorithm 1.

Biological Data. To demonstrate the practical applications and capabilities of the `PaNDA` software tool, we performed a phylogenetic diversity analysis focusing on 23 *Xiphophorus* fish species consisting of three major clades: *northern swordtails*, *southern swordtails*, and *platyfishes*. This genus is known to have undergone extensive hybridization (see, e.g., [11]) and has thus been used as input for several phylogenetic network inference methods [5, 17, 28, 36]. We examine a level-1 phylogenetic network with branch lengths inferred in [5] with the `PhyloNetworks` package [37] (see Figure 5).

Using `PaNDA`, we computed the set of k species that maximize the phylogenetic diversity for various values of k. Interestingly, for $k = 3$, the optimal solution does not include one species from each of the three major clades. Instead, the selected species are *X. hellerii*, *X. malinche*, and *X. clemenciae*. The inclusion of *X. hellerii* can be attributed to its hybrid origin, which allows it to capture phylogenetic signal from multiple ancestral lineages. Similarly, *X. malinche* shares substantial ancestry with a large portion of the platyfishes, thereby contributing to the overall phylogenetic diversity in a way that compensates for the absence of a direct platyfish representative.

4 Discussion

In this paper, we introduced `PaNDA`, the first easy-to-use and scalable software package and graphical user interface designed to compute and optimize phylogenetic diversity (PD) in phylogenetic networks. Although currently only equipped with the all-paths phylogenetic diversity computations considered in this paper (Algorithm 1), in the future it can be extended with other algorithms, such as Algorithm 2 and algorithms for other PD-measures on phylogenetic networks [7, 39, 40], biodiversity indices [42], and algorithms that consider PD with ecological dependencies [13, 19, 26, 30, 34, 35]. In particular, since it is at the moment not clear which variant of phylogenetic diversity on networks is most useful in practice, it would be helpful to have them all in the same software tool so they can be compared and combined easily.

[2] In practice, choosing $k = n$ yields a trivial solution. Nevertheless, it serves to illustrate the computability of our approach even for the largest possible values of k.

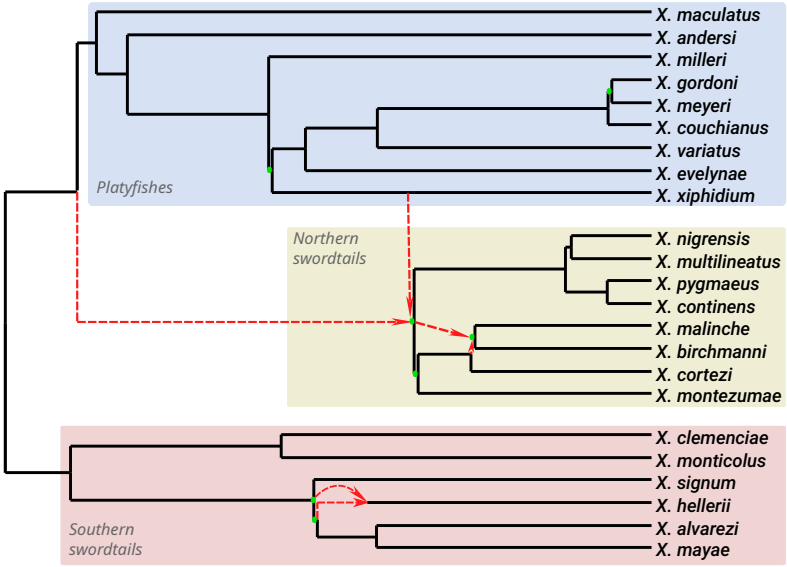

Figure 5: A level-1 phylogenetic network with branch lengths on 23 *Xiphophorus* fishes from [5]. The red dashed edges are reticulation edges, and branches with length zero are indicated by a green dot, to show the network topology.

The implementation of Algorithm 1 demonstrates excellent scalability, enabling the maximization of phylogenetic diversity for large networks (see Figure 4). Our experiments also provide evidence for the value of scanwidth as a practical parameter for algorithm design in phylogenetics, showing that the parameter grows (much) slower than the level of a network, a trend also noted in [20]. Combined with the relative elegance of Algorithm 1 compared to the theoretical contributions from [7, 22, 23], this suggests that scanwidth strikes a good balance between simplicity and tractability. Hence, it may be meaningful to explore whether a *semi-directed scanwidth* can be defined to help solve MapSPD in practice. Other interesting theoretical questions that could be useful to handle empirical data are whether the greedy approximation from [7] can be extended to the semi-directed setting, and whether ultrametricity can be exploited algorithmically.

From a practical point of view, our analysis of the *Xiphophorus* genus shows how `PaNDA` can offer a new perspective on biodiversity prioritization in the presence of reticulate events. It highlights a striking contrast with several traditional biodiversity indices, such as the *Shapley value* and the *Fair Proportion Index*, which basically measure the average contribution to phylogenetic diversity across all possible subsets of species. Such indices may rank two closely related species highly due to their individual contributions to diversity [42]. When prioritizing conservation of the top k highest ranked species [21], this can lead to suboptimal selections, since selecting two species that share a large portion of their ancestry likely yields diminishing returns. In contrast, using `PaNDA` to solve MapPD for a fixed value of k provides a different view that also takes this interdependence into account.

Software and Data Availability. Supporting scripts, data sets and the source code of `PaNDA` are available at `https://github.com/nholtgrefe/panda`.

References

[1] E. S. Allman, H. Baños, J. A. Rhodes, and K. Wicke. NANUQ+: A divide-and-conquer approach to network estimation. *Algorithms for Molecular Biology*, 20(1):14, 2025.

[2] H. Baños. Identifying species network features from gene tree quartets under the coalescent model. *Bulletin of Mathematical Biology*, 81(2):494–534, 2019.

[3] E. Bapteste, L. van Iersel, A. Janke, S. Kelchner, S. Kelk, J. O. McInerney, D. A. Morrison, L. Nakhleh, M. Steel, L. Stougie, and J. Whitfield. Networks: expanding evolutionary thinking. *Trends in Genetics*, 29(8):439–441, 2013.

[4] S. A. Bassett, W. Young, M. P. Barnett, A. L. Cookson, W. C. McNabb, and N. C. Roy. Changes in Composition of Caecal Microbiota Associated with Increased Colon Inflammation in Interleukin-10 Gene-Deficient Mice Inoculated with Enterococcus Species. *Nutrients*, 7(3):1798–1816, 2015.

[5] P. Bastide, C. Solís-Lemus, R. Kriebel, K. William Sparks, and C. Ané. Phylogenetic Comparative Methods on Phylogenetic Networks with Reticulations. *Systematic Biology*, 67(5):800–820, 2018.

[6] V. Berry, C. Scornavacca, and M. Weller. Scanning phylogenetic networks is NP-hard. In *Proceedings of the 46th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2020)*, pages 519–530. Springer, 2020.

[7] M. Bordewich, C. Semple, and K. Wicke. On the complexity of optimising variants of phylogenetic diversity on phylogenetic networks. *Theoretical Computer Science*, 917:66–80, 2022.

[8] E. S. Brondízio, J. Settele, S. Diaz, and H. T. Ngo. *Global assessment report on biodiversity and ecosystem services of the Intergovernmental Science-Policy Platform on Biodiversity and Ecosystem Services*. IPBES, 2019.

[9] C. Bundalovic-Torma, D. Desveaux, and D. S. Guttman. RecPD: A Recombination-aware measure of phylogenetic diversity. *PLoS Computational Biology*, 18(2):e1009899, 2022.

[10] G. Cardona, F. Rosselló, and G. Valiente. Extended Newick: it is time for a standard representation of phylogenetic networks. *BMC Bioinformatics*, 9(1):532, 2008.

[11] R. Cui, M. Schumer, K. Kruesi, R. Walter, P. Andolfatto, and G. G. Rosenthal. Phylogenomics reveals extensive reticulate evolution in Xiphophorus fishes. *Evolution*, 67(8):2166–2179, 2013.

[12] D. P. Faith. Conservation evaluation and phylogenetic diversity. *Biological Conservation*, 61(1):1–10, 1992.

[13] B. Faller, C. Semple, and D. Welsh. Optimizing Phylogenetic Diversity with Ecological Constraints. *Annals of Combinatorics*, 15(2):255–266, 2011.

[14] E. Gross and C. Long. Distinguishing Phylogenetic Networks. *SIAM Journal on Applied Algebra and Geometry*, 2(1):72–93, 2018.

[15] R. Gumbs, C. L. Gray, M. Böhm, I. J. Burfield, O. R. Couchman, D. P. Faith, F. Forest, M. Hoffmann, N. J. Isaac, W. Jetz, et al. The EDGE2 protocol: Advancing the prioritisation of Evolutionarily Distinct and Globally Endangered species for practical conservation action. *PLoS Biology*, 21(2):e3001991, 2023.

[16] R. Gumbs, C. L. Gray, M. Böhm, M. Hoffmann, R. Grenyer, W. Jetz, S. Meiri, U. Roll, N. R. Owen, and J. Rosindell. Global priorities for conservation of reptilian phylogenetic diversity in the face of human impacts. *Nature Communications*, 11(1):2616, 2020.

[17] N. Holtgrefe, K. T. Huber, L. van Iersel, M. Jones, S. Martin, and V. Moulton. Squirrel: Reconstructing Semi-directed Phylogenetic Level-1 Networks from Four-Leaved Networks or Sequence Alignments. *Molecular Biology and Evolution*, 42(4):msaf067, 2025.

[18] N. Holtgrefe, K. T. Huber, L. van Iersel, M. Jones, and V. Moulton. Characterizing semi-directed phylogenetic networks and their multi-rootable variants. *arXiv preprint arXiv:2507.18772*, 2025.

[19] N. Holtgrefe, J. Schestag, and N. Zeh. Limits of Kernelization and Parametrization for Phylogenetic Diversity with Dependencies. Manuscript submitted for publication, 2025.

[20] N. Holtgrefe, L. van Iersel, and M. Jones. Exact and Heuristic Computation of the Scanwidth of Directed Acyclic Graphs. *arXiv preprint arXiv:2403.12734*, 2024.

[21] N. J. Isaac, S. T. Turvey, B. Collen, C. Waterman, and J. E. Baillie. Mammals on the EDGE: Conservation Priorities Based on Threat and Phylogeny. *PloS One*, 2(3):e296, 2007.

[22] M. Jones and J. Schestag. How Can We Maximize Phylogenetic Diversity? Parameterized Approaches for Networks. In *Proceedings of the 18th International Symposium on Parameterized and Exact Computation (IPEC 2023)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2023.

[23] M. Jones and J. Schestag. Parameterized Algorithms for Diversity of Networks with Ecological Dependencies. *arXiv preprint arXiv:2510.09512*, 2025.

[24] J. A. Justison, C. Solis-Lemus, and T. A. Heath. SiPhyNetwork: An R package for simulating phylogenetic networks. *Methods in Ecology and Evolution*, 14(7):1687–1698, 2023.

[25] C. Komusiewicz and J. Schestag. A Multivariate Complexity Analysis of the Generalized Noah's Ark Problem. In *Proceedings of the 19th Cologne-Twente Workshop on Graphs and Combinatorial Optimization (CTW 2023)*, volume 13 of *AIRO*, pages 109–121. Springer, 2023.

[26] C. Komusiewicz and J. Schestag. Maximizing Phylogenetic Diversity under Ecological Constraints: A Parameterized Complexity Study. In *Proceedings of the 44th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2024)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2024.

[27] S. Kong, C. Solís-Lemus, and G. P. Tiley. Phylogenetic networks empower biodiversity research. *Proceedings of the National Academy of Sciences*, 122(31):e2410934122, 2025.

[28] S. Martin, N. Holtgrefe, V. Moulton, and R. M. Leggett. Algebraic invariants for inferring 4-leaf semi-directed phylogenetic networks. *Systematic Biology*, page syaf071, 2025.

[29] R. Molina-Venegas, M. A. Rodriguez, M. Pardo-de Santayana, C. Ronquillo, and D. J. Mabberley. Maximum levels of global phylogenetic diversity efficiently capture plant services for humankind. *Nature Ecology & Evolution*, 5(5):583–588, 2021.

[30] V. Moulton, C. Semple, and M. Steel. Optimizing phylogenetic diversity under constraints. *Journal of Theoretical Biology*, 246(1):186–194, 2007.

[31] F. Pardi and N. Goldman. Species Choice for Comparative Genomics: Being Greedy Works. *PLoS Genetics*, 1(6):e71, 2005.

[32] F. Pardi and N. Goldman. Resource-aware taxon selection for maximizing phylogenetic diversity. *Systematic Biology*, 56(3):431–444, 2007.

[33] D. F. Rosauer and W. Jetz. Phylogenetic endemism in terrestrial mammals. *Global Ecology and Biogeography*, 24(2):168–179, 2015.

[34] J. Schestag. Weighted Food Webs Make Computing Phylogenetic Diversity So Much Harder. In *Proceedings of the 51st International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2026)*. Springer, 2026.

[35] J. Schestag and N. Zeh. A Problem Separating Treewidth and Scanwidth. Manuscript in Preparation, 2025.

[36] C. Solís-Lemus and C. Ané. Inferring Phylogenetic Networks with Maximum Pseudolikelihood under Incomplete Lineage Sorting. *PLoS Genetics*, 12(3):e1005896, 2016.

[37] C. Solís-Lemus, P. Bastide, and C. Ané. PhyloNetworks: A Package for Phylogenetic Networks. *Molecular biology and evolution*, 34(12):3292–3298, 2017.

[38] M. Steel. Phylogenetic Diversity and the Greedy Algorithm. *Systematic Biology*, 54(4):527–529, 2005.

[39] L. van Iersel, M. Jones, J. Schestag, C. Scornavacca, and M. Weller. Phylogenetic Network Diversity Parameterized by Reticulation Number and Beyond. In *Proceedings of the 22nd RECOMB International Workshop on Comparative Genomics (RECOMB-CG 2025)*, pages 107–130. Springer, 2025.

[40] L. van Iersel, J. Schestag, M. Jones, C. Scornavacca, and M. Weller. Average-Tree Phylogenetic Diversity of Networks. In *Proceedings of the 25th International Workshop on Algorithms in Bioinformatics (WABI 2025)*, pages 14:1–14:21, 2025.

[41] L. Volkmann, I. Martyn, V. Moulton, A. Spillner, and A. O. Mooers. Prioritizing Populations for Conservation Using Phylogenetic Networks. *PloS One*, 9(2):e88945, 2014.

[42] K. Wicke and M. Fischer. Phylogenetic diversity and biodiversity indices on phylogenetic networks. *Mathematical Biosciences*, 298:80–90, 2018.

A Appendix

A.1 NP-hardness proof

Theorem 1. MapSPD *is* NP*-hard, even for binary semi-directed networks.*

Proof. We reduce from MapPD, which is NP-hard even for binary networks [7].

Reduction. Let $\mathcal{I} = (\mathcal{N}^+, X, \omega, k, D)$ be an instance of MapPD with $\mathcal{N}^+$ being binary and having root ρ. Construct an instance $\mathcal{I}' = (\mathcal{N}^-, X', \omega', k', D')$ of MapSPD as follows. Let M be the sum of all edge weights plus one, and let y be a new taxon not in X. Set $X' := X \cup \{y\}$, $k' := k+1$ and $D' := D + M$. Denote by $\mathcal{N}_1^+$ the directed network obtained from $\mathcal{N}^+$ by adding the leaf y and the edge (ρ, y). Set $\mathcal{N}^-$ to be the semi-directed network obtained from $\mathcal{N}_1^+$, and set $\omega'(e) := \omega(e)$ for all $e \in E(\mathcal{N}^+)$ with $\omega'((\rho, y)) := M$.

Correctness. Since the out-degree of ρ is three, $\mathcal{N}^-$ can be obtained from $\mathcal{N}_1^+$ by only undirecting all non-reticulation edges. So, $\mathcal{N}_1^+$ and $\mathcal{N}^-$ have the same vertex and edge sets (up to the undirecting of the non-reticulation edges). Thus, $\mathcal{N}^-$ is binary and ω' is defined for all its edges. It remains to show the equivalence of $\mathcal{I}$ and $\mathcal{I}'$. To this end, let $Y \subseteq X$ be arbitrary. Then, we have that $\mathrm{UP}_{\mathcal{N}^-}(Y \cup \{y\}) = \mathrm{RP}_{\mathcal{N}_1^+}(Y \cup \{y\}) = \mathrm{RP}_{\mathcal{N}^+}(Y) \cup \{(\rho, y)\}$. Because $\omega'((\rho, y)) = M$, $\mathrm{SPD}_{\mathcal{N}^-}(Y \cup \{y\}) = \mathrm{PD}_{\mathcal{N}^+}(Y) + M$. Since M being large forces y to be in any optimal solution of $\mathcal{I}'$, the equivalence follows from $D' = D + M$ and $k' = k + 1$. Noting that the reduction takes polynomial time completes the proof. □

A.2 Proofs for Algorithm 1

Lemma 2. *Given an instance* $(\mathcal{N}, X, \omega, k, D)$ *of* MapPD *with* $\mathcal{N}$ *nonbinary and a tree-extension* $\mathcal{T}_\mathcal{N}$ *of* $\mathcal{N}$*, in* $\mathcal{O}(m)$ *time one can compute an equivalent instance* $(\mathcal{N}', X, \omega', k, D)$ *of* MapPD *and a tree-extension* $\mathcal{T}'_\mathcal{N}$ *of* $\mathcal{N}'$ *such that* $\mathcal{N}'$ *and* $\mathcal{T}'_\mathcal{N}$ *are binary,* $m' \in \mathcal{O}(m)$*, and* $\mathrm{sw}_{\mathcal{T}'_\mathcal{N}} = \mathrm{sw}_{\mathcal{T}_\mathcal{N}}$.

Proof. A rooted binary tree T with leaf set $\{x_1, \ldots, x_{d+1}\}$ and $d \geq 1$ is a *caterpillar tree* if the removal of all x_i results in a directed path $P = (v_1, \ldots, v_d)$, called the *spine*. For two leaves x_i, x_j of T with in-neighbors v_i, v_j, respectively, we say that x_i *comes before* x_j if $i \leq j$.

Construction. We let $\mathcal{N}'$ be a carefully constructed *binary resolution* of $\mathcal{N}$ and change $\mathcal{T}_\mathcal{N}$ into $\mathcal{T}'_\mathcal{N}$ accordingly (see below). We set $\omega'(e) := \omega(e)$ for all $e \in E(\mathcal{N})$ and $\omega'(e) := 0$ for all $e \notin E(\mathcal{N})$.

In particular, let $v_1 = v$ be a vertex with $d+1 \geq 3$ out-neighbors $w_1, \ldots, w_{d+1}$ in $\mathcal{N}$. Without loss of generality, assume that the w_i are ordered such that consecutive vertices belong to the same connected component of $\mathcal{T}_\mathcal{N}^{(v)} \setminus \{v\}$, i.e., they are grouped according to the branch of $\mathcal{T}_\mathcal{N}^{(v)}$ they are in. Let T be a caterpillar tree with root v_1, leaf set $\{w_1, \ldots, w_{d+1}\}$, spine $P = (v_1, \ldots, v_d)$ and the property that w_i comes before w_j in T if $i < j$. To obtain $\mathcal{N}'$ from $\mathcal{N}$, we replace v_1 and its outgoing edges (v_1, w_i) in $\mathcal{N}$ by T. To obtain $\mathcal{T}'_\mathcal{N}$ from $\mathcal{T}_\mathcal{N}$, we replace v_1 by P such that the outgoing edges (v_1, z_j) of v_1 in $\mathcal{T}_\mathcal{N}$ are replaced by edges (v_d, z_j). Similarly, let $v = v_1$ be a vertex with $d+1 \geq 3$ in-neighbors $w_1, \ldots, w_{d+1}$ in $\mathcal{N}$. Let T' be a caterpillar tree with root v_1, leaf set $\{w_1, \ldots, w_{d+1}\}$ and spine $P' = (v_1, \ldots, v_d)$. Let T be the graph obtained from T' by reversing all edge directions. To obtain $\mathcal{N}'$ from $\mathcal{N}$, we replace v and its incoming edges in $\mathcal{N}$ by T. To obtain $\mathcal{T}'_\mathcal{N}$ from $\mathcal{T}_\mathcal{N}$, we replace v_1 by the directed path $P = (v_d, \ldots, v_1)$ such that the incoming edge (z, v_1) of v_1 in $\mathcal{T}_\mathcal{N}$ is replaced by (z, v_d).

Correctness. Let $e \in E(\mathcal{N})$ be arbitrary. Then, note that our construction ensures that e is on a root-leaf path to a leaf $x \in X$ in $\mathcal{N}$ if and only if it is so in $\mathcal{N}'$. Since the new edges have a weight of zero, we obtain that $\mathrm{PD}_{\mathcal{N}'}(Y) = \mathrm{PD}_\mathcal{N}(Y)$ for all $Y \subseteq X$. Hence, the two instances are equivalent. Moreover, the construction can be done in a single traversal of the network and tree-extension, resulting in a running time of $\mathcal{O}(m)$.

For the remaining conditions, first note that m' is clearly $\in \mathcal{O}(m)$. Since $\mathcal{N}'$ is a valid binary resolution of $\mathcal{N}$, it is trivially binary. By [6, Lem. 5], we may also assume that $|\Delta^+_{\mathcal{T}_\mathcal{N}}(v)| \leq |\Delta^+_\mathcal{N}(v)|$ for all $v \in V(\mathcal{N})$, and hence our construction ensure that $\mathcal{T}'_\mathcal{N}$ is binary. Next, observe that replacing vertices in $\mathcal{T}_\mathcal{N}$ that have a high in-degree in $\mathcal{N}$ by a path preserves the property $u \succ_{\mathcal{T}'_\mathcal{N}} v$ if $u \succ_{\mathcal{N}'} v$. Similarly, our construction accounting for the different branches of the $\mathcal{T}_\mathcal{N}$ ensures the same property is preserved when replacing vertices with high out-degree in $\mathcal{N}$. Hence, $\mathcal{T}'_\mathcal{N}$ is a tree-extension of $\mathcal{N}'$. Furthermore, we have that $|\mathrm{GW}_{\mathcal{T}'_\mathcal{N}}(v)| = |\mathrm{GW}_{\mathcal{T}_\mathcal{N}}(v)|$ for all $v \in V(\mathcal{T}_\mathcal{N})$. Each new vertex $v_i \in V(\mathcal{T}'_\mathcal{N}) \setminus V(\mathcal{T}_\mathcal{N})$ is part of a path P, starting at an existing vertex $v \in V(\mathcal{T}_\mathcal{N})$. These vertices have the property $|\mathrm{GW}_{\mathcal{T}'_\mathcal{N}}(v_i)| \leq |\mathrm{GW}_{\mathcal{T}_\mathcal{N}}(v)|$. Hence, $\mathrm{sw}_{\mathcal{T}'_\mathcal{N}} = \mathrm{sw}_{\mathcal{T}_\mathcal{N}}$. □

Theorem 2. *Algorithm 1 solves instances* $(\mathcal{N}, X, \omega, k, D)$ *of* MAPPD *in* $\mathcal{O}(2^{\mathrm{sw}}\,\mathrm{sw}\cdot k^2 m)$ *time, if a tree-extension* $\mathcal{T}_\mathcal{N}$ *of* $\mathcal{N}$ *of scanwidth* sw *is given.*

Proof. Correctness. Clearly, the algorithm returns the correct result on Lines 16 to 18 if DP stores the intended values as described above. That this is true for the base cases on Lines 6, 8 and 10 of the algorithm can easily be seen. It remains to show the correctness of the two recurrences on Lines 12 and 14. To this end, let $v \in V(\mathcal{N}) \setminus X$, $\Phi \subseteq \mathrm{GW}(v)$ and $t \in [k]_0$ be arbitrary. Since the recurrence on Line 12 is a simplification of the one on Line 14, we only prove correctness of the latter and assume that v has two children u and w. As induction hypothesis, suppose that for vertices below v in $\mathcal{T}_\mathcal{N}$, DP stores the intended values. We use the shorthand notation $\mathcal{N}_v = \mathcal{N}[V(\mathcal{T}_\mathcal{N}^{(v)})]$ and $L(v, \Phi, t) = \{A \subseteq X \mid A \subseteq V(\mathcal{N}_v), \mathrm{off}(e) \cap A \neq \emptyset\, \forall e \in \Phi, |A| = t\}$.

First suppose that $\mathrm{DP}[v, \Phi, t]$ stores $d \in \mathbb{R}_{\geq 0}$. By the recurrence on line 14 and the induction hypothesis, there is some $\Psi \subseteq \Delta^+(v)$ and some $t' \in [t]_0$ such that there exist sets $A_1 \in L(u, F_u, t')$ and $A_2 \in L(w, F_w, t - t')$, where $F_u = (\Phi \cup \Psi) \cap \mathrm{GW}(u)$ an $F_w = (\Phi \cup \Psi) \cap \mathrm{GW}(w)$. Moreover,

$$d = \mathrm{PD}_{\mathcal{N}_u}(A_1) + \omega(F_u) + \mathrm{PD}_{\mathcal{N}_u}(A_2) + \omega(F_w) + \omega(\Phi \cap \Delta^-(v)). \tag{4}$$

Let $A = A_1 \cup A_2$. Since $A_1 \in L(u, F_u, t')$ and $A_2 \in L(w, F_w, t - t')$, $A \subseteq V(\mathcal{N}_v)$ and $A_1 \cap A_2 = \emptyset$, so $|A| = t > 0$. Let $e \in \Phi$. If $e \in \Delta^-(v)$, $|\,\mathrm{off}(e) \cap A| > 0$ because $|A| > 0$. If $e \notin \Delta^-(v)$, we may assume without loss of generality that $e \in \mathrm{GW}(u)$ and hence $|\,\mathrm{off}(e) \cap A_1| > 0$. So, $A \in L(v, \Phi, t)$.

We now show that A indeed satisfies $d = \mathrm{PD}_{\mathcal{N}_v}(A) + \omega(\Phi)$. First note that since $\Psi \cap \Phi = \emptyset$, the maximum in the recurrence on Line 14 is attained for a maximal Ψ such that neither of the two terms is $-\infty$. Therefore, an edge $e \in \Delta^+(v)$ with offspring in $\mathcal{N}_u$ (resp. $\mathcal{N}_v$) is in $\Psi \subseteq \Delta^+(v)$ if and only if $t' \neq 0$ (resp. $t - t' \neq 0$). Consequently, $\mathrm{PD}_{\mathcal{N}_v}(A) = \mathrm{PD}_{\mathcal{N}_u}(A_1) + \mathrm{PD}_{\mathcal{N}_w}(A_2) + \omega(\Psi)$ $(*)$.

Since $\Psi \cap \Phi = \emptyset$, $\mathrm{GW}(u) \cap \mathrm{GW}(w) = \emptyset$ and $\Psi \subseteq \mathrm{GW}(u) \cup \mathrm{GW}(w)$, we get $\omega(F_u) + \omega(F_w) = \omega(\Phi \cap \mathrm{GW}(u)) + \omega(\Phi \cap \mathrm{GW}(w)) + \omega(\Psi)$ $(**)$. Combining $(*)$ and $(**)$ with Eq. (4), we obtain

$$\begin{aligned}
\mathrm{PD}_{\mathcal{N}_v}(A) &\overset{*}{=} \mathrm{PD}_{\mathcal{N}_u}(A_1) + \mathrm{PD}_{\mathcal{N}_w}(A_2) + \omega(\Psi) \\
&\overset{4}{=} d - \omega(\Phi \cap \Delta^-(v)) + \omega(\Psi) - \omega(F_u) - \omega(F_w) \\
&\overset{**}{=} d - \omega(\Phi \cap \Delta^-(v)) - \omega(\Phi \cap \mathrm{GW}(u)) - \omega(\Phi \cap \mathrm{GW}(w)) \\
&= d - \omega(\Phi).
\end{aligned}$$

So, if $\mathrm{DP}[v, \Phi, t]$ stores $d \in \mathbb{R}_{\geq 0}$, there indeed exists a set $A \in L(v, \Phi, t)$ with $\mathrm{PD}_{\mathcal{N}_v}(A) = d - \omega(\Phi)$.

It remains to show that DP stores $-\infty$ in the correct cases. So, suppose that $\mathrm{DP}[v, \Phi, t] = -\infty$. Then, by the recurrence relation in line 14, for each $t' \in [t]_0$, either $\mathrm{DP}[u, \Phi \cup \mathrm{GW}(u), t']$ or $\mathrm{DP}[w, \Phi \cup \mathrm{GW}(w), t - t']$ stores $-\infty$. Without loss of generality, we may assume the former. Then, by the induction hypothesis, for every set $A' \subseteq X$ of size t' in $\mathcal{N}_u$, there exists an $e \in \Phi \cap \mathrm{GW}(u)$ with $\mathrm{off}(e) \cap A' = \emptyset$. Therefore, for every $A \subseteq X$ in $\mathcal{N}_v$ with the property that exactly t' of its elements are in $\mathcal{N}_u$ (and thus $t - t'$ of its elements are in $\mathcal{N}_w$), there exists an $e \in \Phi \cap \mathrm{GW}(u) \subseteq \Phi$ such that $\mathrm{off}(e) \cap A' = \emptyset$. Moreover, since $e \in \mathrm{GW}(u)$, we even have $\mathrm{off}(e) \cap A = \emptyset$. Since $t' \in [t]_0$ was arbitrary, $\mathrm{DP}[v, \Phi, t]$ correctly stores $-\infty$.

Running time. By Lemma 2, we may assume that $\mathcal{N}$ and $\mathcal{T}_\mathcal{N}$ are binary and instead show a running time of $\mathcal{O}(2^{\mathrm{sw}}\,\mathrm{sw}\cdot k^2 n)$. Since $|\,\mathrm{GW}(v)| \leq \mathrm{sw}$ for all $v \in V(\mathcal{N})$, our table's size is at most $|V(\mathcal{N})| \cdot 2^{\mathrm{sw}} \cdot (k+1)$. For each of these table entries, we perform a maximization over at most $3 \cdot (k+1) \cdot 4$ terms, since we assumed $\mathcal{N}$ is binary. Thus, the time complexity becomes $\mathcal{O}(2^{\mathrm{sw}}\,\mathrm{sw}\cdot k^2 \cdot n)$, as desired. □

A.3 Proofs for Algorithm 2

Lemma 3. *Rule 0 is correct and can be applied exhaustively in* $\mathcal{O}(m)$ *time.*

Proof. Observe that an edge in a 2-blob contributes to the diversity of a set of taxa only if it is on an up-down path between two selected taxa. In that case, all of the edges in the blob, and the two incident cut-edges, are on such an up-down path. This proves correctness. The time complexity follows from the fact that finding and removing the 2-blobs can be done exhaustively in $\mathcal{O}(m)$ time with a traversal of the network. □

Lemma 4. *Rule 1 is correct and can be applied exhaustively in* $\mathcal{O}(nk^2)$ *time.*

Proof. Let $\mathcal{I} = (\mathcal{N}, X, f, \omega, k, D)$ denote an instance of G-MapSPD before the application of a reduction rule, and $\mathcal{I}' = (\mathcal{N}', X', f', \omega', k, D)$ the instance after the application.

It is easily visible that if the original instance $\mathcal{I}$ has a solution σ, then σ', where values of $X \setminus \{x, y\}$ remain unchanged and $\sigma'(z) := \sigma(x) + \sigma(y)$, is a solution for $\mathcal{I}'$. Conversely, from a solution σ' of $\mathcal{I}'$, we can define a solution of $\mathcal{I}$ by setting $\sigma(x) := \arg\max_{j=0}^{\sigma'(z)} \varphi(v, x, y, j, \sigma'(z))$ and $\sigma(y) := \sigma'(z) - \sigma(x)$.

Every application of Rule 1 removes one leaf and takes $\mathcal{O}(k^2)$ time. □

Lemma 5. *Rule 2 is correct and can be applied exhaustively in $\mathcal{O}(n)$ time.*

Proof. Observe that only if leaves on both ends are selected, then one of the edges counts towards the diversity, and in that case, both. This shows the correctness. Further, this rule can be applied once per edge and each time in constant time. The number of edges that are on an edge-path to a leaf is $\mathcal{O}(n)$. □

Lemma 6. *A tree is reduced to a single leaf in $\mathcal{O}(nk^2)$ time, by exhaustively applying Rules 1 to 3.*

Proof. We prove that applying Rules 1 and 2 exhaustively is sufficient to correctly reduce trees to a single edge. Rule 3 then reduces the single edge to a single leaf in $\mathcal{O}(1)$ time. Its correctness can be shown analogously to Lemma 4.

Let $\mathcal{T}$ be a tree. We prove by induction on the number of leaves n, where $n = 2$ is trivial.

Let $n \geq 3$ and the claim be correct for all trees with $n - 1$ leaves. Any tree on n leaves contains a cherry, that is, two leaves x and y that share a common neighbor v, or a degree-2 vertex w. If the latter is true, apply Rule 2 exhaustively. Afterward, we can apply Rule 1 to remove one of the leaves. The resulting graph is a tree on at most $n-1$ vertices, and the claim follows by the induction hypothesis. □

Lemma 7. *Let $\mathcal{I} = (\mathcal{N}, X, f, \omega, k, D)$ be an instance of* G-MapSPD *in which we can not apply any of the Rules 1 to 3. If $\mathcal{N}$ contains a non-trivial blob B, then there exists a vertex $v \in V(B)$ which is above every vertex of B. In particular, no reticulations can be above such vertices v.*

Proof. Let r be a reticulation in B for which there are no reticulations above r (i.e., no reticulations have a semi-directed path to r). Let u be one of the in-neighbors of r. By Corollary 1 of [18], there is an up-down path between u and all vertices in B. By choice of r, there are no reticulations above u, so all such up-down paths must be a semi-directed path from u. This proves the existence part of the lemma.

To see the second part, let $v \in V(B)$ be a vertex above every vertex of B. Suppose for a contradiction that there is a reticulation r above v. Then there is a semi-directed path from r to v. Since v is above r, there is a semi-directed path which is not an edge-path from v to r. Combining these two paths gives a semi-directed cycle, which is a cycle that can be oriented as a directed cycle. But this is not allowed by Theorem 2 of [18], which gives the required contradiction. □

Lemma 8. *Let $\mathcal{I} = (\mathcal{N}, X, f, \omega, k, D)$ be an instance of* G-MapSPD *in which we can not apply any of the Rules 1 to 3. If $\mathcal{N}$ contains a non-trivial blob, then a lowest blob of $\mathcal{N}$ can be found in $\mathcal{O}(m)$ time, where m is the number of edges in $\mathcal{N}$.*

Proof. Suppose first that $\mathcal{N}$ contains one blob B. Clearly, all neighbors of vertices in B, which are not in $V(B)$, are leaves. By Lemma 7, there exists a vertex which is above every vertex of B. By definition, this is a designated vertex of B, and hence B is a lowest vertex.

Now suppose that $\mathcal{N}$ contains at least two blobs. Define $v \geq B$ for a vertex v and a blob B, if there is a semi-directed path from v to *every* vertex of B. Define $B_1 \geq B_2$ for blobs B_1 and B_2, if $v \geq B_2$ for some $v \in V(B_1)$, and if for no vertices $u \in V(B_2)$, $u \geq B_1$ holds.

Let B be a blob of $\mathcal{N}$ which is smallest with respect to this partial ordering $\geq$, which has a neighboring blob B^+ with $B^+ \geq B$. If there is no such blob, then let B be any blob of $\mathcal{N}$. If B has at most one vertex with a non-leaf neighbor not in B, then we continue our argument in the next paragraph; otherwise, B must contain two vertices v_1, v_2 with non-leaf neighbors $u_1, u_2 \notin V(B)$ from blobs B_1, B_2, respectively. Observe that v_i must have a semi-directed path to all vertices in $V(B_i)$ for at least one of $i = 1, 2$, as otherwise, $V(B) \cup V(B_1) \cup V(B_2)$ is part of a bigger blob. Without loss of generality, suppose v_1 has a semi-directed path to all vertices in $V(B_1)$. We continue this process on B_1; note that we can avoid returning to blobs that have already been visited, for example, by updating the partial ordering $\geq$ with $B \geq B_1$.

Since $\mathcal{N}$ is a finite graph, such a process must terminate. We obtain a smallest blob B^* in $\mathcal{N}$ where at most one vertex v of B^* has one non-leaf neighbor that is not in B^*. We claim that v is above every vertex of B^*, thereby showing that v is a designated vertex of B^*, and consequently that B^* is a lowest blob. Since B^* is smallest, no reticulation of B^* is above v. By Lemma 7, there exists a vertex u in B^*

which is above every vertex in B^*. Note that there must be an edge-path between u and v. Otherwise, there would be a reticulation in B^* above either u or v, which is not possible. But this means that v must also be above every vertex in B^*, by appending the edge-path from v to u to the semi-directed paths from u to the other vertices, possibly removing duplicate edges. So B^* must be a lowest blob with designated vertex v.

To show the runtime, note that one can traverse through the blobs as described above by visiting each edge at most once. Therefore, the lowest blob can be found in $O(m)$ time.

□

Lemma 9. *Let $\mathcal{N}$ be a semi-directed network with a lowest blob B with designated vertex v and let P be a set of visible reticulations in B. Let $w, w' \in \mathcal{C}_v$ be vertices with outgoing edges wr and $w'r'$ respectively, where r and r' are reticulations with a directed path to an element in P. The set of vertices and edges on up-down paths between two vertices in $P \cup \{w\}$ and those in $P \cup \{w'\}$ are equivalent.*

In other words, given a set of visible reticulations P, the definitions of $V_{(P,1)}$ and $E_{(P,1)}$ are independent of the vertex w of $\mathcal{C}_v$ which is chosen.

Proof. If $P = \emptyset$ or if $|\mathcal{C}_v| = 1$, the claim follows immediately. So assume $P \neq \emptyset$ and $|\mathcal{C}_v| \geq 2$. Let $w, w' \in \mathcal{C}_v$ be two vertices with outgoing edges wr and $w'r'$, where r and r' are reticulations with a semi-directed path to a reticulation p and p' in P, respectively. We require $w \neq w'$, but the others do not need to be distinct. We show that up-down paths of $U_{w'} := \mathrm{UP}(P \cup \{w'\})$ and $U_w := \mathrm{UP}(P \cup \{w\})$ cover the same set of vertices and edges.

First observe that any up-down path that has both endpoints in P is in both sets, $U_{w'}$ and U_w.

Let $Q_{w+w'}$ be the edge-path from w to w'. Now let $Q \in U_{w'}$ use the edge $w'r'$, then $Q_{w+w'}Q$ is a path in U_w that covers the set of vertices and edges of Q. Let $Q \in U_{w'}$ be an up-down path with $Q_{w+w'} \subseteq Q$. We observe that $Q \setminus Q_{w+w'}$ is in U_w and that $Q_{w+w'}Q_{w',p'}$ is in U_w, where $Q_{w',p'}$ is a directed path from w' to p' containing $w'r'$. These two paths cover the set of vertices and edges of Q For other up-down paths $Q \in U_{w'}$, add $Q_{w+w'}$ and remove doubled parts. Since $Q_{w+w'}$ is already covered, this shows that all vertices and edges that are covered with up-down paths in $U_{w'}$ are covered with up-down paths in U_w. □

Lemma 10. *The tree $T_{(P,\alpha)}$ can be constructed in $\mathcal{O}({\ell_v}^2 \cdot m')$ time, where ℓ_v is the number of visible reticulations and m' the number of edges in the original blob.*

Proof. For every pair of vertices in P_α, we first compute the set of edges and vertices on up-down paths between them in $\mathcal{O}(m')$ time. There are $\mathcal{O}({\ell_v}^2)$ many pairs, so, finding $V_{(P,\alpha)}$ and $E_{(P,\alpha)}$ takes $\mathcal{O}({\ell_v}^2 \cdot m')$ time. The remaining operations can all be done by a constant number of traversals of the blob, each taking $\mathcal{O}(m')$ time, thus proving the lemma. □

Lemma 11. *Let $\mathcal{N}$ be a semi-directed network with a lowest blob B with a designated vertex v. Let x be a leaf neighbor of B. Exactly one of the following is true.*

- *x has an edge-path to v.*
- *x has an edge-path to a reticulation r. The reticulation r is visible with respect to x.*

Proof. By Condition (III) of Theorem 2 in [18], there can be no edge-path between two reticulations in a semi-directed network. Hence, there is at most one edge-path from x to a reticulation in $\mathcal{N}$.

Suppose first that x has no such edge-path. We claim that x has an edge-path to v. If not, then every semi-directed path from v to x contains a reticulation. Such a path must exist since v is above every vertex in B by definition. But then x has an edge-path to a reticulation, which is a contradiction.

Now suppose that x has one edge-path to a reticulation in $\mathcal{N}$, and call this reticulation r. Take any vertex u above r. We claim that every semi-directed path from u to x contains r. Suppose, for a contradiction, that this is not the case. Then we split into two cases.

Case 1: There exists a semi-directed path from u to x that is not an edge-path, which does not contain r. Find the last reticulation r' on this path in the traversal order. The leaf x has an edge-path both to r and to r', contradicting our assumption.

Case 2: There exists an edge-path from u to x. Since there is also an edge-path from x to r, we can combine a subset of the edges on these two edge-paths to obtain an edge-path from u to r. By definition, there is a semi-directed path from u to r which contains at least one reticulation edge. But this gives a semi-directed cycle, a cycle of directed edges and undirected edges which can be oriented to obtain a directed cycle, containing u and r. This is not possible by Condition (II) of Theorem 2 in [18], which gives our required contradiction.

So in particular, for any leaf y above r, all up-down paths between x and y must contain r. Thus r is visible with respect to x. As v is above r, there also cannot be an edge-path between x and v. □

Theorem 3. *Algorithm 2 solves instances $(\mathcal{N}, X, f, \omega, k, D)$ of* G-MapSPD *in $\mathcal{O}(2^{\ell_v}\ell_v^2 \cdot nmk^2)$ time, where ℓ_v is the visible vertex level, n is the number of vertices, and m is the number of edges in $\mathcal{N}$.*

Proof. By Lemma 3, we can preprocess the network such that there are no 2-blobs, and that there are thus $\mathcal{O}(n)$ blobs in total.

We now prove the correctness of Algorithm 2 with three claims. Let $\mathcal{I} = (\mathcal{N}, X, f, \omega, k, D)$ be an instance of G-MapSPD.

Claim 12. *If $\mathcal{N}$ is a tree, then Algorithm 2 correctly returns whether $\mathcal{I}$ is a* `yes`*-instance of* G-MapSPD *in Line 4 or Line 5.*

After Line 1, all subtrees in $\mathcal{N}$ are reduced to leaves. Thus, if $\mathcal{N}$ is a tree, we return `yes` or `no` in Line 4 or 5. The correctness of Claim 12 follows with Lemma 6.

Now let $\mathcal{N}$ be a network which has at least one reticulation and on which we can not apply Rules 1 to 3. Let B denote the lowest blob found in Line 6—and to which the algorithm is applied to until a return—with designated vertex v, and let B_X denote the leaf neighbors of B.

Claim 13. *There is a solution σ of $\mathcal{I}$ which only spends budget on (a subset of) B_X if and only if Algorithm 2 returns* `yes` *in Line 11.*

Finally, let Algorithm 2 not return in Line 4, 5, or 11. Let $\mathcal{I}' = (\mathcal{N}', X', f', \omega', k, D)$ be the instance given into the recursion in Line 22.

Claim 14. *$\mathcal{I}$ is a* `yes`*-instance of* G-MapSPD *if and only if $\mathcal{I}'$ is a* `yes`*-instance of* G-MapSPD*.*

We observe that these three claims ar sufficient to prove the correctness of Algorithm 2. We continue proving Claim 13 and Claim 14, afterward.

Proof of Claim 13. Let σ be a solution of $\mathcal{I}$ which only spends a budget of k on $X_\sigma \subseteq B_X$. Let α be 1 if v has an edge-path to a leaf in X_σ, and let α be 0, otherwise. Let P be the set of reticulations which have an edge-path to a vertex in X_σ. Since each taxon has an edge-path to at most one reticulation (Lemma 11), the set P is well-defined. We show that in tree $T := T_{(P,\alpha)}$, the diversity of X_σ is $\mathrm{PD}_{\mathcal{N}}(X_\sigma) - H_{(P,\alpha)} - \omega(E_{(P,\alpha)}) + m_{(P,\alpha)} \cdot M$. We implicitly use that P_1 is well-defined (Lemma 9). In the light of Lemma 6, this is sufficient to prove that in Line 10 a `yes` is triggered. We consider the steps of creating T. In Step 1, we remove vertices in $\mathrm{hidden}(V_{(P,\alpha)})$, which is equivalent to setting σ to 0 in every leaf in $\mathrm{hidden}(V_{(P,\alpha)})$. We thus reduce the diversity by $H_{(P,\alpha)}$. Recall that $E_{(P,\alpha)}$ is the set of edges on up-down paths between vertices in P. As we can extend these paths on both ends to leaves in X_σ, the weight of $E_{(P,\alpha)}$ is considered in $\mathrm{PD}_{\mathcal{N}}(X_\sigma)$, but compressed in Step 2. In Step 4, we add $m_{(P,\alpha)}$ edges of length M as connection between $\mathcal{C}_r$ and $v_{P,\alpha}$, which thus are all on paths between specific leaves in X_σ. In Step 1 and 5, vertices are removed that are not on up-down paths between vertices in X_σ. In consequence, `yes` is returned in Line 11. This completes this direction.

Conversely, let `yes` be returned in Line 11. Thus, $f(u_{(P,\alpha)}, k) + H_{(P,\alpha)} + \omega(E_{(P,\alpha)}) - m_{(P,\alpha)} \cdot M \geq D$, for some $P \subseteq R_V$ and some $\alpha \in \{0, 1\}$. With Lemma 6, we conclude that there is a function σ that spends a budget of k on leaves of $T_{(P,\alpha)}$ such that $\mathrm{GPD}_{T_{(P,\alpha)}}(\sigma) + H_{(P,\alpha)} + \omega(E_{(P,\alpha)}) - m_{(P,\alpha)} \cdot M \geq D$. With similar arguments as before, we conclude that σ is a solution for $\mathcal{N}$. ◇

Proof of Claim 14. We can show this claim is relatively analogous to Claim 13 and thus only point out differences.

In Step 3 of creating $T_{(P,\alpha)}$, we add two leaf neighbors of v to ensure that there is always an up-down path to v, since we want to spend a positive budget on taxa in $X \setminus B_X$. These are two leaves to create the possibility of spending a budget of $h \in \{0, 1\}$ on B_X. Consequently, we have to allow for a budget of $h + 2$ in $u_{(P,\alpha)}$.

Instead of returning instantly, we search for each h the P, such that the diversity contribution in B is maximized. Which $P \subseteq R_V$ was chosen best for a specific h is irrelevant in the end. ◇

To show the running time, we recall that finding a lowest blob can be done in $\mathcal{O}(m)$ time, Lemma 8, and Rule 0 can be applied exhaustively in $\mathcal{O}(m)$ time, Lemma 3. Note that Rule 0 will only need to be called once before any loop or recursion, since the remainder of the algorithm creates no new 2-blobs.

We observe that the for loop in Line 8 is called $\mathcal{O}(2^{\ell_v})$ times, because $|R_v| \in \mathcal{O}(\ell_v)$. The trees $T_{(P,\alpha)}$ are computed in $\mathcal{O}({\ell_v}^2 \cdot m')$ time each, by Lemma 10, where m' is the number of edges in the blob used to create $T_{(P,\alpha)}$. Applying Rules 1 to 3 is done in $\mathcal{O}(nk^2)$ time, Lemma 6.

Each time Algorithm 2 is called, at least one blob is reduced, such that $\mathcal{O}(n)$ are necessary. Thus, the overall running time is $\mathcal{O}(2^{\ell_v}\ell_v^2 \cdot nmk^2)$. $\square$