A multivariate complexity analysis of the Generalized Noah's Ark Problem

Komusiewicz, Christian; Schestag, Jannik

DOI
10.1016/j.dam.2025.11.037

Publication date
2026

Document Version
Final published version

Published in
Discrete Applied Mathematics

Citation (APA)
Komusiewicz, C., & Schestag, J. (2026). A multivariate complexity analysis of the Generalized Noah's Ark Problem. *Discrete Applied Mathematics*, *382*, 137-154. https://doi.org/10.1016/j.dam.2025.11.037

Discrete Applied Mathematics 382 (2026) 137–154

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

A multivariate complexity analysis of the Generalized Noah's Ark Problem☆

Christian Komusiewicz [a], Jannik Schestag [b,*]

[a] Friedrich Schiller University Jena, Ernst-Abbe-Platz 2, Jena, 00743, Germany
[b] Delft University of Technology, Postbus 5, Delft, 2600 AA, The Netherlands

ARTICLE INFO

Article history:
Received 30 September 2023
Received in revised form 25 August 2025
Accepted 14 November 2025

Keywords:
FPT
Parameterized complexity
Phylogenetic diversity
Multiple-Choice Knapsack
Noah's Ark Problem

ABSTRACT

In the Generalized Noah's Ark Problem, one is given a phylogenetic tree on a set of species X and a set of conservation projects for each species. Each project comes with a cost and raises the survival probability of the corresponding species. The aim is to select a conservation project for each species such that the total cost of the selected projects does not exceed some given threshold and the expected phylogenetic diversity is as large as possible. We study the complexity of Generalized Noah's Ark Problem and some of its special cases with respect to several parameters related to the input structure, such as the number of different costs, the number of different survival probabilities, or the number of species, $|X|$.

1. Introduction

The preservation of biological diversity is one of humanity's most critical challenges. To help systematically address this challenge, it is useful to quantify or predict the effect of interventions. Here, two questions arise: how to measure the biological diversity of ecosystems and how to model the effect of certain actions on the biological diversity of an ecosystem under consideration.

A popular way to measure the biological diversity of an ecosystem, introduced by Faith [9], is to consider the *phylogenetic diversity* of the species present in that system. Here, the phylogenetic diversity is the sum of evolutionary distances between the species when their evolution is modeled by an evolutionary (phylogenetic) tree. The tree then not only gives the phylogenetic diversity of the whole species set but also allows us to infer the phylogenetic diversity of any subset of these species that would remain after some currently present species become extinct. Now, to model the effect of certain actions, a first simple model is that one can afford to protect k species and that all other species go extinct. Maximizing phylogenetic diversity under this model can be solved in polynomial time with a simple greedy algorithm [23,27]. Later, more realistic models were introduced. One step was to model that protecting some species may be more costly than protecting others [24]. Subsequent approaches also allowed us to model uncertainty as follows: performing an action to protect some species does not guarantee the survival of that species but only raises the survival probability [28]. In this model, one now aims to maximize the *expected* phylogenetic diversity. Finally, one may also consider the even more realistic case when, for each species, one may choose from a set of different actions or even from

☆ We thank the German Academic Exchange Service (DAAD), project no. 57556279, for the funding. Partially funded by the Dutch Research Council (NWO), project "Optimization for and with Machine Learning (OPTIMAL)" OCENW.GROOT.2019.015.

* Corresponding author.
E-mail address: j.t.schestag@tudelft.nl (J. Schestag).

https://doi.org/10.1016/j.dam.2025.11.037

combinations of different actions. Each choice is then associated with a cost and with a resulting survival probability. This model was proposed by Pardi [22] and Billionnet [2,3] as Generalized Noah's Ark Problem (GNAP).

Introducing cost differences for species protection makes the problem of maximizing phylogenetic diversity NP-hard [24] and thus all of the even richer models are NP-hard as well. Apart from the NP-hardness and several pseudopolynomial-time algorithms, there is no work that systematically studies which structural properties of the input make the problems tractable. This work aims to fill this gap. More precisely, we study how different parameters related to the input structure influence the algorithmic complexity of GNAP and several of its special cases.

In a nutshell, we show the following. First, GNAP can be solved efficiently when the number of different project costs and survival probabilities is small. Second, while a constant number $|X|$ of species X leads to polynomial-time solvability, algorithms with running time $f(|X|)\cdot|\mathcal{I}|^{\mathcal{O}(1)}$ are unlikely to exist. Finally, restricted cases where, for example the input tree has height 1 or there is exactly one action for each species that guarantees its survival are much easier than the general problem. Some of our results are obtained by observing a close relation to the Multiple-Choice Knapsack problem, for which we also provide a number of new complexity results.

This work is organized as follows. In Section 2, we give formal definitions of the problem and the considered parameters and an overview of our results. In Section 3, we study Multiple-Choice Knapsack. In Section 4, we consider GNAP, first the general case and then the case where the phylogenetic tree has height 1. In Section 5, we consider the special case of GNAP when there are at most two possible actions for each species.

2. Basic definitions and overview of the results

For an integer n, by $[n]$ we denote the set $\{1,\dots,n\}$, and by $[n]_0$ we denote the set $\{0,1,\dots,n\}$. We generalize functions $f : A \to B$ to handle subsets $A' \subseteq A$ by $f(A') := \bigcup_{a\in A'} f(a)$ if B is a family of sets and $f_\Sigma(A') := \sum_{a\in A'} f(a)$ if $B \subseteq \mathbb{R}$. A *partition of* N is a family of pairwise disjoint sets $\{N_1,\dots,N_m\}$ such that $\bigcup_{i=1}^{m} N_i = N$.

2.1. Parameterized complexity

To assess the influence of structural properties of the input on the problem complexity, we study the problems in the framework of parameterized complexity. For a detailed introduction to parameterized complexity refer to the standard monographs [5,7]. We only recall the most important definitions: Instances (I,k) of a parameterized problem consist of a classical problem instance I and a parameter k. A parameterized problem with parameter k is *fixed-parameter tractable* (FPT) if every instance $(\mathcal{I},k)$ can be solved in $f(k)\cdot|\mathcal{I}|^{\mathcal{O}(1)}$ time. If every instance can be solved with an algorithm such that f is a polynomial, then we say the parameterized problem is *polynomial fixed-parameter tractable* (PFPT). A parameterized problem is *slicewise-polynomial* (XP) if every instance can be solved in $|\mathcal{I}|^{g(k)}$ time. Some problems in XP are assumed to have no FPT-algorithm. In particular, there is the complexity class W[1] for which FPT $\subseteq$ W[1] $\subseteq$ XP is known and it is widely assumed that problems that are W[1]-hard have no FPT-algorithm. To prove that a problem is W[1]-hard, one uses parameterized reductions. A *parameterized reduction* from a parameterized problem L to a parameterized problem L' is an algorithm which, given an instance $(\mathcal{I},k)$ of L, constructs in $f(k)\cdot|\mathcal{I}|^{\mathcal{O}(1)}$ time an equivalent instance (I',k') of L' such that $k' \le g(k)$ for some function g. A parameterized problem L is called para-NP-*hard* if there exists some constant value t such that L restricted to instances with $k \le t$ is NP-hard.

In our running time analyses, we assume a unit-cost RAM model where arithmetic operations have constant running time.

2.2. Trees

A *directed graph* G is a tuple (V,E), where V is called the *set of vertices* of G and E the *set of edges* of G, respectively. We denote an edge directed from u to v by (u,v). The *degree of a vertex* v is the number of edges that are incident with v. A *tree* T *with root* r is a directed acyclic graph with a distinguished vertex r such that each vertex of T can be reached from r via exactly one path. A vertex v of a tree T is a *leaf* when the degree of v is one. The *height of a rooted tree* T, denoted height_T, is the maximal distance of its root r to any leaf. For an edge (u,v), we call u *the parent of* v and v *a child of* u. For a vertex v with parent u, the *subtree* T_v *rooted at* v is the connected component containing v in $T-(u,v)$. When v is the root of T, we define $T_v := T$. For a vertex v in a tree T, the *offspring of* v, denoted $\text{off}(v)$, is the set of leaves in T_v. For any vertex v, we assume an arbitrary but fixed ordering of its children. A *star* is a graph $G=(V,E)$ with a vertex $c\in V$ and $E=\{(c,v)\mid v\in V\setminus\{c\}\}$.

2.3. Phylogenetic diversity

A phylogenetic X-*tree* $\mathcal{T}=(V,E,\lambda)$ (in short, X-tree) is a tree T with root r, where $\lambda : E \to \mathbb{N}$ is an edge-weight function and X is the set of leaves of T. In biological applications, the internal vertices of T correspond to hypothetical ancestors of the leaves and $\lambda(e)$ describes the evolutionary distance between the endpoints of e. An X-tree $\mathcal{T}$ is *ultrametric* if there is an integer p such that the sum of the weights of the edges from the root r to x_i equals p for every leaf x_i.

Table 1
Complexity results for GNAP with unbounded number of projects per taxon.

Parameter	GNAP	Star-GNAP
$\|X\|$	W[1]-hard (Theorem 5), XP (Proposition 1(a))	W[1]-hard (Theorem 5), XP (Proposition 1(a))
B	XP (Proposition 1(c))	PFPT $\mathcal{O}(B \cdot \\|\mathcal{P}\\|)$ (Proposition 7)
C	para-NP-h (Theorem 7)	PFPT $\mathcal{O}(C \cdot \\|\mathcal{P}\\| \cdot \|X\|)$ (Proposition 7)
D	para-NP-h (Proposition 2(b))	para-NP-h (Proposition 2(b))
$\text{var}_c + \text{var}_w$	XP $\mathcal{O}(\|X\|^{2\cdot(\text{var}_c+\text{var}_w)+1})$ (Theorem 3)	FPT (Theorem 6)
var_c	para-NP-h (Theorem 7)	XP $\mathcal{O}(\|X\|^{\text{var}_c-1} \cdot \\|\mathcal{P}\\|)$ (Proposition 7)
var_w	para-NP-h (Proposition 2(a))	para-NP-h (Proposition 2(a))
val_λ	para-NP-h (Theorem 5)	para-NP-h (Theorem 5)
D + w-code	open	FPT $\mathcal{O}(D \cdot 2^{\text{w-code}} \cdot \\|\mathcal{P}\\|)$ (Proposition 7)
$B + \text{var}_w$	XP $\mathcal{O}(B \cdot \|X\|^{2\cdot\text{var}_w+1})$ (Theorem 4)	PFPT (Proposition 7)
$D + \text{var}_w$	para-NP-h (Proposition 2(b))	para-NP-h (Proposition 2(b))

For a taxon $x_i \in X$, a *project list* P_i is a tuple $(p_{i,1}, \dots, p_{i,\ell_i})$. Herein, each project $p_{i,j}$ represents an action that may raise the survival probability of x_i at a certain cost. Formally, each *project* is a tuple $(c_{i,j}, w_{i,j}) \in \mathbb{N}_0 \times \mathbb{Q} \cap [0, 1]$, where $c_{i,j}$ is the *cost* and $w_{i,j}$ the *survival probability* of $p_{i,j}$. As a project with higher cost will only be considered when the survival probability is higher, we assume the costs and survival probabilities to be ordered. That is, $c_{i,j} < c_{i,j+1}$ and $w_{i,j} < w_{i,j+1}$ for every project list P_i and every $j < \ell_i$. An *m-collection of project lists* $\mathcal{P}$ is a set of m project lists $\{P_1, \dots, P_m\}$. For a project set S, the *total cost* $\text{Cost}(S)$ *of* S is $\sum_{p_{i,j} \in S} c_{i,j}$.

For a given X-tree $\mathcal{T}$, the *expected phylogenetic diversity* $PD_\mathcal{T}(S)$ *of a set of projects* $S = \{p_{1,j_1}, \dots, p_{|X|,j_{|X|}}\}$ is given by

$$PD_\mathcal{T}(S) := \sum_{(u,v) \in E} \lambda(u, v) \cdot \left(1 - \prod_{x_i \in \text{off}(v)} (1 - w_{i,j_i})\right).$$

Herein, the term $\left(1 - \prod_{x_i \in \text{off}(v)}(1 - w_{i,j_i})\right)$ is the probability that at least one offspring of v (and with it the edge (u, v)) survives when S is implemented.

2.4. Problem definitions, parameters, and results overview

The most general version of the problem under consideration is defined as follows.

Generalized Noah's Ark Problem (GNAP)
Input: An X-tree $\mathcal{T} = (V, E, \lambda)$, an $|X|$-collection of project lists $\mathcal{P}$, and numbers $B \in \mathbb{N}_0, D \in \mathbb{Q}_{\geq 0}$.
Question: Is there a set of projects $S = \{p_{1,j_1}, \dots, p_{|X|,j_{|X|}}\}$, one from each project list of $\mathcal{P}$, such that $PD_\mathcal{T}(S) \geq D$ and $\text{Cost}(S) \leq B$?

Such a project set S is called *a solution for the instance* $\mathcal{I} = (\mathcal{T}, \mathcal{P}, B, D)$.

We consider the following special cases of GNAP:

- Star-GNAP is the special case of GNAP where $\mathcal{T}$ is restricted to have height 1.
- $a_i \xrightarrow{c_i} b_i$ NAP is the special case of GNAP where each project list P_i contains exactly two projects $(0, a_i)$ and (c_i, b_i). In other words, in an instance of $a_i \xrightarrow{c_i} b_i$ NAP we can decide for each taxon x_i whether we want to spend c_i to increase the survival probability of x_i from a_i to b_i. If the survival probability of all species is either 0 or some uniform value, then this is denoted by $0 \xrightarrow{c_i} b$ NAP. For $b = 1$ this corresponds to the choice between guaranteed extinction and guaranteed survival.
- Unit-Cost NAP is the special case of $a_i \xrightarrow{c_i} b_i$ NAP where every project with a positive survival probability has cost 1.

We study GNAP with respect to several parameters which we describe in the following. The results are summarized in Table 1 for the general problem and in Table 2 for the special case with two projects per taxon. The input of GNAP directly gives the natural parameters *number of taxa* $|X|$, *budget* B, and required *diversity* D. Closely related to B is the *maximum cost per project* $C = \max_{i,j} c_{i,j}$. We may assume that no projects have a cost that exceeds the budget, as we can delete them from the input and so $C \leq B$. We may further assume that $B \leq C \cdot |X|$, as otherwise we can compute in polynomial time whether the diversity of the most valuable projects of the taxa exceeds D and return yes, if it does and no, otherwise.

Further, we consider the *maximum number of projects per taxon* $L := \max_i |P_i|$. By definition, $L = 2$ in $a_i \xrightarrow{c_i} b_i$ NAP and in GNAP we have $L \leq C + 1$. We denote the *number of projects* by $\|\mathcal{P}\| = \sum_i |P_i|$. Clearly, $|X| \leq \|\mathcal{P}\|$, $L \leq \|\mathcal{P}\|$, and $\|\mathcal{P}\| \leq |X| \cdot L$. By var_c, we denote the *number of different costs*, that is, $\text{var}_c := |\{c_{i,j} : (c_{i,j}, w_{i,j}) \in P_i, P_i \in \mathcal{P}\}|$. We define the *number of different survival probabilities* var_w analogously. This so-called *number of numbers* parameterization

Table 2
Complexity results for $0 \xrightarrow{c_i} 1$ NAP, the special case where the survival probabilities are only 0 or 1, and UNIT-COST NAP the special case where each project has unit costs. The "—"-sign indicates parameters that are (partially) constant in the specific problem definition and thus are not interesting.

Parameter	$0 \xrightarrow{c_i} 1$ NAP	UNIT-COST NAP
$\lvert X\rvert$	FPT (Proposition 1(b))	FPT (Proposition 1(b))
B	PFPT $\mathcal{O}(B^2 \cdot n)$ [24]	XP (Proposition 1(c))
C	PFPT $\mathcal{O}(C^2 \cdot n^3)$ (Corollary 4)	—
D	PFPT $\mathcal{O}(D^2 \cdot n)$ (Proposition 8)	open
val_λ	PFPT $\mathcal{O}((\text{val}_\lambda)^2 \cdot n^3)$ (Corollary 5)	open
var_c	XP (Corollary 1)	—
var_w	—	XP (Corollary 2)

was introduced by Fellows et al. [10]; it is motivated by the idea that in many real-life instances this parameter may be small. In fact, for nonnegative numbers, the number of numbers is never larger than the maximum value which is used in pseudopolynomial-time algorithms. Also, we consider the *maximum encoding length for survival probabilities* w-code $= \max_{i,j}(\textit{binary length of } w_{i,j})$ and the *maximum edge weight* $\text{val}_\lambda = \max_{e\in E} \lambda(e)$. Observe that because the maximal survival probability of a taxon could be smaller than 1, one cannot assume that $\text{val}_\lambda \le D$.

2.5. Basic observations

We now observe some first complexity classifications that can be obtained from naive brute-force algorithms and from adaptions of a known NP-hardness reduction.

Proposition 1.

(a) GNAP *can be solved in* $\mathcal{O}(L^{|X|} \cdot n^2)$ *time.*
(b) $a_i \xrightarrow{c_i} b_i$ NAP *can be solved in* $\mathcal{O}(2^{|X|} \cdot n^2)$ *time.*
(c) GNAP *can be solved in* $\|P\|^B \cdot |\mathcal{I}|^{\mathcal{O}(1)}$ *time.*

Proof. (a) Consider all possibilities to select a project $p_{i,j_i} \in P_i$ for all taxa x_i. For each chosen set $S := \{p_{i,j_i} \mid i \in [|X|]\}$ compute $\text{Cost}(S)$ and $PD_\mathcal{T}(S)$ and return yes if $\text{Cost}(S) \le B$ and $PD_\mathcal{T}(S) \ge D$. If no considered set S gives a solution, then return no.

For each taxon, the number of different choices is at most L. Moreover, we can compute $\text{Cost}(S)$ and $PD_\mathcal{T}(S) \ge D$ in $\mathcal{O}(n^2)$ time. Hence, the algorithm has running time $\mathcal{O}(L^{|X|} \cdot n^2)$.

(b) This is a direct consequence of the previous with $L = 2$.

(c) A GNAP solution contains at most B projects with positive costs. Hence, a solution can be found by iterating over all size-B project sets, checking whether any of them gives a solution. □

Some hardness results can be obtained directly from a known reduction from the NP-hard KNAPSACK problem to $0 \xrightarrow{c_i} 1$ NAP [24]. In KNAPSACK one is given a set of items N, a cost-function $c : N \to \mathbb{N}$, a value-function $d : N \to \mathbb{N}$, and two integers B and D and asks whether there is an item set N' such that $c_\Sigma(N') \le B$ and $d_\Sigma(N') \ge D$.

Proposition 2.

(a) $0 \xrightarrow{c_i} 1$ NAP *is* NP*-hard, even if the tree* $\mathcal{T}$ *has height 1 [24].*
(b) $0 \xrightarrow{c_i} b$ NAP *is* NP*-hard, even if* $D = 1$*,* $b \in (0, 1]$ *is a constant, and the tree* $\mathcal{T}$ *has height 1.*
(c) $0 \xrightarrow{c_i} b_i$ NAP *is* NP*-hard, even if the tree* $\mathcal{T}$ *is ultrametric with* $\text{height}_\mathcal{T} = \text{val}_\lambda = 1$*, and* $D = 1$.

Proof. (a) Let $\mathcal{I} = (N, c, d, B, D)$ be an instance of KNAPSACK. Define $\mathcal{T} := (V, E, \lambda)$ to be an N-tree with $V := \{w\} \cup N$ and $E := \{(w, x_i) \mid x_i \in N\}$ and $\lambda(w, x_i) := d(x_i)$. For each leaf x_i, define a project list P_i that contains two projects $(0, 0)$ and $(c(x_i), 1)$.

Then, $\mathcal{I}' := (\mathcal{T}, \mathcal{P}, B' := B, D' := D)$ is a yes-instance of $0 \xrightarrow{c_i} 1$ NAP if and only if $\mathcal{I}$ is a yes-instance of KNAPSACK. This can be seen by observing that a set S is a solution for $\mathcal{I}$ if and only if the project set S' obtained by adding $p_{i,1} = (0, 0)$ if $x_i \notin S$ and $p_{i,2} = (c(x_i), 1)$ if $x_i \in S$ is a solution for $\mathcal{I}'$. Indeed this follows from

$$d_\Sigma(S) = \sum_{x_i \in S} d(x_i) = \sum_{x_i \in S} \lambda(w, x_i) = PD_\mathcal{T}(S')$$

Table 3
Complexity results for Multiple-Choice Knapsack.

Parameter	MCKP
m	W[1]-hard, XP (Theorem 2)
B	PFPT $\mathcal{O}(B \cdot \|N\|)$ [25]
C	PFPT $\mathcal{O}(C \cdot \|N\| \cdot m)$ (Proposition 3)
D	PFPT $\mathcal{O}(D \cdot \|N\|)$ [1]
L	para-NP-h [16]
var_c	XP $\mathcal{O}(m^{\text{var}_c - 1} \cdot \|N\|)$ (Proposition 4)
var_d	XP $\mathcal{O}(m^{\text{var}_d - 1} \cdot \|N\|)$ (Proposition 5)
$\text{var}_c + \text{var}_d$	FPT (Theorem 1)

and

$$c_\Sigma(S) = \sum_{x_i \in S} c(x_i) = \sum_{x_i \in S} c_{i,2} = \text{Cost}(S').$$

(b) In this reduction, one could also set $D' := 1$ and replace the survival probability of every project with positive cost to $1/D$. Thus, Star-GNAP is NP-hard even if $D = 1$ and $\text{var}_w = 2$.

(c) We may assume in the previous reduction that there is no project $p = (c, 1/D)$ of taxon x with $\lambda(\rho, x) > 1/D$ and $c < B$, as we have a trivial yes-instance, otherwise. Henceforth, we remove all taxa with project cost $c > B$ and for all remaining projects, we divide the survival probability by $\lambda(\rho, x)$, directly yielding the desired result. □

3. Multiple-choice knapsack

In this section, we consider Multiple-Choice Knapsack (MCKP), a classic variant of Knapsack in which the item set is divided into classes and from every class, exactly one item can be chosen. MCKP has been studied numerous times over the years [1,16,21,25]. MCKP is also special case of the Max d-Dimensional Knapsack problem, for which Gurski, Rehs, and Rethmann [12] provided a parameterized complexity analysis.

To define the problem, let c and d denote the cost and value functions, respectively. Accordingly, for an item a_i, we call $c(a_i)$ the *cost of* a_i and $d(a_i)$ the *value of* a_i. Recall that using our function notation for sets, this means that $c_\Sigma(A)$ and $d_\Sigma(A)$ denote the sum of the costs and values of the elements of A and $c(A)$ and $d(A)$ denote the set of costs and values assumed by some element of A.

Multiple-Choice Knapsack (MCKP)
Input: A set of items $N = \{a_1, \dots, a_n\}$, a partition $\{N_1, \dots, N_m\}$ of N, two functions $c, d : N \to \mathbb{N}$, and two integers B and D.
Question: Is there a set $S \subseteq N$ such that $|S \cap N_i| = 1$ for each $i \in [m]$, $c_\Sigma(S) \le B$, and $d_\Sigma(S) \ge D$?

A set S that fulfills these criteria is called a *solution* for the instance $\mathcal{I}$.

For MCKP, we consider parameters that are closely related to the parameters described for GNAP: The input directly gives the *number of classes* m, the *budget* B, and the desired *value* D. Closely related to B is the *maximum cost for an item* $C = \max_{a_j \in N} c(a_j)$. Since projects whose cost exceeds the budget can be removed from the input, we may assume $C \le B$. Further, we assume that $B \le C \cdot m$, as otherwise, one can return yes if the total value of the most valuable items per class exceeds D, and no otherwise. We also consider $\text{var}_c := |c(N)|$, the *number of different costs*, and $\text{var}_d := |d(N)|$, the *number of different values*. The size of the biggest class is denoted by L. If one class N_i contains two items a_p and a_q with the same cost and $d(a_p) \le d(a_q)$, the item a_p can be removed from the instance. Thus, we may assume that no class contains two items with the same cost, and so $L \le \text{var}_c$. Analogously, we may assume that no class contains two same-valued items and so $L \le \text{var}_w$. Table 3 lists the old and new complexity results for Multiple-Choice Knapsack.

3.1. Algorithms for multiple-choice knapsack

First, we provide some algorithms for MCKP. It is known that MCKP can be solved in $\mathcal{O}(B \cdot |N|)$ time [25], or in $\mathcal{O}(D \cdot |N|)$ time [1]. As we may assume that $C \cdot m \ge B$, we may also observe the following.

Proposition 3. MCKP *can be solved in* $\mathcal{O}(C \cdot |N| \cdot m)$ *time.*

Knapsack is FPT with respect to the number of different costs var_c [8]. This result is shown by a reduction to ILP-Feasibility with $f(\text{var}_c)$ variables. This approach cannot be adopted easily, as it has to be checked whether a solution contains exactly one item per class. In Propositions 4 and 5 we show that MCKP is XP with respect to the number of different costs and different values, respectively. Then, in Theorem 1, we show that MCKP is FPT with respect to the parameter $\text{var}_c + \text{var}_d$.

In the following, let $\mathcal{I} = (N, \{N_1, \ldots, N_m\}, c, d, B, D)$ be an instance of MCKP. We let $c_1, \ldots, c_{\mathrm{var}_c}$ and $d_1, \ldots, d_{\mathrm{var}_d}$ denote the set of different costs and the set of the different values in $\mathcal{I}$ where $c_i < c_{i+1}$ for each $i \in [\mathrm{var}_c - 1]$ and $d_j < d_{j+1}$ for each $j \in [\mathrm{var}_d - 1]$. In other words, c_i is the ith cheapest cost in $c(N)$ and d_j is the jth smallest value in $d(N)$. Since each class N_i contains at most one item with cost c_p and at most one item with value d_q, we have $|N_i| \le \mathrm{var}_c$ and $|N_i| \le \mathrm{var}_d$ for every $i \in [m]$.

Proposition 4. MCKP *can be solved in* $\mathcal{O}(m^{\mathrm{var}_c - 1} \cdot |N|)$ *time, where* var_c *is the number of different costs.*

Proof. We describe a dynamic programming algorithm with a table that has a dimension for all the var_c different costs, except for c_{var_c}. An entry $F[i, p_1, \ldots, p_{\mathrm{var}_c - 1}]$ of the table stores the maximal total value of a set S that contains exactly one item of each set of $N_1, \ldots, N_i$ and contains exactly p_j items with cost c_j for each $j \in [\mathrm{var}_c - 1]$. Subsequently, S contains exactly $p^{(i)}_{\mathrm{var}_c} := i - \sum_{j=1}^{\mathrm{var}_c - 1} p_j$ items with cost c_{var_c}. If $\sum_{j=1}^{\mathrm{var}_c - 1} p_j \ge i$, then store $F[i, p_1, \ldots, p_{\mathrm{var}_c - 1}] = -\infty$. In the rest of the proof, by $\vec{p}$ we denote $(p_1, \ldots, p_{\mathrm{var}_c - 1})$ and by $\vec{p}_{(j)+z}$ we denote $(p_1, \ldots, p_j + z, \ldots, p_{\mathrm{var}_c - 1})$ for an integer z. Let $\vec{0}$ denote the $(\mathrm{var}_c - 1)$-dimensional zero.

For the initialization of the values for $F[1, \vec{p}]$, only size-1 subsets of N_1 are considered. Thus, for every $a \in N_1$ with cost $c(a) = c_j < c_{\mathrm{var}_c}$, store $F[1, \vec{0}_{(j)+1}] = d(a)$. If N_1 contains an item a with cost $c(a) = c_{\mathrm{var}_c}$, then store $F[1, \vec{0}] = d(a)$. For all other $\vec{p}$, store $F[1, \vec{p}] = -\infty$. Once the entries for i classes have been computed, one may compute the entries for $i+1$ classes using the recurrence

$$F[i+1, \vec{p}] = \max_{a \in N_{i+1}} \begin{cases} F[i, \vec{p}_{(j)-1}] + d(a) & \text{if } c(a) = c_j < c_{\mathrm{var}_c} \text{ and } p_j \ge 1 \\ F[i, \vec{p}] + d(a) & \text{if } c(a) = c_{\mathrm{var}_c} \end{cases}. \tag{1}$$

Return yes if $F[m, \vec{p}] \ge D$ for $p^{(m)}_{\mathrm{var}_c} \cdot c_{\mathrm{var}_c} + \sum_{i=1}^{\mathrm{var}_c - 1} p_i \cdot c_i \le B$ and return no, otherwise.

Correctness. For given integers $i \in [m]$ and $\vec{p} \in [i]_0^{\mathrm{var}_c - 1}$, we define $\mathcal{S}^{(i)}_{\vec{p}}$ to be the family of i-sized sets $S \subseteq N$ that contain exactly one item of each of $N_1, \ldots, N_i$ and where p_ℓ is the number of items in S with cost c_ℓ for each $\ell \in [\mathrm{var}_c - 1]$.

For fixed $\vec{p} \in [i]_0^{\mathrm{var}_c - 1}$, we prove that $F[i, \vec{p}]$ stores the maximal value of a set $S \in \mathcal{S}^{(i)}_{\vec{p}}$ by an induction. This implies that the algorithm is correct. For $i = 1$, the claim is easy to verify from the initialization. Now, assume the claim is correct for a fixed $i \in [m-1]$. We first prove that if $F[i+1, \vec{p}] = q$, then there exists a set $S \in \mathcal{S}^{(i+1)}_{\vec{p}}$ with $d_\Sigma(S) = q$. Afterward, we prove that $F[i+1, \vec{p}] \ge d_\Sigma(S)$ for every set $S \in \mathcal{S}^{(i+1)}_{\vec{p}}$.

($\Rightarrow$) Let $F[i+1, \vec{p}] = q$. Let $a \in N_{i+1}$ with $c(a) = c_j$ be an item of N_{i+1} that maximizes the right side of Eq. (1) for $F[i+1, \vec{p}]$. In the first case, assume $c_j < c_{\mathrm{var}_c}$, and thus $F[i+1, \vec{p}] = q = F[i, \vec{p}_{(j)-1}] + d(a)$. By the induction hypothesis, there is a set $S \in S^{(i)}_{\vec{p}_{(j)-1}}$ such that $F[i+1, \vec{p}_{(j)-1}] = d_\Sigma(S) = q - d(a)$. Observe that $S' := S \cup \{a\} \in S^{(i+1)}_{\vec{p}}$. The value of S' is $d_\Sigma(S') = d_\Sigma(S) + d(a) = q$. The other case with $c(a) = c_{\mathrm{var}_c}$ is shown analogously.

($\Leftarrow$) Let $S \in \mathcal{S}^{(i+1)}_{\vec{p}}$ and let $a \in S \cap N_{i+1}$. In the first case, assume $c(a) = c_j < c_{\mathrm{var}_c}$. Observe that $S' := S \setminus \{a\} \in S^{(i)}_{\vec{p}_{(j)-1}}$. Consequently,

$$\begin{aligned} F[i+1, \vec{p}] &\ge F[i, \vec{p}_{(j)-1}] + d(a) & (2)\\ &= \max\{d_\Sigma(S) \mid S \in S^{(i)}_{\vec{p}_{(j)-1}}\} + d(a) & (3)\\ &\ge d_\Sigma(S') + d(a) = d_\Sigma(S). \end{aligned}$$

Herein, Inequality (2) is the definition of the recurrence in Eq. (1), and Eq. (3) follows by the induction hypothesis. The other case with $c(a) = c_{\mathrm{var}_c}$ is shown analogously.

Running time. First, we show how many options of $\vec{p}$s there are and then how many equations have to be computed for one of these options.

For $p_1, \ldots, p_{\mathrm{var}_c - 1}$ with $\sum_{j=1}^{\mathrm{var}_c - 1} p_j \ge m$ and each $i \in [m]$, the entry $F[i, \vec{p}]$ stores $-\infty$. Consequently, we consider a vector $\vec{p}$ with $p_j = m$ only if $p_\ell = 0$ for each $\ell \ne j$. Thus, we are only interested in $\vec{p} \in [m-1]_0^{\mathrm{var}_c - 1}$ or $\vec{p} = \vec{0}_{(j)+m}$ for each $j \in [\mathrm{var}_c - 1]$. So, there are $m^{\mathrm{var}_c - 1} + m \in \mathcal{O}(m^{\mathrm{var}_c - 1})$ options of $\vec{p}$.

For a fixed $\vec{p}$, each item $a \in N_i$ is considered exactly once in the computation of $F[i, \vec{p}]$. Thus, overall $\mathcal{O}(m^{\mathrm{var}_c - 1} \cdot |N|)$ time is needed to compute the table F. Additionally, we need $\mathcal{O}(\mathrm{var}_c \cdot |N|)$ time to check whether $F[m, p_1, \ldots, p_{\mathrm{var}_c - 1}] \ge D$ and $p^{(m)}_{\mathrm{var}_c} \cdot c_{\mathrm{var}_c} + \sum_{i=1}^{\mathrm{var}_c - 1} p_i \cdot c_i \le B$ for any $\vec{p}$. As we may assume $m^{\mathrm{var}_c - 1} > \mathrm{var}_c$, the running time of the entire algorithm is $\mathcal{O}(m^{\mathrm{var}_c - 1} \cdot |N|)$. $\square$

We now adapt of the dynamic programming algorithm described in the proof of Proposition 4. In this adaption, instead of storing the maximum value of a set of items with a given set of costs, we store the minimum cost a set of items with a given set of values. This shows that the problem is in XP with respect to the number of different values of the input items.

Proposition 5. MCKP *can be solved in* $\mathcal{O}(m^{\mathrm{var}_d - 1} \cdot |N|)$, *where* var_d *is the number of different values.*

Proof. We describe a dynamic programming algorithm with a table that has a dimension for all the var_d different values except for d_{var_d}. An entry $F[i, p_1, \ldots, p_{\mathrm{var}_d - 1}]$ of the table stores the minimal cost of a set S that contains exactly one item of each set of $N_1, \dots, N_i$ and contains exactly p_j items with value d_j for each $j \in [\mathrm{var}_d - 1]$. Subsequently, S contains exactly $p^{(i)}_{\mathrm{var}_d} := i - \sum_{j=1}^{\mathrm{var}_d - 1} p_j$ items with value d_{var_d}. If $\sum_{j=1}^{\mathrm{var}_d - 1} p_j \geq i$, then store $F[i, p_1, \dots, p_{\mathrm{var}_d - 1}] = B + 1$. In the rest of the proof, by $\vec{p}$ we denote $(p_1, \dots, p_{\mathrm{var}_d - 1})$ and by $\vec{p}_{(j)+z}$ we denote $(p_1, \dots, p_j + z, \dots, p_{\mathrm{var}_d - 1})$ for an integer z. Let $\vec{0}$ denote the $(\mathrm{var}_d - 1)$-dimensional zero.

For the initialization of the values for $F[1, \vec{p}]$, only size-1 subsets of N_1 are considered. Thus, for every $a \in N_1$ with value $d(a) = d_j < d_{\mathrm{var}_d}$, store $F[1, \vec{0}_{(j)+1}] = c(a)$. If N_1 contains an item a with value $d(a) = d_{\mathrm{var}_d}$, then store $F[1, \vec{0}] = c(a)$. For all other $\vec{p}$ store $F[1, \vec{p}] = B + 1$. Once the entries for i classes have been computed, one may compute the entries for $i + 1$ classes using the recurrence

$$F[i+1, \vec{p}] = \max_{a \in N_{i+1}} \begin{cases} F[i, \vec{p}_{(j)-1}] + c(a) & \text{if } d(a) = d_j < d_{\mathrm{var}_d} \text{ and } p_j \geq 1 \\ F[i, \vec{p}] + c(a) & \text{if } d(a) = d_{\mathrm{var}_d} \end{cases} . \tag{4}$$

Return yes if $F[m, \vec{p}] \leq B$ for $p^{(m)}_{\mathrm{var}_d} \cdot d_{\mathrm{var}_d} + \sum_{i=1}^{\mathrm{var}_d - 1} p_i \cdot d_i \geq D$ and return no, otherwise.

The correctness and running time proof are completely analogous to the proof of Proposition 4. □

By Propositions 4 and 5, MCKP is XP with respect to var_c and var_d, respectively. In the following, we reduce instances of MCKP to instances of ILP-Feasibility with at most $2^{\mathrm{var}_c + \mathrm{var}_d} \cdot \mathrm{var}_c$ variables. Since ILP-Feasibility instances with n variables and input length s can be solved using $s \cdot n^{2.5n+o(n)}$ arithmetic operations [11,19], this shows that MCKP is fixed-parameter tractable with respect to the combined parameter $\mathrm{var}_c + \mathrm{var}_d$.

Theorem 1. *For each instance of* MCKP, *an equivalent instance of* ILP-Feasibility *with* $\mathcal{O}(2^{\mathrm{var}_c + \mathrm{var}_d} \cdot \mathrm{var}_c)$ *variables can be computed in polynomial time. Thus,* MCKP *is* FPT *with respect to* $\mathrm{var}_c + \mathrm{var}_d$.

Proof. *Reduction.* We may assume that a class N_i does not contain two items of the same cost or the same value and that for items $a, b \in N_i$ with $c(a) < c(b)$ we have $d(a) < d(b)$. Thus, each class N_i is fully described by the set of costs $c(N_i)$ of the items in N_i and the set of values $d(N_i)$ of the items in N_i. In the following, we call $T = (C, Q)$ a *type*, for sets $C \subseteq c(N)$, $Q \subseteq d(N)$ with $|C| = |Q|$. Let $\mathcal{T}$ be the family of all types. We say that the *class N_i is of type $T = (c(N_i), d(N_i))$*. For each $T \in \mathcal{T}$, let m_T be the number of classes of type T. Clearly, $\sum_{T \in \mathcal{T}} m_T = m$.

Now, for each class N_j of type $T = (C, Q)$, the cost $c(a)$ of an item $a \in N_j$ directly determines its value $d(a)$. More precisely, if $c(a)$ is the ℓth cheapest cost in C, then the value of a is the ℓth smallest value in Q. Accordingly, if c_i is the ℓth smallest cost in C, we let $d_{T,i}$ denote the ℓth smallest value in Q. For all $i \in [\mathrm{var}_c]$ where $c_i \notin C$, we define $d_{T,i} := -\sum_{i=1}^{m} \max d(N_i)$.

We now define the following instance of ILP-Feasibility that is equivalent to $\mathcal{I}$. The variable $x_{T,i}$ expresses the number of items with cost c_i that are chosen in a class of type T.

$$\sum_{T \in \mathcal{T}_C} \sum_{i=1}^{\mathrm{var}_c} x_{T,i} \cdot c_i \leq B \tag{5}$$

$$\sum_{T \in \mathcal{T}_C} \sum_{i=1}^{\mathrm{var}_c} x_{T,i} \cdot d_{T,i} \geq D \tag{6}$$

$$\sum_{i=1}^{\mathrm{var}_c} x_{T,i} = m_T \qquad \forall T \in \mathcal{T} \tag{7}$$

$$x_{T,i} \geq 0 \qquad \forall T \in \mathcal{T}, i \in [\mathrm{var}_c] \tag{8}$$

Correctness. Observe that if $c_i \notin C$, then Inequality (6) would not be fulfilled if $x_{T,i} > 0$ because we defined $d_{T,i}$ to be $-\sum_{i=1}^{m} \max d(N_i)$. Consequently, $x_{T,i} = 0$ if $c_i \notin C$ for each type $T = (C, Q) \in \mathcal{T}$ and $i \in [\mathrm{var}_c]$. Inequality (5) ensures that the total cost is at most B. Inequality (6) ensures that the total value is at least D. Eq. (7) ensures that exactly m_T elements are picked from the classes of type T for each $T \in \mathcal{T}$. Finally, observe that the constructed instance has $\mathcal{O}(2^{\mathrm{var}_c + \mathrm{var}_d} \cdot \mathrm{var}_c)$ variables because $\mathcal{T} \subseteq 2^{c(N)} \times 2^{d(N)}$ which gives $\mathcal{O}(2^{\mathrm{var}_c + \mathrm{var}_d} \cdot \mathrm{var}_c)$ different options for the variables $x_{T,i}$. □

3.2. Hardness with respect to the number of classes

Second, we contrast the algorithms by the following hardness results. There is a reduction from Knapsack to MCKP in which each item in the instance of Knapsack is added to a unique class with a further, new item that has no costs and no value [16]. This reduction constructs an instance with many classes. In the following, we prove that MCKP is W[1]-hard with respect to the number of classes m, even if $B = D$ and $c(a) = d(a)$ for each $a \in N$. This special case of MCKP is called Multiple-Choice Subset Sum [16].

To show the hardness, we reduce from Subset Sum with multi-set input parameterized by k.

SUBSET SUM
Input: A multi-set of integers $Z = \{z_1, \dots, z_n\}$ and integers Q and k.
Question: Is there a multi-set $S \subseteq Z$ such that $\sum_{s \in S} = Q$ and $|S| = k$?

In parameterized complexity, the problem is often defined for set inputs. The original W[1]-hardness proof for SUBSET SUM relies on a reduction from the PERFECT CODE problem [6, Lemma 4.4]. It is easy to observe that this reduction also works if every constructed integer is added k times to Z. We will not repeat the details of the reduction but give a brief intuition. In the reduction, the target number Q has a value of one at each digit. Now, adding k copies of a number to Z in the construction maintains correctness because including any number twice in the solution produces carries in the summation which destroys the property that every digit has a value of 1.

Proposition 6 (*[6]*). SUBSET SUM *is* W[1]-*hard with respect to k, even when every integer in Z has multiplicity at least k in Z.*

Theorem 2. MCKP *is* XP *and* W[1]-*hard with respect to the number of classes m.*

Proof. The XP-algorithm follows from the fact that we may simply try all $|N|^m$ possibilities to select a set S of items and compute the value and cost for each of them.

Reduction. We reduce from SUBSET SUM with multi-set inputs Z where every integer z occurs k times. Let $\mathcal{I} = (Z, Q, k)$ be such an instance of SUBSET SUM and assume, without loss of generality, that $Z = \{z_{i,j} \mid i \in n, j \in [k]\}$ such that $z_{i,1} = z_{i,2} = \dots = z_{i,k}$ for all $i \in [n]$.

We define $N_j := \{a_{i,j} \mid i \in [n]\}$ for each $j \in [k]$ and set $c(a_{i,j}) = d(a_{i,j}) = z_{i,j}$ for each element $a_{i,j}$. Then $\mathcal{I}'$ is $(N, \{N_1, \dots, N_k\}, Q, Q)$ where N is the union of the sets N_j, $j \in [k]$. Observe that the reduction can be performed in polynomial time and that $m = k$, so the reduction is indeed a parameterized reduction from SUBSET SUM with parameter k to MCKP with parameter m.

Correctness. We show that $\mathcal{I}$ is a yes-instance if and only if $\mathcal{I}$' is a yes-instance.

($\Rightarrow$) Let $S = \{z_{i_1}, \dots, z_{i_k}\}$ be a solution for $\mathcal{I}$. By construction, each set N_j contains one element a_{ℓ_j} such that $c(a_{\ell_j}) = d(a_{\ell_j}) = z_{i_j}$. Then, the set $S' := \{a_{\ell_1}, a_{\ell_2}, \dots, a_{\ell_k}\}$ is a solution for $\mathcal{I}'$ since

$$\sum_{j \in [k]} c(a_{\ell_j}) = \sum_{j \in [k]} d(a_{\ell_j}) = \sum_{j \in [k]} z_{i_j} = Q.$$

($\Leftarrow$) Conversely, let $S' := \{a_{\ell_1}, a_{\ell_2}, \dots, a_{\ell_k}\}$ be a solution for $\mathcal{I}'$. By construction, for each $j \in [k]$, there is a number $z_{i_j} \in Z$ such that $c(a_{\ell_j}) = z_{i_j}$. Then, $S := \{z_{i_j} \mid j \in [k]\}$ is a solution for $\mathcal{I}$ since

$$Q = \sum_{j \in [k]} c(a_{\ell_j}) = \sum_{j \in [k]} z_{i_j}. \quad \square$$

4. The generalized Noah's Ark Problem

We now come back to the maximization of phylogenetic diversity, starting with the most general problem variant, GENERALIZED NOAH'S ARK PROBLEM (GNAP). Subsequently, we turn to STAR-GNAP, making use of the tractability and hardness results for MCKP established above.

4.1. Algorithms for the generalized Noah's Ark Problem

In Theorem 3, we now show that GNAP can be solved in polynomial time when the number of different project costs and the number of different survival probabilities is constant. In the following, let $\mathcal{I} = (\mathcal{T}, \mathcal{P}, B, D)$ be an instance of GNAP, and let $\mathcal{C} := \{c_1, \dots, c_{\text{var}_c}\}$ and $\mathcal{W} := \{w_1, \dots, w_{\text{var}_w}\}$ denote the sets of different costs and different survival probabilities in $\mathcal{I}$, respectively. Without loss of generality, assume $c_i < c_{i+1}$ for each $i \in [\text{var}_c - 1]$ and likewise assume $w_j < w_{j+1}$ for each $j \in [\text{var}_w - 1]$. In other words, c_i is the ith cheapest cost in $\mathcal{C}$, and w_j is the jth smallest survival probability in $\mathcal{W}$. Recall, that we assume that there is at most one item with cost c_p and at most one item with survival probability w_q in every project list P_i, for each $p \in [\text{var}_c]$ and $q \in [\text{var}_w]$. For the rest of the section, by $\vec{a}$ and $\vec{b}$ we denote $(a_1, \dots, a_{\text{var}_c - 1})$ and $(b_1, \dots, b_{\text{var}_w - 1})$, respectively. Further, we let $\vec{p}_{(j)+z}$ denote the vector $\vec{p}$ in which, at position i, the value z is added, and we let $\vec{0}$ denote the $(\text{var}_c - 1)$-dimensional zero.

Theorem 3. GNAP *can be solved in* $\mathcal{O}\left(|X|^{2(\text{var}_c + \text{var}_w - 1)} \cdot (\text{var}_c + \text{var}_w)\right)$ *time.*

Proof. We describe a dynamic programming algorithm with two tables, F and G, that have a dimension for all the var_c different costs, except for c_{var_c} and all the var_w different survival probabilities, except for $\text{var}_w - 1$. Recall that for a vertex v with t children, T_v is the subtree rooted at v and the offspring off(v) of v are the leaves in T_v. For a vertex v with children $w_1, \dots, w_t$ where w_i denotes the ith child, the *i-partial subtree $T_{v,i}$ rooted at v* for $i \in [t]$ is the connected component containing v in $T_v - \{(v, w_{i+1}), \dots, (v, w_t)\}$. For a vertex $v \in V$ and given vectors $\vec{a}$ and $\vec{b}$, we define $\mathcal{S}^{(v)}_{\vec{a},\vec{b}}$ to be the family of sets of projects S such that

1. S contains exactly one project of P_i for each $x_i \in \mathrm{off}(v)$,
2. S contains exactly a_k projects of cost c_k for each $k \in [\mathrm{var}_c - 1]$, and
3. S contains exactly b_ℓ projects of survival probability w_ℓ for each $\ell \in [\mathrm{var}_w - 1]$.

For a vertex $v \in V$ with children $u_1, \dots, u_t$, given vectors $\vec{a}$, and $\vec{b}$ and a given integer $i \in [t]$ we define $\mathcal{S}^{(v,i)}_{\vec{a},\vec{b}}$ analogously, just that exactly one project of P_i is chosen for each $x_i \in \mathrm{off}(u_1) \cup \dots \cup \mathrm{off}(u_i)$.

It follows that we can compute how many projects with cost c_{var_c} and survival probability w_{var_w} a set $S \in \mathcal{S}^{(v)}_{\vec{a},\vec{b}}$ contains. There are $a^{(v)}_{\mathrm{var}_c} := |\mathrm{off}(v)| - \sum_{j=1}^{\mathrm{var}_c - 1} a_j$ projects with cost c_{var_c} and $b^{(v)}_{\mathrm{var}_w} := |\mathrm{off}(v)| - \sum_{j=1}^{\mathrm{var}_w - 1} b_j$ projects with survival probability w_{var_w}. The entries $F[v, \vec{a}, \vec{b}]$ and $G[v, i, \vec{a}, \vec{b}]$ store the maximum expected phylogenetic diversity of the tree T_v for $S \in \mathcal{S}^{(v)}_{\vec{a},\vec{b}}$ and $T_{v,i}$ for $S \in \mathcal{S}^{(v,i)}_{\vec{a},\vec{b}}$, respectively. We further define the total survival probability to be $w(b_{\mathrm{var}_w}, \vec{b}) := 1 - (1 - w_{\mathrm{var}_w})^{b_{\mathrm{var}_w}} \cdot \prod_{i=1}^{\mathrm{var}_w - 1} (1 - w_i)^{b_i}$, when b_{var_w} and $\vec{b}$ describe the number of chosen single survival probabilities.

Fix a taxon x_i with project list P_i. As we want to select exactly one project of P_i, the project is clearly defined by $\vec{a}$ and $\vec{b}$. So, we store $F[x_i, \vec{a}, \vec{b}] = 0$, if P_i contains a project $p = (c_k, w_\ell)$ such that

1. ($k < \mathrm{var}_c$ and $\vec{a} = \vec{0}_{(k)+1}$ or $k = \mathrm{var}_c$ and $\vec{a} = \vec{0}$), and
2. ($\ell < \mathrm{var}_w$ and $\vec{b} = \vec{0}_{(\ell)+1}$ or $\ell = \mathrm{var}_w$ and $\vec{b} = \vec{0}$).

Otherwise, store $F[x_i, \vec{a}, \vec{b}] = -\infty$.

Let v be an internal vertex with children $u_1, \dots, u_t$, we define

$$G[v, 1, \vec{a}, \vec{b}] = F[u_1, \vec{a}, \vec{b}] + \lambda(v, u_1) \cdot w\left(b^{(u_1)}_{\mathrm{var}_w}, \vec{b}\right) \tag{9}$$

and to compute further values of G, we can use the recurrence

$$G[v, i+1, \vec{a}, \vec{b}] = \max_{\substack{\vec{0} \le \vec{a}' \le \vec{a} \\ \vec{0} \le \vec{b}' \le \vec{b}}} \left\{ \begin{array}{l} G[v, i, \vec{a} - \vec{a}', \vec{b} - \vec{b}'] + F[u_{i+1}, \vec{a}', \vec{b}'] \\ +\lambda(v, u_{i+1}) \cdot w\left(b^{(u_{i+1})}_{\mathrm{var}_w}, \vec{b}'\right) \end{array} \right. . \tag{10}$$

Herein, we write $\vec{p} \le \vec{q}$ if $\vec{p}$ and $\vec{q}$ have the same dimension d and $p_i \le q_i$ for every $i \in [d]$. And finally, we define $F[v, \vec{a}, \vec{b}] = G[v, t, \vec{a}, \vec{b}]$.

Return yes if there are $\vec{a}$ and $\vec{b}$ such that $\sum_{i=1}^{\mathrm{var}_c - 1} a_i \le |X|$, and $\sum_{i=1}^{\mathrm{var}_w - 1} b_i \le |X|$, and $a^{(r)}_{\mathrm{var}_c} \cdot c_{\mathrm{var}_c} + \sum_{i=1}^{\mathrm{var}_c - 1} a_i \cdot c_i \le B$, and $F[r, \vec{a}, \vec{b}] \ge D$ where r is the root of $\mathcal{T}$. Otherwise, return no.

Correctness. For any $v, \vec{a}, \vec{b}$ and i, we prove that $F[v, \vec{a}, \vec{b}]$ and $G[v, i, \vec{a}, \vec{b}]$ store $\max PD_{\mathcal{T}_v}(S)$ for $S \in \mathcal{S}^{(v)}_{\vec{a},\vec{b}}$ and $S \in \mathcal{S}^{(v,i)}_{\vec{a},\vec{b}}$, respectively. This implies that the algorithm is correct. For a taxon x_i, the tree T_{x_i} does not contain edges and so there is no diversity. We can only check if $\vec{a}$ and $\vec{b}$ correspond to a feasible project. So, the table F stores the correct value by the initialization. For an internal vertex v, the children $u_1, \dots, u_t$ of v and $i \in [t-1]$, the entry $G[v, 1, \vec{a}, \vec{b}]$ stores the correct value by the observations that $PD_{\mathcal{T}_{v,1}}(S) = PD_{\mathcal{T}_{u_1}}(S) + \lambda(v, u_1) \cdot w(b^{(u_{i+1})}_{\mathrm{var}_w}, \vec{b})$ for $S \in \mathcal{S}^{(v,1)}_{\vec{a},\vec{b}}$, where $w(b^{(u_{i+1})}_{\mathrm{var}_w}, \vec{b})$ is the survival probability at u_1. Further, the value in entry $F[v, \vec{a}, \vec{b}]$ stores the correct value, when $G[v, t, \vec{a}, \vec{b}]$ stores the correct value, because $\mathcal{S}^{(v)}_{\vec{a},\vec{b}} = \mathcal{S}^{(v,t)}_{\vec{a},\vec{b}}$. It remains to show the correctness of the value in $G[v, i+1, \vec{a}, \vec{b}]$.

Now, assume as an induction hypothesis that the computation of $F[u_j, \vec{a}, \vec{b}]$ and $G[v, i, \vec{a}, \vec{b}]$ store the correct values, for an internal vertex v with children $u_1, \dots, u_t$ and $i \in [t-1]$. We first prove that if $G[v, i+1, \vec{a}, \vec{b}] = p$, then there exists a set $S \in \mathcal{S}^{(v,i+1)}_{\vec{a},\vec{b}}$ with $PD_{\mathcal{T}_v}(S) = p$. Afterward, we prove that $G[v, i+1, \vec{a}, \vec{b}] \ge PD_{\mathcal{T}_{v,i+1}}(S)$ for every set $S \in \mathcal{S}^{(v,i+1)}_{\vec{a},\vec{b}}$.

($\Rightarrow$) Let $G'[v, i+1, \vec{a}, \vec{b}] = d$. Let $\vec{a'}$ and $\vec{b'}$ be the vectors that maximize the right side of Eq. (10) for $G[v, i+1, \vec{a}, \vec{b}]$. By the induction hypothesis, there is a set $S_G \in S^{(v,i)}_{\vec{a}-\vec{a'},\vec{b}-\vec{b'}}$ such that $G[v, i, \vec{a} - \vec{a'}, \vec{b} - \vec{b'}] = PD_{\mathcal{T}_{v,i}}(S_G)$ and there is a set $S_F \in S^{(u_{i+1})}_{\vec{a'},\vec{b'}}$ such that $F[u_{i+1}, i, \vec{a'}, \vec{b'}] = PD_{\mathcal{T}_{u_{i+1}}}(S_F)$. Define $S := S_G \cup S_F$. Then,

$$\begin{aligned} PD_{\mathcal{T}_{v,i+1}}(S) &= PD_{\mathcal{T}_{v,i}}(S_G) + PD_{\mathcal{T}_v}(S_F) && (11)\\ &= PD_{\mathcal{T}_v}(S_G) + PD_{\mathcal{T}_{u_{i+1}}}(S_F) + \lambda(v, u_{i+1}) \cdot w(b^{(u_{i+1})}_{\mathrm{var}_w}, \vec{b}). && (12) \end{aligned}$$

This equals the right side of Eq. (10) and we conclude $PD_{\mathcal{T}_v}(S) = p$.

($\Leftarrow$) Let $S \in \mathcal{S}^{(v,i+1)}_{\vec{a},\vec{b}}$. Let S_F be the set of projects p of S that are from a project list of an offspring of u_{i+1} and define $S_G = S \setminus S_F$. Let a_k be the number of projects in S_F with cost c_k and let b_ℓ be the number of projects in S_F with survival probability b_ℓ. Define $\vec{a'} = (a_1, \dots, a_{\mathrm{var}_c - 1})$ and $\vec{b'} = (b_1, \dots, b_{\mathrm{var}_w - 1})$. Then,

$$\begin{aligned} G[v, i+1, \vec{a}, \vec{b}] &\ge G[v, i, \vec{a} - \vec{a}', \vec{b} - \vec{b}'] + F[u_{i+1}, \vec{a}', \vec{b}'] && (13)\\ &\quad +\lambda(v, u_{i+1}) \cdot w\left(b'^{(u_{i+1})}_{\mathrm{var}_w}, \vec{b}'\right) \\ &= PD_{\mathcal{T}_v}(S_G) + PD_{\mathcal{T}_{u_{i+1}}}(S_F) + \lambda(v, u_{i+1}) \cdot w\left(b'^{(u_{i+1})}_{\mathrm{var}_w}, \vec{b}'\right) && (14) \end{aligned}$$

$$= PD_{\mathcal{T}_v}(S) \tag{15}$$

Inequality (13) follows from the recurrence in Eq. (10). By the definition of S_F and S_G, Eq. (14) is correct. Finally, Eq. (15) follows from Eq. (12).

Running time. First, we prove how many options for vectors $\vec{a}$ and $\vec{b}$ there are: Because $\sum_{i=1}^{\text{var}_c -1} a_i \leq |X|$, we conclude that if $a_i = |X|$, then $a_j = 0$ for $i \neq j$. Otherwise, for $a_i \in [|X|-1]_0$ there are $\mathcal{O}(|X|^{\text{var}_c -1})$ options for $\vec{a}$ of not containing $|X|$, such that there are altogether $\mathcal{O}(|X|^{\text{var}_c -1} + |X|) = \mathcal{O}(|X|^{\text{var}_c -1})$ options for a suitable $\vec{a}$. Likewise, there are $\mathcal{O}(|X|^{\text{var}_w -1})$ options for a suitable $\vec{b}$.

Clearly, the initialization can be done in $\mathcal{O}(\|\mathcal{P}\|)$ time. Let v be an internal vertex and let $\vec{a}$ and $\vec{b}$ be fixed. For a vertex $w \in V$, we can compute in $\mathcal{O}(n)$ time the set off(w). It follows that $w(b_{\text{var}_w}^{(u_i)}, \vec{b})$ can be computed in $\mathcal{O}(n+\text{var}_w)$ time such that Eq. (9) can be computed in $\mathcal{O}(n + \text{var}_w)$ time. As for $\vec{a}$ and $\vec{b}$, there are $\mathcal{O}\left(|X|^{\text{var}_c + \text{var}_w -2}\right)$ options to chose $\vec{a}'$ and $\vec{b}'$. Therefore, Eq. (10) can be evaluated in $\mathcal{O}\left(|X|^{\text{var}_c + \text{var}_w -2} \cdot (n + \text{var}_w)\right)$ time.

Eq. (9) has to be computed once for every internal vertex. Eq. (10) has to be computed once for every vertex except the root. Thus, all entries of the tables F and G can be computed in time

$$\mathcal{O}\left(|X|^{2(\text{var}_c + \text{var}_w -2)} \cdot (n + \text{var}_w) + |X| \cdot \|\mathcal{P}\|\right).$$

Additionally, we need $\mathcal{O}(|X|^{\text{var}_c + \text{var}_w -2} \cdot (\text{var}_c + \text{var}_w))$ time to check whether there are $\vec{a}$ and $\vec{b}$ such that

- $F[r, \vec{a}, \vec{b}] \geq D$,
- $\sum_{i=1}^{\text{var}_c -1} a_i \leq |X|$,
- $\sum_{i=1}^{\text{var}_w -1} b_i \leq |X|$, and
- $a_{\text{var}_c}^{(r)} \cdot c_{\text{var}_c} + \sum_{i=1}^{\text{var}_c -1} a_i \cdot c_i \leq B$.

Because $\mathcal{O}(n) = \mathcal{O}(|X|)$ and $\mathcal{O}(\|\mathcal{P}\|) \leq \mathcal{O}(|X| \cdot \text{var}_w)$, the overall running time is $\mathcal{O}\left(|X|^{2(\text{var}_c + \text{var}_w -1)} \cdot (\text{var}_c + \text{var}_w)\right)$. □

For $0 \xrightarrow{c_i} 1$ NAP, where $\text{var}_w = 2$, we get the following running time.

Corollary 1. $0 \xrightarrow{c_i} 1$ NAP *can be solved in* $\mathcal{O}\left(|X|^{2(\text{var}_c +1)} \cdot \text{var}_c\right)$ *time.*

As each project with a cost higher than B can be deleted, we may assume that there are no such projects which implies that $\text{var}_c \leq C + 1 \leq B + 1$. Thus, Theorem 3 also implies that GNAP is XP with respect to $C + \text{var}_w$ and $B + \text{var}_w$ with astronomical running times of $\mathcal{O}\left(|X|^{2(C+\text{var}_w -1)} \cdot (C + \text{var}_w)\right)$ and $\mathcal{O}\left(|X|^{2(B+\text{var}_w -1)} \cdot (B + \text{var}_w)\right)$, respectively. However, we can adjust the algorithm so that B is not in the exponent of the running time. Instead of declaring how many projects of cost c_i for $i \in [\text{var}_c]$ are selected, we declare the budget that can be spent.

Theorem 4. GNAP *can be solved in* $\mathcal{O}\left(B^2 \cdot |X|^{2(\text{var}_w -1)} \cdot \text{var}_w\right)$ *time.*

Proof. We describe a dynamic programming algorithm with two tables F and G that have a dimension for all the var_w different survival probabilities, except for $\text{var}_w -1$. For a vertex $v \in V$, a given vector $\vec{b}$ and $k \in [B]_0$, we define $\mathcal{S}_{k,\vec{b}}^{(v)}$ to be the family of sets of projects S such that

1. S contains exactly one project of P_i for each $x_i \in \text{off}(v)$,
2. $\text{Cost}(S) \leq k$, and
3. S contains exactly b_ℓ projects of survival probability w_ℓ for each $\ell \in [\text{var}_w -1]$.

For a vertex $v \in V$ with children $u_1, \dots, u_t$, a given vector $\vec{b}$ and integers $k \in [B]_0$ and $i \in [t]$ we define $\mathcal{S}_{k,\vec{b}}^{(v,i)}$ analogously, just that exactly one project of P_i is chosen for each $x_i \in \text{off}(u_1) \cup \cdots \cup \text{off}(u_i)$.

It follows that we can compute how many projects with survival probability w_{var_w} a set $S \in \mathcal{S}_{\vec{a},\vec{b}}^{(v)}$ contains. That are $b_{\text{var}_w}^{(v)} := |\text{off}(v)| - \sum_{j=1}^{\text{var}_w -1} b_j$ projects with survival probability w_{var_w}. The entries $F[v, k, \vec{b}]$ and $G[v, i, k, \vec{b}]$ store the expected phylogenetic diversity of the tree T_v for $S \in \mathcal{S}_{k,\vec{b}}^{(v)}$ and $T_{v,i}$ for $S \in \mathcal{S}_{k,\vec{b}}^{(v,i)}$, respectively. We further define the total survival probability to be $w(b_{\text{var}_w}, \vec{b}) := 1-(1-w_{\text{var}_w})^{b_{\text{var}_w}} \cdot \prod_{i=1}^{\text{var}_w -1}(1-w_i)^{b_i}$, when b_{var_w} and $\vec{b}$ describe the number of chosen single survival probabilities.

Fix a taxon x_i with project list P_i. We store $F[x_i, k, \vec{b}] = 0$, if P_i contains a project $p = (c_t, w_\ell)$ such that $c_t \leq k$, and

1. ($\ell < \text{var}_w$ and $\vec{b} = \vec{0}_{(\ell)+1}$ or $\ell = \text{var}_c$ and $\vec{b} = \vec{0}$).

Otherwise, store $F[x_i, k, \vec{b}] = -\infty$.

Let v be an internal vertex with children $u_1, \dots, u_t$, we define

$$G[v, 1, k, \vec{b}] = F[u_1, k, \vec{b}] + \lambda(v, u_1) \cdot w\left(b_{\text{var}_w}^{(u_1)}, \vec{b}\right) \tag{16}$$

and to compute further values of G, we can use the recurrence

$$G[v, i+1, k, \vec{b}] = \max_{\substack{k' \in [k]_0 \\ \vec{0} \le \vec{b'} \le \vec{b}}} \left\{ \begin{array}{l} G[v, i, k-k', \vec{b}-\vec{b'}] + F[u_{i+1}, k', \vec{b'}] \\ +\lambda(v, u_{i+1}) \cdot w\left(b^{(u_{i+1})}_{\mathrm{var}_w}, \vec{b'}\right) \end{array} \right. . \tag{17}$$

Herein, we write $\vec{p} \le \vec{q}$ if $\vec{p}$ and $\vec{q}$ have the same dimension d and $p_i \le q_i$ for every $i \in [d]$. And finally, we define $F[v, k, \vec{b}] = G[v, t, k, \vec{b}]$.

Return yes if there are $k \in [B]_0$ and $\vec{b}$ such that $\sum_{i=1}^{\mathrm{var}_w - 1} b_i \le |X|$, and $F[r, k, \vec{b}] \ge D$ for the root r of $\mathcal{T}$. Otherwise, return no.

The correctness and the running time can be proven analogously to the proof of Theorem 3. □

Generally, we have to assume that B is exponential in the input size. However, there are special cases of GNAP, in which this is not the case, such as the case with unit costs for projects. Since $\mathrm{var}_w \le 2^{\text{w-code}}$, we conclude the following from Theorem 4.

Corollary 2. GNAP *is* XP *with respect to* var_w *and* w-code*, if B is bounded polynomially in the input size.*

4.2. Generalized Noah's Ark Problem on stars

We now consider Star-GNAP, the special case of GNAP where the phylogenetic X-tree $\mathcal{T}$ has height 1. We first show that Star-GNAP is W[1]-hard with respect to the number $|X|$ of taxa. This implies that Proposition 1(a) cannot be improved to an FPT algorithm. Afterward, we prove that most of the FPT and XP algorithms we presented for MCKP can be adopted for Star-GNAP, yielding algorithms with a faster running time than for GNAP.

4.2.1. Hardness

Theorem 5. Star-GNAP *is* W[1]*-hard with respect to $|X|$, even if the given X-tree $\mathcal{T}$ is ultrametric and* $\mathrm{val}_\lambda = D = 1$.

Proof. *Reduction.* We reduce from MCKP, which by Theorem 2 is W[1]-hard with respect to the number of classes m. Let $\mathcal{I} = (N, \{N_1, \dots, N_m\}, c, d, B, D)$ be an instance of MCKP. We define an instance $\mathcal{I}' = (\mathcal{T}, \mathcal{P}, B' := B, D' := 1)$ in which the X-tree $\mathcal{T} = (V, E, \lambda)$ is a star with center v and the vertex set is $V := \{v\} \cup X$, with $X := \{x_1, \dots, x_m\}$. Set $\lambda(e) := 1$ for every $e \in E$. For every class $N_i = \{a_{i,1}, \dots, a_{i,\ell_i}\}$, define a project list P_i with projects $p_{i,j} := (c_{i,j} := c(a_{i,j}), w_{i,j} := d(a_{i,j})/D)$. The $|X|$-collection of project lists $\mathcal{P}$ contains all these project lists P_i.

Correctness. Because we may assume that $0 \le d(a) \le D$ for all $a \in N$, the survival probabilities fulfill $w_{i,j} \in [0, 1]$ for all $i \in [m]$ and $j \in [|N_i|]$. The tree has m taxa and the reduction is clearly computable in polynomial time, so to prove that it is a parameterized reduction, it only remains to show the equivalence.

($\Rightarrow$) Let S be a solution for $\mathcal{I}$ with $S \cap N_i = \{a_{i,j_i}\}$. We show that $S' = \{p_{i,j_i} \mid i \in [m]\}$ is a solution for $\mathcal{I}'$: The cost of the set S' is $\sum_{i=1}^m c_{i,j_i} = \sum_{i=1}^m c(a_{i,j_i}) \le B$ and further $PD_{\mathcal{T}}(S') = \sum_{(v,x_i) \in E} \lambda(v, x_i) \cdot w_{i,j_i} = \sum_{(v,x_i) \in E} 1 \cdot d(a_{i,j})/D = \frac{1}{D} \cdot \sum_{i=1}^m d(a_{i,j}) \ge 1 = D'$.

($\Leftarrow$) Let $S = \{p_{1,i_1}, \dots, p_{m,j_m}\}$ be a solution for $\mathcal{I}'$. We show that $S' = \{a_{1,i_1}, \dots, a_{m,j_m}\}$ is a solution for $\mathcal{I}$: Clearly, S' contains exactly one item per class. The cost of the set S' is $c_\Sigma(S') = \sum_{i=1}^m c(a_{i,j_i}) = \sum_{i=1}^m c_{i,j_i} \le B$. The diversity of S' is $d_\Sigma(S') = \sum_{i=1}^m d(a_{i,j_i}) = \sum_{i=1}^m w_{i,j_i} \cdot D = PD_{\mathcal{T}}(S) \cdot D \ge D$. □

The X-tree constructed in the reduction above is a star and therefore has unbounded maximum degree Δ. In the following, we show that GNAP also is W[1]-hard with respect to $|X|$ when $\mathcal{T}$ is a binary tree.

Corollary 3. GNAP *is* W[1]*-hard with respect to $|X| + D$ even if $\Delta = 3$ and* $\mathrm{val}_\lambda = 1$.

Proof. *Reduction.* We reduce from Star-GNAP, which by Theorem 5 is W[1]-hard with respect to $|X|$, even if $\mathrm{val}_\lambda = D = 1$. Let $\mathcal{I} = (\mathcal{T}, \mathcal{P}, B, D)$ be an instance of GNAP with $D = 1$. Define an X-tree $\mathcal{T}' := (V, E)$ as follows: Let $V := X \cup \{v_1, \dots, v_{|X|}, x^*\}$ and $E := \{(v_i, x_i), (v_i, v_{i+1}) \mid i \in [|X| - 1]\} \cup \{(v_n, x_n), (v_n, x^*)\}$, and set $\lambda(e) := 1$ for each edge e of $\mathcal{T}'$. Define a project-list $P_{x^*} = (p_{*,0} := (0, 0), p_{*,1} := (1, 1))$ for x^*. Finally, let $\mathcal{I}'$ be $(\mathcal{T}', \mathcal{P} \cup P_{x^*}, B' := B + 1, D' := |X| + 1)$.

Correctness. The reduction can be computed in polynomial time and the size of the taxa set has increased by only one. Note that we can assume that a solution of $\mathcal{I}'$ contains $p_{*,1}$ because otherwise exchanging an arbitrary project with $p_{*,1}$ gives a better solution. It thus remains only to show that S is a solution for $\mathcal{I}$ if and only if $S' := S \cup \{p_{*,1}\}$ is a solution for $\mathcal{I}'$. Clearly, $\mathrm{Cost}(S') = \sum_{p_{i,j} \in S'} c_{i,j} = c_{*,1} + \sum_{p_{i,j} \in S} c_{i,j} = \mathrm{Cost}(S) + 1$. Because $w_{*,1} = 1$, we conclude that the survival probability at each vertex v_i is exactly 1. Thus, the value of $PD_{\mathcal{T}'}(S')$ is

$$\sum_{i=1}^{|X|-1} \lambda(v_i, v_{i+1}) \cdot 1 + \lambda(v_{|X|}, x^*) \cdot 1 + \sum_{p_{i,j} \in S} \lambda(v_i, x_i) \cdot w_{i,j} = |X| + PD_{\mathcal{T}}(S).$$

Hence, $\mathrm{Cost}(S') \le B + 1$ and $PD_{\mathcal{T}'}(S') \ge |X| + 1$ if and only if $\mathrm{Cost}(S) \le B$ and $PD_{\mathcal{T}}(S) \ge 1$. □

In Section 3, we presented a set of algorithms for MCKP. We will now make use of these algorithms to obtain several tractability results for Star-GNAP.

Proposition 7. Star-GNAP *can be solved in time*

- $\mathcal{O}(D \cdot 2^{\text{w-code}} \cdot \|\mathcal{P}\| + |\mathcal{I}|)$,
- $\mathcal{O}(B \cdot \|\mathcal{P}\| + |\mathcal{I}|)$,
- $\mathcal{O}(C \cdot \|\mathcal{P}\| \cdot |X| + |\mathcal{I}|)$, *and*
- $\mathcal{O}(|X|^{\text{var}_c - 1} \cdot \|\mathcal{P}\| + |\mathcal{I}|)$.

Proof. To show the statement, we describe how to reduce Star-GNAP to MCKP and then use algorithms presented in Section 3.1.

Reduction. Let $\mathcal{I} = (\mathcal{T}, \lambda, \mathcal{P}, B, D)$ be an instance of Star-GNAP. We define an instance $\mathcal{I}' = (N, \{N_1, \dots, N_{|X|}\}, c, d, B', D')$ of MCKP. Without loss of generality, each survival probability is in the form $w_i = w_i'/2^{\text{w-code}}$ with $w_i' \in \mathbb{N}_0$, and $w_i' \leq 2^{\text{w-code}}$. For every taxon x_i with project list P_i, define the class N_i and for every project $p_{i,j} = (c_{i,j}, w_{i,j}) \in P_i$ add an item $a_{i,j}$ to N_i with cost $c(a_{i,j}) := c_{i,j}$ and value $d(a_{i,j}) := w_{i,j}' \cdot \lambda(w, x_i)$. Set $B' := B$ and $D' := D \cdot 2^{\text{w-code}}$.

Correctness. Let $S = \{p_{1,j_1}, \dots, p_{|X|,j_{|X|}}\}$ be a solution for the instance $\mathcal{I}$ of GNAP. Define the set $S' = \{a_{1,j_1}, \dots, a_{|X|,j_{|X|}}\}$. Clearly, $c_\Sigma(S') = \text{Cost}(S) \leq B$. Further,

$$\begin{aligned}\sum_{i=1}^{|X|} d(a_{i,j_i}) &= \sum_{i=1}^{|X|} w_{i,j_i}' \cdot \lambda(w, x_i)\\ &= 2^{\text{w-code}} \cdot \sum_{i=1}^{|X|} w_{i,j_i} \cdot \lambda(w, x_i)\\ &= 2^{\text{w-code}} \cdot PD_{\mathcal{T}}(S) \geq 2^{\text{w-code}} \cdot D = D'.\end{aligned}$$

Thus, S' is a solution for $\mathcal{I}'$.

Analogously, one can show that if S' is a solution for the instance $\mathcal{I}'$ of MCKP, then S is a solution for the GNAP-instance $\mathcal{I}$.

Running time. The reduction can be computed in $\mathcal{O}(|\mathcal{I}|)$ time. We observe that in $\mathcal{I}'$ the size of the set N equals the number $\|\mathcal{P}\|$ of projects, the number of classes m is the number $|X|$ of taxa, and the budget B remains the same. Because all costs are simply copied, the maximal cost C and the number of different costs var_c remain the same. Because the survival probabilities are multiplied with an edge weight, it follows that $\text{var}_d \in \mathcal{O}(\text{var}_w \cdot \text{val}_\lambda)$. By definition, $D' = D \cdot 2^{\text{w-code}}$.

Thus, with this reduction at hand, we can obtain any of the claimed running times by using the MCKP running time bound given in Table 3 for the corresponding parameter. □

To obtain an FPT algorithm for $\text{var}_c + \text{var}_d$ we cannot directly make use of the corresponding FPT result for MCKP because var_d may be as large as $\text{var}_w \cdot \text{val}_\lambda$. Instead, we present a direct reduction from Star-GNAP to ILP-Feasibility.

Theorem 6. *There is a reduction from* Star-GNAP *to* ILP-Feasibility *with* $\mathcal{O}(2^{\text{var}_c + \text{var}_d} \cdot \text{var}_c)$ *variables. Thus,* Star-GNAP *is* FPT *with respect to* $\text{var}_c + \text{var}_w$.

Proof. *Reduction.* Let $\mathcal{I} = (\mathcal{T}, \lambda, \mathcal{P}, B, D)$ be an instance of Star-GNAP and let ρ denote the root of $\mathcal{T}$.

We may assume that a project list P_i does not contain two projects of the same cost or the same value. In the following, we call $T = (C, W)$ a *type*, for sets $C \subseteq \mathcal{C}$ and $W \subseteq \mathcal{W}$ with $|C| = |W|$, where $\mathcal{C}$ and $\mathcal{W}$ are the sets of different costs and survival probabilities, respectively. Let $\mathcal{X}$ be the family of all types. We say that the *project list* P_i *is of type* $T = (C, W)$ if C and W are the set of costs and survival probabilities of P_i. For each $T \in \mathcal{X}$, we define m_T be the number of classes of type T.

Observe, for each type $T = (C, W)$, project list P_i of type T, and a project $p \in P_i$, we can compute the survival probability of p when we know the cost c_k of p. More precisely, if c_k is the ℓth cheapest cost in C, then the survival probability of p is the ℓth smallest survival probability in W. For a type $T = (C, W)$ and $i \in [\text{var}_c]$, we define the constant $w_{T,i}$ to be $-n \cdot \text{val}_\lambda$ if $c_i \notin C$. Otherwise, let $w_{T,i} \in [0, 1]$ be the ℓth smallest survival probability in W, if c_i is the ℓth smallest cost in C.

For two taxa x_i and x_j with project lists P_i and P_j of the same type T, it is possible that $\lambda(v, x_i) \neq \lambda(v, x_j)$. Hence, it can make a difference if a project is selected for the taxon x_i instead of x_j. For a type T, let $x_{T,1}, \dots, x_{T,m_T}$ be the taxa, such that the project lists $P_{T,1}, \dots, P_{T,m_T}$ are of type T and let $\lambda(v, x_{T,i}) \geq \lambda(v, x_{T,i+1})$ for each $i \in [m_T - 1]$. For each type T, we define a function $f_T : [m_T]_0 \to \mathbb{N}$ by $f_T(0) := 0$ and $f_T(\ell)$ stores total value of the first ℓ edges. More precisely, that is $f_T(\ell) := \sum_{i=1}^{\ell} \lambda(v, x_i)$.

The following describes an instance of ILP-Feasibility.

$$\sum_{T \in \mathcal{X}} \sum_{i=1}^{\text{var}_c} y_{T,i} \cdot c_i \leq B \tag{18}$$

$$\sum_{T\in\mathcal{X}}\sum_{i=1}^{\mathrm{var}_c} w_{T,i}\cdot g_{T,i} \geq D \tag{19}$$

$$f_T\left(\sum_{\ell=i}^{\mathrm{var}_c} y_{T,\ell}\right) - f_T\left(\sum_{\ell=i+1}^{\mathrm{var}_c} y_{T,\ell}\right) = g_{T,i} \qquad \forall T\in\mathcal{X}, i\in[\mathrm{var}_c] \tag{20}$$

$$\sum_{i=1}^{\mathrm{var}_c} y_{T,i} = m_T \qquad \forall T\in\mathcal{X} \tag{21}$$

$$y_{T,i}, g_{T,i} \geq 0 \qquad \forall T\in\mathcal{X}, i\in[\mathrm{var}_c] \tag{22}$$

The variable $y_{T,i}$ expresses the number of projects with cost c_i that are chosen in a project list of type T. We want to assign the most valuable edges that are incident with taxa that have a project list of type T to the taxa in which the highest survival probability is chosen. To receive an overview, in g_{T,var_c} we store the total value of the y_{T,var_c} most valuable edges that are incident with a taxon that has a project list of type T. Then, in g_{T,var_c-1} we store the total value of the next y_{T,var_c-1} most valuable edges, and so on.

For each type T, the function f_T is not necessarily linear. However, for each f_T there are affine linear functions $p_T^{(1)}, \dots, p_T^{(m_T)}$ such that $f_T(i) = \min_\ell p_T^{(\ell)}(i)$ for each $i \in [m_T]$ [8].

Correctness. Observe that if $c_i \notin C$, then because we defined $d_{T,i}$ to be $-n \cdot \mathrm{val}_\lambda$, Inequality (19) would not be fulfilled if $g_{T,i} > g_{T,i+1}$ and consequently $y_{T,i} = 0$ if $c_i \notin C$ for each type $T = (C, W) \in \mathcal{X}$ and $i \in [\mathrm{var}_c]$. Inequality (18) ensures that the total cost is at most B. Inequality (19) ensures that the total phylogenetic diversity is at least D. The variable $g_{T,i}$ stores the total weight of the edges towards the $y_{T,i}$ taxa with projects of project lists of type T, in which a project of cost c_i is selected. All these projects have survival probability $w_{T,i}$, and thus the phylogenetic diversity of these projects is $w_{T,i}\cdot(g_{T,i}-g_{T,i+1})$. Eq. (20) ensures that the value of $g_{T,i}$ is chosen correctly. Eq. (21) ensures that exactly m_T projects are picked from the project lists of type T, for each $T \in \mathcal{X}$. Altogether, this shows the correctness of the reduction. Finally, the constructed instance has $\mathcal{O}(2^{\mathrm{var}_c+\mathrm{var}_d}\cdot\mathrm{var}_c)$ variables because $\mathcal{X} \subseteq 2^{\mathcal{C}} \times 2^{\mathcal{W}}$ which implies that there are $\mathcal{O}(2^{\mathrm{var}_c+\mathrm{var}_d}\cdot\mathrm{var}_c)$ different options for the variables $y_{T,i}$ and $g_{T,i}$. □

5. Restriction to two projects per taxon

We finally study two special cases of $a_i \xrightarrow{c_i} b_i$ NAP—the special case of GNAP where every project list contains exactly two projects.

5.1. Sure survival or extinction for each project

First, we consider $0 \xrightarrow{c_i} 1$ NAP, the special case where each taxon x_i survives definitely if cost c_i is paid and becomes extinct, otherwise. This special case was introduced by Pardi and Goldman [24] who also presented a pseudopolynomial-time algorithm that computes a solution in $\mathcal{O}(B^2 \cdot n)$ time. Because we may assume that $B \leq C \cdot |X|$, we conclude the following.

Corollary 4. $0 \xrightarrow{c_i} 1$ NAP *can be solved in* $\mathcal{O}(C^2 \cdot |X|^3)$ *time.*

We now show that $0 \xrightarrow{c_i} 1$ NAP is FPT with respect to D, with an adaption of the above-mentioned algorithm of Pardi and Goldman [24] for the parameter B.

Proposition 8. $0 \xrightarrow{c_i} 1$ NAP *can be solved in* $\mathcal{O}(D^2 \cdot |X|)$ *time.*

Proof. *Table definition.* We write $c(x_i)$ for the cost of the project with survival probability 1 in P_i. For a set A of vertices, a vertex v, and integers b and d, we call a set of projects S an (A, v, d)*-respecting set*, if $PD_{\mathcal{T}_v}(S) \geq d$, and S contains exactly one project of the project lists of the offspring of A, and S contains at least one project with survival probability 1.

We describe a dynamic programming algorithm with two tables F and G. We want entry $F[v, d]$ for a vertex $v \in V$, an integer $d \in [D]_0$ to store the minimal cost of a $(\{v\}, v, d)$-respecting set. If v is an internal vertex with children $u_1, \dots, u_t$ and $i \in [t]$, then we want entry $G[v, i, d]$ to store the minimal cost of a $(\{u_1, \dots, u_i\}, v, d)$-respecting set.

Algorithm. In the algorithm description, we let $-_{\geq 0}$ denote non-negative subtraction, that is, $a -_{\geq 0} b := \max(a - b, 0)$.

As basic cases for a leaf x_i store $F[x_i, 0] = c(x_i)$ and for each $d > 0$ store $F[x_i, d] = \infty$.

Now, let v be an internal vertex with children $u_1, \dots, u_t$. We define $G[v, 1, d] = F[u_1, d -_{\geq 0} \lambda(vu_1)]$. If for a fixed $i \in [t]$ the values of $G[v, i, d]$ and $F[u_{i+1}, d]$ are known for each $d \in [D]_0$, we compute the value of $G[v, i+1, d]$ with the recurrence

$$G[v, i+1, d] = \min\{G[v, i, d]; \min_{d' \in [d_{i+1}]} \{G[v, i, d_{i+1} - d'] + F[u_{i+1}, d']\}\},$$

where $d_{i+1} := d -_{\geq 0} \lambda(vu_{i+1})$. We set $F[v, d] := G[v, t, d]$, eventually.

We return yes if $F[\rho, D] \leq B$ for the root ρ of $\mathcal{T}$ and no, otherwise.

Correctness. The basic cases, the computation of $G[v, 1, d]$, and the computation of $F[v, d]$ for an internal vertex v and $d \in [D]_0$ are correct by definition. It remains to show that $G[v, i+1, d]$ stores the correct value if $G[v, i, d]$ and $F[u_{i+1}, d]$ do. We first show that if S is a $(\{u_1, \dots, u_{i+1}\}, v, d)$-respecting set, then $G[v, i+1, d] \leq \mathrm{Cost}(S)$. Afterward, we show that if $G[v, i+1, d] = c$, then there is a $(\{u_1, \dots, u_{i+1}\}, v, d)$-respecting set S with $\mathrm{Cost}(S) = c$.

($\Rightarrow$) Let S be a $(\{u_1, \dots, u_{i+1}\}, v, d)$-respecting set. Let S_1 be the set of projects that are in one of the project lists of the offspring of u_{i+1}. Define $S_2 := S \setminus S_1$. If S_2 only contains projects of survival probability 0, then $G[v, i+1, d] = G[v, i, d]$. Otherwise, let $d' := PD_{\mathcal{T}_v}(S_1)$. Then, $PD_{\mathcal{T}_v}(S_2) = PD_{\mathcal{T}_v}(S) - PD_{\mathcal{T}_v}(S_1) \geq d - d'$. We conclude that S_1 is a $(\{u_{i+1}\}, v, d')$-respecting set and S_2 is a $(\{u_1, \dots, u_i\}, v, d - d')$-respecting set. Consequently,

$$G[v, i+1, d] \leq G[v, i, d - d'] + F[u_{i+1}, d' -_{\geq 0} \lambda(vu_{i+1})] \tag{23}$$

$$\leq \mathrm{Cost}(S_G) + \mathrm{Cost}(S_F) = \mathrm{Cost}(S). \tag{24}$$

Here, Inequality (23) follows from the recurrence in Recurrence (23) and Inequality (24) follows from the induction hypothesis.

($\Leftarrow$) Let $G[v, i+1, d]$ store c. Unless $G[v, i+1, d] = G[v, i, d]$ there is an integer $d' \in [d_{i+1}]_0$, such that $G[v, i+1, d] = G[v,_{i+1}, d_{i+1} - d'] + F[u_{i+1}, d']$. By the induction hypothesis, there is a $(\{u_{i+1}\}, v, d')$-respecting set S_1 and a $(\{u_1, \dots, u_i\}, v, d - d')$-respecting set S_2 such that $F[u_{i+1}, d'] = \mathrm{Cost}(S_1)$ and $G[v, i, d - d'] = \mathrm{Cost}(S_2)$. We conclude that $S := S_1 \cup S_2$ is a $(\{u_1, \dots, u_{i+1}\}, v, d)$-respecting set and $c = \mathrm{Cost}(S_1) + \mathrm{Cost}(S_2) = \mathrm{Cost}(S)$.

Running time. Table F has $\mathcal{O}(D \cdot n)$ entries, and each entry can be computed in constant time. Also, G has $\mathcal{O}(D \cdot n)$ entries. Entry $G[v, i+1, d]$ is computed by checking at most $D + 1$ options for d'. Altogether, a solution can be found in $\mathcal{O}(D^2 \cdot n)$ time. □

We may further use this pseudopolynomial-time algorithm to obtain an algorithm for the maximum edge weight val_λ: If we are given an input instance $\mathcal{I}$ of $0 \xrightarrow{c_i} 1$ NAP such that $\sum_{e \in E} \lambda(e) < D$, then we may directly return no since the desired diversity can never be reached. After this check, we may assume $D \leq \sum_{e \in E} \lambda(e) \leq \mathrm{val}_\lambda \cdot (n-1)$. This gives the following running time bound.

Corollary 5. $0 \xrightarrow{c_i} 1$ NAP *can be solved in* $\mathcal{O}((\mathrm{val}_\lambda)^2 \cdot |X|^3)$ *time.*

5.2. Unit costs for each project

Second, we consider Unit-Cost NAP—the special case of GNAP in which every project with a positive survival probability has the same cost. In the following we use the term *solution* to denote only those projects of cost 1 which have been chosen. Further, with $w(x_i)$ we denote the survival probability of the project in P_i which costs 1.

We consider the following basic problem.

Penalty-Sum
Input: A set of tuples $T = \{t_i = (a_i, b_i) \mid i \in [n], a_i \in \mathcal{Q}_{\geq 0}, b_i \in (0, 1)\}$, two integers k, Q, a number $D \in \mathbb{Q}_+$.
Question: Is there a subset $S \subseteq T$ of size k such that $\sum_{t_i \in S} a_i - Q \cdot \prod_{t_i \in S} b_i \geq D$?

In an earlier version of this work [17], we asked for complexity of Penalty-Sum. It has been proven meanwhile that Penalty-Sum is NP-hard [14]. We now present a polynomial-time many-to-one reduction from Penalty-Sum to Unit-Cost NAP, giving the following hardness result.

Theorem 7. Unit-Cost NAP *is NP-hard, even on X-trees with height at most 2 and a root r of degree 1.*

Proof. *Reduction.* Let $\mathcal{I} = (T, k, Q, D)$ be an instance of Penalty-Sum. Let $\mathrm{bin}(a)$ and $\mathrm{bin}(1-b)$ be the maximum binary encoding length of any a_i and $1 - b_i$, respectively, and let $W := 2^{\mathrm{bin}(a)+\mathrm{bin}(1-b)}$. We define an instance $\mathcal{I}' = (\mathcal{T}, \mathcal{P}, B, D')$ of Unit-Cost NAP as follows: Let $\mathcal{T}$ contain the vertices $V := \{r, v, x_1, \dots, x_{|T|}\}$ and let $\mathcal{T}$ be a star with center v and root r. We define $\lambda(r, v) = WQ$ and $\lambda(v, x_i) = W \cdot ({}^{a_i}/{}_{1-b_i})$ for each $t_i \in T$. For each tuple t_i, we define a project list $P_i := ((0, 0), (1, 1-b_i))$. Then, $\mathcal{P}$ is defined to be the set of these project lists. Finally, we set $B := k$, and $D' := W(D+Q)$. The reduction can clearly be computed in polynomial time.

Correctness. We show that $\mathcal{I}$ is a yes-instance of Penalty-Sum if and only if $\mathcal{I}'$ is a yes-instance of Unit-Cost NAP.

($\Rightarrow$) Let S be a solution for $\mathcal{I}$ of Penalty-Sum. Let the set S' contain the project $(1, w(x_i))$ if and only if $t_i \in S$. Then,

$$\sum_{x_i \in S'} \lambda(v, x_i) \cdot w(x_i) + \lambda(r, v) \cdot \left(1 - \prod_{x_i \in S'} (1 - w(x_i))\right)$$

$$= \sum_{t_i \in S'} W \cdot (a_i/1-b_i) \cdot (1-b_i) + WQ \cdot \left(1 - \prod_{t_i \in S'} (1-(1-b_i))\right)$$
$$= \sum_{t_i \in S} Wa_i - WQ \cdot \prod_{t_i \in S} b_i + WQ \geq W \cdot (D+Q) = D'.$$

Because $|S'| = |S| \leq B$, we have that S' is a solution for the instance $\mathcal{I}'$ of Unit-Cost NAP.

($\Leftarrow$) Let S' be a solution for the instance $\mathcal{I}'$ of Unit-Cost NAP. We conclude $\sum_{x_i \in S'} \lambda(v, x_i) \cdot w(x_i) \geq D' - \lambda(r, v) \cdot \left(1 - \prod_{x_i \in S'}(1 - w(x_i))\right)$. Define the set $S \subseteq T$ to contain a tuple t_i if and only if S' contains a project of the taxon x_i. Then,

$$\sum_{t_i \in S} a_i - Q \cdot \prod_{t_i \in S} b_i$$
$$= \sum_{x_i \in S'} W^{-1} \cdot \lambda(v, x_i) \cdot (1-b_i) - W^{-1} \cdot \lambda(r, v) \cdot \prod_{x_i \in S'} (1 - w(x_i))$$
$$= W^{-1} \cdot \left(\sum_{x_i \in S'} \lambda(v, x_i) \cdot w(x_i) - \lambda(r, v) \cdot \prod_{x_i \in S'} (1 - w(x_i))\right)$$
$$\geq W^{-1} \cdot \left(D' - \lambda(r, v) \cdot \left(1 - \prod_{x_i \in S'} (1 - w(x_i))\right) - \lambda(r, v) \cdot \prod_{x_i \in S'} (1 - w(x_i))\right)$$
$$= W^{-1} \cdot \left(D' - \lambda(r, v)\right) = W^{-1} \cdot (W(D+Q) - WQ) = D.$$

Because $|S'| = |S| \leq B$, we may conclude that S is a solution of Penalty-Sum. $\square$

Recall that in an ultrametric tree, the weighted distance from the root to all vertices is the same. Observe that Unit-Cost NAP can be solved exactly on ultrametric trees of height at most 2 by greedily selecting a taxon which adds the highest estimated diversity. In the following theorem, we show that Unit-Cost NAP is NP-hard even when restricted to ultrametric trees of height at most 3.

Theorem 8. Unit-Cost NAP *is NP-hard on ultrametric trees of height at most 3.*

Proof. By Theorem 7, it is sufficient to reduce from Unit-Cost NAP with the restriction that the root has only one child and the height of the tree is 2.

Reduction. Let $\mathcal{I} = (\mathcal{T}, \mathcal{P}, B, D)$ be an instance of Unit-Cost NAP in which the root r of the X-tree $\mathcal{T}$ has only one child v and $\text{height}_{\mathcal{T}} = 2$. Without loss of generality, assume for each $i \in [|X| - 1]$ that $\lambda(v, x_i) \geq \lambda(v, x_{i+1})$ and there is a fixed $s \in [|X|]$ with $w(x_s) \geq w(x_j)$ for each $j \in [|X|]$. Observe, that by the reduction in Theorem 7 we may assume $w(x_j) \neq 1$ for each $x_j \in X$.

We define an instance $\mathcal{I}' := (\mathcal{T}', \mathcal{P}, B', D')$ of Unit-Cost NAP as follows: Let $X_1 \subseteq X$ be the set of vertices x_i with $\lambda(v, x_1) = \lambda(v, x_i)$. If $X_1 = X$, then $\mathcal{I}$ is already ultrametric and the reduction may simply output $\mathcal{I}$. Assume otherwise and define $X_2 := X \setminus X_1$. Fix an integer $W \in \mathbb{N}$ that is large enough such that $1 - W^{-1} > w_{s,1}$ and $W \cdot \lambda(v, x_{|X|}) > \lambda(v, x_1)$. Define a tree $\mathcal{T}' = (V', E', \lambda')$, in which V' contains the vertices V and for every $x_i \in X_2$, we add two vertices u_i and x_i^*. Let X^* be the set that contains all x_i^*. The set of edges is defined by $E' = \{(r, v)\} \cup \{(v, x_i) \mid x_i \in X_1\} \cup \{(v, u_i), (u_i, x_i), (u_i, x_i^*) \mid x_i \in X_2\}$. Observe that the leaf set of $\mathcal{T}$ is $X \cup X^*$. The weights of the edges are defined by:

- $\lambda'(r, v) = \lambda(r, v) \cdot W^{|X_2|} \cdot (W - 1)$.
- For $x_i \in X_1$ we set $\lambda'(v, x_i) = (W - 1) \cdot \lambda(v, x_1)$.
- For $x_i \in X_2$ we set $\lambda'(v, u_i) = W \cdot (\lambda(v, x_1) - \lambda(v, x_i))$ and $\lambda'(u_i, x_i) = \lambda'(u_i, x_i^*) = (W \cdot \lambda(v, x_i)) - \lambda(v, x_1)$.

The collection $\mathcal{P}'$ contains the unchanged project lists P_i for taxa $x_i \in X$ and we add the project lists $P_i^* := ((0, 0), (1, 1-W^{-1}))$ for each taxon $x_i^* \in X^*$. Finally, we define $B' := B + |X^*|$ and

$$D' = (W - 1) \cdot \left(D + |X_2| \cdot (W - 1) \cdot \lambda(v, x_1) + \left(W^{|X_1|} - 1\right) \cdot \lambda(w, v)\right).$$

Fig. 1 depicts an example of this reduction.

Observe that t can be chosen such that $t \in \mathcal{O}(\text{w-code} + \text{val}_\lambda)$. Hence, the reduction can be computed in polynomial time. Moreover, the X-tree $\mathcal{T}'$ has a height of 3.

Before showing the correctness, we first prove that $\mathcal{T}'$ is ultrametric. In the rest of the proof we denote with λ_w the value $\lambda(e)$, where e is the only edge that is directed towards w. To show that $\mathcal{T}'$ is ultrametric, for each $x \in X \cup X^*$ we show that the paths from v to x have the same length as the edge from v to x_1. This is sufficient because every path from the root to a taxon visits v. By definition, the claim is correct for every $x_i \in X_1$. For an $x_i \in X_2$, the path from v to x_i is

$$\lambda'_{u_i} + \lambda'_{x_i} = W \cdot (\lambda_{x_1} - \lambda_{x_i}) + W \cdot \lambda_{x_i} - \lambda_{x_1} = (W - 1) \cdot \lambda_{x_1} = \lambda'_{x_1}$$

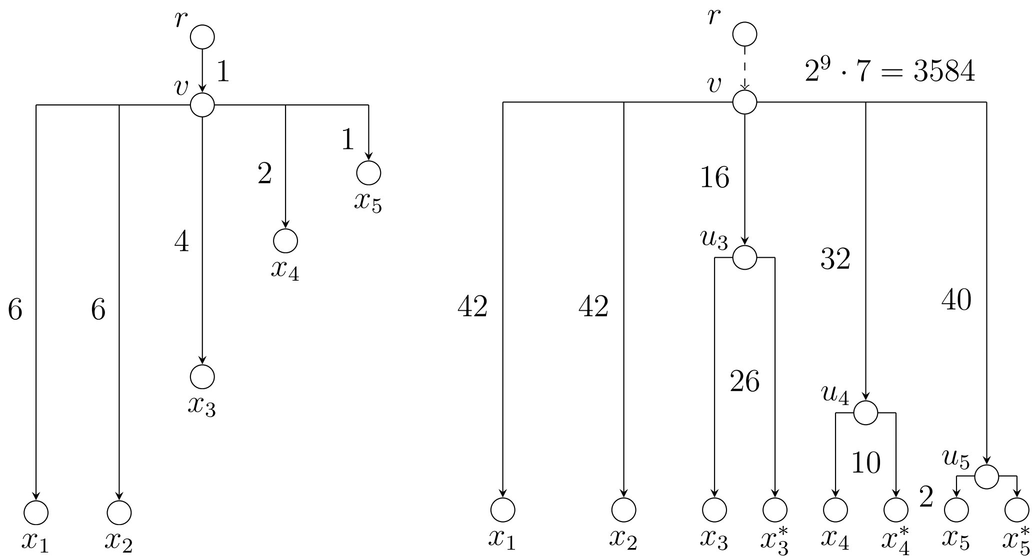

Fig. 1. An example of the reduction presented in Theorem 8, where the left side shows an example-instance $\mathcal{I}$ and the right side shows the instance $\mathcal{I}'$. Here, the survival probabilities are omitted and we assume that $t = 3$.

By definition, this is also the length of the path from v to $x_i^* \in X_i^*$. We conclude that $\mathcal{T}'$ is an ultrametric tree.

Correctness. We now show that $\mathcal{I}$ is a yes-instance of Unit-Cost NAP if and only if $\mathcal{I}'$ is a yes-instance of Unit-Cost NAP.

($\Rightarrow$) Let $S \subseteq X$ be a set of taxa. Define $S' := S \cup X^*$. We show that S is a solution for $\mathcal{I}$ if and only if S' is a solution of $\mathcal{I}'$. First, $|S'| = |S| + |X^*|$ and hence $|S| \le B$ if and only if $|S'| \le B' = B + |X^*|$. We compute the phylogenetic diversity of S' in $\mathcal{T}'$. Here, we first consider the diversity from the subtrees below v and then the diversity from the edge (r, v). For the subtrees rooted at v containing a leaf of X_1 the total contribution is

$$D_1 = \sum_{x_i \in X_1 \cap S} \lambda'_{x_1} \cdot w(x_i) = \sum_{x_i \in X_1 \cap S} (W-1) \cdot \lambda_{x_1} \cdot w(x_i).$$

For the subtrees rooted at v where S' contains x_i^* but not x_i, the total contribution is

$$\begin{aligned} D_2 &= \sum_{x_i \in X_2 \setminus S} (\lambda'_{u_i} + \lambda'_{x_i}) \cdot w(x_i^*) \\ &= \sum_{x_i \in X_2 \setminus S} (W \cdot (\lambda_{x_1} - \lambda_{x_i}) + W \cdot \lambda_{x_i} - \lambda_{x_1}) \cdot (1 - W^{-1}) \\ &= \sum_{x_i \in X_2 \setminus S} (W-1) \cdot \lambda_{x_1} \cdot (1 - W^{-1}). \end{aligned}$$

For the subtrees rooted at v where S' contains x_i^* and x_i^*, the total contribution is

$$\begin{aligned} D_3 &= \sum_{x_i \in X_2 \cap S} \lambda'_{u_i} \cdot (1 - (1 - w(x_i^*)) \cdot (1 - w(x_i))) \\ &\quad + \sum_{x_i \in X_2 \cap S} \lambda'_{x_i} \cdot w(x_i) + \sum_{x_i \in X_2 \cap S} \lambda'_{x_i} \cdot w(x_i^*) \\ &= \sum_{x_i \in X_2 \cap S} W \cdot (\lambda_{x_1} - \lambda_{x_i}) \cdot (1 - W^{-1} \cdot (1 - w(x_i))) \\ &\quad + \sum_{x_i \in X_2 \cap S} (W \cdot \lambda_{x_i} - \lambda_{x_1}) \cdot (w(x_i) + 1 - W^{-1}) \\ &= \sum_{x_i \in X_2 \cap S} (\lambda_{x_1} - \lambda_{x_i}) \cdot (W - (1 - w(x_i))) \\ &\quad + \sum_{x_i \in X_2 \cap S} (W \cdot \lambda_{x_i} - \lambda_{x_1}) \cdot (w(x_i) + 1 - W^{-1}) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{x_i \in X_2 \cap S} \Big[\lambda_{x_1} \cdot (W - (1 - w(x_i)) - w(x_i) - 1 + W^{-1}) \\
&\qquad + \lambda_{x_i} \cdot (-W + (1 - w(x_i)) + W(w(x_i) + 1 - W^{-1}))\Big] \\
&= \sum_{x_i \in X_2 \cap S} \Big[\lambda_{x_1} \cdot (W - 2 + W^{-1}) + \lambda_{x_i} \cdot (W - 1) \cdot w(x_i)\Big] \\
&= \sum_{x_i \in X_2 \cap S} (W - 1) \cdot \Big[\lambda_{x_1} \cdot (1 - W^{-1}) + \lambda_{x_i} \cdot w(x_i)\Big].
\end{aligned}
$$

Finally, for the edge (r, v) the contribution is

$$
\begin{aligned}
D_0 &= \lambda'_v \cdot \left(1 - \prod_{x_i^* \in X_2} (1 - w(x_i^*)) \cdot \prod_{x_i \in S} (1 - w(x_i))\right) \\
&= \lambda_v \cdot W^{|X_2|} \cdot (W - 1) \cdot \left(1 - W^{-|X_2|} \cdot \prod_{x_i \in S} (1 - w(x_i))\right) \\
&= \lambda_v \cdot W^{|X_2|} \cdot (W - 1) - \lambda_v \cdot (W - 1) \cdot \prod_{x_i \in S} (1 - w(x_i)).
\end{aligned}
$$

Altogether we conclude

$$
\begin{aligned}
&PD_{\mathcal{T}'}(S') \\
&= D_0 + D_1 + D_2 + D_3 \\
&= \lambda_v \cdot W^{|X_2|} \cdot (W - 1) - \lambda_v \cdot (W - 1) \cdot \prod_{x_i \in S} (1 - w(x_i)) \\
&\quad + \sum_{x_i \in X_1 \cap S} (W - 1) \cdot \lambda_{x_1} \cdot w(x_i) \\
&\quad + \sum_{x_i \in X_2 \setminus S} (W - 1) \cdot \lambda_{x_1} \cdot (1 - W^{-1}) \\
&\quad + \sum_{x_i \in X_2 \cap S} (W - 1) \cdot (\lambda_{x_1} \cdot (1 - W^{-1}) + \lambda_{x_i} \cdot w(x_i)) \\
&= (W - 1) \cdot \left[\lambda_v \cdot W^{|X_2|} - \lambda_v \cdot \prod_{x_i \in S} (1 - w(x_i)) \right. \\
&\qquad \left. + \sum_{x_i \in S} \lambda_{x_i} \cdot w(x_i) + \sum_{x_i \in X_2} \lambda_{x_1} \cdot (1 - W^{-1})\right] \\
&= (W - 1) \cdot \left[\lambda_v \cdot \left(1 - \prod_{x_i \in S} (1 - w(x_i))\right) + \sum_{x_i \in S} \lambda_{x_i} \cdot w(x_i) \right. \\
&\qquad \left. + \lambda_v \cdot (W^{|X_2|} - 1) + \sum_{x_i \in X_2} \lambda_{x_1} \cdot (1 - W^{-1})\right] \\
&= (W - 1) \cdot \Big[PD_{\mathcal{T}}(S) + (W^{|X_2|} - 1) \cdot \lambda_v + |X_2| \cdot (1 - W^{-1}) \cdot \lambda_{x_1}\Big]
\end{aligned}
$$

It directly follows that $PD_{\mathcal{T}'}(S') \geq D'$ if and only if $PD_{\mathcal{T}}(S) \geq D$.

($\Leftarrow$) We show that $\mathcal{I}'$ has a solution S' with $X^* \subseteq S'$. Then, $S' \setminus X^*$ is a solution for $\mathcal{I}$ as shown in the proof of the converse direction.

Let S' be a solution for $\mathcal{I}'$ that contains a maximum number of elements of X^* among all solutions. If $X^* \subseteq S'$, then we are done. Otherwise, choose some $x_i^* \notin S'$. We show that there is a solution containing x_i^* and all elements of $S' \cap X^*$, contradicting the choice of S'. If $x_i \in S'$, then consider the set $S_1 := (S' \setminus \{x_i\}) \cup \{x_i^*\}$. Now, $|S_1| = |S'| \leq B'$ and because $w(x_i^*) = 1 - W^{-1} > w(x_i)$, we may conclude that $PD_{\mathcal{T}'}(S_1) > PD_{\mathcal{T}'}(S') \geq D'$. If $S' \subset X^*$, then consider the set $S_2 := X^*$. Now, $|S_2| = |X^*| \leq B'$ and $PD_{\mathcal{T}'}(S_2) \geq PD_{\mathcal{T}'}(S') \geq D'$. Finally, assume that $x_i, x_i^* \notin S'$ and $x_j \in S'$ for some $j \neq i$. Again, $w(x_i^*) = 1 - W^{-1} > w(x_i)$ and we know that the length of the path from v to x_i^* and x_j is the same. Consider the set $S_3 := (S' \setminus \{x_j\}) \cup \{x_i^*\}$ and observe $|S_3| = |S'| \leq B'$. Moreover, by the above, $PD_{\mathcal{T}'}(S_3) > PD_{\mathcal{T}'}(S') \geq D'$. □

6. Discussion

In this paper, we studied the Generalized Noah's Ark Problem (GNAP), a natural generalization of the classical problem $0 \xrightarrow{c_i} 1$ NAP. We established several tractability and intractability results for GNAP and some of its special cases. Specifically, we showed that GNAP is W[1]-hard with respect to the number of taxa, $|X|$, but can be solved in polynomial time when there are only a constant number of costs and survival probabilities of projects. We further introduced Unit-Cost NAP, a problem where all taxa are saved at the same cost, but may differ in their survival probability. We showed that Unit-Cost NAP is NP-hard, even on ultrametric trees of height three.

Naturally, several open questions remain. For example, it is not known whether GNAP admits a pseudopolynomial-time algorithm or whether GNAP is strongly NP-hard. Moreover, it remains open whether GNAP is FPT with respect to $\mathrm{var}_c + \mathrm{var}_w$ or the budget B. The latter is unresolved even for Unit-Cost NAP.

While projects, modeled through costs and survival probabilities, are one way to capture a better decision-making process in conservation planning, other approaches have also been studied. One direction incorporates ecological dependencies, such as a food webs which represent predator–prey relationships. A species can then only be preserved if it is either a source of the food web, or can find sufficient prey among other preserved species [20,26]. So far, the complexity of the problems has been analyzed only in the simple setting where k taxa whose survival probability is lifted from 0 to 1 are selected [18,20,26]. Another line of research generalizes phylogenetic trees to networks, allowing *horizontal gene transfer* and *hybridization* in so called *reticulations*. This concept, however, requires new definitions for phylogenetic diversity on networks. Over the years, several such definitions have been introduced [4,13,29]. More recently, these two perspectives — ecological dependencies and phylogenetic networks — have been combined, analyzed, and efficient algorithms have been presented [15]. It seems only natural to study the complexity of GNAP and its various special cases when they are combined with these richer models of phylogenetic diversity.

Data availability

No data was used for the research described in the article.

References

[1] M. Bansal, V. Venkaiah, Improved fully polynomial time approximation scheme for the 0-1 multiple-choice knapsack problem, Int. Inst. Inf. Technol. Tech. Rep. (2004) 1–9.
[2] A. Billionnet, Solution of the generalized Noah's ark problem, Syst. Biol. 62 (1) (2013) 147–156.
[3] A. Billionnet, How to take into account uncertainty in species extinction probabilities for phylogenetic conservation prioritization, Environ. Model. Assess. 22 (6) (2017) 535–548.
[4] M. Bordewich, C. Semple, K. Wicke, On the complexity of optimising variants of phylogenetic diversity on phylogenetic networks, Theoret. Comput. Sci. 917 (2022) 66–80.
[5] M. Cygan, F.V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, S. Saurabh, Parameterized Algorithms, Springer, 2015.
[6] R.G. Downey, M.R. Fellows, Fixed-parameter tractability and completeness II: On completeness for W[l], Theoret. Comput. Sci. 141 (1–2) (1995) 109–131.
[7] R.G. Downey, M.R. Fellows, Fundamentals of Parameterized Complexity, in: Texts in Computer Science, Springer, 2013.
[8] M. Etscheid, S. Kratsch, M. Mnich, H. Röglin, Polynomial kernels for weighted problems, J. Comput. System Sci. 84 (2017) 1–10.
[9] D.P. Faith, Conservation evaluation and phylogenetic diversity, Biol. Cons. 61 (1) (1992) 1–10.
[10] M.R. Fellows, S. Gaspers, F.A. Rosamond, Parameterizing by the number of numbers, Theory Comput. Syst. 50 (4) (2012) 675–693.
[11] A. Frank, É. Tardos, An application of simultaneous diophantine approximation in combinatorial optimization, Combinatorica 7 (1) (1987) 49–65.
[12] F. Gurski, C. Rehs, J. Rethmann, Knapsack problems: A parameterized point of view, Theoret. Comput. Sci. 775 (2019) 93–108.
[13] L. van Iersel, M. Jones, J. Schestag, C. Scornavacca, M. Weller, Average-tree phylogenetic diversity of networks, in: Proceedings of the 25th International Workshop on Algorithms in Bioinformatics, WABI 2025, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2025, pp. 14:1–14:21.
[14] L. van Iersel, M. Jones, J. Schestag, C. Scornavacca, 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, Springer, 2025, pp. 107–130.
[15] M. Jones, J. Schestag, Parameterized algorithms for diversity of networks with ecological dependencies, in: Proceedings of the 20th International Symposium on Parameterized and Exact Computation, IPEC 2025, Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2025.
[16] H. Kellerer, U. Pferschy, D. Pisinger, Knapsack Problems, Springer, 2004.
[17] C. Komusiewicz, 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, Springer, 2023, pp. 109–121.
[18] C. Komusiewicz, 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, pp. 28:1–28:18.
[19] H.W. Lenstra Jr., Integer programming with a fixed number of variables, Math. Oper. Res. 8 (4) (1983) 538–548.
[20] V. Moulton, C. Semple, M. Steel, Optimizing phylogenetic diversity under constraints, J. Theoret. Biol. 246 (1) (2007) 186–194.
[21] R.M. Nauss, The 0–1 knapsack problem with multiple choice constraints, European J. Oper. Res. 2 (2) (1978) 125–131.
[22] F. Pardi, Algorithms on Phylogenetic Trees (Ph.D. thesis), University of Cambridge Cambridge, UK, 2009.
[23] F. Pardi, N. Goldman, Species choice for comparative genomics: Being greedy works, PLoS Genet. 1 (6) (2005) e71.
[24] F. Pardi, N. Goldman, Resource-aware taxon selection for maximizing phylogenetic diversity, Syst. Biol. 56 (3) (2007) 431–444.
[25] D. Pisinger, A minimal algorithm for the multiple-choice knapsack problem, European J. Oper. Res. 83 (2) (1995) 394–410.
[26] 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.
[27] M. Steel, Phylogenetic diversity and the greedy algorithm, Syst. Biol. 54 (4) (2005) 527–529.
[28] M.L. Weitzman, The Noah's ark problem, Econometrica (1998) 1279–1298.
[29] K. Wicke, M. Fischer, Phylogenetic diversity and biodiversity indices on phylogenetic networks, Math. Biosci. 298 (2018) 80–90.