FRIEDRICH-SCHILLER-
UNIVERSITÄT
JENA

Who Should Have a Place on the Ark? Parameterized Algorithms for the Maximization of Phylogenetic Diversity

DISSERTATION
FOR THE DEGREE OF DOCTOR OF NATURAL SCIENCES
DOCTOR RERUM NATURALIUM (DR. RER. NAT.)

Submitted to the Council of the Faculty of Mathematics and Computer Science
of the Friedrich Schiller University Jena
on July 29, 2024, by
Jannik "Theodor" Schestag, M.Sc.,
born May 20, 1995 in Heilbronn, Germany.

Expert Reviewer (**Gutachter**):
Prof. Dr. Christian Komusiewicz, Friedrich-Schiller-Universität Jena, Germany
Prof. Dr.ir. Leo J.J. van Iersel, TU Delft, The Netherlands
Dr. Mathias Weller, Université Gustave Eiffel, Paris, France
Defended on **January 17, 2025**

Ehrenwörtliche Erklärung

Hiermit erkläre ich,

- dass mir die Promotionsordnung der Fakultät bekannt ist,
- dass ich die Dissertation selbst angefertigt habe, keine Textabschnitte oder Ergebnisse eines Dritten oder eigenen Prüfungsarbeiten ohne Kennzeichnung übernommen und alle von mir benutzten Hilfsmittel, persönlichen Mitteilungen und Quellen in meiner Arbeit angegeben habe,
- dass ich die Hilfe eines Promotionsberaters nicht in Anspruch genommen habe und dass Dritte weder unmittelbar noch mittelbar geldwerte Leistungen von mir für Arbeiten erhalten haben, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen,
- dass ich die Dissertation noch nicht als Prüfungsarbeit für eine staatliche oder andere wissenschaftliche Prüfung eingereicht habe.

Bei der Auswahl und Auswertung des Materials sowie bei der Herstellung des Manuskripts haben mich folgende Personen unterstützt:

Leo van Iersel (Co-Autor), Mark Jones (Co-Autor), Christian Komusiewicz (Co-Autor), Celine Scornavacca (Co-Autorin) und Mathias Weller (Co-Autor).

Ich habe die gleiche, eine in wesentlichen Teilen ähnliche bzw. eine andere Abhandlung bereits bei einer anderen Hochschule als Dissertation eingereicht: Ja/Nein*.
(*Zutreffendes unterstreichen)
Wenn Ja, Name der Hochschule: —
Ergebnis: —

Ort, Datum Jannik Schestag, Unterschrift

Schestag, Jannik

Who Should Have a Place on the Ark?
Parameterized Algorithms for the Maximization of Phylogenetic Diversity

Dissertation,
Friedrich-Schiller-Universität Jena, submitted July 2024, defended January 2025.

Curriculum vitae

- since January 2025: Member of the *Discrete Mathematics and Optimization* group in a postdoctoral position,
  Technische Universiteit Delft, The Netherlands.
- October 2023 to July 2024: Member of the *Algorithm Engineering* group,
  Friedrich-Schiller-Universität Jena, Germany.
- April 2023 to September 2023: Member of the *Discrete Mathematics and Optimization* group,
  Technische Universiteit Delft, The Netherlands.
- May 2022 to March 2023: Member of the *Algorithmics* group,
  Philipps-Universität Marburg, Germany.
- April 2019 to February 2022: M.Sc. in *Computer Science*,
  Philipps-Universität Marburg, Germany.
- April 2016 to December 2018: B.Sc. in *Computer Science*,
  Philipps-Universität Marburg, Germany.
- October 2015 to June 2020: B.Sc. in *Economical Mathematics*,
  Philipps-Universität Marburg, Germany.

Acknowledgments

First and foremost, I humbly thank my redeemer, Jesus Christ, through whom the foundations of the heavens and the earth were created (Colossians 1:16, John 1:3) and who died on the cross as I deserve (Romans 6:23) so that I may live as He deserves (2 Corinthians 5:21). Jesus not only became my God but also my best friend. All the strength, concentration, and willpower I needed to accomplish every part of this work were surely given by Him (Philippians 2:13). All glory, honor, and power be unto Him forever and ever (Revelation 4:11). Amen!

Academically, I express my deepest gratitude to my supervisor, Christian Komusiewicz, for his multitude of great ideas and quick answers to my questions. Additionally, I thank him for the endless hours spent writing applications and for always staying supportive, even in situations where many other professors would have opted out.

Though officially he is just a co-author, I want to thank Mark Jones as a 'minor' supervisor. His knowledge, humor, and way of working during the cooperative time left a very positive feeling with me and definitively improved my working spirit.

Furthermore, I am grateful to Niels Grüttemeier and Frank Sommer, who initially supervised my Bachelor's and Master's degrees and later became colleagues. Also, I thank my other colleagues Alexander Bille, Jaroslav Garvardt, Nils Morawietz, Luca Staus, and the Master's student Anton Herrmann, for creating a pleasant working atmosphere, sharing one or the other joke in times, and providing deep talks. I further want to thank all of my co-authors, listed in lexicographic order: Matthias Bentert, Jaroslav Garvardt, Niels Grüttemeier, Leo van Iersel, Mark Jones, Christian Komusiewicz, Simone Linz, Ber Lorke, Nils Morawietz, Malte Renken, Frank Sommer, Celine Scornavacca, and Mathias Weller.

Gratefully, I received a financial support by the Deutscher Akademischer Austauschdienst (DAAD—German Academic Exchange Service), project 57556279, from April to September 2023. I thank the TU Delft and especially Leo van Iersel, Mark Jones, and their colleagues for hosting me during this period.

Last but not least, for their unwavering support throughout my life, I want to express my sincerest thanks to my mother, Kerstin Schestag, and my sister, Mona Felina Schestag. Surely, I would have not come to where I am without them. While it is impossible to list all the important people in my life, I explicitly want to express thanks to my grandmother Edeltraud Schestag, Klaus Pieper, Holger & Regine Hellmich, Ricarda Schwarz, the Slembeck family, the Köhler family, Niklas Vogt, Jan Mischon, my late father Joachim Schestag, my late grandfather Ulrich Hellmich, and everyone who supportively prayed for me!

Preface

Within this thesis, I present the results of the research I conducted in the field of parameterized algorithms for the maximization of phylogenetic diversity from May 2022 until June 2024. During these two years, I have at first been part of the Algorithmics research group led by Christian Komusiewicz at the Philipps-Universität Marburg, Germany. Because of receiving a six-month scholarship I worked some time in the Discrete Mathematics and Optimization research group led by Karen Aardal at the TU Deft, The Netherlands. After my pleasant stay in the Netherlands, I again moved to Germany to work with Christian Komusiewicz; this time in the Algorithm Engineering research group at the Friedrich-Schiller-Universität Jena, Germany.

The results presented in this thesis are predominantly published in scientific papers on the platform ArXiv. Many of them are however not yet published in conference proceedings or journals. In the following, I will explain which chapters are based on which publications and what participation I had in the research and writing process. Afterward, I will give a brief list of other scientific publications I coauthored that are not included in this thesis.

It is worth mentioning that the order of chapters in this thesis does not corresponding to when the research occurred. To minimize confusion: The order of research was Generalized Noah's Ark Problem, Max-All-Paths-PD, Time-PD, Max-Net-PD, and finally Optimizing PD with Dependencies. This order may be good to keep in min for readers, as algorithmic ideas for Max-All-Paths-PD appear in Chapter 6 of this thesis but are basis for many algorithms in Chapters 4 and 5.

Chapter 3 is based on the publication "A Multivariate Complexity Analysis of the Generalized Noah's Ark Problem" written with Christian Komusiewicz, which can be found on the platform ArXiv [KS23a].A preliminary version of this publication appeared in the *Proceedings of the 19th Cologne-Twente Workshop on Graphs and Combinatorial Optimization (CTW 2023)* [KS23b]. During my initial literature research on what kind of generalizations of Max-PD there are, I first considered studying Generalized Noah's Ark Problem (and Optimizing PD with Dependencies). As it was my first project as a Ph.D. student and I was at first only working from home, I was relatively slow in the research and first had to start with basic concepts. I made the key discovery to find the strong connection to the Multiple-Choice Knapsack. The overall research was conducted in cooperation with Christian whereby Christian gave the better ideas. The writing-up process was predominately conducted by me. Christian provided helpful hints and wrote the introduction. I gave the talk at CTW 2023. This work also contains a small

observation, Observation 3.14, which is not published elsewhere.

Chapter 4 is based on the publication "Maximizing Phylogenetic Diversity under Time Pressure: Planning with Extinctions Ahead" written with Mark Jones, which can be found on the platform ArXiv [JS24]. The research of TIME-PD and S-TIME-PD was initialized by me during my stay in Delft, after Mark and I finished the research on MAX-ALL-PATHS-PD. We were actually stuck in the research process for quite a while because it was difficult to find out how to cope with the scheduling and tree-structures at the same time. Mark made the final breakthrough and had the rough idea for the FPT-algorithm for TIME-PD with respect to D (Lemma 4.7). I quickly had the idea of how to utilize the idea to show that also S-TIME-PD is FPT when parameterized by D (Lemma 4.8). Afterward, it took us several months of work to show that S-TIME-PD is FPT when parameterized by $\overline{D}$ (Theorem 4.2). The according proof is arguably the most technical and difficult in this thesis. Mark and I had a fair share in the writing-up process of the results. I additionally provided the connection to the scheduling field and he wrote up most of the introduction.

Chapter 5 is based on the publication "Maximizing Phylogenetic Diversity under Ecological Constraints: A Parameterized Complexity Study" written with Christian Komusiewicz, which can be found on the platform ArXiv [KS24a]. A preliminary version of this publication appeared in the *Proceedings of the 44th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2024)* [KS24b]. As mentioned above, the idea of the study of OPTIMIZING PD WITH DEPENDENCIES came along with my first literature research. Christian and I conducted the research roughly on an equal level of scientific input. The writing-up process was predominately conducted by me. Christian provided helpful hints and wrote the introduction.

Chapter 6 is predominantly based on the research of MAX-ALL-PATHS-PD and only Section 6.6 is about MAX-NET-PD. However, also the introduction and discussion of the chapter contain material from the MAX-NET-PD-paper. The research is contained in the two following publications. Firstly, "How Can We Maximize Phylogenetic Diversity? Parameterized Approaches for Networks" was written with Mark Jones, which can be found on the platform archive [JS23a]. A preliminary version of this publication appeared in the *Proceedings of the 18th International Symposium on Parameterized and Exact Computation (IPEC 2023)* [JS23b]. Secondly, "Maximizing Network Phylogenetic Diversity" was written with Leo van Iersel, Mark Jones, Celine Scornavacca, and Mathias Weller, which can be found on the platform ArXiv [vIJS$^+$24]. Christian Komusiewicz suggested me also to study phylogenetic diversity in phylogenetic networks. We then came into contact with Mark Jones and Leo van Iersel and asked them whether I could join them in Delft for a time. After

their confirmation, I applied for a scholarship from DAAD with a great deal of help from Christian. Mark and I were both very active in the research of Max-All-Paths-PD so that Leo did not join the paper in the end. Mark and I conducted the research roughly on an equal level of scientific input and also the writing-up roughly on equal sides. I gave the talk at IPEC 2023. After the conference version, I proposed a color-coding algorithm with respect to $\overline{D}$ (Theorem 6.4) which improves to the previous algorithm in running time and working on general graphs and not only on binary graphs. Further, Mark and I together developed a kernelization algorithm with respect to the reticulation number. Arising from the research of Max-All-Paths-PD, Mark proposed to study Max-Net-PD in an established research group where I joined. In the meetings, Mark and I quickly showed that there is a reduction from GNAP to Max-Net-PD on level-1-networks (Theorem 6.9). We then collectively also proved the NP-hardness of Penalty Sum (Section 2.6) and an FPT-algorithm for the number of reticulations. I wrote up what is Section 6.6 in this thesis, Mark what is Section 2.6 in this thesis, Mathias the FPT-algorithm, and Celine and Leo the intro, prelims, and discussion of the paper.

Chapters 1, 2, and 7, these are the introduction, the preliminaries, and the conclusion, are based on parts of all five papers. The examination of Multiple-Choice Knapsack, presented in Section 2.5, is part of the GNAP-paper [KS23b]. The examination of Penalty Sum, presented in Section 2.6, is part of the Max-Net-PD-paper [vIJS+24], except for Proposition 2.21 which was not published elsewhere.

During my time as a Ph.D. student, I also participated in the research of the following publications. These are ordered by when the research project started.

- "On Critical Node Problems with Vulnerable Vertices", with Niels Grüttemeier, Christian Komusiewicz, and Frank Sommer. This paper is based on my Master's thesis. Journal: *Journal of Graph Algorithms and Applications* [SGKS24]. Conference: *33rd International Workshop on Combinatorial Algorithms (IWOCA 2022)* [SGKS22].
- "On the Complexity of Parameterized Local Search for the Maximum Parsimony Problem", with Christian Komusiewicz, Simone Linz, and Nils Morawietz. Conference: *34th Annual Symposium on Combinatorial Pattern Matching (CPM 2023)* [KLMS23].
- "Finding Degree-Constrained Acyclic Orientations", with Jaroslav Garvardt, Malte Renken, and Mathias Weller. Conference: *18th International Symposium on Parameterized and Exact Computation (IPEC 2023)* [GRSW23].
- "On the Complexity of Finding a Sparse Connected Spanning Subgraph in a Non-Uniform Failure Model", with Matthias Bentert, and Frank Sommer. Conference: *18th International Symposium on Parameterized and Exact Computation (IPEC 2023)* [BSS23a]. ArXiv: [BSS23b].
- "Protective and Nonprotective Subset Sum Games: A Parameterized Complexity Analysis", with Jaroslav Garvardt, Christian Komusiewicz, and Ber Lorke. Conference: *8th International Conference on Algorithmic Decision Theory (ADT 2024)* [GKLS24].

Abstract

Phylogenetic Diversity (PD) is a well-regarded measure of the overall biodiversity of a set of present-day species (taxa) that indicates its ecological significance. In the Maximize Phylogenetic Diversity (Max-PD) problem one is asked to find a small set of taxa in a phylogenetic tree for which this measure is maximized. Max-PD is particularly relevant in conservation planning, where limited resources necessitate prioritizing certain taxa to minimize biodiversity loss. Although Max-PD can be solved in polynomial time [Steel, Systematic Biology, 2005; Pardi and Goldman, PLoS Genetics, 2005], its generalizations—which aim to model biological processes and other aspects in conservation planning with greater accuracy—often exhibit NP-hardness, making them computationally challenging. This thesis explores a selection of these generalized problems within the framework of parameterized complexity.

In Generalized Noah's Ark Problem (GNAP), each taxon only survives at a certain survival probability, which can be increased by investing more money in the taxon. We show that GNAP is W[1]-hard with respect to the number of taxa but is XP with respect to the number of different costs and different survival probabilities. Additionally, we show that unit-cost-NAP, a special case of GNAP, is NP-hard.

In Time Sensitive Maximization of Phylogenetic Diversity (Time-PD), different extinction times of taxa are considered after which they can no longer be saved. For Time-PD, we present color-coding algorithms that prove that Time-PD is fixed-parameter tractable (FPT) with respect to the threshold of diversity and the acceptable loss of diversity.

In Optimizing PD with Dependencies (PDD), each saved taxon must be a source in the ecological system or a predator of another saved species. These dependencies are given in a food-web. We show that PDD is FPT when parameterized with the size of the solution plus the height of the phylogenetic tree. Further, we consider parameters characterizing the food-web and prove that PDD is FPT with respect to the treewidth if the phylogenetic tree is a star, disproving an open conjecture.

Phylogenetic Diversity is traditionally defined on phylogenetic trees, but phylogenetic networks have gained popularity as they enable the modeling of so-called reticulation events. Max-All-Paths-PD (MapPD) and Max-Net-PD are two problems in maximizing phylogenetic diversity on phylogenetic networks. MapPD is W[2]-hard with respect to the solution size but we show that MapPD is FPT with respect to the number of reticulations and the treewidth, two parameters that show how tree-like the phylogenetic network is. Max-Net-PD however remains NP-hard even on level-1-networks.

Zusammenfassung (Translation of the Abstract)

Die phylogenetische Diversität (PD) ist ein anerkanntes Maß für die gesamte Biodiversität, das die ökologische Bedeutung einer Gruppe heutiger Arten (Taxa) anzeigt. Im Problem Maximize Phylogenetic Diversity (Maximierung der phylogenetischen Diversität, Max-PD) wird gefordert, eine kleine Menge an Taxa in einem phylogenetischem Baum zu finden, für welche dieses Maß maximiert wird. Dieses Problem ist besonders relevant in der Naturschutzplanung, bei der begrenzte Ressourcen die Priorisierung bestimmter Taxa zur Minimierung des Biodiversitätsverlusts erfordern. Obwohl Max-PD in polynomialer Zeit gelöst werden kann [Steel, Systematic Biology, 2005; Pardi und Goldman, PLoS Genetics, 2005], weisen seine Verallgemeinerungen - welche biologische Prozesse und andere Aspekte des Artenschutzes genauer modellieren sollen - oft NP-Härte auf, was sie rechnerisch anspruchsvoller macht. Diese Arbeit untersucht eine Auswahl dieser generalisierten Probleme im Rahmen der parametrisierten Komplexität.

In Generalized Noah's Ark Problem (Generalisiertes Noahs-Arche-Problem, GNAP) überlebt jedes Taxon nur mit einer bestimmten Überlebenswahrscheinlichkeit, die durch mehr Investitionen in das Taxon erhöht werden kann. Wir zeigen, dass GNAP in Bezug auf die Anzahl der Taxa W[1]-schwer, jedoch XP in Bezug auf die Anzahl der verschiedenen Kosten und Überlebenswahrscheinlichkeiten ist. Zusätzlich zeigen wir, dass das unit-cost-NAP (Einheitspreis-NAP), ein Spezialfall von GNAP, NP-schwer ist.

In Time Sensitive Maximization of Phylogenetic Diversity (zeitkritische Maximierung der phylogenetischen Diversität, Time-PD) werden unterschiedliche Aussterbezeiten der Taxa berücksichtigt, nach denen diese nicht mehr gerettet werden können. Für Time-PD präsentieren wir Farbmarkierungsalgorithmen, die beweisen, dass Time-PD in Bezug auf die geforderte Diversität und den akzeptablen Verlust an Diversität festparameter handhabbar (FPT) ist.

In Optimizing PD with Dependencies (Optimieren phylogenetischer Diversität bei Abhängigkeiten, PDD) muss jedes gerettete Taxon eine Quelle im ökologischen System oder ein Raubtier einer anderen geretteten Art sein. Diese Abhängigkeiten sind in einem Nährungsnetz gegeben. Wir zeigen, dass PDD FPT ist, wenn es in Bezug auf die Größe der Lösung plus die Höhe des phylogenetischen Baumes parametrisiert wird. Darüber hinaus betrachten wir Parameter, die das Nährungsnetz charakterisieren, und beweisen, dass PDD in Bezug auf die Baumweite FPT ist, wenn der phylogenetische Baum ein Stern ist, womit wir eine offene Vermutung widerlegen.

Phylogenetische Diversität wird traditionell auf phylogenetischen Bäumen defi-

niert, aber phylogenetische Netzwerke haben an Popularität gewonnen, da sie auch die Modellierung sogenannter retikulierender Ereignisse ermöglichen. Max-All-Paths-PD (Maximierung aller Pfade [zur Bestimmung] der phylogenetischen Diversität, MapPD) und Max-Net-PD (Maximierung der phylogenetischen Netzwerk-Diversität) sind zwei Probleme zur Maximierung der phylogenetischen Diversität in phylogenetischen Netzwerken. MapPD ist in Bezug auf die Lösungsgröße W[2]-schwer, aber wir zeigen, dass MapPD in Bezug auf die Anzahl der Retikulationen und die Baumweite FPT ist, zwei Parameter, welche zeigen, wie Baum-ähnlich das phylogenetische Netzwerk ist. Max-Net-PD bleibt jedoch auch auf Level-1-Netzwerken NP-schwer.

Contents

Chapter 1

Introduction

Human activities [CME17] in the current Western economic system, driven by significant overconsumption and the externalization of costs to society and the environment [Mar20, LLS15], have significantly contributed to the world's ecological system approaching a sixth mass extinction [RWN+17] within the last decades and thereby created arguably one of mankind's biggest challenges for this century. While activists and scientists alike are waiting for a serious political response to this crisis [MBR+23], the looming threat posed by climate change is predicted to further exacerbate the decline in species. The 2023 report of the Intergovernmental Panel on Climate Change (IPCC) [IPCC23, Page 73] warns that as the average global temperature increases, thousands of plant and animal species around the world face increasing risk of extinction. With a "climate breakdown" [Hic20] on the horizon, media outlets worldwide report with drastic words on an increasing number of species that are already extinct or are inevitably about to become extinct in the near future [BBC23, Nic23, DJ22].

The reason for quantifying the loss of biodiversity in public media by naming the number of extinct species is relatively evident. The sheer number of extinct species is a very easy-to-understand metric. Additionally, it subliminally carries a political message: Not a single further species should face extinction. However, a bit more realism reveals the tragedy that certainly not enough resources—of monetary nature, political willpower, land-wise, time-wise, and further—are available to save each and every single species alive today. Already in 1990, May [May90] argued in a seminal note that the subsequent question of which set of species should be saved, "raises larger questions about the ways in which relative values are assigned to different creatures".

Indeed, when it seems impossible to save the entire "tree of life", some species have to be selected to focus on. May further describes this selection as "making choices

for the ineluctably limited number of places on the ark" [May90], where he compares to the biblical narrative of Noah who built an ark to save his family and pairs of animals from a great flood. In the de facto approach, a lot of effort in protection is spent on species which are perceived to be "cute" or "dangerous" [PZZ+21] or which have the "right color" [PF13].

1.1 Phylogenetic Diversity

While humans can bond to species [Wil84] and therefore feel the need to protect a species that they consider aesthetically pleasing, such a factor most certainly does not state how important a species is in the ecological system. However, requiring the information which species have such an importance is crucial, as "it is foolish to allow destruction of nature without knowing what it is worth" [Cro97]. In the years after May's note, a lot of effort in the scientific world has been put into the question of how to measure the relevance of a set of taxa or species. The key for this search was the usage of *phylogenetic trees* [HWVW95], which are represented by graphs with vertices and edges. Each leaf represents a living taxon (species) and internal vertices represent common ancestors of the leaves that can be reached or biological categories—technically known as operational taxonomic units. Usually, the set of taxa is denoted with X and one then refers to phylogenetic X-trees.[1]

As an inaugural idea, it was proposed to count the number of internal vertices that the spanning tree of a given set of taxa in the phylogenetic tree has [VWHW91, NW92] to measure the diversity of this set. This measure, called *cladistic diversity*, was soon improved by adding weights on the edges of the phylogenetic tree. Independently, Faith [Fai92] and Weitzmann [Wei92] proposed *phylogenetic diversity*, where Weitzmann's definition even generalizes the usage of this metric to arbitrary items such as books in a library. The phylogenetic diversity of a set A of taxa, denoted by $\mathrm{PD}_{\mathcal{T}}(A)$, is the total weight of all edges that are on a path from the root to a leaf that corresponds to a taxon in A. In Figure 1.1 an example of cladistic and phylogenetic diversity is given.

Intuitively, with phylogenetic diversity one measures the expected range of biological features of a given set of taxa. We, however, need to be a bit cautious as it is not always correct to estimate the number of features with the phylogenetic diversity [WMS21]. Nevertheless, maximizing phylogenetic diversity has become the standard, albeit imperfect, measure for the biological diversity of a set

[1] Note that phylogenetic trees may be rooted or unrooted. In this thesis, we only consider the rooted variant.

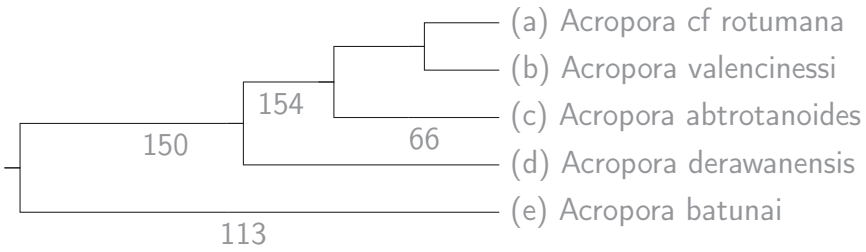


Figure 1.1: A weighted phylogenetic tree of a selection of Acropora corals. For edges without numbers, the evolutionary distances are not specified [FR12].
The cladistic diversity of the sets of corals $\{a, c, e\}$ and $\{a, e\}$ is 4, each. The phylogenetic diversity of the sets of corals $\{a, c, e\}$ is 483 and of $\{a, c\}$ is 370, assuming that edges without numbers have a weight of 0.

of taxa [GCJW+15, ITC+07, MPC+18]. This measure forms the basis of the Fair Proportion Index and the Shapley Value [HKS08, Har13, RM06], which are used to evaluate the individual contribution of individual taxa to overall biodiversity. Over the years, phylogenetic diversity nevertheless became the most well-regarded metric in measuring the biodiversity and therefore the value of a set of species (see [Cro97]), which for example is used by the IUCN's Phylogenetic Diversity Task Force (`https://www.pdtf.org/`) and the Zoological Society of London's EDGE of Existence program [ITC+07]. Despite all the mentioned positive features of phylogenetic diversity, to not present a one-sided picture, we also want to mention that there are some objections raised against the usage of phylogenetic diversity [WDS13] and many preservation projects also do not consider phylogenetic diversity [MP08, SM12].

Based on the phylogenetic diversity index, Daniel Faith proposed a maximization problem, Max-PD, in which one has to find a relatively small set of taxa that achieves maximal phylogenetic diversity [Fai92]. The set needs to be small because limited resources impose monetary, environmental, or other restrictions to the preservation project. More formally, Max-PD, formulated as a decision problem, is defined as follows.

Maximize Phylogenetic Diversity (Max-PD)

Input: A phylogenetic X-tree $\mathcal{T}$ and integers k and D.
Question: Is there a set of taxa $S \subseteq X$ of size at most k such that $\mathrm{PD}_{\mathcal{T}}(S) \geq D$?

Faith already stated that Max-PD can be computed with a greedy algorithm and therefore is computationally easy [Fai92]. Formal proofs were given by Steel [Ste05] and Pardi and Goldman [PG05], independently. Consequently, Max-PD can be

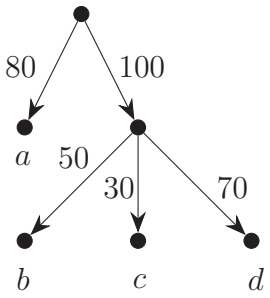
solved within seconds, even on large instances [MKvH06].

1.2 Modeling Real-World Biological Processes

As much as the lightweight definition of Max-PD is an advantage in understanding and quickly solving the problem, just as much of a drawback it is in modeling real-world biological processes. Among these disadvantages stand that it is not realistic that all species can be saved for the same price, that all species have the same remaining time before an extinction, or that the survival of each species is decided without regarding the interaction of species. To cope with these issues, several further problems were introduced which, in contrast to Max-PD, are NP-hard [PG07, HS06, JS24, MSS07]. Therefore, it is unlikely that these problems can be solved with an algorithm that only consumes time polynomial to the size of the input. In this thesis, we examine a selection of NP-hard problems in which one aims to maximize phylogenetic diversity. We study these problems from a classical and parameterized point of view and want to help understanding what makes these problems tractable and provide algorithms that break this intractability to a certain degree. While more mathematical definitions follow in the next chapter, the reader at this point only needs to understand that if a problem is XP or FPT, then an algorithm exists which solves the problem in a desirable time. For W[1]-hard problems, the existence of an FPT-algorithm is unlikely. A deeper view of the theoretical framework is given in the next chapter. In the following, we briefly present the problems considered in this work and summarize some results.

Generalized Noah's Ark Problem. One of the first steps was to allow that the protection of taxa may have different costs [PG07]. This approach introduced the problem Budgeted NAP, also referred to as $0 \xrightarrow{c_i} 1$ [2]-NAP, which is shown to be NP-hard by a reduction from Knapsack [PG07].

Subsequent approaches also allowed to model uncertainty as follows. For example, performing an action to protect some species does not guarantee the survival of that species but only raises the survival probability [Wei98]. In this model, one now aims to maximize the *expected* phylogenetic diversity. That is, the weight of an edge is only added with the probability that at least one of the offspring in the subtree survives. 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 combinations of different actions. Each choice is then associated with a cost and with a resulting survival probability. This model was studied by Hartmann and Steel [HS06], Pardi [Par09]



Taxon a	
0	0
1	0.3
2	0.5
5	0.8
10	0.9

Taxon b	
0	0
1	0.2
3	0.5
10	0.75

Taxon c	
0	0
1	0.1
2	0.3
3	0.5
7	0.8

Taxon d	
0	0
1	0.2
2	0.4
5	0.6
7	0.9

Figure 1.2: An example of an instance of GNAP with a phylogenetic tree on the left and the lists of projects to the right. In each table, in the left column the cost of the project is shown and in the right column the associated survival probability. Spending 2 on Taxon a, 1 on Taxon b, 0 on Taxon c, and 5 on Taxon d would cost 8 and give an expected diversity of $80 \cdot 0.5 + 50 \cdot 0.2 + 30 \cdot 0 + 70 \cdot 0.6 + 100 \cdot (1 - 0.8 \cdot 0.4) = 40 + 10 + 42 + 68 = 160$.

and Billionnet [Bil13, Bil17] as Generalized Noah's Ark Problem (GNAP). An example of an instance of GNAP is given in Figure 1.2.

It feels natural that putting more effort into a certain species increases their survival probability. While the biblical story tells us that for God it was possible to restore species with a single male and female after the great flood, such a project seems unlikely to succeed for men without divine skills. So, in GNAP one can consider the projects to refer to the cost and the according survival probability that are associated with the protection of a certain amount of individuals of the same kind. However, projects can also be a lot more general.

We consider GNAP and also put a focus on a special case of GNAP in which every taxon only has a single project that has a cost of 1 and a positive survival probability. We show that this special case, which we call unit-cost-NAP, is NP-hard even if the phylogenetic tree of the input has a height of 2 or is ultrametric and has a height of 3. Furthermore, we show that GNAP is XP with respect to the number of unique costs or the number of unique survival probabilities, but when parameterized with the number of taxa, GNAP is W[1]-hard and therefore unlikely to admit an FPT algorithm.

Consideration of Extinction Times. Efforts in another direction were made to consider that species could have differing times, after which they will die out if they have not been protected. In this extension of Budgeted NAP, denoted Time Sensitive Maximization of Phylogenetic Diversity (Time-PD), each taxon has an associated *rescue length* (the amount of time it takes to save them) and also

an *extinction time* (the time after which the taxon can not be saved anymore). Additionally, we know when preservation teams are able to work on saving taxa. Thus, to ensure that a set of taxa can be saved with the available resources, it is not enough to guarantee that their total cost is below a certain threshold. One also needs to ensure that there is a schedule for the involved intervention teams under which each taxon is saved before its moment of extinction. The first steps to addressing this issue were made by defining and analyzing Time-PD and s-Time-PD [JS24], which is a part of this thesis. s-Time-PD and Time-PD are related versions of the problem which differ only in the specification of whether operating teams may cooperate on saving a specific taxon or not.

Time-PD and s-Time-PD have much in common with machine scheduling problems, insofar as we may think of the taxa as corresponding to jobs with a certain due date and the teams corresponding to machines. One may think of Time-PD and s-Time-PD as machine scheduling problems, in which the objective to be maximized is the phylogenetic diversity of the set of completed tasks—which are the saved taxa.

We show that both problems, Time-PD and s-Time-PD, are FPT when parameterized with the threshold of diversity D. Further, we present an FPT-algorithm for Time-PD with respect to the acceptable loss of phylogenetic diversity. Such an algorithm is unlikely to exist for s-Time-PD, as this problem is NP-hard even if every taxon needs to be saved.

Considering Ecological Dependencies. Moulton et al. [MSS07] were the first to consider a formal problem in which the survival of some taxa may also depend on the survival of other taxa and modeled the Optimizing PD with Dependencies (PDD). Here, the input additionally contains a directed acyclic graph (DAG)[2] $\mathcal{F}$ with vertex set X where an arc uv is present if the existence of taxa u provides all the necessary foundations for the existence of taxon v. In other words, $\mathcal{F}$ models ecological dependencies between taxa. Now, a taxon v may survive only if (i) it does not depend on other taxa at all, that is, it has no incoming arcs, or (ii) at least one taxon u survives such that $\mathcal{F}$ contains the arc uv. The most widespread interpretation of such ecological dependency networks are food-webs where the arc uv means that taxon v feeds on taxon u.[3] A subset of taxa X where every vertex fulfills (i) or (ii) is called *viable*. The task in PDD is to select a *viable* set of k taxa that achieves

[2] A DAG is a directed graph without loops in which there is no path from vertex v to vertex u if there is a path from u to v.

[3] We remark that previous works [MSS07, FSW11] consider a reversed interpretation of the arcs. We define the order in such a way that a source of the network also corresponds to a source of the ecosystem.

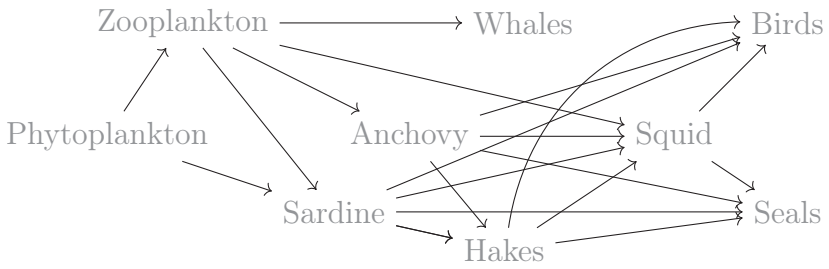


Figure 1.3: Here, a model of the food-web of the Benguela ecosystem is depicted [PLS14]. Phytoplankton is the only source. If whales are to be saved, then zooplankton and phytoplankton also need to be saved.

maximal phylogenetic diversity. An example of a food-web is given in Figure 1.3.

We consider PDD and the special case S-PDD, where the phylogenetic tree is a star. We show that PDD is FPT when parameterized by the size of the solution k plus the height of the phylogenetic tree and therefore with respect to the desired diversity D. We further examine PDD with respect to parameters analyzing the structure of the food-web. Herein, we show that S-PDD is FPT when parameterized by the distance to cluster or the treewidth of the food-web. With the distance to cluster it is measures how few vertices could be removed to result in a cluster graph and with the treewidth, one can state how similar a graph is to a tree. The latter disproves a conjecture of Faller et al. [FSW11, Conjecture 4.2] stating that S-PDD is NP-hard even when the food-web is a tree, unless P = NP.

1.3 Phylogenetic Networks

Phylogenetic trees are the familiar model to represent the inheritance of a set of taxa. Even though such a representation may models the biological ground truth acceptably well for many applications, it is not flawless. In the interaction between taxa it is possible that reticulation events occur. These events include hybridization, horizontal gene transfer and other forms of genetic recombination [HB06]. With hybridization, evolutionists describe the crossing of two groups of species that have developed different features over time [Ste59, SPSS23]. Bacteria and fungi use the mechanism of horizontal gene transfer to pass DNA from a donor cell to a receiver cell in order to make themselves more resistant [NBP19].

These reticulation events can not be represented in a phylogenetic tree. Biologists have therefore generalized phylogenetic trees and have introduced the concept of *phylogenetic networks*. A phylogenetic network is a directed acyclic graph (DAG)

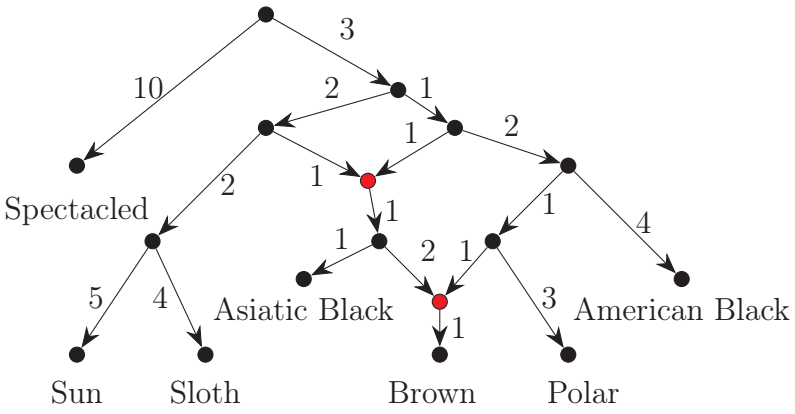


Figure 1.4: This figure depicts a likely heritage of several bears in a weighted phylogenetic network [KLB+17]. The two reticulations are depicted in red.

in which there is exactly one vertex with an in-degree of 0 and each vertex with an in-degree bigger than 1—which are called *reticulations*—has an out-degree of 1. As the name suggests, in the reticulations, reticulation events can be represented. Similarly as in phylogenetic trees, vertices with an out-degree of 0 represent (a subset of) present day taxa and the other vertices represent assumed extinct ancestors. Weights on the edges of a phylogenetic network mark a distance between the two endpoints and can stand for a number of features or an estimated evolutionary distance. We note that this definition of phylogenetic networks is sometimes referred to as phylogenetic *reticulate* networks or *explicit* phylogenetic networks and that other types of phylogenetic networks exist. An overview of different types of phylogenetic networks can be found in [HB06, HRS10]. An example of a phylogenetic reticulate network is given in Figure 1.4.

Phylogenetic networks generalize phylogenetic trees. Therefore, the question of what measure generalizes phylogenetic diversity on phylogenetic networks naturally arises. First concepts have been introduced for so-called phylogenetic split systems [VMM+14, CKvHM16] and later also for explicit phylogenetic reticulate networks [WF18, BSW22].

Arguably the most easy-to-understand measure of phylogenetic diversity of a set of taxa A in a phylogenetic network $\mathcal{N}$ is *all-paths phylogenetic diversity*, denoted with AP-PD$_{\mathcal{N}}(A)$, which was first defined as "phylogenetic subnet diversity" in [WF18]. In all-paths phylogenetic diversity, one simply adds the weight of all the edges uv for which there is a path from v to a taxon in A.

In all-paths phylogenetic diversity it is therefore assumed that taxa which result

from reticulation events have all the features of the parents. A more realistic assumption is that all parents contribute some features at some inheritance proportion. As a consequence, to compute *network phylogenetic diversity*, denoted with Net-PD$_{\mathcal{N}}(A)$, of a set of taxa A in a phylogenetic network $\mathcal{N}$ we need to know the inheritance proportion $p(e)$ of every edge e incoming at some reticulation v. Then, if a taxon below v is selected, the features above e only survive with a probability $p(e)$—if not covered by a better path. Observe that if all the inheritance proportions are 1, then the network phylogenetic diversity takes the same value as the all-paths phylogenetic diversity.

Corresponding to these two measures, decision problems Max-All-Paths-PD (MapPD) and Max-Net-PD were introduced [BSW22] in which we are given a phylogenetic network, two integers k and D, and inheritance proportions in the case of the latter problem. It is asked whether a subset S of taxa exists such that S has a size of at most k and AP-PD$_{\mathcal{N}}(S) \geq D$ or Net-PD$_{\mathcal{N}}(S) \geq D$, respectively.

Based on a known hardness reduction for MapPD [BSW22], we establish the W[2]-hardness with respect to the size k of a solution. We then show that MapPD is, however, FPT when parameterized by the threshold of diversity D or by the acceptable loss of diversity $\overline{D}$. We further show that MapPD is FPT with respect to the number of reticulations and with respect to the number of edges incoming in reticulations and admits a kernelization of polynomial size.

As Max-Net-PD generalizes MapPD, the hardness-results for MapPD also hold for Max-Net-PD. We moreover show that Max-Net-PD remains NP-hard in instances in which the phylogenetic network has a level of 1. The level measures the tree-likeness of a phylogenetic network.

1.4 Structure of this Thesis

In the following chapter, we formally define phylogenetic diversity in the considered variants and the problems regarded in this thesis. Furthermore, we give an overview of further relevant definitions such as terms of computational complexity. At the end of Chapter 2, we show parameterized complexity results for Multiple-Choice Knapsack and Penalty Sum, two problems that are unrelated to phylogenetic diversity but are relevant for results in the following chapters.

In Chapter 3, we consider Generalized Noah's Ark Problem with its special cases; in Chapter 4, we consider Time-PD and s-Time-PD; and in Chapter 5, we consider Optimizing PD with Dependencies with its special cases. At the beginning of each of these chapters, we give an overview of the known results and

then we examine the respective problem within the framework of parameterized complexity.

In Chapter 6, with Max-All-Paths-PD and Max-Net-PD, we examine two problems that are based on measures of phylogenetic diversity in phylogenetic networks. As in the chapters before, we first present known results. Afterward, we show parameterized complexity results for Max-All-Paths-PD before we show that Max-Net-PD is NP-hard even on networks of level 1.

Finally, in Chapter 7 we discuss open problems and future research ideas.

Chapter 2

Preliminaries

In this chapter, we give the fundamental definitions that we use throughout this thesis. We start with some mathematical notation, then formally define phylogenetic diversity and Maximize Phylogenetic Diversity. Afterward, we define notions of classic and parameterized complexity. Finally, in Sections 2.5 and 2.6, we consider two problems, Multiple-Choice Knapsack and Penalty Sum, to which we refer later in the thesis. These problems are not about the maximization of phylogenetic diversity but are very useful for results later in the thesis.

2.1 Mathematical Notation

For an integer n, let $[n]$ denote the set $\{1, \ldots, n\}$ and let $[n]_0$ denote the set $\{0\} \cup [n]$. Define $\mathbb{R}_{[0,1]} := \{x \in \mathbb{R} \mid 0 \leq x \leq 1\}$ and $\mathbb{R}_{(0,1)} := \mathbb{R}_{[0,1]} \setminus \{0, 1\}$. A *partition of a set* N is a family of pairwise disjoint sets $\{N_1, \ldots, N_m\}$ such that $\bigcup_{i=1}^{m} N_i = N$.

We extend functions $f : A \to B$, where B is a family of sets, to handle subsets $A' \subseteq A$ of the domain by defining $f(A') := \bigcup_{a \in A'} f(a)$. Functions $f : A \to \mathbb{N}$ are extended to subsets $A' \subseteq A$ of the domain by defining $f_\Sigma(A') := \sum_{a \in A'} f(a)$.

Throughout the thesis, we will use both natural and binary logarithms. We will write $\ln x$ to denote the natural logarithm of x (to the base e), and $\log_i x$ to denote the logarithm of x to the base $i \in \mathbb{N}$. If i is not further specified, then we use $i = 2$.

The *encoding length* of a number is the number z of bits necessary to encode z in binary. Let $\mathbf{0}$ denote the multidimensional-dimensional zero. For a vector $\mathbf{p}$ of numbers, we use the following operations. We let $\mathbf{p}_{(j)+z}$ denote the vector $\mathbf{p}$ in which in position i, value z is added. We write $\mathbf{p} \leq \mathbf{q}$ if $\mathbf{p}$ and $\mathbf{q}$ have the same dimension d and $p_i \leq q_i$ for every $i \in [d]$.

2.1.1 Graph-Theoretic Notation

We define the graph-theoretic notion used throughout the work. Standard monographs provide a deeper view [Wes00, Die12].

For a set V and a set $E \subseteq \{uv \mid u, v \in V\}$, 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 write uv or (u, v) for a *(directed) edge* from u to v, and $\{u, v\}$ for an *undirected edge* between u and v. For any graph G, we write $V(G)$ and $E(G)$, respectively, to denote the set of vertices and edges of G. If E is a family of undirected edges, then $G = (V, E)$ is called an *undirected graph*. For a directed graph $G = (V, E)$, the directed graph $G' := (V, E')$ with $E' := \{\{u, v\} \mid uv \in E\}$ is called the *underlying undirected graph*.

An edge $e = uv$ or $e = \{u, v\}$ is *incident* with u and v. Further, the edge uv is *incoming* at u and *outgoing* from v. We say two vertices are *adjacent* or *neighbors* if they are incident with the same edge, and similarly we say two edges are *adjacent* if they are incident with the same vertex.

The *degree of a vertex* v is the number of edges that are incident with v. Similarly, the *in-degree* (and *out-degree*, respectively) of v is the number of incoming (outgoing) edges of v.

For a graph G and a set of vertices $V' \subseteq V(G)$, *the subgraph of* G *induced by* V' is $G[V'] := (V', E')$, where $E' := \{e = uv \in E(G) \mid u, v \in V'\}$. Moreover, with $G - V' := G[V \setminus V']$ with $V' \subseteq V$, we denote the graph obtained from G by removing V' and its incident edges. For an edge set $E' \subseteq E$, we define $G - E'$ as the graph $(V, E \setminus E')$.

We say that there is *a path of length* p *from* u *to* w if $v = w$ and $p = 0$ or there is an edge uv or $\{u, v\}$ such that there is a path of length $p - 1$ from v to w. We sometimes omit the length of the path. Two vertices u and v are *connected* if there is a path from u to v or from v to u. An undirected graph G is *connected* if the vertices in $V(G)$ are pairwise connected. For an undirected graph G and a non-empty set of vertices $V' \subseteq V(G)$, the subgraph of G induced by V' is a *connected component* if $G[V']$ is connected, and vertices u and v are not connected for each $v \in V(G) \setminus V'$ and each $u \in V'$. An undirected graph G is *biconnected* if $G - \{v\}$ is connected for every vertex $v \in V(G)$.

A *topological order* induced by a given directed graph $G = (V, E)$ is a bijective mapping $f : V \to [|V|]$ such that $f(u) \leq f(v)$ for every edge $uv \in E$.

The *complement graph* of an undirected graph is obtained by replacing edges with non-edges and vice versa.

Other Graph Classes. An undirected graph G is *cyclic* if there are vertices $u, v, w \in V(G)$ such that neighbors u and w are neighbors of v and u and w are connected in $G - \{v\}$. An undirected graph is not cyclic is *acyclic*. A self-loop is an edge uu for some vertex u. A *directed, acyclic graph* (DAG) is a directed graph that does not contain self-loops and we can conclude $u = v$ if there is a path from u to v and from v to u.

An undirected graph $G = (V, E)$ is *bipartite* if there is a partition $\{V_1, V_2\}$ of the vertex set V such that $e \cap V_1$ and $e \cap V_2$ are both non-empty for each edge $e \in E$. In other words, each edge is between V_1 and V_2. The vertex sets V_1 and V_2 are a *vertex bipartition*.

An undirected graph is a *cluster graph* if the existence of edges $\{u, v\}$ and $\{v, w\}$ implies the existence of the edge $\{u, w\}$. Equivalently, a graph is a cluster graph if every connected component is a clique. An undirected graph is a *co-cluster graph* if its complement graph is a cluster graph. In other words, a graph is a co-cluster graph if its vertex set can be partitioned into independent sets such that each pair of vertices from different independent sets is adjacent.

For a graph class Π defined exclusively on undirected graphs, we say that a directed graph is of class Π if the underlying undirected graph is of class Π.

Trees. A *rooted tree T with root ρ* is a directed, acyclic graph with $\rho \in V(T)$ where 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 out-degree of v is 0. We refer to the non-leaf vertices of a tree as the *internal vertices*. In a rooted tree, the *height of a vertex v* is the length of the (only) path from the root ρ to v for each vertex v. The *height* height_T *of a rooted tree T* is the maximal height of one of the vertices of T. A *star* is a tree with a height of 1.

For an edge uv of a rooted tree, we call *u the parent of v* and *v a child of u*. For two edges e and $\hat{e}$ incident with the same vertex v, we say $\hat{e}$ is the *parent-edge* of e if $\hat{e} = uv$ and $e = vw$. If $e = vu$ and $\hat{e} = vw$, we say $\hat{e}$ is a *sibling-edge* of e. For a vertex v with parent u, the *subtree T_v rooted at v* is the connected component containing v in $T - \{u\}$. In the special case that ρ is the root of T, we define $T_\rho := T$. For a vertex v with children $w_1, \ldots, w_t$, the *i-partial subtree $T_{v,i}$ rooted at v* for $i \in [t]$ is the connected component containing v in $T_v - \{w_{i+1}, \ldots, w_t\}$.

For an undirected graph G and a set of vertices $V' \subseteq V(G)$, a *spanning tree of V'* is a tree T with $V' \subseteq V(T) \subseteq V(G)$ and $E(T) \subseteq E(G)$. The *size of a spanning tree* is the number of vertices. A *minimum spanning tree of V'* is a spanning tree of V' of minimum size. For a tree $T = (V, E)$ and a vertex set $V' \subseteq V$, the *minimum spanning tree of V'* is denoted by $T\langle V' \rangle$.

Graph Parameters. For an undirected graph $G = (V, E)$ and a graph class Π the *distance to* Π of G is the smallest number d such that there exists a set $Y \subseteq V$ of d such that $G - Y$ is of class Π. The set Y in this definition is called a *modulator* to Π.

The *max leaf number* of an undirected graph $G = (V, E)$ is the maximum number of leaves a spanning tree of V has.

Definition 2.1 (Tree Decomposition)**.** A *tree decomposition* of a directed graph G is a rooted tree T and a mapping which assigns each node $t \in T$ a subset $Q_t \subseteq V(G)$ such that

(a) every vertex of G is contained in some bag,

(b) for every edge $e = \{u, v\}$, there is some bag containing u and v, and

(c) the nodes of T whose bags contain any particular vertex v are connected.

A tree decomposition is *nice* [CNP+22] if the bags of the root and all leaves of T are empty, and every non-leaf node t has one of the following types

> *Introduce vertex node*: t has only one child t' and $Q_{t'} = Q_t \setminus \{v\}$ for some vertex v which is said to *be introduced* at t;
>
> *Introduce edge node*: t has only one child t' and $Q_{t'} = Q_t$ and t is labeled with an edge $e \subseteq Q_t$ which is said to *be introduced* at t;
>
> *Forget node*: t has only one child t' and $Q_{t'} = Q_t \cup \{v\}$ for some vertex v;
>
> *Join node*: t has exactly two children t_1, and t_2 and $Q_t = Q_{t_1} = Q_{t_2}$.

Every edge of G must be introduced exactly once, and we may assume this happens as high in T as possible, i.e. we can introduce edges right before some of their endpoints is forgotten. For a node $t \in T$, we denote by $G_t = (V_t, E_t)$ the subgraph of G that contains exactly those vertices V_t and edges E_t and edges that are introduced at t or any descendant of t.

If we do not define introduce edge nodes, then all edges are introduced once both vertices are introduced. The *width* of a tree decomposition is the size of the biggest bag minus 1. The *treewidth* of an undirected graph G is the minimum width of a tree decomposition of G.

For a graph parameter κ defined exclusively on undirected graphs, we define κ on directed graphs with the size of κ in the underlying undirected graph.

2.2 Phylogenetics

For a given set X, a *phylogenetic X-tree* $\mathcal{T} = (V, E, \lambda)$ (in short, X-tree or phylogenetic tree) is a tree $T = (V, E)$ with an *edge-weight* function $\lambda : E \to \mathbb{N}_{>0}$ and a bijective labeling of the leaves with elements from X where all non-leaves in $\mathcal{T}$ have out-degree at least 2. The set X is a set of *taxa* (species). Because of the bijective labeling, we interchangeably use the words taxon and leaf. We write $\max_\lambda$ to denote the biggest edge weight in $\mathcal{T}$. An X-tree $\mathcal{T}$ is *ultrametric* if there is an integer p such that the sum of the weights of the edges of the path from the root ρ to x_i equals p for every leaf x_i.

If there is a directed path from u to v in $\mathcal{T}$ (including when $u = v$ or when v is a child of u), we say that u is an *ancestor* of v and v is a *descendant* of u. If in addition $u \neq v$, we say u is a *strict ancestor* of v and v a *strict descendant* of u. The sets of ancestors and descendants of v are denoted $\mathrm{anc}(v)$ and $\mathrm{desc}(v)$, respectively. The set of descendants of v which are in X are *offspring* $\mathrm{off}(v)$ of a vertex v. Moreover, we denote $\mathrm{off}(e) = \mathrm{off}(v)$ for an edge $e = uv \in E$.

Phylogenetic Diversity. Given a subset of taxa $A \subseteq X$, let $E_\mathcal{T}(A)$ denote the set of edges in $e \in E$ with $\mathrm{off}(e) \cap A \neq \emptyset$. The *phylogenetic diversity* $\mathrm{PD}_\mathcal{T}(A)$ of A is defined by

$$\mathrm{PD}_\mathcal{T}(A) := \sum_{e \in E_\mathcal{T}(A)} \lambda(e). \tag{2.1}$$

That is, $\mathrm{PD}_\mathcal{T}(A)$ is the total weight of all edges uv in $\mathcal{T}$ so that there is a path from v to a vertex in A. The phylogenetic diversity model assumes that features of interest appear along the edges of the tree with frequency proportional to the weight of that edge and that any feature belonging to one species is inherited by all its descendants. Thus, $\mathrm{PD}_\mathcal{T}(Z)$ corresponds to the expected number of distinct features appearing in all species in Z.

See Figure 2.1 for an example of the definition of phylogenetic diversity.

In Section 3.2.1 and Section 6.2.1, we present the generalizations of phylogenetic diversity to a) handle that species do not survive indefinite but at a given probability and b) phylogenetic networks, respectively.

Phylogenetic Networks. A *phylogenetic X-network* $\mathcal{N} = (V, E, \lambda)$ (in short, X-network or network) is a directed acyclic graph with an *edge-weight* function $\lambda : E \to \mathbb{N}_{>0}$ and a single vertex with an in-degree of 0 (the *root* ρ), in which the vertices with an out-degree of 0 (the *leaves*) have an in-degree of 1 and are

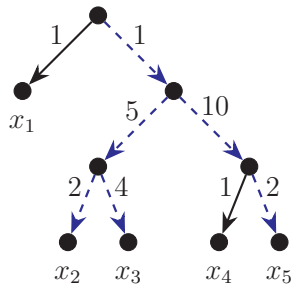


Figure 2.1: An example of a phylogenetic X-tree $\mathcal{T}$ with taxa $X = \{x_1, x_2, x_3, x_4, x_5\}$. The set $A = \{x_2, x_3, x_5\}$ has a phylogenetic diversity of $\mathrm{PD}_{\mathcal{T}}(A) = 1{+}5{+}10{+}2{+}4{+}2 = 24$. The set of edges $E_{\mathcal{T}}(A)$ is blue and dashed.

bijectively labeled with elements from a set X, and such that all vertices either have an in-degree of at most 1 or an out-degree at most 1. The vertices with in-degree at least 2 and out-degree 1 are called *reticulations*; the other non-leaf vertices are called *tree vertices*. Every edge incoming at a reticulation is a *reticulation edge*. A network is *binary* if the root has an out-degree of 2 and every other non-leaf vertex has a degree of 3.

The *number of reticulations* of a network $\mathcal{N}$ is denoted with $\mathrm{ret}_{\mathcal{N}}$. The *number of reticulation-edges* $\text{e-ret}_{\mathcal{N}}$ of a network $\mathcal{N}$ is the number of edges that need to be removed such that $\mathcal{N}$ is a tree. We note that this is not state what the number of the reticulation edges is. Some authors refer to the number of reticulation-edges as the reticulation number of $\mathcal{N}$. If $\mathcal{N}$ is binary, then the number of reticulations is exactly the number of reticulation-edges. The *level* of a network $\mathcal{N}$ is the maximum reticulation number of a subgraph $\mathcal{N}[V']$ for some $V' \subseteq V(\mathcal{N})$ where we require that the underlying undirected graph of $\mathcal{N}[V']$ is biconnected.

2.3 Classic and Parameterized Complexity

In this section, we provide the necessary background in complexity theory, both for classical concepts of NP-hardness and NP-completeness, and for parameterized complexity. Readers are referred to monographs for classic computational complexity [GJ79, Pap07, AB09], and to monographs for parameterized complexity [FG06, Nie06, DF13, CFK$^+$15] for more detailed introductions.

2.3.1 Classic Computational Complexity

Within the field of classic computational complexity, we deal with *computational problems* and the question of how many computational resources are required to find an answer to the raised question. In this thesis, we only consider *decision problems* Π (problems for short). The closely related *optimization problems* can usually be solved with only a small overhead, if an algorithm is given that solves the decision variant. We consider the following formal definitions.

Definition 2.2 (Languages, Decision Problems and Instances)**.**

(a) A *language* $L \subseteq \Sigma^*$ is a set of (finitely long) strings over a finite *alphabet* Σ.

(b) The decision problem associated with a language L is to determine if $x \in L$ for any given *instance* $x \in \Sigma^*$.

(c) An instance $x \in \Sigma^*$ is a `yes`-*instance* if $x \in L$ and a `no`-*instance* if $x \notin L$.

An *algorithm* is a defined order of computational steps to solve a computational problem. An algorithm $\mathcal{A}$ solves a decision problem Π if, after a finite number of computation steps, $\mathcal{A}$ correctly determines whether an instance is a `yes`-instance or a `no`-instance. If such an algorithm exists, then Π is *decidable*. In this thesis, we only consider decidable decision problems.

The complexity measures of algorithms are the computational resources that are required to let the algorithm run on any instance, usually in correspondence with the *encoding length* $|x|$ of the instance x. The most frequently used complexity measures are *time* and *space*. With time, we refer to the number of steps the computation needs, and with space we refer to the required memory. Throughout this thesis, we only consider the running time of algorithms. The complexity of a decision problem Π is given by the fastest, usually unknown algorithm solving Π.

In our running time analyses, we assume a unit-cost RAM model where arithmetic addition and multiplication for numbers of any length have a constant running time. This model is unrealistically strong but avoids that we have to add factors to the running time which are not all too interesting from a theoretical point of view. We will, in order to have more clarity, recall this fact in the running time analyses of algorithms in which we operate with numbers that are potentially bigger than $\mathcal{O}(\log(|\mathcal{I}|))$, where $|\mathcal{I}|$ is the size of the entire instance.

There are several *complexity classes* for decision problems, of which `P` and `NP` are the most prominent.

Definition 2.3 (The classes P and NP)**.**

(a) A decision problem is in P if a *deterministic* Turing machine exists which, for each given instance $x \in \Sigma^*$, determines whether x is a yes-instance within poly($|x|$) time.

(b) A decision problem is in NP if a *non-deterministic* Turing machine exists which, for each given instance $x \in \Sigma^*$, accepts x within poly($|x|$) time.

A full definition of non-determinism is beyond the scope of this work. Informally, in a non-deterministic Turing machine some configurations may have several successor configurations. A Turing machine *accepts* an input x if it is possible to reach an accepting configuration from the starting configuration.

Every problem in P is also in NP, but it is assumed that certain problems are in NP but not in P, and, therefore, P $\neq$ NP. In other words, it is believed that there are problems in NP that are not deterministically solvable in polynomial time—the NP-complete problems. To mathematically define these, we introduce the concept of reductions, to show that a problem is at least as hard as another. For two decision problems $\Pi_1, \Pi_2 \subseteq \Sigma^*$, a *reduction* from Π_1 to Π_2 is an algorithm—a computable function $f : \Sigma^* \to \Sigma^*$—which takes instances of Π_1 and returns instances of Π_2, and we require that $x \in \Sigma^*$ is a yes-instance of Π_1 if and only if $f(x) \in \Sigma^*$ is a yes-instance of Π_2. The instances x and $f(x)$ are then called *equivalent*. A reduction f is a *polynomial-time reduction* if $f(x)$ is computed within poly($|x|$) time. A decision problem Π is NP-*hard* if there is a polynomial-time reduction from Ψ to Π for every decision problem Ψ in NP. A decision problem Π is NP-complete if Π is NP-hard and in NP. If there was an NP-complete problem which is also in P, then all problems in NP could be solved deterministically in polynomial time. As discussed earlier, it is not believed that such a decision problem exists. In Section 2.4, a small list of NP-complete problems is given.

A function $f : \mathbb{N} \to \mathbb{N}$ is in $\mathcal{O}(g(n))$ for another function $g : \mathbb{N} \to \mathbb{N}$, if there are integers c and m such that $f(n) \leq c \cdot g(n)$ for each integer $n \geq m$. Similarly, function $f : \mathbb{N} \to \mathbb{N}$ is in $\mathcal{O}^*(g(n))$ for another function $g : \mathbb{N} \to \mathbb{N}$, if there is an integer m such that $f(n) \leq g(n) \cdot \text{poly}(n)$ for each integer $n \geq m$. In other words, if $f(n) \in \mathcal{O}(g(n))$ then g grows at least as fast as f. We use in the $\mathcal{O}$ and the $\mathcal{O}^*$-notation to describe the running times of algorithms and in the $\mathcal{O}^*$-notation we omit factors polynomial in the input size.

2.3.2 Parameterized Complexity

In this section, we give a brief overview of formal definitions of parameterized complexity.

Fixed-Parameter Tractability. Here, we first define what a parameterized problem is before we define the complexity classes FPT and XP. The formal definition for parameterized languages and parameterized decision problems are similar to the definition of languages and decision problems given in Definition 2.2.

Definition 2.4 (Parameterized Languages and Parameterized Problems)**.**

(a) A *parameterized language* $L \subseteq \Sigma^* \times \mathbb{N}_0$ is a set of tuples (x, k), where x is a (finitely long) string over a finite alphabet Σ and k is an integer.

(b) We call x the *input*, k the *parameter*, and (x, k) the *instance*.

(c) The *parameterized decision problem* associated with a parameterized language L is to determine if $(x, k) \in L$ for any given *instance* $(x, k) \in \Sigma^* \times \mathbb{N}_0$.

(d) An instance $(x, k) \in \Sigma^* \times \mathbb{N}_0$ is a `yes`-*instance* if $(x, k) \in L$ and a `no`-*instance* if $(x, k) \notin L$.

The *encoding length* of an instance (x, k) is denoted by $|(x, k)|$. We usually simply use the term problem, when the context makes it clear that it is a parameterized decision problem. Now, with the definition of parameterized decision problems at hand, we define the two most relevant classes of parameterized decision problems, FPT and XP.

Definition 2.5 (Fixed-Parameter Tractability)**.**

(a) An algorithm $\mathcal{A}$ for a parameterized decision problem Π is *fixed-parameter* if for every instance $(x, k) \in \Sigma^* \times \mathbb{N}_0$, within $f(k) \cdot \text{poly}(|x|)$ (deterministic) time the algorithm $\mathcal{A}$ correctly determines whether (x, k) is a `yes`- or a `no`-instance of Π. Here, the function f can be any computable function that only depends on k. We write in short that $\mathcal{A}$ is an FPT-algorithm.

(b) A parameterized decision problem is *fixed-parameter tractable* (FPT) if it can be solved by a fixed-parameter algorithm.

(c) FPT is the complexity class of all parameterized decision problems that are fixed-parameter tractable.

An algorithm has a *pseudo-polynomial running time* if it has a running time polynomial in the input size when all numbers are encoded in unary, but the running time is not polynomial in the input size when all numbers are encoded in binary. A parameterized decision problem is *polynomial fixed-parameter tractable* (PFPT) if the function f in Definition 2.5(a) is a polynomial.

Definition 2.6 (Slice-wise Polynomial (XP))**.**

(a) An algorithm $\mathcal{A}$ for a parameterized decision problem Π is *slice-wise polynomial* if for every instance $(x, k) \in \Sigma^* \times \mathbb{N}_0$, within $\text{poly}(|(x, k)|)^{g(k)}$ (deterministic) time the algorithm $\mathcal{A}$ correctly determines whether (x, k) is a yes- or a no-instance of Π. Here, the function g can be any computable function that only depends on k. We write in short that $\mathcal{A}$ is an XP-algorithm.

(b) A parameterized decision problem is slice-wise polynomial (XP) if it can be solved by an XP-algorithm.

(c) XP is the complexity class of all parameterized decision problems that are slice-wise polynomial.

We observe that the definition of FPT is stricter than the definition of XP. Consequently, any FPT-algorithm is an XP-algorithm, but the converse is not given. We conclude that $\text{FPT} \subseteq \text{XP}$.

We observe further that problems which are XP can be solved in polynomial time if we require the parameter to be a constant. In order to show that a parameterized decision problem Π is not XP, we therefore show that Π is NP-hard even for constant values of the parameter. Next, we define how we exclude that a parameterized decision problem is FPT under some complexity-theoretic assumptions.

The W-Hierarchy. Of course, we do not assume that all parameterized decision problems are FPT. Proving that a parameterized decision problem Π is NP-hard for a constant size of the parameter shows that Π is not in XP (assuming $\text{P} \neq \text{NP}$). Because $\text{FPT} \subseteq \text{XP}$ we can also conclude that Π is not in FPT in that case. But then automatically the question arises of how to provide evidence that a problem in XP is not in FPT.

To answer this question, Downey and Fellows defined the *W-Hierarchy* [DF95a, DF95b, DF13]. This hierarchy is defined with parameterized reductions which generalize reductions.

Definition 2.7 (Parameterized Reductions)**.** For two parameterized decision problems Π_1 and Π_2 and two computable functions $f, g : \mathbb{N} \to \mathbb{N}$, a *parameterized reduction* from Π_1 to Π_2 is an algorithm $\mathcal{A}$ which takes instances $(x, k) \in \Sigma^* \times \mathbb{N}$ of Π_1 and returns instances $(x', k') \in \Sigma^* \times \mathbb{N}$ of Π_2 such that

(a) (x, k) is a `yes`-instance of Π_1 if and only if (x', k') is a `yes`-instance of Π_2, and

(b) the computation of $\mathcal{A}$ takes $f(k) \cdot \text{poly}(|x|)$ time, and

(c) $k' \leq g(k)$.

Essential for the W-hierarchy are the following problems. In Weighted Circuit Satisfiability over $\mathcal{C}_{t,d}$, we are given a logic circuit with a depth of d and a weft of t. It is asked whether there is an input that satisfies the circuit. The depth of a circuit is the maximum number of nodes on a path from an input variable to the output and the weft of a circuit is the maximum number of nodes with in-degree greater than 2 on a path from an input variable to the output. It is evident that Weighted Circuit Satisfiability over $\mathcal{C}_{t,d}$ generalizes the famous Satisfiability problem. The parameter we consider in the definition of the W-hierarchy is the weft of the circuit.

Definition 2.8 (The W-Hierarchy)**.**

(a) The complexity class $\mathtt{W}[i]$ for an integer i consists of the parameterized decision problems Π for which there is a parameterized reduction from Π to Weighted Circuit Satisfiability over $\mathcal{C}_{i,d}$ for some $d \in \mathbb{N}$.

(b) A parameterized decision problem Π is $\mathtt{W}[i]$-*hard* if there is a parameterized reduction from Ψ to Π for every parameterized decision problem Ψ in $\mathtt{W}[i]$.

(c) A parameterized decision problem Π is $\mathtt{W}[i]$-*complete* if Π is $\mathtt{W}[i]$-hard and in $\mathtt{W}[i]$.

It is know that $\mathtt{FPT} \subseteq \mathtt{W}[1] \subseteq \mathtt{W}[2] \subseteq \cdots \subseteq \mathtt{XP}$. Further, it is widely believed that $\mathtt{FPT} \subsetneq \mathtt{W}[1] \subsetneq \mathtt{W}[2] \subsetneq \cdots \subsetneq \mathtt{XP}$, but like $\mathtt{P} \neq \mathtt{NP}$, this claim is not yet proven.

Kernelization. For parameterized decision problems, the concept of kernelization provides a means of measuring how much the preprocessing decreases the size of the instance.

Definition 2.9 (Kernelization Algorithms)**.** A *kernelization algorithm* (also kernelization, kernel or problem kernel) for a parameterized decision problem Π is an algorithm $\mathcal{A}$ which takes instances $(x, k) \in \Sigma^* \times \mathbb{N}$ of Π and returns instances $(x', k') \in \Sigma^* \times \mathbb{N}$ of Π such that

(a) the computation of $\mathcal{A}$ takes time polynomial in $|(x,k)|$, and

(b) (x,k) is a `yes`-instance of Π if and only if (x',k') is a `yes`-instance of Π, and

(c) $|x'| + k' \leq g(k)$ for some computable function $g : \mathbb{N} \to \mathbb{N}$.

Kernelization algorithms are usually explained in several steps. These steps are called reduction rules. A *(data) reduction rule* for a parameterized decision problem Π is an algorithm that given an instance (x,k) of Π returns an instance (x',k') of Π. A reduction rule is *correct* if (x,k) is a `yes`-instance of Π if and only if (x',k') is a `yes`-instance of Π. We say that a reduction rule has been *exhaustively applied* on an instance if an application does not change the instance.

Kernelizations have a close connection to the class `FPT`, as we see in this theorem.

Theorem 2.1 ([CFK+15, DF13])**.** *A parameterized decision problem* Π *admits a problem kernel if and only if* Π *is* `FPT`*.*

A *polynomial kernelization (algorithm)* is a kernelization where the function g in Definition 2.9 is a polynomial. To disprove that a parameterized decision problem admits a polynomial kernelization, we use one of the following two concepts. All these concepts are based on the assumption that `NP` $\not\subseteq$ `coNP/poly`. Even though it is not proven yet, it is widely believed that `NP` $\not\subseteq$ `coNP/poly`. In particular, if `NP` $\subseteq$ `coNP/poly`, then the polynomial hierarchy would collapse at the third level [Yap83].

Definition 2.10 (Polynomial Parameter Transformation)**.** For two parameterized decision problems Π_1 and Π_2, a *polynomial parameter transformation (PPT)* is an algorithm $\mathcal{A}$ which maps instances $(x,k) \in \Sigma^* \times \mathbb{N}$ of Π_1 to instances $(x',k') \in \Sigma^* \times \mathbb{N}$ of Π_2 such that

(a) the computation of $\mathcal{A}$ takes time polynomial in $|(x,k)|$, and

(b) (x,k) is a `yes`-instance of Π if and only if (x',k') is a `yes`-instance of Π, and

(c) $k' \leq p(k)$ for some polynomial $p : \mathbb{N} \to \mathbb{N}$.

If there is a PPT from Π_1 to Π_2 for decision problems Π_1 and Π_2 with $\Pi_1 \in$ `NP` and Π_2 is `NP`-hard and admits a kernel of polynomial size, then also Π_1 admits a kernel of polynomial size. We conclude that if there is a PPT from Π_1 to Π_2 and Π_1 does not admit a polynomial kernelization, assuming `NP` $\not\subseteq$ `coNP/poly`, then also Π_2 does not admit a polynomial kernelization, assuming `NP` $\not\subseteq$ `coNP/poly`.

Next, we define cross-compositions, another technique for excluding polynomial kernelizations. For this definition, we require polynomial equivalence relations.

Definition 2.11 (Polynomial Equivalence Relations)**.** A *polynomial equivalence relation* is an equivalence relation R on Σ^* for which the following holds.

(a) There is an algorithm that for given strings $x, y \in \Sigma^*$ can check if $x \sim_R y$ in $\text{poly}(|x| + |y|)$ time, and

(b) for any finite set $S \subseteq \Sigma^*$ there are at most $\text{poly}(\max_{x \in S} |x|)$ equivalence classes with regard to the relation R.

Definition 2.12 (Cross-Compositions, [BJK14, CFK$^+$15])**.** Let a decision problems $\Pi_1 \subseteq \Sigma^*$, a parameterized decision problems $\Pi_2 \subseteq \Sigma^* \times \mathbb{N}$, and a polynomial equivalence relation R, and an integer t be given. A *cross-composition* from Π_1 into Π_2 is an algorithm $\mathcal{A}$ which takes 2^t instances $x_1, \ldots, x_{2^t}$ of Π_1 that are pairwise equivalent with respect to R and returns an instance (x', k') of Π_2 such that

(a) the computation of $\mathcal{A}$ takes time polynomial in $\sum_{i=1}^{2^t} |x_i|$, and

(b) x_i is a `yes`-instance of Π_1 for some $i \in [2^t]$ if and only if (x', k') is a `yes`-instance of Π_2, and

(c) $k' \leq p(t + \max_{i=1}^{2^t} |x_i|)$ for some polynomial $p : \mathbb{N} \to \mathbb{N}$.

Exponential Time Hypothesis (ETH). The *Exponential Time Hypothesis* (`ETH`) and the *Strong Exponential Time Hypothesis* (`SETH`) are important conjectures in the field of parameterized complexity and have been proposed by Impagliazzo, Paturi, and Zane [IPZ01]. Relevant to the definition of `ETH` and `SETH` is q-SAT, a special case of Satisfiability. In q-SAT, we are given a formula ψ with n variables that is a conjunction of disjunctions of at most q literals. It is asked whether there is an assignment of the variables that fulfills ψ. The problem q-SAT is `NP`-hard for each $q \geq 3$ [Kar72].

Now, for any q let C_q be the set of numbers such that q-SAT can be solved in $\mathcal{O}(2^{c \cdot n})$ time. As q-SAT is a special case of $q+1$-SAT, we conclude $C_q \subseteq C_{q+1}$. Let δ_q be the infimum of C_q. The formal definitions of `ETH` and `SETH` are as follows.

Definition 2.13 (`ETH` and `SETH`)**.**

(a) The Exponential Time Hypothesis (`ETH`) states $\delta_3 > 0$.

(b) The Strong Exponential Time Hypothesis (`SETH`) states $\lim_{q \to \infty} \delta_q = 1$.

It is widely believed that `ETH` is correct while there are some doubts for the correctness of `SETH`. However, we again do not have a prove of the correctness or incorrectness of either `ETH` or `SETH`, yet.

Color Coding. Here, we briefly want to define some mathematical objects that are relevant for the technique of color coding which we use several times throughout this thesis. For an in-depth treatment of color coding, we refer the reader to [CFK+15, Sec. 5.2 and 5.6] and [AYZ95]. The technique of color-coding is traditionally used for randomized algorithms. Over the years, concepts for derandomization have been developed. Derandomization is necessary to meet the formal definition of (deterministic) FPT-algorithms. However, we want to mention that randomized algorithms with a very low error probability usually have faster running times.

Definition 2.14 (Perfect Hash Families)**.** For integers n and k, an (n, k)-*perfect hash family* $\mathcal{H}$ is a family of functions $f : [n] \to [k]$ such that for every subset Z of $[n]$ of size k, some $f \in \mathcal{H}$ exists that is injective when restricted to Z.

We will also resort to another data structure relevant for color coding.

Definition 2.15 (Universal Sets)**.** For integers n and k, an (n, k)-*universal set* is a family $\mathcal{U}$ of subsets of $[n]$ such that for any $S \subseteq [n]$ of size k, $\{A \cap S \mid A \in \mathcal{U}\}$ contains all 2^k subsets of S.

It has been proven how big (n, k)-perfect hash families and (n, k)-universal sets can be and how fast they can be computed.

Theorem 2.2 ([NSS95])**.** *For any integers $n, k \geq 1$, an (n, k)-perfect hash family which contains $e^k k^{\mathcal{O}(\log k)} \cdot \log n$ functions can be constructed in time $e^k k^{\mathcal{O}(\log k)} \cdot n \log n$.*

Theorem 2.3 ([NSS95])**.** *For any integers $n, k \geq 1$, an (n, k)-universal set which contains $2^k k^{\mathcal{O}(\log k)} \cdot \log n$ functions can be constructed in time $2^k k^{\mathcal{O}(\log k)} \cdot n \log n$.*

2.4 A List of Frequently Used Problems

In this section, we define some problems and results that we frequently refer to throughout this thesis. This list is far from complete and we also recall problem definitions when we use them.

Knapsack

Input: A set of items $N = \{a_1, \ldots, a_n\}$, two functions $c, d : N \to \mathbb{N}$, and two integers B and D.

Question: Is there a set $S \subseteq N$ such that $c_\Sigma(S) \leq B$ and $d_\Sigma(S) \geq D$?

k-Subset Sum

Input: A set of items $N = \{a_1, \ldots, a_n\}$, a function $c : N \to \mathbb{N}$, and two integers k and G.

Question: Is there a set $S \subseteq N$ such that $|S| = k$ and $c_\Sigma(S) = G$?

We mostly use an analogous definition in which we say that we are given a multiset of integers instead of N and the function c. Knapsack is a generalization of k-Subset Sum. Both problems are NP-hard [Kar72] and W[1]-hard with respect to the size of the solution [DF95b].

Vertex Cover

Input: An undirected graph $G = (V, E)$ and an integer k.

Question: Is there a vertex set $C \subseteq V$ of size at most k such that $e \cap C \neq \emptyset$ for each edge $e \in E$?

In other words, every edge has at least one endpoint in C. Such a vertex set C is called a *vertex cover* of G. Vertex Cover is NP-hard even if each vertex in G has a degree of exactly 3 [Moh01].

Set Cover

Input: A universe $\mathcal{U}$, a family $\mathcal{F}$ of subsets over $\mathcal{U}$, and an integer k.

Question: Are there sets $F_1, \ldots, F_k \in \mathcal{F}$ such that $\mathcal{U} := \bigcup_{i=1}^{k} F_i$?

In other words, every item of $\mathcal{U}$ occurs at least in one of the sets $F_1, \ldots, F_k \in \mathcal{F}$. Set Cover is NP-hard [Kar72] and W[2]-complete when parameterized by k [DF13]. Assuming NP $\not\subseteq$ coNP/poly, Set Cover does not admit a polynomial kernel when parameterized by the size of the universe $|\mathcal{U}|$ [DLS14].

Red-Blue Non-Blocker

Input: An undirected bipartite graph G with vertex bipartition $V(G) = V_r \cup V_b$ and an integer k.

Question: Is there a set $S \subseteq V_r$ of size at least k such that each vertex v of V_b has a neighbor in $V_r \setminus S$?

Red-Blue Non-Blocker is W[1]-hard when parameterized by k [DF95b].

ILP-Feasibility

Input: A matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and a vector $\mathbf{b} \in \mathbb{R}^m$.

Question: Does a vector $\mathbf{x} \in \mathbb{Z}^n$ exist such that $\mathbf{Ax} \leq \mathbf{b}$?

It is possible to redefine equations $\mathbf{ax} = b$ for some vectors $\mathbf{a}$ and $\mathbf{x}$, and a number b

to be two inequalities $\mathbf{ax} \leq b$ and $-\mathbf{ax} \leq -b$. In general, ILP-Feasibility is NP-hard [Kar72]. However, it is known that instances of ILP-Feasibility with n variables and input length s can be solved using $s \cdot n^{2.5n+o(n)}$ arithmetic operations [FT87, LJ83].

2.5 Multiple-Choice Knapsack

In this section, we consider Multiple-Choice Knapsack (MCKP), a variant of Knapsack, in which the set of items is divided into classes. From every class, exactly one item can be chosen. MCKP is algorithmically closely related with Generalized Noah's Ark Problem and therefore particularly interesting for the examination we want to conduct in the next chapter. While MCKP has been studied from a classical and approximation point of view [KPP04], a parameterized point of view has not been considered yet, to the best of our knowledge. We want to close this gap in this section. The problem is formally defined as follows.

Multiple-Choice Knapsack Problem (MCKP)

Input: A set of items $N = \{a_1, \ldots, a_n\}$, a partition $\{N_1, \ldots, N_m\}$ 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 $c_\Sigma(S) \leq B$, $d_\Sigma(S) \geq D$, and $|S \cap N_i| = 1$ for each $i \in [m]$?

Recall that we write $c_\Sigma(A) := \sum_{a_i \in A} c(a_i)$ and $d_\Sigma(A) := \sum_{a_i \in A} d(a_i)$ for a set $A \subseteq N$. We call $c(a_i)$ the *cost of* a_i and $d(a_i)$ the *value of* a_i. Further, for a set $A \subseteq N$ we define $c(A) := \{c(a) \mid a \in A\}$ and $d(A) := \{d(a) \mid a \in A\}$. A set S holding the criteria of the question is called a *solution* for the instance $\mathcal{I}$ of MCKP. The sets N_i for $i \in [m]$ are called *classes*.

We examine MCKP with respect to the following parameters. 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* $C := \max_{a_j \in N} c(a_j)$. By var_c, we denote the *number of different costs*, that is, $\text{var}_c := |\{c(a_j) \ : \ a_j \in N\}|$. We define the *number of different values* var_d analogously. The size of the biggest class is denoted by L. If a class N_i contains two items a_p and a_q with the same cost and $d(a_p) \leq d(a_q)$, 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 $L \leq \text{var}_c$. Analogously, we may assume that no class contains two same-valued items and consequently $L \leq \text{var}_w$.

Since projects whose cost exceeds the budget can be removed from the input, we may assume $C \leq B$. Further, we assume that $B \leq C \cdot m$, as otherwise, we can

Table 2.1: Complexity results for Multiple-Choice Knapsack. The two question marks indicate unknown results.

Parameter	PFPT	FPT	XP	Remarks
Number of classes m	✗	✗	✓	Thm. 2.5
Budget B	✓	✓	✓	$\mathcal{O}(B \cdot \lvert N\rvert)$ [Pis95]
Maximum cost C	✓	✓	✓	$\mathcal{O}(C \cdot \lvert N\rvert \cdot m)$; Obs. 2.16
Threshold D	✓	✓	✓	$\mathcal{O}(D \cdot \lvert N\rvert)$ [BV04]
Largest class L	✗	✗	✗	NP-hard for $L = 2$ [KPP04]
Number of costs var_c	✗	?	✓	$\mathcal{O}(m^{\mathrm{var}_c - 1} \cdot \lvert N\rvert)$; Prop. 2.17
Number of values var_d	✗	?	✓	$\mathcal{O}(m^{\mathrm{var}_d - 1} \cdot \lvert N\rvert)$; Prop. 2.18
$\mathrm{var}_c + \mathrm{var}_d$	✗	✓	✓	Thm. 2.4

return yes if the total value of the most valuable items per class exceeds D, and no otherwise.

Table 2.1 presents an overview of known and new complexity results for MCKP. Observe that because var_c and var_d are bound in the size of the instance and MCKP is NP-hard [KPP04], MCKP can not be PFPT with respect to $\mathrm{var}_c + \mathrm{var}_d$, unless P = NP.

2.5.1 Algorithms for Multiple-Choice Knapsack

First, we provide some algorithms that solve MCKP. It is known that MCKP can be solved in $\mathcal{O}(B \cdot |N|)$ time [Pis95], or in $\mathcal{O}(D \cdot |N|)$ time [BV04]. As we may assume that $C \cdot m \geq B$, we may also observe the following.

Observation 2.16. MCKP *can be solved in* $\mathcal{O}(C \cdot |N| \cdot m)$ *time.*

In the following we want to study MCKP with respect to the number of different costs and different values. Knapsack is FPT with respect to the number of different costs, var_c [EKMR17]. This result is shown by a reduction to ILP-Feasibility instances with $f(\mathrm{var}_c)$ variables. This approach can not be adopted easily, as it has to be checked whether a solution contains exactly one item per class. In Proposition 2.17 and 2.18 we show that MCKP is XP with respect to the number of different costs and different values, respectively. Then, in Theorem 2.4 we show that MCKP is FPT with respect to the parameter $\mathrm{var}_c + \mathrm{var}_d$. In the following, let $\mathcal{I} = (N, \{N_1, \ldots, N_m\}, c, d, B, D)$ be an instance of MCKP, and let $\{c_1, \ldots, c_{\mathrm{var}_c}\} := c(N)$ and $\{d_1, \ldots, d_{\mathrm{var}_d}\} := d(N)$ denote the set of different costs and the set of the different values in $\mathcal{I}$, respectively. Without loss of generality, assume $c_i < c_{i+1}$ for each $i \in [\mathrm{var}_c - 1]$ and likewise we can assume $d_j < d_{j+1}$ 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 $d(N)$. Recall also that we assume that there is at most one item

with cost c_p and at most one item with value d_q in N_i, for every $i \in [m]$, $p \in [\mathrm{var}_c]$, and $q \in [\mathrm{var}_d]$.

Proposition 2.17. 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. Table definition. We describe a dynamic programming algorithm with a table DP that has var_c dimensions. We want to store the largest 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 of cost c_j for each $j \in [\mathrm{var}_c - 1]$ in entry $\mathrm{DP}[i, p_1, \ldots, p_{\mathrm{var}_c - 1}]$. Consequently, S contains exactly $p^{(i)}_{\mathrm{var}_c} := i - \sum_{j=1}^{\mathrm{var}_c - 1} p_j$ items of cost c_{var_c}.

We denote by $\mathbf{p}$ in the following $(p_1, \ldots, p_{\mathrm{var}_c - 1})$.

Algorithm. As a base case, we consider $i = 1$. For entries $\mathrm{DP}[1, \mathbf{p}]$, only subsets of N_1 with a single number are considered. Thus, for every $a \in N_1$ with a cost of $c(a) = c_j < c_{\mathrm{var}_c}$, store $\mathrm{DP}[1, \mathbf{0}_{(j)+1}] = d(a)$. If N_1 contains an item a with a cost of $c(a) = c_{\mathrm{var}_c}$, then store $\mathrm{DP}[1, \mathbf{0}] = d(a)$ and otherwise store $\mathrm{DP}[1, \mathbf{0}] = -\infty$. For all other $\mathbf{p}$, store $\mathrm{DP}[1, \mathbf{p}] = -\infty$.

Fix an $i \in [m]$. Once the entries $\mathrm{DP}[i, \mathbf{p}]$ for all $\mathbf{p}$ have been computed, we use the following recurrence to compute further values

$$\mathrm{DP}[i+1, \mathbf{p}] \;=\; \max_{a \in N_{i+1}} \begin{cases} \mathrm{DP}[i, \mathbf{p}_{(j)-1}] + d(a) & \text{if } c(a) = c_j < c_{\mathrm{var}_c} \text{ and } p_j \geq 1 \\ \mathrm{DP}[i, \mathbf{p}] + d(a) & \text{if } c(a) = c_{\mathrm{var}_c} \end{cases} \tag{2.2}$$

Return `yes` if $\mathrm{DP}[m, \mathbf{p}] \geq D$ for some $\mathbf{p}$ with $p^{(m)}_{\mathrm{var}_c} \cdot c_{\mathrm{var}_c} + \sum_{i=1}^{\mathrm{var}_c - 1} p_i \cdot c_i \leq B$ and return `no`, otherwise.

Correctness. For given integers $i \in [m]$ and $\mathbf{p} \in [i]_0^{\mathrm{var}_c - 1}$, we define $\mathcal{S}^{(i)}_{\mathbf{p}}$ 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 a $\mathbf{p} \in [i]_0^{\mathrm{var}_c - 1}$, we prove that $\mathrm{DP}[i, \mathbf{p}]$ stores the largest value of a set $S \in \mathcal{S}^{(i)}_{\mathbf{p}}$, by an induction. This implies that the algorithm is correct. The base cases are correct. Now, as an induction hypothesis assume that the claim is correct for a fixed $i \in [m-1]$. We first prove that if $\mathrm{DP}[i+1, \mathbf{p}] = q$, then there exists a set $S \in \mathcal{S}^{(i+1)}_{\mathbf{p}}$ with $d_\Sigma(S) = q$. Afterward, we prove that $\mathrm{DP}[i+1, \mathbf{p}] \geq d_\Sigma(S)$ for every set $S \in \mathcal{S}^{(i+1)}_{\mathbf{p}}$.

Now, let $\mathrm{DP}[i+1, \mathbf{p}] = q$. Let $a \in N_{i+1}$ be an item with a cost of $c(a) = c_j$ be an item of N_{i+1} that maximizes the right side of Equation (2.2) for $\mathrm{DP}[i+1, \mathbf{p}]$. Assume first that $c_j < c_{\mathrm{var}_c}$, and thus $\mathrm{DP}[i+1, \mathbf{p}] = q = \mathrm{DP}[i, \mathbf{p}_{(j)-1}] + d(a)$. By the induction hypothesis, there is a set $S \in S^{(i)}_{\mathbf{p}_{(j)-1}}$ such that $\mathrm{DP}[i+1, \mathbf{p}_{(j)-1}] = d_\Sigma(S) = q - d(a)$.

Observe that $S' := S \cup \{a\} \in S^{(i+1)}_{\mathbf{p}}$. 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.

Conversely, let $S \in \mathcal{S}^{(i+1)}_{\mathbf{p}}$ be a set of items and let $a \in S \cap N_{i+1}$ be an item. Assume first that $c(a) = c_j < c_{\mathrm{var}_c}$. Observe that $S' := S \setminus \{a\} \in S^{(i)}_{\mathbf{p}_{(j)-1}}$. Consequently,

$$\begin{aligned} \mathrm{DP}[i+1, \mathbf{p}] &\geq \mathrm{DP}[i, \mathbf{p}_{(j)-1}] + d(a) && (2.3)\\ &= \max\{d_\Sigma(S) \mid S \in S^{(i)}_{\mathbf{p}_{(j)-1}}\} + d(a) && (2.4)\\ &\geq d_\Sigma(S') + d(a) = d_\Sigma(S). \end{aligned}$$

Herebin, Inequality (2.3) is the definition of the recurrence in Equation (2.2), and Equation (2.4) 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 vectors $\mathbf{p}$ 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 $\sum_{j=1}^{\mathrm{var}_c - 1} p_j \geq m$ and each $i \in [m]$, the entry $\mathrm{DP}[i, \mathbf{p}]$ stores $-\infty$. Consequently, we consider a vector $\mathbf{p}$ with $p_j = m$ only if $p_\ell = 0$ for each $\ell \neq j$. Thus, we are only interested in $\mathbf{p} \in [m-1]_0^{\mathrm{var}_c - 1}$ or $\mathbf{p} = \mathbf{0}_{(j)+m}$ 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 $\mathbf{p}$.

For a fixed $\mathbf{p}$, each item $a \in N_i$ is considered exactly once in the computation of $\mathrm{DP}[i, \mathbf{p}]$. 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 $\mathrm{DP}[m, p_1, \ldots, p_{\mathrm{var}_c - 1}] \geq D$ and $p^{(m)}_{\mathrm{var}_c} \cdot c_{\mathrm{var}_c} + \sum_{i=1}^{\mathrm{var}_c - 1} p_i \cdot c_i \leq B$ for any $\mathbf{p}$. As we may assume $m^{\mathrm{var}_c - 1} > \mathrm{var}_c$, the running time of the entire algorithm is $\mathcal{O}(m^{\mathrm{var}_c - 1} \cdot |N|)$. □

One can define a dynamic programming algorithm which is very similar to the one in Proposition 2.17 in which the table has var_d dimensions. Herein, 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 can have.

Proposition 2.18. MCKP *can be solved in* $\mathcal{O}(m^{\mathrm{var}_d - 1} \cdot |N|)$*, where* var_d *is the number of different values.*

By Propositions 2.17 and 2.18, MCKP is `XP` with respect to var_c and var_d, respectively. In the following, we show that MCKP is `FPT` with respect to the combined parameter $\mathrm{var}_c + \mathrm{var}_d$. To prove this, we reduce an instance of MCKP to an instance of ILP-Feasibility, in which the number of variables is in $2^{\mathrm{var}_c + \mathrm{var}_d} \cdot \mathrm{var}_c$.

Theorem 2.4. *For an instance of* MCKP *one can define an equivalent instance of* ILP-Feasibility *with* $\mathcal{O}(2^{\mathrm{var}_c + \mathrm{var}_d} \cdot \mathrm{var}_c)$ *variables. Thus,* MCKP *is* `FPT` *with respect to* $\mathrm{var}_c + \mathrm{var}_d$.

Proof. Description. We may assume that any class N_i does not contain two items of the same cost or the same value. We conclude that $c(a) \neq c(b)$, $d(a) \neq d(b)$, and if $c(a) < c(b)$ then $d(a) < d(b)$ for items $a, b \in N_i$. Thus, each class N_i is 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 types. We say that *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, $\sum_{T \in \mathcal{T}} m_T = m$.

Observe, for every class N_j of type $T = (C, Q)$, and each item $a \in N_j$ with a cost of $c(a) \in C$ the value $d(a) \in Q$ can be determined. More precisely, if $c(a)$ is the ℓth cheapest cost in C, then the value of a is the ℓth smallest value in Q. For every type $T = (C, Q)$ and each $i \in [\mathrm{var}_c]$, we define a constant $d_{T,i} := -\sum_{i=1}^{m} \max d(N_i)$ if $c_i \notin C$. Otherwise, let $d_{T,i}$ be the ℓth smallest value in Q, if c_i is the ℓth smallest cost in C.

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

$$\sum_{T \in \mathcal{T}_C} \sum_{i=1}^{\mathrm{var}_c} x_{T,i} \cdot c_i \leq B \tag{2.5}$$

$$\sum_{T \in \mathcal{T}_C} \sum_{i=1}^{\mathrm{var}_c} x_{T,i} \cdot d_{T,i} \geq D \tag{2.6}$$

$$\sum_{i=1}^{\mathrm{var}_c} x_{T,i} = m_T \qquad \forall T \in \mathcal{T} \tag{2.7}$$

$$x_{T,i} \geq 0 \qquad \forall T \in \mathcal{T}, i \in [\mathrm{var}_c] \tag{2.8}$$

Correctness. Observe that if $c_i \notin C$, then Inequality (2.6) 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 (2.5) can only be correct if the total cost is at most B. Inequality (2.6) can only be correct if the total value is at least D. Equation (2.7) can only be correct if exactly m_T elements are picked from the classes of type T, for each $T \in \mathcal{T}$. It remains to show that the instance of the ILP-Feasibility has $\mathcal{O}(2^{\mathrm{var}_c + \mathrm{var}_d} \cdot \mathrm{var}_c)$ variables. Because $\mathcal{T} \subseteq 2^{c(N)} \times 2^{d(N)}$, the size of $\mathcal{T}$ is $\mathcal{O}(2^{\mathrm{var}_c + \mathrm{var}_d})$. Consequently, there are $\mathcal{O}(2^{\mathrm{var}_c + \mathrm{var}_d} \cdot \mathrm{var}_c)$ different options for the variables $x_{T,i}$. □

Observe that with the same technique, an instance of ILP-Feasibility with $2^{\mathrm{var}_c + \mathrm{var}_d} \cdot \mathrm{var}_d$ variables can be described.

2.5.2 Hardness With Respect to the Number of Classes

Kellerer et al. gave a reduction from Knapsack to MCKP in which each item in the instance of Knapsack is added to a unique class with a new item that has no costs and no value [KPP04].

Observation 2.19 ([KPP04])**.** MCKP *is* NP*-hard even if every class contains two items.*

The reduction above 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 [KPP04].

Theorem 2.5. MCKP *is* XP *and* W[1]*-hard with respect to the number of classes* m*.*

To show the hardness, we reduce from Multi-Selectable Subset Sum, a version of Subset Sum in which every integer can be chosen arbitrarily often, parameterized by k. More formally, in Multi-Selectable Subset Sum, a multi-set $Z = \{z_1, \ldots, z_n\}$ of integers and two integers Q and k are given, and it is asked, whether there is a *multi*-set $S \subseteq Z$ of size k such that $\sum_{s \in S} = Q$.

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 [DF95a, 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 2.20 ([DF95a])**.** Subset Sum *is* W[1]*-hard with respect to* k*, even when every integer in* Z *has multiplicity at least* k*.*

Proof of Theorem 2.5. Slicewice-polynomial. We first discuss the XP-algorithm and then focus on the W[1]-hardness. For every class N_i, iterate over the items $a_{j_i} \in N_i$ such that there are m nested loops. We check whether $\sum_{i=1}^{m} c(a_{j_i}) \leq B$

and $\sum_{i=1}^{m} d(a_{j_i}) \geq D$ in $\mathcal{O}(m)$ time, such that a solution of MCKP is computed in $\mathcal{O}(L^m \cdot m)$ time.

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 $z_{i,1} = z_{i,2} = \ldots = z_{i,k}$ 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 $(N, \{N_1, \ldots, N_k\}, Q, Q)$ where N is the union of the sets N_j, $j \in [k]$. Observe that $m = k$.

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

Let $S = \{z_{i_1}, \ldots, z_{i_k}\}$ be a solution for $\mathcal{I}$. Observe that, by construction, for each $j \in k$, the 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 $S' := \{a_{\ell_1}, a_{\ell_2}, \ldots, a_{\ell_k}\}$ is a solution for $\mathcal{I}'$ since

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

Conversely, let $S' := \{a_{\ell_1}, a_{\ell_2}, \ldots, a_{\ell_k}\}$ 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, since

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

$S := \{z_{i_j} \mid j \in [k]\}$ is a solution for $\mathcal{I}$. □

2.6 Penalty Sum

In this section, we examine Penalty Sum, a problem that has been introduced in [KS23b]. In the next chapter we will present a hardness reduction which builds on results from this chapter. Penalty Sum is defined as follows.

Penalty Sum

Input: A set of tuples $\{t_i = (a_i, b_i) \mid i \in [m], a_i \in \mathbb{Q}_+ \cup \{0\}, b_i \in (0, 1)\}$, two integers k and Q, and a number $D \in \mathbb{Q}_+$.

Question: Is there a set $S \subseteq [m]$ such that $|S| = k$ and $\sum_{i \in S} a_i - Q \cdot \prod_{i \in S} b_i \geq D$?

We first show that Penalty Sum can be solved in polynomial running time if the numbers in the input are given in unary. Afterward, we show that Penalty Sum nevertheless is NP-hard in general.

Proposition 2.21. PENALTY SUM *can be solved in* $\mathcal{O}((Q + \lceil D \rceil) \cdot m \cdot k)$ *time.*

Proof. Let $\mathcal{I} = (T, k, Q, D)$ be an instance of PENALTY SUM, where T is a set of tuples $t_i = (a_i, b_i)$ of size m. We say that a set $S \subseteq [m]$ is (c, A)-*feasible* for integers c and A if S has a size of c and $\sum_{j \in S} a_j = A$.

Table definition. We describe a dynamic programming algorithm with a table DP. We want that entry $\mathrm{DP}[i, c, A]$ stores the smallest value $\prod_{j \in S} b_j$ for a (c, A)-feasible set $S \subseteq [i]$ for each $i \in [m]$, each $c \in [k]$, and each $A \in [Q + \lceil D \rceil]$. We want that $\mathrm{DP}[i, c, A]$ stores ∞ if no such set S exists.

Algorithm. As a base case, for any $i \in [m]$, store $\mathrm{DP}[i, 0, 0] = 1$ and for $c, A > 0$ store $\mathrm{DP}[i, c, 0] = \infty$ and $\mathrm{DP}[i, 0, A] = \infty$. Further, store $\mathrm{DP}[1, c, A] = \infty$ if $c > 1$ or $A \notin \{0, a_1\}$. We consider $\mathrm{DP}[i, c, A] = \infty$ when we call invalid values such as $A < 0$.

We compute the table for increasing values of i. For a fixed i, let the values $\mathrm{DP}[i, c, A]$ be computed for each c and A. To compute further values we use the recurrence

$$\mathrm{DP}[i+1, c, A] \;=\; \min\{\, \mathrm{DP}[i, c, A];\ \mathrm{DP}[i, c-1, A - a_{i+1}] \cdot b_{i+1} \,\}. \tag{2.9}$$

If there is an $A \in [Q + \lceil D \rceil]$ such that $\mathrm{DP}[m, k, A] = B$ and $A - BQ \geq D$ return `yes`. Otherwise, return `no`.

Correctness. The base cases are correct. We show first that Recurrence (2.9) is correct. Afterward, we prove that this algorithm returns the right value.

As an induction hypothesis, assume that $\mathrm{DP}[i, c, A]$ stores the desired value for a fixed $i \in [m]$. We show first that if $\mathrm{DP}[i+1, c, A] = B$, then there is a (c, A)-feasible set $S \subseteq [i+1]$. Then, we show that $\mathrm{DP}[i+1, c, A] \leq \prod_{j \in S} b_j$ for every (c, A)-feasible set $S \subseteq [i+1]$.

If $\mathrm{DP}[i+1, c, A] = B$, then by Recurrence (2.9) $\mathrm{DP}[i, c-1, A - a_{i+1}] \cdot b_{i+1} = B$ or $\mathrm{DP}[i, c, A] = B$. In the latter case, the induction hypothesis directly proves that there is a (c, A)-feasible set $S \subseteq [i]$. In the first case, a $(c-1, A - a_{i+1})$-feasible set $S \subseteq [i]$ exists. Consequently, $a_{i+1} + \sum_{j \in S} a_j = A$ such that $S \cup \{i+1\}$ is (c, A)-feasible.

Now, let $S \subseteq [i+1]$ be a (c, A)-feasible set. If $i + 1 \notin S$, then we conclude that $\mathrm{DP}[i+1, c, A] \leq \mathrm{DP}[i, c, A] \leq \prod_{j \in S} b_j$, where the first inequality follows from Recurrence (2.9) and the second by the induction hypothesis. Otherwise, if $i+1 \in S$, then $S' := S \setminus \{i+1\}$ is a $(c-1, A - a_{i+1})$-feasible set and with the arguments from before $\mathrm{DP}[i+1, c, A] \leq \mathrm{DP}[i, c-1, A - a_{i+1}] \cdot b_{i+1} \leq b_{i+1} \cdot \prod_{j \in S'} b_j = \prod_{j \in S} b_j$.

Finally, assume that there is an $A \in [Q + \lceil D \rceil]$ such that $\mathrm{DP}[m, k, A] = B$ and $A - BQ \geq D$. Consequently, a (k, A)-feasible set $S \subseteq [m]$ with $\prod_{j \in S} b_j = B$

exists and so S is a solution for $\mathcal{I}$. Conversely, let S be a solution for $\mathcal{I}$. We define $A := \sum_{j \in S} a_j$, then S is a (c, A)-feasible set. So, $B = \mathrm{DP}[m, k, A] \leq \prod_{j \in S} b_j$ and so $A - BQ \geq A - Q \prod_{j \in S} b_j \geq D$.

Running time. The table contains $m \cdot k \cdot (Q + \lceil D \rceil)$ entries. Each of these stores a number between 0 and 1 (if not ∞), that is a multiplication of at most k numbers b_j. Consequently, the encoding length of each table entry is at most k times the longest encoding length of a b_j. Then, Recurrence (2.9) can be computed in constant time in our RAM-model. We want to declare, however, that the table entries may store numbers with an encoding length of up to $|k| \cdot \max_b$, where $\max_b$ is the maximum encoding length of a number b. □

2.6.1 Hardness of Subset Product

Now, we prove the NP-hardness of Penalty Sum. To this end, we first show the NP-hardness of the following variant of Subset Product.

Subset Product
Input: A multiset of positive integers $\{v_1, v_2, \ldots, v_m\}$ and integers M and k.
Question: Is there a set $S \subseteq [m]$ such that $|S| = k$ and $\prod_{i \in S} v_i = M$?

We note the definition of Subset Product is slightly different here from the formulation that appears for example in the book of Garey and Johnson [GJ79]. In particular, we assume that the size k of the set S is given and that all integers are positive. This makes the subsequent NP-hardness reductions slightly simpler.

The NP-hardness of Subset Product is not a new result. It was stated by [GJ79] without a full proof (the authors indicate that the problem is NP-hard by reduction from Exact Cover by 3-Sets (X3C), citing "Yao, private communication") and a full proof appears in the book of Moret [Mor97]. We reprove it here for our slightly adapted variant.

In X3C, the input is a universe $\mathcal{U}$ and a family $\mathcal{C}$ of subsets of $\mathcal{U}$ which have a size of three each. It is asked whether there is a subset $\mathcal{C}'$ of $\mathcal{C}$ such that $\mathcal{U} = \bigcup_{C \in \mathcal{C}'} C$ and the sets in $\mathcal{C}'$ are pairwise disjoint. In other words, for each item u of $\mathcal{U}$ there is exactly one set in $\mathcal{C}'$ containing u.

Lemma 2.22 ([Mor97])**.** Subset Product *is* NP*-hard.*

Proof. Reduction. Let $(\mathcal{U} := \{u_1, \ldots, u_{3n}\}, \mathcal{C} := \{C_1, \ldots, C_m\})$ be an instance of X3C. Let $p_1, \ldots, p_{3n}$ be the first $3n$ prime numbers, so that we may associate

each $u_j \in \mathcal{U}$ with a unique prime number p_j. For each set $C_i = \{u_a, u_b, u_c\}$, define $v_i := p_a \cdot p_b \cdot p_c$, that is, v_i is the product of the three primes associated with the elements of C_i. Now, let M be the product of the prime numbers p_1 to p_{3n}. Finally, set $k := n$. This completes the construction of the instance $(\{v_1, \dots v_m\}, M, k)$ of Subset Product.

Correctness. Now, observe that if $\prod_{i \in S} v_i = M$ for some $S \subseteq [m]$, then by the uniqueness of prime factorization, every prime number $p_1, \dots, p_m$ must appear exactly once across the prime factorizations of all numbers in $\{v_i \mid i \in S\}$. It follows by the construction that the collection of subsets $\mathcal{C}' := \{C_i \mid i \in S\}$ contains each element of $\mathcal{U}$ exactly once. Thus, if $(\{v_1, \dots v_m\}, M, k)$ is a `yes`-instance of Subset Product then $(\mathcal{U}, \mathcal{C})$ is a `yes`-instance of X3C. Conversely, if $(\mathcal{U}, \mathcal{C})$ is a `yes`-instance of X3C with solution $\mathcal{C}'$, then we can define $S := \{i \in [m] \mid C_i \in \mathcal{C}'\}$. Since every element of $\mathcal{U}$ appears in exactly one $C_i \in \mathcal{C}'$ and $|C_i| = 3$ for all $i \in [m]$, we have that $|\mathcal{C}'| = |\mathcal{U}|/3 = n = k$, and $\prod_{i \in S} v_i = p_1 \cdot \dots \cdot p_{3n} = M$. Thus, $(\{v_1, \dots v_m\}, M, k)$ is a `yes`-instance of Subset Product.

It remains to show that the construction of $(\{v_1, \dots v_m\}, M, k)$ from $(\mathcal{U}, \mathcal{C})$ takes polynomial time. In particular, we need to show that each of the primes $p_1, \dots p_{3n}$ (and thus the product M) can be constructed in polynomial time. This can be shown using two results from number theory: $p_j < j(\ln j + \ln \ln j)$ for $j \geq 6$, [Ros41, Dus99] and the set of all prime numbers in $[Z]$ can be computed in time $\mathcal{O}(Z/\ln \ln Z)$ [AB04]. Combining these, we have that the first $3n$ prime numbers can be generated in time $\mathcal{O}(n \ln n / \ln \ln n)$.

Given the prime numbers $p_1, \dots, p_{3n}$, it is clear that the numbers $\{v_i \mid i \in [m]\}$ can also be computed in polynomial time. The number M, being the product of $3n$ numbers each less than $3n(\ln 3n + \ln \ln 3n)$, can also be computed in time polynomial in n (though M itself is not polynomial in n). It follows that $(\{v_1, \dots v_m\}, M, k)$ can be constructed in polynomial time. □

2.6.2 Hardness of Penalty Sum

In the following, we first describe a simple reduction from an instance of Subset Product to an equivalent 'instance' of Penalty Sum, but one in which the numbers involved are irrational (and as such, cannot be produced in polynomial time). We then show how this transformation can be turned into a polynomial-time reduction by replacing the irrational numbers with suitably chosen rationals.

The reduction from Subset Product to Penalty Sum can be informally described as follows: For an instance $(\{v_1, \dots v_m\}, M, k')$ of Subset Product and a big integer A, we let a_i be (a rational close to) $A - \ln v_i$ and let $b_i := 1/v_i$, for

each $i \in [n]$. Let $Q := M$, let $k := k'$, and let D be (a rational close to) $kA - \ln Q - 1$.

Observe that we cannot set $a_i := A - \ln v_i$ or $D := kA - \ln Q - 1$ exactly, because in general these numbers are irrational and cannot be calculated exactly in finite time, or even stored in finite space. We temporarily forget about the need for rational numbers, and consider how the function $\sum_{i \in S} a_i - Q \cdot \prod_{i \in S} b_i$ behaves when we drop the 'a rational close to' qualifiers from the descriptions above. In particular, we show that the function reaches its theoretical maximum exactly when S is a solution to the Subset Product instance.

Reduction with irrational numbers

Construction 2.23. *Let $(\{v_1, \dots v_m\}, M, k)$ be an instance of* Subset Product. *We define the following (not necessarily rational) numbers.*

- *Define $A := \lceil \max_{i \in [m]}(\ln v_i) \rceil + 1$;*
- *Define $a_i^* := A - \ln v_i$ for each $i \in [m]$;*
- *Define $b_i := 1/v_i$ for each $i \in [m]$;*
- *Define $Q := M$;*
- *Define $D^* := kA - \ln Q - 1$.*

Finally, output instance $(\{(a_i^*, b_i) \mid i \in [m]\}, k, Q, D^*)$ of Penalty Sum.

We note the purpose of A is to ensure that $a_i^* > 0$ for each $i \in [m]$, as is required by the formulation of Penalty Sum. Now, define f^* to be a function mapping a subset S of $[m]$ to $\mathbb{R}$ by

$$f^*(S) := \sum_{i \in S} a_i^* - Q \cdot \prod_{i \in S} b_i. \tag{2.10}$$

Lemma 2.24. *For every set $S \subseteq [m]$ of size k the following hold.*

1. *$f^*(S) \leq D^*$, and*
2. *$f^*(S) = D^*$ if and only if $\prod_{i \in S} v_i = Q$.*

Proof. Observe that given S with $|S| = k$, the value $f^*(S)$ can be written as

$$f^*(S) = kA - \sum_{i \in S} \ln v_i - Q / \prod_{i \in S} v_i = kA - \ln \left(\prod_{i \in S} v_i \right) - Q / \prod_{i \in S} v_i$$

Letting $x_S := \prod_{i \in S} v_i$, we therefore have $f^*(S) = kA - \ln x_S - Qx_S^{-1}$. Define a function $g^* : \mathbb{R}_{>0} \to \mathbb{R}$ by $g^*(x) := kA - \ln x - Qx^{-1}$ and observe $f^*(S) = g^*(x_S)$ for any $S \in \binom{[m]}{k}$. Recall that a function f has a critical point at x' if $\frac{\mathrm{d}f}{\mathrm{d}x}(x') = 0$. Since $\frac{\mathrm{d}g^*}{\mathrm{d}x} = -x^{-1} + Qx^{-2}$, there are critical points x' in $x'^{-1} = Qx'^{-2}$, which is the case in $x' = Q$. Moreover, for $Q > x > 0$, we have $Qx^{-1} > 1$, implying

$$\frac{\mathrm{d}g^*}{\mathrm{d}x} = -x^{-1} + Qx^{-2} > -x^{-1} + x^{-1} = 0.$$

Further, for $x > Q > 0$, we have $Qx^{-1} < 1$, implying

$$\frac{\mathrm{d}g^*}{\mathrm{d}x} = -x^{-1} + Qx^{-2} < -x^{-1} + x^{-1} = 0.$$

It follows that $g^*(x)$ is strictly increasing in the range $0 < x < Q$ and strictly decreasing in the range $x > Q$. Thus, g^* has a unique maximum in the range $x > 0$, which is in $x = Q$. In particular, for all $S \subseteq [m]$, we have

$$f^*(S) = g^*(x_S) \leq g^*(Q) = kA - \ln Q - 1 = D^*. \tag{2.11}$$

With equality if and only if $x_S = \prod_{i \in S} v_i = Q$. □

The above result implies that $(\{(a_i^*, b_i) \mid i \in [m]\}, k, Q, D^*)$ is a `yes`-instance of 'Penalty Sum' if and only if $(\{v_1, \ldots v_m\}, M, k')$ is a `yes`-instance of Subset Product, when allowing irrational numbers.

We are now ready to fully describe the polynomial-time reduction from Subset Product to Penalty Sum, showing how we can adapt the ideas above to work for rational a_i and D.

Reduction with rational numbers

Let $(\{v_1, \ldots v_m\}, M, k')$ be an instance of Subset Product. So far we have seen that if we define a_i^*, b_i, Q, k, and D^* as beforehand, then, by Lemma 2.24, for any $S \subseteq [m]$, $f^*(S) = \sum_{i \in S} a_i^* - Q \cdot \prod_{i \in S} b_i \geq D^*$ if and only if $\prod_{i \in S} v_i = Q = M$.

Our task now is to show how to replace a_i^* and D^* with rationals a_i and D, in such a way that this property—$\sum_{i \in S} a_i - Q \cdot \prod_{i \in S} b_i \geq D$ if and only if $\prod_{i \in S} v_i = M$—still holds, so that an instance of Penalty Sum can be constructed in polynomial time. The key idea is to find rational numbers that can be encoded in polynomially many bits, but are close enough to their respective irrationals such that the difference between $f^*(S)$ and $f(S)$ (and between D^* and D) is guaranteed to be small. To this end, let us fix a positive integer H to be defined later, and we will require all

numbers a_i, b_i, and D to be a multiple of 2^{-H}. This ensures that the denominator part of any of these rationals can be encoded using $\mathcal{O}(H)$ bits.

Given a number $x \in \mathbb{R}$ and a positive integer H, we define $\lfloor x \rfloor_H := r_x/2^H$, where r_x is the largest integer such that $r_x/2^H \leq x$. Observe for example that because $25/2^3 < \pi < 26/2^3$ we have $r_\pi = 25$ and $\lfloor \pi \rfloor_3 = 3.125 = 25/2^3$. One may think of $\lfloor x \rfloor_H$ as the number derived from the binary representation of x by deleting all digits more than H positions after the binary point. Thus, as the binary expansion of π begins `11.00100 10000 11111...`, the binary expression of $\lfloor \pi \rfloor_3$ is `11.001`. Similarly, define $\lceil x \rceil_H := s_x/2^H$, where s_x is the smallest integer such that $x \leq s_x/2^H$. Finally, define $\delta := 1/2^H$.

Observation 2.25. *Let $x \in \mathbb{R}$. Then, $x - \delta < \lfloor x \rfloor_H \leq x \leq \lceil x \rceil_H < x + \delta$.*

We can now describe the reduction from Subset Product to Penalty Sum.

Construction 2.26. *Let $(\{v_1, \ldots v_m\}, M, k)$ be an instance of* Subset Product.

- *Define $A := \lceil \max_{i \in [m]}(\ln v_i) \rceil + 1$;*
- *Define $a_i := \lceil a_i^* \rceil_H = \lceil A - \ln v_i \rceil_H$ for each $i \in [m]$;*
- *Define $b_i := 1/v_i$ for each $i \in [m]$;*
- *Define $Q := M$;*
- *Define $D := \lfloor D^* \rfloor_H = \lfloor kA - \ln Q - 1 \rfloor_H$.*

Finally, output instance $\mathcal{I} := (\{(a_i, b_i) \mid i \in [m]\}, k, Q, D)$ of Penalty Sum.

In the following, we show that the two instances are equivalent. Now, define f to be a function mapping a subset S of $[m]$ to $\mathbb{R}$ by

$$f(S) := \sum_{i \in S} a_i - Q \cdot \prod_{i \in S} b_i.$$

Note that f is the same as the function f^* defined previously but with each a_i^* replaced by a_i. Then, $\mathcal{I}$ is a `yes`-instance of Penalty Sum if and only if there is some $S \subseteq [m]$ such that $f(S) \geq D$. The next lemma shows the close relation between f^* and f, and between D^* and D. This will be used in both directions to show the equivalence between `yes`-instances of Subset Product and Penalty Sum.

Lemma 2.27. *$f^*(S) \leq f(S) < f^*(S) + k\delta$ for each $S \subseteq [m]$; and $D^* - \delta < D \leq D^*$.*

Proof. By Observation 2.25, $\lceil x \rceil_H - x < \delta$ and $x - \lfloor x \rfloor_H < \delta$ for each $x \in \mathbb{R}$.

Observe that $f(S) - f^*(S) = \sum_{i \in S}(a_i - a_i^*)$ and $|S| = k$. We conclude that $0 \leq a_i - a_i^* = \lceil a_i^* \rceil_H - a_i^* < \delta$ for all $i \in [m]$. Consequently, $0 \leq f(S) - f^*(S) < k\delta$ and $D^* - D = D^* - \lfloor D^* \rfloor_H < \delta$. □

Corollary 2.28. *For every $S \subseteq [m]$ with $\prod_{i \in S} v_i = Q$ it follows $f(S) \geq D$.*

Proof. $D \overset{\text{Lem. 2.27}}{\leq} D^* \overset{\text{Lem. 2.24 (2)}}{=} f^*(S) \overset{\text{Lem. 2.27}}{\leq} f(S)$. □

We now have that $(\{v_1, \ldots v_m\}, M, k')$ being a `yes`-instance of Subset Product implies $\mathcal{I}$ being a `yes`-instance of Penalty Sum. To show the converse, we show that if $\prod_{i \in S} v_i = Q' \neq Q$ for every $S \subseteq [m]$, then $f(S) < D$. Since $f(S) < f^*(S) + k\delta$ and $D^* - \delta < D$, it is sufficient to show that $f^*(S) + k\delta \leq D^* - \delta$. This is equivalent to $(k+1)\delta \leq D^* - f^*(S)$. To this end, we first establish a lower bound on $D^* - f^*(S')$ in terms of Q, using the following technical lemma.

Lemma 2.29. *For every pair of positive integers Q and Q' with $Q \geq 2$ and $Q \neq Q'$ it follows that $\ln Q' - \ln Q + Q/Q' - 1 > Q^{-4}$.*

We explicitly note that we use the natural logarithm. For other logarithms, this lemma is not true. Consider for example integers $Q = 2$ and $Q' = 1$ in the logarithm on basis 2. We have $\log_2(1) - \log_2(2) + 2/1 - 1 = 0 - 1 + 2 - 1 = 0 < 2^{-4}$.

Proof. We first show that it is enough to consider the cases $Q' = Q+1$ and $Q' = Q-1$. Fix an integer $Q \in \mathbb{N}_+$ with $Q \geq 2$. Consider the function $h_Q : \mathbb{R}_{>0} \to \mathbb{R}$ given by

$$h_Q(x) = \ln x - \ln Q + Q/x - 1.$$

So, we aim to show that $h_Q(Q') \geq Q^{-4}$. Similar to the proof of Lemma 2.24, we can observe that the derivation

$$\frac{\mathrm{d}h_Q}{\mathrm{d}x} = x^{-1} - Qx^{-2} = \frac{1}{x}\left(1 - \frac{Q}{x}\right).$$

is less than 0 if $x < Q$; exactly 0 if $x = Q$; and greater than 0 if $x > Q$. It follows that in the range $x > 0$, the function h_Q has a unique minimum at $x = Q$, and is decreasing in the range $x < Q$ and increasing in the range $x > Q$. Thus, in particular $h_Q(Q') \geq h_Q(Q-1)$ if $Q' \leq Q - 1$ and $h_Q(Q') \geq h_Q(Q+1)$ if $Q' \geq Q + 1$. Since either $Q' \leq Q - 1$ or $Q' \geq Q + 1$ for any integer $Q' \neq Q$, it remains to show that $h_Q(Q-1) > Q^{-4}$ and $h_Q(Q+1) > Q^{-4}$.

To show $h_Q(Q-1) > Q^{-4}$ for any $Q \in \mathbb{N}_{\geq 2}$, we define the function $\lambda : \mathbb{R}_{>0} \to \mathbb{R}$ given by

$$\begin{aligned} \lambda(Q) &= h_Q(Q-1) - Q^{-4} \\ &= \ln(Q-1) - \ln Q + Q/(Q-1) - 1 - Q^{-4} \\ &= \ln(Q-1) - \ln Q + 1/(Q-1) - Q^{-4}. \end{aligned}$$

We then observe

$$\begin{aligned} \frac{\mathrm{d}\lambda}{\mathrm{d}Q} &= (Q-1)^{-1} - Q^{-1} + (Q-1)^{-2} + 4Q^{-5} \\ &> (Q-1)^{-2} + 4Q^{-5} > 0. \end{aligned}$$

Therefore, λ is a (strictly) increasing function. We conclude $\lambda(Q) > 0$, because of $\lambda(2) = 0 - \ln 2 + 1 - 1/16 \approx 0.244 > 0$, for all $Q \geq 2$, and thus $h_Q(Q-1) > Q^{-4}$.

To show that $h_Q(Q+1) > Q^{-4}$ for all $Q \in \mathbb{N}_{\geq 2}$, observe first that if $Q = 2$, $h_Q(Q+1) = \ln(3) - \ln(2) + 2/3 - 1 \approx 0.0721 > 0.0625 = 2^{-4}$ and so the claim is true. For any $Q \geq 3$, observe that $h_Q(Q+1) = \ln(Q+1) - \ln Q + \frac{Q}{Q+1} - 1 = \ln(\frac{Q+1}{Q}) - \frac{1}{Q+1}$. We use the Mercator series for the natural logarithm.

$$\ln\left(\frac{Q+1}{Q}\right) = \ln\left(1 + \frac{1}{Q}\right) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{kQ^k} = \frac{1}{Q} - \frac{1}{2Q^2} + \frac{1}{3Q^3} - \frac{1}{4Q^4} + \dots$$

Since $\frac{1}{kQ^k} - \frac{1}{(k+1)Q^{k+1}} > 0$ for all $k > 0$, we conclude

$$\ln\left(\frac{Q+1}{Q}\right) > \frac{1}{Q} - \frac{1}{2Q^2} = \frac{2Q-1}{2Q^2}.$$

Then,

$$\begin{aligned} \ln\left(\frac{Q+1}{Q}\right) - \frac{1}{Q+1} &> \frac{2Q-1}{2Q^2} - \frac{1}{Q+1} \\ &= \frac{(2Q-1)(Q+1) - 2Q^2}{2Q^2(Q+1)} \\ &= \frac{2Q^2 + Q - 1 - 2Q^2}{2Q^2(Q+1)} \\ &= \frac{Q-1}{2Q^2(Q+1)} \\ &\geq \frac{1}{Q^2(Q+1)} \\ &> \frac{1}{Q^4} \end{aligned}$$

where the last two inequalities use $Q \geq 3$. □

Corollary 2.30. *If* $\prod_{i \in S} v_i = Q' \neq Q$ *for some* $Q \geq 2$ *and* $S \subseteq [m]$, *then* $D^* - f^*(S) > Q^{-4}$.

Proof. Recall $f^*(S) = kA - \ln(\prod_{i \in S} v_i) - Q/(\prod_{i \in S} v_i) = kA - \ln Q' - Q/Q'$ and that $D^* = kA - \ln Q - 1$. Then, $D^* - f^*(S) = \ln Q' - \ln Q + Q/Q' - 1$. With Lemma 2.29 we conclude $D^* - f^*(S) > Q^{-4}$. □

Given the above, we can now fix a suitable value for H. Given that we want to conclude $D^* - f^*(S) \geq (k+1)\delta = \frac{(k+1)}{2^H}$ from $\prod_{i \in S} v_i \neq Q$, and assuming without loss of generality that $k < Q$, it is sufficient to set $H := 5\lceil \log_2 Q \rceil$.

Corollary 2.31. *If* $H = 5\lceil \log_2 Q \rceil$ *and* $\delta = (1/2^H)$, *then the instance of* Penalty Sum *constructed in Construction 2.26 holds* $Q^{-4} \geq (k+1)\delta$.

Proof. We assume $k < Q$. Then, $(k+1)\delta \leq Q/2^H \leq Q/Q^5 = Q^{-4}$. □

We now have all the necessary pieces to reduce from Subset Product to Penalty Sum and therefore show the intractability of Penalty Sum.

Theorem 2.6. Penalty Sum *is* NP*-hard.*

Proof. Given an instance $\mathcal{I} = (\{v_1, \ldots v_m\}, M, k')$ of Subset Product, define $Q := M$, $H := 5\lceil \log_2 Q \rceil$, and $\delta := (1/2^H)$. Construct A, a_i, b_i, k, and D as in Construction 2.26. That is:

- Define $A := \lceil \max_{i \in [m]}(\ln v_i) \rceil + 1$;
- Define $a_i := \lceil a_i^* \rceil_H = \lceil A - \ln v_i \rceil_H$ for each $i \in [m]$;
- Define $b_i := 1/v_i$ for each $i \in [m]$;
- Define $Q := M$;
- Define $D := \lfloor D^* \rfloor_H = \lfloor kA - \ln Q - 1 \rfloor_H$.

Let $\mathcal{I}' = (\{(a_i, b_i) \mid i \in [m]\}, k, Q, D)$ be the resulting instance of Penalty Sum.

We first show that $\mathcal{I}'$ is a `yes`-instance of Penalty Sum if and only if $\mathcal{I}$ is a `yes`-instance of Subset Product. Suppose first that $\mathcal{I}$ is a `yes`-instance of Subset Product. Then, $\prod_{i \in S} v_i = M = Q$ for some $S \subseteq [m]$. Consequently, by Corollary 2.28, $f(S) \geq D$ and so instance $\mathcal{I}'$ is a `yes`-instance of Penalty Sum.

Conversely, suppose that $\mathcal{I}'$ is a `yes`-instance of Penalty Sum. Then, there is some $S \subseteq [m]$ such that $f(S) \geq D$. By Lemma 2.27 and Corollary 2.31, we conclude $f^*(S) > f(S) - k\delta \geq D - k\delta > D^* - (k+1)\delta \geq D^* - Q^{-4}$. Consequently,

$D^* - f^*(S) \leq Q^{-4}$, which by Corollary 2.30 implies that $\prod_{i \in S} v_i = Q$. Therefore, $\mathcal{I}$ is a `yes`-instance of Subset Product.

It remains to show that the reduction takes polynomial time. For this, it is sufficient to show that the rationals A, k, D, a_i, and b_i can all be computed in polynomial time for each $i \in [m]$. Observe that $A = \lceil \max_{i \in [m]} (\ln v_i) \rceil$ is the unique integer such that $e^A > \max_{i \in [m]} v_i > e^{A-1}$. We have $1 \leq A \leq \lceil \max_{i \in [m]} \log_2 v_i \rceil$ since $\ln v_i < \log_2 v_i$. So we can compute A in polynomial time by checking all integers in this range.

For each $i \in [m]$, $a_i = \lceil A - \ln v_i \rceil_H = r_i / 2^H$, where r_i is the minimum integer such that $A - \ln v_i \leq r_i / 2^H$. Thus, we can compute r_i by checking $e^{A - r_i/2^H} \leq v_i$ with $r_i = 2^H \cdot (A - \lceil \ln v_i \rceil_H)$, setting r_i to its successor if the inequality is not satisfied. Thus, we can construct a_i in polynomial time, and a_i can be represented with $\mathcal{O}(\log_2 r + H)$ bits. The construction of D can be handled in a similar way.

For each $i \in [m]$, rational $b_i = 1/v_i$ can be represented with $\mathcal{O}(\log_2 v_i)$ bits (recall that we represent $1/v_i$ with binary representations of the integers 1 and v_i). It takes $\mathcal{O}(\log_2 v_i)$ time to construct b_i. The integers Q and k are taken directly from the instance $\mathcal{I}$ of Subset Product. □

Chapter 3

The Generalized Noah's Ark Problem

3.1 Introduction

The definition of Maximize Phylogenetic Diversity was given with the intention of helping to systematically address the challenge of preservation of biological diversity [Fai92, Cro97]. We, however, have to acknowledge the fact that selecting a species for protection in an ecological intervention will not necessarily help to protect them. Real-world interaction always have to consider uncertainties such as diseases, famines, floods, or other natural catastrophes. It is therefore useful to consider the protection of a given taxon only at a specific survival probability [Wei98]. Further, we may assume that investing more into a given species will increase their survival probability by a certain amount.

In such a model, we therefore have to consider the *expected* phylogenetic diversity. In the case of Generalized Noah's Ark Problem (GNAP), for each species one may choose from a set of different actions. Each choice is then associated with a cost and with a resulting survival probability. With a preprocessing step, we can even consider combinations of different actions.

Introducing cost differences for species protection makes the problem of maximizing phylogenetic diversity NP-hard [PG07] and thus all of the even richer models are NP-hard as well. However, in special cases even then a solution can still be computed greedily [HS06, HS07].

Billionnet provided some practical algorithmic ideas and results for GNAP with integer linear programming [Bil13, Bil17]. Pardi showed that GNAP can be solved in pseudo-polynomial running time in the budged plus the number of possible solutions [Par09].

Apart from these algorithms and the NP-hardness, there is no work that system-

atically studies which structural properties of the input make GNAP tractable. In this chapter, we try to fill this gap. On the way, we also observe close relations to Multiple-Choice Knapsack and to Penalty Sum, which we observed in the previous chapter.

Structure of the chapter. In the next section, we give a formal definition of Generalized Noah's Ark Problem and we give an overview over results. Further, we provide first observations. In Section 3.3, we provide the algorithmic ideas for solving GNAP in its most generalized form. We, additionally, prove that GNAP is W[1]-hard when parameterized by the number of taxa. In Section 3.4, we consider the special case of GNAP in which for each taxon we can chose between only two options.

3.2 Preliminaries

In this section, we consider definitions used in this chapter, an overview over the results and some preliminary observations.

3.2.1 Projects and Phylogenetic Diversity

A *project* $p_{i,j}$ is a tuple $(c_{i,j}, w_{i,j}) \in \mathbb{N}_0 \times \mathbb{Q} \cap \mathbb{R}_{[0,1]}$, where $c_{i,j}$ is the *cost* and $w_{i,j}$ is the *survival probability* of $p_{i,j}$. For a given phylogenetic X-tree $\mathcal{T}$ and a taxon $x_i \in X$, a *project list* P_i is an ℓ_i-tuple of projects $(p_{i,1}, \ldots, p_{i,\ell_i})$. As a project with a higher cost will only be considered when the survival probability is higher, we assume the costs and the 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 each $j \in [\ell_i - 1]$. An *m-collection of projects* $\mathcal{P}$ is a set of m project lists $\{P_1, \ldots, P_m\}$. For a project set S, the *total cost* $\mathrm{Cost}(S)$ *of* S is $\sum_{p_{i,j} \in S} c_{i,j}$.

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

$$PD_{\mathcal{T}}(S) := \sum_{uv \in E} \lambda(uv) \cdot \left(1 - \prod_{x_i \in \mathrm{off}(v)} (1 - w_{i,j_i})\right). \tag{3.1}$$

The formula $\left(1 - \prod_{x_i \in \mathrm{off}(v)} (1 - w_{i,j_i})\right)$ describes the likelihood that at least one offspring of v survives. Thus, the phylogenetic diversity is the sum of the expected values of the edges of $\mathcal{T}$ when applying S. If the survival probabilities of all projects

in S is either 0 or 1, then this definitions coincides with the standard definition of phylogenetic diversity given in Equation (2.1).

In this chapter, we use the convention that $n := |V(\mathcal{T})|$. Observe $n \in \mathcal{O}(|X|)$.

3.2.2 Problem Definitions

We now define this chapter's main problem, Generalized Noah's Ark Problem, and the special case where each species has two projects, $a_i \xrightarrow{c_i} b_i$ [2]-Noah's Ark Problem.

Generalized Noah's Ark Problem (GNAP)

Input: A phylogenetic X-tree $\mathcal{T} = (V, E, \lambda)$, an $|X|$-collection of projects $\mathcal{P}$, an integer $B \in \mathbb{N}_0$, and a number $D \in \mathbb{Q}_{\geq 0}$.

Question: Is there a set of projects $S = \{p_{1,j_1}, \ldots, p_{|X|,j_{|X|}}\}$, one from each project list of $\mathcal{P}$, such that $PD_{\mathcal{T}}(S) \geq D$ and $\text{Cost}(S) \leq B$?

A project set S is called *a solution for instance* $\mathcal{I} = (\mathcal{T}, \mathcal{P}, B, D)$ if S satisfies the conditions in the question of the problem definition.

$a_i \xrightarrow{c_i} b_i$ [2]-Noah's Ark Problem ($a_i \xrightarrow{c_i} b_i$ [2]-NAP)

Input: A phylogenetic X-tree $\mathcal{T} = (V, E, \lambda)$, a $|X|$-collection of projects $\mathcal{P}$ in which the project list P_i contains exactly two projects $(0, a_i)$ and (c_i, b_i) for each $i \in [|X|]$, an integer $B \in \mathbb{N}_0$, and a number $D \in \mathbb{Q}_{\geq 0}$.

Question: Is there a set of projects $S = \{p_{1,j_1}, \ldots, p_{|X|,j_{|X|}}\}$, one from each project list of $\mathcal{P}$, such that $PD_{\mathcal{T}}(S) \geq D$, and $\text{Cost}(S) \leq B$?

In other words, in an instance $a_i \xrightarrow{c_i} b_i$ [2]-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 P_i contains two projects $(\bar{c}, a_i)$ and $(\hat{c}, b_i)$ with $\bar{c} > 0$ for an $i \in [|X|]$, we can reduce the cost of the two projects and the budget by $\bar{c}$ each to obtain an equivalent instance. Therefore, we will assume that the project with the lower survival probability has a cost of 0 and we refer to the cost of the other project as c_i.

As a special case of this problem, we consider $0 \xrightarrow{c} b_i$ [2]-NAP where every project with a positive survival probability has a cost of c. Observe that for each $c \in \mathbb{N}$, an instance $\mathcal{I} = (\mathcal{T}, \mathcal{P}, B, D)$ of $0 \xrightarrow{c} b_i$ [2]-NAP can be reduced to an equivalent instance $\mathcal{I}' = (\mathcal{T}, \mathcal{P}', B', D)$ of $0 \xrightarrow{1} b_i$ [2]-NAP by replacing each project (c, b_i) with $(1, b_i)$, and setting $B' = \lfloor B/c \rfloor$. Thus, $0 \xrightarrow{c} b_i$ [2]-NAP can be considered as the special case of GNAP with unit costs for projects. In the following we refer to this problem as unit-cost-NAP.

3.2.3 Parameters, and Results Overview

We study GNAP and the special cases of GNAP with respect to several parameters which we describe in the following; For an overview of the results see Table 3.1. If not stated differently, we assume in the following that $i \in [|X|]$ and $j \in [|P_i|]$. The input of GNAP directly gives the following natural parameters: The *number of taxa* $|X|$, the *budget* B, and the required *diversity* D. Closely related to B is $C := \max_{i,j} c_{i,j}$, the *maximum cost of a project*. 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 $a_i \overset{c_i}{\to} b_i$ [2]-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 $\|\mathcal{P}\| \leq |X| \cdot L$. We denote with $\text{var}_c := |\{c_{i,j} : (c_{i,j}, w_{i,j}) \in P_i, P_i \in \mathcal{P}\}|$, the *number of different costs*. We define the *number of different survival probabilities* var_w, analogously. The consideration of this type of parameterization, called the *number of numbers* parameterization was initiated by Fellows et al. [FGR12]; It is motivated by the idea that in many real-life instances the number of numbers may be small. In fact, for non-negative numbers, it is never larger than the maximum values which are used in pseudo-polynomial time algorithms. Also, we consider the *maximum encoding length for survival probabilities* w-code $:= \max_{i,j}$(*binary length of* $w_{i,j}$) and the *maximum edge weight* $\max_\lambda := \max_{e \in E} \lambda(e)$. Observe that because the maximal survival probability of a taxon could be smaller than 1, one can not assume that $\max_\lambda \leq D$, in this chapter.

3.2.4 Observations for GNAP

We first present some basic observations that provide some first complexity classifications. In the problem with exactly two projects per taxa, $a_i \overset{c_i}{\to} b_i$ [2]-NAP, one can iterate over all subsets X' of taxa and check if it is a possible solution pay c_i to increase the survival probability for each $x_i \in X'$. To this end, we check if $\sum_{x_i \in X'} c_i \leq B$ and compute if the phylogenetic diversity is at least D, when the survival probability of every $x_i \in X'$ is b_i and a_i otherwise. Thus, $a_i \overset{c_i}{\to} b_i$ [2]-NAP is fixed-parameter tractable with respect to the number of taxa.

Observation 3.1. $a_i \overset{c_i}{\to} b_i$ [2]-NAP *can be solved in* $2^{|X|} \cdot \text{poly}(|\mathcal{I}|)$ *time.*

Table 3.1: Complexity results for Generalized Noah's Ark Problem. Recall that the special cases mentioned are $0 \overset{c_i}{\to} 1$ [2]-NAP which is the special case where the survival probabilities are only 0 or 1, and unit-cost-NAP is the special case where each project has unit costs. Entries with the sign "—" mark parameters that are (partially) constant in the specific problem definition and thus are not interesting.

Parameter	GNAP		GNAP with height$_{\mathcal{T}} = 1$	
$\|X\|$	W[1]-hard, XP	Thm. 3.3, Prop. 3.5	W[1]-hard, XP	Thm. 3.3, Prop. 3.5
B	XP; FPT is *open*	Obs. 3.2	PFPT $\mathcal{O}(B \cdot \|\mathcal{P}\|)$	Prop. 3.10
C	NP-h for $C = 1$	Thm. 3.5	PFPT $\mathcal{O}(C \cdot \|\mathcal{P}\| \cdot \|X\|)$	Prop. 3.10
D	NP-h for $D = 1$	Obs. 3.4	NP-h for $D = 1$	Obs. 3.4
$\max_\lambda$	NP-h for $\max_\lambda = 1$	Thm. 3.3	NP-h for $\max_\lambda = 1$	Thm. 3.3
var_c	NP-h for $\text{var}_c = 2$	Thm. 3.5	XP $\mathcal{O}(\|X\|^{\text{var}_c - 1} \cdot \|\mathcal{P}\|)$	Prop. 3.10
var_w	NP-h for $\text{var}_w = 2$	Obs. 3.3	NP-h for $\text{var}_w = 2$	Obs. 3.3
D + w-code	*open*		FPT $\mathcal{O}(D \cdot 2^{\text{w-code}} \cdot \|\mathcal{P}\|)$	Prop. 3.10
$B + \text{var}_w$	XP $\mathcal{O}(B \cdot \|X\|^{2 \cdot \text{var}_w + 1}$	Thm. 3.2	PFPT	Prop. 3.10
$D + \text{var}_w$	NP-h for $D = 1, \text{var}_w = 2$	Obs. 3.4	NP-h for $D = 1, \text{var}_w = 2$	Obs. 3.4
$\text{var}_c + \text{var}_w$	XP $\mathcal{O}(\|X\|^{2 \cdot (\text{var}_c + \text{var}_w) + 1}$	Thm. 3.1	FPT	Thm. 3.4

Parameter	$0 \overset{c_i}{\to} 1$ [2]-NAP		unit-cost-NAP	
$\|X\|$	FPT	Obs. 3.1	FPT	Obs. 3.1
B	PFPT $\mathcal{O}(B^2 \cdot n)$	[PG07]	XP	Obs. 3.2
C	PFPT $\mathcal{O}(C^2 \cdot n^3)$	Cor. 3.11	—	
D	PFPT $\mathcal{O}(D^2 \cdot n)$	Prop. 3.12	*open*	
$\max_\lambda$	PFPT $\mathcal{O}((\max_\lambda)^2 \cdot n^3)$	Cor. 3.13	*open*	
var_c	XP	Cor. 3.6	—	
var_w	—		XP	Cor. 3.7

A GNAP solution contains at most B projects with positive costs. Hence, a solution can be found by iterating over all B-sized subsets X' of taxa and checking every combination of projects for X'. Like before, we have to check that the budget is not exceeded and the phylogenetic diversity of the selected projects is at least D. This brute-force algorithm shows that GNAP is XP with respect to the budget.

Observation 3.2. GNAP *can be solved in* $(|X| \cdot L)^B \cdot \text{poly}(|\mathcal{I}|)$ *time.*

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 $c_\Sigma(N') \leq B$ and $d_\Sigma(N') \geq D$. We describe briefly a known reduction from Knapsack to $0 \overset{c_i}{\to} 1$ [2]-NAP [PG07]. Let $\mathcal{I} = (N, c, d, B, D)$ be an instance of Knapsack. Define an N-tree $\mathcal{T} := (V, E, \lambda)$ with vertices $V := \{\rho\} \cup N$ and edges $E := \{\rho x_i \mid x_i \in N\}$ of weight $\lambda(\rho x_i) := d(x_i)$. For each taxon x_i we define a project list P_i that contains two projects $(0, 0)$ and $(c(x_i), 1)$. Then, the instance $(\mathcal{T}, \mathcal{P}, B' := B, D' := D)$ is a yes-instance of $0 \overset{c_i}{\to} 1$ [2]-NAP if and only if (N, c, d, B, D) is a yes-instance of Knapsack. Because Knapsack is NP-hard, also $0 \overset{c_i}{\to} 1$ [2]-NAP is NP-hard.

Observation 3.3 ([PG07]). $0 \xrightarrow{c_i} 1$ [2]-NAP *is* NP*-hard, even if the tree* $\mathcal{T}$ *is a star.*

Because $0 \xrightarrow{c_i} 1$ [2]-NAP is a special case of GNAP in which $L = 2$, w-code $= 1$, and $\text{var}_w = 2$, we conclude that GNAP is NP-hard, even if $\text{height}_{\mathcal{T}} = \text{w-code} = 1$ and $L = \text{var}_w = 2$. In this reduction, one could also set $D' := 1$ and set the survival probability of every project with a positive cost to $1/D$.

Observation 3.4. $0 \xrightarrow{c_i} b$ [2]-NAP *is* NP*-hard, even if* $b \in (0, 1]$ *is a constant,* $D = 1$*, and the given phylogenetic* X*-tree* $\mathcal{T}$ *is a star.*

3.3 The Generalized Noah's Ark Problem

In this section, with GNAP we consider the most general form of the problem.

3.3.1 Algorithms for the Generalized Noah's Ark Problem

First, we observe that for a constant number of taxa, we can solve GNAP in polynomial time by considering all the possible project choices for each taxon.

Proposition 3.5. GNAP *is* XP *with respect to* $|X|$.

Proof. For every taxon $x_i \in X$, iterate over the projects p_{i,j_i} of P_i such that there are $|X|$ nested loops to compute a set $S := \{p_{i,j_i} \mid i \in [|X|]\}$. Return yes, if $\text{Cost}(S) \leq B$ and $PD_{\mathcal{T}}(S) \geq D$. Otherwise, return no, after the iteration.

This algorithm is clearly correct. We can check whether $\text{Cost}(S) \leq B$ and $PD_{\mathcal{T}}(S) \geq D$ in $\mathcal{O}(n^2)$ time. Therefore, in $\mathcal{O}(L^{|X|} \cdot n^2)$ time a solution is computed. □

In Theorem 3.1, we 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 $\mathcal{C} := \{c_1, \ldots, c_{\text{var}_c}\}$ and $\mathcal{W} := \{w_1, \ldots, w_{\text{var}_w}\}$ 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 assume $w_j < w_{j+1}$ for each $j \in [\text{var}_w - 1]$, likewise. 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 $\mathbf{a}$ and $\mathbf{b}$ we denote vectors $(a_1, \ldots, a_{\text{var}_c - 1})$ and $(b_1, \ldots, b_{\text{var}_w - 1})$, respectively.

Theorem 3.1. GNAP *can be solved in* $\mathcal{O}\left(|X|^{2(\mathrm{var}_c+\mathrm{var}_w-1)}\cdot(\mathrm{var}_c+\mathrm{var}_w)\right)$ *time.*

Proof. Table Defintion. We describe a dynamic programming algorithm with two tables DP and DP′ 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 w_{var_w}. Recall that T_v is the subtree rooted at v and the offspring $\mathrm{off}(v)$ of v are the leaves in T_v, and that the i-partial subtree $T_{v,i}$ rooted at v is the subtree of T_v containing only the first children $w_1,\dots,w_i$ of v for some $i\in[t]$ for a vertex v with children $w_1,\dots,w_t$ of v. For a vertex $v\in V$ and given vectors $\mathbf{a}$ and $\mathbf{b}$, we define $\mathcal{S}^{(v)}_{\mathbf{a},\mathbf{b}}$ to be the family of sets of projects S such that

- S contains exactly one project of P_i for each $x_i\in\mathrm{off}(v)$,
- S contains exactly a_k projects of cost c_k for each $k\in[\mathrm{var}_c-1]$, and
- 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 $\mathbf{a}$, and $\mathbf{b}$ and a given integer $i\in[t]$ we define $\mathcal{S}^{(v,i)}_{\mathbf{a},\mathbf{b}}$ analogously, just that exactly one project of P_j is chosen for each $x_j\in\mathrm{off}(u_1)\cup\dots\cup\mathrm{off}(u_i)$.

It follows that we can compute how many projects with cost c_{var_c} and survival probability w_{var_w} a set $S\in\mathcal{S}^{(v)}_{\mathbf{a},\mathbf{b}}$ contains. That are $a^{(v)}_{\mathrm{var}_c}:=|\mathrm{off}(v)|-\sum_{j=1}^{\mathrm{var}_c-1}a_j$ projects with a cost of c_{var_c} and $b^{(v)}_{\mathrm{var}_w}:=|\mathrm{off}(v)|-\sum_{j=1}^{\mathrm{var}_w-1}b_j$ projects with a survival probability of w_{var_w}. We want entries $\mathrm{DP}[v,\mathbf{a},\mathbf{b}]$ to store the largest expected phylogenetic diversity $\mathrm{PD}_{T_v}(S)$ of the tree T_v, for a set $S\in\mathcal{S}^{(v)}_{\mathbf{a},\mathbf{b}}$. We analogously want $\mathrm{DP}'[v,i,\mathbf{a},\mathbf{b}]$ to store the largest expected phylogenetic diversity $\mathrm{PD}_{T_{v,i}}(S)$ of the tree $T_{v,i}$, for a set $S\in\mathcal{S}^{(v,i)}_{\mathbf{a},\mathbf{b}}$. We further define the total survival probability to be

$$w(b_{\mathrm{var}_w},\mathbf{b}):=1-(1-w_{\mathrm{var}_w})^{b_{\mathrm{var}_w}}\cdot\prod_{i=1}^{\mathrm{var}_w-1}(1-w_i)^{b_i},\tag{3.2}$$

when b_{var_w} and $\mathbf{b}$ describe the number of chosen single survival probabilities.

Algorithm. As a base case, 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 $\mathbf{a}$ and $\mathbf{b}$. So, we store $\mathrm{DP}[x_i,\mathbf{a},\mathbf{b}]=0$, if P_i contains a project $p=(c_k,w_\ell)$ such that

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

Otherwise, store $\mathrm{DP}[x_i, \mathbf{a}, \mathbf{b}] = -\infty$.

Let v be an internal vertex with children $u_1, \ldots, u_t$. We define

$$\mathrm{DP}'[v, 1, \mathbf{a}, \mathbf{b}] \;=\; \mathrm{DP}[u_1, \mathbf{a}, \mathbf{b}] + \lambda(vu_1) \cdot w\left(b^{(u_1)}_{\mathrm{var}_w}, \mathbf{b}\right). \tag{3.3}$$

To compute further values of DP', we use the recurrence

$$\begin{aligned} &\mathrm{DP}'[v, i+1, \mathbf{a}, \mathbf{b}] \\ = \;& \max_{\mathbf{a}', \mathbf{b}'} \; \mathrm{DP}'[v, i, \mathbf{a} - \mathbf{a}', \mathbf{b} - \mathbf{b}'] + \mathrm{DP}[u_{i+1}, \mathbf{a}', \mathbf{b}'] + \lambda(vu_{i+1}) \cdot w\left(b^{(u_{i+1})}_{\mathrm{var}_w}, \mathbf{b}'\right). \end{aligned} \tag{3.4}$$

Herein, $\mathbf{a}'$ and $\mathbf{b}'$ are selected to satisfy $\mathbf{0} \leq \mathbf{a}' \leq \mathbf{a}$ and $\mathbf{0} \leq \mathbf{b}' \leq \mathbf{b}$.

Finally, we define $\mathrm{DP}[v, \mathbf{a}, \mathbf{b}] = \mathrm{DP}'[v, t, \mathbf{a}, \mathbf{b}]$.

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

Correctness. For any vertex v, vectors $\mathbf{a}$, $\mathbf{b}$, and an integer i, we prove that $\mathrm{DP}[v, \mathbf{a}, \mathbf{b}]$ and $\mathrm{DP}'[v, i, \mathbf{a}, \mathbf{b}]$ store $\max PD_{\mathcal{T}_v}(S)$ for $S \in \mathcal{S}^{(v)}_{\mathbf{a},\mathbf{b}}$ and $\max PD_{\mathcal{T}_{v,i}}(S)$ for $S \in \mathcal{S}^{(v,i)}_{\mathbf{a},\mathbf{b}}$, 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 $\mathbf{a}$ and $\mathbf{b}$ correspond to a feasible project. So, the table F stores the correct value in the base cases. For an internal vertex v with children $u_1, \ldots, u_t$, and $i \in [t-1]$, observe that $PD_{\mathcal{T}_{v,1}}(S) = PD_{\mathcal{T}_{u_1}}(S) + \lambda(vu_1) \cdot w(b^{(u_{i+1})}_{\mathrm{var}_w}, \mathbf{b})$ for $S \in \mathcal{S}^{(v,1)}_{\mathbf{a},\mathbf{b}}$, where $w(b^{(u_{i+1})}_{\mathrm{var}_w}, \mathbf{b})$ is the survival probability at u_1. Thus entry $\mathrm{DP}'[v, 1, \mathbf{a}, \mathbf{b}]$ stores the correct value. Further, the value in entry $\mathrm{DP}[v, \mathbf{a}, \mathbf{b}]$ stores the correct value, when $\mathrm{DP}'[v, t, \mathbf{a}, \mathbf{b}]$ stores the correct value, because $\mathcal{S}^{(v)}_{\mathbf{a},\mathbf{b}} = \mathcal{S}^{(v,t)}_{\mathbf{a},\mathbf{b}}$. It remains to show that the correct value is stored in $\mathrm{DP}'[v, i+1, \mathbf{a}, \mathbf{b}]$.

Now, assume as an induction hypothesis that in $\mathrm{DP}[u_j, \mathbf{a}, \mathbf{b}]$ and $\mathrm{DP}'[v, i, \mathbf{a}, \mathbf{b}]$ the correct value is stored, for an internal vertex v with children $u_1, \ldots, u_t$ and $i \in [t-1]$. We first prove that if $\mathrm{DP}'[v, i+1, \mathbf{a}, \mathbf{b}] = d$, then there exists a set $S \in \mathcal{S}^{(v,i+1)}_{\mathbf{a},\mathbf{b}}$ with $PD_{\mathcal{T}_v}(S) = d$. Afterward, we prove that $\mathrm{DP}'[v, i+1, \mathbf{a}, \mathbf{b}] \geq PD_{\mathcal{T}_{v,i+1}}(S)$ for every set $S \in \mathcal{S}^{(v,i+1)}_{\mathbf{a},\mathbf{b}}$.

Let $\mathrm{DP}'[v, i+1, \mathbf{a}, \mathbf{b}] = d$. Let $\mathbf{a}'$ and $\mathbf{b}'$ be the vectors that maximize the right side of Equation (3.4) for $\mathrm{DP}'[v, i+1, \mathbf{a}, \mathbf{b}]$. By the induction hypothesis, there is a set $S_G \in S^{(v,i)}_{\mathbf{a}-\mathbf{a}', \mathbf{b}-\mathbf{b}'}$ such that $\mathrm{DP}'[v, i, \mathbf{a} - \mathbf{a}', \mathbf{b} - \mathbf{b}'] = PD_{\mathcal{T}_{v,i}}(S_G)$ and there is a set $S_F \in S^{(u_{i+1})}_{\mathbf{a}',\mathbf{b}'}$ such that $\mathrm{DP}[u_{i+1}, i, \mathbf{a}', \mathbf{b}'] = PD_{\mathcal{T}_{u_{i+1}}}(S_F)$. Define $S := S_G \cup S_F$. Then,

$$\begin{aligned} PD_{\mathcal{T}_{v,i+1}}(S) \;&=\; PD_{\mathcal{T}_{v,i}}(S_G) + PD_{\mathcal{T}_v}(S_F) && (3.5) \\ &=\; PD_{\mathcal{T}_{v,i}}(S_G) + PD_{\mathcal{T}_{u_{i+1}}}(S_F) + \lambda(vu_{i+1}) \cdot w(b^{(u_{i+1})}_{\mathrm{var}_w}, \mathbf{b}). && (3.6) \end{aligned}$$

This equals the right side of Recurrence (3.4) and we conclude $PD_{\mathcal{T}_v}(S) = d$.

Conversely, let $S \in S^{(v,i+1)}_{\mathbf{a},\mathbf{b}}$. Let S_F be the subset of projects 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 a cost of c_k and let b_ℓ be the number of projects in S_F with a survival probability of b_ℓ. Define $\mathbf{a}' = (a_1, \ldots, a_{\mathrm{var}_c - 1})$ and $\mathbf{b}' = (b_1, \ldots, b_{\mathrm{var}_w - 1})$. Then,

$$\begin{aligned}
& \mathrm{DP}'[v, i+1, \mathbf{a}, \mathbf{b}] && (3.7)\\
\geq\ & \mathrm{DP}'[v, i, \mathbf{a} - \mathbf{a}', \mathbf{b} - \mathbf{b}'] + \mathrm{DP}[u_{i+1}, \mathbf{a}', \mathbf{b}'] + \lambda(vu_{i+1}) \cdot w\left(b'^{(u_{i+1})}_{\mathrm{var}_w}, \mathbf{b}'\right) && (3.8)\\
=\ & PD_{\mathcal{T}_{v,i}}(S_G) + PD_{\mathcal{T}_{u_{i+1}}}(S_F) + \lambda(vu_{i+1}) \cdot w\left(b'^{(u_{i+1})}_{\mathrm{var}_w}, \mathbf{b}'\right) && (3.9)\\
=\ & PD_{\mathcal{T}_{v,i+1}}(S). && (3.10)
\end{aligned}$$

Inequality (3.7) follows from Recurrence (3.4). By the definition of S_F and S_G, Equation (3.9) is correct. Finally, Equation (3.10) follows from Equation (3.6).

Running time. First, we prove how many options for vectors $\mathbf{a}$ and $\mathbf{b}$ there are. Because $\sum_{i=1}^{\mathrm{var}_c - 1} a_i \leq |X|$, 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 $\mathcal{O}(|X|^{\mathrm{var}_c - 1})$ options for $\mathbf{a}$ of not containing $|X|$, such that altogether there are $\mathcal{O}(|X|^{\mathrm{var}_c - 1} + |X|) = \mathcal{O}(|X|^{\mathrm{var}_c - 1})$ options for a suitable $\mathbf{a}$. Likewise, there are $\mathcal{O}(|X|^{\mathrm{var}_w - 1})$ options for a suitable $\mathbf{b}$.

Clearly, the base cases can be computed in $\mathcal{O}(\|\mathcal{P}\|)$ time, each. Let v be an internal vertex and fix $\mathbf{a}$ and $\mathbf{b}$. For a vertex $w \in V$, we can compute in $\mathcal{O}(n)$ time the set $\mathrm{off}(w)$. It follows that $w(b^{(u_i)}_{\mathrm{var}_w}, \mathbf{b})$ can be computed in $\mathcal{O}(n + \mathrm{var}_w)$ time such that Recurrence (3.3) can be computed in $\mathcal{O}(n + \mathrm{var}_w)$ time for fixed v, i, $\mathbf{a}$, and $\mathbf{b}$. For fixed $\mathbf{a}$ and $\mathbf{b}$, there are $\mathcal{O}\left(|X|^{\mathrm{var}_c + \mathrm{var}_w - 2}\right)$ options to choose $\mathbf{a}'$ and $\mathbf{b}'$. Therefore, Recurrence (3.4) can be evaluated in $\mathcal{O}\left(|X|^{\mathrm{var}_c + \mathrm{var}_w - 2} \cdot (n + \mathrm{var}_w)\right)$ time.

Recurrence (3.3) has to be computed once for every internal vertex. Recurrence (3.4) has to be computed once for every vertex except the root. Altogether, in $\mathcal{O}\left(|X|^{2(\mathrm{var}_c + \mathrm{var}_w - 2)} \cdot (n + \mathrm{var}_w) + |X| \cdot \|\mathcal{P}\|\right)$ time all entries of the tables DP and DP$'$ can be computed. Additionally, $\mathcal{O}(|X|^{\mathrm{var}_c + \mathrm{var}_w - 2} \cdot (\mathrm{var}_c + \mathrm{var}_w))$ time is needed to check whether there are vectors $\mathbf{a}$ and $\mathbf{b}$ such that

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

Because $\mathcal{O}(n) = \mathcal{O}(|X|)$ and $\mathcal{O}(\|\mathcal{P}\|) \leq \mathcal{O}(|X| \cdot \mathrm{var}_w)$, the overall running time is $\mathcal{O}\left(|X|^{2(\mathrm{var}_c + \mathrm{var}_w - 1))} \cdot (\mathrm{var}_c + \mathrm{var}_w)\right)$ in our RAM-model. We want to declare, however, that the table entries may store numbers with an encoding length of up to $|X| \cdot \text{w-code} + \log(D)$, which is not linear in the input size. □

The algorithm of Theorem 3.1 can easily be adjusted for $0 \xrightarrow{c_i} 1$ [2]-NAP, in which the only survival probabilities are 0 and 1. In this problem, we additionally can compute $w(b^{(u_1)}_{\mathrm{var}_w}, \mathbf{b})$ faster, as it can only be 0 or 1.

Corollary 3.6. $0 \xrightarrow{c_i} 1$ [2]-NAP *can be solved in* $\mathcal{O}\left(|X|^{2(\mathrm{var}_c + 1)} \cdot \mathrm{var}_c\right)$ *time.*

As each project with a cost higher than B can be deleted, we may assume that there are no such projects. This implies that $\mathrm{var}_c \leq C{+}1 \leq B{+}1$. Thus, Theorem 3.1 also implies that GNAP is XP with respect to $C{+}\mathrm{var}_w$ and $B{+}\mathrm{var}_w$ with astronomical running times of $\mathcal{O}\left(|X|^{2(C+\mathrm{var}_w - 1)} \cdot (C + \mathrm{var}_w)\right)$ and $\mathcal{O}\left(|X|^{2(B+\mathrm{var}_w - 1)} \cdot (B + \mathrm{var}_w)\right)$, 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 $i \in [\mathrm{var}_c]$ are selected, we declare the budget that can be spent.

Theorem 3.2. GNAP *can be solved in* $\mathcal{O}\left(B^2 \cdot |X|^{2(\mathrm{var}_w - 1)} \cdot \mathrm{var}_w\right)$ *time.*

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

- S contains exactly one project of P_i for each $x_i \in \mathrm{off}(v)$,
- $\mathrm{Cost}(S) \leq k$, and
- 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 $u_1, \ldots, u_t$, a given vector $\mathbf{b}$ and integers $k \in [B]_0$ and $i \in [t]$ we define $\mathcal{S}^{(v,i)}_{k,\mathbf{b}}$ analogously, just that exactly one project of P_j is chosen for each $x_j \in \mathrm{off}(u_1) \cup \cdots \cup \mathrm{off}(u_i)$.

As in the previous proof, there are $b^{(v)}_{\mathrm{var}_w} := |\mathrm{off}(v)| - \sum_{j=1}^{\mathrm{var}_w - 1} b_j$ projects with survival probability w_{var_w}. We want entries $\mathrm{DP}[v, k, \mathbf{b}]$ to store the largest expected phylogenetic diversity $\mathrm{PD}_{T_v}(S)$ of the tree T_v, for a set $S \in \mathcal{S}^{(v)}_{k,\mathbf{b}}$ and, respectively, $\mathrm{DP}'[v, i, k, \mathbf{b}]$ to store the largest expected phylogenetic diversity $\mathrm{PD}_{T_{v,i}}(S)$ of the tree $T_{v,i}$, for a set $S \in \mathcal{S}^{(v,i)}_{k,\mathbf{b}}$. We further define the total survival probability analogously as in Equation (3.2).

Algorithm. As a base case, 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 k and $\mathbf{b}$. So, we store $\mathrm{DP}[x_i, k, \mathbf{b}] = 0$, if P_i contains a project $p = (c_t, w_\ell)$ such that $c_t \leq k$, and

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

Otherwise, store $\mathrm{DP}[x_i, k, \mathbf{b}] = -\infty$.

Let v be an internal vertex with children $u_1, \ldots, u_t$. We define

$$\mathrm{DP}'[v, 1, k, \mathbf{b}] \;=\; \mathrm{DP}[u_1, k, \mathbf{b}] + \lambda(vu_1) \cdot w\left(b^{(u_1)}_{\mathrm{var}_w}, \mathbf{b}\right). \tag{3.11}$$

To compute further values of G, we can use the recurrence

$$\begin{aligned} &\mathrm{DP}'[v, i+1, k, \mathbf{b}] \\ =\;& \max_{k', \mathbf{b}'} \mathrm{DP}'[v, i, k-k', \mathbf{b}-\mathbf{b}'] + \mathrm{DP}[u_{i+1}, k', \mathbf{b}'] + \lambda(vu_{i+1}) \cdot w\left(b^{(u_{i+1})}_{\mathrm{var}_w}, \mathbf{b}'\right). \end{aligned} \tag{3.12}$$

Herein, k and $\mathbf{b}'$ are selected to satisfy $k' \in [k]_0$ and $\mathbf{0} \leq \mathbf{b}' \leq \mathbf{b}$.

Finally, we define $\mathrm{DP}[v, \mathbf{a}, \mathbf{b}] = \mathrm{DP}'[v, t, \mathbf{a}, \mathbf{b}]$.

Return `yes` if there is a vector $\mathbf{b}$ such that $\sum_{i=1}^{\mathrm{var}_w - 1} b_i \leq |X|$, and $\mathrm{DP}[\rho, B, \mathbf{b}] \geq D$ where ρ is the root of $\mathcal{T}$. Otherwise, if no such vector $\mathbf{b}$ exists, return `no`.

Correctness and running time. The correctness and the running time can be proven analogously to the correctness and running time of the algorithm in Theorem 3.1. □

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 $\mathrm{var}_w \leq 2^{\text{w-code}}$, we conclude the following from Theorem 3.2.

Corollary 3.7. GNAP *is* `XP` *with respect to* var_w *and* w-code*, if B is bounded polynomially in the input size.*

3.3.2 Generalized Noah's Ark Problem on Stars

We now consider the special case of GNAP where the given phylogenetic tree is a star. We first show that this special case—and therefore GNAP in general—is `W`[1]-hard with respect to the number of taxa, $|X|$. This implies that the algorithm in Proposition 3.5 is tight in some sense. Afterward, we prove that most of the `FPT` and `XP` algorithms that, in Section 2.5, we presented for MCKP can also be adopted for this special case of GNAP.

Hardness

Theorem 3.3. GNAP *is* W[1]*-hard with respect to* $|X| + \Delta$*, even on ultrametric phylogenetic trees with* $\max_\lambda = \text{height}_\mathcal{T} = 1$*, and* $D = 1$*, where* Δ *is the largest degree of a vertex in the phylogenetic tree.*

Proof. Reduction. We reduce from MCKP, which by Theorem 2.5 is W[1]-hard with respect to the number of classes m. Let $\mathcal{I} = (N, \{N_1, \ldots, N_m\}, c, d, B, D)$ be an instance of MCKP. We define an instance $\mathcal{I}' = (\mathcal{T}, \mathcal{P}, B' := B, D' := 1)$ of GNAP in which the phylogenetic X-tree $\mathcal{T} = (V, E, \lambda)$ is a star with root ρ and the vertex set is $V := \{\rho\} \cup X$, with $X := \{x_1, \ldots, x_m\}$. Set $\lambda(e) := 1$ for every $e \in E$. For every class $N_i = \{a_{i,1}, \ldots, a_{i,\ell_i}\}$, define projects $p_{i,j} := (c_{i,j} := c(a_{i,j}), w_{i,j} := d(a_{i,j})/D)$ of a project list P_i for taxon x_i. The $|X|$-collection of projects $\mathcal{P}$ contains all these project lists P_i.

Correctness. Because we may assume $0 \leq d(a) \leq D$ for all $a \in N$, the survival probabilities $w_{i,j}$ are in $\mathbb{R}_{[0,1]}$ for all $i \in [m]$ and $j \in [|N_i|]$. The tree has m taxa and a maximum degree of m. The reduction is clearly computable in polynomial time, so it only remains to show the equivalence.

Let S be a solution for instance $\mathcal{I}$ and assume $S \cap N_i = \{a_{i,j_i}\}$ without loss of generality. 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 $\sum_{i=1}^m c_{i,j_i} = \sum_{i=1}^m c(a_{i,j_i}) \leq B$ and further

$$\begin{aligned} PD_\mathcal{T}(S') &= \sum_{(v,x_i)\in E} \lambda(vx_i) \cdot w_{i,j_i} \\ &= \sum_{(v,x_i)\in E} 1 \cdot d(a_{i,j})/D \\ &= \frac{1}{D} \cdot \sum_{i=1}^m d(a_{i,j}) \geq 1 = D'. \end{aligned}$$

Conversely, let $S = \{p_{1,i_1}, \ldots, p_{m,j_m}\}$ be a solution for instance $\mathcal{I}'$. We show that $S' = \{a_{1,i_1}, \ldots, a_{m,j_m}\}$ is a solution for $\mathcal{I}$. Clearly, S' contains exactly one item per class. The cost of the set S' is $c_\Sigma(S') = \sum_{i=1}^m c(a_{i,j_i}) = \sum_{i=1}^m c_{i,j_i} \leq B$. The value of S' is $d_\Sigma(S') = \sum_{i=1}^m d(a_{i,j_i}) = \sum_{i=1}^m w_{i,j_i} \cdot D = PD_\mathcal{T}(S) \cdot D \geq D$. □

By Observation 2.19, MCKP is NP-hard, even if every class contains at most two items (of which one has no cost and no value). Because the above reduction is computed in polynomial time, we conclude the following.

Corollary 3.8. $0 \xrightarrow{c_i} b_i$ [2]-NAP *is* NP*-hard, even on ultrametric phylogenetic trees with* $\text{height}_{\mathcal{T}} = \max_\lambda = 1$*, and* $D = 1$*.*

The X-tree that has been constructed in the reduction in the proof of Theorem 3.3, is a star and therefore has a relatively high degree. In the following, we show that GNAP is also W[1]-hard with respect to $|X|$ on binary phylogenetic trees.

Corollary 3.9. GNAP *is* W[1]*-hard with respect to* $|X|+D+\text{height}_{\mathcal{T}}$ *even on binary phylogenetic trees with* $\max_\lambda = 1$*.*

Proof. Reduction. We reduce from GNAP, which by Theorem 3.3 is W[1]-hard with respect to $|X|$, even if $\max_\lambda = \text{height}_{\mathcal{T}} = D = 1$. Let $\mathcal{I} = (\mathcal{T}, \mathcal{P}, B, D)$ be an instance of GNAP with $D = 1$. Define a phylogenetic tree $\mathcal{T}' := (V, E)$ as follows. Let V be the union of X and a set of vertices $\{v_1, \ldots, v_{|X|}, x^*\}$. Let the edges be $E := \{v_i x_i, v_i v_{i+1} \mid i \in [|X| - 1]\} \cup \{v_n x_n, v_n x^*\}$ and let every edge have a weight of 1. Define a project-list $P_{x^*} = (p_{*,0} := (0, 0), p_{*,1} := (1, 1))$ for x^*. Finally, let $\mathcal{I}' := (\mathcal{T}', \mathcal{P} \cup P_{x^*}, B' := B + 1, D' := |X| + 1)$ be an instance of GNAP.

Correctness. The reduction can be computed in polynomial time. We show that S is a solution for instance $\mathcal{I}$ if and only if $S' := S \cup \{p_{*,1}\}$ is a solution for instance $\mathcal{I}'$. Clearly, $\text{Cost}(S') = \sum_{p_{i,j} \in S'} c_{i,j} = c_{*,1} + \sum_{p_{i,j} \in S} c_{i,j} = \text{Cost}(S) + 1$. 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, the set S satisfies $\text{Cost}(S') \leq B + 1$ and $PD_{\mathcal{T}'}(S') \geq |X| + 1$ if and only if $\text{Cost}(S) \leq B$ and $PD_{\mathcal{T}}(S) \geq 1$. Therefore, S is a solution for $\mathcal{I}$. We can assume that a solution of $\mathcal{I}'$ contains $p_{*,1}$ because otherwise, we can exchange one project with $p_{*,1}$ to obtain a better solution. □

Algorithmic results

In Section 2.5, we presented algorithms solving MCKP. Many of these algorithms can be adopted for instances of GNAP in which the phylogenetic tree $\mathcal{T}$ is a star.

Proposition 3.10. GNAP *can be solved*

(a) in $\mathcal{O}(D \cdot 2^{\text{w-code}} \cdot \|\mathcal{P}\| + |\mathcal{I}|)$ *time,*

(b) in $\mathcal{O}(B \cdot \|\mathcal{P}\| + |\mathcal{I}|)$ *time,*

(c) in $\mathcal{O}(C \cdot \|\mathcal{P}\| \cdot |X| + |\mathcal{I}|)$ *time, or*

(d) in $\mathcal{O}(|X|^{\text{var}_c - 1} \cdot \|\mathcal{P}\| + |\mathcal{I}|)$ *time,*

if the given phylogenetic tree is a star. Herein, $\|\mathcal{P}\| = \sum_{i=1}^{|X|} |P_i|$ *is the number of projects and* $|\mathcal{I}|$ *is the size of the input.*

Proof. To see the correctness of the statement, we reduce an instance of GNAP with a phylogenetic star tree to an instance MCKP and then use algorithms presented in Section 2.5.1.

Reduction. Let $\mathcal{I} = (\mathcal{T}, \lambda, \mathcal{P}, B, D)$ be an instance of GNAP with $\text{height}_{\mathcal{T}} = 1$. We define an instance $\mathcal{I}' = (N, \{N_1, \ldots, N_{|X|}\}, c, d, B', D')$ of MCKP. Without loss of generality, each survival probability is in the form $w_i = w_i'/2^{\text{w-code}}$ with $w_i' \in [2^{\text{w-code}}]_0$. For every taxon x_i with project list P_i, we define a class N_i. We add an item $a_{i,j}$ with cost $c(a_{i,j}) := c_{i,j}$ and value $d(a_{i,j}) := w_{i,j}' \cdot \lambda(\rho x_i)$ to N_i for every project $p_{i,j} = (c_{i,j}, w_{i,j}) \in P_i$. We set $B' := B$ and $D' := D \cdot 2^{\text{w-code}}$.

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

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

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

Running time. The instance $\mathcal{I}'$ of MCKP is computed in $\mathcal{O}(|\mathcal{I}|)$ time. We observe that in $\mathcal{I}'$ the size of N equals the number of projects $\|\mathcal{P}\|$, the number of classes m is the number of taxa $|X|$, and the budget B remains unchanged. 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 $\text{var}_d \in \mathcal{O}(\text{var}_w \cdot \max_\lambda)$. By definition, $D' = D \cdot 2^{\text{w-code}}$.

Thus, after computing instance $\mathcal{I}'$, one can use any algorithm for solving MCKP of which we saw some in Section 2.5.1 to compute an optimal solution in the stated time. □

By Proposition 3.10 and Theorem 2.4, we conclude that GNAP is FPT with respect to $\mathrm{var}_c + \mathrm{var}_w + \max_\lambda$ when restricted to instances in which the phylogenetic tree is a star. In the following, we present a better algorithm—a reduction from an instance of GNAP in which the phylogenetic tree is a star to an instance of ILP-Feasibility, in which the number of variables is bound in $\mathcal{O}(2^{\mathrm{var}_c + \mathrm{var}_d} \cdot \mathrm{var}_c)$. This reduction uses a technique that was used to show that Knapsack is FPT with respect to var_c [EKMR17], which was not necessary for proving Theorem 2.4.

Theorem 3.4. *There is a reduction from instances of* GNAP *in which the phylogenetic tree is a star to instances of* ILP-Feasibility *with* $\mathcal{O}(2^{\mathrm{var}_c + \mathrm{var}_d} \cdot \mathrm{var}_c)$ *variables. Thus,* GNAP *is* FPT *with respect to* $\mathrm{var}_c + \mathrm{var}_w$ *when restricting to instances in which the phylogenetic tree is a star.*

Proof. Description. Let $\mathcal{I} = (\mathcal{T}, \lambda, \mathcal{P}, B, D)$ be an instance of GNAP in which $\mathcal{T}$ is a star and has root ρ.

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 $\mathcal{F}$ 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 $T \in \mathcal{F}$, we define m_T to 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 determine the survival probability of p when we know the cost c of p. More precisely, if c 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 $i \in [\mathrm{var}_c]$, we define the constant $w_{T,i}$ to be $-n \cdot \max_\lambda$ if $c_i \notin C$. Otherwise, let $w_{T,i} \in \mathbb{R}_{[0,1]}$ 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 $\lambda(\rho x_i) \neq \lambda(\rho x_j)$. 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 $x_{T,1}, \ldots, x_{T,m_T}$ be the taxa, such that the project lists $P_{T,1}, \ldots, P_{T,m_T}$ are of type T and $\lambda(\rho x_{T,i}) \geq \lambda(\rho x_{T,i+1})$ 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 $f_T(\ell) := \sum_{i=1}^{\ell} \lambda(\rho x_i)$.

The following describes an instance of ILP-Feasibility.

$$\sum_{T\in\mathcal{F}}\sum_{i=1}^{\mathrm{var}_c} y_{T,i}\cdot c_i \leq B \tag{3.13}$$

$$\sum_{T\in\mathcal{F}}\sum_{i=1}^{\mathrm{var}_c} w_{T,i}\cdot g_{T,i} \geq D \tag{3.14}$$

$$f_T\left(\sum_{\ell=i}^{\mathrm{var}_c} y_{T,\ell}\right) - f_T\left(\sum_{\ell=i+1}^{\mathrm{var}_c} y_{T,\ell}\right) = g_{T,i} \qquad \forall T\in\mathcal{F}, i\in[\mathrm{var}_c] \tag{3.15}$$

$$\sum_{i=1}^{\mathrm{var}_c} y_{T,i} = m_T \qquad \forall T\in\mathcal{F} \tag{3.16}$$

$$y_{T,i}, g_{T,i} \geq 0 \qquad \forall T\in\mathcal{F}, i\in[\mathrm{var}_c] \tag{3.17}$$

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 $g_{T,j}$ we store the total value of the $y_{T,j}$ most valuable edges that are incident with a taxon that has a project list of type T for $j\in[\mathrm{var}_c]$.

For each type T, the function f_T is not necessarily linear. However, for each f_T there are affine linear functions $p_T^{(1)},\ldots,p_T^{(m_T)}$ such that $f_T(i) = \min_\ell p_T^{(\ell)}(i)$ for each $i\in[m_T]$ [EKMR17].

Correctness. Observe that if $c_i\notin C$, then because we defined $d_{T,i}$ to be $-n\cdot\max_\lambda$, Inequality (3.14) would not be satisfied if $g_{T,i} > g_{T,i+1}$ and consequently $y_{T,i} = 0$ if $c_i\notin C$ for each type $T=(C,W)\in\mathcal{F}$ and $i\in[\mathrm{var}_c]$. Inequality (3.13) can only be satisfied if the total cost is at most B. The correctness of Inequality (3.14): The variable $g_{T,i}$ stores is 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 a have survival probability of $w_{T,i}$, and thus the phylogenetic diversity of these projects is $w_{T,i}\cdot(g_{T,i}-g_{T,i+1})$. Consequently, Inequality (3.14) can only be satisfied if the total phylogenetic diversity is at least D. Equation (3.15) ensures that the value of $g_{T,i}$ is chosen correctly. Equation (3.16) can only be correct if exactly m_T projects are picked from the project lists of type T, for each $T\in\mathcal{F}$. It remains to show that the instance of the ILP-Feasibility has $\mathcal{O}(2^{\mathrm{var}_c+\mathrm{var}_d}\cdot\mathrm{var}_c)$ variables. Because $\mathcal{F}\subseteq 2^{\mathcal{C}}\times 2^{\mathcal{W}}$, the size of $\mathcal{F}$ is $\mathcal{O}(2^{\mathrm{var}_c+\mathrm{var}_d})$. We follow 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 $g_{T,i}$. □

3.4 Restriction to Two Projects per Taxon

We finally study two special cases of $a_i \xrightarrow{c_i} b_i$ [2]-NAP—the special case of GNAP where every project list contains exactly two projects. In this section, we write $c(x_i)$ for the cost of the project with survival probability 1 in P_i.

3.4.1 Sure Survival or Extinction For Each Project

First, we consider $0 \xrightarrow{c_i} 1$ [2]-NAP, the special case where each taxon x_i survives definitely if c_i is paid and becomes extinct, otherwise. This special case was introduced by Pardi and Goldman [PG07] under the name BUDGETED NAP. They also presented a pseudopolynomial-time algorithm that computes a solution for an instance in $\mathcal{O}(B^2 \cdot n)$ time. Because we may assume that $B \leq C \cdot |X|$, we conclude the following.

Corollary 3.11. $0 \xrightarrow{c_i} 1$ [2]-NAP *can be solved in* $\mathcal{O}(C^2 \cdot n^3)$ *time.*

We observe that $0 \xrightarrow{c_i} 1$ [2]-NAP is FPT with respect to D, with an adaption of the above-mentioned algorithm of Pardi and Goldman [PG07] for the parameter B.

Proposition 3.12. $0 \xrightarrow{c_i} 1$ [2]-NAP *can be solved in* $\mathcal{O}(D^2 \cdot n)$ *time.*

Proof. Table definition. 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, b)-*respecting set*, if $PD_{\mathcal{T}_v}(S) \geq d$, and S contains exactly one project of the projects lists of the offspring of A, and S contains at least b projects with survival probability 1.

We describe a dynamic programming algorithm with two tables DP and DP′. We want entry $\mathrm{DP}[v, d, b]$ for a vertex $v \in V$, an integer $d \in [D]_0$, and a boolean $b \in \{0, 1\}$ to store the minimal cost of a $(\{v\}, v, d, b)$-respecting set. If v is an internal vertex with children $u_1, \ldots, u_t$ and $i \in [t]$, then we want entry $\mathrm{DP}'[v, i, d, b]$ to store the minimal cost of an $(\{u_1, \ldots, u_i\}, v, d, b)$-respecting set.

We use non-negative subtraction, $-_{\geq 0}$, which for integers a and b is $a -_{\geq 0} b = a - b$ if $a \geq b$ and $a -_{\geq 0} b = 0$, otherwise.

Algorithm. As a base case, for a leaf x_i store $\mathrm{DP}[x_i, 0, 1] = c(x_i)$ and for each $d > 0$ store $\mathrm{DP}[x_i, d, 1] = \infty$. For any vertex v and each $d \in [D]_0$ store $\mathrm{DP}[v, d, 0] = 0$ and $\mathrm{DP}[v, i, d, 0] = 0$.

For an internal vertex v, we define $\mathrm{DP}'[v, 1, d, 1] = \mathrm{DP}[u_1, d -_{\geq 0} \lambda(vu_1), 1]$.

Now, let v be an internal vertex with children $u_1, \ldots, u_t$ and we assume that for a fixed $i \in [t]$ the values of $\mathrm{DP}'[v, i, d, b]$ and $\mathrm{DP}[u_{i+1}, d, b]$ are known for each $d \in [D]_0$

and $b \in \{0,1\}$. To compute the value of $\mathrm{DP}'[v, i+1, d]$, we use the recurrence

$$\begin{aligned} & \mathrm{DP}'[v, i+1, d, 1] \\ = \; & \min\{\mathrm{DP}'[v, i, d, 1]; \min_{d' \in [d_{i+1}]} \{\mathrm{DP}'[v, i, d_{i+1} - d', 1] + \mathrm{DP}[u_{i+1}, d', 1]\}\}, \end{aligned} \tag{3.18}$$

where $d_{i+1} := d -_{\geq 0} \lambda(vu_{i+1})$. We store $\mathrm{DP}[v, d, 1] = \mathrm{DP}'[v, t, d, 1]$, eventually.

We return `yes` if $\mathrm{DP}[\rho, D] \leq B$ for the root ρ of $\mathcal{T}$. Otherwise, we return `no`.

Correctness. The base case, as well as the computation of $\mathrm{DP}'[v, 1, d, 1]$, and the computation of $\mathrm{DP}[v, d, 1]$ for an internal vertex v and integers $d \in [D]_0$ are correct by definition. It remains to show that $\mathrm{DP}'[v, i+1, d, 1]$ stores the correct value if $\mathrm{DP}'[v, i, d, 1]$ and $\mathrm{DP}[u_{i+1}, d, 1]$ store the correct value. We first show that if S is an $(\{u_1, \ldots, u_{i+1}\}, v, d, 1)$-respecting set, then $\mathrm{DP}'[v, i+1, d, 1] \leq \mathrm{Cost}(S)$. Afterwards, we show that if c is stored in $\mathrm{DP}'[v, i+1, d, 1]$, then there is an $(\{u_1, \ldots, u_{i+1}\}, v, d, 1)$-respecting set S with $\mathrm{Cost}(S) = c$.

Let S be an $(\{u_1, \ldots, u_{i+1}\}, v, d, 1)$-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}. We define $S_2 := S \setminus S_1$. Then $\mathrm{DP}'[v, i+1, d, 1] = \mathrm{DP}'[v, i, d, 1]$, if S_2 only contains projects with a survival probability of 0. 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_2 is an $(\{u_1, \ldots, u_i\}, v, d - d', 1)$-respecting set, and S_1 is an $(\{u_{i+1}\}, v, d', 1)$-respecting set. Consequently,

$$\begin{aligned} \mathrm{DP}'[v, i+1, d, 1] &\leq \mathrm{DP}'[v, i, d - d', 1] + \mathrm{DP}[u_{i+1}, d' -_{\geq 0} \lambda(vu_{i+1}), 1] && (3.19) \\ &\leq \mathrm{Cost}(S_G) + \mathrm{Cost}(S_F) = \mathrm{Cost}(S). && (3.20) \end{aligned}$$

Here, Inequality (3.19) follows from the recurrence in Recurrence (3.18) and Inequality (3.20) follows from the induction hypothesis.

Conversely, we assume that $\mathrm{DP}'[v, i+1, d, 1]$ stores c. Unless $\mathrm{DP}'[v, i+1, d, 1]$ takes the value of $\mathrm{DP}'[v, i, d, 1]$ there is an integer $d' \in [d_{i+1}]_0$, such that $\mathrm{DP}'[v, i+1, d, 1] = \mathrm{DP}'[v_{,i+1}, d_{i+1} - d', 1] + \mathrm{DP}[u_{i+1}, d', 1]$. By the induction hypothesis, there is an $(\{u_1, \ldots, u_i\}, v, d - d', 1)$-respecting set S_2 and an $(\{u_{i+1}\}, v, d', 1)$-respecting set S_1 such that $\mathrm{DP}[u_{i+1}, d', 1] = \mathrm{Cost}(S_1)$ and $\mathrm{DP}'[v, i, d - d', 1]$ stores $\mathrm{Cost}(S_2)$. We conclude that $S := S_1 \cup S_2$ is an $(\{u_1, \ldots, u_{i+1}\}, v, d, 1)$-respecting set and $c = \mathrm{Cost}(S_1) + \mathrm{Cost}(S_2) = \mathrm{Cost}(S)$.

Running time. Table DP has $\mathcal{O}(D \cdot n)$ entries and each entry can be computed in constant time. Also, DP' has $\mathcal{O}(D \cdot n)$ entries. Entry $\mathrm{DP}'[v, i+1, d]$ 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 $\max_\lambda$: For an instance $\mathcal{I}$ of $0 \xrightarrow{c_i} 1$ [2]-NAP with $\sum_{e\in E} \lambda(e) < D$, we can return `no` since the desired diversity can never be reached. Otherwise, we may assume $D \leq \sum_{e\in E} \lambda(e) \leq \max_\lambda \cdot (n-1)$. This gives the following running time-bound.

Corollary 3.13. $0 \xrightarrow{c_i} 1$ [2]-NAP *can be solved in* $\mathcal{O}((\max_\lambda)^2 \cdot n^3)$ *time.*

3.4.2 Unit Costs For Each Project

Next, we consider UNIT-COST-NAP—the special case of GNAP in which every project with a positive survival probability has the same cost. Observe, every instance $\mathcal{I} = (\mathcal{T}, \mathcal{P}, B, D)$ of $0 \xrightarrow{c} b_i$ [2]-NAP for each $c \in \mathbb{N}$ can be reduced to an equivalent instance $\mathcal{I}' = (\mathcal{T}, \mathcal{P}', B', D)$ of $0 \xrightarrow{1} b_i$ [2]-NAP by setting the costs of every project with a positive survival probability to 1, and $B' = \lfloor B/c \rfloor$. Thus, the problem $0 \xrightarrow{1} b_i$ [2]-NAP can be considered as UNIT-COST-NAP.

In the remainder of this chapter, we use the term *solution* to denote only those projects with a cost of 1 which have been chosen. Further, with $w(x_i)$ we denote the survival probability of the project in P_i which a costs of 1.

We show that even in very restricted instances, UNIT-COST-NAP is `NP`-hard by a reduction from PENALTY SUM. Recall that in PENALTY SUM we are given a set of tuples $T = \{t_i = (a_i, b_i) \mid i \in [n], a_i \in \mathcal{Q}_{\geq 0}, b_i \in \mathbb{R}_{(0,1)}\}$, two integers k, and Q, and a number $D \in \mathbb{Q}_+$. It is asked whether there is a set of k tuples $S \subseteq T$, such that $\sum_{t_i\in S} a_i - Q \cdot \prod_{t_i\in S} b_i \geq D$. PENALTY SUM is `NP`-hard by Theorem 2.6.

Theorem 3.5. UNIT-COST-NAP *is* `NP`*-hard even on instances with a phylogenetic tree with a height of 2 and a degree of 1 in the root.*

Proof. Reduction. Let $\mathcal{I} = (T, k, Q, D)$ be an instance of PENALTY SUM. Let $bin(a)$ and $bin(1-b)$ be the maximum binary encoding length of a_i and $1-b_i$, respectively, and define t to be $bin(a) + bin(1-b)$. We define an instance $\mathcal{I}' = (\mathcal{T}, \mathcal{P}, B, D')$ of UNIT-COST-NAP as follows. Let $\mathcal{T}$ contain the vertices $V := \{\rho, v, x_1, \ldots, x_{|T|}\}$ and let the underlying undirected graph of $\mathcal{T}$ be a star with center v. Therefore, v is the only child of the root ρ and $x_1, \ldots, x_{|T|}$ are the leaves. We define $\lambda(\rho v) = 2^t Q$ and $\lambda(v x_i) = 2^t(a_i/1-b_i)$ 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 $D' := 2^t(D+Q)$. The reduction can clearly be computed in polynomial time.

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

First, let S be a solution of an instance $\mathcal{I}$ of Penalty Sum. We define a set $S' := \{(1, w(x_i)) \mid t_i \in S\}$. Then,

$$\begin{aligned}
&\sum_{x_i \in S'} \lambda(vx_i) \cdot w(x_i) + \lambda(\rho v) \cdot \left(1 - \prod_{x_i \in S'} (1 - w(x_i))\right) \\
= &\sum_{t_i \in S'} 2^t \cdot ({}^{a_i}/_{1-b_i}) \cdot (1 - b_i) + 2^t Q \cdot \left(1 - \prod_{t_i \in S'} (1 - (1 - b_i))\right) \\
= &\sum_{t_i \in S} 2^t a_i - 2^t Q \cdot \prod_{t_i \in S} b_i + 2^t Q \geq 2^t \cdot (D + Q) = D'.
\end{aligned}$$

Then, S' is a solution for the instance $\mathcal{I}'$, because $|S'| = |S| \leq B$.

Consequently, let S' be a solution for the instance $\mathcal{I}'$ of unit-cost-NAP. We conclude $\sum_{x_i \in S'} \lambda(vx_i) \cdot w(x_i) \geq D' - \lambda(\rho v) \cdot \left(1 - \prod_{x_i \in S'}(1 - w(x_i))\right)$. Define $S \subseteq T$ to contain a tuple t_i if and only if S' contains a project of the taxon x_i.

Further, observe that $a_i = 2^{-t} \cdot \lambda(vx_i) \cdot (1 - b_i)$. Then,

$$\begin{aligned}
&\sum_{t_i \in S} a_i - Q \cdot \prod_{t_i \in S} b_i \\
= &\sum_{x_i \in S'} 2^{-t} \cdot \lambda(vx_i) \cdot (1 - b_i) - 2^{-t} \cdot \lambda(\rho v) \cdot \prod_{x_i \in S'} (1 - w(x_i)) \\
= &\ 2^{-t} \cdot \left(\sum_{x_i \in S'} \lambda(vx_i) \cdot w(x_i) - \lambda(\rho v) \cdot \prod_{x_i \in S'} (1 - w(x_i))\right) \\
\geq &\ 2^{-t} \cdot \left(D' - \lambda(\rho v) \cdot \left(1 - \prod_{x_i \in S'} (1 - w(x_i))\right) - \lambda(\rho v) \cdot \prod_{x_i \in S'} (1 - w(x_i))\right) \\
= &\ 2^{-t} \cdot (D' - \lambda(\rho v)) = 2^{-t} \cdot (2^t (D + Q) - 2^t Q) = D.
\end{aligned}$$

Because $|S'| = |S| \leq B$, we conclude that S is a solution of instance $\mathcal{I}$. □

Recall that in an ultrametric tree, the weighted distance from the root to a vertex is the same for all vertices. Observe that in an instance of unit-cost-NAP in which the phylogenetic tree is ultrametric and has a height of at most 2, and the root has only one child one can select the set of taxa that have the highest survival probability. Therefore, we can solve general instances of unit-cost-NAP with an ultrametric phylogenetic tree that has a height of at most 2 with a dynamic

programming algorithm which assigns each child of the root the number of taxa that are saved in their offspring. We conclude the following.

Observation 3.14. UNIT-COST-NAP *can be solved in polynomial time on instances in which the phylogenetic tree is ultrametric and has a height of at most 2.*

In the following theorem, we show that, however, UNIT-COST-NAP is NP-hard even when restricted to ultrametric phylogenetic trees with a height of 3.

Theorem 3.6. UNIT-COST-NAP *is* NP*-hard even if the given phylogenetic tree is ultrametric and has a height of at most 3.*

Proof. By Theorem 3.5, it suffices 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 ρ of $\mathcal{T}$ has only one child v and the height of $\mathcal{T}$ is 2. Without loss of generality, assume $\lambda(vx_i) \geq \lambda(vx_{i+1})$ for each $i \in [|X| - 1]$ 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 3.5 we may assume $w(x_j) \neq 1$ for each $x_j \in X$.

Consider Figure 3.1 for an illustration of this reduction.

We define an instance $\mathcal{I}' := (\mathcal{T}', \mathcal{P}, B', D')$ of UNIT-COST-NAP. Let $X_1 \subseteq X$ be the set of vertices x_i with $\lambda(vx_1) = \lambda(vx_i)$. If $X_1 = X$, then $\mathcal{I}$ is already ultrametric and simply output $\mathcal{I}$. Otherwise, define $X_2 := X \setminus X_1$. Fix an integer $t \in \mathbb{N}$ that is large enough such that $1 - 2^{-t} > w_{s,1}$ and $2^t \cdot \lambda(vx_{|X|}) > \lambda(vx_1)$. Define a tree $\mathcal{T}' = (V', E', \lambda')$, in which V' contains the vertices V and add two vertices u_i and x_i^* for every $x_i \in X_2$. Let X^* be the set of leaves x_i^*. The set of edges is defined by $E' = \{\rho v\} \cup \{vx_i \mid x_i \in X_1\} \cup \{vu_i, u_i x_i, u_i x_i^* \mid x_i \in X_2\}$. Observe that the leaf set of $\mathcal{T}$ is $X \cup X^*$. The weights of the edges are defined as:

- We set $\lambda'(\rho v) = \lambda(\rho v) \cdot 2^{t \cdot |X_2|} \cdot (2^t - 1)$.
- For $x_i \in X_1$ we set $\lambda'(vx_i) = (2^t - 1) \cdot \lambda(vx_1)$.
- For $x_i \in X_2$ we set $\lambda'(vu_i) = 2^t \cdot (\lambda(vx_1) - \lambda(vx_i))$ and $\lambda'(u_i x_i) = \lambda'(u_i x_i^*) = (2^t \cdot \lambda(vx_i)) - \lambda(vx_1)$.

Define project lists $P_i^* := ((0,0),(1,1-2^{-t}))$ for each taxa $x_i^* \in X^*$. Then, define $\mathcal{P}' := \mathcal{P} \cup \{P_i^* \mid x_i^* \in X^*\}$. Finally, we set $B' := B + |X^*|$ and

$$D' = (2^t - 1) \cdot \left(D + |X_2| \cdot (2^t - 1) \cdot \lambda(vx_1) + \left(2^{t|X_1|} - 1\right) \cdot \lambda(wv)\right).$$

Correctness. Observe that t can be chosen to be in $\mathcal{O}(\text{w-code} + \max_\lambda)$. Hence, the reduction can be computed in polynomial time. Moreover, the new phylogenetic tree

has a height of 3. Before showing that $\mathcal{I}$ is a `yes`-instance if and only if $\mathcal{I}'$ is a `yes`-instance, we first prove that $\mathcal{T}'$ is ultrametric.

To show that $\mathcal{T}'$ is ultrametric, for each $x_i \in X$ and $x_i^* \in X^*$ we show that the paths from v to x_i and from v to x_i^* 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

$$\begin{aligned}\lambda'(vu_i) + \lambda'(u_ix_i) &= 2^t \cdot (\lambda(vx_1) - \lambda(vx_i)) + 2^t \cdot \lambda(vx_i) - \lambda(vx_1) \\ &= (2^t - 1) \cdot \lambda(vx_1) = \lambda'(vx_1).\end{aligned}$$

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. 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.

Let $S \subseteq X$ be a set of taxa. Define $S' := S \cup X_i^*$. 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 $|S| \leq B$ if and only if $|S'| \leq B' = B + |X^*|$. We compute the phylogenetic diversity of S' in $\mathcal{T}'$. Here, we first consider the diversity from the subtree that consists of v, u_i, x_i, and x_i^* for both options, whether S' contains x_i or not. Afterward, we consider the additional value from the edge ρv.

If S' contains x_i^* but not x_i, the contribution is

$$\begin{aligned}&\sum_{x_i \in X_2 \setminus S} (\lambda'(vu_i) + \lambda'(u_ix_i)) \cdot w'(x_i^*) \\ = &\sum_{x_i \in X_2 \setminus S} (2^t \cdot (\lambda(vx_1) - \lambda(vx_i)) + 2^t \cdot \lambda(vx_i) - \lambda(vx_1)) \cdot (1 - 2^{-t}) \\ = &\sum_{x_i \in X_2 \setminus S} \lambda(vx_1) \cdot (2^t - 1) \cdot (1 - 2^{-t}).\end{aligned}$$

For those vertices where S' contains x_i and x_i^*, the contributed phylogenetic

diversity is

$$
\begin{aligned}
& \sum_{x_i \in X_2 \cap S} \lambda'(vu_i) \cdot (1 - (1 - w'(x_i^*)) \cdot (1 - w'(x_i))) \\
& + \sum_{x_i \in X_2 \cap S} \lambda'(u_i x_i) \cdot w'(x_i) + \sum_{x_i \in X_2 \cap S} \lambda'(u_i x_i^*) \cdot w'(x_i^*) \\
= & \sum_{x_i \in X_2 \cap S} 2^t \cdot (\lambda(vx_1) - \lambda(vx_i)) \cdot (1 - 2^{-t} \cdot (1 - w(x_i))) \\
& + \sum_{x_i \in X_2 \cap S} (2^t \cdot \lambda(vx_i) - \lambda(vx_1)) \cdot (w(x_i) + 1 - 2^{-t}) \\
= & \sum_{x_i \in X_2 \cap S} (\lambda(vx_1) - \lambda(vx_i)) \cdot (2^t - (1 - w(x_i))) \\
& + \sum_{x_i \in X_2 \cap S} (2^t \cdot \lambda(vx_i) - \lambda(vx_1)) \cdot (w(x_i) + 1 - 2^{-t}) \\
= & \sum_{x_i \in X_2 \cap S} \lambda(vx_1) \cdot (2^t - (1 - w(x_i)) - w(x_i) - 1 + 2^{-t})) \\
& + \lambda(vx_i) \cdot (-2^t + (1 - w(x_i)) + 2^t(w(x_i) + 1 - 2^{-t})) \\
= & \sum_{x_i \in X_2 \cap S} \lambda(vx_1) \cdot (2^t - 2 + 2^{-t}) + \lambda(vx_i) \cdot ((2^t - 1) \cdot w(x_i)) \\
= & (2^t - 1) \cdot \left(\sum_{x_i \in X_2 \cap S} \lambda(vx_1) \cdot (1 - 2^{-t}) + \lambda(vx_i) \cdot w(x_i) \right).
\end{aligned}
$$

Finally, for the edge ρv the contribution is

$$
\begin{aligned}
& \lambda'(\rho v) \cdot \left(1 - \prod_{x_i^* \in X^*} (1 - w'(x_i^*)) \cdot \prod_{x_i \in S} (1 - w'(x_i)) \right) \\
= & \lambda(\rho v) \cdot 2^{t \cdot |X_2|} \cdot (2^t - 1) \cdot \left(1 - 2^{-t \cdot |X_2|} \cdot \prod_{x_i \in S} (1 - w(x_i)) \right) \\
= & \lambda(\rho v) \cdot 2^{t \cdot |X_2|} \cdot (2^t - 1) - \lambda(\rho v) \cdot (2^t - 1) \cdot \prod_{x_i \in S} (1 - w(x_i)).
\end{aligned}
$$

Altogether, we conclude

$$
\begin{aligned}
& PD_{\mathcal{T}'}(S') \\
= \; & \lambda'(\rho v) \cdot \prod_{x_i^* \in X^*} (1 - w'(x_i^*)) \cdot \prod_{x_i \in S} (1 - w'(x_i)) \\
& + \sum_{x_i \in X_1 \cap S} \lambda'(v x_1) \cdot w'(x_i) \\
& + \sum_{x_i \in X_2 \cap S} \lambda'(v u_i) \cdot (1 - (1 - w'(x_i^*)) \cdot (1 - w'(x_i))) \\
& + \sum_{x_i \in X_2 \cap S} \lambda'(u_i x_i) \cdot w'(x_i) + \sum_{x_i \in X_2 \cap S} \lambda'(v x_1) \cdot (1 - 2^t) \\
& + \sum_{x_i \in X_2 \setminus S} (\lambda'(v u_i) + \lambda'(u_i x_i)) \cdot w'(x_i^*) \\
= \; & \lambda(\rho v) \cdot 2^{t \cdot |X_2|} \cdot (2^t - 1) - \lambda(\rho v) \cdot (2^t - 1) \cdot \prod_{x_i \in S} (1 - w(x_i)) \\
& + \sum_{x_i \in X_1 \cap S} (2^t - 1) \cdot \lambda(v x_1) \cdot w(x_i) \\
& + (2^t - 1) \cdot \left(\sum_{x_i \in X_2 \cap S} \lambda(v x_1) \cdot (1 - 2^{-t}) + \lambda(v x_i) \cdot w(x_i) \right) \\
& + \sum_{x_i \in X_2 \setminus S} \lambda(v x_1) \cdot (2^t - 1) \cdot (1 - 2^{-t}) \\
= \; & (2^t - 1) \cdot \Bigg[\lambda(\rho v) \cdot 2^{t \cdot |X_2|} - \lambda(\rho v) \cdot \prod_{x_i \in S} (1 - w(x_i)) \\
& + \sum_{x_i \in S} \lambda(v x_i) \cdot w(x_i) + \sum_{x_i \in X_2} \lambda(v x_1) \cdot (1 - 2^{-t}) \Bigg] \\
= \; & (2^t - 1) \cdot \Bigg[\lambda(\rho v) \cdot \left(1 - \prod_{x_i \in S} (1 - w(x_i)) \right) + \sum_{x_i \in S} \lambda(v x_i) \cdot w(x_i) \\
& + \lambda(\rho v) \cdot (2^{t \cdot |X_2|} - 1) + \sum_{x_i \in X_2} \lambda(v x_1) \cdot (1 - 2^{-t}) \Bigg] \\
= \; & (2^t - 1) \cdot \left[PD_{\mathcal{T}}(S) + (2^{t \cdot |X_2|} - 1) \cdot \lambda(\rho v) + |X_2| \cdot (1 - 2^{-t}) \cdot \lambda(v x_1) \right].
\end{aligned}
$$

It follows from this equation that $PD_{\mathcal{T}'}(S') \leq D'$ if and only if $PD_{\mathcal{T}}(S) \leq D$.

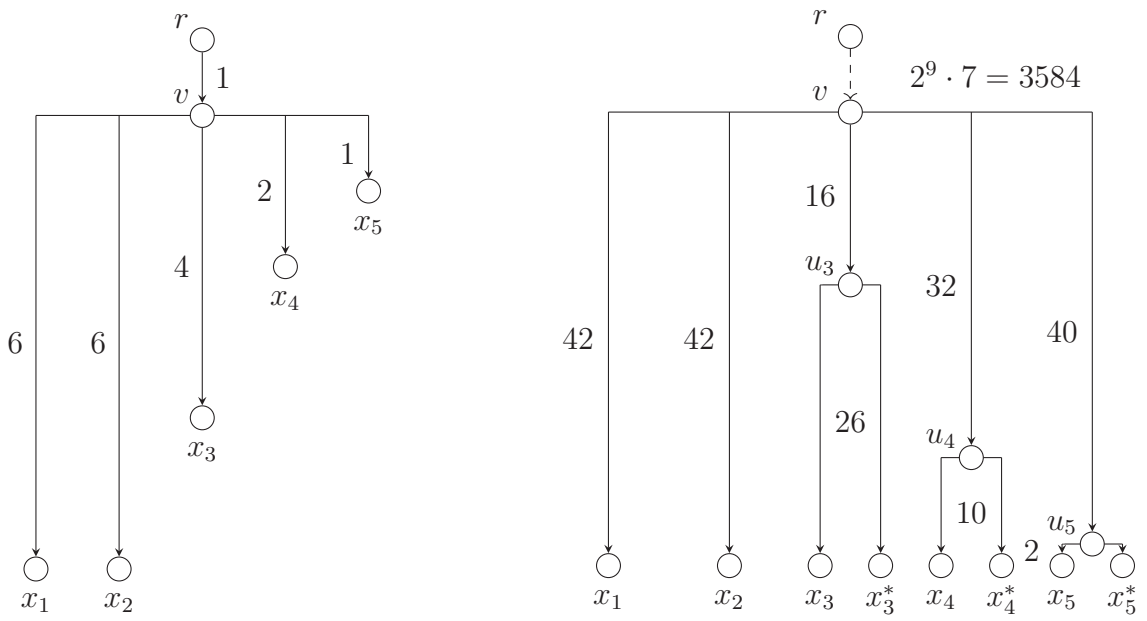


Figure 3.1: This figure shows an example of the reduction presented in Theorem 3.6, where on the left side the tree of an example-instance $\mathcal{I}$ and on the right side the tree of instance $\mathcal{I}'$ is depicted. Here, the survival probabilities are omitted and we used $t = 3$.

Conversely, we show that $\mathcal{I}'$ has a solution S' with $X^* \subseteq S'$. This then implies that $S' \setminus X^*$ is a solution for $\mathcal{I}$ by the argument we saw beforehand.

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. Assume otherwise and 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 is in S', then we consider the set $S_1 := (S' \setminus \{x_i\}) \cup \{x_i^*\}$. Now, $|S_1| = |S'| \leq B'$ and because we defined that $w(x_i^*) = 1 - 2^{-t} > w(x_i)$, we conclude that $PD_{\mathcal{T}'}(S_1) > PD_{\mathcal{T}'}(S') \geq D'$. If $S' \subsetneq X^*$, then we 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, $w(x_i^*) = 1 - 2^{-t} > w(x_i)$ 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, $PD_{\mathcal{T}'}(S_3) > PD_{\mathcal{T}'}(S') \geq D'$. This completes the proof. □

3.5 Discussion

In this chapter, we considered GNAP and three special cases of GNAP—namely, $0 \xrightarrow{c_i} 1$ [2]-NAP, UNIT-COST-NAP, and the case that the phylogenetic tree is a star—and we provided several tractability and intractability results. Our most important results are as follows. GNAP is W[1]-hard with respect to the number of taxa but can be solved with an XP-algorithm when parameterized by the number of unique costs plus survival probabilities, $\text{var}_c + \text{var}_w$. We presented some pseudo-polynomial running-time algorithms for $a_i \xrightarrow{c_i} b_i$ [2]-NAP and GNAP restricted to trees is a star. We finally showed that UNIT-COST-NAP is NP-hard even in very restricted cases.

Naturally, several open questions remain. We want to list some of them.

Most notably, we do not known whether GNAP is weakly or strongly NP-hard. We therefore can not exclude a pseudo-polynomial running-time algorithm for GNAP yet. While we showed that PENALTY SUM can be solved in pseudo-polynomial running-time by Proposition 2.21, it remains open whether such an algorithm exists even for UNIT-COST-NAP.

Moreover, it remains open whether the result of Theorem 3.1 can be improved so that GNAP is FPT with respect to $\text{var}_c + \text{var}_w$.

In Observation 3.4, we presented an easy reduction to show that GNAP—or more precisely even the special case $0 \xrightarrow{c_i} b$ [2]-NAP—is already NP-hard when $D = 1$. In this reduction, we, however, increase the encoding-length of each survival probability. We therefore wonder if $0 \xrightarrow{c_i} b$ [2]-NAP or even GNAP are FPT when parameterized with D plus the binary encoding size of the survival probabilities. In Proposition 3.10, we showed that this holds at least in the case that the phylogenetic tree has a size of 1.

Chapter 4

Phylogenetic Diversity with Extinction Times

4.1 Introduction

In this chapter, we consider an extension of the classic Maximize Phylogenetic Diversity (Max-PD) problem, in which species (taxa) have differing *extinction times*, after which they will die out if they have not been saved. In Max-PD, we are given a phylogenetic tree $\mathcal{T}$ and integers k, and D as input, and we are asked if there exists a subset of k taxa whose phylogenetic diversity is at least D. Max-PD is polynomial-time solvable by a greedy algorithm [Ste05, PG05]. However, already the generalization Budgeted NAP of Max-PD, in which each taxon has an associated integer cost which would be necessary to be paid to save the taxon, is NP-hard [PG07].

In this extension of Max-PD, the cost of a taxon may, just as well as financial or space capacities, be used to represent the fact that different taxa may take a different amount of time to save from extinction. However, to the best of our knowledge, previously studied versions of Max-PD do not take into account that different taxa may have different amounts of *remaining* time before extinction. Thus, to ensure that a set of taxa can be saved with the available resources, it is not enough to guarantee that their total cost is below a certain threshold. One also needs to ensure that there is a schedule under which each taxon is saved before its moment of extinction.

We take the first step in addressing this issue by introducing two extensions of Budgeted NAP, denoted Time Sensitive Maximization of Phylogenetic Diversity (Time-PD) and Strict Time Sensitive Maximization of Phylogenetic Diversity (s-Time-PD), in which each taxon has an associated *rescue length* (the amount of time it takes to save the taxon) and also an *extinction time*

(the time after which the taxon can not be saved anymore). In each problem, there is a set of available teams that can work towards saving the taxa; under TIME-PD different teams may collaborate on saving a taxon while in S-TIME-PD they may not.

These problems have much in common with machine scheduling problems, insofar as we may think of the taxa as corresponding to jobs with a certain due date and the teams as corresponding to machines. One may think of TIME-PD and S-TIME-PD as machine scheduling problems, in which the objective to be maximized is the phylogenetic diversity (as determined by the input tree $\mathcal{T}$) of the set of completed tasks (which are the saved taxa).

Related Work in Scheduling. Scheduling problems are denoted by a three-field notation introduced by Graham et al. [GLLK79]. Herein, many problems are written as a triple $\alpha|\beta|\gamma$, where α is the machine environment, β are job characteristics and scheduling constraints, and γ is the objective function. For a more detailed view, we refer to [GLLK79, MVB18].

The scheduling problem most closely related to the problems studied in this chapter is $Pt||\sum w_j(1-U_j)$. In this problem, we are given a set of jobs, each with an integer weight, a processing time, and a due date. We are also given t identical machines that can each process one job at a time. The task is to schedule jobs on the available machines in such a way that we maximize the total weight of jobs completed before the due date. (Here w_j denotes the weight of job j, and $U_j = 1$ if job j is not completed on time, and $U_j = 0$, otherwise). This is similar to S-TIME-PD, in the case that the t available teams have identical starting and ending times. We may think of taxa as analogous to jobs, with the extinction times corresponding to due dates. The key difference is that in S-TIME-PD, rather than maximizing the total weight of the completed jobs, we aim to maximize their phylogenetic diversity, as determined by the input tree $\mathcal{T}$.

Although approximation algorithms for scheduling problems are common, parameterized algorithms for scheduling problems are rare [MVB18]. The most commonly investigated special case of S-TIME-PD is $1||\sum w_j(1-U_j)$—that is $Pt||\sum w_j(1-U_j)$ with only one machine. This problem is weakly NP-hard and is solvable in pseudo-polynomial time [LM69, Sah76], while the unweighted version $1||\sum(1-U_j)$ is solvable in polynomial time [Moo68, Max70, Sid73]. Parameters studied for $1||\sum w_j(1-U_j)$ include the number of different due dates, the number of different processing times, and the number of different weights. The problem is FPT when parameterized by the sum of any two of these parameters [HKPS21]. When parameterized by one of the latter two parameters, it is W[1]-hard [HH24]

but XP [HKPS21]. Also a version has been studied in the light of parameterized algorithms in which there are also few distinct deadlines [HHS23].

Our Contribution. With Time-PD and s-Time-PD we introduce problems in maximizing phylogenetic diversity in which extinction times of taxa may vary. We further provide a connection to the well-regarded field of scheduling.

Both problems turn out to be NP-hard; this result is perhaps unsurprising given their close relation to scheduling problems, and we therefore analyze Time-PD and s-Time-PD within the framework of parameterized complexity. Our most important results are $\mathcal{O}^*(2^{2.443 \cdot D + o(D)})$-time algorithms for Time-PD and s-Time-PD (Section 4.3), and a $\mathcal{O}^*(2^{6.056 \cdot \overline{D} + o(\overline{D})})$-time algorithm for Time-PD (Section 4.4). Moreover, both problems are FPT with respect to the available person-hours $H_{\max_{\text{ex}}}$ (Proposition 4.15). A detailed list of known results for Time-PD and s-Time-PD is given in Table 4.1. In the table and the rest of the chapter we use the convention $n := |X|$.

A key challenge for the design of FPT algorithms for Time-PD and s-Time-PD is the combination of the tree structure and extinction time constraints. Tree-based problems often suit a dynamic programming (DP) approach, where partial solutions are constructed for subtrees of the input tree, (starting with the individual leaves and proceeding to larger subtrees). Scheduling problems with due dates (such as our extinction times) may also suit a DP approach, where partial solutions are constructed for subsets of tasks with due dates below some bound. These two DP approaches have a conflicting structure for the desired processing order, which makes designing a DP algorithm for Time-PD and s-Time-PD more difficult than it may first appear. Our solution involves the careful use of color coding to reconcile the two approaches. We believe that this approach may also be applicable to other extensions of Max-PD.

Structure of the Chapter. In Section 4.2, we formally define the two problems Time-PD and s-Time-PD and prove some simple initial results. In Section 4.3, we introduce an FPT algorithm for Time-PD and s-Time-PD parameterized by the target D, and in Section 4.4, we give an FPT algorithm for Time-PD parameterized by the acceptable loss of diversity $\overline{D}$. In Section 4.5, we prove a number of other parameterized algorithms. In Section 4.6, we provide a brief outlook on future research ideas.

Table 4.1: Parameterized complexity results for Time-PD and s-Time-PD.

Parameter	Time-PD		s-Time-PD	
Diversity D	FPT $\mathcal{O}^*(2^{2.443\cdot D+o(D)})$	Thm. 4.1	FPT $\mathcal{O}^*(2^{2.443\cdot D+o(D)})$	Thm. 4.1
Diversity loss $\overline{D}$	FPT $\mathcal{O}^*(2^{6.056\cdot \overline{D}+o(\overline{D})})$	Thm. 4.2	NP-hard even if $\overline{D}=0$	[GJS76]
# Taxa n	FPT $\mathcal{O}^*(2^n)$	Prop. 4.6a	FPT $\mathcal{O}^*(n!)$	Prop. 4.6b
# Teams $\lvert T\rvert$	NP-hard even if $\lvert T\rvert=1$	Prop. 4.4a	NP-hard even if $\lvert T\rvert=1$	Prop. 4.4a
Solution size k	W[1]-hard	Prop. 4.4b	W[1]-hard	Prop. 4.4b
Max time $\max_{\text{ex}}$	*open*		*open*	
Unique times var_{ex}	NP-hard even if $\text{var}_{\text{ex}}=1$	Prop. 4.4a	NP-hard even if $\text{var}_{\text{ex}}=1$	Prop. 4.4a
Unique lengths var_ℓ	*open*		W[1]-hard	[HH24]
Person-hours $H_{\max}$	FPT $\mathcal{O}^*((\lvert T\rvert+1)^{2\max_{\text{ex}}})$	Prop. 4.15a	FPT $\mathcal{O}^*(3^{\lvert T\rvert+\max_{\text{ex}}})$	Prop. 4.15c
	FPT $\mathcal{O}^*((H_{\max_{\text{ex}}})^{2\,\text{var}_{\text{ex}}})$	Prop. 4.15b		
$\text{var}_\ell+\text{var}_{\text{ex}}$	XP $\mathcal{O}(n^{2\,\text{var}_\ell\cdot\text{var}_{\text{ex}}+1})$	Prop. 4.16	*open*	

4.2 Preliminaries

In this section, we present the formal definition of the problems, and the parameterization. We further start with some preliminary observations.

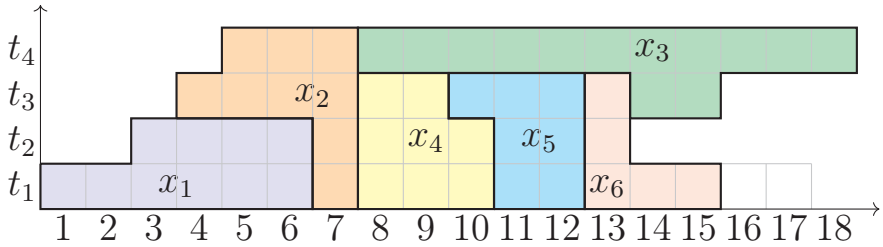
4.2.1 Definitions

Problem Definitions and Parameterizations. In the following, we are given a set of taxa X and a phylogenetic X-tree $\mathcal{T}$, which will be used to calculate the phylogenetic diversity of any subset of taxa $A \subseteq X$. In addition, we are given an integer *extinction time* $\text{ex}(x)$ for each taxon $x \in X$ representing the amount of time remaining to save that taxon before it goes extinct, and an integer *rescue length* $\ell(x)$ representing how much time needs to be spent in order to save that taxon. Thus, we need to spend $\ell(x)$ units of time before $\text{ex}(x)$ units of time have elapsed, if we wish to save x from extinction.

In addition, we are given a set of teams $T = \{t_1, \ldots, t_{|T|}\}$ that are available to work on saving taxa, where each team t_i is represented by a pair of integers (s_i, e_i) with $0 \leq s_i < e_i$. Here s_i and e_i represent the starting time and the ending time of team t_i, respectively. Thus, team t_i is available for $(e_i - s_i)$ units of time, starting after s_i time steps. For convenience, we refer to the units of time in this paper as *person-hours*, although of course in practice the units of time may be days or weeks.

We define $\mathcal{H}_T := \{(i,j) \in \mathbb{N}^2 \mid t_i \in T, s_i < j \leq e_i\}$. That is, $\mathcal{H}_T$ is the set of all pairs (i,j) where team t_i is available to work in *timeslot* j. For $j \in [\text{var}_{\text{ex}}]$, we define $H_j := |\{(i,j') \in \mathcal{H}_T : j' \leq \text{ex}_j\}|$. That is, H_j is the number of person-hours available until time ex_j.

Then, a *T-schedule* is a function $f : \mathcal{H}_T \to A \cup \{\text{NONE}\}$ for a set of taxa $A \subseteq X$.



t_i	t_1	t_2	t_3	t_4
s_i	0	2	3	4
e_i	17	13	15	18

x_i	x_1	x_2	x_3	x_4	x_5	x_6
$\ell(x_i)$	10	9	13	8	7	5
$\text{ex}(x_i)$	7	7	18	12	12	18

$i =$	1	2	3
ex_i	7	12	18
H_i	19	39	55

Figure 4.1: This is a hypothetical valid schedule saving the set of taxa $\{x_1, \ldots, x_6\}$. Each square marks a tuple $(i, j) \in \mathcal{H}_T$.

That is, a T-schedule is a mapping from the available timeslots for each team to the taxa in A (or NONE). Intuitively, f shows which teams will work to save which taxa in A, and at which times. See Figure 4.1 for an example.

We say that a T-schedule f is *valid* if $\text{ex}(x) \geq j$ for each $x \in A$, and $(i, j) \in \mathcal{H}$ with $f((i, j)) = x$. That is, f does not assign a team to work on taxon x after its extinction time. We say that f *saves* A if f is valid and $|f^{-1}(x)| \geq \ell(x)$ for all $x \in A$. That is, a schedule f saves $A \subseteq X$ if every taxon $x \in A$ has at least $\ell(x)$ person-hours assigned to x by its extinction time $\text{ex}(x)$.

The definition of a valid schedule allows for several teams to be assigned to the same taxon at the same time, so that for example the task of preserving a taxon can be done in half the time by using twice the number of teams. Whether this is realistic or not depends on the nature of the tasks involved in preservation. For instance the task of preparing a new enclosure for animals might be completed faster by several teams working in parallel, whilst the task of rearing infants to adulthood cannot be sped up in the same way. Due to these concerns, we will consider two variations of the problem, differentiating in whether a schedule must be *strict*.

A T-schedule f saving A is *strict* if $|\{i \in [|T|] : f((i, j)) = x\}| = 1$ for each $x \in A$. That is, there is only one team t_i assigned to save each taxon x. We state without a proof that we may assume $f^{-1}(x) = \{(i, j + 1), (i, j + 2), \ldots, (i, j + \ell(x)\}$ for some $j \geq s_i$. That is, once started, team t_i continuously works on x. We note that in a non-strict schedule f, it is even possible that multiple teams may work on the same taxa x at once.

Formally, the problems we regard in this chapter are as follows.

Time Sensitive Maximization of Phylogenetic Diversity (Time-PD)

Input: A phylogenetic X-tree $\mathcal{T} = (V, E, \lambda)$, integers $\mathrm{ex}(x)$ and $\ell(x)$ for each $x \in X$, a set of teams T, and a target diversity $D \in \mathbb{N}_0$.

Question: Is there a valid T-schedule saving S, for some $S \subseteq X$ such that $PD_{\mathcal{T}}(S) \geq D$?

The problem s-Time-PD is the same as Time-PD, except for the restriction that the valid T-schedule should be strict.

Strict Time Sensitive Maximization of Phylogenetic Diversity (s-Time-PD)

Input: A phylogenetic X-tree $\mathcal{T} = (V, E, \lambda)$, integers $\mathrm{ex}(x)$ and $\ell(x)$ for each $x \in X$, a set of teams T, and a target diversity $D \in \mathbb{N}_0$.

Question: Is there a strict valid T-schedule saving S, for some $S \subseteq X$ such that $PD_{\mathcal{T}}(S) \geq D$?

A set S that satisfies these conditions for an instance $\mathcal{I}$ of Time-PD or s-Time-PD is called a *solution of* $\mathcal{I}$.

These two problem definitions only differ in the fact that valid T-schedule in the latter needs to be strict while in the former it is not necessary.

Observe that if there is only one team, every valid schedule is also strict. Thus, an instance $\mathcal{I} = (\mathcal{T}, \ell, \mathrm{ex}, T, D)$ with $|T| = 1$ is a `yes`-instance of Time-PD if and only if $\mathcal{I}$ is a `yes`-instance of s-Time-PD. Lemma 4.2 and 4.3 elaborate on the conditions in these questions in more detail.

Additional Definitions. Now, we introduce some additional definitions that will be helpful for the parameterizations and proofs that follow.

Definition 4.1 ($\mathrm{var}_{\mathrm{ex}}$ and the Classes Y_i and Z_i)**.** Let $\mathrm{var}_{\mathrm{ex}} := |\{\mathrm{ex}(x) : x \in X\}|$ and $\max_{\mathrm{ex}} := \max\{\mathrm{ex}(x) : x \in X\}$. That is, $\mathrm{var}_{\mathrm{ex}}$ is the number of different extinction times for taxa in X, and $\max_{\mathrm{ex}}$ is the latest extinction time.

Let $\mathrm{ex}_1 < \mathrm{ex}_2 < \cdots < \mathrm{ex}_{\mathrm{var}_{\mathrm{ex}}} = \max_{\mathrm{ex}}$ denote the elements of $\mathrm{ex}(X)$. For each $j \in [\mathrm{var}_{\mathrm{ex}}]$ the *class* $Y_j \subseteq X$ is the set of taxa x with $\mathrm{ex}(x) = \mathrm{ex}_j$ and we define $Z_j = Y_1 \cup \cdots \cup Y_j$. Further, we define $\mathrm{ex}^*(x) = j$ for each $x \in Y_j$.

Along similar lines to $\mathrm{var}_{\mathrm{ex}}$ and $\max_{\mathrm{ex}}$, we define $\mathrm{var}_\ell := |\{\ell(x) : x \in X\}|$ and $\max_\ell := |\{\ell(x) : x \in X\}|$. That is, var_ℓ is the number of different rescue lengths

of taxa, and $\max_\ell$ is the largest rescue length. We let $\ell_1 < \ell_2 < \cdots < \ell_{\text{var}_\ell} = \max_\ell$ denote the elements of $\{\ell(x) \mid x \in X\}$.

Given an instance $\mathcal{I}$ of Time-PD or s-Time-PD with target diversity D, we define $\overline{D} := \text{PD}_\mathcal{T}(X) - D = \sum_{e \in E} \lambda(e) - D$. Thus, $\overline{D}$ is the acceptable loss of diversity—if we save a set of taxa $A \subseteq X$ with $\text{PD}_\mathcal{T}(A) \geq D$, then the amount of diversity we lose from $\mathcal{T}$ as a whole is at most $\overline{D}$.

We define $\overline{H_j}$ to be $\sum_{x \in Z_j} \ell(x) - H_j$ for each $j \in [\text{var}_{\text{ex}}]$. That is, $\overline{H_j}$ is the difference between the number of person-hours needed to save all taxa in Z_j, and the number of person-hours available to save them.

4.2.2 Observations

In the following, we present some easy observations.

Lemma 4.2. *There exists a valid T-schedule saving a set of taxa $A \subseteq X$ if and only if $\sum_{x \in A \cap Z_j} \ell(x) \leq H_j$ for each $j \in [\text{var}_{\text{ex}}]$.*

Proof. Recall that ex_j is the jth extinction time and $\text{ex}^*(x) = j$ for each $x \in Y_j$

Suppose first that f is a valid schedule saving A. Then, for each $x \in A$, the set $f^{-1}(x)$ contains at least $\ell(x)$ pairs (i, j') with $j' \leq \text{ex}(x)$. It follows that $f^{-1}(Z_j)$ contains at least $\sum_{x \in A \cap Z_j} \ell(x)$ pairs (i, j') with $j' \leq \text{ex}_j$, and so for each $j \in [\text{var}_{\text{ex}}]$, we conclude $\sum_{x \in A \cap Z_j} \ell(x) \leq |\{(i, j') \in \mathcal{H}_T \mid j' \leq \text{ex}_j\}| = H_j$.

Conversely, suppose $\sum_{x \in A \cap Z_i} \ell(x) \leq H_j$ for each $j \in [\text{var}_{\text{ex}}]$. Then order the elements of $\mathcal{H}_T$ such that (i', j') appears before (i'', j'') if $j' < j''$, and order the elements of A such that x appears before y if $\text{ex}^*(x) < \text{ex}^*(y)$ (thus, all elements of $A \cap Y_j$ appear before all elements of $A \cap Y_{j+1}$). Now define $f : \mathcal{H}_T \to A \cup \{\text{NONE}\}$ by repeatedly choosing the first available taxon x of A, and assigning it to the first available $\ell(x)$ elements of $\mathcal{H}_T$. Then, the first $\sum_{x \in A \cap Z_j} \ell(x) \leq H_j$ elements of $\mathcal{H}_T$ are used to save taxa in Z_j, and these elements of $\mathcal{H}_T$ are all of the form (i, j') for $j' \leq \text{ex}_j$. It follows that f is a valid schedule saving A. □

By Lemma 4.2, we have that $\mathcal{I}$ is a `yes`-instance of Time-PD (or s-Time-PD with $|T| = 1$) if and only if there exists a set $S \subseteq X$ with $\text{PD}_\mathcal{T}(S) \geq D$ such that $\sum_{x \in A \cap Z_i} \ell(x) \leq H_i$ for each $i \in [\text{var}_{\text{ex}}]$. The following lemma follows from the definitions of valid and strict schedules.

Lemma 4.3. *There exists a strict valid T-schedule saving $A \subseteq X$ if and only if there is a partition of A into sets $A_1 \ldots, A_{|T|}$ such that for each $i \in [|T|]$, there is a valid $\{t_i\}$-schedule saving A_i.*

For the parameterized point of view, we first show that TIME-PD and S-TIME-PD are NP-hard, even for quite restricted instances. Our proof adapts the reduction of Richard Karp used to show that the scheduling problem $1||\sum w_j(1-U_j)$ is NP-hard [Kar72]. While Karp gives a reduction from KNAPSACK, we use a reduction from k-SUBSET SUM. This result was also observed by [PG07, HS06] for BUDGETED NAP—MAX-PD with integer cost on taxa—which is a special case of TIME-PD or S-TIME-PD.

Proposition 4.4 ([Kar72, PG07, HS06])**.**

(a) TIME-PD *and* S-TIME-PD *are* NP*-hard.*

(b) It is W[1]*-hard with respect to k to decide whether an instance of* TIME-PD *or* S-TIME-PD *has a solution in which k taxa are saved.*

Both statements hold even if the tree in the input is a star and $|T| = \mathrm{var}_{\mathrm{ex}} = 1$.

Proof. We reduce from k-SUBSET SUM, which is NP-hard and W[1]-hard when parameterized by k, the size of the solution [DF95b]. In k-SUBSET SUM we are given a multiset of integers $\mathcal{Z} = \{z_1, \ldots, z_{|\mathcal{Z}|}\}$ and two integers k and G. It is asked whether there is a set $S \subseteq \mathcal{Z}$ of size k such that $\sum_{z \in S} z = G$.

Reduction. Let $(\mathcal{Z}, G)$ be an instance of k-SUBSET SUM. Let Q be an integer which is bigger than the sum of all elements in $\mathcal{Z}$.

We define an instance $\mathcal{I} = (\mathcal{T}, \ell, \mathrm{ex}, T, D)$ with a star tree $\mathcal{T}$ that has a root ρ and a set of leaves $X = \{x_1, \ldots, x_{|\mathcal{Z}|}\}$. For each $x_i \in X$, we set edge-weights $\lambda(\rho x_i)$ and the rescue length $\ell(x_i)$ to be $z_i + Q$. Further, define an extinction time of $\mathrm{ex}(x_i) := G + kQ$, which is the same for each taxon and the only team operates from $s_1 := 0$ to $e_1 := G + kQ$. Finally, we set $D := G + kQ$.

Correctness. The reduction is computed in polynomial time for a suitable Q. Because there is only one team, $\mathcal{I}$ is a yes-instance of TIME-PD if and only if $\mathcal{I}$ is a yes-instance of S-TIME-PD. It remains to show that any solution for $\mathcal{I}$ saves exactly k taxa, and that $\mathcal{I}$ is a yes-instance of TIME-PD if and only if $(\mathcal{Z}, G)$ is a yes-instance of k-SUBSET SUM.

So let $(\mathcal{Z}, G)$ be a yes-instance of k-SUBSET SUM with solution S. Define a set of taxa $S' := \{x_i \in X \mid z_i \in S\}$. Clearly, S' and S are of size k. Then,

$$\mathrm{PD}_{\mathcal{T}}(S') = \sum_{x_i \in S'} \lambda(\rho x_i) = \sum_{z_i \in S} (z_i + Q) = kQ + \sum_{z_i \in S} z_i = G + kQ = D,$$

and analogously $\sum_{x_i \in S'} \ell(x_i) = \sum_{z_i \in S} (z_i + Q) = G + kQ = H_1$. Consequently, S' is a solution for $\mathcal{I}$.

Let $\mathcal{I}$ be a yes-instance with solution S'. Because

$$G + kQ = D \leq \mathrm{PD}_{\mathcal{T}}(S') = \sum_{x_i \in S'} \lambda(\rho x_i) = \sum_{z_i \in S} (z_i + Q) = \sum_{x_i \in S'} \ell(x_i) \leq G + kQ,$$

we conclude that $G + kQ = \sum_{z_i \in S}(z_i + Q)$, from which it follows that $|S'| = k$ and $\sum_{z_i \in S} z_i = G$. Therefore, $S := \{z_i \mid x_i \in S'\}$ is a solution of $(\mathcal{Z}, G)$. □

The scheduling variant $\mathrm{P3}||\mathrm{C}_{\max}$ in which it is asked for whether the given jobs can be scheduled on three parallel machines such that the *makespan* (the maximum time taken on any machine) is at most $\mathrm{C}_{\max}$, is strongly NP-hard [GJS76, GJ78]. $\mathrm{P3}||\mathrm{C}_{\max}$ can be seen as a special case of $\mathrm{P3}||\sum w_j(1-U_j)$ and therefore s-Time-PD in which there are three teams, the tree is a star on which all weights are 1, and we have to save every taxon in the time defined by the makespan.

Proposition 4.5 ([GJS76])**.** s-Time-PD *is strongly* NP*-hard even if the tree is a star, every taxon has to be saved (*$\overline{D} = 0$*), there are three teams,* $\max_\lambda = 1$ *and* $\mathrm{var}_{\mathrm{ex}} = 1$*.*

Our final observation in this section is that for an instance of Time-PD and a set of taxa $S \subseteq X$, we can compute the diversity $\mathrm{PD}_{\mathcal{T}}(S)$ and check whether there is a valid T-schedule saving S in polynomial time (by Lemma 4.2). Therefore, checking each subset of X yields an $\mathcal{O}^*(2^n)$-time-algorithm for Time-PD.

For an instance of s-Time-PD, in $\mathcal{O}^*(n!)$ time we can also guess the order of the taxa and then assign taxa to the first team until they have no more capacity. Afterward, assign the next taxa to the second team and so on. The instance is a yes-instance if and only if for some order this assignment yields a strict valid T-schedule of a set of taxa with diversity at least D. We conclude the following.

Proposition 4.6.

(a) Time-PD *can be solved in* $2^n \cdot \mathrm{poly}(|\mathcal{I}|)$ *time.*

(b) s-Time-PD *can be solved in* $n! \cdot \mathrm{poly}(|\mathcal{I}|)$ *time.*

4.3 The Diversity D

In this section, we show that Time-PD and s-Time-PD are FPT when parameterized by the threshold of diversity D. When one tries to use the standard approach with a dynamic programming algorithm over the vertices of the tree, one will struggle with

the question of how to handle the different extinction times of the taxa. While it is straightforward how to handle taxa of a specific extinction time, already with a second, it is not trivial how to combine sub-solutions. In order to overcome these issues, we use the technique of color coding in addition to dynamic programming.

In the following, we consider colored versions of the problems, c-Time-PD and c-s-Time-PD. In these, additionally to the input of the respective uncolored variant, we are given a coloring c which assigns each edge $e \in E$ *a set of colors* $c(e)$: A subset of $[D]$ of size $\lambda(e)$. For a taxon $x \in X$, we define $c(x)$ to be the union of colors of the edges on a path from the root to x. Further, for a set S of taxa, we define $c(S)$ to be $\bigcup_{x \in S} c(x)$. In c-Time-PD (respectively, c-s-Time-PD), we ask whether there is a subset S of taxa and a (strictly) valid T-schedule saving S such that $c(S) = [D]$.

Next, we show that c-Time-PD and c-s-Time-PD are FPT with respect to D. The difficulty of combining sub-solutions for different extinction times also arises in these colored versions of the problem. However, the color enables us to consider the tree as a whole. We select taxa with ordered extinction time such that we are able to check whether there is a (strictly) valid T-schedule which can indeed save the selected set of taxa.

Lemma 4.7. c-Time-PD *can be solved in* $\mathcal{O}(2^D \cdot D \cdot \mathrm{var}_{\mathrm{ex}}^2 \cdot n)$ *time.*

Proof. Let $\mathcal{I} = (\mathcal{T}, \ell, \mathrm{ex}, T, D, c)$ be an instance of c-Time-PD. For $p \in [\mathrm{var}_{\mathrm{ex}}]$, we call a set S of taxa *p-compatible* if $\ell(S \cap Z_q) \leq H_i$ for each $q \in [p]$. For a set of colors $C \subseteq [D]$ and $p \in [\mathrm{var}_{\mathrm{ex}}]$, we call a set S of taxa *(C,p)-feasible* if

- C is a subset of $c(S)$,
- S is a subset of Z_p, and
- S is p-compatible.

Intuitively, the (C,p)-feasible sets S consist of taxa which overall cover the colors C if saved and for which a valid T-schedule saving S exists.

Table definition. In the following dynamic programming algorithm with table DP, for each $C \subseteq [D]$ and $p \in [\mathrm{var}_{\mathrm{ex}}]$ we want entry $\mathrm{DP}[C,p]$ to store the minimum length $\ell(S)$ of a (C,p)-feasible set of taxa $S \subseteq X$, with $\mathrm{DP}[C,p] = \infty$ if no (C,p)-feasible set exists.

Algorithm. As a base case, we store $\mathrm{DP}[\emptyset, p] = 0$ for each $p \in [\mathrm{var}_{\mathrm{ex}}]$.

To compute further values, we use the recurrence

$$\mathrm{DP}[C,p] = \min_{x \in Z_p, c(x) \cap C \neq \emptyset} \psi(\mathrm{DP}[C \setminus c(x), \mathrm{ex}^*(x)] + \ell(x), H_{\mathrm{ex}^*(x)}). \tag{4.1}$$

Recall $\text{ex}^*(x) = q$ for $x \in Y_q$; and $\psi(a, b) = a$ if $a \leq b$ and otherwise $\psi(a, b) = \infty$.

We return `yes` if $\text{DP}[[D], \text{var}_{\text{ex}}] \leq \max_{\text{ex}}$, and otherwise we return `no`.

Correctness. Let $\mathcal{I}$ be an instance of c-Time-PD. From the definition of (C, p)-feasible sets it follows that there exists a $([D], \text{var}_{\text{ex}})$-feasible set of taxa S if and only if $\mathcal{I}$ is a `yes`-instance of c-Time-PD. It remains to show that $\text{DP}[C, p]$ stores the smallest length of a (C, p)-feasible set S.

As the set $S = \emptyset$ is $(\emptyset, p)$-feasible for each $p \in [\text{var}_{\text{ex}}]$, the base case is correct.

As an induction hypothesis, assume that for a fixed and non-empty set of colors $C \subseteq [D]$ and a fixed integer $p \in [\text{var}_{\text{ex}}]$, entry $\text{DP}[K, q]$ stores the correct value for each $K \subsetneq C$ and $q \leq p$. We prove that $\text{DP}[C, p]$ stores the correct value by showing that if a (C, p)-feasible set of taxa S exists then $\text{DP}[C, p] \leq \ell(S)$ and that if $\text{DP}[C, p] = a < \infty$ then there is a (C, p)-feasible set S with $\ell(S) = a$.

Let $S \subseteq X$ be a (C, p)-feasible set of taxa. Observe that if $c(x) \cap C = \emptyset$ for a taxon $x \in S$ then also $S \setminus \{x\}$ is (C, p)-feasible. So, we assume that $c(x) \cap C \neq \emptyset$ for each $x \in S$. Let $y \in S$ be a taxon such that $\text{ex}(y) \geq \text{ex}(x)$ for each $x \in S$. Observe that the set $S \setminus \{y\}$ is $(C \setminus c(y), \text{ex}^*(y))$-feasible. Thus, because $y \in S \subseteq Z_p$ and $c(y) \cap C \neq \emptyset$, $\text{DP}[C \setminus c(y), \text{ex}^*(y)] \leq \ell(S \setminus \{y\}) = \ell(S) - \ell(y)$ by the induction hypothesis. We conclude $\text{DP}[C \setminus c(y), \text{ex}^*(y)] + \ell(y) \leq \ell(S) \leq H_{\text{ex}^*(y)}$ because S is $\text{ex}^*(y)$-compatible. Hence, $\text{DP}[C, p] \leq \text{DP}[C \setminus c(y), \text{ex}^*(y)] + \ell(y) \leq \ell(S)$.

Conversely, suppose $\text{DP}[C, p] < \infty$. Then, by Recurrence (4.1), a taxon $x \in Z_p$ exists such that $c(x) \cap C \neq \emptyset$ and $\text{DP}[C, p] = \text{DP}[C \setminus c(x), \text{ex}^*(x)] + \ell(x)$. By the induction hypothesis, there is a $(C \setminus c(x), \text{ex}^*(x))$-feasible set S' such that $\text{DP}[C \setminus c(x), \text{ex}^*(x)] = \ell(S')$. Because $c(x) \cap (C \setminus c(x))$ is empty, x is not in S'. Further, $\text{DP}[C, p] = \text{DP}[C \setminus c(x), \text{ex}^*(x)] + \ell(x) = \ell(S') + \ell(x) = \ell(S' \cup \{x\})$ and $S' \cup \{x\}$ is the desired (C, p)-feasible set.

Running time. The table has $\mathcal{O}(2^D \cdot \text{var}_{\text{ex}})$ entries. For each $x \in X$ and every set of colors C, we can compute whether $c(x)$ and C have a non-empty intersection in $\mathcal{O}(D)$ time. Then, we can compute the set $C \setminus c(x)$ in $\mathcal{O}(D)$ time. Each entry stores an integer that is at most $H_{\max_{\text{ex}}}$ (or ∞). In our RAM model, the addition and the comparison in ψ can be done in constant time, and so the right side of Recurrence (4.1) can be computed in $\mathcal{O}(D \cdot \text{var}_{\text{ex}} \cdot n)$ time. Altogether, we can compute a solution in $\mathcal{O}(2^D \cdot D \cdot \text{var}_{\text{ex}}^2 \cdot n)$ time. □

For the algorithm in Lemma 4.7 it is crucial that the (C, j)-feasibility of a set S depends only on $\ell(S)$ and the values $H_1, \ldots, H_j$. This is not the case for c-s-Time-PD, as the available times of the teams impact a possible schedule. Our solution for this issue is that we divide the colors (representing the diversity) that are supposed to be

saved and delegate the colors to specific teams. The problem then can be solved individually for the teams. This division and delegation happens in the Recurrence (4.3).

Lemma 4.8. c-s-Time-PD *can be solved in* $\mathcal{O}^*(2^D \cdot D^3)$ *time.*

Proof. Table definition. We define a dynamic programming algorithm with three tables DP_0, DP_1 and DP_2. Entries of table DP_0 are computed with the idea of Lemma 4.7. DP_1 is an auxiliary table. A final solution can be found in table DP_2, then. For this lemma, similar to H_i, we define $H_i^{(q)} := |\{(i,j) \in \mathcal{H}_T \mid j \leq \mathrm{ex}_q\}|$. That is, $H_i^{(q)}$ are the person-hours of team t_i until ex_q.

Let $\mathcal{I}$ be an instance of c-s-Time-PD. We define terms similar to Lemma 4.7. For a team $t_i \in T$ (and an integer $p \in [\mathrm{var_{ex}}]$), a set $S \subseteq X$ is t_i-*compatible* (respectively, (p, t_i)-*compatible*) if $\ell(S \cap Z_q) \leq H_i^{(q)}$ for each $q \in [\mathrm{var_{ex}}]$ ($q \in [p]$). Inspired by Lemma 4.3, for a $i \in [|T|]$ we call a set S of taxa $[i]$-*compatible* if there is a partition $S_1, \ldots, S_i$ of S such that S_j is t_j-compatible for each $j \in [i]$. For a set of colors $C \subseteq [D]$, an integer $p \in [\mathrm{var_{ex}}]$ and a team $t_i \in T$, we call a set S of taxa (C, p, t_i)-*feasible* if

- S is a subset of Z_p,
- S is (p, t_i)-compatible,
- C is a subset of $c(S)$, and
- $(S = \emptyset$ or $S \cap Y_p \neq \emptyset)$.

Formally, for a set of colors $C \subseteq [D]$, an integer $p \in [\mathrm{var_{ex}}]$ and a team $t_i \in T$ we want that

- in $\mathrm{DP}_0[C, p, i]$ the minimum length $\ell(S)$ of a (C, p, t_i)-feasible set of taxa $S \subseteq X$ is stored,
- in $\mathrm{DP}_1[C, i]$ stores 1 if there is a t_i-compatible set $S \subseteq X$ with $c(S) \supseteq C$ and 0 otherwise, and
- in $\mathrm{DP}_2[C, i]$ stores 1 if there is a $[i]$-compatible set $S \subseteq X$ with $c(S) \supseteq C$ and 0 otherwise.

Algorithm. As a base case, we store $\mathrm{DP}_0[\emptyset, p, i] = 0$ for each $p \in [\mathrm{var_{ex}}]$ and $i \in [|T|]$.

To compute further values of DP_0, we use the recurrence

$$\mathrm{DP}_0[C, p, i] = \min_{x \in Y_p, c(x) \cap C \neq \emptyset, q \leq p} \psi(\mathrm{DP}[C \setminus c(x), q, i] + \ell(x), H_i^{(q)}). \tag{4.2}$$

Recall that $\psi(a,b) = a$ if $a \leq b$ and otherwise $\psi(a,b) = \infty$. Once all values in DP_0 are computed, we compute $z_{C,i} := \min_{q \in [\mathrm{var_{ex}}]} \mathrm{DP}[C,q,i]$ and set $\mathrm{DP}_1[C,i] = 1$ if $z_{C,i} \leq H_i^{(\mathrm{max_{ex}})}$ and $\mathrm{DP}_1[C,i] = 0$, otherwise.

Finally, we define $\mathrm{DP}_2[C,1] := \mathrm{DP}_1[C,1]$ and compute further values with

$$\mathrm{DP}_2[C,i+1] = \min_{C' \subseteq C} \mathrm{DP}_2[C',i] \cdot \mathrm{DP}_1[C \setminus C', i+1]. \tag{4.3}$$

We return `yes` if $\mathrm{DP}_2[[D],|T|] = 1$, and otherwise, we return `no`.

Correctness. Analogously as in Lemma 4.7, one can prove that the entries of DP_0 store the correct value. It directly follows by the definition that the entries of DP_1 store the correct value. Likewise $\mathrm{DP}_2[C,1]$ stores the correct value for colors $C \subseteq [D]$ be the definition.

Fix a set of colors $C \subseteq [D]$ and an integer $i \in [|T|-1]$. We assume as an induction hypothesis that entry $\mathrm{DP}_2[K,j]$ stores the correct value for each $K \subsetneq C$ and $j \leq i$. To conclude that $\mathrm{DP}_2[C,j+1]$ stores the correct value, we show that $\mathrm{DP}_2[C,i+1] \leq \ell(S)$ for each $[i+1]$-compatible set $S \subseteq X$ with $c(S) \supseteq C$ and if $\mathrm{DP}_2[C,i+1] = a < \infty$ then there is an $[i+1]$-compatible set $S \subseteq X$ with $c(S) \supseteq C$ and $\ell(S) = a$.

Let $S \subseteq X$ be an $[i+1]$-compatible set with $c(S) \supseteq C$. Let $S_1, \ldots, S_{i+1}$ be a partition of S such that S_j is t_j-compatible for each $j \in [i+1]$. Define $\hat{S}$ to be the union of the S_j and $\hat{C} := C \cap c(\hat{S})$. Then, $\hat{S}$ is $[i]$-compatible with $c(\hat{S}) \supseteq \hat{C}$. Therefore, entry $\mathrm{DP}_2[\hat{C},i]$ stores at least $\ell(\hat{S})$ by the induction hypothesis. Because $c(S_{i+1}) \supseteq C \setminus \hat{C}$ we conclude that $\mathrm{DP}_1[C \setminus \hat{C}, i+1] \leq \ell(S_{i+1})$. We conclude that $\mathrm{DP}_2[C,i+1] \leq \mathrm{DP}_2[\hat{C},i] + \mathrm{DP}_1[C \setminus \hat{C}, i+1] \leq \ell(\hat{S}) + \ell(S_{i+1}) = \ell(S)$.

Let $\mathrm{DP}_2[C,i+1]$ store the value $a < \infty$. By Recurrence (4.3), there is a set of colors $C' \subseteq C$ such that $a = \mathrm{DP}_2[C',i] + \mathrm{DP}_1[C \setminus C', i+1]$. By the induction hypothesis, we conclude then that there is an $[i]$-compatible set S_1 and a t_{i+1}-compatible set S_2 such that $\mathrm{DP}_2[C',i] = \ell(S_1)$ and $\mathrm{DP}_1[C \setminus C', i+1] = \ell(S_2)$. Assume first that S_1 and S_2 are disjoint. We define $S := S_1 \cup S_2$ and conclude with the disjointness that $\ell(S) = \ell(S_1) \cup \ell(S_2) = a$. Further, we conclude that S is $[i+1]$-compatible and $c(S) = c(S_1) \cup c(S) = C' \cup (C \setminus C') = C$ such that S is the desired set. It remains to argue that non-disjoint S_1 and S_2 contradict the optimality of C'.

Now, let A be the intersection of S_1 and S_2 and define $\hat{C} := c(S_1)$ and $\hat{S} := S_2 \setminus A$. Observe that $\hat{S}$ is a t_{i+1}-compatible set with $c(\hat{S}) \supseteq (C \setminus C') \setminus c(A) = C \setminus \hat{C}$. Therefore,

$$\begin{aligned} \mathrm{DP}_2[\hat{C},i] + \mathrm{DP}_1[C \setminus \hat{C}, i+1] &\leq \ell(S_1) + \ell(\hat{S}) \\ &= \ell(S_1) + \ell(S_2) - \ell(A) \\ &= \mathrm{DP}_2[C',i] + \mathrm{DP}_1[C \setminus C', i+1] - \ell(A), \end{aligned}$$

which contradicts the optimallity of C', unless A is empty.

Running time. By Lemma 4.7, in $\mathcal{O}(2^D \cdot D \cdot \mathrm{var}_{\mathrm{ex}}^2 \cdot n)$ time all entries of DP_0 are computed. Table DP_1 contains $2^D \cdot |T|$ entries which in $\mathcal{O}(\mathrm{var}_{\mathrm{ex}})$ time can be computed, each.

To compute the values of table DP_2 in Recurrence (4.3), we use convolutions. Readers unfamiliar with this topic, we refer to [CFK+15, Sec. 10.3]. We define functions $f_i, g_i : 2^{[D]} \to \{0,1\}$ where $f_i(C) := \mathrm{DP}_2[C,i]$ and $g_i(C) := \mathrm{DP}_1[C,i]$. Then, we can express Recurrence (4.3) as $\mathrm{DP}_2[C,i+1] = f_{i+1}(C) = (f_i * g_{i+1})(C)$. Therefore, for each $i \in [|T|]$ we can compute all values of $\mathrm{DP}_2[\cdot,i] = f_i(\cdot)$ in $\mathcal{O}(2^D \cdot D^3)$ time [CFK+15, Thm. 10.15].

Altogether, in $\mathcal{O}(2^D \cdot D^3 \cdot |T| \cdot \mathrm{var}_{\mathrm{ex}}^2 \cdot n)$ time, we can compute a solution of an instance of c-s-Time-PD. □

Our following procedure in a nutshell is to use standard color coding techniques to show that in $\mathcal{O}^*(2^{\mathcal{O}(D)})$ time one can reduce an instance $\mathcal{I}$ of Time-PD (respectively, s-Time-PD) to $z \in \mathcal{O}^*(2^{\mathcal{O}(D)})$ instances $\mathcal{I}_1, \ldots, \mathcal{I}_z$ of c-Time-PD (c-s-Time-PD), which can be solved using Lemma 4.7 and Lemma 4.8. In particular, if S is a solution for $\mathcal{I}$ then $c(S) = D$ under the coloring c in at least one $\mathcal{I}_i$. Then, $\mathcal{I}$ is a `yes`-instance, if and only if any $\mathcal{I}_i$ is a `yes`-instance.

Theorem 4.1. Time-PD *and* s-Time-PD *can be solved in* $\mathcal{O}^*(2^{2.443 \cdot D + o(D)})$ *time.*

For an overview of color coding, we refer the reader to [CFK+15, Sec. 5.2 and 5.6].

The key idea is that we construct a family $\mathcal{C}$ of colorings on the edges of $\mathcal{T}$, where each edge $e \in E$ is assigned a subset of $[D]$ of size $\lambda(e)$. Using these, we generate $|\mathcal{C}|$ instances of the colored version of the problem, which we then solve in $\mathcal{O}^*(2^D \cdot D)$ time. Central for this will be the concept of a perfect hash family as defined in Definition 2.14.

Proof. We focus on Time-PD and omit the analogous proof for s-Time-PD.

Reduction. Let $\mathcal{I} = (\mathcal{T}, \mathrm{ex}, \ell, T, D)$ be an instance of Time-PD. We assume that for each taxon $x \in X$ some valid T-schedule saving $\{x\}$ exists, as we can delete x from $\mathcal{T}$, otherwise. Therefore, if there is an edge e with $\lambda(e) \geq D$, then $\{x\}$ is a solution for each $x \in \mathrm{off}(e)$ and we have a trivial `yes`-instance. Hence, we assume that $\max_\lambda < D$.

Now, order the edges $e_1, \ldots, e_m$ of $\mathcal{T}$ arbitrarily and define integers $W_0 := 0$ and $W_j := \sum_{i=1}^{j} \lambda(e_i)$ for each $j \in [m]$. We set $W := W_m$. Let $\mathcal{F}$ be a (W,D)-perfect hash family. Now, we define a family of colorings $\mathcal{C}$ as follows. For every $f \in \mathcal{F}$, let c_f be the coloring such that $c_f(e_j) := \{f(W_{j-1}+1), \ldots, f(W_j)\}$ for each $e_j \in E(\mathcal{T})$.

For each $c_f \in \mathcal{C}$, let $\mathcal{I}_{c_f} = (\mathcal{T}, \mathrm{ex}, \ell, T, D, c_f)$ be the corresponding instance of c-Time-PD. Now, solve every instance $\mathcal{I}_{c_f}$ using the algorithm described in Lemma 4.8, and return `yes` if and only if $\mathcal{I}_{c_f}$ is a `yes`-instance for some $c_f \in \mathcal{C}$.

Correctness. For any subset of edges E' with $\lambda(E') \geq D$, there is a corresponding subset of $[W]$ of size at least D. Since $\mathcal{F}$ is a (W, D)-perfect hash family, it follows that $c_f(E') = [D]$, for some $c_f \in \mathcal{C}$. Thus, in particular, if $S \subseteq X$ is a solution for instance $\mathcal{I}$, then $c_f(S) = [D]$, for some $c_f \in \mathcal{C}$, where $c_f(S)$ is the set of colors assigned by c_f to the edges between the root and a taxon of S. It follows that one of the constructed instances of c-Time-PD is a `yes`-instance.

Conversely, it is easy to see that any solution S for one of the constructed instances of c-Time-PD is also a solution for $\mathcal{I}$.

Running Time. The construction of $\mathcal{C}$ takes $e^D D^{\mathcal{O}(\log D)} \cdot W \log W$ time, and for each coloring $c \in \mathcal{C}$, the construction of instance $\mathcal{I}_c$ of c-Time-PD takes time polynomial in $|\mathcal{I}|$. Solving instances of c-Time-PD takes $\mathcal{O}^*(2^D \cdot D)$ time, and the number of instances is $|\mathcal{C}| = e^D D^{\mathcal{O}(\log D)} \cdot \log W$.

Thus, the total running time is $\mathcal{O}^*(e^D D^{\mathcal{O}(\log D)} \log W \cdot (W + (2^D \cdot D)))$. This simplifies to $\mathcal{O}^*((2e)^D \cdot 2^{\mathcal{O}(\log^2(D))})$, because $W = \mathrm{PD}_{\mathcal{T}}(X) < 2n \cdot D$. □

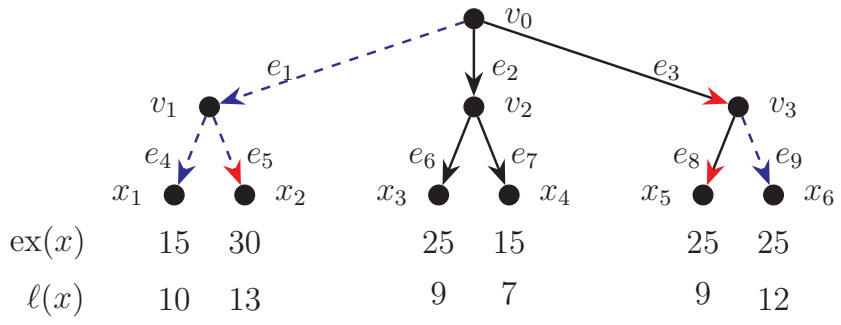
4.4 The Acceptable Loss of Diversity $\overline{D}$

In this section, we show that Time-PD is `FPT` with respect to the acceptable loss of phylogenetic diversity $\overline{D}$. Recall that $\overline{D}$ is defined as $\mathrm{PD}_{\mathcal{T}}(X) - D$. For a set of taxa $A \subseteq X$, define $E_d(A) := \{e \in E \mid \mathrm{off}(e) \subseteq A\} = E \setminus E_{\mathcal{T}}(X \setminus A)$. That is, $E_d(A)$ is the set of edges that *do not* have offspring in $X \setminus A$. Then, one may think of Time-PD as the problem of finding a set of taxa S such that there is a valid T-schedule saving S and $\lambda_\Sigma(E_d(X \setminus S))$ is at most $\overline{D}$.

In order to show the desired result, we again use the techniques of color coding and dynamic programming. For an easier understanding of the following definitions consider Figure 4.2.

We start by defining a colored version of the problem. With $E_{\leq \overline{D}}$ (or $E_{> \overline{D}}$) we denote the set of edges $e \in E$ with $\lambda(e) \leq \overline{D}$ (or $\lambda(e) > \overline{D}$). An *extinction-$\overline{D}$-colored* X-tree is a phylogenetic X-tree $\mathcal{T} = (V, E, \lambda, \hat{c}, c^-)$, where $\hat{c}$ assigns each edge $e \in E$ a *key-color* $\hat{c}(e) \in [2\overline{D}]$ and c^- assigns edges $e \in E_{\leq \overline{D}}$ a set of colors $c^-(e) \subseteq [2\overline{D}] \setminus \{\hat{c}(e)\}$ of size $\lambda(e) - 1$. With $c(e)$ we denote the union of the two sets $c^-(e) \cup \{\hat{c}(e)\}$ for each edge $e \in E_{\leq \overline{D}}$.

Observe that while in Section 4.3 we wanted the set of edges with an offspring in S to use every color at least once, here we want that the edges in $E_d(S)$ use *at*



e	$\hat{c}(e)$	$c^-(e)$
e_1	2	$\{1,7\}$
e_2	9	$\{2,3\}$
e_3	12	$\in E_b$
e_4	10	$\{9\}$
e_5	6	$\{4,11\}$
e_6	3	$\{7\}$
e_7	8	$\in E_b$
e_8	4	$\emptyset$
e_9	5	$\{3,8\}$

Figure 4.2: A hypothetical extinction-6-colored X-tree is shown. The anchored taxa set $\mathcal{A} := \{(x_1, v_1, e_5), (x_2, v_0, e_3), (x_6, v_3, e_8)\}$ with $c(E^+(\mathcal{A})) = c(\{e_1, e_4, e_5, e_9\}) = [11]$ and $\hat{c}(E_s(\mathcal{A})) = \hat{c}(\{e_3, e_5, e_8\}) = \{4, 6, 12\}$ is color-respectful. The edges in $E^+(\mathcal{A})$ are blue and dashed. The edges in $E_s(\mathcal{A})$ have a red arrow. In contrast to $\mathcal{A}$, the anchored taxa set $\mathcal{A}' := \{(x_1, v_0, e_3), (x_2, v_1, e_4), (x_6, v_3, e_8)\}$ does not have a valid ordering.

most a certain number of colors and therefore each color at most once.

The key idea behind our approach is as follows. We seek a solution S in which not only the edges of $E_d(X \setminus S)$ use each color at most once, but also every highest edge in $E_d(X \setminus S)$ has a sibling edge with a key-color not in $c(E_d(X \setminus S))$. Suppose such a set S exists, and let $x_1, \ldots, x_{|X \setminus S|}$ be some ordering of $X \setminus S$. Observe now that for any $i \in [|X \setminus S|]$, the set of edges in $E_d(\{x_1, \ldots, x_i\}) \setminus E_d(\{x_1, \ldots, x_{i-1}\})$ form a path in $\mathcal{T}$ from some vertex v_i to x_i. Furthermore, as the incoming edge of v_i is not in $E_d(\{x_1, \ldots, x_i\})$, there is an outgoing edge e_i of v_i not in $E_d(\{x_1, \ldots, x_i\})$. We may assume $\hat{c}(e_i) \notin E_d(\{x_1, \ldots, x_i\})$, either because $e_i \in E_d(X \setminus S)$ (and $E_d(X \setminus S)$ uses each color at most once), or because $e_i \notin E_d(X \setminus S)$ and so e_i is a sibling edge of a higest edge in $E_d(X \setminus S)$.

Taking the tuple (x_i, v_i, e_i) for each $i \in [|X \setminus S|]$ gives us an *anchored taxa set* that gives us all the information we need about $E_d(X \setminus S)$. Formally speaking, an *anchored taxa set* $\mathcal{A}$ is a set of tuples (x, v, e), where $x \in X$ is a taxon, v is a strict ancestor of x and e is an outgoing edge of v with $x \notin \mathrm{off}(e)$. We plan not to select taxa individually but to define an anchored set of taxa. If a tuple (x, v, e) is selected, then we want to let taxon x go extinct and we also want to arrange that the diversity of the edges on the path from v and x gets lost, but the edge e outgoing of v has an

offspring which gets saved, or will be selected to go extinct later. In the rest of the section, whenever we refer to a tuple (x, v, e), we always assume $x \in \text{off}(v)$ and e is an outgoing edge of v with $x \notin \text{off}(e)$. We denote with $X(\mathcal{A}) := \{x \mid (x, v, e) \in \mathcal{A}\}$ the taxa of an anchored taxa set. For a phylogenetic X-tree $\mathcal{T}$, a taxon $x \in X$, and a vertex $v \in \text{anc}(x)$, we denote $P_{v,x}$ to be the set of edges on the path from v and x.

For an anchored taxa set $\mathcal{A}$, we define two edge sets $E^+(\mathcal{A}) := \bigcup_{(x,v,e)\in\mathcal{A}} P_{v,x}$ and $E_s(\mathcal{A}) := \{e \mid (x, v, e) \in \mathcal{A}\}$. Informally, $E^+(\mathcal{A})$ is the set of edges that connect the taxa of the anchored taxa set with the anchors, and the edges in $E_s(\mathcal{A})$ are sibling-edges of the topmost edges of $P_{v,x}$ for each $(x, v, e) \in \mathcal{A}$. These sibling-edges e may or may not be in $E_d(X(\mathcal{A}))$, depending on whether e is part of $P_{u,y}$ for some other tuple $(y, u, e') \in \mathcal{A}$.

A set of edges E' has *unique colors* (or *unique key-colors*), if the sets $c(e_1)$ and $c(e_2)$ (respectively, $\{\hat{c}(e_1)\}$ and $\{\hat{c}(e_2)\}$) are disjoint for $e_1, e_2 \in E'$, with $e_1 \neq e_2$. An anchored taxa set $\mathcal{A}$ has a *valid ordering* $(x_1, v_1, e_1), \ldots, (x_{|\mathcal{A}|}, v_{|\mathcal{A}|}, e_{|\mathcal{A}|})$ if $(x_i, v_i, e_i) \in \mathcal{A}$, $\text{ex}(x_i) \leq \text{ex}(x_j)$ and $\hat{c}(e_j) \notin c(E^+(\mathcal{A}_j))$ for each pair $i, j \in [|\mathcal{A}|]$, with $i \leq j$. With a valid ordering we want to enforce that these tuples are selected in an order such that when letting taxa $x_1, \ldots, x_{|\mathcal{A}|}$ go extinct then at most $\lambda(E^+(\mathcal{A}))$ diversity is lost. We define $\mathcal{A}_j := \{t_i \in \mathcal{A} \mid i \leq j\}$. An anchored taxa set $\mathcal{A}$ is *color-respectful* if

CR a) $E^+(\mathcal{A})$ has unique colors,
CR b) $E_s(\mathcal{A})$ has unique key-colors,
CR c) $E^+(\mathcal{A})$ and $E_{>\overline{D}}$ are disjoint,
CR d) $\mathcal{A}$ has a valid ordering, and
CR e) $P_{v,x}$ and $P_{u,y}$ are disjoint for any tuples $(x, v, e), (y, u, e') \in \mathcal{A}$.

The existence of a color-respectful anchored taxa set will be used to show that an instance of Time-PD is a `yes`-instance. To formally show this, we first define a colored version of the problem.

extinction $\overline{D}$-colored Time Sensitive Maximization of Phylogenetic Diversity (ex-$\overline{D}$-c-Time-PD)

Input: An extinction-$\overline{D}$-colored X-tree $\mathcal{T} = (V, E, \lambda, \hat{c}, c^-)$, integers $\ell(x)$ and $\text{ex}(x)$ for every taxon $x \in X$, and a set of teams T.

Question: Is there an anchored taxa set $\mathcal{A}$ such that $\mathcal{A}$ is color-respectful, $|c(E^+(\mathcal{A}))| \leq \overline{D}$ and there is a valid T-schedule saving $X \setminus X(\mathcal{A})$?

The following lemma shows how ex-$\overline{D}$-c-Time-PD is relevant to Time-PD.

Lemma 4.9. *Let $\mathcal{I}' = (\mathcal{T}', \mathrm{ex}, \ell, T, D)$ with $\mathcal{T}' = (V, E, \lambda, \hat{c}, c^-)$, be an instance of* EX-$\overline{D}$-C-TIME-PD *and let $\mathcal{I} = (\mathcal{T}, \mathrm{ex}, \ell, T, D)$ with $\mathcal{T} = (V, E, \lambda)$ be the instance of* TIME-PD *induced by $\mathcal{I}'$ when omitting the coloring. If $\mathcal{I}'$ is a* `yes`*-instance of* EX-$\overline{D}$-C-TIME-PD*, then $\mathcal{I}$ is a* `yes`*-instance of* TIME-PD*.*

Proof. Let $\mathcal{A}$ be a solution for $\mathcal{I}'$. As there exists a T-schedule saving the set of taxa $S := X \setminus X(\mathcal{A})$, it is sufficient to show that $\mathrm{PD}_{\mathcal{T}}(S) \geq D$.

Since every edge in $\mathcal{T}$ either has an offspring in S or is in $E_d(X \setminus S)$, it follows that $\lambda(E_d(X \setminus S)) = \mathrm{PD}_{\mathcal{T}}(X) - \mathrm{PD}_{\mathcal{T}}(S)$. Thus, in order to show $\mathrm{PD}_{\mathcal{T}}(S) \geq D$ it remains to show that $\lambda(E_d(X \setminus S)) \leq \mathrm{PD}_{\mathcal{T}}(X) - D = \overline{D}$.

We prove that $E_d(X(\mathcal{A}_j)) \subseteq E^+(\mathcal{A}_j)$ for each $j \in [|\mathcal{A}|]$ by an induction on j.

For the base case $j = 1$, we have that $X(\mathcal{A}_1) = \{x_1\}$ and $E_d(\{x_1\})$ consists of the single incoming edge of x. As this edge is in P_{v_1,x_1}, the claim is satisfied.

For the induction step, assume $j > 1$ and $E_d(X(\mathcal{A}_{j-1})) \subseteq E^+(\mathcal{A}_{j-1})$. It suffices to show $E_d(X(\mathcal{A}_j)) \setminus E_d(X(\mathcal{A}_{j-1})) \subseteq P_{v_j,x_j}$, as $E^+(\mathcal{A}_j) = E^+(\mathcal{A}_{j-1}) \cup P_{v_j,x_j}$. To see this, consider the edge e_j. Because $\mathcal{A}$ has a valid ordering, $\hat{c}(e_j) \notin c(E^+(\mathcal{A}_j))$, which implies that e_j is not in $E^+(\mathcal{A}_j)$. By the inductive hypothesis, we conclude $E_d(X(\mathcal{A}_{j-1})) \subseteq E^+(\mathcal{A}_{j-1}) \subseteq E^+(\mathcal{A}_j)$ and so $e_j \notin E_d(X(\mathcal{A}_{j-1}))$. Thus, e_j has an offspring z which is not in $X(\mathcal{A}_{j-1})$. By definition, $x_j \notin \mathrm{off}(e_j)$. So, we conclude that $z \notin X(\mathcal{A}_{j-1}) \cup \{x_j\} = X(\mathcal{A}_j)$. Furthermore, $z \in \mathrm{off}(e')$ for any edge e' incoming at an ancestor of v_j, and so $e' \notin E_d(X(\mathcal{A}_j))$ for any such edge e'. It follows that the only edges in $E_d(X(\mathcal{A}_j)) \setminus E_d(X(\mathcal{A}_{j-1}))$ must be between v_j and x_j, that is, in P_{v_j,x_j}. We conclude that $E_d(X(\mathcal{A}_j)) \subseteq E_d(X(\mathcal{A}_{j-1})) \cup P_{v_j,x_j} \subseteq E^+(\mathcal{A}_{j-1}) \cup P_{v_j,x_j} = E^+(\mathcal{A}_j)$.

Letting $j = |\mathcal{A}|$, we have $E_d(X(\mathcal{A})) \subseteq E^+(\mathcal{A})$. Thereby, we can conclude that $\lambda(E_d(X(\mathcal{A}))) \leq \lambda(E^+(\mathcal{A})) = \sum_{e \in E^+(\mathcal{A})} |c(e)| = |c(E^+(\mathcal{A}))|$, where the last equality holds because $E^+(\mathcal{A})$ has unique colors. We have $\lambda(E_d(X(\mathcal{A}))) \leq \overline{D}$ since $|c(E^+(\mathcal{A}))| \leq \overline{D}$ and so $\mathrm{PD}_{\mathcal{T}}(S) \geq D$, as required. □

In the following, we continue to define a few more things. Then, we present Algorithm ($\overline{D}$) for solving EX-$\overline{D}$-C-TIME-PD. Afterward, we analyze the correctness and the running time of Algorithm ($\overline{D}$). And finally, we show how to reduce instances of TIME-PD to instances of EX-$\overline{D}$-C-TIME-PD to conclude the desired `FPT`-result.

Define $\overline{H_p} := \ell(Z_p) - H_p$ to be the number of person-hours one would need additionally to be able to save all taxa in Z_p (when not regarding the time constraints $\mathrm{ex}_1, \ldots, \mathrm{ex}_{p-1}$). A set of taxa $A \subseteq X$ is *ex-q-compatible* for $q \in [\mathrm{var}_{\mathrm{ex}}]$ if $\ell(A \cap Z_p) \geq \overline{H_p}$ for each $p \in [q-1]$. Note that for there to be a valid T-schedule saving $S \subseteq X$, it must be satisfied that $\ell(S \cap Z_p) \leq H_p$, and so $\ell((X \setminus S) \cap Z_p) \geq \overline{H_p}$,

for each $p \in [\text{var}_{\text{ex}}]$. Thus, if A is ex-q-compatible then there is a valid T-schedule saving $(X \setminus A) \cap Z_{q-1}$. Observe that this schedule does not necessarily save $(X \setminus A) \cap Z_q$.

In our dynamic programming algorithm, we need to track the existence of color-respectful anchored taxa sets using particular sets of colors. To this end, we define the notion of *ex-(C_1, C_2, q)-feasibility*. For sets of colors $C_1, C_2 \subseteq [2\overline{D}]$ and an integer $q \in [\text{var}_{\text{ex}}]$, an anchored taxa set $\mathcal{A}$ is *ex-(C_1, C_2, q)-feasible* if

F a) $\mathcal{A}$ is color-respectful,
F b) $c(E^+(\mathcal{A}))$ is a subset of C_1,
F c) $\hat{c}(e)$ is in $C_2 \cup c(E^+(\mathcal{A} \setminus \{(x, v, e)\}))$ for each $(x, v, e) \in \mathcal{A}$,
F d) $X(\mathcal{A})$ is a subset of Z_q, and
F e) $X(\mathcal{A})$ is ex-q-compatible.

Intuitively, we want anchored taxa sets $\mathcal{A}$ to be ex-(C_1, C_2, q)-feasible if it is possible to save $X \setminus X(\mathcal{A})$ and only lose the colors in C_1 and have the colors in C_2 available for later actions. As an example, observe that the anchored taxa set $\mathcal{A}$ in Figure 4.2 is ex-(C_1, C_2, q)-feasible for $C_1 = [11]$, $C_2 = \{12\}$, and $q = 3$ if and only if $\mathcal{A}$ is q-compatible (which we don't know as the teams are not given). This would be the case if and only if $\overline{H_1} \geq 10$, $\overline{H_2} \geq 22$, and $\overline{H_3} \geq 35$.

Our algorithm calculates for each combination C_1, C_2, and q the maximum rescue length of the taxa of an ex-(C_1, C_2, q)-feasible anchored taxa set. In order to calculate these values recursively, we declare which tuples (x, v, e) are suitable for consideration in the recurrence of the algorithm.

A tuple (x, v, e) is *(C_1, C_2)-good* for sets of colors $C_1, C_2 \subseteq [2\overline{D}]$ if

G a) $P_{v,x}$ has unique colors,
G b) $c(P_{v,x})$ is a subset of C_1,
G c) $\hat{c}(e)$ is in C_2, and
G d) $P_{v,x}$ and $E_{>\overline{D}}$ are disjoint.

Let $\mathcal{X}_{(q)}$ be the set of tuples (x, v, e) such that $x \in Z_q$, $P_{v,x}$ has unique colors, $P_{v,x} \subseteq E_{\leq \overline{D}}$ and $\hat{c}(e) \notin c(P_{v,x})$. Disjoint sets of colors $C_1, C_2 \subseteq [2\overline{D}]$ are *q-grounding*, if $c(P_{v,x}) \not\subseteq C_1$ or $\hat{c}(e) \notin C_2$ for every tuple $(x, v, e) \in \mathcal{X}_{(q)}$. In Figure 4.2 the set $\mathcal{X}_{(1)}$ would be $\{(x_1, v_1, e_5), (x_1, v_0, e_3)\}$—because both paths P_{v_0,x_4} and P_{v_1,x_4} contain the edge $e_7 \in E_{>\overline{D}}$, and $\hat{c}(e_2) = 9 \in c(e_4)$ so (x_1, v_0, e_2) is not contained in $\mathcal{X}_{(1)}$. Therefore, the 1-grounding colors are any disjoint sets of colors C_1 and C_2 with ($c(e_4) \not\subseteq C_1$ or $\hat{c}(e_5) = 6 \notin C_2$) and ($c(\{e_1, e_4\}) \not\subseteq C_1$ or $\hat{c}(e_3) = 12 \notin C_2$). One non-trivial example are the sets $C_1 = \{6, 7, \ldots, 11\}$ and $C_2 = [5] \cup \{12\}$. The empty sets C_1 and C_2 are q-grounding for each q.

Algorithm ($\overline{D}$).
Let $\mathcal{I}$ be an instance of EX-$\overline{D}$-C-TIME-PD. In the following, we define a dynamic programming algorithm with table DP. For all disjoint sets of colors $C_1, C_2 \subseteq [2\overline{D}]$ with $|C_1| \leq \overline{D}$, and all $q \in [\mathrm{var_{ex}}]$, we compute a value $\mathrm{DP}[C_1, C_2, q]$. The entries of DP are computed in some order such that $\mathrm{DP}[C_1', C_2', q']$ is computed before $\mathrm{DP}[C_1'', C_2'', q'']$ if $C_1' \subsetneq C_1''$. We want $\mathrm{DP}[C_1, C_2, q]$ to store the *maximum* rescue length $\ell(X(\mathcal{A}))$ of an anchored taxa set $\mathcal{A}$ that is ex-(C_1, C_2, q)-feasible. If there exists no ex-(C_1, C_2, q)-feasible anchored taxa set, we want $\mathrm{DP}[C_1, C_2, q]$ to store $-\infty$.

As a base case, given $q \in [\mathrm{var_{ex}}]$ and q-grounding sets of colors C_1 and C_2. If $\overline{H_p} \leq 0$ for each $p \in [q-1]$, we store $\mathrm{DP}[C_1, C_2, q] = 0$. Otherwise, if C_1 and C_2 are q-grounding but $\overline{H_p} > 0$ for any $p \in [q-1]$, we store $\mathrm{DP}[C_1, C_2, q] = -\infty$.

To compute further values, we use the recurrence

$$\mathrm{DP}[C_1, C_2, q] = \max_{(x,v,e)} \mathrm{DP}[C_1', C_2', \mathrm{ex}^*(x)] + \ell(x). \tag{4.4}$$

Here, $C_1' := C_1 \setminus c(P_{v,x})$ and $C_2' := (C_2 \cup c(P_{v,x})) \setminus \{\hat{c}(e)\}$. The maximum in Recurrence (4.4) is taken over (C_1, C_2)-good tuples (x, v, e) satisfying $x \in Z_q$ and $\mathrm{DP}[C_1', C_2', \mathrm{ex}^*(x)] + \ell(x) \geq \max\{\overline{H_{\mathrm{ex}^*(x)}}, \overline{H_{\mathrm{ex}^*(x)+1}}, \ldots, \overline{H_{q-1}}\}$. If no such tuple exists, we set $\mathrm{DP}[C_1, C_2, q] = -\infty$. Recall that $\mathrm{ex}^*(x) = q$ for each $x \in Y_q$.

We return `yes` if $\mathrm{DP}[C_1, C_2, \mathrm{var_{ex}}]$ stores at least $\overline{H_{\mathrm{var_{ex}}}}$ for some disjoint sets of colors $C_1, C_2 \subseteq [2\overline{D}]$ with $|C_1| \leq \overline{D}$. Otherwise, if $\mathrm{DP}[C_1, C_2, \mathrm{var_{ex}}] \geq \overline{H_{\mathrm{var_{ex}}}}$ for all disjoint sets of colors C_1 and C_2, we return `no`.

We continue to analyze the algorithm by showing that $\mathrm{DP}[C_1, C_2, q]$ stores the largest rescue length $\ell(X(\mathcal{A}))$ of an ex-(C_1, C_2, q)-feasible anchored taxa set $\mathcal{A}$. We start with the base case.

Lemma 4.10. *For each $q \in [\mathrm{var_{ex}}]$ and all q-grounding sets of colors C_1 and C_2, $\mathrm{DP}[C_1, C_2, q]$ stores the largest rescue length $\ell(X(\mathcal{A}))$ of an ex-(C_1, C_2, q)-feasible anchored taxa set $\mathcal{A}$ if such an $\mathcal{A}$ exists, and $-\infty$, otherwise.*

Proof. Let $q \in [\mathrm{var_{ex}}]$ and let C_1 and C_2 be q-grounding sets of colors, and suppose for a contradiction that some non-empty anchored taxa set $\mathcal{A}$ is ex-(C_1, C_2, q)-feasible. Let (x, v, e) be the last tuple in a valid ordering of $\mathcal{A}$. Such an ordering exists because $\mathcal{A}$ is color-respectful. Observe that $\hat{c}(e) \notin c(E^+(\mathcal{A}))$, and in particular $\hat{c}(e) \notin c(P_{v,x})$. As $\mathcal{A}$ satisfies CR a), CR c) and F d), we also have that $c(P_{v,x})$ is uniquely colored, $P_{v,x} \subseteq E_{\leq \overline{D}}$ and $x \in Z_q$. Thus, (x, v, e) is in $\mathcal{X}_{(q)}$. Because $\mathcal{A}$ satisfies F b) we have $c(P_{v,x}) \subseteq C_1$, and because $\mathcal{A}$ satisfies F c) we have $\hat{C}_2 \cup c(e) \in c(E^+(\mathcal{A}))$, which together with $\hat{c}(e) \notin c(E^+(\mathcal{A}))$ implies $\hat{c}(e) \in C_2$.

But then (x, v, e) is a tuple in $\mathcal{X}_{(q)}$ with $c(P_{v,x}) \subseteq C_1$ and $\hat{c}(e) \in C_2$, a contradiction to the assumption that C_1 and C_2 are q-grounding. It follows that no non-empty anchored taxa set is ex-(C_1, C_2, q)-feasible.

Observe that the empty set is ex-(C_1, C_2, q)-feasible if and only if the empty set is ex-q-compatible, as the empty set trivially satisfies all other conditions. Since $\ell(\emptyset \cap Z_p) = \ell(\emptyset) = 0$ for each $p \in [\mathrm{var}_{\mathrm{ex}}]$, we observe that the empty set is ex-q-compatible if and only if $\overline{H_p} \leq 0$ for each $p \in [q-1]$. Exactly in these cases, entry $\mathrm{DP}[C_1, C_2, q]$ stores at least $\ell(\emptyset) = 0$. □

In the following, we show an induction over $|C_1|$. Observe that $C_1 = \emptyset$ and any set of colors $C_2 \subseteq [2\overline{D}]$ are q-grounding for each $q \in [\mathrm{var}_{\mathrm{ex}}]$. Therefore, Lemma 4.10 is the base case of the induction.

As an induction hypothesis, we assume that for a fixed set of colors $C_1 \subseteq [2\overline{D}]$ and a fixed $q \in [\mathrm{var}_{\mathrm{ex}}]$, for each $K_1 \subsetneq C_1$, $K_2 \subseteq [2\overline{D}] \setminus K_1$, and $p \in [q]$, entry $\mathrm{DP}[K_1, K_2, p]$ stores the largest rescue length $\ell(X(\mathcal{A}))$ of an ex-(K_1, K_2, p)-feasible anchored taxa set $\mathcal{A}$.

In Lemma 4.11 and Lemma 4.12, we proceed to show that with this hypothesis we can conclude that $\mathrm{DP}[C_1, C_2, p]$ stores the desired value.

Lemma 4.11. *If an anchored taxa set $\mathcal{A}$ is ex-(C_1, C_2, q)-feasible for disjoint sets of colors C_1 and C_2 that are not q-grounding, then* $\mathrm{DP}[C_1, C_2, q] \geq \ell(X(\mathcal{A}))$.

Proof. Let C_1 and C_2 be disjoint sets of colors that are not q-grounding. Furthermore, let $\mathcal{A}$ be an ex-(C_1, C_2, q)-feasible anchored taxa set.

We prove the claim by an induction on $|\mathcal{A}|$. First, consider $\mathcal{A}$ being empty. Since C_1 and C_2 are not q-grounding, there is a tuple $(x, v, e) \in \mathcal{X}_{(q)}$ such that $P_{v,x} \subseteq C_1$ and $\hat{c}(e) \in C_2$. Because $(x, v, e) \in \mathcal{X}_{(q)}$, the anchored taxa set $\mathcal{A}' := \{(x, v, e)\}$ is color-respectful, and by construction, $\mathcal{A}$ satisfies F b), F c), and F d). It only remains to show that $\{x\}$ is ex-q-compatible to prove that $\mathcal{A}'$ is ex-(C_1, C_2, q)-feasible. Since $\mathcal{A} = \emptyset$ is ex-(C_1, C_2, q)-feasible, $\ell(x) > \ell(\emptyset) \geq \overline{H_p}$ for each $p \in [q-1]$. Thus, $\mathcal{A}'$ is ex-(C_1, C_2, q)-feasible. This is sufficient to show that $\mathrm{DP}[C_1, C_2, q] \geq \ell(x)$. So, we may assume that $\mathcal{A}$ is not empty.

Now, let (x, v, e) be the last tuple in a valid ordering of $\mathcal{A}$. Such an ordering exists because $\mathcal{A}$ is color-respectful. Define an anchored taxa set $\mathcal{A}'$ resulting from removing (x, v, e) from $\mathcal{A}$.

We first show that $\mathcal{A}'$ is ex-$(C_1', C_2', \mathrm{ex}^*(x))$-feasible.

F a) [$\mathcal{A}$ is color-respectful]
Because $\mathcal{A}$ is color-respectful and x is the taxon of the largest order, we directly conclude that $\mathcal{A}'$ satisfies all properties of color-respectfulness.

F b) $[c(E^+(\mathcal{A})) \subseteq C_1]$
Because $E^+(\mathcal{A}') = E^+(\mathcal{A}) \setminus P_{v,x}$ and $E^+(\mathcal{A})$ has unique colors, we conclude that $c(E^+(\mathcal{A}')) = c(E^+(\mathcal{A})) \setminus c(P_{v,x}) \subseteq C_1 \setminus c(P_{v,x}) = C_1'$.

F c) $[\hat{c}(e) \in C_2 \cup c(E^+(\mathcal{A} \setminus \{(x,v,e)\}))$ for each $(x,v,e) \in \mathcal{A}]$
Fix a tuple $(y,u,e') \in \mathcal{A}'$. We want to show that $\hat{c}(e')$ is in

$$\begin{aligned} & C_2' \cup c(E^+(\mathcal{A}' \setminus \{(y,u,e')\})) \\ = \; & ((C_2 \cup c(P_{v,x})) \setminus \{\hat{c}(e)\}) \cup (c(E^+(\mathcal{A} \setminus \{(y,u,e')\})) \setminus c(P_{v,x})) \\ = \; & (C_2 \cup c(E^+(\mathcal{A} \setminus \{(y,u,e')\}))) \setminus \{\hat{c}(e)\} \end{aligned}$$

It holds $\hat{c}(e') \in C_2 \cup c(E^+(\mathcal{A} \setminus \{(y,u,e')\}))$, because $\mathcal{A}$ satisfies F c). Since $E_s(\mathcal{A})$ has unique key-colors, $\hat{c}(e') \neq \hat{c}(e)$. Therefore, $\hat{c}(e') \in C_2' \cup c(E^+(\mathcal{A}' \setminus \{(y,u,e')\}))$.

F d) $[X(\mathcal{A}) \subseteq Z_q]$
$X(\mathcal{A}') \subseteq X(\mathcal{A}) \subseteq Z_{\mathrm{ex}^*(x)}$ by the definition of x.

F e) $[X(\mathcal{A})$ is ex-q-compatible]
Because $X(\mathcal{A})$ is ex-q-compatible, $\ell(X(\mathcal{A}') \cap Z_p) = \ell(X(\mathcal{A}) \cap Z_p) \geq \overline{H_p}$ for each $p < \mathrm{ex}^*(x)$. Consequently, $X(\mathcal{A}')$ is ex-$\mathrm{ex}^*(x)$-compatible.

Next, we show that (x,v,e) is (C_1, C_2)-good.

G a) $[P_{v,x}$ has unique colors]
Because $E^+(\mathcal{A})$ has unique colors, $P_{v,x} \subseteq E^+(\mathcal{A})$ has unique colors, too.

G b) $[c(P_{v,x}) \subseteq C_1]$
$c(P_{v,x}) \subseteq c(E^+(\mathcal{A})) \subseteq C_1$ because $\mathcal{A}$ satisfies F b).

G c) $[\hat{c}(e) \in C_2]$
Because $\mathcal{A}$ satisfies F c) and $X(\mathcal{A})$ has a valid ordering, $\hat{c}(e) \in C_2 \cup c(E^+(\mathcal{A}'))$ and $\hat{c}(e) \notin c(E^+(\mathcal{A}))$. Therefore, especially $\hat{c}(e) \notin c(E^+(\mathcal{A}')) \subseteq c(E^+(\mathcal{A}))$ and we conclude that $\hat{c}(e) \in C_2$.

G d) $[P_{v,x}$ and $E_{>\overline{D}}$ are disjoint]
Because $E^+(\mathcal{A})$ has an empty intersection with $E_{>\overline{D}}$, also $P_{v,x} \subseteq E^+(\mathcal{A})$ has empty intersection with $E_{>\overline{D}}$.

Since $\mathcal{A}'$ is ex-$(C_1', C_2', \mathrm{ex}^*(x))$-feasible and $|\mathcal{A}'| < |\mathcal{A}|$, we have by the inductive hypothesis that $\mathrm{DP}[C_1', C_2', \mathrm{ex}^*(x)] \geq \ell(X(\mathcal{A}'))$. Furthermore $\mathrm{DP}[C_1', C_2', \mathrm{ex}^*(x)] + \ell(x) \geq \ell(X(\mathcal{A}')) + \ell(x) \geq \ell(X(\mathcal{A}))$. As $X(\mathcal{A})$ is ex-(C_1, C_2, q)-compatible we have $\ell(X(\mathcal{A}) = \ell(X(\mathcal{A}) \cap Z_p) \geq \overline{H_p}$ for each p with $\mathrm{ex}^*(x) \leq p < q$, and so $\mathrm{DP}[C_1', C_2', \mathrm{ex}^*(x)] + \ell(x) \geq \max\{\overline{H_{\mathrm{ex}^*(x)}}, \overline{H_{\mathrm{ex}^*(x)+1}}, \ldots, \overline{H_{q-1}}\}$. This together with the fact that (x,v,e) is (C_1, C_2)-good implies that (x,v) satisfies the conditions of Recurrence (4.4), from which we can conclude that $\mathrm{DP}[C_1, C_2, q] \geq \mathrm{DP}[C_1', C_2', \mathrm{ex}^*(x)] + \ell(x) \geq \ell(X(\mathcal{A}))$. □

Lemma 4.12. *If* $\mathrm{DP}[C_1, C_2, q] = a \geq 0$ *for some disjoint sets of colors* C_1 *and* C_2*, then there is an ex-*(C_1, C_2, q)*-feasible anchored taxa set* $\mathcal{A}$ *with* $\ell(X(\mathcal{A})) = a$*.*

Proof. Assume that $\mathrm{DP}[C_1, C_2, q] = 0$. Then, C_1 and C_2 must be q-grounding, as otherwise the algorithm would apply Recurrence (4.4), and then $\mathrm{DP}[C_1, C_2, q]$ would store either $-\infty$ or a value which is at least $\ell(x) > 0$ for some $x \in X$. Lemma 4.10 shows the correctness of this case.

Now, assume that $\mathrm{DP}[C_1, C_2, q] = a > 0$. By Recurrence (4.4), there is a (C_1, C_2)-good tuple (x, v, e) such that $x \in Z_q$ and $\mathrm{DP}[C_1, C_2, q] = \mathrm{DP}' + \ell(x)$. Here, we define $\mathrm{DP}' := \mathrm{DP}[C_1', C_2', q']$ where (as in Recurrence (4.4)) $C_1' := C_1 \setminus c(P_{v,x})$, $C_2' := (C_2 \cup c(P_{v,x})) \setminus \{\hat{c}(e)\}$, and $q' := \mathrm{ex}^*(x)$. Further, we know that $\mathrm{DP}' + \ell(x) \geq \max\{\overline{H_{q'}}, \overline{H_{q'+1}}, \ldots, \overline{H_{q-1}}\}$. We conclude by the induction hypothesis that there is an ex-(C_1', C_2', q')-feasible anchored taxa set $\mathcal{A}'$ such that $\ell(X(\mathcal{A}')) = \mathrm{DP}'$. Define $\mathcal{A} := \mathcal{A}' \cup \{(x, v, e)\}$. Clearly, $a = \mathrm{DP}' + \ell(x) = \ell(X(\mathcal{A}))$. It remains to show that $\mathcal{A}$ is an ex-(C_1, C_2, q)-feasible set.

F a) [$\mathcal{A}$ is color-respectful]
We show that $\mathcal{A}$ is color-respectful.

CR a) [$E^+(\mathcal{A})$ has unique colors]
As $\mathcal{A}'$ satisfies CR a) and F b), $E^+(\mathcal{A}')$ has unique colors and $c(E^+(\mathcal{A}')) \subseteq C_1' = C_1 \setminus c(P_{v,x})$. Consequently, the colors of $E^+(\mathcal{A}')$ and $P_{v,x}$ are disjoint. We conclude with the (C_1, C_2)-goodness of (x, v, e), that $P_{v,x}$ has unique colors and so also $E^+(\mathcal{A})$.

CR b) [$E_s(\mathcal{A})$ has unique key-colors]
As $\mathcal{A}'$ satisfies CR b), $E_s(\mathcal{A}')$ has unique key-colors. Observe $E_s(\mathcal{A}) = E_s(\mathcal{A}') \cup \{e\}$. Because $\mathcal{A}'$ satisfies F c), $\hat{c}(E_s(\mathcal{A}')) \subseteq C_2' \cup c(E^+(\mathcal{A}')) \subseteq C_1' \cup C_2'$. Further, because $\hat{c}(e) \in C_2$ and $C_1 \cap C_2 = \emptyset$ we have $\hat{c}(e) \notin C_1$, which implies $\hat{c}(e) \notin C_1 \cup (C_2 \setminus \{\hat{c}(e)\}) = C_1' \cup C_2'$. Therefore, $E_s(\mathcal{A})$ has unique key-colors.

CR c) [$E^+(\mathcal{A})$ and $E_{>\overline{D}}$ are disjoint]
$E^+(\mathcal{A}')$ and $E_{>\overline{D}}$ are disjoint because $\mathcal{A}'$ is color-respectful. Further, because (x, v, e) is (C_1, C_2)-good, $P_{v,x}$ and $E_{>\overline{D}}$ are disjoint and so $E^+(\mathcal{A}') \subseteq E_{\leq \overline{D}}$.

CR d) [$\mathcal{A}$ has a valid ordering]
$\mathcal{A}'$ has a valid ordering $(y_1, u_1, e_1), \ldots, (y_{|\mathcal{A}'|}, u_{|\mathcal{A}'|}, e_{|\mathcal{A}'|})$. As $X(\mathcal{A}') \subseteq Z_{\mathrm{ex}^*(x)}$ we conclude $\mathrm{ex}(y) \leq \mathrm{ex}(x)$ for each $y \in A'$. Further, $\hat{c}(e) \in C_2$ and $c(P_{v,x}) \subseteq C_1$ because (x, v, e) is (C_1, C_2)-good. Since $\mathcal{A}'$ satisfies F b), $c(E^+(A')) \subseteq C_1' \subseteq C_1$. Because C_1 and C_2 are disjoint, we conclude that $\hat{c}(e) \notin c(E^+(\mathcal{A}))$. We conclude that $(y_1, u_1, e_1), \ldots, (y_{|\mathcal{A}'|}, u_{|\mathcal{A}'|}, e_{|\mathcal{A}'|}), (x, v, e)$ is a valid ordering of $\mathcal{A}$.

CR e) [$P_{v,x}$ and $P_{u,y}$ are disjoint for any $(x, v, e), (y, u, e') \in \mathcal{A}$]
We know that $\mathcal{A}'$ satisfies CR e). Further, as already seen, $c(E^+(\mathcal{A}'))$ and $c(P_{v,x})$

are disjoint and therefore also $P_{v,x}$ and $P_{u,y}$ are disjoint for each $(y, u, e') \in \mathcal{A}'$. It follows that $\mathcal{A}$ is color-respectful.

F b) $[c(E^+(\mathcal{A})) \subseteq C_1]$
$c(E^+(\mathcal{A})) = c(E^+(\mathcal{A}')) \cup c(P_{v,x}) \subseteq C_1' \cup c(P_{v,x}) = C_1$.

F c) $[\hat{c}(e) \in C_2 \cup c(E^+(\mathcal{A} \setminus \{(x, v, e)\}))$ for each $(x, v, e) \in \mathcal{A}]$
Observe $\hat{c}(e) \in C_2$ because (x, v, e) is (C_1, C_2)-good. Each $(y, u, e') \in \mathcal{A}'$ satisfies $\hat{c}(e') \subseteq C_2' \cup c(E^+(\mathcal{A}' \setminus \{(y, u, e')\}))$, as $\mathcal{A}'$ satisfies F c). Observe $x \neq y$ and $C_2' \subseteq C_2 \cup c(P_{v,x})$ and therefore $\hat{c}(e') \subseteq C_2 \cup c(E^+(\mathcal{A} \setminus \{(y, u, e')\}))$.

F d) $[X(\mathcal{A}) \subseteq Z_q]$
$X(\mathcal{A}) \subseteq Z_q$ because $X(\mathcal{A}') \subseteq Z_q$ and $x \in Z_q$.

F e) [$X(\mathcal{A})$ is ex-q-compatible]
Because $\mathcal{A}'$ is ex-q'-compatible, we conclude that $\ell(X(\mathcal{A}) \cap Z_p) = \ell(X(\mathcal{A}') \cap Z_p) \geq \overline{H_p}$ for each $p \in [q' - 1]$. Then, with $\mathrm{DP}' + \ell(x) \geq \max\{\overline{H_{q'}}, \overline{H_{q'+1}}, \ldots, \overline{H_{q-1}}\}$ we conclude that $\ell(X(\mathcal{A}) \cap Z_p) = \ell(X(\mathcal{A})) \geq \overline{H_p}$ for each $p \in \{q', \ldots, q - 1\}$. Therefore, $X(\mathcal{A})$ is ex-q-compatible and ex-(C_1, C_2, q)-feasible. □

Lemma 4.13. *An instance $\mathcal{I}$ of* ex-$\overline{D}$-c-Time-PD *is a* `yes`*-instance if and only if disjoint sets of colors $C_1, C_2 \subseteq [2\overline{D}]$ with $|C_1| \leq \overline{D}$ and an ex-$(C_1, C_2, \mathrm{var}_{\mathrm{ex}})$-feasible anchored taxa set $\mathcal{A}$ with $\ell(X(\mathcal{A})) \geq \overline{H_{\mathrm{var}_{\mathrm{ex}}}}$ exist.*

Proof. Suppose first that $\mathcal{I}$ is a `yes`-instance of ex-$\overline{D}$-c-Time-PD. That is, there exists a color-respectful anchored taxa set $\mathcal{A}$ with $|c(E^+(\mathcal{A}))| \leq \overline{D}$ such that there is a valid T-schedule saving $S := X \setminus X(\mathcal{A})$. Let $C_1 := c(E^+(\mathcal{A}))$ and let C_2 be the key-colors of the edges $E_s(\mathcal{A})$ that are not in C_1. We show that $\mathcal{A}$ is ex-$(C_1, C_2, \mathrm{var}_{\mathrm{ex}})$-feasible. First, $\mathcal{A}$ satisfies F a), F b), and F d) by definition.

F c) $[\hat{c}(e) \in C_2 \cup c(E^+(\mathcal{A} \setminus \{(x, v, e)\}))$ for each $(x, v, e) \in \mathcal{A}]$
By the definition, $\hat{c}(E(\mathcal{A})) \subseteq C_2 \cup C_1 = C_2 \cup c(E^+(\mathcal{A}))$. Fix a tuple $(x, v, e) \in \mathcal{A}$. Because $\mathcal{A}$ has a valid ordering, we conclude that $\hat{c}(e) \notin c(P_{v,x})$. It follows that if $\hat{c}(e)$ is in $c(E^+(\mathcal{A}))$, then explicitly $\hat{c}(e) \in c(E^+(\mathcal{A} \setminus \{(x, v, e)\}))$. Therefore, $\mathcal{A}$ satisfies F c).

F e) [$X(\mathcal{A})$ is ex-q-compatible]
Fix some $q \in [\mathrm{var}_{\mathrm{ex}}]$. Since S has a valid schedule, we conclude $\ell(S \cap Z_q) \leq H_q$ for each $q \in [\mathrm{var}_{\mathrm{ex}}]$. Consequently, $\ell(X(\mathcal{A}) \cap Z_q) = \ell(Z_q) - \ell(S \cap Z_q) \geq \overline{H_q}$. Likewise with $\ell(S) \leq H_{\mathrm{var}_{\mathrm{ex}}}$, we conclude that $\ell(X(\mathcal{A})) \geq \overline{H_{\mathrm{var}_{\mathrm{ex}}}}$.

For the converse, suppose that $\mathcal{A}$ is an ex-$(C_1, C_2, \mathrm{var}_{\mathrm{ex}})$-feasible anchored taxa set with $\ell(X(\mathcal{A})) \geq \overline{H_{\mathrm{var}_{\mathrm{ex}}}}$ for disjoint sets of colors $C_1, C_2 \subseteq [2\overline{D}]$ with $|C_1| \leq \overline{D}$. Observe that the rescue time of all taxa in $S = X \setminus X(\mathcal{A})$ that are in Z_q

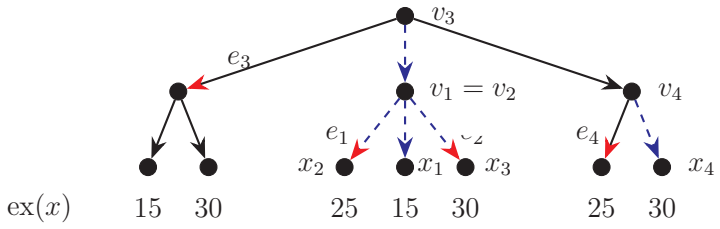


Figure 4.3: Example of the construction of an anchored taxa set $\mathcal{A}$ (Theorem 4.2). Here given $X \setminus S = \{x_1, x_2, x_3, x_4\}$, the constructed $\mathcal{A}$ is $\{(x_i, v_i, e_i) \mid i \in [4]\}$. Blue dashed edges are in $E_d(X \setminus S) = E^+(\mathcal{A})$ and edges with red arrow are in $E_s(\mathcal{A})$. $\mathcal{A}_1$ could have been $\{(x_1, v_1, e_2)\}$ provoking that (x_1, v_1, e_2) must have been replaced in step $i = 2$.

is $\ell(S \cap Z_q) = \ell((X \setminus X(\mathcal{A})) \cap Z_q) = \ell(Z_q) - \ell(X(\mathcal{A}) \cap Z_q)$. Because $\mathcal{A}$ is ex-var$_{\text{ex}}$-compatible, $\ell(X(\mathcal{A}) \cap Z_q) \geq \overline{H_q}$ and so $\ell(S \cap Z_q) \leq \ell(Z_q) - \overline{H_q} = H_q$. Then, by Lemma 4.2, there is a valid T-schedule saving S. By definition, $\mathcal{A}$ is color-respectful and $|c(E^+(\mathcal{A}))| \leq |C_1| \leq \overline{D}$. Therefore, $\mathcal{I}$ is a `yes`-instance of EX-$\overline{D}$-C-TIME-PD □

Lemma 4.14. *Algorithm* ($\overline{D}$) *computes solutions for instances of* EX-$\overline{D}$-C-TIME-PD *in time* $\mathcal{O}^*(9^{\overline{D}} \cdot \overline{D})$.

Proof. Since DP$[C_1, C_2, q]$ is computed for C_1 and C_2 disjoint subsets of $[2\overline{D}]$, the table DP contains $3^{2\overline{D}} \cdot \text{var}_{\text{ex}}$ entries.

For given $q \in [\text{var}_{\text{ex}}]$ and color sets C_1 and C_2, we can compute whether C_1 and C_2 are q-grounding by testing the conditions for each tuple (x, v, e). Therefore, the test can be done in $\mathcal{O}(\overline{D} \cdot n^3)$ time. Here, the $\overline{D}$-factor in the running time comes from testing whether $P_{v,x} \subseteq C_1$.

Analogously, for a given $q \in [\text{var}_{\text{ex}}]$ and sets of colors C_1 and C_2, we can compute whether a tuple (x, v, e) satisfies the conditions of Recurrence (4.4) in $\mathcal{O}(\overline{D} \cdot n)$ time in our RAM model. We, however, want to note that we add two numbers of size $\max\{\overline{H_1}, \overline{H_2}, \ldots, \overline{H_{\text{var}_{\text{ex}}}}\}$. □

Theorem 4.2. TIME-PD *can be solved in* $\mathcal{O}^*(2^{6.056 \cdot \overline{D} + o(\overline{D})})$ *time.*

Just like in the previous section, the key idea is that we construct a family of colorings $\mathcal{C}$ on the edges of $\mathcal{T}$, where each edge $e \in E$ is assigned a key-color and additionally for each $e \in E_{\leq \overline{D}}$ a subset $c^-(e)$ of $[2\overline{D}]$ of size $\lambda(e) - 1$. Using these, we generate $|\mathcal{C}|$ instances of EX-$\overline{D}$-C-TIME-PD, which with Algorithm ($\overline{D}$) we solve

in $\mathcal{O}^*(9^{\overline{D}} \cdot \overline{D})$ time. The colorings are constructed in such a manner that $\mathcal{I}$ is a `yes`-instance if and only if at least one of the constructed EX-$\overline{D}$-C-TIME-PD instances is a `yes`-instance. Central for this is the concept of a perfect hash family as defined in Definition 2.14.

Proof of Theorem 4.2. Reduction. Let an instance $\mathcal{I} = (\mathcal{T}, \mathrm{ex}, \ell, T, D)$ of TIME-PD be given. Let $|E_{\leq \overline{D}}| = m_1$ and $|E_{>\overline{D}}| = m - m_1$. Order the edges $e_1, \ldots, e_{m_1}$ of $E_{\leq \overline{D}}$ and the edges $e_{m_1+1}, \ldots, e_m$ of $E_{>\overline{D}}$, arbitrarily. We define integers $W_0 := m$ and $W_j := m + \sum_{i=1}^{j}(\lambda(e_i) - 1)$ for each $j \in [m_1]$.

Let $\mathcal{F}$ be a $(W_m, 2\overline{D})$-perfect hash family. Now, we define the family $\mathcal{C}$ of colorings as follows. For every $f \in \mathcal{F}$, let $\hat{c}_f$ and c_f^- be colorings such that $\hat{c}_f(e_j) = f(j)$ for each $e_j \in E(\mathcal{T})$; and $c_f^-(e_j) := \{f(W_{j-1}+1), \ldots, f(W_j)\}$ for each $e_j \in E_{\leq \overline{D}}$.

For each $c_f \in \mathcal{C}$, let $\mathcal{I}_f = (\mathcal{T}, \mathrm{ex}, \ell, T, D, \hat{c}_f, c_f^-)$ be the corresponding instance of EX-$\overline{D}$-C-TIME-PD. Now, solve each instance $\mathcal{I}_f$ using Algorithm $(\overline{D})$ and return `yes` if $\mathcal{I}_f$ is a `yes`-instance for some $c_f \in \mathcal{C}$. Otherwise, return `no`.

Correctness. If one of the constructed instances of EX-$\overline{D}$-C-TIME-PD is a `yes`-instance, then $\mathcal{I}$ is a `yes`-instance of TIME-PD, by Lemma 4.9.

For the converse, if $S \subseteq X$ is a solution for $\mathcal{I}$, then $\lambda(E') \geq D$ where E' is the set of edges which have at least one offspring in S. Therefore, $\lambda(E \setminus E') \leq \lambda(E) - D = \overline{D}$. Define $E_1 := E \setminus E' = E_d(X \setminus S)$ and let E_2' be the set of edges $e \in E'$ which have a sibling-edge in E_1. If e and e' are sibling-edges and both are in E_2', then remove e' from E_2'. Continuously repeat the previous step to receive E_2 in which any two edges $e, e' \in E_2$ are not sibling-edges. Observe $|E_2| \leq |E_1| \leq \lambda(E_1) \leq \overline{D}$.

Let $Z \subseteq [W_m]$ be the set of colors of E_1 and the key-colors of E_2. More precisely, $Z := \{j \in [m] \mid e_j \in E_1 \cup E_2\} \cup \{W_{j-1}+1, \ldots, W_j \mid e_j \in E_1\}$. Since $\mathcal{F}$ is a $(W_m, 2\overline{D})$-perfect hash family and $|Z| = \lambda(E_1) + |E_2| \leq 2\overline{D}$, there exists a function $f \in \mathcal{F}$ such that $f(z) \neq f(z')$ for distinct $z, z' \in Z$. It follows that E_1 has unique colors, E_2 has unique key-colors, and $c(E_1) \cap \hat{c}(E_2) = \emptyset$ in the constructed instance $\mathcal{I}_f$.

Let $C_1 = c(E_1)$ and $C_2 = \hat{c}(E_2)$. We claim that instance $\mathcal{I}_f$ has a color-respectful anchored taxa set $\mathcal{A}$ with $E^+(\mathcal{A}) = E_1$, $E_s(\mathcal{A}) \subseteq E_1 \cup E_2$, and $X(\mathcal{A}) = X \setminus S$. Since $|c(E_1)| = |C_1| \leq \overline{D}$ and there is a valid T-schedule saving S, this implies that $\mathcal{I}_f$ is a `yes`-instance of EX-$\overline{D}$-C-TIME-PD.

We define $\mathcal{A} := \{(x_i, v_i, e_i) \mid 1 \leq i \leq |X \setminus S|\}$, where x_i, v_i, and e_i are constructed iteratively as follows. Let $x_1, \ldots, x_{|X \setminus S|}$ be an ordering of the set of taxa $X \setminus S$ such that $\mathrm{ex}(x_i) \leq \mathrm{ex}(x_j)$ if $i \leq j$. Define $E_d^{(i)} := E_d(\{x_1, \ldots, x_i\})$ for each $1 \leq i \leq |X \setminus S|$ and let $E_d^{(0)}$ be empty. For each $i \in [|X \setminus S|]$, let $e' := v_i w_i$ be the highest edge in $E_d^{(i)} \setminus E_d^{(i-1)}$. This completes the construction of x_i and v_i. Construct e_i as follows.

If $v_j \neq v_i$ for each $j > i$, then let e_i be an arbitrary outgoing edge of v_i not in $E_d^{(i)}$. Such an edge exists as $v_i w_i$ is the topmost edge of $E_d^{(i)} \setminus E_d^{(i-1)}$ and therefore not all edges outgoing of v_i are in $E_d^{(i)}$. Otherwise, let j_i be the minimum integer such that $j_i > i$ and $v_{j_i} = v_i$. Then, let $e_i := v_i w_{j_i}$. Note that e_i is not in $E_d^{(i)}$ since e_i is the topmost edge of $E_d^{(j_i)} \setminus E_d^{(j_i-1)}$. See Figure 4.3 for an example.

It remains to show that $\mathcal{A}$ is color-respectful. $\mathcal{A}$ satisfies CR a) as by construction $E^+(\mathcal{A}) = \bigcup_{i=1}^{|X \setminus S|} P_{v_i,x_i} = E_d(X \setminus S) = E_1$. Similarly, $\mathcal{A}$ satisfies CR b) because $E_s(\mathcal{A}) \subseteq E_1 \cup E_2$. $\mathcal{A}$ satisfies CR c) by the fact that $\lambda(E_1) \leq \overline{D}$. $\mathcal{A}$ satisfies CR d), as $(x_1, v_1, e_1), \ldots, (x_{|\mathcal{A}|}, v_{|\mathcal{A}|}, e_{|\mathcal{A}|})$ is a valid ordering of $\mathcal{A}$.

To see that $\mathcal{A}$ satisfies CR e), consider any $(x_i, v_i, e_i), (x_j, v_j, e_j)$ for $i < j$. As vertex v_j is an ancestor of x_j, every edge in P_{v_j,x_j} has an offspring not in $\{x_1, \ldots, x_i\}$ and therefore P_{v_j,x_j} is disjoint from $E_d(x_1, \ldots, x_i)$, which contains all edges of P_{v_i,x_i}. It follows that P_{v_i,x_i} and P_{v_j,x_j} are disjoint.

Running Time. The construction of $\mathcal{C}$ takes $e^{2\overline{D}}(2\overline{D})^{\mathcal{O}(\log(2\overline{D}))} \cdot W_m \log W_m$ time, and for each $c \in \mathcal{C}$ the construction of each instance of EX-$\overline{D}$-C-TIME-PD takes time polynomial in $|\mathcal{I}|$. By Lemma 4.14, solving an instance of EX-$\overline{D}$-C-TIME-PD takes $\mathcal{O}^*(9^{\overline{D}} \cdot 2\overline{D})$ time, and $|\mathcal{C}| = e^{2\overline{D}}(2\overline{D})^{\mathcal{O}(\log(2\overline{D}))} \cdot \log W_m$ is the number of constructed instances of EX-$\overline{D}$-C-TIME-PD.

Thus, $\mathcal{O}^*(e^{2\overline{D}}(2\overline{D})^{\mathcal{O}(\log(2\overline{D}))} \log W_m \cdot (W_m + (9^{\overline{D}} \cdot 2\overline{D})))$ is the total running time. Because $W_m = \lambda(E_{\leq \overline{D}}) + |E_{>\overline{D}}| \leq |E| \cdot \overline{D} \leq 2n \cdot \overline{D}$, the running time further simplifies to $\mathcal{O}^*((3e)^{2\overline{D}} \cdot 2^{\mathcal{O}(\log^2(\overline{D}))}) = \mathcal{O}^*(2^{6.056 \cdot \overline{D} + o(\overline{D})})$. □

4.5 Further Parameterized Complexity Results

4.5.1 Parameterization by the Available Person-hours

In this subsection, we consider parameterization by the available person-hours $H_{\max_{\text{ex}}}$. Observe we may assume that $|T|, \max_{\text{ex}} \leq H_{\max_{\text{ex}}} \leq |T| \cdot \max_{\text{ex}}$.

Proposition 4.15. TIME-PD *and* S-TIME-PD *are* FPT *when parameterized by the available person-hours* $H_{\max_{\text{ex}}}$. *More precisely,*

(a) TIME-PD *can be solved in* $\mathcal{O}((|T|+1)^{2 \cdot \max_{\text{ex}}} \cdot n)$ *time,*

(b) TIME-PD *can be solved in* $\mathcal{O}((H_{\max_{\text{ex}}})^{2\,\text{var}_{\text{ex}}} \cdot n)$ *time, and*

(c) S-TIME-PD *can be solved in* $\mathcal{O}(3^{|T| \cdot \max_{\text{ex}}} \cdot |T| \cdot n)$ *time.*

Proof of Proposition 4.15a. Table definition. Let $\mathcal{I}$ be an instance of Time-PD and let $\mathbf{a} = (a_1, \ldots, a_{\max_{\text{ex}}})$ be a $\max_{\text{ex}}$-dimensional vector with $a_i \in [|T|]_0$.

We define a dynamic programming algorithm with a table DP. In entry $\mathrm{DP}[v, \mathbf{a}, b]$, for vertex v and integer $b \in \{0, 1\}$, we store 0 if $b = 0$; if $b = 1$, then we store the maximum value of $\mathrm{PD}_{\mathcal{T}_v}(S)$ for a non-empty subset $S \subseteq \mathrm{off}(v)$ that can be saved when having a_j available teams in timeslot $j \in [\max_{\text{ex}}]$. That is, if for every $j \in [\mathrm{var}_{\text{ex}}]$ we have $\sum_{x \in S \cap Z_j} \ell(x) \leq \sum_{i=1}^{\text{ex}_j} a_i$. If there is no such non-empty S, we store $-\infty$. We define an auxiliary table DP' in which in entry $\mathrm{DP}'[v, i, \mathbf{a}, b]$ we only consider non-empty sets $S \subseteq \mathrm{off}(u_1) \cup \cdots \cup \mathrm{off}(u_i)$ where $u_1, \ldots, u_z$ are the children of v.

Algorithm. As a base case, for each taxon x we store $\mathrm{DP}[x, \mathbf{a}, b] = 0$ if $\sum_{i=1}^{\text{ex}(x)} a_i \geq \ell(x)$ or if $b = 0$. Otherwise, we store $\mathrm{DP}[x, \mathbf{a}, 1] = -\infty$.

Let v be an internal vertex with children $u_1, \ldots, u_z$.

We define $\mathrm{DP}'[v, \mathbf{a}, b, 1] := \mathrm{DP}[u_1, \mathbf{a}, b] + \lambda(vu_1) \cdot b$ and we compute further values with the recurrence

$$\mathrm{DP}'[v, i+1, \mathbf{a}, b] = \max_{\mathbf{a}', b_1, b_2} \mathrm{DP}'[v, i, \mathbf{a}', b_1] + \mathrm{DP}[u_{i+1}, \mathbf{a} - \mathbf{a}', b_2] + \lambda(vu_{i+1}) \cdot b_2. \quad (4.5)$$

Here, we select $\mathbf{a}'$, b_1, and b_2 such that $a'_j \in [a_j]_0$ for each $j \in [\max_{\text{ex}}]$ and $b_1, b_2 \in [b]_0$. We finally set $\mathrm{DP}[v, \mathbf{a}, b] := \mathrm{DP}'[v, z, \mathbf{a}, b]$.

We return `yes` if $\mathrm{DP}[\rho, \mathbf{a}^*, 1] \geq D$ where ρ is the root of the phylogenetic tree $\mathcal{T}$ and a_i^* is the number of teams $t_j = (s_j, e_j)$ with $s_j < i \leq e_j$. Otherwise, return `no`.

Correctness. In vector $\mathbf{a}^*$ the number of available teams per timeslot is stored, such that $\mathcal{I}$ is a `yes`-instance if and only if $\mathrm{DP}[\rho, \mathbf{a}^*, 1] \geq D$. It remains to show that DP and DP' stores the intended value.

If x is a leaf then the subtree $\mathcal{T}_x$ rooted at x does not contain edges and so $\mathrm{PD}_{\mathcal{T}_x}(\{x\}) = \mathrm{PD}_{\mathcal{T}_x}(\emptyset) = 0$. As $S = \{x\}$ is the only non-empty set of taxa with $S \subseteq \mathrm{off}(x)$, in the case that $b = 1$, we need to ensure that in $\mathbf{a}$, enough person-hours are declared for $\{x\}$, which is $\sum_{i=1}^{\text{ex}(x)} a_i \geq \ell(x)$. Thus, the base case is correct.

Let v be a vertex with children $u_1, \ldots, u_z$. To see that $\mathrm{DP}'[v, i, \mathbf{a}, b]$, for $i \in [q]$, stores the correct value, observe that the diversity of edge vu_i can be added if and only if at least one taxon in $\mathrm{off}(u_i)$ survives, which is happening if and only if $b_2 = 1$ (or $b = 1$ in the case of $i = 1$). Further, the available person-hours at v can naturally be divided between the children v. Therefore, DP' stores the correct value. The correctness of $\mathrm{DP}[v, \mathbf{a}, b]$ directly follows from the correctness of $\mathrm{DP}'[v, z, \mathbf{a}, b]$.

Running time. The tables contain $\mathcal{O}(n \cdot (|T|+1)^{\max_{\text{ex}}})$ entries. For a leaf x, the value of $\mathrm{DP}[x, \mathbf{a}, 1]$ can be computed in constant time, in our RAM model. For an internal vertex v with z children, we need to copy the value of $\mathrm{DP}'[v, z, \mathbf{a}, b]$ in constant time.

In Recurrence (4.5), there are $\mathcal{O}((|T|+1)^{\max_{\text{ex}}})$ options to choose $\mathbf{a}'$ and at most 4 options to choose b_1 and b_2, and so the values of the entries can be computed in $\mathcal{O}((|T|+1)^{\max_{\text{ex}}} \cdot n)$ time.

Finally, we need to iterate over the options for $\mathbf{a}^*$. Altogether, we can compute a solution for an instance of Time-PD in $\mathcal{O}((|T|+1)^{2\cdot\max_{\text{ex}}} \cdot n)$ time.

We note that the table entries store values of $\mathcal{O}(D)$ and therefore the running time is also feasible in more restricted RAM models. □

Proof of Proposition 4.15b. Table definition. Let $\mathbf{a} = (a_1, \ldots, a_{\text{var}_{\text{ex}}})$ be a var_{ex}-dimensional vector with values $a_i \in [H_i]_0$.

We define a dynamic programming algorithm with a table DP. In entry $\text{DP}[v, \mathbf{a}, b]$, for a vertex v and an integer $b \in \{0, 1\}$, we store 0 if $b = 0$; if $b = 1$, then we store the maximum value of $\text{PD}_{\mathcal{T}_v}(S)$ for a subset S of offspring of v such that $\sum_{x \in S \cap Z_j} \leq a_j$ for all $j \in [\text{var}_{\text{ex}}]$. We define an auxiliary table DP' in which in entry $\text{DP}[v, i, \mathbf{a}, b]$ we only consider non-empty sets $S \subseteq \text{off}(u_1) \cup \cdots \cup \text{off}(u_i)$ where $u_1, \ldots, u_z$ are the children of v.

Algorithm. As a base case, for each leaf x we store $\text{DP}[x, \mathbf{a}, b] = 0$ if $b = 0$ or $a_i \geq \ell(x)$, for each $i \geq \text{ex}(x)$. Otherwise, we store $\text{DP}[x, \mathbf{a}, 1] = -\infty$.

Let v be an internal vertex with children $u_1, \ldots, u_z$.

We define $\text{DP}'[v, 1, \mathbf{a}, b] := \text{DP}[u_1, \mathbf{a}, b] + \lambda(vu_1) \cdot b$ and we compute further values with the recurrence

$$\text{DP}'[v, i+1, \mathbf{a}, b] = \max_{\mathbf{a}', b_1, b_2} \text{DP}'[v, i, \mathbf{a}', b_1] + \text{DP}[u_{i+1}, \mathbf{a} - \mathbf{a}', b_2] + \lambda(vu_{i+1}) \cdot b_2. \quad (4.6)$$

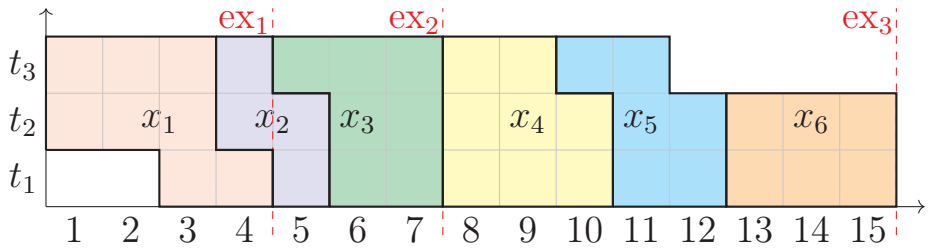
Here, we select $\mathbf{a}'$, b_1, and b_2 such that $a'_j \in [a_j]_0$ for each $j \in [\max_{\text{ex}}]$ and $b_1, b_2 \in [b]_0$. We finally set $\text{DP}[v, \mathbf{a}, b] := \text{DP}'[v, z, \mathbf{a}, b]$.

We return `yes` if $\text{DP}[\rho, \mathbf{a}^*, 1] \geq D$ where ρ is the root of the given phylogenetic tree $\mathcal{T}$ and $a^*_i = H_i$. Otherwise, we return `no`.

Correctness. In the base case, it is enough to check whether $a_i \geq \ell(x)$ for $i \geq \text{ex}(x)$. A visualization is given in Figure 4.4.

This algorithm is similar to the algorithm behind Proposition 4.15a. Instead of remembering the available teams each timeslot, in $\mathbf{a}$, we store the available person-hours per unique remaining time. Therefore, the correctness proof is analogous to the correctness proof of Proposition 4.15a.

Running time. The tables contain $\mathcal{O}(n \cdot (H_{\max_{\text{ex}}})^{\text{var}_{\text{ex}}})$ entries. Each entry in DP can be computed in constant time. In Recurrence (4.6), there are $\mathcal{O}((H_{\max_{\text{ex}}})^{\text{var}_{\text{ex}}})$ options to choose $\mathbf{a}'$ and at most 4 to choose b_1 and b_2, and so the value of each entry can be computed in $\mathcal{O}((H_{\max_{\text{ex}}})^{\text{var}_{\text{ex}}} \cdot n)$ time. Altogether, we can compute a solution for Time-PD in $\mathcal{O}((H_{\max_{\text{ex}}})^{2\,\text{var}_{\text{ex}}} \cdot n)$ time. □



t_i	t_1	t_2	t_3
s_i	2	0	0
e_i	15	15	11

x_i	x_1	x_2	x_3	x_4	x_5	x_6
$\ell(x_i)$	8	4	7	8	6	6
$\mathrm{ex}(x_i)$	4	7	7	15	15	15

$i=$	1	2	3
ex_i	4	7	15
H_i	10	19	49

Figure 4.4: This is a hypothetical schedule of taxa $x_1, \ldots, x_6$. This schedule is only possible, as x_2 is already started before ex_1. Thus, it is not sufficient to assign $x_2, x_3 \in Y_2$ the person-hours between ex_1 and ex_2, which are $H_2 - H_1 = 9$.

Proof of Proposition 4.15c. Table definition. Let $\mathbf{A} = (A_1, \ldots, A_{\max_{\mathrm{ex}}})$ be a $\max_{\mathrm{ex}}$-tuple in which A_i are subsets of T.

We define a dynamic programming algorithm with a table DP. In entry $\mathrm{DP}[v, \mathbf{A}, b]$, for a vertex v and an integer $b \in \{0, 1\}$, we store 0 if $b = 0$; if $b = 1$, then we store the maximum value of $\mathrm{PD}_{\mathcal{T}_v}(S)$ for a subset of taxa $S \subseteq \mathrm{off}(v)$ that can be saved using only teams from A_j at each timeslot $j \in [\max_{\mathrm{ex}}]$. We define an auxiliary table DP', in which, in entry $\mathrm{DP}'[v, i, \mathbf{a}, b]$ we only consider non-empty sets $S \subseteq \mathrm{off}(u_1) \cup \cdots \cup \mathrm{off}(u_i)$ where $u_1, \ldots, u_z$ are the children of v.

Algorithm. As a base case, for each leaf x we store $\mathrm{DP}[x, \mathbf{A}, b] = 0$ if $b = 0$ or if there is a team $t_j \in T$ and an integer $i \in [\max_{\mathrm{ex}} - \ell(x)]_0$ such that $t_j \in A_{i+i'}$ for each $i' \in [\ell(x)]$. Otherwise, we store $\mathrm{DP}[x, \mathbf{A}, 1] = -\infty$.

Let v be an internal vertex with children $u_1, \ldots, u_z$. We compute the entry with the equation $\mathrm{DP}'[v, 1, \mathbf{A}, b] := \mathrm{DP}[u_1, \mathbf{A}, b] + \lambda(vu_1) \cdot b$ and we compute further values with the recurrence

$$\mathrm{DP}'[v, i+1, \mathbf{A}, b] = \max_{\mathbf{A}', b_1, b_2} \mathrm{DP}'[v, i, \mathbf{A}', b_1] + \mathrm{DP}[u_{i+1}, \mathbf{A} - \mathbf{A}', b_2] + \lambda(vu_{i+1}) \cdot b_2. \quad (4.7)$$

Here, we select $\mathbf{A}'$, b_1, and b_2 such that $A'_j \subseteq A_j$ for each $j \in [\max_{\mathrm{ex}}]$ and $b_1, b_2 \in [b]_0$. We finally set $\mathrm{DP}[v, \mathbf{A}, b] := \mathrm{DP}'[v, z, \mathbf{A}, b]$.

We return `yes` if $\mathrm{DP}[\rho, \mathbf{A}^*, 1] \geq D$ where ρ is the root of the given phylogenetic tree $\mathcal{T}$ and A^*_i for $i \in [\max_{\mathrm{ex}}]$ contains team $t_j = (s_j, e_j) \in T$ if and only if $s_j < i \leq e_j$. Otherwise, we return `no`.

Correctness. If DP stores the intended value, $\mathcal{I}$ is a `yes`-instance if and only if $\mathrm{DP}[\rho, \mathbf{A}^*, 1] \geq D$, where ρ is the root of the phylogenetic tree $\mathcal{T}$. It remains to prove that DP stores the intended value.

For a leaf x, if $\mathrm{DP}[x, \mathbf{A}, b] = 0$ then either $b = 0$ or there are $\ell(x)$ consecutive A_is in which a team t_j occurs. Thus, x can be saved and $\mathrm{PD}_{\mathcal{T}_x}(\{x\}) = 0$. Likewise, we show it the other way round and the base case is correct.

The correctness of the recurrence can be shown analogously to the correctness of Proposition 4.15a.

Running time. Observe that for each $i \in [\max_{\mathrm{ex}}]$ there are $2^{|T|}$ options to select A_i. Thus, there are $(2^{|T|})^{\max_{\mathrm{ex}}} = 2^{|T| \cdot \max_{\mathrm{ex}}}$ options for $\mathbf{A}$. Therefore, the tables contain $\mathcal{O}(n \cdot 2^{|T| \cdot \max_{\mathrm{ex}}})$ entries. For a leaf x, we need to iterate over the teams and the timeslots to check whether there is a team contained in $\ell(x)$ consecutive A_is. Thus, we can compute all entries of DP in $\mathcal{O}(2^{|T| \cdot \max_{\mathrm{ex}}} \cdot |T|^{1+\max_{\mathrm{ex}}} \cdot n)$ time.

Observe that in Recurrence (4.7), A'_j is a subset of A_j, so there are $\mathcal{O}(3^{|T| \cdot \max_{\mathrm{ex}}})$ viable options for $\mathbf{A}$ and $\mathbf{A}'$. Thus, we conclude that all entries of DP$'$ can be computed in $\mathcal{O}(3^{|T| \cdot \max_{\mathrm{ex}}} \cdot n)$.

Consequently, because $|T|^{1+\max_{\mathrm{ex}}} \leq 1.5^{|T| \cdot \max_{\mathrm{ex}}} \cdot |T|$ we can compute a solution for an instance of s-Time-PD in $\mathcal{O}(3^{|T| \cdot \max_{\mathrm{ex}}} \cdot |T| \cdot n)$ time. □

4.5.2 Few Rescue Lengths and Remaining Times

The scheduling problem $1||\sum w_j(1-U_j)$ is `FPT` with respect to $\mathrm{var}_\ell + \mathrm{var}_{\mathrm{ex}}$ [HKPS21]. In this subsection, we describe a dynamic programming algorithm, similar to the approach in Theorem 3.1, to prove that Time-PD is at least `XP` when parameterized by $\mathrm{var}_\ell + \mathrm{var}_{\mathrm{ex}}$. Here, var_ℓ and $\mathrm{var}_{\mathrm{ex}}$ are the number of unique needed rescue lengths and unique remaining times, respectively.

Let $\mathcal{I}$ be an instance of Time-PD, let $\ell(X) = \{\ell_1, \ldots, \ell_{\mathrm{var}_\ell}\}$ for each $\ell_i < \ell_{i+1}$ with $i \in [\mathrm{var}_\ell - 1]$ be the unique rescue lengths, and let $\mathrm{ex}(X) = \{\mathrm{ex}_1, \ldots, \mathrm{ex}_{\mathrm{var}_{\mathrm{ex}}}\}$ with $\mathrm{ex}_i < \mathrm{ex}_{i+1}$ for each $i \in [\mathrm{var}_{\mathrm{ex}} - 1]$ be the unique remaining times.

Proposition 4.16. Time-PD *can be solved in* $\mathcal{O}(n^{2\,\mathrm{var}_\ell \cdot \mathrm{var}_{\mathrm{ex}}} \cdot (n + \mathrm{var}_\ell \cdot \mathrm{var}_{\mathrm{ex}}^2))$ *time.*

Proof. Table definition. By $\mathbf{A}$ we denote an integer-matrix of size $\mathrm{var}_\ell \times \mathrm{var}_{\mathrm{ex}}$. Further, we denote $A_{i,j}$ to be the entry in row $i \in [\mathrm{var}_\ell]$ and column $j \in [\mathrm{var}_{\mathrm{ex}}]$ of $\mathbf{A}$. With $\mathbf{A}_{(i,j)+z}$ we denote the matrix resulting from $\mathbf{A}$ in which in row i and column j, the value z is added.

We define a dynamic programming algorithm with table DP. In entry $\mathrm{DP}[v, \mathbf{A}, b]$, we want to store 0 if $b = 0$. If $b = 1$, store in $\mathrm{DP}[v, \mathbf{A}, b]$ the maximum diversity that can be achieved in the subtree rooted at v in which at most $A_{i,j}$ taxa x are

chosen with $\ell(x) = \ell_i$ and $\mathrm{ex}(x) = \mathrm{ex}_j$. We define an auxiliary table DP' in which in entry $\mathrm{DP}[v, \mathbf{A}, b, i]$ we only consider the first i children of v.

Algorithm. For each leaf x with $\ell(x) = \ell_i$ and $\mathrm{ex}(x) = \mathrm{ex}_j$, we store $\mathrm{DP}[x, \mathbf{A}, b] = 0$ if $b = 0$ or $A_{i,j} > 0$. Otherwise, we store $\mathrm{DP}[x, \mathbf{A}, b] = -\infty$.

Let v be an internal vertex with children $u_1, \ldots, u_z$.

We define $\mathrm{DP}'[v, 1, \mathbf{A}, b] := \mathrm{DP}[u_1, \mathbf{A}, b] + \lambda(vu_1) \cdot b$ and we compute further values with the recurrence

$$\mathrm{DP}'[v, i+1, \mathbf{A}, b] = \max_{\mathbf{B}, b_1, b_2} \mathrm{DP}'[v, i, \mathbf{B}, b_1] + \mathrm{DP}[u_{i+1}, \mathbf{A} - \mathbf{B}, b_2] + \lambda(vu_{i+1}) \cdot b_2. \quad (4.8)$$

Here, we select $\mathbf{B}$ such that $B_{i,j} \leq A_{i,j}$ for each $i \leq [\mathrm{var}_\ell]$, $j \leq [\mathrm{var}_{\mathrm{ex}}]$ and b_1, and b_2 are selected to be in $[b]_0$. We finally set $\mathrm{DP}[v, \mathbf{A}, b] := \mathrm{DP}'[v, z, \mathbf{A}, b]$.

We return `yes` if $\mathrm{DP}[\rho, \mathbf{A}] \geq D$ for the root ρ of $\mathcal{T}$ and a matrix $\mathbf{A}$ such that $(\ell_1, \ldots, \ell_{\mathrm{var}_\ell}) \cdot \mathbf{A} \cdot \mathbf{1}_i \leq H_i$ for each $i \in [\mathrm{var}_{\mathrm{ex}}]$. Here, $\mathbf{1}_i$ is a $\mathrm{var}_{\mathrm{ex}}$-dimensional vector in which the first i positions are 1 and the remaining are 0 for each $i \in [\mathrm{var}_i]$.

Correctness. By the definition of DP, we see that an instance is a `yes`-instance of Time-PD if and only if $\mathrm{DP}[\rho, \mathbf{A}] \geq D$ for the root ρ of $\mathcal{T}$ and a matrix $\mathbf{A}$ such that $(\ell_1, \ldots, \ell_{\mathrm{var}_\ell}) \cdot \mathbf{A} \cdot \mathbf{1}_i \leq H_i$ for each $i \in [\mathrm{var}_{\mathrm{ex}}]$. It only remains to show that DP stores the correct value.

Observe that a leaf x can be saved if and only if we are allowed to save a taxon with $\ell(x) = \ell_i$ and $\mathrm{ex}(x) = \mathrm{ex}_j$. Thus $\mathrm{DP}[x, \mathbf{A}, b]$ should store 0 if $b = 0$ or if $A_{i,j} \geq 1$. The other direction follows as well. The correctness of the recurrence can be shown analogously to the correctness of Proposition 4.15a.

Running time. The matrix $\mathbf{A}$ contains $\mathrm{var}_\ell \cdot \mathrm{var}_{\mathrm{ex}}$ entries with integers in $[n]_0$. We can assume that if $A_{i,j} = n$ for some $i \leq [\mathrm{var}_\ell]$, $j \leq [\mathrm{var}_{\mathrm{ex}}]$, then $A_{p,q} = 0$ for each $p \neq i$ and $q \neq j$. Such that there are $n^{\mathrm{var}_{\mathrm{ex}} \cdot \mathrm{var}_{\mathrm{ex}}} + n$ options for a matrix $\mathbf{A}$. Therefore, the tables DP and DP' contain $\mathcal{O}(n \cdot n^{\mathrm{var}_{\mathrm{ex}} \cdot \mathrm{var}_{\mathrm{ex}}})$ entries.

Each entry in DP can be computed in linear time. To compute entries of DP', in Recurrence (4.8) we iterate over possible matrices $\mathbf{B}$ and booleans b_1, b_2. Therefore, we can compute each entry in DP' in $\mathcal{O}(n^{2\,\mathrm{var}_{\mathrm{ex}} \cdot \mathrm{var}_{\mathrm{ex}} + 1})$ time.

For the initialization, we need to iterate over possible matrices $\mathbf{A}$ and compute whether $(\ell_1, \ldots, \ell_{\mathrm{var}_\ell}) \cdot \mathbf{A} \cdot \mathbf{1}_i \leq H_i$ in $\mathcal{O}(n^{\mathrm{var}_{\mathrm{ex}} \cdot \mathrm{var}_{\mathrm{ex}}} \cdot \mathrm{var}_\ell \cdot \mathrm{var}_{\mathrm{ex}}^2)$ time. That proves the overall running time. □

4.5.3 Pseudo-polynomial Running Time on Stars

In this subsection, we show that Time-PD can be solved in pseudo-polynomial time if the input tree is a star. In the light of Proposition 4.5, such a result is unlikely to hold for s-Time-PD.

Proposition 4.17. *If the given input tree is a star,* Time-PD *can be solved*

(a) in $\mathcal{O}((\max_{\text{ex}})^2 \cdot n)$ *time,*

(b) in $\mathcal{O}(D^2 \cdot n)$ *time,*

(c) in $\mathcal{O}(\overline{D}^2 \cdot n)$ *time, or*

(d) in $\mathcal{O}((\max_{\lambda})^2 \cdot n^3)$ *time, where* $\max_{\lambda}$ *is the maximum edge weight.*

Proof. Let $\mathcal{I}$ be an instance of Time-PD in which the given phylogenetic tree is a star. With the help of Knapsack, we solve the problem separately on the different classes Y_i and use a dynamic programming algorithm to combine the solutions. In Knapsack, we are given a set of items N, each with a weight λ_i and a profit p_i, a capacity C, and a desired profit P. The question is whether we can select a set of the items $S \in N$ such that the sum of profits p_i in S is at least P while the sum of weights is not exceeding C.

A solution for an instance of Knapsack can be found in $\mathcal{O}(C \cdot |N|)$ time [Wei66, GRR19] with a dynamic programming algorithm, indexing solutions by the capacity $C' \in [C]$ and the number of items $i \in [|N|]$, and storing the maximum profit for a subset of the first i items that has total weight of at most C'. We note that such an algorithm also computes the maximum profit for every capacity $C' \in [C]$, and so we may assume that in $\mathcal{O}(C \cdot |N|)$ time it is possible to construct a table DP_a such that $DP_a[i, C']$ stores the maximum P such that $(Y_i, \lambda'_j, p_j, C, P)$ is a `yes`-instance of Knapsack.

Along similar lines, there exists a dynamic programming algorithm with a running time of $\mathcal{O}(P \cdot |N|^2)$ [Wei66, GRR19], where entries are indexed by $P' \in [P]$ and $i \in [|N|]$, and we store the minimum capacity C' for which there is a subset of the first i items with a total profit of at least P'. Adapting this algorithm, we receive a running time of $\mathcal{O}(\overline{P} \cdot |N|^2)$, where $\overline{P} = \sum_{a_i \in N} p_i - P$ and $\overline{C} = \sum_{a_i \in N} \lambda_i - C$. In such an algorithm, we index solutions by $\overline{P}' \leq \overline{P}$ and $i \leq N$, and store the *maximum* total weight of a subset of the first i items whose total profit is *at most* $\overline{P}'$. If such a set exists with a weight of at least $\overline{C}$ for $\overline{P}' = \overline{P}$, then the complement set is a solution for the Knapsack instance.

Algorithms. We describe the algorithm with a running time of $\mathcal{O}((\max_{\text{ex}})^2 \cdot n)$ and omit the similar cases for the other variants of the algorithm.

Recall that Y_i is the set of taxa with $\text{ex}(x) = i$. In the following, for each set Y_i in $\mathcal{I}$, we consider an instance of Knapsack with item set Y_i in which $x_j \in Y_i$ with incoming edge e_j has a weight of $\lambda'_j := \ell(x_j)$ and a profit of $p_j := \lambda(e_j)$. We

define a table DP, in which for any $i \in [var_r]$ and $C \in [\mathrm{ex}_i]$, entry $\mathrm{DP}[i, C]$ stores the maximum desired profit P such that $(Y_i, \lambda'_j, p_j, C, P)$ is a `yes`-instance of Knapsack.

We define a table DP' in which we combine the sub-solutions on Z_i, next. As a base case, we store $\mathrm{DP}'[1, C] := \mathrm{DP}[1, C]$ for each $C \in [\mathrm{ex}_1]$, $P \in [D]$, and $\overline{P} \in [\overline{D}]$. To compute further values, we can use the recurrence

$$\mathrm{DP}'[i+1, C] = \max_{C' \in [C]_0} \mathrm{DP}'[i, C'] + \mathrm{DP}[i+1, C - C'] \tag{4.9}$$

Finally, we return `yes` if $\mathrm{DP}'[\mathrm{var}_{\mathrm{ex}}, \mathrm{max}_{\mathrm{ex}}] \geq D$.

Correctness. For convenience, we will write $\lambda(S)$ to denote $\sum_{x \in S} \lambda(\rho x)$ for a set S of taxa. Observe that $\lambda(S) = \mathrm{PD}_{\mathcal{T}}(S)$, as $\mathcal{T}$ is a star. The correctness of the values in DP follows from the correctness of the Knapsack-algorithms. Assume that for some $i \in [\mathrm{var}_{\mathrm{ex}}]$ the correct value is stored in $\mathrm{DP}'[i, C]$ for each $C \in [D]$.

Let $\mathrm{DP}'[i+1, C]$ store $a \geq 0$. Then, by the construction there is an integer C' such that $a = a_1 + a_2 = \mathrm{DP}'[i, C'] + \mathrm{DP}[i+1, C - C']$. Consequently, there are sets of taxa $S_1 \subseteq Z_i$ and $S_2 \subseteq Y_{i+1}$ such that $\ell(S_1) + \ell(S_2) = C' + (C - C') = C$ and $\lambda(S_1) + \lambda(S_2) = a_1 + a_2 = a$. Therefore, $S := S_1 \cup S_2$ is a set with $\ell(S) = C$ and $\mathrm{DP}'[i+1, C] = \lambda(S)$.

Conversely, let $S \subseteq Z_{i+1}$ be a set of taxa with $\ell(S) = C$. Define $S_1 := S \cap Z_i$ and $S_2 := S \cap Y_{i+1}$ and let C' be $\ell(S_1)$. We conclude that

$$\lambda(S) = \lambda(S_1) + \lambda(S_2) \geq \mathrm{DP}'[i, C'] + \mathrm{DP}[i+1, C - C'].$$

Running time. Note that the algorithm solving the Knapsack-instance is also a dynamic programming algorithm, such that all the values of the table DP are computed in $\mathcal{O}(\mathrm{max}_{\mathrm{ex}} \cdot n)$ time.

The table DP' has $\mathrm{var}_{\mathrm{ex}} \cdot \mathrm{max}_{\mathrm{ex}}$ entries. To compute a value with the recurrence, we need to check the $\mathcal{O}(\mathrm{max}_{\mathrm{ex}})$ options to select C and add two numbers of size at most D, and so we can compute all values of DP' in $\mathcal{O}((\mathrm{max}_{\mathrm{ex}})^2 \cdot \mathrm{var}_{\mathrm{ex}})$ time.

For the other running times: We observe that Knapsack can be computed in $\mathcal{O}(D^2 \cdot n)$, and in $\mathcal{O}(\overline{D}^2 \cdot n)$ time. The rest follows analogously.

We can assume that $D \leq n \cdot \mathrm{max}_\lambda$, as we are dealing with a trivial `no`-instance otherwise. Therefore, in $\mathcal{O}((n \cdot \mathrm{max}_\lambda)^2 \cdot n) = \mathcal{O}((\mathrm{max}_\lambda)^2 \cdot n^3)$ time we can solve this special case of Time-PD. □

4.6 Discussion

With Time-PD and s-Time-PD, we introduced two `NP`-hard generalizations of Max-PD in which taxa may have distinct extinction times. Most relevantly, we

have proven that both problems are FPT when parameterized by the target diversity D and that Time-PD is also FPT when parameterized by the acceptable loss of phylogenetic diversity $\overline{D}$. We have further proven that both problems are FPT when parameterized by the available person-hours.

Time-PD is NP-hard, but it remains an open question whether Time-PD is solvable in pseudo-polynomial time. Indeed, we do not know if Time-PD or s-Time-PD can be solved in polynomial time even when the maximum rescue length needed to save a taxon is 2. We note that the scheduling problem $1||\sum w_j(1-U_j)$ is W[1]-hard when parameterized by the unique number of processing times [HH24]; this implies that even on stars s-Time-PD is W[1]-hard when parameterized by the unique number of rescue lengths.

We further ask whether the $\mathcal{O}(2^n)$ running time for Time-PD (Proposition 4.6) can be improved to $\mathcal{O}(2^{o(n)})$, or whether this bound can be shown to be tight under SETH or ETH. It also remains an open question whether Time-PD or s-Time-PD are FPT with respect to the largest extinction time $\max_{\text{ex}}$.

We have not regarded kernelization algorithms. An interesting open question therefore is whether Time-PD or Time-PD admit a kernelization of polynomial size with respect to D or $\overline{D}$.

Chapter 5

Phylogenetic Diversity with Ecological Dependencies

5.1 Introduction

As we have seen in the previous chapters, the inherently limited amount of resources that one may devote to the task of saving taxa, necessitates decisions on which conservation strategies to pursue. To support such decisions, one needs to incorporate quantitative information on the possible impact and the success likelihood of conservation strategies. In this context, one task is to compute an optimal conservation strategy in the light of this information.

To find a conservation strategy with the best positive impact, one would ideally aim to maximize the functional diversity of the surviving taxa (species). However, measuring this diversity even is impossible in many scenarios [MPC+18].

Luckily, Max-PD can be computed in polynomial time [Ste05, PG05] and therefore became the standard in measuring biodiversity [Cro97]. Computing an optimal conservation strategy becomes much more difficult, however, when the success likelihood of a strategy is included in the model. One way to achieve this is to add concrete survival probabilities for protected taxa, leading in its most general form to the NP-hard Generalized Noah's Ark Problem [HS06, KS23b], which has been observed in Chapter 3. This problem formulation, however, still has a central drawback: It ignores that the survival of some taxa may also depend on the survival of other taxa. This aspect was first considered by Moulton et al. [MSS07] in the Optimizing PD with Dependencies (PDD) problem.

Dependencies of taxa can take any thinkable form. One model of dependencies are so called *food-webs* in which the relationship between predators and prey are

presented. Food-webs are especially relevant in the observation of an ecosystem because, in them, one easily sees the role of a taxon within the outside world and the overall flow of biomass [CDRG+18, Lin42].

Moulton et al. [MSS07] showed that PDD can be solved by the greedy algorithm if the objective of maximizing phylogenetic diversity agrees with the viability constraint in a precise technical sense. Later, PDD was conjectured to be NP-hard in [SNM08]. This conjecture was confirmed by Faller et al. [FSW11], who showed that PDD is NP-hard even if the food-web $\mathcal{F}$ is a tree. Further, Faller et al. [FSW11] considered S-PDD, the special case where the phylogenetic tree is restricted to be a star, and showed that S-PDD is NP-hard even for food-webs which have a bipartite graph as underlying graph. Finally, polynomial-time algorithms were provided for very restricted special cases, for example for PDD when the food-web is a *directed* tree [FSW11].

Our contribution. Because PDD has been shown to be NP-hard already on very restricted instances [FSW11], we turn to parameterized complexity in order to overcome this intractability. Here, we consider the most natural parameters related to the solution, such as the solution size k and the threshold of diversity D, and parameters that describe the structure of the input food-web $\mathcal{F}$. We formally consider the decision problem, where we ask for the existence of a viable solution with diversity at least D, but our algorithms actually solve the optimization problem as well.

Our most important results are as follows. In Theorem 5.2, we prove that PDD is FPT when parameterized with the solution size k plus the height of the phylogenetic tree $\mathcal{T}$. This also implies that PDD is FPT with respect to D, the diversity threshold. However, both problems, PDD and S-PDD, are unlikely to admit a kernel of polynomial size when parameterized by D. We also consider the dual parameter $\overline{D}$, that is, the amount of diversity that is lost from $\mathcal{T}$, and show that PDD is W[1]-hard with respect to $\overline{D}$.

We then consider the structure of the food-web. In particular, we consider the special case that each connected component of the food-web $\mathcal{F}$ is a complete digraph—so called cluster graphs. As we will show, this case is structurally equivalent to the case that each connected component of $\mathcal{F}$ is a star with one source vertex. Thus, this case describes a particularly simple dependency structure, where taxa are either completely independent or have a common source. We further show that PDD is NP-hard in this special case while S-PDD admits an FPT-algorithm when parameterized by the vertex deletion distance to cluster graphs. Our results thus yield structured classes of food-webs where the complexity of S-PDD and PDD strongly differ. Finally, we show that S-PDD is FPT with respect to the treewidth of the food-web and therefore

Table 5.1: An overview over the parameterized complexity results for PDD and s-PDD. "D.t. τ" stands for "Distance to τ"—the number of vertices that need to be removed to obtain graph class τ.

Parameter	s-PDD		PDD	
Budget k	FPT	Thm. 5.1	XP	Obs. 5.9
Diversity D	FPT	Thm. 5.3	FPT	Thm. 5.3
	no poly kernel	Thm. 5.4	no poly kernel	Thm. 5.5
Species-loss $\overline{k}$	W[1]-hard, XP	Prop. 5.19, Obs. 5.9	W[1]-hard, XP	Prop. 5.19, Obs. 5.9
Diversity-loss $\overline{D}$	W[1]-hard, XP	Prop. 5.19, Obs. 5.9	W[1]-hard, XP	Prop. 5.19, Obs. 5.9
D.t. Cluster	FPT	Thm. 5.6	NP-h for 0	Thm. 5.7
D.t. Co-Cluster	FPT	Thm. 5.8	FPT	Thm. 5.8
Treewidth	FPT	Thm. 5.9	NP-h for 1	[FSW11]
Max Leaf #	FPT	Thm. 5.9	NP-h for 2	Cor. 5.29
D.t. Dom.-Source	NP-h for 1	Prop. 5.30	NP-h for 1	Prop. 5.30
D.t. Bipartite	NP-h for 0	[FSW11]	NP-h for 0	[FSW11]
Max Degree	NP-h for 3	[FSW11]	NP-h for 3	[FSW11]

can be solved in polynomial time if the food-web is a tree (Theorem 5.9). Our result disproves a conjecture of Faller et al. [FSW11, Conjecture 4.2] stating that s-PDD is NP-hard even when the food-web is a tree. Again, this result shows that s-PDD can be substantially easier than PDD on some structured classes of food-webs.

Table 5.1 gives an overview over the results in this chapter. Here, Dom.-Source stands for Dominant-Source; a graph class for DAGs $\mathcal{F}$ in which there is a single source σ and an edge σx in $\mathcal{F}$ for each $x \in V(\mathcal{F}) \setminus \{\sigma\}$. Figure 5.2 gives results for structural parameters and sets these parameters into relation.

Structure of the Chapter. In Section 5.2, we formally define Optimizing PD with Dependencies and prove some simple initial results. In Section 5.3 and Section 5.4, we consider parameterization by the budget k and the threshold of diversity D, the two integers of the input. In Section 5.5, we consider PDD with respect to the number of taxa that go extinct and the acceptable loss of diversity. In Section 5.6, we consider parameterization by structural parameters of the food-web. Finally, in Section 5.7, we discuss future research ideas.

5.2 Preliminaries

In this section, we present the formal definition of the problems, and the parameterization. We further start with some preliminary observations.

5.2.1 Definitions

Food-Webs. For a given set of taxa X, a *food-web* $\mathcal{F} = (X, E)$ *on* X is a directed acyclic graph. If xy is an edge of E then x is *prey* of y and y is a *predator* of x. The set of prey and predators of x is denoted with $N_<(x)$ and $N_>(x)$, respectively. A taxon x with an empty set of prey is a *source* and sources($\mathcal{F}$) denotes the set of sources in the food-web $\mathcal{F}$.

For a given taxon $x \in X$, we define $X_{\leq x}$ to be the set of taxa X which can reach x in $\mathcal{F}$. Analogously, $X_{\geq x}$ is the set of taxa that can be reached from x in $\mathcal{F}$.

For a given food-web $\mathcal{F}$ and a set $Z \subseteq X$ of taxa, a set of taxa $A \subseteq Z$ is *Z-viable* if sources($\mathcal{F}[A]$) $\subseteq$ sources($\mathcal{F}[Z]$). A set of taxa $A \subseteq X$ is *viable* if A is X-viable. In other words, a set $A \subseteq Z \subseteq X$ is Z-viable or viable if each vertex with an in-degree of 0 in $\mathcal{F}[A]$ also has in-degree 0 in $\mathcal{F}[Z]$ or in $\mathcal{F}$, respectively.

Problem Definitions and Parameterizations. Formally, the main problem we regard in this chapter is defined as follows.

Optimizing PD with Dependencies (PDD)

Input: A phylogenetic X-tree $\mathcal{T}$, a food-web $\mathcal{F}$ on X, and integers k and D.
Question: Is there a viable set $S \subseteq X$ such that $|S| \leq k$, and $\mathrm{PD}_{\mathcal{T}}(S) \geq D$?

Additionally, in Optimizing PD in Vertex-Weighted Food-Webs (s-PDD) we consider the special case of PDD in which the phylogenetic X-tree $\mathcal{T}$ is a star.

Throughout this chapter, we adopt the common convention that n is the number of taxa $|X|$ and we let m denote the number of edges in the food-web $|E(\mathcal{F})|$. Observe that $\mathcal{T}$ has $\mathcal{O}(n)$ edges. Such a relation does not necessarily hold for m.

For an instance $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D)$ of PDD, we define the parameter $\overline{D}$ to be $\mathrm{PD}_{\mathcal{T}}(X) - D = \sum_{e \in E} \lambda(e) - D$. Informally, $\overline{D}$ is the acceptable loss of diversity: If we save a set of taxa $A \subseteq X$ with $\mathrm{PD}_{\mathcal{T}}(A) \geq D$, then the total amount of diversity we lose from $\mathcal{T}$ is at most $\overline{D}$. Similarly, we define $\overline{k} := |X| - k$. That is, $\overline{k}$ is the minimum number of species that need to become extinct.

5.2.2 Preliminary Observations

We present some observations and reduction rules which we use throughout this chapter.

Observation 5.1. *Let $\mathcal{F}$ be a food-web. A set $A \subseteq X$ is viable if and only if there are edges $E_A \subseteq E(\mathcal{F})$ such that every connected component in the graph (A, E_A) is a tree with the root in* sources($\mathcal{F}$).

Proof. If A is viable then sources($\mathcal{F}[A]$) is a subset of sources($\mathcal{F}$). It follows that for each taxon $x \in A$, either x is a source in $\mathcal{F}$ or A contains a prey y of x.

Conversely, if a graph (A, E_A) exists in which all connected components are trees, then explicitly the sources of $\mathcal{F}[A]$ are a subset of sources($\mathcal{F}$). □

Observation 5.2. *Let $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D)$ be a* `yes`*-instance of* PDD*. Then, unless $k > n$, a viable set $S \subseteq X$ with $PD_{\mathcal{T}}(S) \geq D$ exists which has a size of exactly k.*

Proof. Let S be a solution for $\mathcal{I}$. If S has a size of k, nothing remains to show. Otherwise, observe that $S \cup \{x\}$ is viable and $\mathrm{PD}_{\mathcal{T}}(S \cup \{x\}) \geq \mathrm{PD}_{\mathcal{T}}(S)$ for each taxon $x \in (N_{>}(S) \cup \text{sources}(\mathcal{F})) \setminus S$. Because $(N_{>}(S) \cup \text{sources}(\mathcal{F})) \setminus S$ is non-empty, unless $S = X$, we conclude that $S \cup \{x\}$ is a solution and iteratively, there is a solution of size k. □

Observation 5.3. *Let $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D)$ be an instance of* PDD*. In $\mathcal{O}(|\mathcal{I}|^2)$ time, an equivalent instance $\mathcal{I}' = (\mathcal{T}', \mathcal{F}', k', D')$ of* PDD *with $D' \in \mathcal{O}(D)$ and only one source in $\mathcal{F}'$, can be computed.*

Proof. Construction. Let $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D)$ be an instance of PDD. Add a new taxon $\star$ to $\mathcal{F}$ and add edges from $\star$ to each taxon x of sources($\mathcal{F}$) to obtain $\mathcal{F}'$. To obtain $\mathcal{T}'$, add $\star$ as a child to the root ρ of $\mathcal{T}$ and set $\lambda'(\rho\star) = D+1$ and $\lambda'(e) = \lambda(e)$ for each $e \in E(\mathcal{T})$. Finally, set $k' := k + 1$ and $D' := 2 \cdot D + 1$.

Correctness. All steps can be performed in $\mathcal{O}(|\mathcal{I}|^2)$ time. Because $S \subseteq X$ is a solution for $\mathcal{I}$ if and only if $S \cup \{\star\}$ is a solution for $\mathcal{I}'$, the instance $\mathcal{I}' = (\mathcal{T}', \mathcal{F}', k+1, 2 \cdot D+1)$ is a `yes`-instance of PDD if and only if $\mathcal{I}$ is a `yes`-instance of PDD. □

Reduction Rule 5.4. *Let $R \subseteq X$ be the set of taxa which have a distance of at least k to every source. Then, set $\mathcal{F}' := \mathcal{F} - R$ and $\mathcal{T}' := \mathcal{T} - R$.*

Lemma 5.5. *Reduction Rule 5.4 is correct and in $\mathcal{O}(n + m)$ time can be applied exhaustively.*

Proof. By definition, each viable set of taxa which has a size of k is disjoint from R. Therefore, the set R is disjoint from every solution. With a breadth-first search, the set R can be found in $\mathcal{O}(n+m)$ time . This is also the total running time, since one application of the rule is exhaustive. □

Reduction Rule 5.6. *Apply Reduction Rule 5.4 exhaustively. If $\max_{\lambda} \geq D$ return* `yes`*.*

After Reduction Rule 5.4 has been applied exhaustively, for any taxon $x \in X$ there is a viable set S_x of size at most k with $x \in S_x$. If edge e has a weight of at least D, then for each taxon x which is an offspring of e, the set S_x is viable, has a size of at most k, and $\mathrm{PD}_{\mathcal{T}}(S_x) \geq \mathrm{PD}_{\mathcal{T}}(\{x\}) \geq D$. So, S_x is a solution.

Reduction Rule 5.7. *Given an instance $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D)$ of* PDD *with edges $vw, uw \in E(\mathcal{F})$ for taxa v, w and each $u \in N_<(v)$. If v is not a source, then remove vw from $E(\mathcal{F})$.*

Lemma 5.8. *Definition 5.7 is correct and can be applied exhaustively in $\mathcal{O}(n^3)$ time.*

Proof. Correctness. If $\mathcal{I}'$ is a `yes`-instance, then so is $\mathcal{I}$.

Conversely, let $\mathcal{I}$ be a `yes`-instance of PDD with solution S. If $v \notin S$, then S is also a solution for instance $\mathcal{I}'$. If $v \in S$ then because S is viable in $\mathcal{F}$, some vertex u of $N_<(v)$ is in S. Consequently, S is also viable in $\mathcal{F} - \{vw\}$, as w still could be fed by u (if $w \in S$).

Running time. For two taxa v and w, we can check $N_<(v) \subseteq N_<(w)$ in $\mathcal{O}(n)$ time. Consequently, an exhaustive application of Reduction Rule 5.7 takes $\mathcal{O}(n^3)$ time. □

5.3 The Solution Size k

In this section, we consider parameterization by the size of the solution k. First, we observe that PDD is `XP` when parameterized by k and $\overline{k}$. Recall that $\overline{k} := n - k$ is the minimum number of taxa which need to go extinct. In Section 5.3.1, we show that s-PDD is `FPT` with respect to k. We generalize this result in Section 5.3.2 by showing that PDD is `FPT` when parameterized by $k + \mathrm{height}_{\mathcal{T}}$.

Observation 5.9. PDD *can be solved in $\mathcal{O}(n^{k+2})$ and $\mathcal{O}(n^{\overline{k}+2})$ time.*

Proof. Algorithm. Iterate over the sets $S \subseteq X$ of size k. Return `yes` if there is a viable set S with $\mathrm{PD}_{\mathcal{T}}(S) \geq D$. Return `no` if there is no such set.

Correctness and Running time. The correctness of the algorithm follows from Observation 5.2. Checking whether a set S is viable and has diversity of at least D can be done $\mathcal{O}(n^2)$ time. The claim follows because there are $\binom{n}{k} = \binom{n}{n-k} = \binom{n}{\overline{k}}$ subsets of X of size k. □

5.3.1 s-PDD With k

We now show that s-PDD is FPT when parameterized by the size of the solution k.

Theorem 5.1. s-PDD *can be solved in* $\mathcal{O}(2^{3.03k+o(k)} \cdot nm \cdot \log n)$ *time.*

In order to prove this theorem, we color the taxa and require that a solution should contain at most one taxon of each color. Formally, the auxiliary problem which we consider is defined as follows. In k-colored Optimizing PD in Vertex-Weighted Food-Webs (k-c-s-PDD), alongside the usual input $(\mathcal{T}, \mathcal{F}, k, D)$ of s-PDD, we are given a vertex-coloring $c : X \to [k]$ which assigns each taxon a *color* $c(x) \in [k]$. We ask for whether there is a viable set $S \subseteq X$ of taxa such that $\mathrm{PD}_{\mathcal{T}}(S) \geq D$ and $c(S)$ is *colorful*. A set $c(S)$ is colorful if c is injective on S. Observe that each colorful set S satisfies $|S| \leq k$. We continue to show how to solve k-c-s-PDD before we apply tools of the color coding toolbox to extend this result to the uncolored version.

Lemma 5.10. k-c-s-PDD *can be solved in* $\mathcal{O}(3^k \cdot n \cdot m)$ *time.*

Proof. Table definition. Let $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D, c)$ be an instance of k-c-s-PDD, and by Observation 5.3 we assume that $\star \in X$ is the only source in $\mathcal{F}$.

Given $x \in X$, a set of colors $C \subseteq [k]$, and a set of taxa $X' \subseteq X$: A set $S \subseteq X'$ is (C, X')-*feasible* if

- $c(S) = C$,
- $c(S)$ is colorful, and
- S is X'-viable.

We define a dynamic programming algorithm with tables DP and DP′. For a taxon $x \in X$ and a set of colors $C \subseteq [k]$, we want entry $\mathrm{DP}[x, C]$ to store the maximum $\mathrm{PD}_{\mathcal{T}}(S)$ of $(C, X_{\geq x})$-feasible sets S. Recall that $X_{\geq x}$ is the set of taxa which x can reach in $\mathcal{F}$. If no $(C, X_{\geq x})$-feasible set $S \subseteq X'$ exists, we want $\mathrm{DP}[x, C]$ to store $-\infty$. In other words, in $\mathrm{DP}[x, C]$ we store the biggest phylogenetic diversity of a set S which is $X_{\geq x}$-viable and c bijectively maps S to C.

For a taxon x, let $y_1, \ldots, y_q$ be an arbitrary but fixed order of $N_>(x)$. In the auxiliary table DP′, we want entry $\mathrm{DP}'[x, p, C]$ for $p \in [q]$, and $C \subseteq [k]$ to store the maximum $\mathrm{PD}_{\mathcal{T}}(S)$ of (C, X')-feasible sets $S \subseteq X'$, where $X' = \{x\} \cup X_{\geq y_1} \cup \cdots \cup X_{\geq y_p}$. If no (C, X')-feasible set $S \subseteq X'$ exists, we want $\mathrm{DP}'[x, p, C]$ to store $-\infty$.

Algorithm. As a base case, for each $x \in X$ and $p \in [|N_>(x)|]$ let entries $\mathrm{DP}[x, \emptyset]$ and $\mathrm{DP}[x, p, \emptyset]$ store 0 and let entry $\mathrm{DP}[x, C]$ store $-\infty$ if C is non-empty and $c(x)$

does not occur in C. For each $x \in X$ with $N_>(x) = \emptyset$, we store $\lambda(\rho x)$ in $\mathrm{DP}[x, \{c(x)\}]$. Recall that ρx is an edge because $\mathcal{T}$ is a star.

Fix $x \in X$. For every $Z \subseteq C \setminus \{c(x)\}$, we set $\mathrm{DP}'[x, 1, \{c(x)\} \cup Z] := \mathrm{DP}[y_1, Z]$. To compute further values, once $\mathrm{DP}'[x, q, Z]$ for each $q \in [p]$, and every $Z \subseteq C$ is computed, for $Z \subseteq C \setminus \{c(x)\}$ we use the recurrence

$$\mathrm{DP}'[x, p+1, \{c(x)\} \cup Z] := \max_{Z' \subseteq Z} \mathrm{DP}'[x, p, \{c(x)\} \cup Z \setminus Z'] + \mathrm{DP}[y_{p+1}, Z']. \quad (5.1)$$

Finally, we set $\mathrm{DP}[x, C] := \mathrm{DP}'[x, q, C]$ for every $C \subseteq [k]$.

We return `yes` if $\mathrm{DP}[\star, C]$ stores 1 for some $C \subseteq [k]$. Otherwise, we return `no`.

Correctness. The base cases are correct.

The tables are computed first for taxa further away from the source and with increasing size of C. Assume that for a fixed taxon x with predators $y_1, \ldots, y_q$ and a fixed $p \in [q]$, the entries $\mathrm{DP}[x', Z]$ and $\mathrm{DP}'[x, p', Z]$, for each $x' \in N_>(x)$, for each $p' \in [p]$, and every $Z \subseteq [k]$, store the desired value. Fix a set $C \subseteq [k]$ with $c(x) \in C$. We show that if $\mathrm{DP}'[x, p+1, C]$ stores d then there is a (C, X')-feasible set $S \subseteq X' \cup X_{\geq y_{p+1}}$ for $X' := \{x\} \cup X_{\geq y_1} \cup \cdots \cup X_{\geq y_p}$ with $\mathrm{PD}_{\mathcal{T}}(S) = d$. Afterward, we show that if $S \subseteq X' \cup X_{\geq y_{p+1}}$ with $\mathrm{PD}_{\mathcal{T}}(S) = d$ is a (C, X')-feasible set then $\mathrm{DP}'[x, p+1, C]$ stores at least d.

If $\mathrm{DP}'[x, p+1, C] = d > 0$, then, by Recurrence (5.1), a set $Z \subseteq C \setminus \{c(x)\}$ exists such that $\mathrm{DP}'[x, p, C \setminus Z] = d_x$ and $\mathrm{DP}[y_{p+1}, Z] = d_y$ with $d = d_x + d_y$. Therefore, there is a $(C \setminus Z, X')$-feasible set $S_x \subseteq X'$ with $\mathrm{PD}_{\mathcal{T}}(S_x) = d_x$ and a $(Z, X_{\geq y_{p+1}})$-feasible set $S_y \subseteq X_{\geq y_{p+1}}$ with $\mathrm{PD}_{\mathcal{T}}(S_y) = d_y$. Define $S := S_x \cup S_y$ and observe that $\mathrm{PD}_{\mathcal{T}}(S) = d$, because $\mathcal{T}$ is a star, and $c(S_x)$ and $c(S_y)$ are disjoint and therefore S_x and S_y. It remains to show that S is a $(C, X' \cup X_{\geq y_{p+1}})$-feasible set. First, observe that because $C \setminus Z$ and Z are disjoint, we conclude that $c(S)$ is colorful. Then, $c(S) = c(S_x) \cup c(S_y) = C \setminus Z \cup Z = C$ where the first equation is satisfied because $c(S)$ is colorful. The taxa x and y_{p+1} are the only sources in $\mathcal{F}[X_{\geq x}]$ and $\mathcal{F}[X_{\geq y_{p+1}}]$, respectively. Therefore, x is in S_x and y_{p+1} is in S_y, unless S_y is empty. If $S_y = \emptyset$ then $S = S_x$ and S is $X' \cup X_{\geq y_{p+1}}$-viable because S is X'-viable. Otherwise, if S_y is non-empty then because S_y is $X_{\geq y_{p+1}}$-viable, we conclude $\mathrm{sources}(\mathcal{F}[S_y]) = \{y_{p+1}\}$. As $x \in S$ and $y_{p+1} \in N_>(x)$, we conclude $\mathrm{sources}(\mathcal{F}[S]) = \{x\}$ and so S is $X' \cup X_{\geq y_{p+1}}$-viable. Therefore, S is a $(C, X' \cup X_{\geq y_{p+1}})$-feasible set.

Conversely, let $S \subseteq X' \cup X_{\geq y_{p+1}}$ be a non-empty $(C, X' \cup X_{\geq y_{p+1}})$-feasible set with $\mathrm{PD}_{\mathcal{T}}(S) = d$. Observe that X' and $X_{\geq y_{p+1}}$ are not necessarily disjoint. We define S_y to be the set of taxa of $X_{\geq y_{p+1}}$ which are connected to y_{p+1} in $\mathcal{F}[X_{\geq y_{p+1}}]$. Further, define $Z := c(S_y)$ and define $S_x := S \setminus S_y$. As $c(S)$ is colorful, especially $c(S_x)$ and $c(S_y)$ are colorful. Thus, S_y is a $(Z, X_{\geq y_{p+1}})$-feasible time. Further, we conclude

that $c(S_x) = C \setminus c(S_y) = C \setminus Z$. As sources$(\mathcal{F}[S])$ = sources$(\mathcal{F}[X' \cup X_{\geq y_{p+1}}]) = \{x\}$, we conclude $x \in S$. Because $\mathcal{F}$ is acyclic and y_{i+1} is a predator of x, we conclude x is not in $X_{\geq y_{p+1}}$ and so x is in S_x. Each vertex of S which can reach y_{p+1} in $\mathcal{F}[S]$ is in $F_{\geq y_{p+1}}$ and therefore in S_y. Consequently, because S is $X' \cup X_{\geq y_{p+1}}$-viable we conclude sources$(\mathcal{F}[S_x]) = \{x\}$. Thus, S_x is $(C \setminus Z, X')$-feasible. Therefore, $\mathrm{DP}[y_{p+1}, Z] = \mathrm{PD}_{\mathcal{T}}(S_x)$ and $\mathrm{DP}'[x, p, C \setminus Z] = \mathrm{PD}_{\mathcal{T}}(S_y)$. Hence, $\mathrm{DP}'[x, p+1, C]$ stores at least $\mathrm{PD}_{\mathcal{T}}(S)$.

Running time. The base cases can be checked in $\mathcal{O}(k)$ time. As each $c \in [k]$ in Recurrence (5.1) can either be in Z', in $\{c(x)\} \cup Z \setminus Z'$ or in $[k] \setminus (\{c(x)\} \cup Z)$, all entries of the tables can be computed in $\mathcal{O}(3^k \cdot n \cdot m)$ time.

We note that the table entries store values of $\mathcal{O}(D)$ and therefore the running time is also feasible in more restricted RAM models. □

For the following proof we construct a perfect hash family $\mathcal{H}$ which is defined in Definition 2.14, Central for proving Theorem 5.1 is to define and solve an instance of k-c-s-PDD for each function in $\mathcal{H}$.

Proof of Theorem 5.1. Reduction. Let $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D)$ be an instance of PDD. We assume that $\mathcal{F}$ only has one source, by Observation 5.3.

Let $x_1, \ldots, x_n$ be an order of the taxa. Compute an (n, k)-perfect hash family $\mathcal{H}$. For every $f \in \mathcal{H}$, let c_f be a coloring such that $c_f(x_j) = f(j)$ for each $x_j \in X$.

For every $f \in \mathcal{H}$, construct an instance $\mathcal{I}_f = (\mathcal{T}, \mathcal{F}, k, D, c_f)$ of k-c-s-PDD and solve $\mathcal{I}_f$ using Lemma 5.10 and return `yes` if and only if $\mathcal{I}_f$ is a `yes`-instance for some $f \in \mathcal{H}$.

Correctness. We show that if $\mathcal{I}$ has a solution S then there is an $f \in \mathcal{H}$ such that $\mathcal{I}_f$ is a `yes`-instance of k-c-s-PDD. Let S be a solution for $\mathcal{I}$. Thus, S is viable, $\mathrm{PD}_{\mathcal{T}}(S) \geq D$, and S has a size of at most k. We may assume $|S| = k$ by Observation 5.2. By the definition of (n, k)-perfect hash families, there exists a function $f \in \mathcal{H}$ such that $c_f(S)$ is colorful. So, S is a solution for $\mathcal{I}_f$.

Conversely, a solution of $\mathcal{I}_f$ for any $f \in \mathcal{H}$ is a solution for $\mathcal{I}$.

Running Time. The instances $\mathcal{I}_f$ can be constructed in $e^k k^{\mathcal{O}(\log k)} \cdot n \log n$ time. An instance of k-c-s-PDD can be solved in $\mathcal{O}(3^k \cdot n \cdot m)$ time, and the number of instances is $|\mathcal{C}| = e^k k^{\mathcal{O}(\log k)} \cdot \log n$. Thus, the total running time is $\mathcal{O}^*(e^k k^{\mathcal{O}(\log k)} \log n \cdot (3^k \cdot nm))$ which simplifies to $\mathcal{O}((3e)^k \cdot 2^{\mathcal{O}(\log^2(k))} \cdot nm \cdot \log n)$. □

5.3.2 PDD With $k + \mathrm{height}_{\mathcal{T}}$

In this subsection, we generalize the result of the previous subsection by showing that PDD is `FPT` when parameterized with the size of the solution k plus $\mathrm{height}_{\mathcal{T}}$, the

height of the phylogenetic tree. This algorithm uses the techniques of color coding, data reduction by reduction rules, and the enumeration of trees.

Theorem 5.2. PDD *can be solved in* $\mathcal{O}^*(K^K \cdot 2^{3.03K+o(K)})$ *time. Herein, we write* $K := k \cdot \text{height}_{\mathcal{T}}$.

We define *pattern-trees* $\mathcal{T}_P = (V_P, E_P, c_P)$ to be a tree (V_P, E_P) with a vertex-coloring $c_P : V_P \to [k \cdot \text{height}_{\mathcal{T}}]$. Recall that $\mathcal{T}\langle Y\rangle$ is the spanning tree of the vertices in Y. To show the result of Theorem 5.3.2, we use a subroutine for solving the following problem.

In Optimizing PD with Pattern-Dependencies (PDD-pattern), we are given alongside the usual input $(\mathcal{T}, \mathcal{F}, k, D)$ of PDD a pattern-tree $\mathcal{T}_P = (V_P, E_P, c_P)$, and a vertex-coloring $c : V(\mathcal{T}) \to [k \cdot \text{height}_{\mathcal{T}}]$. We ask whether there is a viable set $S \subseteq X$ of taxa such that S has a size of at most k, $c(\mathcal{T}\langle S \cup \{\rho\}\rangle)$ is colorful, and $\mathcal{T}\langle S \cup \{\rho\}\rangle$ and $\mathcal{T}_P$ are *color-equal*. That is, there is an edge uv of $\mathcal{T}\langle S \cup \{\rho\}\rangle$ with $c(u) = c_u$ and $c(v) = c_v$ if and only if there is an edge $u'v'$ of $\mathcal{T}_P$ with $c(u') = c_u$ and $c(v') = c_v$. Informally, given a pattern-tree, we want that it matches the colors of the spanning tree induced by the root and a solution.

Next, we present reduction rules with which we can reduce the phylogenetic tree in an instance of PDD-pattern to be a star which subsequently can be solved with Theorem 5.1. Afterward, we show how to apply this knowledge to compute a solution for PDD.

Reduction Rule 5.11. *Let* uv *be an edge of* $\mathcal{T}$. *If there is no edge* $u'v' \in E_P$ *with* $c_P(u') = c(u)$ *and* $c_P(v') = c(v)$, *then set* $\mathcal{T}' := \mathcal{T} - \text{desc}(v)$ *and* $\mathcal{F}' := \mathcal{F} - \text{off}(v)$.

Lemma 5.12. *Reduction Rule 5.11 is correct and can be applied exhaustively in* $\mathcal{O}(n^3)$ *time.*

Proof. Correctness. Assume that $S \subseteq X$ is a solution of the instance of PDD-pattern. As there is no edge $u'v' \in E_P$ with $c_P(u') = c(u)$ and $c_P(v') = c(v)$ we conclude that $S \cap \text{desc}(v) = \emptyset$ and so the reduction rule is safe.

Running Time. To check whether Reduction Rule 5.11 can be applied, we need to iterate over both $E(\mathcal{T})$ and E_P. Therefore, a single application can be executed in $\mathcal{O}(n^2)$ time. In each application of Reduction Rule 5.11 we remove at least one vertex so that an exhaustive application can be computed in $\mathcal{O}(n^3)$ time. □

Reduction Rule 5.13. *Let* $u'v'$ *be an edge of* $\mathcal{T}_P$. *For each vertex* $u \in V(\mathcal{T})$ *with* $c(u) = c_P(u')$ *such that* u *has no child* v *with* $c(v) = c_P(v')$, *set* $\mathcal{T}' := \mathcal{T} - \text{desc}(v)$ *and* $\mathcal{F}' := \mathcal{F} - \text{off}(v)$.

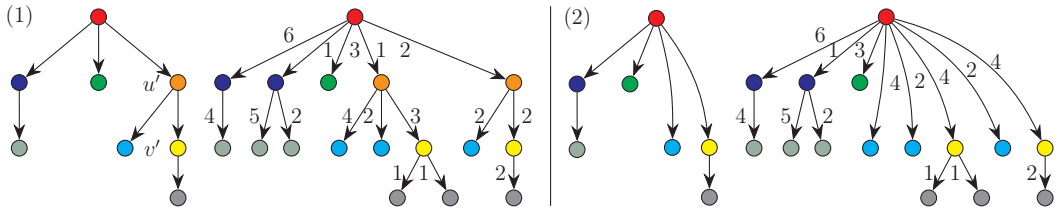


Figure 5.1: An example for Reduction Rule 5.15. (1) An instance of PDD-PATTERN. (2) The instance of (1) after an application of Reduction Rule 5.15 to the marked vertices. In both instances, the pattern-tree is on the left and the phylogenetic tree is on the right.

Lemma 5.14. *Reduction Rule 5.13 is correct and can be applied exhaustively in $\mathcal{O}(n^3)$ time.*

Proof. Correctness. Let S be a solution for an instance of PDD-PATTERN. The spanning tree $\mathcal{T}\langle S \cup \{\rho\}\rangle$ contains exactly one vertex w of color $c(u)$. As $c(w) = c_P(u')$ we conclude that w has a child w' and $c(w') = c(v')$. Consequently, $S \cap \mathrm{desc}(u) = \emptyset$ and $w \neq u$.

Running Time. Like in Reduction Rule 5.11, iterate over the edges of $\mathcal{T}$ and $\mathcal{T}_P$. Each application either removes at least one vertex or concludes that the reduction rule is applied exhaustively. □

Observe that the application of the previous two reduction rules may create leaves in the phylogenetic tree which are not taxa. We can safely remove these from the tree. Now, let us come to our final reduction rule.

Reduction Rule 5.15. *Apply Reduction Rules 5.11 and 5.13 exhaustively.*

Let ρ be the root of $\mathcal{T}$ and let ρ_P be the root of $\mathcal{T}_P$. Let v' be a grand-child of ρ_P and let u' be the parent of v'.

(a) For each vertex u of $\mathcal{T}$ with $c(u) = c_P(u')$, add edges ρv to $\mathcal{T}$ for every child v of u.

(b) Set the weight of ρv to $\lambda(uv)$ if $c(v) \neq c_P(v')$, or $\lambda(uv) + \lambda(\rho u)$ if $c(v) = c_P(v')$.

(c) Add edges $\rho_P w'$ to $\mathcal{T}_P$ for every child w' of u'.

(d) Set $\mathcal{T}'_P := \mathcal{T}_P - u'$ and $\mathcal{T}' := \mathcal{T} - u$.

Figure 5.1 depicts an application of Reduction Rule 5.15.

Lemma 5.16. *Reduction Rule 5.15 is correct and can be applied exhaustively in $\mathcal{O}(n^3)$ time.*

Proof. Correctness. Assume that $\mathcal{I}$ is a `yes`-instance of PDD-PATTERN with solution S. Because $\mathcal{T}\langle S \cup \{\rho\}\rangle$ and $\mathcal{T}_P$ are color-equal also $\mathcal{T}'\langle S \cup \{\rho\}\rangle$ and $\mathcal{T}'_P$ are color-equal. Let u^* and w_1 be the unique vertices in $\mathcal{T}\langle S \cup \{\rho\}\rangle$ with $c(u^*) = c_P(u')$ and $c(w_1) = c_P(v')$. Let $w_2, \ldots, w_\ell$ be the other children of u^*. Because $\mathrm{PD}_{\mathcal{T}'}(S)$ is the sum of the weights of the edges of $\mathcal{T}'\langle S \cup \{\rho\}\rangle$, we conclude

$$\mathrm{PD}_{\mathcal{T}'}(S) = \mathrm{PD}_{\mathcal{T}}(S) - (\lambda(\rho u^*) + \sum_{i=1}^{\ell} \lambda(u^* w_i)) + \sum_{i=1}^{\ell} \lambda'(\rho w_i).$$

Since $\lambda'(\rho w_1) = \lambda(\rho u^*) + \lambda(u^* w_1)$ and $\lambda'(\rho w_i) = \lambda(u^* w_i)$ for $i \in [\ell] \setminus \{1\}$, we conclude that $\mathrm{PD}_{\mathcal{T}'}(S) = \mathrm{PD}_{\mathcal{T}}(S) \geq D$. Therefore, S is a solution for $\mathcal{I}'$.

The other direction is shown analogously.

Running Time. For a given grand-child v' of ρ_P, one needs to perform $\mathcal{O}(n)$ color-checks and add $\mathcal{O}(n)$ edges to the tree. As the reduction rule can be applied at most $|\mathcal{T}_P| \in \mathcal{O}(K) \in \mathcal{O}(n)$ times, an exhaustive application takes $\mathcal{O}(n^2)$ time. So, the predominant factor in the running time is the exhaustive application of the other reduction rules. □

With these reduction rules, we can reduce the phylogenetic tree of a given instance of PDD-PATTERN to only be a star and then solve PDD-PATTERN by applying Theorem 5.1.

Lemma 5.17. PDD-PATTERN *can be solved in $\mathcal{O}(3^k \cdot n \cdot m + n^3)$ time.*

Proof. Algorithm. Let an instance $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D, \mathcal{T}_P = (V_P, E_P, c_P), c)$ of PDD-PATTERN be given. If there is a vertex $v \in V_P$ and $c_P(v) \notin c(V(\mathcal{T}))$, then return `no`. If $c(\rho) \neq c_P(\rho_P)$ where ρ and ρ_P are the roots of $\mathcal{T}$ and $\mathcal{T}_P$ respectively, return `no`.

Apply Reduction Rule 5.15 exhaustively. Then, both $\mathcal{T}_P$ and $\mathcal{T}$ are stars. Return `yes` if and only if $(\mathcal{T}', \mathcal{F}', k, D, c)$ is a `yes`-instance of k-C-S-PDD.

Correctness. If $\mathcal{T}_P$ contains a vertex v with $c_P(v) \notin c(V(\mathcal{T}))$, or if $c(\rho) \neq c_P(\rho_P)$, then $\mathcal{I}$ is a `no`-instance. Then, the correctness follows by Lemma 5.16 and Lemma 5.10.

Running time. Reduction Rule 5.15 can be applied exhaustively in $\mathcal{O}(n^3)$ time. With the application of Lemma 5.10, the overall running time is $\mathcal{O}(3^k \cdot n \cdot m + n^3)$. □

Now, we have everything to prove Theorem 5.2. We reduce from PDD to PDD-PATTERN and apply Lemma 5.17. For this, we use the fact that there are n^{n-2} labeled

directed trees on n vertices [Sho95] which can be enumerated in $\mathcal{O}(n^{n-2})$ time [BH80]. To solve an instance $\mathcal{I}$ of PDD, we check each of these trees as a pattern-tree for a given coloring of the phylogenetic tree. These colorings are defined with a perfect hash family as defined in Definition 2.14. Recall that $K = k \cdot \text{height}_{\mathcal{T}}$.

Proof of Theorem 5.2. Algorithm. Let $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D)$ be an instance of PDD. Let the vertices of $\mathcal{T}$ be ordered as $v_1, \ldots, v_{|V(\mathcal{T})|}$. Iterate over $i \in [\min\{K, |V(\mathcal{T})|\}]$. Compute a $(|V(\mathcal{T})|, i)$-perfect hash family $\mathcal{H}_i$. Compute the set $\mathcal{P}_i$ of labeled directed trees with i vertices.

For every $\mathcal{T}_P = (V_P, E_P, c_P) \in \mathcal{P}_i$, proceed as follows. Assume that the labels of $\mathcal{T}_P$ are in $[i]$. For every $f \in \mathcal{H}_i$, let c_f be a coloring such that $c_f(v_j) = f(j)$ for each $v_j \in V(\mathcal{T})$.

For every $f \in \mathcal{H}_i$, solve instance $\mathcal{I}_{\mathcal{T}_P,f} := (\mathcal{T}, \mathcal{F}, k, D, \mathcal{T}_P, c_f)$ of PDD-PATTERN using Lemma 5.17. Return `yes` if and only if $\mathcal{I}_{\mathcal{T}_P,f}$ is a `yes`-instance for some $f \in \mathcal{H}_i$ and some $\mathcal{T}_P \in \mathcal{P}_i$.

Correctness. Any solution of an instance $\mathcal{I}_{\mathcal{T}_P,f}$ of PDD-PATTERN is a solution for $\mathcal{I}$.

Conversely, we show that if S is a solution for $\mathcal{I}$, then there are $\mathcal{T}_P$ and f such that $\mathcal{I}_{\mathcal{T}_P,f}$ is a `yes`-instance of PDD-PATTERN. So let S be a viable set of taxa with $|S| \leq k$ and $\text{PD}_{\mathcal{T}}(S) \geq D$. Let $V^* \subseteq V(\mathcal{T})$ be the set of vertices v that have offspring in S. It follows that $|V^*| \leq \text{height}_{\mathcal{T}} \cdot |S| \leq K$. Then, there is a hash function $f \in \mathcal{H}_{V^*}$ mapping V^* bijectively to $[|V^*|]$. Consequently, $\mathcal{P}_{|V^*|}$ contains a tree $\mathcal{T}_P$ which is isomorphic to $\mathcal{T}[V^*]$ with labels c_f. Hence, $\mathcal{I}_{\mathcal{T}_P,f}$ is a `yes`-instance of PDD-PATTERN.

Running Time. For a fixed $i \in [K]$, the set $\mathcal{H}_i$ contains $e^i i^{\mathcal{O}(\log i)} \cdot \log n$ hash functions and the set $\mathcal{P}_i$ contains $\mathcal{O}(i^{i-2})$ labeled trees. In $\mathcal{O}(i^{i-2} \cdot n \log n)$ time, both of the sets can be computed. Given a hash function and a tree, each instance $\mathcal{I}_{\mathcal{T}_P,f}$ of PDD-PATTERN is constructed in $\mathcal{O}(n)$ time and can be solved in $\mathcal{O}(3^k \cdot n^3)$ time. Thus, the overall running time is $\mathcal{O}(K \cdot e^K K^{K-2+\mathcal{O}(\log K)} \cdot 3^k \cdot n^3 \log n)$, which summarizes to $\mathcal{O}(K^K \cdot 2^{1.44K+1.58k+o(K)} \cdot n^3 \log n)$. □

5.4 The Diversity D

In this section, we consider parameterization with the required threshold of diversity D. As the edge-weights are integers, we conclude that we can return `yes` if $k \geq D$ or if the height of the phylogenetic tree $\mathcal{T}$ is at least D, after Reduction Rule 5.4 has been applied exhaustively. Otherwise, $k + \text{height}_{\mathcal{T}} \in \mathcal{O}(D)$ and thus the FPT-algorithm for $k + \text{height}_{\mathcal{T}}$ (Theorem 5.2) directly gives an FPT-algorithm for PDD in that case.

In Section 5.4.1, we present a faster FPT-algorithm for the parameter D. Afterward, we show that it is unlikely that PDD admits a polynomial kernel for D, even in very restricted cases.

5.4.1 FPT-Algorithm For D

By Theorem 5.2, PDD is FPT with respect to the desired diversity D. Here, we present another algorithm with a faster running time. To obtain this algorithm, we implicitly subdivide edges of the phylogenetic tree according to their edge weights. We then use color coding on the vertices of the subdivided tree.

Theorem 5.3. PDD *can be solved in* $\mathcal{O}(2^{3.03(2D+k)+o(D)} \cdot nm + n^2)$ *time.*

We define D-COLORED OPTIMIZING PD WITH DEPENDENCIES (D-C-PDD), a colored version of PDD, as follows. In addition to the usual input of PDD, we receive two colorings c and $\hat{c}$ which assign each edge $e \in E(\mathcal{T})$ a subset $c(e)$ of $[D]$, called *a set of colors*, which is of size $\lambda(e)$ and each taxon $x \in X$ a *color* $\hat{c}(x) \in [k]$. We extend the function c to also assign color sets to taxa $x \in X$ by defining $c(x) := \bigcup_{e \in E'} c(e)$ where E' is the set of edges with $x \in \text{off}(e)$. In D-C-PDD, we ask whether there is a viable set $S \subseteq X$ of taxa such that $c(S) = [D]$, and $\hat{c}$ is colorful.

In the following we show how to solve D-C-PDD and then we show how to apply standard color coding techniques to reduce from PDD to D-C-PDD.

Finding a solution for D-C-PDD can be done with techniques similar to the ones used to prove Lemma 5.10.

Lemma 5.18. D-C-PDD *can be solved in* $\mathcal{O}(3^{D+k} \cdot n \cdot m)$ *time.*

Proof. Table definition. Let $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D, c, \hat{c})$ be an instance of D-C-PDD and we assume that $\star \in X$ is the only source in $\mathcal{F}$ by Observation 5.3.

Given $x \in X$, sets of colors $C_1 \subseteq [D]$, $C_2 \subseteq [k]$, and a set of taxa $X' \subseteq X$, a set $S \subseteq X' \subseteq X$ is (C_1, C_2, X')-*feasible* if

- C_1 is a subset of $c(S)$,
- $\hat{c}(S) = C_2$,
- $\hat{c}(S)$ is colorful, and
- S is X'-viable.

We define a dynamic programming algorithm with tables DP and DP$'$. We want entry DP$[x, C_1, C_2]$, for $x \in X$ and sets of colors $C_1 \subseteq [D]$, $C_2 \subseteq [k]$, to store 1 if a $(C_1, C_2, X_{\geq x})$-feasible set S exists. Otherwise, we want DP$[x, C_1, C_2]$ to store 0.

For a taxon $x \in X$, let $y_1, \ldots, y_q$ be an arbitrary but fixed order of $N_>(x)$. In the auxiliary table we want entry $\mathrm{DP}'[x, p, C_1, C_2]$, for $p \in [q]$, $C_1 \subseteq [D]$, and $C_2 \subseteq [k]$, to store 1 if a (C_1, C_2, X')-feasible set $S \subseteq X'$ exists, where $X' := \{x\} \cup X_{\leq y_1} \cup \cdots \cup X_{\leq y_p}$. If no (C_1, C_2, X')-feasible set $S \subseteq X'$ exists, we want $\mathrm{DP}'[x, p, C_1, C_2]$ to store 0.

Algorithm. As a base case, for each $x \in X$ and each $p \in [|N_>(x)|]$ let $\mathrm{DP}[x, \emptyset, \emptyset]$ and $\mathrm{DP}'[x, p, \emptyset, \emptyset]$ store 1. Further, let $\mathrm{DP}[x, C_1, C_2]$ and $\mathrm{DP}'[x, p, C_1, C_2]$ store 0 if $C_1 \not\subseteq c(x)$, or $\hat{c}(x) \notin C_2$, or $|C_2| > |X_{\geq x}|$ for each $x \in X$, every $C_1 \subseteq [D]$, and every $C_2 \subseteq [k]$. For each taxon $x \in X$ with no predators, we store 1 in $\mathrm{DP}[x, C_1, C_2]$ if $C_1 = C_2 = \emptyset$ or if $C_1 \subseteq c(x)$ and $C_2 = \{\hat{c}(x)\}$. Otherwise, we store 0.

Fix a taxon $x \in X$. Assume that $\mathrm{DP}[y, C_1, C_2]$ is computed for each $y \in N_>(x)$, every $C_1 \subseteq [D]$, and every $C_2 \subseteq [k]$. For $C_1 \subseteq [D] \setminus c(x)$ and $C_2 \subseteq [k] \setminus \{\hat{c}(x)\}$, we set

$$\mathrm{DP}'[x, 1, c(x) \cup C_1, \{\hat{c}(x)\} \cup C_2] := \mathrm{DP}[y_1, C_1, C_2]. \tag{5.2}$$

Fix an integer $p \in [q]$. We assume that $\mathrm{DP}'[x, p, C_1, C_2]$ is computed for every $C_1 \subseteq [D]$, and every $C_2 \subseteq [k]$. Then, for $C_1 \subseteq [D] \setminus c(x)$ and $C_2 \subseteq [k] \setminus \{\hat{c}(x)\}$ we use the recurrence

$$\begin{aligned} & \mathrm{DP}'[x, p+1, c(x) \cup C_1, \{\hat{c}(x)\} \cup C_2] \\ := & \max_{C_1' \subseteq C_1, C_2' \subseteq C_2} \mathrm{DP}'[x, p, c(x) \cup C_1 \setminus C_1', \{\hat{c}(x)\} \cup C_2 \setminus C_2'] \cdot \mathrm{DP}[y_{p+1}, C_1', C_2']. \end{aligned} \tag{5.3}$$

Finally, set $\mathrm{DP}[x, C_1, C_2] := \mathrm{DP}'[x, q, C_1, C_2]$ for every $C_1 \subseteq [D]$, and every $C_2 \subseteq [k]$.

Return `yes` if $\mathrm{DP}[\star, [D], C_2]$ stores 1 for some $C_2 \subseteq [k]$. Otherwise, return `no`.

Correctness. The correctness can be shown analogously to the correctness of Lemma 5.10.

Running time. The tables DP and DP′ contain $\mathcal{O}(2^{D+k} \cdot n \cdot m)$ entries. Each entry in the base case can be computed in $\mathcal{O}(D^2 \cdot n)$ time. In Recurrence (5.3), each color can occur either in C_1', in $c(x) \cup C_1 \setminus C_1'$ or not in $c(x) \cup C_1$. Likewise, with $\{\hat{c}(x)\} \cup Z$ and Z'. Therefore, all values in Recurrence (5.3) can be computed in $\mathcal{O}(3^{D+k} \cdot n \cdot m)$ time which is thus also the overall running time. □

We can now prove Theorem 5.3 by showing how the standard PDD can be reduced to D-c-PDD via color coding. This proof is analogous to the proof of Theorem 5.1.

Proof of Theorem 5.3. Reduction. Let $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D)$ be an instance of PDD. We assume that $\mathcal{F}$ only has one source by Observation 5.3 and that $\max_\lambda < D$ by Reduction Rule 5.6.

Let $e_1, \ldots, e_{|E(\mathcal{T})|}$ and $x_1, \ldots, x_n$ be an order of the edges and taxa of $\mathcal{T}$, respectively. We define integers $W_0 := 0$ and $W_j := \sum_{i=1}^{j} \lambda(e_i)$ for each $j \in [|E(\mathcal{T})|]$. Set $W := W_{|E(\mathcal{T})|}$.

Compute a (W, D)-perfect hash family $\mathcal{H}_D$ and an (n, k)-perfect hash family $\mathcal{H}_k$. For every $g \in \mathcal{H}_k$, let $c_{g,2}$ be a coloring such that $c_{g,2}(x_j) = g(j)$ for each $x_j \in X$. For every $f \in \mathcal{H}_D$, let $c_{f,1}$ be a coloring such that $c_{f,1}(e_j) = \{f(W_{j-1} + 1), \ldots, f(W_j)\}$ for each $e_j \in E(\mathcal{T})$.

For hash functions $f \in \mathcal{H}_D$ and $g \in \mathcal{H}_k$, construct an instance $\mathcal{I}_{f,g}$ of D-c-PDD with $\mathcal{I}_{f,g} := (\mathcal{T}, \mathcal{F}, k, D, c_{f,1}, c_{g,2})$. Solve every instance $\mathcal{I}_{f,g}$ using Lemma 5.18 and return `yes` if and only if $\mathcal{I}_{f,g}$ is a `yes`-instance for some $f \in \mathcal{H}_D$, and some $g \in \mathcal{H}_k$.

Correctness. We first show that if $\mathcal{I}$ is a `yes` instance then $\mathcal{I}_{f,g}$ is a `yes`-instance for some $f \in \mathcal{H}_D$, $g \in \mathcal{H}_k$. For any set of edges E' with $\lambda(E') \geq D$, there is a corresponding subset of $[W]$ of size at least D. Since $\mathcal{H}_D$ is a (W, D)-perfect hash family, $c_{f,1}(E') = [D]$, for some $f \in \mathcal{H}_D$. Analogously, for each set X' of taxa of size k there is a hash function $g \in \mathcal{H}_k$ such that $c_{g,2}(X') = [k]$. Thus in particular, if $S \subseteq X$ is a solution of size k for instance $\mathcal{I}$, then $c_{f,1}(S) = [D]$, for some $f \in \mathcal{H}_D$ and $c_{g,2}(S) = [k]$, for some $g \in \mathcal{H}_k$. It follows that one of the constructed instances of D-c-PDD is a `yes`-instance.

Conversely, a solution for $\mathcal{I}_{f,g}$ for some $f \in \mathcal{H}_D$, and some $g \in \mathcal{H}_k$ is also a solution for $\mathcal{I}$.

Running Time. We apply Reduction Rule 5.6 and Observation 5.3 in $\mathcal{O}(n^2)$ time. By Observation 5.3, $k' \leq k + 1$ and $D' \leq 2D + 1$. We can construct $\mathcal{H}_D$ and $\mathcal{H}_k$ in $e^{D'} D'^{\mathcal{O}(\log D')} \cdot W \log W$ time. By Lemma 5.18, instances of D-c-PDD can be solved in $\mathcal{O}(3^{D'+k'} \cdot nm)$ time each, and the number of instances is $|\mathcal{H}_{D'}| \cdot |\mathcal{H}_{k'}| = e^{D'} D'^{\mathcal{O}(\log D')} \cdot \log W \cdot e^{k'} k'^{\mathcal{O}(\log k')} \cdot \log n \in e^{D'+k'+o(D)} \cdot \log W$.

Thus, the total running time is $\mathcal{O}(e^{2D+k+o(D)} \cdot \log W \cdot (W + 3^{2D+k} \cdot nm))$. This simplifies to $\mathcal{O}((3e)^{2D+k+o(D)} \cdot nm + n^2)$, as $W = \mathrm{PD}_{\mathcal{T}}(X) < 2n \cdot D$. □

5.4.2 No Poly Kernels For D

In this subsection, first, we give a reduction from Set Cover to s-PDD to show that s-PDD does not admit a polynomial kernelization algorithm when parameterized by D, assuming `NP` $\not\subseteq$ `coNP/poly` (Theorem 5.4). This result also holds for PDD. However, we afterward provide a compression from Graph Motif to PDD to show the following stronger result. Even if $\mathcal{F}$, is a directed forest, PDD does not admit a polynomial kernelization algorithm when parameterized by D, assuming `NP` $\not\subseteq$ `coNP/poly` (Theorem 5.5).

In the following, we reduce from Set Cover to s-PDD. In Set Cover, an input consists of a universe $\mathcal{U}$, a family $\mathcal{Q}$ of subsets of $\mathcal{U}$, and an integer k. It is asked whether there exists a sub-family of sets $\mathcal{Q}' \subseteq \mathcal{Q}$ such that $\mathcal{Q}'$ has a carnality of at most k and the union of $\mathcal{Q}'$ covers the entire universe. Assuming $\texttt{NP} \not\subseteq \texttt{coNP/poly}$, Set Cover does not admit a polynomial kernel when parameterized by the size of the universe $|\mathcal{U}|$ [DLS14, CFK+15].

Our reduction from Set Cover to s-PDD is similar to the reduction from Vertex Cover to s-PDD presented in [FSW11].

Theorem 5.4. s-PDD *does not admit a polynomial kernelization algorithm with respect to D, assuming* $\texttt{NP} \not\subseteq \texttt{coNP/poly}$.

Proof. Reduction. Let an instance $\mathcal{I} = (\mathcal{Q}, \mathcal{U}, k)$ of Set Cover be given. We define an instance $\mathcal{I}' = (\mathcal{T}, \mathcal{F}, k', D)$ of s-PDD as follows. Let $\mathcal{T}$ be a star with root ρ and leaves $X := \mathcal{Q} \cup \mathcal{U}$. We set $\lambda(\rho Q) = 1$ for each $Q \in \mathcal{Q}$ and $\lambda(\rho u) = 2$ for each $u \in \mathcal{U}$. Further, the food-web $\mathcal{F}$ is a graph with vertices X and we add an edge Qu to $\mathcal{F}$ if and only if $u \in Q$ for $u \in \mathcal{U}$ and $Q \in \mathcal{Q}$. Finally, we set $k' := k + |\mathcal{U}|$ and $D := k + 2|\mathcal{U}|$.

Correctness. We may assume that $k \leq |\mathcal{U}|$. Therefore, D is bounded in $|\mathcal{U}|$.

Now, assume that $\mathcal{Q}'$ is a solution for $\mathcal{I}$. If necessary, add further sets to $\mathcal{Q}'$ until $|\mathcal{Q}'| = k$. Because $\mathcal{Q}'$ is a solution for $\mathcal{I}$, for each $u \in \mathcal{U}$ there is a $Q \in \mathcal{Q}'$ such that $u \in Q$. Hence, $S := \mathcal{Q}' \cup \mathcal{U}$ is viable, has a size of $|S| = k + |\mathcal{U}| = k'$ and $\mathrm{PD}_{\mathcal{T}}(S) = |\mathcal{Q}'| + 2|\mathcal{U}| = k + 2|\mathcal{U}|$.

Conversely, let S be a solution for instance $\mathcal{I}'$ and assume $|S| = k'$. Let $S_{\mathcal{Q}}$ be the intersection of S with $\mathcal{Q}$. Define $a := |S_{\mathcal{Q}}|$ and $b := |S| - a$. Then, we have $\mathrm{PD}_{\mathcal{T}}(S) = a + 2b = |S| + b$ and $\mathrm{PD}_{\mathcal{T}}(S) \geq k + 2|\mathcal{U}| = k' + |\mathcal{U}|$. We conclude that $b \geq |\mathcal{U}|$ and so $a = |S| - b \leq k' - |\mathcal{U}| = k$. Since S is viable, for each $u \in \mathcal{U}$ there is a $Q \in S_{\mathcal{Q}}$ with $u \in Q$. Therefore, $S_{\mathcal{Q}}$ is a solution for $\mathcal{I}$. □

For PDD we want to show the non-existence of a polynomial kernel even in the case that the food-web is restricted to a forest. Here, we define a cross-composition from Graph Motif, a problem in which one is given a graph G with vertex-coloring χ and a multiset of colors M. It is asked whether G has a connected set of vertices whose multiset of colors equals M. Graph Motif was shown to be NP-hard even on trees [LFS06]. It remains NP-hard to compute a solution for an instance of Graph Motif on trees of maximum vertex-degree three, even if each item appears at most once in M, and there is a color $c^* \in M$ that only one vertex of G takes [FFHV11].

Theorem 5.5. *Even if the given food-web $\mathcal{F}$ is a directed forest,* PDD *does not admit a polynomial kernelization algorithm with respect to D, assuming* NP $\not\subseteq$ coNP/poly.

Proof. We describe a cross-composition from Graph Motif to PDD. We refer readers unfamiliar to cross-compositions to [CFK+15, Chapter 15.1].

Cross-Composition. Fix a set of colors M. Let $\mathcal{I}_1, \ldots, \mathcal{I}_q$ be instances of Graph Motif with $\mathcal{I}_i = (G_i = (V_i, E_i), M)$ such that all G_i are trees with maximum vertex-degree of three. Let v_i be the only vertex of V_i with $\chi(v_i) = c^* \in M$.

We define an instance $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D)$ of PDD as follows. Let $\mathcal{T}$ be a tree with vertex set $\{\rho\} \cup M \cup X$ and leaves $X := \bigcup_{i=1}^{q} V_i$. Add an edge ρc to $\mathcal{T}$ for each $c \in M$ and add an edge cv if and only if $\chi(v) = c$. Each edge has a weight of 1. Orient each edge in G_i away from v_i to obtain H_i for each $i \in [q]$. Let $\mathcal{F}$ be the food-web which is the union of $H_1, \ldots, H_q$. We define $k := |M|$ and $D := 2 \cdot |M|$.

Correctness. We show that some instance of $\mathcal{I}_1, \ldots, \mathcal{I}_q$ is a yes-instance of Graph Motif if and only if $\mathcal{I}$ is a yes-instance of PDD.

Let $\mathcal{I}_i$ be a yes-instance of Graph Motif for an $i \in [q]$. Consequently, there is a set of vertices $S \subseteq V_i$ such that $|S| = |M|$, $\chi(S) = M$ and $G_i[S]$ is connected. We conclude that $v_i \in S$ because v_i is the only vertex with $\chi(v_i) = c^*$. Thus, $H_i[S]$ is a connected subtree of H_i which contains v_i, the only source in H_i. We conclude that S is viable in $\mathcal{I}$. By definition, $|S| = |M|$ and because each vertex in S has another color we conclude that $\mathrm{PD}_{\mathcal{T}}(S) = 2 \cdot |M|$. Hence, S is a solution for $\mathcal{I}$.

Conversely, let $\mathcal{I}$ be a yes-instance of PDD. Consequently, there is a viable set S of taxa such that $|S| \leq |M|$ and $\mathrm{PD}_{\mathcal{T}}(S) \geq 2 \cdot |M|$. We say that a taxon $x \in X$ has color $c \in M$ if cx is an edge in $\mathcal{T}$. Observe that $\mathrm{PD}_{\mathcal{T}}(A \cup \{x\}) \leq \mathrm{PD}_{\mathcal{T}}(A) + 2$ for any set of taxa A. Further, $\mathrm{PD}_{\mathcal{T}}(A \cup \{x\}) = \mathrm{PD}_{\mathcal{T}}(A) + 2$ if and only if x has a color that none of the taxa of A has. We conclude that the taxa in S have unique colors. Fix $j \in [q]$ such that $v_j \in S$. The index j is uniquely defined because there is a unique taxon in S with the color c^*. Because the vertices $v_1, \ldots, v_q$ are the only sources in $\mathcal{F}$ and S is viable, we conclude $u \in V_i$ can not occur in S for $i \neq j$. Therefore, we conclude $S \subseteq V_j$. Because S is viable, $H_i[S]$ is connected and therefore also $G_i[S]$. Thus, S is a solution for instance $\mathcal{I}_j$ of Graph Motif. □

5.5 The Loss of Species $\overline{k}$ and Diversity $\overline{D}$

5.5.1 Hardness For the Acceptable Diversity Loss $\overline{D}$

In some instances, the diversity threshold D may be very large. Then, however, the acceptable loss of diversity $\overline{D}$ would be relatively small. Recall, $\overline{D}$ is defined

as $\mathrm{PD}_{\mathcal{T}}(X) - D$. Encouraged by this observation, recently, several problems in maximizing phylogenetic diversity have been studied with respect to the acceptable diversity loss [JS23b, JS24]. In this section, we show that, unfortunately, s-PDD is W[1]-hard with respect to $\overline{D}$ even if edge-weights are at most two.

To show this result, we reduce from Red-Blue Non-Blocker. In Red-Blue Non-Blocker, the input is an undirected bipartite graph G with vertex bipartition $V = V_r \cup V_b$ and an integer k. The question is whether there is a set $S \subseteq V_r$ of size at least k such that the neighborhood of $V_r \setminus S$ is V_b. Red-Blue Non-Blocker is W[1]-hard when parameterized by the size of the solution k [DF95b].

Proposition 5.19. s-PDD *is* W[1]*-hard with respect to* $\overline{D}$*, even if* $\max_\lambda = 2$.

Proof. Reduction. Let $\mathcal{I} := (G = (V = V_r \cup V_b, E), k)$ be an instance of Red-Blue Non-Blocker. We construct an instance $\mathcal{I}' = (\mathcal{T}, \mathcal{F}, k', D)$ of s-PDD as follows. Let $\mathcal{T}$ be a star with root $\rho \notin V$ and leaves V. In $\mathcal{T}$, an edge $e = \rho u$ has a weight of 1 if $u \in V_r$ and otherwise $\lambda(e) = 2$, if $u \in V_b$. Define a food-web $\mathcal{F}$ with vertices V and for each edge $\{u, v\} \in E$, and every pair of vertices $u \in V_b$, $v \in V_r$, add an edge uv to $\mathcal{F}$. Finally, set $k' := |V| - k$ and $D := 2 \cdot |V_b| + |V_r| - k$, or equivalently $\overline{k} = \overline{D} = k$.

Correctness. The reduction can be computed in polynomial time. We show that if $\mathcal{I}$ is a yes-instance of Red-Blue Non-Blocker then $\mathcal{I}'$ is a yes-instance of PDD. Afterward, we show the converse.

Assume that $\mathcal{I}$ is a yes-instance of Red-Blue Non-Blocker. Therefore, there is a set $S \subseteq V_r$ of size at least k such that $N_G(V_r \setminus S) = V_b$. We assume $|S| = k$ as $N_G(V_r \setminus S) = V_b$ still holds if we shrink S. We define $S' := V \setminus S$ and show that S' is a solution for $\mathcal{I}'$. The size of S' is $|V \setminus S| = |V| - |S| = k'$. Further, $\mathrm{PD}_{\mathcal{T}}(S) = 2 \cdot |V_b| + |V_r \setminus S| = 2 \cdot |V_b| + |V_r| - k = D$. By definition, the vertices in V_r are sources. Further, because S is a solution for $\mathcal{I}$, each vertex of V_b has a neighbor in $V_r \setminus S$. So, S' is viable and $\mathcal{I}'$ is a yes-instance of s-PDD.

Conversely, let $S' \subseteq V$ be a solution for instance $\mathcal{I}'$ of s-PDD. Without loss of generality, S' contains r vertices from V_r and b vertices of V_b. Consequently, $|V| - k \geq |S'| = b + r$ and $2 \cdot |V_b| + |V_r| - k = D \leq \mathrm{PD}_{\mathcal{T}}(S') = 2b + r$. Therefore, $r \leq |V| - k - b$ and so $2b \geq 2 \cdot |V_b| + |V_r| - k - r \geq 2 \cdot |V_b| + |V_r| - k - (|V| - k - b) = |V_b| + b$. We conclude $b = |V_b|$ and $V_b \subseteq S'$. Further, $r = |V_r| - k$. We define $S := V_r \setminus S'$ and conclude $|S| = |V_r| - r = k$. Because S' is viable, each vertex in V_b has a neighbor in $S' \setminus V_b$. Therefore, S is a solution for the yes-instance $\mathcal{I}$ of Red-Blue Non-Blocker. □

5.5.2 Parameter $\overline{k}$ When Each Taxon Has At Most One Prey

It is known that PDD remains NP-hard if the food-web is acyclic and every vertex has at most one prey [FSW11]. By Proposition 5.19 we know that PDD is W[1]-hard when parameterized by the acceptable loss of diversity $\overline{D}$ or the minimum number of extincting taxa $\overline{k}$. In the following, we show that in the special case that each taxon has at most one prey, PDD is FPT when parameterized with $\overline{k}$. Observe that each taxon has at most one prey if and only if the food-web is an out-forest.

Proposition 5.20. PDD *can be solved in time* $\mathcal{O}(2^{3\overline{k}+o(\overline{k})}n\log n+n^5)$*, if each taxon has at most one prey.*

In order to show the claimed result we again resort to the technique of color coding. We define a colored version of the problem, show how to solve it, and at the end of the section, we show how to reduce instances of the uncolored problem to the colored version.

We define 2-colored Optimizing PD with Dependencies (2-c-PDD), a colored version of the problem, as follows. Additionally to the usual input $\mathcal{T}$, $\mathcal{F}$, k, and D of PDD, we receive a coloring c which assigns each taxon $x \in X$ either 0 or 1. In 2-c-PDD, we ask whether there is a set $S \subseteq X$ that holds each of the following

(a) the size of S is at most k,

(b) the phylogenetic diversity of S is at least D,

(c) S is viable,

(d) each taxon $x \in X \setminus S$ satisfies $c(x) = 0$,

(e) each neighbor y of $X \setminus S$ in the food-web $\mathcal{F}$ satisfies $c(y) = 1$, and

(f) If $\mathrm{off}(v) \subseteq X \setminus S$ then $\mathrm{off}(u) \subseteq X \setminus S$ or there is a taxon $y \in \mathrm{off}(u)$ with $c(y) = 1$ for each edge $uv \in E(\mathcal{T})$.

Observe that if the color $c(x)$ of a taxon $x \in X$ is 1, then x has to be saved, while x may be saved or may go extinct if the color $c(x)$ is 0.

Observation 5.21. *Given a* yes*-instance* $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D, c)$ *of* 2-c-PDD *with solution* S *and a (possibly internal) vertex* $w \in V(\mathcal{T})$ *with* $c(x) = 0$ *for each* $x \in \mathrm{off}(w)$*. Then either* $\mathrm{off}(w) \subseteq S$ *or* $\mathrm{off}(w) \subseteq X \setminus S$*.*

Proof. If $\text{off}(w) \subseteq S$, we are done.

Let there be a taxon $x \in \text{off}(w)$ which is not in S. Let v be the descendant of x (possibly $v = x$) such that $\text{off}(v) \subseteq X \setminus S$ and $\text{off}(u) \cap S \neq \emptyset$ for the parent u of v. Because S is a solution, $\text{off}(v) \subseteq X \setminus S$, and $\text{off}(u) \cap S \neq \emptyset$, we conclude that there is a taxon $y \in \text{off}(u)$ with $c(y) = 1$. Therefore, w is a descendant of v and we conclude that $\text{off}(w) \subseteq \text{off}(v) \subseteq X \setminus S$. □

Definition 5.22.

a) Given a set of taxa and a coloring $c : X \to \{0, 1\}$, we define $c^{-1}(0)$ and $c^{-1}(1)$ to be the subsets of X that c maps to 0 or 1, respectively.

b) Given a food-web $\mathcal{F}$ and a coloring c, we define $\mathcal{F}_{c,0}$ to be the underlying undirected graph of $\mathcal{F}$ induced by $c^{-1}(0)$.

Observation 5.23. *Given a* `yes`*-instance* $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D, c)$ *of* 2-C-PDD *with solution* S *and let* C *be a connected component of* $\mathcal{F}_{c,0}$*. Then either* $V(C) \subseteq S$ *or* $V(C) \subseteq X \setminus S$*.*

Proof. If $V(C) \subseteq S$, we are done.

Since S is a solution, each neighbor y of $X \setminus S$ in the food-web satisfies $c(y) = 1$. Therefore, if there is a taxon $x \in V(C)$ which is not in S, then each neighbor of x is in $X \setminus S$ or in $c^{-1}(1)$. By definition, C is a connected component and $V(C) \subseteq c^{-1}(0)$. We conclude that $V(C) \subseteq X \setminus S$. □

Lemma 5.24. 2-C-PDD *can be solved in* $\mathcal{O}(n^4)$ *time if each taxon has at most one prey.*

Proof. Intuition. We reduce an instance $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D, c)$ of 2-C-PDD to an instance of KNAPSACK, in which the threshold of profit is limited in $\overline{k}$. In KNAPSACK we are given a set of items $\mathcal{A}$, a cost-function $c : \mathcal{A} \to \mathbb{N}$, a value-function $d : \mathcal{A} \to \mathbb{N}$, and two integers B—the budget—and D—the required value. It is asked whether there is a set $A \subseteq \mathcal{A}$ such that $c_\Sigma(A) \leq B$ and $d_\Sigma(A) \geq D$. KNAPSACK is `NP`-hard [Kar72] and can be solved in $\mathcal{O}(D \cdot |\mathcal{A}|^2)$ time [GRR19].

Algorithm. Compute the set Z of edges uv of $\mathcal{T}$ which hold that $\text{off}(v) \subseteq c^{-1}(0)$ and there is a taxon $y \in \text{off}(u)$ with $c(y) = 1$. For each $e = uv \in Z$, define integers $p_v := |\text{off}(v)|$ and $q_v := \lambda(e) + \sum_{e' \in E(\mathcal{T}_v)} \lambda(e')$, where $\mathcal{T}_v$ is the subtree of $\mathcal{T}$ rooted at v. Intuitively, p_v is the number of taxa and q_v is the diversity that would be lost if all taxa in $\text{off}(v)$ would go extinct.

Define a graph G on the vertex set $V_G := \{v \mid uv \in Z\}$. Compute the connected components $C_1, \ldots, C_\ell$ of $\mathcal{F}_{c,0}$. Iterate over C_i. Compute the edges $u_1v_1, \ldots, u_qv_q \in Z$ with $\text{off}(v_j) \cap V(C_i) \neq \emptyset$ for $j \in [q]$. Make $v_1, \ldots, v_q$ a clique in G.

Compute the set $\mathcal{A}$ of connected components of G.

Iterate over $v \in V_G$. If there is a taxon $y \in c^{-1}(1)$ and in $\mathcal{F}$ there is a path from some taxon $x \in \text{off}(v)$ to y, then remove the connected component A from $\mathcal{A}$.

We define a Knapsack instance $\mathcal{I}' := (\mathcal{A}, c', d, B, D')$ as follows. The set of items is $\mathcal{A}$, we define the budget B as $\overline{D} = \text{PD}_\mathcal{T}(X) - D$ and the desired profit D' as $\overline{k}$. The cost- and value-function are defined as follows. For an item $A \in \mathcal{A}$, we define $c'(A)$ to be $\sum_{v \in V(A)} q_v$, and $d(A)$ to be $\sum_{v \in V(A)} p_v$.

If $\mathcal{I}'$ is a `yes`-instance of Knapsack, we return `yes`. Otherwise, we return `no`.

Correctness. Observe that $\{\text{off}(v) \mid uv \in Z\}$ and $\{C_1, \ldots, C_\ell\} =: \mathcal{C}_\mathcal{F}$ are both partitions of $c^{-1}(0)$. We define the set $X(A) := \{x \in X \mid x \in \text{off}(v), v \in V(A)\}$, for connected components $A \in \mathcal{A}$ of G. We show that the instance $\mathcal{I}$ of 2-c-PDD is a `yes`-instance if and only if the instance $\mathcal{I}'$ of Knapsack is a `yes`-instance.

Let $\mathcal{I}'$ be a `yes`-instance of Knapsack. So, there is a set $S \subseteq \mathcal{A}$ with $d_\Sigma(S) \geq \overline{k}$ and $c'_\Sigma(S) \leq \text{PD}_\mathcal{T}(X) - D$. Let $P \subseteq V_G$ be the union of the vertices $V(A)$ for A in S and let $Q \subseteq X$ to be the set union of taxa $X(A)$ for A in S. We want to show that $R := X \setminus Q$ is a solution of the instance $\mathcal{I}$ of 2-c-PDD. Because $d_\Sigma(S) \leq \overline{k}$ we conclude $\overline{k} \geq \sum_{v \in P} p_v = \sum_{v \in P} |\text{off}(v)| \geq |Q|$. Consequently, the size of R is at most $|X| - \overline{k} = k$. By the definition of Z, we know that each $v \in P$ has a parent u which satisfys that there is a taxon $y \in \text{off}(u)$ with $c(y) = 1$. Consequently,

$$\text{PD}_\mathcal{T}(R) = \text{PD}_\mathcal{T}(X) - \sum_{v \in P} q_v = \text{PD}_\mathcal{T}(X) - d_\Sigma(S) \geq \text{PD}_\mathcal{T}(X) - \overline{D} = D. \tag{5.4}$$

Let $x \in Q$ and $y \in X$ be taxa which satisfy that y can be reached from x in $\mathcal{F}$. Because x is in Q, there is a connected component $A \in S$ of G and a vertex $v \in V(A)$ that satisfies $x \in \text{off}(v)$. By the construction, we know that each taxon that can be reached from x, including y, are colored with 0 because A would have been removed from $\mathcal{A}$, otherwise. Consequently, y is in the same connected component as x in $\mathcal{F}_{c,0}$ and thus there is a vertex $v' \in V(A)$ that satisfies $y \in \text{off}(v')$. Therefore, y is also in Q and we conclude that R is viable. By the definition of $\mathcal{A}$, we conclude $Q \subseteq c^{-1}(0)$. By the construction the taxa of a connected component of $\mathcal{F}_{c,0}$ are a subset of the union $\bigcup_{v \in V(A)} \text{off}(v)$ for some unique $A \in \mathcal{A}$. Therefore, the neighbors of Q in $\mathcal{F}$ are colored with 1. So, R is a solution of the `yes`-instance $\mathcal{I}$ of 2-c-PDD.

Conversely, let $\mathcal{I}$ be a `yes`-instance of 2-c-PDD with solution S. Let Z be the set of edges as defined in the algorithm and let $C_1, \ldots, C_\ell$ be the connected components of $\mathcal{F}_{c,0}$. Let $\mathcal{A}$ be the set of connected components of G. By Observations 5.21

and 5.23, we conclude that for each connected component A in $\mathcal{A}$ either $X(A) \subseteq S$ or $X(A) \subseteq X \setminus S$. Now, let $\mathcal{S} \subseteq \mathcal{A}$ be the set of connected components A of G that satisfy $X(A) \subseteq X \setminus S$. We observe that $X \setminus S$ has a size of $\overline{k}$ and $\mathrm{PD}_{\mathcal{T}}(S) \geq D$, as S is a solution of $\mathcal{I}$ and so $d_{\Sigma}(\mathcal{S}) = \sum_{A \in \mathcal{S}, v \in A} p_v = \sum_{A \in \mathcal{S}, v \in A} |\operatorname{off}(v)| = |X \setminus S| \geq \overline{k} = D'$. Further, $c'_{\Sigma}(\mathcal{S}) = \sum_{A \in \mathcal{S}, v \in A} q_v = \mathrm{PD}_{\mathcal{T}}(X) - \mathrm{PD}_{\mathcal{T}}(S) \geq \mathrm{PD}_{\mathcal{T}}(X) - D = B$. Hence, $\mathcal{S}$ is a solution to the `yes`-instance $\mathcal{I}'$ of Knapsack.

Running time. Observe that $|E(\mathcal{F})| \in \mathcal{O}(n)$ because each taxon has at most one prey. The size of Z is at most n and therefore also the size of $|\mathcal{A}|$. All steps in the reduction can be computed in $\mathcal{O}(n^4)$ time. Defining the instance $\mathcal{I}'$ is done in $\mathcal{O}(n)$ time. Computing whether $\mathcal{I}'$ is a `yes`-instance of Knapsack can be done in $\mathcal{O}(\overline{k} \cdot n^2)$ time. Therefore, the overall running time is $\mathcal{O}(n^4)$. □

It remains to show how to utilize the result of Lemma 5.24 to compute a solution for an instance of PDD. Other than in the proofs of Theorems 5.1, 5.2, and 5.3, we resort on the concept of (n, k)-universal sets instead of (n, k)-perfect hash functions.

Proof (of Proposition 5.20). Algorithm. Let $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D)$ be an instance of PDD. Let $x_1, \ldots, x_n \in X$ be an arbitrary order of the taxa.

Compute an $(n, 3\overline{k})$-universal set $\mathcal{U}$. Iterate over $A \in \mathcal{U}$ and define 2-colorings $c_A : X \to \{0, 1\}$ by setting $c_A(x_i) := 1$ if and only if $i \in A$. Then, solve the instances $\mathcal{I}_A := (\mathcal{T}, \mathcal{F}, k, D, c_A)$ of 2-c-PDD with Lemma 5.24. Return `yes` if there is an $A \in \mathcal{U}$, such that $\mathcal{I}_A$ is a `yes`-instance. Otherwise, return `no`.

Correctness. If $\mathcal{I}_A$ for some $A \in \mathcal{U}$ is a `yes`-instance of 2-c-PDD and let $S \subseteq X$ be a solution. Then, by the definition, S is viable, $|S| \leq k$, and $\mathrm{PD}_{\mathcal{T}}(S) \geq D$. Therefore, instance $\mathcal{I}$ is a `yes`-instance of PDD with solution S.

Assume that $\mathcal{I}$ is a `yes`-instance of PDD with solution S. Define $Y := X \setminus S$—the species that die out. Observe that $|Y| = |X| - |S| \geq |X| - k = \overline{k}$. Let $u_1v_1, \ldots, u_\ell v_\ell$ be the edges in $E(\mathcal{T})$ that satisfy $\operatorname{off}(v_i) \subseteq Y$ and there is a taxon $\bar{x}_i \in \operatorname{off}(u_i) \setminus Y$. Define $Z_1 := \{\bar{x}_1, \ldots, \bar{x}_\ell\}$ and observe $|Z_1| \leq \ell \leq |Y| \leq \overline{k}$. Now, let Z_2 be the set of neighbors of Y in $\mathcal{F}$. Because each taxon has at most one prey, we conclude that if $x \in Y$ then also $N_{>}(x) \subseteq Y$. Therefore, each taxon in Z_2 is the prey of at least one taxon of Y. Consequently, $|Z_2| \leq |Y| \leq \overline{k}$. Define $Z := Y \cup Z_1 \cup Z_2$ and by the previous, we know $|Z| \leq 3\overline{k}$. If necessary, add taxa to Z, such that contains $3\overline{k}$ taxa. Define sets $Y' := \{i \mid x_i \in Y\}$ and $Z' := \{i \mid x_i \in Z\}$.

Because $\mathcal{U}$ is an $(n, 3\overline{k})$-universal set, $\{A \cap S \mid A \in \mathcal{U}\}$ contains all $2^{3\overline{k}}$ subsets of S for any $S \subseteq [n]$ of size $3\overline{k}$. Thus, there is an $A^* \in \mathcal{U}$ such that $A \cap Z' = Z' \setminus Y'$. Therefore, c_{A^*} maps each $y \in Y$ to 0, each $z \in Z_1 \cup Z_2$ to 1 and each other taxon to any value of $\{0, 1\}$. We conclude that the instance $\mathcal{I}_{A^*} := (\mathcal{T}, \mathcal{F}, k, D, c_{A^*})$ is a `yes`-instance of 2-c-PDD.

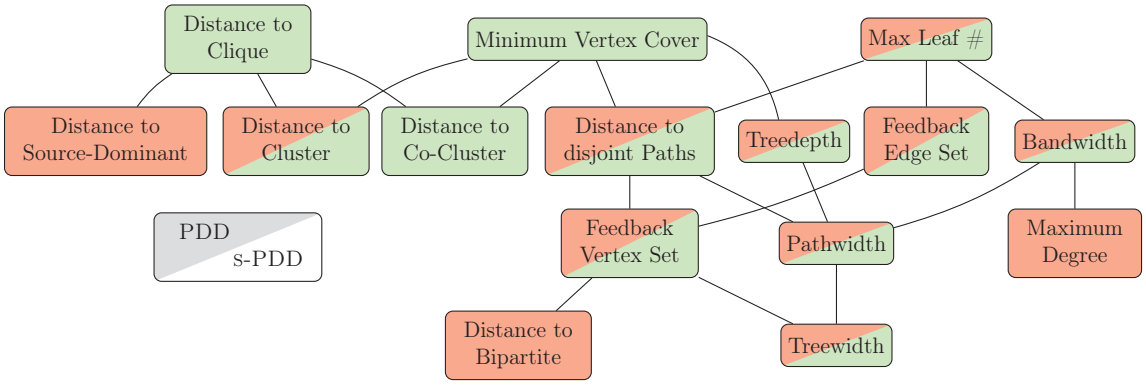


Figure 5.2: This figure depicts the relationship between a structural parameter of the food-web and the complexity of solving PDD and s-PDD. A parameter π is marked green (◎) if PDD admits an FPT-algorithm with respect to π, red (◎) if s-PDD is NP-hard for constant values of π, or red and green (◎) if PDD is NP-hard for constant values of π while s-PDD admits an FPT-algorithm with respect to π. Two parameters π_1 and π_2 are connected with an edge if in every graph the parameter π_1 further up van be bounded by a function in π_2. A more in-depth look into the hierarchy of graph-parameters can be found in [SWF+20].

Running Time. $(n, 3\overline{k})$-universal sets of size $2^{3\overline{k}}(3\overline{k})^{\mathcal{O}(\log(3\overline{k}))}\log n = 2^{3\overline{k}}2^{\mathcal{O}(\log^2(\overline{k}))}\log n$ are constructed in time $2^{3\overline{k}}(3\overline{k})^{\mathcal{O}(\log(3\overline{k}))}n\log n = 2^{3\overline{k}}2^{\mathcal{O}(\log^2(\overline{k}))}n\log n$ [NSS95].

For a given $A \in \mathcal{U}$, we can construct $\mathcal{I}_A$ in $\mathcal{O}(|A|) = \mathcal{O}(n)$ time. By Lemma 5.24, we can compute whether $\mathcal{I}_A$ is a yes-instance in $\mathcal{O}(n^4)$ time. Therefore, the overall running time is $\mathcal{O}(2^{3\overline{k}}2^{\mathcal{O}(\log^2(\overline{k}))}n\log n + n^5) = \mathcal{O}(2^{3\overline{k}+o(\overline{k})}n\log n + n^5)$. □

5.6 Structural Parameters

In this section, we study how the structure of the food-web affects the complexity of s-PDD and PDD. Figure 5.2 and Table 5.1 depict a summary of these complexity results.

Some of these results are already shown by Faller et al. [FSW11]. These include that PDD remains NP-hard on instances in which the food-web is a bipartite graph [FSW11] and s-PDD is already NP-hard if the food-web is a tree of height three [FSW11]. Further, Faller et al. [FSW11] also gave a reduction from Vertex Cover to PDD. Because Vertex Cover is NP-hard for graphs of maximum degree three [Moh01], also the constructed food-web has a maximum degree of three.

In the next subsection, we have an in-depth look into the parameterization by the distance to cluster, also called cluster vertex deletion number (dist-clust) of the

food-web. There, we show that PDD is NP-hard even if the underlying undirected graph of the food-web is a cluster graph but S-PDD is FPT when parameterized by dist-clust. Afterward, we show that PDD is FPT when parameterized by the distance to co-cluster (dist-co-clust) and that S-PDD is FPT with respect to the treewith ($\mathrm{tw}_{\mathcal{F}}$) of the food-web. Consequently, S-PDD can be solved in polynomial time if the food-web is a tree, disproving Conjecture 4.2. in [FSW11].

5.6.1 Distance to Cluster

In this subsection, we consider the special case that given an instance of PDD or S-PDD, we need to remove only a few vertices from the food-web $\mathcal{F}$ to obtain a cluster graph in the underlying undirected graph. Recall that a graph is a cluster graph if the existence of edges $\{u, v\}$ and $\{v, w\}$ imply the existence of the edge $\{u, w\}$.

Because $\mathcal{F}$ is acyclic, we have the following: Every clique in $\mathcal{F}$ has exactly one vertex $v_0 \in V(C)$ such that $v_0 \in N_<(v)$ for each $v \in V(C) \setminus \{v_0\}$. In fact, after applying Reduction Rule 5.7 exhaustively to any cluster graph, every connected component is a star in which all edges are directed away from the center.

In this subsection, we further use the following classification of instances. Recall that $\mathcal{T}\langle Y\rangle$ is the spanning tree of the vertices in Y.

Definition 5.25. An instance $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D)$ of PDD or S-PDD is *source-separating* if $\mathcal{T}\langle\{\rho\} \cup \mathrm{sources}(\mathcal{F})\rangle$ and $\mathcal{T}\langle\{\rho\} \cup (X \setminus \mathrm{sources}(\mathcal{F}))\rangle$ only have ρ as common vertex. Here, ρ is the root of $\mathcal{T}$.

Figure 5.3 depicts in (1) an example of a source-separating instance.

In Theorem 5.6, we show that S-PDD is FPT with respect to the distance to cluster of the food-web. Afterward, we show that PDD, however, is NP-hard on source-separating instances in which the food-web has a cluster graph as underlying graph. Then, we show that PDD can be solved in polynomial time in a further special case.

Theorem 5.6. S-PDD *can be solved in* $\mathcal{O}(6^d \cdot n^2 \cdot m \cdot k^2)$ *time, when we are given a set* $Y \subseteq X$ *of size* d *such that* $\mathcal{F} - Y$ *is a cluster graph.*

To prove this theorem, we first consider a special case in the following auxiliary lemma.

Lemma 5.26. *Given an instance* $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D)$ *of* S-PDD *and a set* $Y \subseteq X$ *of size* d *such that* $\mathcal{F} - Y$ *is a source-dominant graph, we can compute whether there is a viable set* $S \cup Y$ *with* $|S \cup Y| \leq k$ *and* $PD_{\mathcal{T}}(S \cup Y) \geq D$ *in* $\mathcal{O}(3^d \cdot n \cdot k^2)$ *time.*

Proof. We provide a dynamic programming algorithm. Let $C_1, \dots, C_c$ be the connected components of $\mathcal{F} - Y$ and let x_i be the only source in C_i for each $i \in [c]$.

Table definition. A set $S \subseteq X \setminus Y$ of taxa is (ℓ, Z)-*feasible*, if $|S| \leq \ell$ and $S \cup Z$ is viable, for an integer ℓ and a set Z of taxa. The dynamic programming algorithm has tables DP and DP_i for each $i \in [c]$. We want entry $\mathrm{DP}[i, \ell, Z]$, for $i \in [c]$, $\ell \in [k]_0$, and $Z \subseteq Y$, to store the largest phylogenetic diversity $\mathrm{PD}_{\mathcal{T}}(S)$ of an (ℓ, Z)-feasible set $S \subseteq C_1 \cup \dots \cup C_i$, and $-\infty$ if no such set S exists.

The table entries $\mathrm{DP}_i[j, b, \ell, Z]$ additionally have a dimension b with $b \in \{0, 1\}$. For $b = 0$, an entry $\mathrm{DP}_i[j, b, \ell, Z]$ stores the largest phylogenetic diversity $\mathrm{PD}_{\mathcal{T}}(S)$ of an (ℓ, Z)-feasible set $S \subseteq \{x_1^{(i)}, \dots, x_j^{(i)}\}$. For $b = 1$, additionally some vertex $v_{j'}^{(i)}$ with $j' < j$ needs to be contained in S.

Algorithm. Iterate over the edges of $\mathcal{F}$. For each edge $uv \in E(\mathcal{F})$ with $u, v \in Y$, remove all edges incoming at v, including uv, from $E(\mathcal{F})$. After this removal, v is a new source.

We initialize the base cases of DP_i by setting $\mathrm{DP}_i[j, 0, 0, Z] := 0$ for each $i \in [c]$, each $j \in [|C_i|]$, and every $Z \subseteq \mathrm{sources}(\mathcal{F})$. Moreover, $\mathrm{DP}_i[1, b, \ell, Z] := \lambda(\rho v_1^{(i)})$ if $\ell \geq 1$ and $Z \subseteq N_>(v_1^{(i)}) \cup \mathrm{sources}(\mathcal{F})$; and $\mathrm{DP}_i[1, b, \ell, Z] := -\infty$, otherwise.

To compute further values for $j \in [|C_i| - 1]$, $b \in \{0, 1\}$, and $\ell \in [k]$ we use the recurrences

$$\mathrm{DP}_i[j+1, b, \ell, Z] = \max\{\mathrm{DP}_i[j, b, \ell, Z], \mathrm{DP}_i[j, b', \ell - 1, Z \setminus N_>(v_{j+1}^{(i)})] + \lambda(\rho v_{j+1}^{(i)})\}, \tag{5.5}$$

where $b' = 0$ if there is an edge from a vertex in Y to $x_{j+1}^{(i)}$ and otherwise $b' = 1$.

Finally, we set $\mathrm{DP}[1, \ell, Z] := \mathrm{DP}_1[|C_1|, 0, \ell, Z]$ and compute further values with

$$\mathrm{DP}[i+1, \ell, Z] = \max_{Z' \subseteq Z, \ell' \in [\ell]_0} \mathrm{DP}[i, \ell', Z'] + \mathrm{DP}_{i+1}[|C_{i+1}|, 0, \ell - \ell', Z \setminus Z']. \tag{5.6}$$

There is a viable set $S \cup Y$ with $|S \cup Y| \leq k$ and $\mathrm{PD}_{\mathcal{T}}(S \cup Y) \geq D$ if and only if $\mathrm{DP}[c, k - |Y|, Z] \geq D - \mathrm{PD}_{\mathcal{T}}(Y)$.

Correctness. Assume that DP stores the intended values. Then, there is an (ℓ, Z)-feasible set $S \subseteq X \setminus Y$, if $\mathrm{DP}[c, k - |Y|, Z] \geq D - \mathrm{PD}_{\mathcal{T}}(Y)$. First, this implies that $S \cup Y$ is viable. Moreover, since S has size at most $k - |Y|$, we obtain $|S \cup Y| \leq k$. Finally, because $\mathcal{T}$ is a star and S and Y are disjoint, $\mathrm{PD}_{\mathcal{T}}(S) \geq D - \mathrm{PD}_{\mathcal{T}}(Y)$ implies $\mathrm{PD}_{\mathcal{T}}(S \cup Y) \geq D$. The converse direction can be shown analogously.

It remains to show that DP and DP_i store the right values. The base cases are correct. Towards the correctness of Recurrence (5.5), as an induction hypothesis, assume that $\mathrm{DP}_i[j, b, \ell, Z]$ stores the desired value for a fixed $j \in [|C_i| - 1]$, each $i \in [c]$, $b \in \{0, 1\}$, $\ell \in [k]_0$ and every $Z \subseteq Y$. Let $\mathrm{DP}_i[j+1, b, \ell, Z]$ store d. We show that

there is an (ℓ, Z)-feasible set $S \subseteq \{x_1^{(i)}, \ldots, x_{j+1}^{(i)}\}$ with $\mathrm{PD}_{\mathcal{T}}(S) = d$. We conclude with Recurrence (5.5) that $\mathrm{DP}_i[j, b, \ell, Z]$ stores d or $\mathrm{DP}_i[j, 1, \ell - 1, Z \setminus N_>(v_{j+1}^{(i)})]$ stores $d - \lambda(\rho v_{j+1}^{(i)})$. If $\mathrm{DP}_i[j, b, \ell, Z]$ stores d, then because there is an (ℓ, Z)-feasible set $S \subseteq \{x_1^{(i)}, \ldots, x_j^{(i)}\} \subseteq \{x_1^{(i)}, \ldots, x_{j+1}^{(i)}\}$ we are done. If $\mathrm{DP}_i[j, 1, \ell-1, Z \setminus N_>(v_{j+1}^{(i)})]$ stores $d-\lambda(\rho v_{j+1}^{(i)})$ then there is an $(\ell-1, Z\setminus N_>(v_{j+1}^{(i)}))$-feasible set $S' \subseteq \{x_1^{(i)}, \ldots, x_j^{(i)}\}$ containing $x_1^{(i)}$ or some $x_{j'}^{(i)} \in N_>(Y)$. Consequently, also $S' \cup \{x_{j+1}^{(i)}\}$ is (ℓ, Z)-feasible and $\mathrm{PD}_{\mathcal{T}}(S' \cup \{x_{j+1}^{(i)}\}) = d$.

Now conversely, let $S \subseteq \{x_1^{(i)}, \ldots, x_{j+1}^{(i)}\}$ be an (ℓ, Z)-feasible set. We show that $\mathrm{DP}_i[j + 1, b, \ell, Z]$ stores at least $\mathrm{PD}_{\mathcal{T}}(S)$. If $S \subseteq \{x_1^{(i)}, \ldots, x_j^{(i)}\}$ then we know from the induction hypothesis that $\mathrm{DP}_i[j, b, \ell, Z]$ stores $\mathrm{PD}_{\mathcal{T}}(S)$ and therefore also $\mathrm{DP}_i[j + 1, b, \ell, Z]$ stores $\mathrm{PD}_{\mathcal{T}}(S)$. If $x_{j+1}^{(i)} \in S$, then S contains $x_1^{(i)}$ or some $x_{j'}^{(i)} \in N_>(Y)$. Define $S' := S \setminus \{x_{j+1}^{(i)}\}$. Then, $|S'| = \ell - 1$, and $S' \cup (Z \setminus N_>(x_{j+1}^{(i)}))$ is viable because S is (ℓ, Z)-feasible. Consequently, $\mathrm{DP}_i[j, 1, \ell - 1, Z \setminus N_>(x_{j+1}^{(i)})] \geq \mathrm{PD}_{\mathcal{T}}(S') = \mathrm{PD}_{\mathcal{T}}(S) - \lambda(\rho x_{j+1}^{(i)})$ and therefore, $\mathrm{DP}_i[j + 1, b, \ell, Z] \geq \mathrm{PD}_{\mathcal{T}}(S)$.

Now, we focus on the correctness of Recurrence (5.6). Let $\mathrm{DP}[i + 1, \ell, Z]$ store d. We show that there is an (ℓ, Z)-feasible set $S \subseteq C_1 \cup \cdots \cup C_{i+1}$ with $\mathrm{PD}_{\mathcal{T}}(S) = d$. Because $\mathrm{DP}[i + 1, \ell, Z]$ stores d, by Recurrence (5.6), there are $Z' \subseteq Z$ and $\ell' \in [\ell]_0$ such that $\mathrm{DP}[i, \ell', Z'] = d_1$, $\mathrm{DP}_{i+1}[|C_{i+1}|, 0, \ell - \ell', Z \setminus Z'] = d_2$, and $d_1 + d_2 = d$. By the induction hypothesis, there is an (ℓ', Z')-feasible set $S_1 \subseteq C_1 \cup \cdots \cup C_i$ and an $(\ell - \ell', Z \setminus Z')$-feasible set $S_2 \subseteq C_{i+1}$ such that $\mathrm{PD}_{\mathcal{T}}(S_1) = d_1$ and $\mathrm{PD}_{\mathcal{T}}(S_2) = d_2$. Then, $S := S_1 \cup S_2$ satisfies $|S| \leq |S_1| + |S_2| \leq \ell' + (\ell - \ell') = \ell$. Further, because Y has no outgoing edges, $Z' \subseteq N_>(S_1) \cup \mathrm{sources}(\mathcal{F})$ and $Z \setminus Z' \subseteq N_>(S_2) \cup \mathrm{sources}(\mathcal{F})$. Therefore, $Z \subseteq N_>(S) \cup \mathrm{sources}(\mathcal{F})$ and $S \cup Z$ is viable. We conclude that S is the desired set.

Let $S \subseteq C_1 \cup \cdots \cup C_{i+1}$ be an (ℓ, Z)-feasible set with $\mathrm{PD}_{\mathcal{T}}(S) = d$. We show that $\mathrm{DP}[i + 1, \ell, Z] \geq d$. Define $S_1 := S \cap (C_1 \cup \cdots \cup C_i)$ and $Z' := N_>(S_1) \cap Z$. We conclude that $S_1 \cap Z'$ is viable. Then, S_1 is (ℓ', Z')-feasible, where $\ell' := |S_1|$. Define $S_2 := S \cap C_{i+1} = S \setminus S_1$. Because $S \cup Z$ is viable and Z does not have outgoing edges, we know that $Z \subseteq N_>(S) \cup \mathrm{sources}(\mathcal{F})$. So, $Z \setminus Z' \subseteq N_>(S_2) \cup \mathrm{sources}(\mathcal{F})$ and because $|S_2| = |S| - |S_1| = \ell - \ell'$, we conclude that S_2 is $(\ell - \ell', Z \setminus Z')$-feasible. Consequently, $\mathrm{DP}[i, \ell', Z'] \geq \mathrm{PD}_{\mathcal{T}}(S_1)$ and $\mathrm{DP}_{i+1}[|C_{i+1}|, \ell - \ell', Z \setminus Z'] \geq \mathrm{PD}_{\mathcal{T}}(S_2)$. Hence, $\mathrm{DP}[i + 1, \ell, Z] \geq \mathrm{PD}_{\mathcal{T}}(S_1) + \mathrm{PD}_{\mathcal{T}}(S_2) = \mathrm{PD}_{\mathcal{T}}(S)$ because $\mathcal{T}$ is a star.

Running time. The tables DP and DP_i for $i \in [c]$ have $\mathcal{O}(2^d \cdot n \cdot k)$ entries in total. Whether one of the base cases applies can be checked in linear time. We can

compute the set $Z \setminus N_>(x)$ for any given $Z \subseteq Y$ and $x \in X$ in $\mathcal{O}(d^2)$ time. Therefore, the $\mathcal{O}(2^d \cdot n \cdot k)$ times we need to apply Recurrence (5.2) consume $\mathcal{O}(2^d d^2 \cdot n \cdot k)$ time in total. In Recurrence (5.3), each $x \in Y$ can be in Z', in $Z \setminus Z'$ or in $Y \setminus Z$ so that we can compute all the table entries of DP in $\mathcal{O}(3^d \cdot n \cdot k^2)$ which is also the overall running time. □

Proof (of Theorem 5.6). Algorithm. We iterate over all the subsets Z of Y and define $R_0 := Y \setminus Z$. We want that Z is the set of taxa that are surviving while the taxa in R_0 are going extinct. Compute the set $R \subseteq X$ of taxa r for which in $\mathcal{F}$ each path from r to s contains a taxon of R_0, for each $s \in \text{sources}(\mathcal{F})$. Compute $\mathcal{T}_Z := \mathcal{T} - R$ and $\mathcal{F}_Z := \mathcal{F} - R$. Continue with the next Z if $Z \cap R \neq \emptyset$.

With Lemma 5.26, compute whether there is a viable set $S' = S \cup Z$ in instance $\mathcal{I} = (\mathcal{T}_Z, \mathcal{F}_Z, k, D)$ such that $|S'| \leq k$ and $\text{PD}_{\mathcal{T}}(S') \geq D$. Return `yes`, if such a set exists. Otherwise, continue with the next subset Z of Y. After iterating over all subsets Z of Y, return `no`.

Correctness. Let Z be a given subset of Y. Assume that there is a solution S with $Y \cap S = Z$. Let x be a vertex for which each path from x to a source s of $\mathcal{F}$ contains a vertex of $Y \setminus Z$. Because S is viable, x is not in S.

The rest of the correctness follows by Lemma 5.26.

Running time. For each subset Z of Y, we can compute $\mathcal{T}_Z$ and $\mathcal{F}_Z$ in $\mathcal{O}(m \cdot n)$ time. By Lemma 5.26, we can compute whether there is a solution S with $Y \cap S = Z$ in $\mathcal{O}(3^d \cdot n \cdot k^2)$ time. Therefore, the overall running time is $\mathcal{O}(6^d \cdot n^2 \cdot m \cdot k^2)$. □

Next, we show that PDD, in contrast to S-PDD, is NP-hard even when the food-web is restricted to have a cluster graph as an underlying undirected graph. We obtain this hardness by a reduction from Vertex Cover on cubic graphs. In cubic graphs, the degree of each vertex is *exactly* three. Recall, in Vertex Cover we are given an undirected graph $G = (V, E)$ and an integer k and we ask whether a set $C \subseteq V$ of size at most k exists such that $u \in C$ or $v \in C$ for each edge $\{u, v\} \in E$. The set C is called a *vertex cover*. Vertex Cover is NP-hard on cubic graphs [Moh01].

Theorem 5.7. PDD *is* NP*-hard on source-separating instances in which the food-web is a cluster graph and each connected component has at most four vertices.*

Proof. Reduction. Let (G, k) be an instance of Vertex Cover, where $G = (V, E)$ is cubic. We define an instance $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k', D)$ of PDD as follows. Let $\mathcal{T}$ have a root ρ. For each vertex $v \in V$, we add a child v of ρ. For each edge $e = \{u, v\} \in E$, we add a child e of ρ and two children $[u, e]$ and $[v, e]$ of e. Let N be a big integer.

We set the weight of ρe to $N-1$ for each edge e in E. All other edges of $\mathcal{T}$ have a weight of 1. Additionally, for each edge $e = \{u, v\} \in E$ we add edges $(u, [u, e])$ and $(v, [v, e])$ to $\mathcal{F}$. Finally, we set $k' := |E| + k$ and $D := N \cdot |E| + k$.

Correctness. The instance $\mathcal{I}$ of PDD is constructed in polynomial time for a suitable N. The leaves in $\mathcal{T}$ are $V \cup \{[u, e], [v, e] \mid e = \{u, v\} \in E\}$. The sources of $\mathcal{F}$ are V. Therefore, $\mathcal{I}$ is source-separating. Let e_1, e_2, and e_3 be the edges incident with $v \in V(G)$. Each connected component in $\mathcal{F}$ contains four vertices, v, and $[v, e_i]$ for $i \in \{1, 2, 3\}$.[1]

We show that (G, k) is a `yes`-instance of Vertex Cover if and only if $\mathcal{I}$ is a `yes`-instance of PDD. Let $C \subseteq V$ be a vertex cover of G of size at most k. If necessary, add vertices to C until $|C| = k$. For each edge $e \in E$, let v_e be an endpoint of e that is contained in C. Note that v_e exists since C is a vertex cover. We show that $S := C \cup \{[v_e, e] \mid e \in E\}$ is a solution for $\mathcal{I}$: The size of S is $|C| + |E| = k + |E| = k'$. By definition, for each taxon $[v_e, e]$ we have $v_e \in C \subseteq S$, so S is viable. Further, as S contains a taxon $[v_e, e]$ for each edge $e \in E$, we conclude that $\mathrm{PD}_{\mathcal{T}}(S) \geq N \cdot |E| + \mathrm{PD}_{\mathcal{T}}(C) = N \cdot |E| + k = D$. Therefore, S is a solution of $\mathcal{I}$.

Conversely, let S be a solution of instance $\mathcal{I}$ of PDD. Define $C := S \cap V(G)$ and define $S' := S \setminus C$. Because $\mathrm{PD}_{\mathcal{T}}(S) \geq D$, we conclude that for each $e \in E$ at least one taxon $[u, e]$ with $u \in e$ is contained in S'. Thus, $|S'| \geq |E|$ and we conclude that the size of C is at most k. Because S is viable we conclude that u is in C for each $[u, e] \in S'$. Therefore, C is a vertex cover of size at most k of G. □

PDD is `NP`-hard even if each connected component in the food-web is a path of length three [FSW11]. So, we wonder if the hardness still holds if we restrict the food-web even more such that each connected component of the food-web contains at most two vertices. We were not able to show that PDD is `NP`-hard for this special case. However, in the next proposition, we show that PDD can be solved in polynomial time if we restrict the instance further to be source-separating as well.

Proposition 5.27. PDD *can be solved in* $\mathcal{O}(k \cdot n^2 \cdot \log^2 n)$ *time on source-separating instances, if the food-web only has isolated edges.*

Proof. In this algorithm, we use techniques similar to the way Bordewich et al. showed that phylogeny across two trees can be computed efficiently [BSS09]. We reduce a source-separating instance of PDD to $\mathcal{O}(k)$ instances of Minimum-Cost

[1] Observe that we only constructed the instance so that the connected components are stars, but one could add edges $([v, e_1], [v, e_2])$, $([v, e_1], [v, e_3])$, and $([v, e_2], [v, e_3])$ for each vertex v, to meet the formal requirement of a cluster graph.

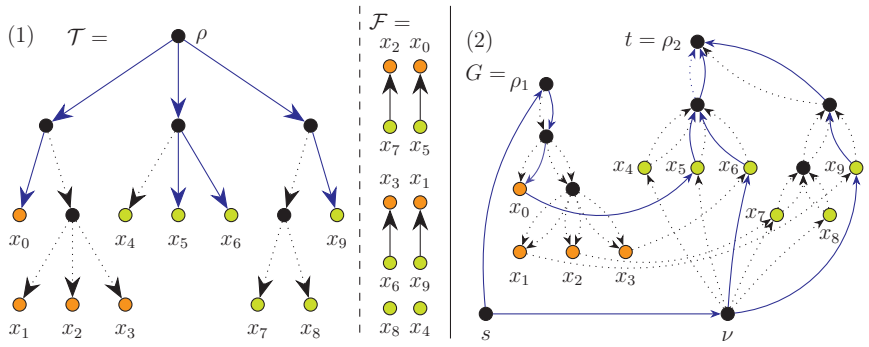


Figure 5.3: This figure shows in (1) a hypothetical source-separating instance of PDD. The sources are drawn in green and the predators in orange. In (2), the instance of MCNF is shown, as it would have been constructed in Proposition 5.27. For the sake of readability, capacities and costs are omitted. The blue and solid edges mark how a solution $\{x_0, x_5, x_6, x_9\}$ transfer to a flow.

Network Flow (MCNF), which can be solved in $\mathcal{O}(n^2 \log^2 n)$ time, each [Min86, AMO95].

In MCNF one is given a directed graph $G = (V, E)$ with two vertices $s, t \in V$—the *source* s and the *sink* t—and two integers F and C, where each edge $e \in E$ has a positive capacity $c(e)$ and a cost $a(e)$. The cost may be negative and G may have parallel edges.

A *flow* f assigns each edge e a non-negative integer value $f(e)$. The *cost of a flow* f is $\sum_{e \in E} f(e) \cdot a(e)$. A flow f is *q-proper* if $q = \sum_{w \in V} f(sw) = \sum_{u \in V} f(ut)$, $\sum_{u \in V} f(uv) = \sum_{w \in V} f(vw)$ for each vertex $v \in V \setminus \{s, t\}$, and $f(e) \leq c(e)$ for each edge e. For an instance (G, s, t, F, C, c, a) of MCNF, we ask whether G contains an F-proper flow of cost at most C.

Algorithm. Let $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D)$ be a source-separating instance of PDD. Define $\mathcal{T}_1$ and $\mathcal{T}_2$ to be the subtree of $\mathcal{T}$ spanning over the vertices $\{\rho\} \cup \text{sources}(\mathcal{F})$, and, respectively, $\{\rho\} \cup (X \setminus \text{sources}(\mathcal{F}))$. To avoid confusion, for $i \in \{1, 2\}$ the root of $\mathcal{T}_i$ is called ρ_i. For an example of the following reduction, consider Figure 5.3.

Iterate over $k' \in [\lfloor k/2 \rfloor]_0$. We define an instance $\mathcal{I}'_{k'} = (G, s, t, F, C, c, a)$ of MCNF as follows. The set of vertices of G are the vertices of $\mathcal{T}_1$ and $\mathcal{T}_2$ and two new vertices s and ν. The vertex s is the source and $t := \rho_2$ is the sink of G. We add edges $s\rho_1$ and $s\nu$, where $c(s\rho_1) = k'$ and $c(s\nu) = k - 2k'$. For each leaf x of $\mathcal{T}_2$, add an edge νx of capacity k. For each edge xy of $\mathcal{F}$, add an edge yx with a capacity of 1. Each edge so far has a cost of 0. For each edge $e = uv$ in $\mathcal{T}_1$, add two parallel edges e and e' where e has a capacity of 1 and a cost of $-\lambda(e)$ and e' has a capacity

of $k-1$ and a cost of 0. For each edge uv in $\mathcal{T}_2$, add two parallel edges vu and $(vu)'$ (note that the edges are reversed) where vu has a capacity of 1 and a cost of $-\lambda(uv)$ and $(vu)'$ has a capacity of $k-1$ and a cost of 0. To complete the instance, we set $F := k - k'$ and $C := -D$.

We compute a solution for $\mathcal{I}'_{k'}$ and return `yes` if $\mathcal{I}'_{k'}$ is a `yes`-instance of MCNF. If $\mathcal{I}'_{k'}$ is a `no`-instance of MCNF for each k', then we return `no`.

Correctness. The correctness is shown similarly as the correctness of the algorithm in [BSS09]. We show first that if $\mathcal{I}$ is a `yes`-instance of PDD then there is a k' such that $\mathcal{I}'_{k'}$ is a `yes`-instance of MCNF. Afterward, we show that if $\mathcal{I}'_{k'}$ is a `yes`-instance of MCNF for specific k' then $\mathcal{I}$ is a `yes`-instance of PDD.

Assume that $\mathcal{I}$ is a `yes`-instance of PDD with solution $S \subseteq X$. If necessary, add vertices to S until $|S| = k$. Let S_1 be the subset of vertices of S which are not sources. Let k' be the size of S_1. Further, denote the set $\{x \in X \mid xy \in E(\mathcal{F}), y \in S_1\}$ with S_2. Because S is viable and every connected component in $\mathcal{F}$ contains at most two vertices, we conclude $S_2 \subseteq S$. Then, we define $S_3 := S \setminus (S_1 \cup S_2)$. We define a flow f in $\mathcal{I}'_{k'}$ as follows. Let E_i be the set of edges on a path between ρ and S_i in $\mathcal{T}$, for $i \in [3]$. We set $f(e) = 1$ for each $e \in E_1$ and $f(vu) = 1$ for each $uv \in E_2 \cup E_3$. So far we have defined the flow that ensures that the cost is $-D$. Now, we ensure that the flow f is $k-k'$-proper. For each edge $xy \in E(\mathcal{F})$ with $y \in S_1$ we set $f(yx) = 1$. Further, for each $x \in S_3$ we set $f(\nu x) = 1$. We then set $f(s\rho_1) = k'$ and $f(sv) = k - 2k'$. For each edge e of $\mathcal{T}_1$, we set $f(e') = |\operatorname{off}(e) \cap S_1| - 1$. That is, the $f(e')$ is the number of offspring e' has in S_1 minus 1. For each edge $e = uv$ of $\mathcal{T}_2$ we set $f((vu)') = |\operatorname{off}(e) \cap (S_2 \cup S_3)| - 1$. It remains to show that f is $k - k'$-proper.

Claim: The flow f is $k - k'$-proper.

Proof. One can easily verify that $f(e) \leq c(e)$ for each $e \in E(G)$.

Observe that the size of S_3 is $|S| - |S_1 \cup S_2| = k - 2k'$. The flow leaving s is $\sum_{u \in V} f(su) = f(s\rho_1) + f(sv) = k' + k - 2k' = k - k'$. The flow entering $t = \rho_2$ is $\sum_{w \in V} f(wt) = |S_2| + |S_3| = k' + (k - 2k') = k - k'$. By definition, the flow entering and leaving ρ_1 has size k'. The flow leaving ν is $|S_3|$ and so equals to the flow entering ν, $f(s\nu) = k - 2k'$. Each leaf x has, if $x \in S$, a flow of 1 incoming and leaving, and 0, otherwise. For an internal vertex $v \neq \rho$ of $\mathcal{T}$ with children $w_1, \ldots, w_z$ and parent u, we observe $\operatorname{off}(uv) = \bigcup_{i=1}^{z} \operatorname{off}(vw_i)$. With this observation we conclude that f is $k - k'$-proper. ◇

Conversely, assume now that f is a $k - k'$-proper flow of $\mathcal{I}'_{k'}$. Let S be the set of vertices that are corresponding to leaves in $\mathcal{T}$ and that have a positive entering flow. We show that S is a solution for instance $\mathcal{I}$ of PDD. Since $f(s\rho_1) \leq k'$, we conclude that $|S \cap \operatorname{off}(\rho_1)| \leq k'$. We conclude $|S \cap \operatorname{off}(\rho_2)| \leq k - 2k' + |S \cap \operatorname{off}(\rho_1)| \leq k - k'$,

because $f(s\nu) \leq k - 2k'$. Thus, the size of S is at most k. If $x \in \text{off}(\rho_1)$ is in S then x has a positive flow leaving x and we conclude that $f(xy) > 0$ for $yx \in E(\mathcal{F})$. Therefore, y is in S and S is viable. Let E_1 be the set of edges e with $f(e) > 0$ and $a(e) < 0$. Recall that $a(e)$ is the cost of e. Only edges corresponding to an edge in $\mathcal{T}$ are in E_1. Let E_1' be the corresponding edges of E_1, especially, edges in $\mathcal{T}_2$ are turned around. Then, we observe

$$\text{PD}_{\mathcal{T}}(S) \geq \sum_{e \in E_1'} \lambda(e) = -\sum_{e \in E_1} f(e) \cdot a(e) \geq D. \tag{5.7}$$

Therefore, S is a solution for instance $\mathcal{I}$ of PDD.

Running time. For a given k', we can construct the instance $\mathcal{I}_{k'}'$ in linear time. A solution for an instance of MCNF can be computed in $\mathcal{O}(n^2 \log^2 n)$ time [Min86, AMO95]. An instance $\mathcal{I}_{k'}'$ contains at most $2n$ vertices and at most $6n$ edges. So, the overall running time is $\mathcal{O}(k \cdot n^2 \cdot \log^2 n)$. □

5.6.2 Distance to Co-Cluster

In this section, we show that PDD is FPT with respect to the distance to co-cluster of the food-web. A graph is a co-cluster graph if its complement graph is a cluster graph.

In our algorithm, we solve as a subroutine an auxiliary problem, called Hitting Set with Tree-Profits, in which we are given a universe $\mathcal{U}$, a family of subsets $\mathcal{W}$ of $\mathcal{U}$, a $\mathcal{U}$-tree $\mathcal{T}$, and integers k and D. We ask whether there is a set $S \subseteq \mathcal{U}$ of size at most k such that $\text{PD}_{\mathcal{T}}(S) \geq D$ and $S \cap W \neq \emptyset$ for each $W \in \mathcal{W}$.

Lemma 5.28. Hitting Set with Tree-Profits *can be solved in* $\mathcal{O}(3^{|\mathcal{W}|} \cdot n)$ *time.*

Proof. We adopt the dynamic programming algorithm with which one can solve Max-PD with weighted costs for saving taxa in pseudo-polynomial time [PG07].

Table definition. We define two tables DP and an auxiliary table DP′. We want that entry $\text{DP}[v, 0, \mathcal{M}]$, for $v \in V(\mathcal{T})$, and $\mathcal{M} \subseteq \mathcal{W}$, stores 0 if $\mathcal{M}$ is empty and $\text{DP}[v, 1, \mathcal{M}]$ stores the biggest phylogenetic diversity $\text{PD}_{\mathcal{T}_v}(S)$ of a set $S \subseteq \text{off}(v)$ in the subtree $\mathcal{T}_v$ rooted at v where $S \cap M \neq \emptyset$ for each $M \in \mathcal{M}$. Otherwise, we store $-\infty$. For a vertex $v \in V(\mathcal{T})$ with children $w_1, \ldots, w_j$, in $\text{DP}'[v, i, b, \mathcal{M}]$ we only consider sets $S \subseteq \text{off}(w_1) \cup \cdots \cup \text{off}(w_i)$.

Algorithm. For a leaf $u \in \mathcal{U}$ of $\mathcal{T}$, in $\text{DP}[u, 1, \mathcal{M}]$ store 0 if $u \in M$ for each $M \in \mathcal{M}$. Otherwise, store $-\infty$. For a given vertex v of $\mathcal{T}$, in $\text{DP}[v, 0, \mathcal{M}]$ and $\text{DP}'[v, i, 0, \mathcal{M}]$ store 0 if $\mathcal{M} = \emptyset$. Otherwise, store $-\infty$.

Let v be an internal vertex of $\mathcal{T}$ with children $w_1, \ldots, w_z$. For each $b \in \{0,1\}$ and every family $\mathcal{M}$ of subsets of $\mathcal{W}$, we set $\mathrm{DP}'[v,1,b,\mathcal{M}] := \mathrm{DP}[w_1,b,\mathcal{M}] + b \cdot \lambda(vw_1)$. To compute further values, we use the recurrence

$$\mathrm{DP}'[v,i+1,1,\mathcal{M}] = \max_{\mathcal{M}',b_1,b_2} \mathrm{DP}'[v,i,b_1,\mathcal{M}'] + \mathrm{DP}[w_{i+1},b_2,\mathcal{M}\setminus\mathcal{M}'] + b_2 \cdot \lambda(vw_{i+1}). \tag{5.8}$$

Here, the maximum is taken over $\mathcal{M}' \subseteq \mathcal{M}$ and $b_1, b_2 \in \{0,1\}$ where we additionally require $b_1 + b_2 \geq 1$; and if $\mathcal{M}' \neq \emptyset$ then $b_1 = 1$; and if $\mathcal{M}' \neq \mathcal{M}$ then $b_2 = 1$. Finally, we set $\mathrm{DP}[v,b,\mathcal{M}] := \mathrm{DP}'[v,z,b,\mathcal{M}]$.

If $\mathrm{DP}[\rho,1,\mathcal{W}] \geq D$, return `yes`. Otherwise, return `no`.

Correctness. The base cases are correct. Overall, the correctness of Recurrence (5.8) can be shown analogously to [PG07].

Running time. As a tree has at most $2n$ vertices, the overall size of DP and DP′ is $\mathcal{O}(2^{|\mathcal{W}|} \cdot n)$. In Recurrence (5.8), there are three options for each $M \in \mathcal{W}$. Hence, the the total number of terms considered in the entire table can be computed in $\mathcal{O}(3^{|\mathcal{W}|} \cdot n)$ time. □

In the following, we reduce from PDD to Hitting Set with Tree-Profits. Herein, we select a subset of the modulator Y to survive. Additionally, we select the first taxon x_i which survives in $X \setminus Y$. Because $\mathcal{F} - Y$ is a co-cluster graph, x_i is in a specific independent set $I \subseteq X$ and any taxon $X \setminus (I \cup Y)$ feed on x_i. Then, by saving a taxon $x_j \in X \setminus (I \cup Y)$, any other taxon in $X \setminus Y$ has some prey. Subsequently, a solution is found by Lemma 5.28.

Theorem 5.8. PDD *can be solved in* $\mathcal{O}(6^d \cdot n^3)$ *time, when we are given a set* $Y \subseteq X$ *of size* d *such that* $\mathcal{F} - Y$ *is a co-cluster graph.*

Proof. Algorithm. Given an instance $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D)$ of PDD. Let $x_1, \ldots, x_n$ be a topological ordering of X which is induced by $\mathcal{F}$. Iterate over the subsets Z of Y. Let P_Z be the sources of $\mathcal{F}$ in $X \setminus Y$ and let Q_Z be $N_>(Z) \setminus Y$, the taxa in $X \setminus Y$ which are being fed by Z. Further, define $R_Z := P_Z \cup Q_Z \subseteq X \setminus Y$. Iterate over the vertices $x_i \in R_Z$. Let x_i be from the independent set I of the co-cluster graph $\mathcal{F} - Y$. Iterate over the vertices $x_j \in X \setminus (Y \cup I)$.

For each set Z, and taxa x_i and x_j, with Lemma 5.28 we compute the optimal solution for the case that Z is the set of taxa of Y that survive while all taxa of $Y \setminus Z$ go extinct, x_i is the first taxon in $X \setminus Y$, and x_j the first taxon in $X \setminus (Y \cup I)$ to survive. (The special cases that only taxa from $I \cup Y$ or only from Y survive are omitted here.)

More formally, we define instances $\mathcal{I}_{Z,i,j}$ of Hitting Set with Tree-Profits for $Z \subseteq Y$, and $i, j \in [n]$ with $i < j$ as follows. Let the universe $\mathcal{U}_{i,j}$ be the union of $\{x_{i+1}, \ldots, x_{j-1}\} \cap I$ and $\{x_{j+1}, \ldots, x_n\} \setminus Y$. In other words, we let the taxa in Y and in $\{x_1, \ldots, x_i\}$ and in $\{x_1, \ldots, x_j\} \setminus I$ go extinct. For each taxon $x \in Z$ compute the set $N_<(x)$. If $x \notin \text{sources}(\mathcal{F})$ and $N_<(x) \cap (Z \cup \{x_i, x_j\}) = \emptyset$, then add $N_<(x) \setminus Y$ to the family of sets $\mathcal{W}_{Z,i,j}$. For each edge $uv \in E(\mathcal{T})$ for which in $\text{off}(uv)$ a vertex of $Z \cup \{x_i, x_j\}$ is contained, add an edge uw for each child w of v and remove v and its incident edges to obtain $\mathcal{T}_{Z,i,j}$. Finally, we set $k' := k - |Z| - 2$ and $D' := D - \text{PD}_{\mathcal{T}}(Z \cup \{x_i, x_j\})$.

Solve $\mathcal{I}_{Z,i,j}$. If $\mathcal{I}_{Z,i,j}$ is a `yes`-instance then return `yes`. Otherwise, continue with the iteration. If $\mathcal{I}_{Z,i,j}$ is a `no`-instance for every $Z \subseteq Y$, and each $i, j \in [n]$, then return `no`.

Correctness. We show that if the algorithm returns `yes`, then $\mathcal{I}$ is a `yes`-instance of PDD. Afterward, we show the converse.

Let the algorithm return `yes`. Consequently, there is a set $Z \subseteq Y$, and there are taxa $x_i \in X \setminus Y$ and $x_j \in X \setminus (Y \cup V(I))$ such that $\mathcal{I}_{Z,i,j}$ is a `yes`-instance of Hitting Set with Tree-Profits. As before, I is the independent set such that $x_i \in V(I)$. Consequently, there is a set $S \subseteq \mathcal{U}_{i,j}$ of size at most $k - |Z| - 2$ such that $\text{PD}_{\mathcal{T}_{Z,i,j}}(S) \geq D' = D - \text{PD}_{\mathcal{T}}(Z \cup \{x_i, x_j\})$ and $S \cap W \neq \emptyset$ for each $W \in \mathcal{W}_{Z,i,j}$. We show that $S^* := S \cup Z \cup \{x_i, x_j\}$ is a solution for instance $\mathcal{I}$ of PDD. Clearly, $|S^*| = |S| + |Z| + 2 \leq k$ and $\text{PD}_{\mathcal{T}}(S^*) = \text{PD}_{\mathcal{T}_{Z,i,j}}(S) + \text{PD}_{\mathcal{T}}(Z \cup \{x_i, x_j\}) \geq D$ because $\mathcal{T}_{Z,i,j}$ is the tree which results from $\mathcal{T}$ after saving $Z \cup \{x_i, x_j\}$. Further, by definition x_i is a source, or is fed by Z. Because $\mathcal{F} - Y$ is a co-cluster graph and x_j is not in I, the independent set in which x_i is, we conclude that $x_j \in N_>(x_i)$. As $S \cap W \neq \emptyset$ for each $W \in \mathcal{W}_{Z,i,j}$, each taxon $x \in Z$ has a prey in $Z \cup \{x_i, x_j\}$ or in S so that $N_<(x) \cap S^* \neq \emptyset$. Therefore, S^* is viable and indeed a solution for $\mathcal{I}$.

Assume now that S is a solution for instance $\mathcal{I}$ of PDD. We define $Z := S \cap Y$ and let x_i and x_j be the taxa in $S \setminus Y$, respectively $S \setminus (Y \cup I)$, with the smallest index. As before, I is the independent set of x_i. We show that instance $\mathcal{I}_{Z,i,j}$ of Hitting Set with Tree-Profits has $S^* := S \setminus (Z \cup \{x_i, x_j\})$ as a solution. Clearly, the size of S^* is $|S| - |Z| - 2 \leq k'$ and by the definition of $\mathcal{T}_{Z,i,j}$, we also conclude that $\text{PD}_{\mathcal{T}_{Z,i,j}}(S^*) \geq D'$. Let $M \in \mathcal{W}_{Z,i,j}$. There is a taxon $z \in Z$ that, by definition, satisfies $M = N_<(z) \setminus Y$, and $z \notin \text{sources}(\mathcal{F})$, and $N_<(z) \cap (Z \cup \{x_i, x_j\}) = \emptyset$. Consequently, as S is viable, there is a taxon $x \in S \cap N_<(z)$ so that $S \cap M \neq \emptyset$. Hence, S^* is a solution of instance $\mathcal{I}_{Z,i,j}$ of Hitting Set with Tree-Profits.

Running time. For a given $Z \subseteq Y$, we can compute the topological order $x_1, \ldots, x_n$ and the set R_Y in $\mathcal{O}(n^2)$ time. The iterations over x_i and x_j take $\mathcal{O}(n^2)$ time. Observe, $|\mathcal{W}_{Z,i,j}| \leq |Z|$. By Lemma 5.28, checking whether $\mathcal{I}_{Z,i,j}$ is a `yes`-instance

takes $\mathcal{O}(3^d n)$ time each. So, the overall running time is $\mathcal{O}(6^d \cdot n^3)$ time. □

5.6.3 Treewidth

Conjecture 4.2., formulated by Faller et al. [FSW11] supposes that S-PDD remains NP-hard on instances where the underlying graph of the food-web is a tree. In this subsection, assuming P ≠ NP, we disprove this conjecture by showing that S-PDD can be solved in polynomial time on food-webs which are trees. We even show a stronger result: S-PDD is FPT with respect to the treewidth of the food-web.

Theorem 5.9. S-PDD *can be solved in* $\mathcal{O}(9^{\mathrm{tw}_{\mathcal{F}}} \cdot nk)$ *time.*

To show Theorem 5.9, we define a dynamic programming algorithm over a tree-decomposition of $\mathcal{F}$. In each bag, we divide the taxa into three sets indicating that they a) "are supposed to go extinct", b) "will be saved" but "still need prey", c) or "will be saved" without restrictions. The algorithm is similar to the standard treewidth algorithm for Dominating Set [CFK+15].

Proof. Let $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D)$ be an instance of S-PDD. We define a dynamic programming algorithm over a nice tree-decomposition T of $\mathcal{F} = (V_{\mathcal{F}}, E_{\mathcal{F}})$.

For a node $t \in T$, let Q_t be the bag associated with t and let V_t be the union of bags in the subtree of T rooted at t.

Definition of the Table. We index solutions by a partition $R \cup G \cup B$ of Q_t, and a non-negative integer s. For a set of taxa $Y \subseteq V_t$, we call a vertex $u \in V_t$ *red with respect to* Y if u is in Y and u has a predator but no prey in Y. We call u *green with respect to* Y if u is in Y and a) u is a source in $\mathcal{F}$, or b) has prey in Y. Finally, we call u *black with respect to* Y if u is not in Y.

For a node t of the tree-decomposition, a partition $R \cup G \cup B$ of Q_t, and an integer s, a set of taxa $Y \subseteq V_t$ is called (t, R, G, B, s)-*feasible*, if all the following conditions hold.

(T1) Each vertex in $Y \setminus Q_t$ is green with respect to Y.

(T2) The vertices $R \subseteq Q_t$ are red with respect to Y.

(T3) The vertices $G \subseteq Q_t$ are green with respect to Y.

(T4) The vertices $B \subseteq Q_t$ are black with respect to Y.

(T5) The size of Y is s.

In entry $\mathrm{DP}[t, A, R, G, B, s]$, we want to store the largest diversity $\mathrm{PD}_{\mathcal{T}}(Y)$ of (t, R, G, B, s)-feasible sets Y. If there is no (t, R, G, B, s)-feasible sets Y, we want $-\infty$ to be stored. Let r be the root of the nice tree-decomposition T. Then, $\mathrm{DP}[r, \emptyset, \emptyset, \emptyset, k]$ stores an optimal diversity. So, we return `yes` if $\mathrm{DP}[r, \emptyset, \emptyset, \emptyset, k] \geq D$ and `no` otherwise.

In the following, any time a non-defined entry $\mathrm{DP}[t, R, G, B, s]$ is called (in particular, if $s < 0$), we take $\mathrm{DP}[t, R, G, B, s]$ to be $-\infty$.

We regard the different types of nodes of a tree-decomposition separately and discuss their correctness.

Leaf Node. For a leaf t of T, the bags Q_t and V_t are empty. We store

$$\mathrm{DP}[t, \emptyset, \emptyset, \emptyset, 0] \quad = \quad 0. \tag{5.9}$$

For all other values, we store $\mathrm{DP}[t, R, G, B, s] = -\infty$.

Recurrence (6.3) is correct by definition.

Introduce Node. Suppose that t is an *introduce node*, that is, t has a single child t' with $Q_t = Q_{t'} \cup \{v\}$. We store $\mathrm{DP}[t, R, G, B, s] = \mathrm{DP}[t', R, G, B \setminus \{v\}, s]$ if $v \in B$. If $v \in G$ but v is not a source in $\mathcal{F}$ and v does not have prey in $R \cup G$, then we store $\mathrm{DP}[t, R, G, B, s] = -\infty$. Otherwise, we store

$$\mathrm{DP}[t, R, G, B, s] \quad = \quad \max_{A \subseteq N_>(v) \cap (R \cup G)} \mathrm{DP}[t', R', G', B, s-1] + \lambda(\rho v). \tag{5.10}$$

Herein, we define R' and G' depending on A to be $R' := (R \setminus (\{v\} \cup N_>(v))) \cup A$ and $G' := (G \cup (N_>(v) \cap (R \cup G))) \setminus (A \cup \{v\})$.

If $v \in B$, then $v \notin Y$ for any (t, R, G, B, s)-feasible set Y. If $v \in G$ but is not a source and has no prey in $R \cup G$, then v is not green with respect to Y for any Y. So, these two cases store the desired value. Towards the correctness of Recurrence (6.4): If $v \in G \cup R$, then we want to select v and therefore we need to add $\lambda(\rho v)$ to the value of $\mathrm{DP}[t, R, G, B, s]$. Further, by the selection of v we know that the predators of v could be green (but maybe are still stored as red) in t, but in t' could be red and therefore we reallocate $N_>(v) \cap (R \cup G)$ and let A be red beforehand.

Forget Node. Suppose that t is a *forget node*, that is, t has a single child t' with $Q_t = Q_{t'} \setminus \{v\}$. We store

$$\mathrm{DP}[t, R, G, B, s] = \max\{\mathrm{DP}[t', R, G \cup \{v\}, B, s]; \mathrm{DP}[t', R, G, B \cup \{v\}, s]\}. \tag{5.11}$$

Recurrence (6.6) follows from the definition that vertices in $v \in V_t \setminus Q_t$, depending on whether they are chosen or not, are either black or green with respect to (t, R, G, B, s)-feasible sets.

	B_1	R_1	G_1
B_2	B	R	G
R_2	R	R	G
G_2	G	G	G

Figure 5.4: This table shows the relationship between the three partitions $R_1 \cup G_1 \cup B_1$, $R_2 \cup G_2 \cup B_2$ and $R \cup G \cup B$ in the case of a join node, when $R_1 \cup G_1 \cup B_1$ and $R_2 \cup G_2 \cup B_2$ are qualified for $R \cup G \cup B$. The table shows which of the sets R, G, or B an element $v \in Q_t$ will be in, depending on its membership in R_1, G_1, B_1, R_2, G_2, and B_2. For example if $v \in R_1$ and $v \in B_2$, then $v \in R$.

Join Node. Suppose that $t \in T$ is a *join node*, that is, t has two children t_1 and t_2 with $Q_t = Q_{t_1} = Q_{t_2}$. We call two partitions $R_1 \cup G_1 \cup B_1$ and $R_2 \cup G_2 \cup B_2$ of Q_t *qualified* for $R \cup G \cup B$ if $R = (R_1 \cup R_2) \setminus (G_1 \cup G_2)$ and $G = G_1 \cup G_2$ (and consequently $B = B_1 \cap B_2$). See Figure 5.4. We store

$$\mathrm{DP}[t, R, G, B, s] = \max_{(\pi_1, \pi_2) \in \mathcal{Q}, s'} \mathrm{DP}[t_1, R_1, G_1, B_1, s'] + \mathrm{DP}[t_2, R_2, G_2, B_2, s - s'], \quad (5.12)$$

where $\mathcal{Q}$ is the set of pairs of partitions $\pi_1 = R_1 \cup G_1 \cup B_1$ and $\pi_2 = R_2 \cup G_2 \cup B_2$ that are qualified for $R \cup G \cup B$, and $s' \in [s]_0$.

By Figure 5.4, we observe that in Recurrence (6.7) we consider the correct combinations of R, G, and B.

Running Time. The table contains $\mathcal{O}(3^{\mathrm{tw}_{\mathcal{F}}} \cdot nk)$ entries, as a tree decomposition contains $\mathcal{O}(n)$ nodes. Each leaf and forget node can be computed in linear time. An introduce node can be computed in $\mathcal{O}(2^{\mathrm{tw}_{\mathcal{F}}} \cdot n)$ time. In a join node, considering only the qualified sets, (π_1, π_2) already define R, G, and B. Thus, all join nodes can be computed in $\mathcal{O}(9^{\mathrm{tw}_{\mathcal{F}}} \cdot nk)$ time, which is also the overall running time. □

5.6.4 Hardness Results

Max Leaf Number. Based on results of Faller et al. [FSW11], in Corollary 5.29 we show that PDD is NP-hard on instances in which the underlying undirected graph of the food-web has a max leaf number of 2.

Corollary 5.29. PDD *is* NP*-hard even if the food-web has max leaf number two.*

Proof. When regarding the reduction of Faller et al. [FSW11, Theorem 5.1.] from Vertex Cover to PDD we observe three things. The vertex x has been added to ensue that y functions as a root in their unrooted definition of the problem.

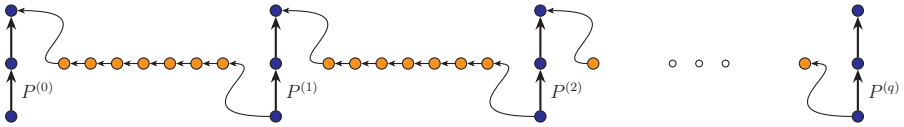


Figure 5.5: This figure shows an example of the food-web $\mathcal{F}'$ we reduce to in Corollary 5.29. The vertices of X are blue and the vertices in Y are orange. Here, (despite $7 < |Y| + 1$) we use $N = 7$.

Therefore, we do not need x for our definition of PDD, leaving paths with a length of 3. Further, some edges in the reduction have a weight of 0 but it would have not caused problems setting them to 1 and multiplying each other edge with a big constant.

Reduction. Let $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D)$ be an instance of PDD in which the set of taxa is X and each connected component of $\mathcal{F}$ is a path with a length of 3. We construct an instance $\mathcal{I}' = (\mathcal{T}', \mathcal{F}', k, D')$ of PDD as follows. Let $P^{(0)}, P^{(1)}, \ldots, P^{(q)}$ be the connected components of $\mathcal{F}$ where $P^{(i)}$ contains the taxa $\{x_{i,0}, x_{i,1}, x_{i,2}\}$ and edges $x_{i,0}x_{i,1}$ and $x_{i,1}x_{i,2}$. Let N and M be constants bigger than $|X| + 1$ and such that $N \cdot q < M$.

Let Y be a set of new taxa $y_{i,j}$ for $i \in [q]$ and $j \in [N]$. Our new set of taxa is $X \cup Y$. Multiply each edge-weight in $\mathcal{T}$ with M and add the taxa Y as children of the root ρ to obtain $\mathcal{T}'$. Set the weight of the edges $\rho y_{i,j}$ to 1 for each $i \in [q]$, and $j \in [N]$. To obtain $\mathcal{F}'$, we add Y as vertices to $\mathcal{F}$ and add edges $y_{i,j}y_{i,j+1}$, $x_{i-1,0}y_{i,1}$ and $y_{i,N}x_{i,2}$ for each $i \in [q]$, and each $j \in [N]$. Figure 5.5 depicts an example of how to create $\mathcal{F}'$. Finally, we set $k' = k$ and set $D' := D \cdot M$.

Correctness. The reduction can be computed in polynomial time. The underlying graph of $\mathcal{F}'$ is a path and therefore has a max leaf number of two. Any solution for $\mathcal{I}$ is also a solution for $\mathcal{I}'$.

Conversely, let $\mathcal{I}'$ be a `yes`-instance of PDD with solution $S \subseteq X \cup Y$. Define $Y^{(i)}$ to be the set of the taxa $y_{i,j}$ for $j \in [N]$. If $x_{i,2}$ is in S but $x_{i,1}$ is not in S then, because S is viable, we know that $Y^{(i)} \subseteq S$. Observe $\mathrm{PD}_{\mathcal{T}}(Y^{(i)}) = N$ and $\mathrm{PD}_{\mathcal{T}}(x_{i,1}) \geq M$. Thus, also $S' := (S \setminus Y^{(i)}) \cup \{x_{i,0}, x_{i,1}\}$ is a solution for $\mathcal{I}'$. Therefore, we assume that if $x_{i,j}$ is in S then also $x_{i,j-1}$ for $j \in \{1, 2\}$. Define sets $S_X = S \cap X$ and $S_Y = S \cap Y$. Then, $\mathrm{PD}_{\mathcal{T}}(S_X)$ is dividable by M and $\mathrm{PD}_{\mathcal{T}}(S_Y) \leq q \cdot N < M$. We conclude $\mathrm{PD}_{\mathcal{T}}(S_X) \geq M \cdot D$ and S_X is a solution for instance $\mathcal{I}$ of PDD. □

We observe that the construction in the previous corollary creates a food-web with an anti-chain of size $2q + 1$. We, therefore, ask whether PDD is still `NP`-hard if

the DAG-width of the food-web is a constant.

Distance to Source-Dominant. In this paragraph, we consider *source-dominant* food-webs. A food-web is source-dominant, if there is a vertex v, which is a source and every other vertex is a predator of v. Observe that if a directed acyclic graph G has a clique graph as underlying undirected graph, then G is source-dominant.

In Theorem 5.6, we showed that S-PDD is FPT when parameterized with distance to cluster. The algorithm can easily be adopted to the distance to stars. This raises the question of whether S-PDD is also FPT with respect to the distance to source-dominant. Here, we shortly want to discuss that such a result is unlikely.

First, if the food-web is a source-dominant, then it is necessary to save the only source, after which all other taxa can be saved without further restrictions. Therefore, after saving the source, we can run the greedy-algorithm for MAX-PD.

Now, let $\mathcal{I} = (\mathcal{T}, \mathcal{F}, k, D)$ be an instance of S-PDD. By Observation 5.3, we can assume that $\mathcal{F}$ has a single source s. For a big integer N, we add N to $\lambda(\rho x)$ for each taxon $x \in X$. Then, we add a new taxon $\star$ and add an edge from $\star$ to each taxon $x \in X \setminus \{s\}$. We finally set $\lambda(\rho\star) := 1$ and $D' := D + N \cdot k$ and obtain an instance $\mathcal{I}'$. By the construction, it is not possible to save $\star$ and so $S \subseteq X$ is a solution for $\mathcal{I}$ if and only if S is a solution for $\mathcal{I}'$. Also, observe that $\mathcal{F}' - \{s\}$ is source-dominant.

We conclude the following.

Proposition 5.30.

(a) PDD *can be solved in polynomial time if the food-web is source-dominant.*

(b) S-PDD *is* NP*-hard on instances in which the food-web has a distance to source-dominant of* 1*.*

5.7 Discussion

In this chapter, we studied the algorithmic complexity of PDD and S-PDD with respect to various parameterizations. PDD is FPT when parameterized with the solution size plus the height of the phylogenetic tree. Consequently, PDD is FPT with respect to D, the threshold of diversity. However, both problems, PDD and S-PDD, are unlikely to admit a kernel of polynomial size. Further, unlike some other problems on maximizing phylogenetic diversity [JS23b, JS24], PDD probably does not admit an FPT-algorithm with respect to $\overline{D}$, the acceptable loss of phylogenetic diversity.

We also considered the structure of the food-web. Among other results, we showed that PDD remains NP-hard even if the food-web is a cluster graph. On the positive side, we proved that PDD is FPT with respect to the number of vertices that need to be removed from the food-web to obtain a co-cluster. We showed further that S-PDD is FPT with respect to the treewidth of the food-web and therefore can be solved in polynomial time if the food-web is a tree.

Unsurprisingly, several interesting questions remain open after our examination of PDD and S-PDD. Arguably the most relevant one is whether PDD is FPT with respect to k, the size of the solution. Also, it remains open whether PDD can be solved in polynomial time if each connected component in the food-web contains at most two vertices.

Clearly, further structural parameterizations can be considered. We only considered structural parameters which consider the underlying graph. But parameters which also consider the orientation of edges, such as the longest anti-chain, could give a better view on the structure of the food-web than parameters which only consider the underlying graph.

Liebermann et al. [LHN05] introduced and analyzed weighted food-webs. Such a weighted model may provide a more realistic view of a species' effect on and interaction with other species [CDRG$^+$18]. Maximizing phylogenetic diversity with respect to a weighted food-web in which one potentially needs to save several prey per predator would be an interesting generalization for our work and has the special case in which one needs to save all prey for each predator.

Chapter 6

Phylogenetic Diversity in Networks

6.1 Introduction

Phylogenetic Diversity (PD) as originally proposed by Faith is defined for phylogenetic trees. Assuming it is not possible to preserve all threatened species, e.g. due to limited resources, we would like to find a subset of species that can be preserved, for which the overall diversity is maximized.

However, when the evolution of the species under interest is also shaped by reticulation events such as hybrid speciation, lateral gene transfer or recombination, then the picture is no longer as rosy as for Max-PD. In reticulation events, a single species may inherit genetic material and, thus, features from multiple direct ancestors and its evolution should be represented by a phylogenetic network [HRS10] rather than a tree. Several ways of extending the notion of PD for phylogenetic networks have been proposed [WF18, BSW22], of which two are called AP-PD$_{\mathcal{N}}$ and Net-PD$_{\mathcal{N}}$.

Unfortunately, unlike in phylogenetic trees, the optimization of phylogenetic diversity on phylogenetic trees is NP-hard for both cases [BSW22]. For this reason, we study the problem from the perspective of parameterized complexity.

Related Work. All-paths phylogenetic diversity as a measure on networks was first introduced in [WF18] under the name 'phylogenetic subnet diversity'. Net-PD$_{\mathcal{N}}$ has been defined in [BSW22].

Phylogenetic diversity forms the basis of the Shapley Value, a measure that describes how much a *single* species contributes to overall biodiversity. The definition of the Shapley Value involves the phylogenetic diversity of every possible subset of species, and so is difficult to calculate directly. However, the Shapley Value is equivalent to the Fair Proportion Index on phylogenetic trees [RM06, FJ15], and it can

be calculated in polynomial time. In the case of phylogenetic networks, it was shown that this result also extends to Shapley Value based on the all-paths phylogenetic diversity measure. This is in contrast to the NP-hardness result—while it is easy to determine the individual species that contributes the most phylogenetic diversity across all sets of species, it is NP-hard to find a *set* of species for which AP-PD$_{\mathcal{N}}$ or Net-PD$_{\mathcal{N}}$ is maximal [BSW22].

The computational complexity of MAPPD was first studied in [BSW22], where the authors showed that the problem is NP-hard and cannot be approximated in polynomial time with approximation ratio better than $1 - \frac{1}{e}$ unless P = NP, but is polynomial-time solvable on level-1-networks. As MAX-NET-PD is a generalization of MAPPD, the hardness result naturally extend. Moreover, MAX-NET-PD remains NP-hard even for the restricted class of phylogenetic networks called *normal* networks [BSW22]. MAX-NET-PD, on a positive side, is FPT when parameterized by the number of reticulations [vIJS+24], that is the number of vertices with at least two incoming edges.

Extensions of phylogenetic diversity to phylogenetic networks have also been considered, both for splits systems [SNM08, MKvH07, MPKvH09].

Our Contribution and Structure of the Chapter. We study several parameterizations of the problems MAX-ALL-PATHS-PD (MAPPD) and MAX-NET-PD. We formally define both problems in the next section. In Section 6.3 we establish an equivalence between the solution size parameterizations of MAPPD and a generalization of SET COVER that we call ITEM-WEIGHTED PARTIAL SET COVER. Consequently, MAPPD is W[2]-hard. We also show that MAPPD is W[1]-hard with respect to the minimum number of taxa that need to go extinct. On the positive side, we show in Section 6.4.1 that MAPPD is fixed-parameter tractable with respect to D, the threshold of phylogenetic diversity and also with respect to the acceptable loss in phylogenetic diversity. Afterward, we turn to structural parameters. In Section 6.4.2 we give single-exponential fixed-parameter algorithms for MAPPD with respect to the number of reticulations in the network, and with respect to the treewidth of the underlying graph of the network. In the case of reticulations, this algorithm is asymptotically tight under SETH.

Finally, in Section 6.6, we study MAX-NET-PD and show that MAX-NET-PD is NP-hard even on networks with a level of 1. This result answers an open question from [BSW22].

Table 6.1: An overview over the parameterized complexity results for MAPPD and MAX-NET-PD.

Parameter	MAX-ALL-PATHS-PD		MAX-NET-PD	
Budget k	W[2]-hard	Thm. 6.1	W[2]-hard	Thm. 6.1
Diversity D	FPT	Cor. 6.6	*open*	
Species-loss $\overline{k}$	W[1]-hard	Thm. 6.2	W[1]-hard	Thm. 6.2
Diversity-loss $\overline{D}$	FPT	Thm. 6.4	*open*	
Number of reticulations $\mathrm{ret}_{\mathcal{N}}$	FPT poly-kernel *open*	Thm. 6.5	FPT poly-kernel *open*	[vIJS+24]
Number of ret.-edges $\text{e-ret}_{\mathcal{N}}$	FPT poly-kernel	Thm. 6.5 Thm. 6.8	FPT poly-kernel *open*	[vIJS+24]
Level	FPT	Thm. 6.6	NP-hard for 1	Thm. 6.9
Treewidth	FPT	Thm. 6.6	NP-hard for 2	Thm. 6.9

6.2 Preliminaries

In this section, we present the formal definition of the problems, and the parameterization. We further start with some preliminary observations.

6.2.1 Phylogenetic Diversity in Phylogenetic Networks

We assume that every edge e in a network $\mathcal{N} = (V, E)$ has an associated *weight* $\lambda(e)$, which is a positive integer.

All-Paths Phylogenetic Diversity. For a set of taxa Y, an edge e is *affected by* Y if $\mathrm{off}(e) \cap Y \neq \emptyset$ and *strictly affected by* Y if $\mathrm{off}(e) \subseteq Y$. The sets T_Y and E_Y are the strictly affected and affected edges by Y, respectively. For a set of taxa Y, the *all-paths phylogenetic diversity AP-PD*$_{\mathcal{N}}(Y)$ *of* Y is

$$\text{AP-PD}_{\mathcal{N}}(Y) := \sum_{e \in E_Y} \lambda(e). \tag{6.1}$$

That is, $\text{AP-PD}_{\mathcal{N}}(Y)$ is the total weight of all edges uv in $\mathcal{T}$ so that there is a path from v to a vertex in Y. If $\mathcal{N}$ is a phylogenetic tree, then this definition coincides with the definition of phylogenetic diversity in Equation (2.1).

Network Diversity. The $\text{Net-PD}_{\mathcal{N}}$-measure of phylogenetic diversity in networks allows the case that reticulations may not inherit all of the features from every par-

ent. This is modeled via an *inheritance proportion* $p(e) \in [0,1]$ on each reticulation edge $e = uv$. Here, $p(e)$ represents the expected proportion of features present in u that are also present in v; or equivalently, $p(e)$ is the probability that a feature in u is inherited by v. We assume these probabilities are distributed independently and identically at random. Non-reticulation edges can be considered as having an inheritance proportion of 1.

For a subset of taxa $Z \subseteq X$, the measure Net-PD$_{\mathcal{N}}(Z)$ represents the expected number of distinct features appearing in taxa in Z [BSW22]. For each edge uv, this measure is obtained by multiplying the number $\lambda(uv)$ of features developed on the branch uv (which is assumed to be proportional to the length of the branch) with the *probability* $\gamma_Z^p(uv)$ that a random feature appearing in v or developed on uv will survive when preserving Z.

Formally, we define $\gamma_Z^p(uv)$ as follows:

Definition 6.1. Given a phylogenetic X-network $\mathcal{N} = (V, E)$ with an edge weight function $\lambda : E \to \mathbb{N}$, inheritance proportions $p : E \to [0,1]$ and a set of taxa $Z \subseteq X$. We define $\gamma_Z^p : E \to [0,1]$ for each edge $uv \in E$ as follows:

- If v is a leaf, then $\gamma_Z^p(uv) := 1$ if $v \in Z$, and $\gamma_Z^p(uv) := 0$, otherwise.
  Intuition: The features of v survive if and only if v is preserved by Z.
- If v is a reticulation with outgoing edge vw, then set $\gamma_Z^p(uv) := p(uv) \cdot \gamma_Z^p(vw)$.
  Intuition: The features of v are a mixture of features of its parents and the features of u have a certain inheritance proportion $p(uv)$ of being included in this mix and, thereby, survive in preserved descendants of x.
- If v is a tree vertex with children $x_1, \ldots, x_\ell$.
  We set $\gamma_Z^p(uv) := 1 - \prod_{i=1}^{\ell}(1 - \gamma_Z^p(vx_i))$. In the special case that v has two children, x_1 and x_2, this is equivalent to $\gamma_Z^p(vx_1) + \gamma_Z^p(vx_2) - \gamma_Z^p(vx_1) \cdot \gamma_Z^p(vx_2)$.
  Intuition: To lose a feature of v, it has to be lost in all children of v, which are assumed to be independent events, since all copies of the feature developed independently.

Further, we only consider values of p on edges incoming to leaves or reticulations, so we may restrict the domain of p to those edges. We now define the measure Net-PD$^{(p)}_{\mathcal{N}}(Z)$ for a subset of taxa Z as follows:

$$\text{Net-PD}^{(p)}_{\mathcal{N}}(Z) := \sum_{e \in E} \lambda(e) \cdot \gamma_Z^p(e). \tag{6.2}$$

We will omit the superscript (p) usually, as we do not operate with several inheritance proportions ps simultaneously.

Observe that $\gamma_Z^p(e)$ and Net-PD$_\mathcal{N}(Z)$ are monotone, that is, $\gamma_{Z'}^p(e) \leq \gamma_Z^p(e)$ and Net-PD$_\mathcal{N}^p(Z') \leq$ Net-PD$_\mathcal{N}^p(Z)$ for all $Z' \subseteq Z \subseteq X$.

Observe that if a vertex has no reticulation descendants, then $\gamma_Z^p(e) = 1$ if off$(e) \cap Z \neq \emptyset$, and otherwise $\gamma_Z^p(e) = 0$. Therefore, Net-PD$_\mathcal{N}$ coincides with AP-PD$_\mathcal{N}$ if $\mathcal{N} = \mathcal{T}$ is a tree.

6.2.2 Problem Definitions and Parameterizations.

This chapter's main object of study are the following two problems, introduced in [WF18, BSW22]:

Max-All-Paths-PD (MapPD)

Input: A phylogenetic X-network $\mathcal{N} = (V, E, \lambda)$ and two integers k and D.
Question: Is there a subset $Z \subseteq X$ with a size of at most k and AP-PD$_\mathcal{N}(Z) \geq D$?

Max-Net-PD

Input: A phylogenetic X-network $\mathcal{N} = (V, E, \lambda)$ with inheritance proportions $p : E \to [0, 1]$, and two integers k and D.
Question: Is there a subset $Z \subseteq X$ with a size of at most k and Net-PD$_\mathcal{N}(Z) \geq D$?

In Section 6.3, we show that there is a strong connection between MapPD and the problem Item-Weighted Partial Set Cover, which we define as follows.

Item-Weighted Partial Set Cover (wpSC)

Input: A universe $\mathcal{U}$, a family $\mathcal{F}$ of subsets over $\mathcal{U}$, an integer weight $\lambda(u)$ for each item $u \in \mathcal{U}$, and two integers k and D.
Question: Are there sets $F_1, \ldots, F_k \in \mathcal{F}$ such that the sum of the weights of the elements in $L := \bigcup_{i=1}^k F_i$ is at least D?

The condition in wpSC can be formalized as follows: $\sum_{u \in L} \lambda(u) \geq D$. Observe, Set Cover is a special case of wpSC with $D = \lambda_\Sigma(\mathcal{U})$.

Parameters. We examine MapPD within the framework of parameterized complexity. In addition to the parameters k and D, that are the number of saved taxa and the threshold of preserved phylogenetic diversity, we also study the dual parameters which are the minimum number of species that will go extinct $\overline{k} := |X| - k$ and the acceptable loss of phylogenetic diversity $\overline{D} :=$ AP-PD$_\mathcal{N}(X) - D$. By ret$_\mathcal{N}$, we denote the number of reticulations in $\mathcal{N}$, and by tw$_\mathcal{N}$ we denote the treewidth

of the underlying undirected graph of $\mathcal{N}$. By e-ret$_\mathcal{N}$, we denote the number of reticulation-edges that need to be removed such that $\mathcal{N}$ is a tree. More formally, e-ret$_\mathcal{N} := \sum_{r \in R}(\deg^-(r)-1) = |E|-|V|+1$, where R is the set of reticulations of $\mathcal{N}$ and $\deg^-(v)$ is the in-degree of a vertex v. By $\max_\lambda$, we denote the biggest weight of an edge.

Further, we show that Max-Net-PD is NP-hard on level-1-networks. Recall that the level of a network $\mathcal{N}$ is the maximum reticulation number of a subgraph $\mathcal{N}[V']$ for some $V' \subseteq V(\mathcal{N})$ where we require that the underlying undirected graph of $\mathcal{N}[V']$ is biconnected.

In this section, we use the convention that n is the number of vertices in the network and m is the number of edges in the network. Unlike in trees, we can not assume $n \in \mathcal{O}(|X|)$.

Binary Networks. A phylogenetic X-network is called *binary* if the root has a degree of 2, each leaf has a degree of 1, and each other vertex has a degree of 3. In this chapter, we do not assume networks to be binary; in particular, we allow tree vertices to have an in-degree and an out-degree of 1. Bordewich et al. [BSW22] required that the given network $\mathcal{N}$ is binary. In the following, we show that algorithmically, there is hardly any difference for MapPD.

Lemma 6.2. *When given an instance $(\mathcal{N}, k, D)$ of* MapPD*, in $\mathcal{O}(|E|)$ time an equivalent instance $(\mathcal{N}', k', D')$ of* MapPD *with a binary network $\mathcal{N}'$, $\mathrm{tw}_{\mathcal{N}'} = \mathrm{tw}_\mathcal{N}$, and $|E'| \leq 2|E|$ can be computed.*

Proof. Algorithm. Let $\mathcal{I} := (\mathcal{N} := (V, E, \lambda), k, D)$ be an instance of MapPD. We set $k' := k$ and $D' := D \cdot (|E|+1)$. Iterate over E and set $\lambda'(e) = \lambda(e) \cdot (|E|+1)$. Iterate over the vertices v and compute the degree $\deg(v)$ of v. If $\deg(v) \in \{1, 3\}$ then continue with the next vertex. If $\deg(v) = 2$ and the edges $e_1 = uv$ and $e_2 = vw$ are incident with v, then delete e_1 and e_2 and v from $\mathcal{N}$ and insert the edge uw with weight $\lambda'(uv) + \lambda'(vw)$.

If v is a reticulation with $\deg(v) > 3$ and the edges $e_i = u_i v$ for $i \in [\deg(v) - 1]$ and $\hat{e} = vw$ are incident with v. Then, replace v with vertices $a_1, \ldots, a_{\deg(v)-2}$ and add edges u_1a_1, $a_{\deg(v)-2}w$, $a_{i-1}a_i$, and u_ia_{i-1} for $i \in \{2, 3, \ldots, \deg(v) - 1\}$, where the weight of the outgoing edge of u_i is $\lambda'(u_iv)$ and all other new edges have a weight of 1. Otherwise, if $\deg(v) > 3$ and v is a tree vertex or the root, proceed analogously with inverted edges. Figure 6.1 depicts this procedure for a reticulation.

Correctness. The network $\mathcal{N}'$ is clearly binary. The treewidth is not effected by the operations we did. We add $\mathcal{O}(\deg(v))$ edges for each vertex v to $\mathcal{N}'$ such that there are at most $|E| + \sum_{v \in V} \deg(v) = 2|E|$ edges in $\mathcal{N}'$.

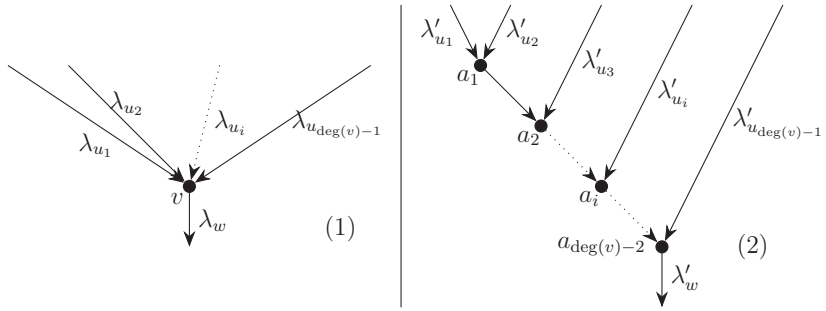


Figure 6.1: An example of the construction of $\mathcal{N}'$ in the case of a reticulation with multiple incoming edges. We use the weights $\lambda'_p := \lambda_p \cdot (|E| + 1)$. Unlabeled edges have a weight of 1.

It remains to show the correctness of the algorithm. Let Y be a solution for $\mathcal{I}$ and let $e_1, \ldots, e_\ell \in E$ be the edges that are affected by Y, of which $e_{t+1}, \ldots, e_\ell$ are incident with a vertex with a degree of 2. Thus, $e_1, \ldots, e_t$ are also edges in $\mathcal{N}'$ and there are edges $\hat{e}_1, \ldots, \hat{e}_p$ that are affected by Y with $(|E| + 1) \cdot \sum_{i=t+1}^{\ell} \lambda(e_i) = \sum_{i=1}^{p} \lambda(\hat{e}_i)$. Consequently,

$$\begin{aligned} D' &\leq (|E| + 1) \cdot \text{AP-PD}_{\mathcal{N}}(Y) \\ &= (|E| + 1) \cdot \sum_{i=1}^{\ell} \lambda(e_i) \\ &= \sum_{i=1}^{t} \lambda'(e_i) + \sum_{i=1}^{p} \lambda(\hat{e}_i) \\ &\leq \text{AP-PD}_{\mathcal{N}'}(Y). \end{aligned}$$

Analogously, let Y be a solution for $\mathcal{I}'$ and let $e_1, \ldots, e_t$ be the edges with a weight of higher than 1. We can find edges $\hat{e}_1, \ldots, \hat{e}_p \in E$ that are affected by Y with $\sum_{i=1}^{t} \lambda'(e_i) = (|E| + 1) \cdot \sum_{i=1}^{p} \lambda(e_i)$. Observe that the edges with a weight of 1 are between the vertices a_i and a_{i+1}, such that there are $\mathcal{O}(|E|)$ of these edges. Thus, $D' \leq \sum_{i=1}^{t} \lambda'(e_i) = (|E| + 1) \cdot \sum_{i=1}^{p} \lambda(e_i) = \text{AP-PD}_{\mathcal{N}}(Y)$. Hence, Y is also a solution for instance $\mathcal{I}$.

Running Time. For each vertex v, we perform at most $\mathcal{O}(\deg(v))$ operations. Therefore, as $|E| = \sum_{v \in V} \deg(v)$, the overall running time is $\mathcal{O}(|E|)$. □

6.3 MapPD and Item-Weighted Partial Set Cover

In this section, we showcase a relationship between MapPD and wpSC by presenting reductions in both directions. Bordewich et al. already presented a similar reduction from Set Cover to MapPD [BSW22].

Theorem 6.1. *For every instance $\mathcal{I} = (\mathcal{U}, \mathcal{F}, \lambda, k, D)$ of* wpSC*,*

(a) an equivalent instance $\mathcal{I}' = (\mathcal{N}, k', D')$ of MapPD *with $|X| = \mathrm{ret}_{\mathcal{N}} = |\mathcal{F}|$ and $k' = k$ can be computed in time polynomial in $|\mathcal{U}| + |\mathcal{F}|$;*

(b) an equivalent instance $\mathcal{I}'_2 = (\mathcal{N} = (V, E, \lambda'), k', D')$ of MapPD *in which $k' = k$ and each edge has a weights of 1 can be computed in time polynomial in $|\mathcal{U}| + |\mathcal{F}| + \max_\lambda$.*

This theorem has several applications for the complexity of MapPD. Because Set Cover is W[2]-hard with respect to the size of the solution k [DF13], MapPD is as well. This is in contrast to the fact that MapPD can be solved in polynomial time when the network does not have reticulations and, therefore, is a phylogenetic tree [Ste05].

Corollary 6.3. MapPD *is* W[2]*-hard when parameterized with k, even if* $\max_\lambda = 1$.

Recall that in Red-Blue Non-Blocker an undirected bipartite graph G with vertex bipartition $V(G) = V_r \cup V_b$ and an integer k are given. The question is whether there is a set $S \subseteq V_r$ of size at least k such that each vertex v of V_b has a neighbor in $V_r \setminus S$. There is a standard reduction from Red-Blue Non-Blocker to Set Cover: Let V_b be the universe. For each vertex $v \in V_r$ add a set $F_v := N(v)$ to $\mathcal{F}$ and finally set $k' := |V_r| - k$. Red-Blue Non-Blocker is W[1]-hard when parameterized by the size of the solution [DF95b]. Hence, Set Cover is W[1]-hard with respect to $|\mathcal{F}| - k$, and with Theorem 6.1, we conclude as follows.

Theorem 6.2. MapPD *is* W[1]*-hard when parameterized with $\overline{k} = |X| - k$.*

MapPD can be solved in $\mathcal{O}^*(2^{|X|})$ with a brute force algorithm that tries every possible subset of species as a solution. In Theorem 6.5, we prove that MapPD can be solved in $\mathcal{O}^*(2^{\mathrm{ret}_{\mathcal{N}}})$ time. In order to prove that these algorithms can not be improved significantly, we apply the well-established `Strong Exponential Time Hypothesis` (`SETH`).

Unless `SETH` fails, Set Cover can not be solved in $O^*(2^{\epsilon \cdot |F|})$ time for any $\epsilon < 1$ [CDL+16, Lin19]. Thus, Theorem 6.1 shows that under `SETH`, not a lot of hope

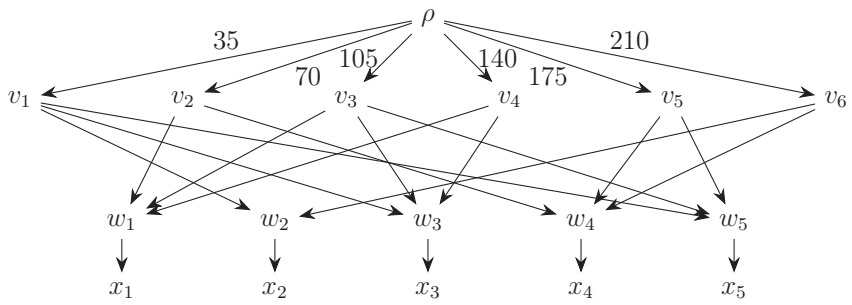


Figure 6.2: This figure depicts the network $\mathcal{N}$ that we reduce to from the instance $(\mathcal{U} := \{u_1, \ldots, u_6\}, \mathcal{F} := \{F_1, \ldots, F_5\}, \lambda, k, D)$ of wpSC with $\lambda(u_i) = i$, $F_1 := \{u_2, u_3, u_4\}$, $F_2 := \{u_1, u_6\}$, $F_3 := \{u_1, u_3, u_4\}$, $F_4 := \{u_2, u_5, u_6\}$, $F_5 := \{u_1, u_3, u_5\}$. Unlabeled edges have a weight of 1. Here $n = 6, m = 5$ and $Q = 35$. The value of k' would be k and D' would be $35D + 1$.

remains to find faster algorithms for MapPD than these two algorithms. Thus, these two algorithms, with respect to the number of taxa $|X|$ and reticulations $\text{ret}_{\mathcal{N}}$, for MapPD are tight with the lower bounds.

Corollary 6.4. *Unless* SETH *fails,* MapPD *can not be solved in* $2^{\epsilon \cdot |X|} \cdot \text{poly}(|\mathcal{I}|)$ *time or in* $2^{\epsilon \cdot \text{ret}_{\mathcal{N}}} \cdot \text{poly}(|\mathcal{I}|)$ *time for any* $\epsilon < 1$.

So now, without further ado, we prove Theorem 6.1.

Proof of Theorem 6.1. Reduction. Let $\mathcal{I} = (\mathcal{U}, \mathcal{F}, k, D)$ be an instance of wpSC. Let $\mathcal{U}$ consist of the items $u_1, \ldots, u_n$ and let $\mathcal{F}$ contain the sets $F_1, \ldots, F_m$. We may assume that for each u_i there is a set F_j which contains u_i. We define an instance $\mathcal{I}' = (\mathcal{N}, k, D')$ of MapPD as follows. Let k stay unchanged and define $D' := D \cdot Q + 1$ for $Q := m(n+1)$. We define a network $\mathcal{N}$ with leaves $x_1, \ldots, x_m$, and further vertices $\rho, v_1, \ldots, v_n, w_1, \ldots, w_m$.

Let the set of edges consist of the edges ρv_i for $i \in [n]$, $w_j x_j$ for $j \in [m]$, and let $v_i w_j$ be an edge if and only if $u_i \in F_j$. We define the weight of ρv_i to be $\lambda(u_i) \cdot Q$ for each $i \in [n]$ and 1 for each other edge. Figure 6.2 depicts an example of this reduction.

This completes the construction of instance $\mathcal{I}'$ in case (a) of the theorem. We now describe how to construct an instance $\mathcal{I}'_2$ from $\mathcal{I}'$ in which the maximum weight of an edge is 1, completing the construction for case (b). For each edge $e = \rho v_i$ with $w(e) > 1$, make $\lambda(e) - 1$ subdivisions and attach a new leaf as the child of each subdividing vertex. We call these newly added leaves *false leaves*, in contrast to the other leaves of $\mathcal{N}$, that we call *true leaves*.

Correctness. The instance $\mathcal{I}'$ is computed in time polynomial in $|\mathcal{U}| + |\mathcal{F}|$ and the instance $\mathcal{I}'_2$ is computed in time polynomial in $|\mathcal{U}| + |\mathcal{F}| + \max_\lambda$. Clearly, in $\mathcal{I}'$ we observe $k' = k$ and $|X| = \text{ret}_\mathcal{N} = |\mathcal{F}|$; and in $\mathcal{I}'_2$ we observe $k' = k$ and $\max_{\omega'} = 1$. It remains to show the equivalence of the instances.

Without loss of generality, let $S := \{F_1, \ldots, F_\ell\}$ with $\ell \leq k$ be a solution for the instance $\mathcal{I}$ of WPSC that covers the items $u_1, \ldots, u_p$. We show that $Y := \{x_1, \ldots, x_\ell\}$ is a solution for the instances $\mathcal{I}'$ and $\mathcal{I}'_2$ of MAPPD. Clearly, the size of Y is at most k. Now, consider the phylogenetic diversity of Y in the network of $\mathcal{I}'$. Let $\hat{E}$ be the edges in $\mathcal{N}$ between two vertices of $v_1, \ldots, v_p, w_1, \ldots, w_\ell$. Then,

$$\begin{aligned}\text{AP-PD}_\mathcal{N}(Y) &= \sum_{i=1}^{\ell} \lambda'(w_i x_i) + \sum_{e \in \hat{E}} \lambda'(e) + \sum_{i=1}^{p} \lambda'(\rho v_i) \geq \lambda'(w_1 x_1) + \sum_{i=1}^{p} \lambda'(\rho v_i) \\ &= 1 + \sum_{i=1}^{p} \lambda(u_i) \cdot Q = 1 + Q \cdot \sum_{i=1}^{p} \lambda(u_i) \geq 1 + QD = D'.\end{aligned}$$

Here, the first inequality holds because the sets in S cover the items $u_1, \ldots, u_p$ and the last inequality holds because S is a solution. It is easy to see that the phylogenetic diversity of the set Y in the network of $\mathcal{I}'_2$ is identical. Hence, Y is a solution of $\mathcal{I}'$ and $\mathcal{I}'_2$.

Without loss of generality, let $Y := \{x_1, \ldots, x_\ell\}$ with $\ell \leq k$ be a solution for the instance $\mathcal{I}'$ of MAPPD. We show that $S := \{F_1, \ldots, F_\ell\}$ is a solution of WPSC. Clearly, the size of S is at most k. Without loss of generality, let $v_1, \ldots, v_p$ be the ancestors of Y that are children of ρ. Thus, there is a path from v_i to a taxon $x_j \in Y$, for each $i \in [p]$. By the construction of $\mathcal{I}'$, we conclude $u_i \in F_j$. Again, letting $\hat{E}$ be the edges between two vertices of $v_1, \ldots, v_p, w_1, \ldots, w_\ell$, we have

$$D' \leq \text{AP-PD}_\mathcal{N}(Y) = \sum_{i=1}^{\ell} \lambda'(w_i x_i) + \sum_{e \in \hat{E}} \lambda'(e) + \sum_{i=1}^{p} \lambda'(\rho v_i) \leq m + nm + \sum_{i=1}^{p} \lambda'(\rho v_i).$$

Consequently, $\sum_{i=1}^{p} Q \cdot \lambda(u_i) = \sum_{i=1}^{p} \lambda'(\rho v_i) \geq D' - Q = Q(D-1) + 1$. We conclude that $\sum_{i=1}^{p} \lambda(u_i) \geq D - 1 + 1/Q$. Since the weights of u_i are integers, it follows that $\sum_{i=1}^{p} \lambda(u_i) \geq D$.

It remains to show that for each solution Y of instance $\mathcal{I}'_2$, an equivalent solution of $\mathcal{I}'$ exists. If Y does not contain false leaves, then as previously observed, the phylogenetic diversity of Y is the same in $\mathcal{I}'$ as in $\mathcal{I}'_2$. Assume now otherwise and let z be a false leaf in Y and let p_z be the parent of z. We consider two different cases; That p_z has an offspring in $Y \setminus \{z\}$, or not. In the former case, p_z has an offspring in $Y \setminus \{z\}$,

we observe AP-PD$_{\mathcal{N}}(Y)$ = AP-PD$_{\mathcal{N}}(Y \setminus \{z\})$ + $\lambda(p_z z)$ = AP-PD$_{\mathcal{N}}(Y \setminus \{z\})$ + 1. Consequently, we can replace z with any true leaf that is not yet in Y to obtain another solution of $\mathcal{I}_2'$. In the second case, let the true leaf x_i be an offspring of p_z. Because of the assumption, $x_i \notin Y$.

Then, AP-PD$_{\mathcal{N}}(Y)-\lambda(p_z z) \leq$ AP-PD$_{\mathcal{N}}((Y\setminus\{z\})\cup\{x_i\})-\lambda(w_i x_i)$. Consequently, $Y \setminus \{z\} \cup \{x_i\}$ is also a solution. Therefore, we can remove all false leaves and then we are done. □

In the proof of Theorem 6.1, we can see that in the root ρ, we model an operation that ensures that at least D of the children of ρ are selected and further, these tree vertices ensure that at least one of the reticulations below them are selected. It might appear that by adding more layers of reticulations and tree vertices to the construction of $\mathcal{N}$, one could reduce from problems even more complex than WPSC, and thereby show that MAPPD has an an even higher position in the W-hierarchy. This, however, is unlikely, because of the following reduction to WPSC.

Theorem 6.3. *For every instance $\mathcal{I} = (\mathcal{N}, k, D)$ of* MAPPD*, we can compute an equivalent instance $(\mathcal{U}, \mathcal{F}, \lambda, k', D')$ of* WPSC *with $k' = k$, $D' = D$ and $\max_{\lambda'} = \max_{\lambda}$ in time polynomial in $|\mathcal{I}|$.*

Proof. Reduction. Let $\mathcal{I} = (\mathcal{N}, k, D)$ be an instance of MAPPD. We define an instance $\mathcal{I}' = (\mathcal{U}, \mathcal{F}, \lambda', k, D)$ of WPSC as follows. Let k and D stay unchanged. For each edge e of $\mathcal{N}$, define an item u_e with a weight of $\lambda'(u_e) = \lambda(e)$ and let $\mathcal{U}$ be the set of these u_e. For each taxon x, define a set F_x which contains item u_e if and only if e is affected by $\{x\}$. Let $\mathcal{F}$ be the family of these sets.

Correctness. Clearly, the reduction is computed in polynomial time. We show the equivalence of the two instances. Let Y be a solution for the instance $\mathcal{I}$ of MAPPD. Without loss of generality, assume $Y = \{x_1, \ldots, x_\ell\}$ with $\ell \leq k$. We show that $F_1, \ldots, F_\ell$ is a solution for instance $\mathcal{I}'$ of WPSC. We know by definition that $\ell \leq k$. Let E_Y be the edges affected by Y. Observe that e is in E_Y if and only if u_e is in $F^+ := \bigcup_{i=1}^{\ell} F_i$. Then, $D \leq$ AP-PD$_{\mathcal{N}}(Y) = \sum_{e\in E_Y} \lambda(e) = \sum_{u_e \in F^+} \lambda'(u_e)$. Hence, $F_1, \ldots, F_\ell$ is a solution for instance $\mathcal{I}'$ of WPSC.

Now, without loss of generality, let $F_1, \ldots, F_\ell$ be a solution for instance $\mathcal{I}'$ of WPSC. Let $u_{e_1}, \ldots, u_{e_p}$ be the items in the union of $F_1, \ldots, F_\ell$. By the construction, the edges $e_1, \ldots, e_p$ are affected by $Y = \{x_1, \ldots, x_\ell\}$. Then,

$$\text{AP-PD}_{\mathcal{N}}(Y) \geq \sum_{i=1}^{p} \lambda(e_i) = \sum_{i=1}^{p} \lambda'(u_{e_i}) \geq D.$$

The size of Y is at most k. Hence, Y is a solution for $\mathcal{I}$ of MAPPD. □

To the best of our knowledge, it is unknown if WPSC is W[2]-complete, like SET COVER. Nevertheless, we obtain the following connection between WPSC and MAPPD.

Corollary 6.5. MAPPD *is* W[i]*-complete with respect to* k *if and only if* WPSC *is* W[i]*-complete with respect to* k*.*

6.4 Fixed-Parameter Tractability of MapPD

6.4.1 Preserved and Lost Diversity

In this subsection, we show that MAPPD is FPT with respect to D, the threshold of phylogenetic diversity, and $\overline{D} := \text{AP-PD}_{\mathcal{N}}(X) - D$, the acceptable loss of phylogenetic diversity.

Let $\mathcal{I}$ be an instance of MAPPD. If there is an edge e with $\lambda(e) \geq D$ and $k \geq 1$, then for each offspring x of e we have $\text{AP-PD}_{\mathcal{N}}(\{x\}) \geq \lambda(e) \geq D$, and so $\{x\}$ is a solution for $\mathcal{I}$. So, we may assume that $\max_{\lambda} < D$. Therefore, each edge e can be subdivided $\lambda(e) - 1$ times in $\mathcal{O}(D \cdot m)$ time such that $\lambda'(e) = 1$ for each edge e of the new network $\mathcal{N}'$. Bläser showed that WPSC can be solved in $\mathcal{O}^*(2^{\mathcal{O}(D)})$ time when $\lambda(u) = 1$ for each item $u \in \mathcal{U}$ [Blä03]. Consequently, with Theorem 6.3 and the result from Bläser we conclude the following.

Corollary 6.6. MAPPD *can be solved in* $\mathcal{O}^*(2^{\mathcal{O}(D)})$ *time.*

As SET COVER is a special case of WPSC with $D = \sum_{u \in \mathcal{U}} \lambda(u)$, WPSC is para-NP-hard with respect to the dual $\sum_{u \in \mathcal{U}} \lambda(u) - D$. Observe $\overline{D} > \overline{k} := |X| - k$. By contrast, we show in the following that MAPPD is FPT with respect to $\overline{D}$. More precisely, we show the following.

Theorem 6.4. MAPPD *can be solved in* $\mathcal{O}(2^{\overline{D}+\overline{k}+o(\overline{D})} \cdot n \log n)$ *time.*

To this end, we use the technique of color coding. Recall that $\text{off}(e) = \text{off}(w)$ for each edge $e = vw$. We define the following auxiliary problem, in which we assign a *color*, red or green, to each taxon. A set $Y \subseteq X$ is *color-fitting* if each taxon $x \in Y$ is red and for each vertex $v \in V(\mathcal{N})$, at least one of the following conditions is satisfied:

- v has a green offspring,
- all offspring of v are in Y, or
- all offspring of v are in $X \setminus Y$.

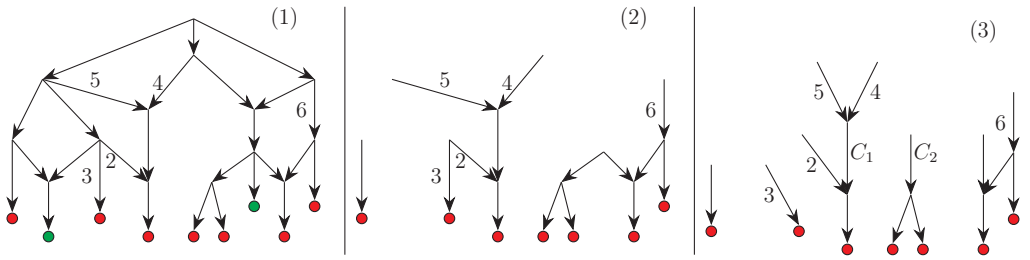


Figure 6.3: In this figure, an example for the transformation in the proof of Lemma 6.7 is given. A hypothetical network $\mathcal{N}$ with a coloring is given in (1). In (2), the graphs G, and in (3) the graph G' is depicted. Some edges are labeled (to increase readability). Unlabeled edges have weight 1. The connected components C_1 and C_2 have weight 13 and 3 and value 1 and 2, respectively.

We define the colored version of MAPPD as follows.

COLORED-MAX-ALL-PATHS-PD (COLORED-MAPPD)

Input: A phylogenetic X-network $\mathcal{N}$, integers k and D, and a coloring on the taxa $c : X \to \{\text{red}, \text{green}\}$.

Question: Is there a subset $S \subseteq X$ of taxa such that $|S| \leq k$, AP-PD$_{\mathcal{N}}(S) \geq D$, and $X \setminus S$ is color-fitting?

Lemma 6.7. COLORED-MAPPD *can be solved in* $\mathcal{O}(\overline{D} \cdot m)$ *time.*

Proof. Algorithm. Let $\mathcal{I} := (\mathcal{N} := (V, E, \lambda), k, D, c)$ be an instance of COLORED-MAPPD. Delete all edges uv for which v has a green offspring, and then delete all isolated vertices. For any vertex u with an in-degree of 0 with children $v_1, \ldots, v_q$, replace u with q vertices $u_1, \ldots, u_q$ and add an edge $u_i v_i$ with a weight of $\lambda(uv_i)$, for each $i \in [q]$. Each in-degree-0 vertex has one child, now. Let G' be the resulting graph. An example of this transformation is depicted in Figure 6.3.

Compute the set of connected components of the underlying graph of G'. For connected components $C = (V_C, E_C)$, proceed as follows. Define an item I_C with a *weight* of $\lambda(E_C)$ and a *value* of $|Y_C|$, where Y_C is the set of taxa in V_C.

Let M be the set of these items. Now, return `yes` if there is a subset of items in M whose total weight is at most $\overline{D}$ and whose total value is at least $\overline{k} = |X| - k$, and `no`, otherwise. Here, $\overline{k}$ and $\overline{D}$ are taken from the original network $\mathcal{N}$. Observe that this can be determined by solving an instance of KNAPSACK with a set of items M, budget $\overline{D}$, and target value $\overline{k}$. This can be done in $\mathcal{O}(\overline{D} \cdot |M|) \in \mathcal{O}(\overline{D} \cdot |X|)$ time [Wei66, GRR19].

Correctness. We show first that if $\mathcal{I}$ is a `yes`-instance of COLORED-MAPPD, then the algorithm return `yes` and secondly we show the converse.

Assume that $\mathcal{I}$ is a `yes`-instance of COLORED-MAPPD. Thus, there is a set $S \subseteq X$ of size at most k with AP-PD$_{\mathcal{N}}(S) \geq D$ and $X \setminus S$ is color-fitting. We claim that $X \setminus S$ is also color-fitting in G'. Indeed, suppose for a contradiction that this is not the case. Then, there exists some vertex v in G' that v has an offspring in $X \setminus S$ and an offspring in S. Let v be a lowest vertex satisfying this condition. Then, v does not have an in-degree of 0 in G', as in this case v has a unique child with the same offspring. Furthermore, v does not have any green offspring in $\mathcal{N}$, as the incoming edges of v were not deleted in the construction of G'. Then, in fact, v has the same offspring in $\mathcal{N}$ as in G', implying that $X \setminus S$ is not color-fitting.

Since $X \setminus S$ is color-fitting in G', and G' has no green taxa, every vertex v in G satisfies $\mathrm{off}(v) \subseteq X \setminus S$ or $\mathrm{off}(v) \subseteq S$. Each edge e in $\mathcal{N}$ that is not affected by S is strictly affected by $X \setminus S$. Because S is a solution we conclude each taxon in $X \setminus S$ is red and so e is still an edge in G'. Let C_e be the connected component of the underlying undirected graph of G' that contains e. Observe that every edge in C_e is also strictly affected by $X \setminus S$—indeed, for any edge uv in C, u has all its offspring in $X \setminus S$ if and only if v has all its offspring in $X \setminus S$. Thus, C_e satisfies the conditions to be in M for each edge e that is not affected by S. Let $C_1, \ldots, C_t$ be the unique connected components that contain the edges that are not affected by S. We now conclude that $\lambda(C_1 \cup \cdots \cup C_t) \leq \overline{D}$ and $C_1 \cup \cdots \cup C_t$ contain the leaves $X \setminus S$, which are at least $\overline{k}$. Hence, $I_{C_1}, \ldots, I_{C_t}$ is a solution for the KNAPSACK-instance and the algorithm returns `yes`.

For the converse, assume that the algorithm returns `yes` and let $I_{C_1}, \ldots, I_{C_t}$ be a solution for the KNAPSACK-instance. Let Y_i be the set of taxa of C_i and define the set $Y := \bigcup_{i=1}^{t} Y_i$. We prove that $S := X \setminus Y$ is a solution for the instance $\mathcal{I}$ of COLORED-MAPPD. By the construction, we conclude that Y_i are colored red and Y_i and Y_j are disjoint for any $i \neq j$. Each vertex $v \in \mathcal{N}$ that has some but not all offspring in Y_i has at least one green offspring. Consequently, Y is color-fitting. Further, for any edge $e = uv$ that is strictly affected by Y in $\mathcal{N}$, we have that v has no green offspring, and therefore e is not deleted in the construction of G'. Moreover, as v has offspring in Y_i for some $i \in [t]$, we conclude that e is in C_i. Because $\sum_{i=1}^{t} \lambda(E_{C_i}) \leq \overline{D}$, we conclude that the phylogenetic diversity of S is AP-PD$_{\mathcal{N}}(S) =$ AP-PD$_{\mathcal{N}}(X) - \sum_{i=1}^{t} \lambda(E_{C_i}) \geq$ AP-PD$_{\mathcal{N}}(X) - \overline{D} = D$. Likewise, because $\sum_{i=1}^{t} |Y_i| \geq \overline{k}$, we conclude $|S| = |X| - \sum_{i=1}^{t} |Y_i| \leq |X| - \overline{k} = k$.

Running Time. The graph G' can be computed from $\mathcal{N}$ in $\mathcal{O}(m)$ time. The connected components of the underlying graph of G' can be computed in $\mathcal{O}(m)$ time as well. For each connected component $C = (V_C, E_C)$ the item I_C of N is computed

in $\mathcal{O}(|E_C|)$ time. Consequently, we can compute N in $\mathcal{O}(m)$ time. As the instance of KNAPSACK can be solved in $\mathcal{O}(\overline{D} \cdot |X|)$ time [Wei66, GRR19], we have an overall running time of $\mathcal{O}(m + \overline{D} \cdot |X|) \in \mathcal{O}(\overline{D} \cdot m)$. □

To show that MAPPD is FPT with respect to $\overline{D}$, we present an reduction from MAPPD to COLORED-MAPPD using standard color coding techniques. In particular, we show that there exists a family $\mathcal{F}$ of 2-colorings $c : E \to \{\text{red}, \text{green}\}$, with $|\mathcal{F}|$ bounded by a function of $\overline{D}$ times a polynomial in n, such that $(\mathcal{N}, k, D)$ is a yes-instance of MAPPD if and only if $(\mathcal{N}, k, D, c)$ is a yes-instance of COLORED-MAPPD for some $c \in \mathcal{F}$.

Recall that an (n, k)-universal set is a family $\mathcal{U}$ of subsets of $[n]$ such that for any $S \subseteq [n]$ of size k, $\{A \cap S \mid A \in \mathcal{U}\}$ contains all 2^k subsets of S. For any $n, k \geq 1$, one can construct an (n, k)-universal set of size $2^k k^{\mathcal{O}(\log k)} \log n$ in time $2^k k^{\mathcal{O}(\log k)} n \log n$ [NSS95].

Proof of Theorem 6.4. Algorithm. Let $\mathcal{I} := (\mathcal{N} := (V, E, \lambda), k, D)$ be an instance of MAPPD. Arbitrarily order the taxa $x_1, \ldots, x_n$. Construct an $(n, \overline{k} + \overline{D})$-universal set $\mathcal{U}$.

Now, for each $A \in \mathcal{U}$, construct a 2-coloring $c_A : X \to \{\text{red}, \text{green}\}$ where x_i is colored green if and only if $i \in A$, and solve COLORED-MAPPD on $(\mathcal{N}, k, D, c_A)$. Return yes, if $(\mathcal{N}, k, D, c_A)$ is a yes-instance for some $A \in \mathcal{U}$. Otherwise, return no.

Correctness. First observe that if $(\mathcal{N}, k, D, c)$ is a yes-instance of COLORED-MAPPD for any coloring $c : X \to \{\text{red}, \text{green}\}$, then $(\mathcal{N}, k, D)$ is also a yes-instance of MAPPD.

Now, suppose $(\mathcal{N}, k, D)$ is a yes-instance. Let $S \subseteq X$ be a subset of taxa of size at most k and with AP-PD$_{\mathcal{N}}(S) \geq D$. If necessary, add taxa to S until $|S| = k$. Consequently, $X \setminus S$ has a size of $\overline{k}$. Let V_Y be the set of vertices u of $\mathcal{N}$ which have an offspring x_u in S and have a child v with $\text{off}(v) \subseteq X \setminus S$. Define $Y := \{x_u \mid u \in V_Y\}$. Observe that if we can define a coloring which colors the taxa in $X \setminus S$ in red and the taxa in Y in green, then $X \setminus S$ would be color-fitting.

Define an operation $\texttt{ind} : 2^X \to 2^{[n]}$ by $\texttt{ind}(X') := \{i \mid x_i \in X'\}$ for sets $X' \subseteq X$.

For each $u \in V_Y$ and each child v of u with $\text{off}(v) \subseteq X \setminus S$, the edge uv is strictly affected by $X \setminus S$. We conclude $|Y| \leq |V_Y| \leq \sum_{u \in V_Y} \lambda(uv) \leq \overline{D}$. Therefore, $Z := Y \cup (X \setminus S) \subseteq X$ is a set of size at most $\overline{k} + \overline{D}$. If necessary, add taxa until Z has a size of $\overline{D} + \overline{k}$. Consequently, there is a set $A \in \mathcal{U}$ with $A \cap \texttt{ind}(Z) = \texttt{ind}(Y)$. So, S is a solution for the instance $(\mathcal{N}, k, D, c_A)$ of COLORED-MAPPD.

Running Time. The construction of $\mathcal{U}$ takes $2^{\overline{D}+\overline{k}+\mathcal{O}(\log^2(\overline{D}))} n \log n$ time, and for each of the $2^{\overline{D}+\overline{k}+\mathcal{O}(\log^2(\overline{D}))} \log n$ sets in $\mathcal{U}$ we solve an instance of COLORED-MAPPD. This

can be done in $\mathcal{O}(\overline{D} \cdot m)$ time by Lemma 6.7.

The overall running time is $\mathcal{O}(2^{\overline{D}+\bar{k}+o(\overline{D})} \cdot n \log n)$. □

6.4.2 Proximity to Trees

MAPPD can be solved in polynomial time with Faith's Greedy-Algorithm, if the given network is a tree [Fai92, Ste05]. Therefore, in this subsection, we examine MAPPD with respect to two parameters that classify the network's proximity to a tree, the number of reticulations $\text{ret}_{\mathcal{N}}$ and the smaller parameter treewidth $\text{tw}_{\mathcal{N}}$.

Theorem 6.5. MAPPD *can be solved in* $\mathcal{O}(2^{\text{ret}_{\mathcal{N}}} \cdot k \cdot m)$ *time.*

Observe that by Corollary 6.4, MAPPD can not be solved in $O^*(2^{\epsilon \cdot \text{ret}_{\mathcal{N}}})$ time for any $\epsilon < 1$, unless `SETH` fails. Therefore, the running time of this theorem is tight, to some extent.

Proof. Algorithm. For a reticulation v in a network $\mathcal{N}$ with child u, let $E^{(\uparrow vu)}$ be the set of edges of $\mathcal{N}$ that are between two vertices of $\text{anc}(v) \cup \{u\}$. Recall that $\text{off}(e) \subseteq X$ is the set of offspring of w for an edge $e = vw$ and the strictly affected edges T_Y for a set of taxa $Y \subseteq X$ is the set of edges e with $\text{off}(e) \subseteq Y$. Define two operations, called `take` and `leave`, that for an instance $\mathcal{I} = (\mathcal{N}, k, D)$ and a reticulation v of $\mathcal{N}$ return another instance of MAPPD. Every subset of taxa Y that *does* contain an offspring of v should be a solution for $\mathcal{I}$ if and only if Y is a solution for $\texttt{take}(\mathcal{I}, v)$. Similarly, every subset of taxa Y that *does not* contain an offspring of v should be a solution for $\mathcal{I}$ if and only if Y is a solution for $\texttt{leave}(\mathcal{I}, v)$.

We define $\texttt{leave}(\mathcal{I}, v)$ to be the instance $\mathcal{I}' = (\mathcal{N}', k, D)$ of MAPPD, in which k and D are unchanged and $\mathcal{N}'$ is the network that results from deleting the edges $T_{\text{off}(v)}$ and the resulting isolated vertices from $\mathcal{N}$. Recall that $\overline{D} := \sum_{e \in E} \lambda(e) - D$. We define $\texttt{take}(\mathcal{I}, v)$ to be the instance $\mathcal{I}' = (\mathcal{N}', k, D')$ of MAPPD with an unchanged k and $D' := D + \overline{D}$. Here, $\mathcal{N}'$ is the network that results from $\mathcal{N}$ by deleting the edges $E^{(\uparrow vu)}$, merging all the ancestors of v to a single vertex ρ, adding an edge ρu, and setting the weight of ρu to $\lambda(E^{(\uparrow vu)}) + \overline{D}$. For each vertex $w \neq u$ with $t \geq 1$ parents $u_1, \ldots, u_t$ in $\text{anc}(v)$, we add an edge ρw that has a weight of $\sum_{i=1}^{t} \lambda(u_i w)$. Observe that $\text{AP-PD}_{\mathcal{N}'}(X) = \text{AP-PD}_{\mathcal{N}}(X) + \overline{D}$. Figure 6.4 depicts an example of the operations `take` and `leave`.

Now, we are at the position to define the branching algorithm. Let $\mathcal{I} = (\mathcal{N}, k, D)$ be an instance of MAPPD. If $\mathcal{N}$ is a phylogenetic tree, solve the instance $\mathcal{I}$ with Faith's Algorithm [Ste05, Fai92]. Otherwise, let v be a reticulation of $\mathcal{N}$. Then, return `yes` if $\texttt{take}(\mathcal{I}, v)$ or $\texttt{leave}(\mathcal{I}, v)$ is a `yes`-instance of MAPPD and `no`, otherwise.

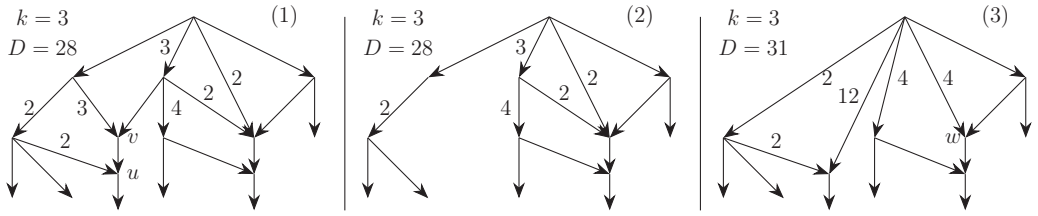


Figure 6.4: In this figure, an example for the usage of `leave` and `take` is given. A hypothetical instance $\mathcal{I}$ is given in (1). Here, the value of $\overline{D}$ is 3. In (2), the instance $\texttt{leave}(\mathcal{I}, v)$, and in (3) the instance $\texttt{take}(\mathcal{I}, v)$ is depicted. Unlabeled edges have a weight of 1. Observe in (3), the weight of the edge ρw is 4, as w has two edges from ancestors of v in $\mathcal{I}$ which have a weight of 2 each. The weight of ρu is 12, as in $\mathcal{I}$ the edges of $E^{(\uparrow vu)}$ have a combined weight of 9.

Correctness. The correctness of the base case is given by the correctness of Faith's Algorithm. We show that if $\mathcal{N}$ contains a reticulation v, then $\mathcal{I}$ is a `yes`-instance of MAPPD if and only if $\texttt{take}(\mathcal{I}, v)$ or $\texttt{leave}(\mathcal{I}, v)$ is a `yes`-instance of MAPPD.

Consider any set of taxa $Y \subseteq X$. Firstly, we claim that if $Y \cap \text{off}(e) = \emptyset$, then $\text{AP-PD}_{\mathcal{N}'}(Y) = \text{AP-PD}_{\mathcal{N}}(Y)$, where $\mathcal{N}'$ is the network in $\texttt{leave}(\mathcal{I}, v)$. Indeed, $\mathcal{N}'$ contains all the vertices and edges of $\mathcal{N}$ that have an offspring outside of $\text{off}(v)$. Therefore, $\text{AP-PD}_{\mathcal{N}'}(Y) = \text{AP-PD}_{\mathcal{N}}(Y)$. Secondly, we claim that if Y contains a vertex of $\text{off}(v)$, then $\text{AP-PD}_{\mathcal{N}'}(Y) = \text{AP-PD}_{\mathcal{N}}(Y) + \overline{D}$, where $\mathcal{N}'$ is the network in $\texttt{take}(\mathcal{I}, v)$. Recall that each edge $e = u_1u_2$ with $u_1 \neq \rho$ of $E(\mathcal{N}')$ is also an edge of $\mathcal{N}$ and $\lambda'(e) = \lambda(e)$. Further, for each edge $e = \rho u_2$ with $u_2 \neq u$ of $E(\mathcal{N}')$ there are edges $e_1 = u_{i_1}u_2, \ldots, e_t = u_{i_t}u_2$ of $E(\mathcal{N})$ with $\lambda'(e) = \sum_{i=1}^{t} \lambda(e_i)$. Now, let $Q = Q_1 \cup Q_2 \cup \{\rho u\}$ be the edges of $\mathcal{N}'$ that have at least one offspring in Y, of which edges in Q_1 have both endpoints in $V(\mathcal{N}') \setminus \{\rho\}$, and Q_2 are outgoing edges of ρ. Further, let $P = P_1 \cup P_2 \cup E^{(\uparrow vu)}$ be the edges of $\mathcal{N}$ that have at least one offspring in Y, of which edges in P_1 have both endpoints in $V(\mathcal{N}')$, and P_2 are edges with one endpoint in $\text{anc}(v) \setminus \{v\}$ and one endpoint in $V(\mathcal{N}') \setminus \{\rho\}$. Observe that since any vertex in $V(\mathcal{N}')$ has the same offspring in $\mathcal{N}$ as in $\mathcal{N}'$, we know $Q_1 = P_1$, and $\lambda'_\Sigma(Q_1) = \lambda_\Sigma(P_1)$. Further, $\lambda'_\Sigma(Q_2) = \lambda_\Sigma(P_2)$ as for each $u_2 \in V(\mathcal{N}') \setminus \{\rho\}$, the total weight of edges u_1u_2 with $u_1 \in \text{anc}(v) \setminus \{v\}$ in $\mathcal{N}$ is equal to the weight of the edge ρu_2 in $\mathcal{N}'$. It follows that

$$\begin{aligned} \text{AP-PD}_{\mathcal{N}'}(Y) &= \lambda'_\Sigma(Q_1) + \lambda'_\sigma(Q_2) + \lambda'(\rho u) \\ &= \lambda_\Sigma(P_1) + \lambda_\Sigma(P_2) + \lambda_\Sigma(E^{(\uparrow vu)}) + \overline{D} \\ &= \text{AP-PD}_{\mathcal{N}}(Y) + \overline{D}. \end{aligned}$$

By the above it follows that if Y is a solution for $\mathcal{I}$, then either Y is a solution

for $\texttt{leave}(\mathcal{I}, v)$ or Y is a solution for $\texttt{take}(\mathcal{I}, v)$. Conversely, if Y is a solution for $\texttt{leave}(\mathcal{I}, v)$ then $Y \cap \text{off}(e) = \emptyset$ and thus AP-PD$_{\mathcal{N}}(Y)$ = AP-PD$_{\mathcal{N}'}(Y) \geq D$, so Y is also a solution for $\mathcal{I}$. Finally, if Y is a solution for $\texttt{take}(\mathcal{I}, v)$ then $Y \cap \text{off}(e) \neq \emptyset$, as otherwise

$$\begin{aligned} \text{AP-PD}_{\mathcal{N}'}(Y) &\leq \text{AP-PD}_{\mathcal{N}'}(X) - \lambda'(\rho y) \\ &= D + 2\overline{D} - (\lambda_{\Sigma}(E^{(\uparrow vu)}) + \overline{D}) \leq D + \overline{D} - 1 < D'. \end{aligned}$$

Then, AP-PD$_{\mathcal{N}'}(Y)$ = AP-PD$_{\mathcal{N}}(Y) + \overline{D}$. It follows that AP-PD$_{\mathcal{N}}(Y) \geq D' - \overline{D} = D$ and Y is also a solution for $\mathcal{I}$.

Running Time. Let $\mathcal{I}$ be an instance of MAPPD that contains a reticulation v. The number of reticulations in $\mathcal{I}$ is greater than the number of reticulations in $\texttt{take}(\mathcal{I}, v)$ and $\texttt{leave}(\mathcal{I}, v)$, because at least the reticulation v is removed and no new reticulations are added. Therefore, the search tree contains $\mathcal{O}(2^{\text{ret}_{\mathcal{N}}})$ nodes. It can be checked in $\mathcal{O}(m)$ time, if $\mathcal{N}$ contains a reticulation. Faith's Algorithm takes $\mathcal{O}(k \cdot m)$ time [Ste05].

The sets off(v) and anc(v) for a vertex v, and T_Y for a set Y can be computed in $\mathcal{O}(m)$ time. Once anc(v) is computed, we can iterate over E to find the edges that are outgoing from anc(v) and in $\mathcal{O}(m)$ time we can compute the value for an edge ρw in $\mathcal{N}'$, which is also the time needed to compute $\lambda(\rho u)$ which needs $\overline{D}$ and the weight of $E^{(\uparrow vu)}$. Therefore, the instances $\texttt{take}(\mathcal{I}, v)$ and $\texttt{leave}(\mathcal{I}, v)$ can be computed in $\mathcal{O}(m)$ time.

Thus, a solution for MAPPD can be computed in $\mathcal{O}(2^{\text{ret}_{\mathcal{N}}} \cdot k \cdot m)$ time. □

Bordewich et al. showed that MAPPD can be solved in polynomial time on level-1-networks [BSW22]. We extend this result by showing that MAPPD is fixed-parameter tractable with respect to treewidth.

Theorem 6.6. MAPPD *can be solved in* $\mathcal{O}(9^{\text{tw}_{\mathcal{N}}} \cdot \text{tw}_{\mathcal{N}} \cdot k^2 \cdot m)$ *time.*

We first provide a sketch of the main ideas, here. This algorithm has many similarities with Theorem 5.9, however, here we have a tree-decomposition over a network and not a food-web.

We aim to find a set of edges E' that have an overall weight of at least D and that are incident with at most k leaves. Further, for each edge $e = uv \in E'$ we require that either v is a leaf or there is an edge $vw \in E'$. In this dynamic program algorithm over a nice tree decomposition, we index feasible partial solutions by a 3-coloring of the vertices. At a given node of the tree decomposition, a vertex v is colored:

- red, if it is still mandatory that we select an outgoing edge of v (because we have selected an incoming edge of v),

- green, if we can select incoming edges of v and do not need to select an outgoing edge of v (because v is a leaf or we have already selected an outgoing edge of v),
- black, if we have to not yet selected an edge incident with v (such that only the selection of an incoming edge of v makes the selection of an outgoing edge of v necessary).

We introduce each leaf as a green vertex and the other vertices as black vertices. In order to consider only feasible solutions, a vertex must be green or black when it is forgotten. The most important step of the algorithm is in the introduction of an edge, where colors may be adjusted depending on whether or not the new edge is included in E'.

Proof. We define a dynamic programming algorithm over a nice tree-decomposition T of the underlying undirected graph of an X-network $\mathcal{N} = (V, E, \lambda)$ in which vertices are introduced without incident edges and all edges are induced exactly once. We note that at every join bag the graph $\mathcal{N}_t[Q_t]$ is edgeless, if we only introduce edges right before we forget one of the incident vertices. Here, $\mathcal{N}_t$ is the graph with vertices V_t and edges E_t that are introduced at t or at decedants of t and are not forgotten.

Definition of the Table. Let $\mathcal{I} = (\mathcal{N}, k, D)$ be an instance of MAPPD and let T be a nice tree decomposition of $\mathcal{N}$. We index solutions by a partition $R \cup G \cup B$ of Q_t, and a non-negative integer s. For a set of edges $F \subseteq E_t$, we call a vertex $u \in V_t$ *green with respect to* F if u is a leaf or has an outgoing edge in F. We call u *red with respect to* F if u is not a leaf and has an incoming but no outgoing edge in F. Finally, we call u *black with respect to* F if u is not a leaf and has no incident edges in F. For a node $t \in T$, a partition $R \cup G \cup B$ of Q_t and an integer s, we call a set of edges F over V_t *feasible for* t, R, G, B, *and* s, if all the following conditions hold:

(F1) If uv is an edge in F and $v \notin Q_t$, then v is a leaf of $\mathcal{N}$ or v has an outgoing edge in F.

(F2) The vertices $R \subseteq Q_t$ are red with respect to F.

(F3) The vertices $G \subseteq Q_t$ are green with respect to F.

(F4) The vertices $B \subseteq Q_t$ are black with respect to F.

(F5) The number of leaves in $\mathcal{N}$ with an incoming edge in F is s.

We define $\mathcal{S}^*_{t,R,G,B,s}$ to be the family of all sets F that are feasible for t, R, G, B, and s.

We define a dynamic programming algorithm over a nice tree decomposition T. In a table entry $\mathrm{DP}[t, A, R, G, B, s]$, we store the greatest weight $\lambda_\Sigma(F)$ of a set F that is feasible for t, R, G, B, and s. If there is no feasible F, then we store a big negative value. Let r be the root of the nice tree-decomposition T. Then, $\mathrm{DP}[r, \emptyset, \emptyset, \emptyset, k]$ stores the greatest phylogenetic diversity that can be preserved with a budget of k. Hence, we can return `yes` if $\mathrm{DP}[r, \emptyset, \emptyset, \emptyset, k] \geq D$ and `no`, otherwise.

Now, we have everything we need to define the dynamic programming algorithm. In the calculations that follow, any time a value $\mathrm{DP}[t, R, G, B, s]$ is called for which $\mathrm{DP}[t, R, G, B, s]$ is not defined (in particular, if $s < 0$), we take $\mathrm{DP}[t, R, G, B, s]$ to be a large negative value. For our purposes a value of $-m \cdot \max_\lambda -1$ suffices, as even if every edge would be chosen, the entry at the root of the tree decomposition is still negative.

Leaf Node. For a leaf t of T the bags Q_t and V_t are empty. So if $s = 0$, we trivially store

$$\mathrm{DP}[t, \emptyset, \emptyset, \emptyset, s] = 0. \tag{6.3}$$

Otherwise, we store $\mathrm{DP}[t, R, G, B, s] = -m \cdot \max_\lambda -1$.

Introduce Vertex Node. Suppose now that t is an *introduce vertex node*, i.e. t has a single child t', $Q_t = Q_{t'} \cup \{v\}$, and v is an isolated vertex in $\mathcal{N}_t$. If either $v \in B$, or $v \in G$ and v is a leaf, we store

$$\mathrm{DP}[t, R, G, B, s] = \mathrm{DP}[t', R, G \setminus \{v\}, B \setminus \{v\}, s]. \tag{6.4}$$

Otherwise, we store $\mathrm{DP}[t, R, G, B, s] = -m \cdot \max_\lambda -1$.

Introduce Edge Node. Suppose now that t is an *introduce edge node*, i.e. t has a single child t', $Q_t = Q_{t'}$, and $e = vw$ is introduced at t'. The algorithm must decide whether e is affected by the solution, or not. If $v \notin G$ or $w \in B$, edge e can not be an affected edge and so we store $\mathrm{DP}[t, R, G, B, s] = \mathrm{DP}[t', R, G, B, s]$. Otherwise, we store

$$\mathrm{DP}[t, R, G, B, s] = \max\{\mathrm{DP}[t', R, G, B, s]; \max_{R', G', B'} \mathrm{DP}[t', R', G', B', s'] + \lambda(e)\}, \tag{6.5}$$

where the second maximum is over all possible partitions $R' \cup G' \cup B'$ of Q_t, such that $S \setminus \{v, w\} = S' \setminus \{v, w\}$ for all $S \in \{R, G, B\}$ (i.e. the two partitions agree on $Q_T \setminus \{v, w\}$), and such that if $w \in G'$ then $w \in G$; and $w \in R$, otherwise. Here, $s' = s - 1$ if w is a leaf, and $s' = s$ if not.

	B_1	R_1	G_1
B_2	B	R	G
R_2	R	R	G
G_2	G	G	G

Figure 6.5: This table shows the relationship between the three partitions $R_1 \cup G_1 \cup B_1$, $R_2 \cup G_2 \cup B_2$ and $R \cup G \cup B$ in the case of a join node, when $R_1 \cup G_1 \cup B_1$ and $R_2 \cup G_2 \cup B_2$ are qualified for $R \cup G \cup B$. The table shows which of the sets R, G, or B an element $v \in Q_t$ will be in, depending on its membership in R_1, G_1, B_1, R_2, G_2, and B_2. For example if $v \in R_1$ and $v \in B_2$, then $v \in R$.

Forget Node. Suppose now that t is *forget node*, i.e. t has a single child t' and $Q_t = Q_{t'} \setminus \{v\}$. We store

$$\mathrm{DP}[t, R, G, B, s] = \max\{\mathrm{DP}[t', R, G, B \cup \{v\}, s]; \mathrm{DP}[t', R, G \cup \{v\}, B, s]\}. \quad (6.6)$$

Join Node. Suppose now that $t \in T$ is an *join node*, i.e. t in T has two children t_1 and t_2 with $Q_t = Q_{t_1} = Q_{t_2}$. We call two partitions $R_1 \cup G_1 \cup B_1$ and $R_2 \cup G_2 \cup B_2$ of Q_t *qualified* for $R \cup G \cup B$ if $R = (R_1 \cup R_2) \setminus (G_1 \cup G_2)$ and $G = G_1 \cup G_2$ (and consequently $B = B_1 \cap B_2$). See Figure 6.5. We store

$$\mathrm{DP}[t, R, G, B, s] = \max_{(\pi_1, \pi_2) \in \mathcal{Q}, s'} \mathrm{DP}[t_1, R_1, G_1, B_1, s'] + \mathrm{DP}[t_2, R_2, G_2, B_2, s - s'], \quad (6.7)$$

where $s' \in [s]_0$, and $\mathcal{Q}$ is the set of pairs of partitions $\pi_1 = R_1 \cup G_1 \cup B_1$ and $\pi_2 = R_2 \cup G_2 \cup B_2$ that are qualified for R, G, B.

Correctness. Let t be a node of the nice tree-decomposition T, s an integer, and $R \cup G \cup B$ a partition of Q_t. We show that the value of $\mathrm{DP}[t, R, G, B, s]$ is correct, for each type of node individually.

If t is a *leaf node*, then Q_t and V_t are empty. Consequently, $R = G = B = \emptyset$ and any feasible set F is also empty. Therefore, we also conclude with (F5) that $s = 0$ and $\lambda_\Sigma(F) = 0$ for any $F \in \mathcal{S}^*_{t,R,G,B,s}$. Hence, we store the correct value in Recurrence (6.3).

Let t be an *introduce vertex node* with child t' and $Q_t = Q_{t'} \cup \{v\}$. Then, the graph $\mathcal{N}_t$ is the graph $\mathcal{N}_{t'}$ with an additional isolated vertex v. Thus, v is black with respect to any $F \subset E_t$, unless v is a leaf in which case v is green. Thus, there is no feasible set F for $\mathcal{S}^*_{t,R,G,B,s}$ unless either $v \in B$ or $v \in G$ and v is a leaf. Assuming this is the case, any set $F \subseteq E_t$ is feasible for t, R, G, B, and s if and only if it is feasible for t', R, $G \setminus \{v\}$, $B \setminus \{v\}$, and s. Hence, we store the right value.

Let t be an *introduce edge node* with child t' and $Q_t = Q_{t'}$ and $\mathcal{N}_t - e = \mathcal{N}_{t'}$. Define $e := vw$. Clearly, every set $F \subseteq E_t \setminus \{e\}$ is feasible for t, R, G, B, and s if and only if F is also feasible for t', R, G, B, and s, because then $\mathcal{N}_t[F]$ is isomorphic to $\mathcal{N}_{t'}[F]$. Now, consider a set $F \subseteq E_t$ with $vw \in F$. Note that v is green, and w cannot be black, with respect to any such set F. Therefore, such an F only exists if $v \in G$ and $w \notin B$. Now, let F' be $F \setminus \{vw\}$. Every vertices u, that is not v or w, is red/green/black with respect to F if and only if u is red/green/black with respect to F'. Observe that if w is green with respect to F if and only if it is green with respect to F' (as it has the same outgoing edges in F and F'). If w is red or black with respect to F', then it is red with respect to F (as it has an incoming edge and no outgoing edges). On the other hand, v could be any color with respect to F', but is necessarily green with resepct to F. Likewise, we can show the correctness of s'. Thus, the correct value is stored.

Let t be a *forget node* with child t' and $Q_t = Q_{t'} \setminus \{v\}$. We show that a set F is feasible for t, R, G, B, and s if and only if F is also feasible for t', R, G, $B \cup \{v\}$, and s or for t', R, $G \cup \{v\}$, B, and s. It then follows that $\mathrm{DP}[t, R, G, B, s]$ stores the correct value.

To this end, let F be a set of edges that is feasible for t, R, G, B, and s. Since $v \notin Q_t$ and condition (F1) satisfies, v either has an outgoing edge in F, is a leaf, or does not have any incoming edge in F. It follows that v is green or black with respect to F. From this it is straightforward to confirm that F is feasible for t', R, G, $B \cup \{v\}$, and s, or for t', R, $G \cup \{v\}$, B, and s. Conversely, if F is feasible for t', R, G, $B \cup \{v\}$, and s, or for t', R, $G \cup \{v\}$, B, and s, then F also satisfies condition (F1) for t', R, G, $B \cup \{v\}$, and s (in particular, the condition is satisfied for v as v is green or black with respect to F). It is straightforward to confirm that F the remaining conditions to be feasible with respect to $\mathcal{S}^*_{t,R,G,B,s}$.

Let t be a *join node* with children t_1 and t_2. We show that there is an $F \in \mathcal{S}^*_{t,R,G,B,s}$ if and only if there are partitions $R_1 \cup G_1 \cup B_1$ and $R_2 \cup G_2 \cup B_2$, qualified for R, G, and B and $s' \leq s$ such that there are $F_1 \in \mathcal{S}^*_{t_1,R_1,G_1,B_1,s'}$ and $F_2 \in \mathcal{S}^*_{t_2,R_2,G_2,B_2,s-s'}$ with $F_1 \cup F_2 = F$.

First, let F be feasible for t, R, G, B, and s. Let F_i be the subset of edges of F that occur in $\mathcal{N}_{t_i}$ for $i \in \{1, 2\}$. Because every edge is introduced exactly once, the sets F_1 and F_2 are disjoint. We define s' to be the number of edges of F_1 that are incident with a leaf. Further, we define R_i to be the set of vertices in Q_t that are not leaves in $\mathcal{N}$ and have an incoming but no outgoing edge in F_i for $i \in \{1, 2\}$. Similarly, we define G_i to be the set of vertices that are leaves or have an outgoing edge in F_i for $i \in \{1, 2\}$. We show that $F_1 \in \mathcal{S}^*_{t_1,R_1,G_1,B_1,s'}$ and $F_2 \in \mathcal{S}^*_{t_2,R_2,G_2,B_2,s-s'}$. Let $e = uv$ be an edge in F_i and let v be an internal-vertex and $v \notin Q_{t_i}$ for $i \in \{1, 2\}$.

Because $Q_t = Q_{t_i}$, we conclude that $v \notin Q_t$. Thus, with (F1) we conclude that v has an outgoing edge e^* in F. Because v is not in Q_t and v is in V_{t_i}, the vertex v is not in $V_{t_{3-i}}$. Thus, the edge e^* is also in F_i and so F_1 and F_2 satisfy (F1). By definition, F_1 and F_2 satisfy conditions (F2) to (F4). Also by definition, F_1 satisfies (F5). As F_2 contains the edges of F that are not in F_1, there are $s - s'$ edges in F_2 that are incident with a leaf. Hence, F_2 satisfies (F5) and $F_1 \in \mathcal{S}^*_{t_1,R_1,G_1,B_1,s'}$ and $F_2 \in \mathcal{S}^*_{t_2,R_2,G_2,B_2,s-s'}$. It remains to show that $R_1 \cup G_1 \cup B_1$ and $R_2 \cup G_2 \cup B_2$ are qualified for R, G, and B. If v is a leaf, we conclude that v is in G_1, G_2 and G. Further, v has an outgoing edge in F, if and only if v has an outgoing edge in F_1 or F_2. We conclude that $G = G_1 \cup G_2$. For each $v \in R$ there is an incoming edge e but no outgoing edge in F. Consequently, if $v \in R$ then $v \notin G_1 \cup G_2$. Further, without loss of generality, $e \in F_1$. However, there is no outgoing edge of v in F_1. Thus, $v \in R_1$ and so $R \subseteq (R_1 \cup R_2) \setminus (G_1 \cup G_2)$. If $v \in (R_1 \cup R_2) \setminus (G_1 \cup G_2)$ then there is an incoming edge of v in F_1 or F_2 and therefore in F, but no outgoing edges. Thus, v is red and in R.

Secondly, assume that there are R_1, G_1, R_2, and G_2 qualified for R and G, and $s' \leq s$ such that there are $F_1 \in \mathcal{S}^*_{t_1,R_1,G_1,B_1,s'}$ and $F_2 \in \mathcal{S}^*_{t_2,R_2,G_2,B_2,s-s'}$. We show that $F := F_1 \cup F_2$ is feasible for t, R, G, B, and s. Let $e = uv$ be an edge in F with v is not a leaf and $v \notin Q_t$. Without loss of generality, $e \in F_1$. Because $Q_t = Q_{t_1}$, we conclude that $v \notin Q_{t_1}$. Thus, v has an outgoing edge in F_1 and therefore in F, we know F satisfies condition (F1). Let v be a vertex of Q_t. If $v \in R = (R_1 \cup R_2) \setminus (G_1 \cup G_2)$, then there is an edge e incoming at v in F_1 or F_2, but F_1 and F_2 do not contain an outgoing edge of v. Consequently, v has an incoming edge in F but no outgoing edges. Analogously, we can see that if $v \in G = G_1 \cup G_2$ then v has an outgoing edge in F and if $v \in B = B_1 \cap B_2$, then v is not incident with edges. Because that covers all options, F satisfies conditions (F2) to (F4). Let $\hat{E}_1$ and $\hat{E}_2$ be the edges of F_1 and F_2, respectively, that are incident with a leaf. Thus, $|\hat{E}_1| = s'$ and $|\hat{E}_2| = s - s'$. Because every edge is introduced exactly once, the sets F_1 and F_2 and therefore $\hat{E}_1$ and $\hat{E}_2$ are disjoint. Consequently, $\hat{E}_1 \cup \hat{E}_2$ contains the s edges that are incoming at leaves. Thus, F satisfies condition (F5). Hence, the value that is stored with Recurrence (6.7) is correct.

Running Time. The tree decomposition T has $\mathcal{O}(m)$ nodes.

For a leaf node, an introduce vertex nodes and a forget node, the value of each entry $\mathrm{DP}[t, R, G, B, s]$ can clearly be computed in time linear in $\mathrm{tw}_{\mathcal{N}}$. Checking the conditions in an introduce edge node requires checking the colors of v and w. Further, the partition $R' \cup G' \cup B'$ agrees on all vertices but v and w. Therefore, we can also compute the values of entries in time linear in $\mathrm{tw}_{\mathcal{N}}$ for an introduce edge node. Altogether, we can compute the value of all entries of nodes that are not join

nodes in $\mathcal{O}(3^{\mathrm{tw}_\mathcal{N}} \cdot \mathrm{tw}_\mathcal{N} \cdot k \cdot m)$ time.

For the computation in a join node t, we first store a big negative integer in each entry $\mathrm{DP}[t, R, G, B, s]$ for partitions $R \cup G \cup B$ of Q_t and an integer $s \in [k]_0$. Then, iterate over partitions $R_1 \cup G_1 \cup B_1$ and $R_2 \cup G_2 \cup B_2$ of Q_t. From the partitions $R_1 \cup G_1 \cup B_1$ and $R_2 \cup G_2 \cup B_2$, compute the implicitly defined partition $R \cup G \cup B$ of Q_t in $\mathcal{O}(\mathrm{tw}_\mathcal{N})$ time. See Figure 6.5. Iterate over $s \in [k]_0$ and $s' \in [s]_0$. If $\mathrm{DP}[t_1, R_1, G_1, B_1, s'] + \mathrm{DP}[t_2, R_2, G_2, B_2, s - s']$ exceeds the value of $\mathrm{DP}[t, R, G, B, s]$, then replace it. Otherwise, let $\mathrm{DP}[t, R, G, B, s]$ stay unchanged. After the iterations, we have computed the values of $\mathrm{DP}[t, R, G, B, s]$ for all partitions $R \cup G \cup B$ of Q_t and integer $s \in [k]_0$. Therefore, $\mathcal{O}(9^{\mathrm{tw}_\mathcal{N}} \cdot \mathrm{tw}_\mathcal{N} \cdot k^2)$ time is required to compute all values of entries for each join node t.

Hence, a solution for MAPPD can be computed in $\mathcal{O}(9^{\mathrm{tw}_\mathcal{N}} \cdot \mathrm{tw}_\mathcal{N} \cdot k^2 \cdot m)$ time. □

6.5 A Kernelization for Reticulation-Edges

In Theorem 6.5, we presented a branching algorithm that proves that MAPPD is FPT when parameterized by the number of reticulations, $\mathrm{ret}_\mathcal{N}$. In this section, we show that MAPPD admits a kernelization algorithm of polynomial size with respect to $\text{e-ret}_\mathcal{N}$. Recall that $\text{e-ret}_\mathcal{N}$ is the number of reticulation-edges which need to be removed such that $\mathcal{N}$ is a tree. Observe $\text{e-ret}_\mathcal{N} \geq \mathrm{ret}_\mathcal{N}$ and in binary networks they are equal.

We first show how to bound the number of vertices and edges by a polynomial in $\text{e-ret}_\mathcal{N}$, without giving any such bound on the weights of the edges. Afterwards, we will apply a result from [EKMR17, FT87] to get an appropriate bound on the edge weights.

Theorem 6.7. *Given an instance $\mathcal{I} = (\mathcal{N}, k, D)$ of* MAPPD*, an equivalent instance $\mathcal{I}^* = (\mathcal{N}^* = (V^*, E^*, \lambda^*), k^*, D^*)$ of* MAPPD *with $|V^*|, |E^*| \in \mathcal{O}(\text{e-ret}_\mathcal{N}{}^2)$ and $k^* \in \mathcal{O}(\text{e-ret}_\mathcal{N})$ can be computed in $\mathcal{O}(m^2 \log^2 m \cdot \log \max_\lambda)$ time.*

Throughout this section, assume that $\mathcal{I} = (\mathcal{N} = (V, E, \lambda), k, D)$ is an instance of MAPPD with ρ being the root of $\mathcal{N}$. Let $R \subseteq V$ be the set of reticulation vertices of $\mathcal{N}$. We apply the following reduction rules exhaustively, and each rule is applied only if none of the previous rules apply. After any of the reduction rules let $\mathcal{N}' = (V, E', w')$ denote the new network.

Reduction Rule 6.8. *Let $v \in V$ be a vertex with children x and y that are leaves. Assume $\lambda(vx) \geq \lambda(vy)$. If $v \neq \rho$, then replace the edge vy with an edge ρy of weight $\lambda'(\rho y) = \lambda(vx)$.*

Lemma 6.9. *Reduction Rule 6.8 is correct and can be applied in $\mathcal{O}(n)$ time.*

Proof. Correctness. We first show that if $\mathcal{I}$ is a `yes`-instance of MAPPD then $\mathcal{I}' := (\mathcal{N}', k, D)$ is a `yes`-instance of MAPPD. Afterward, we show the converse. If $\mathcal{I}$ is a `yes`-instance then let $S \subseteq X$ be a solution. If $y \in S$ but $x \notin S$, then define $S' := (S \cup \{x\}) \setminus \{y\}$. We conclude that S' is also a solution because $\text{AP-PD}_{\mathcal{N}}(S') = \text{AP-PD}_{\mathcal{N}}(S) + \lambda(vx) - \lambda(vy) \geq \text{AP-PD}_{\mathcal{N}}(S)$. Therefore, we may assume without loss of generality that $x \in S$ or $y \notin S$. If $x \notin S$ and $y \notin S$, then observe that $\text{AP-PD}_{\mathcal{N}}(S) = \text{AP-PD}_{\mathcal{N}'}(S)$ since all edges affected by S appear in both networks with the same weight. Similarly, if $x \in S$ and $y \notin S$, then $\text{AP-PD}_{\mathcal{N}}(S) = \text{AP-PD}_{\mathcal{N}'}(S)$. Finally if $x \in S$ and $y \in S$, then any edge $e \in E(\mathcal{N}) \setminus \{vy\}$ is affected by S in $\mathcal{N}$ if and only if it is affected by S in $\mathcal{N}'$ (in particular, if y is an offspring of e in $\mathcal{N}$ then x is an offspripng of e in $\mathcal{N}'$). It follows that $\text{AP-PD}_{\mathcal{N}}(S) = \text{AP-PD}_{\mathcal{N}'}(S) - \lambda'(\rho y) + \lambda(vx) = \text{AP-PD}_{\mathcal{N}'}(S)$. Thus, S is a solution of $\mathcal{I}'$ in all cases.

Conversely, observe that if an edge $e \in E(\mathcal{N}') \setminus \{\rho y\}$ is affected by $S \subseteq X$ in $\mathcal{N}'$ then it is also affected by S in $\mathcal{N}$. Thus, $\text{AP-PD}_{\mathcal{N}'}(S) \leq \text{AP-PD}_{\mathcal{N}}(S)$ and consequently each solution for $\mathcal{I}'$ is also a solution for $\mathcal{I}$. Thus, if $\mathcal{I}'$ is a `yes`-instance of MAPPD, then so is $\mathcal{I}$.

Running Time. In $\mathcal{O}(n)$ time, we can determine whether there exists a vertex v which has two children in X. If such a vertex v exists, we only need to compare $\lambda(vx)$ and $\lambda(vy)$ in $\mathcal{O}(\log \max_\lambda)$ time. □

Note that Reduction Rule 6.8 might create degree-2-vertices; these are handled by the next reduction rule.

Reduction Rule 6.10. *Let $v \in V$ be a vertex of degree-2. Let u be the parent and w be the child of v. Remove v, its incident edges and create an edge uw with a weight of $\lambda'(uw) = \lambda(uv) + \lambda(vw)$.*

Lemma 6.11. *Reduction Rule 6.10 is correct and can be applied in $\mathcal{O}(n)$ time.*

Proof. Correctness. The correctness follows from the observation that for any $S \subseteq X$ we have $\text{AP-PD}_{\mathcal{N}}(S) = \text{AP-PD}_{\mathcal{N}'}(S)$.

Running Time. In $\mathcal{O}(n)$ time, we can find out whether there exists a vertex with a degree of 2. If such a vertex v exists, it takes constant time to remove v and its incident edges and create the new edge uw with the appropriate weight. □

To evaluate the size of the network after applying reduction rules, we adapt the language of *network generators*. We refer the reader to [GBP09, vIKK+09] for an in-depth study.

Definition 6.12.

(a) A vertex $v \in V \setminus R$ is a *core-vertex* if v has two children, u_1 and u_2, where $u_1 \neq u_2$, and $\mathrm{desc}(u_i) \cap R \neq \emptyset$ for each $i \in \{1, 2\}$.

(b) Let Q be the set of core-vertices of $\mathcal{N}$.

(c) A *side-path* in $\mathcal{N}$ is a path from u to w for two vertices $u, w \in R \cup Q$ with no internal vertices in $R \cup Q$. The internal vertices of a side-path are *side-vertices*.

(d) Let Z be the set of side-vertices of $\mathcal{N}$.

Note that after applying Reticulation Rules 6.8 and 6.10, every non-leaf vertex is either a reticulation or has at least one reticulation as a descendant. Therefore, every non-leaf vertex is in Q, in R, or in Z. Every side-vertex has exactly one child which is a leaf. The following result is similar to one in [GBP09]. For completeness we prove it here.

Observation 6.13. *Every network $\mathcal{N}$ has $\mathcal{O}(\text{e-ret}_\mathcal{N})$ side-paths. Further, it is satisfied that $|R| + |Q| \in \mathcal{O}(\text{e-ret}_\mathcal{N})$.*

Proof. Observe that $|R| = \text{ret}_\mathcal{N} \leq \text{e-ret}_\mathcal{N}$. As every side-path in $\mathcal{N}$ ends at a core vertex or reticulation, we have that the number of side-paths is at most

$$\sum_{v \in Q \cup R} \deg^-(v) = \sum_{r \in R} (\deg^-(r) - 1) + |Q| + |R| \tag{6.8}$$

$$= \text{e-ret}_\mathcal{N} + |Q| + |R| \leq 2 \cdot \text{e-ret}_\mathcal{N} + |Q|. \tag{6.9}$$

Conversely, each core-vertex has at least two side-paths leaving it, from which it follows that the number of side-paths is at least $2 \cdot |Q|$. We conclude that $2 \cdot |Q| \leq 2 \cdot \text{e-ret}_\mathcal{N} + |Q|$ and so $|Q| \leq 2 \cdot \text{e-ret}_\mathcal{N}$. This implies that the number of side-paths is at most $4 \cdot \text{e-ret}_\mathcal{N}$. □

Reduction Rule 6.14. *Let v and w be side-vertices and let v be the parent of w. Further, let x_v and x_w be leaves which are children of v and w, respectively. If $\lambda(vx_v) \leq \lambda(wx_w) + \lambda(vw)$ then replace the edge vx_v with an edge ρx_v with a weight of $\lambda'(\rho x_v) = \lambda(vx_v)$.*

Lemma 6.15. *Reduction Rule 6.14 is correct and can be applied in $\mathcal{O}(n)$ time.*

Proof. Correctness. The proof follows similar lines as the proof of Lemma 6.9. Let $S \subseteq X$ be a solution of $\mathcal{I}$. Assume that $x_v \in S$ but $S \cap \text{off}(w) = \emptyset$. Then, let $S' := (S \cup \{x_w\}) \setminus \{x_v\}$. Observe that wx_w and vw are affected by S' but not by S, while the only edge affected by S and not by S' is vx_v. Thus, AP-PD$_{\mathcal{N}}(S') =$ AP-PD$_{\mathcal{N}}(S) - \lambda(vx_v) + \lambda(wx_w) + \lambda(vw)$, which by the condition of the reduction rule is at least AP-PD$_{\mathcal{N}}(S)$. Therefore, also S' is a solution of $\mathcal{I}$ with $x_v \notin S$. Thus, we may now assume that S contains a taxon from off(w) or $x_v \notin S$. In either case, we can observe that AP-PD$_{\mathcal{N}}(S) =$ AP-PD$_{\mathcal{N}'}(S)$.

Conversely, observe that any edge $e \in E(\mathcal{N}') \setminus \{\rho x_v\}$ affected by S in $\mathcal{N}'$ is also affected by S in $\mathcal{N}$. Thus AP-PD$_{\mathcal{N}'}(S) \leq$ AP-PD$_{\mathcal{N}}(S)$ for any $S \subseteq X$, and so each solution for $\mathcal{I}'$ is also a solution for $\mathcal{I}$.

Running Time. In $\mathcal{O}(n)$ time, we can find out whether there exists a vertex satisfying the conditions of v in Reduction Rule 6.14, and if so edit the network accordingly. □

Reduction Rule 6.16. *Let u and w be core-vertices or reticulations with $u \neq \rho$ and for some $\ell > 1$ let $v_1, \ldots, v_\ell$ be side-vertices such that $uv_1, v_\ell w, v_i v_{i+1} \in E$ for each $i \in [\ell - 1]$. Remove the edge v_1v_2 and add edges ρv_2 and $v_1 w$ with a weight of $\lambda'(\rho v_2) = \lambda(v_1v_2)$, and $\lambda'(v_1 w) = 0$.*

An application of this reduction rule is depicted in Figure 6.6. Here, we slightly bend our own definitions in which we required that $\lambda(e) > 0$ for each edge e and that each vertex either has an in-degree of 1 or an out-degree of 1. We note that after applying the reduction rules exhaustively, we can adjust the instance to ensure these requirements are met. First, we can replace each vertex v which has an in-degree and an out-degree of larger than 1 with v_{in} receiving all the incoming edges of v and v_{out} receiving v in all the outgoing edges and adding an edge $v_{\text{in}}v_{\text{out}}$ with a weight of 0. Furthermore, we can multiply each weight and D by $m + 1$. Then, we can set the edges with a weight of 0 to a weight of 1.

Lemma 6.17. *Reduction Rule 6.16 is correct and can be applied in $\mathcal{O}(m)$ time.*

Proof. Correctness. Observe that if an edge $e \in E(\mathcal{N}') \setminus \{\rho v_2, v_1 w\}$ is affected by $S \subseteq X$ in $\mathcal{N}'$ then e is also affected by S in $\mathcal{N}$. Furthermore, ρv_2 is affected by S in $\mathcal{N}'$ if and only if v_1v_2 is affected by S in $\mathcal{N}$. Therefore, for every set $S \subseteq X$ we have AP-PD$_{\mathcal{N}'}(S) \leq$ AP-PD$_{\mathcal{N}}(S) + w(v_1 w) =$ AP-PD$_{\mathcal{N}}(S)$, and each solution for $\mathcal{I}'$ is also a solution for $\mathcal{I}$.

We now show the converse. We first observe some facts. Let x_i denote the leaf child of v_i for each $i \in [\ell]$. We observe that due to Reduction Rule 6.14, we may

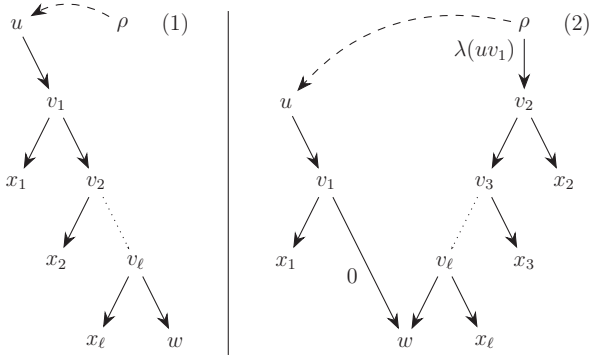


Figure 6.6: This figure in (1) depicts the path from a core-vertex v to another core vertex w and in (2) an application of Reduction Rule 6.16 to the path depicted in (1).

assume that $\lambda(v_i x_i) \geq \lambda(v_i v_{i+1}) + \lambda(v_{i+1} x_{i+1})$ for each $i \in [\ell - 1]$. Consequently,

$$\begin{aligned} \lambda(v_1 x_1) &\geq \lambda(v_2 x_2) + \lambda(v_1 v_2) && (6.10) \\ &\geq \lambda(v_3 x_3) + \lambda(v_2 v_3) + \lambda(v_2 v_1) && (6.11) \\ &\geq \lambda(v_i x_i) + \sum_{j=1}^{i-1} \lambda(v_j v_{j+1}) && (6.12) \end{aligned}$$

for each $i \in [\ell - 1]$.

Let S be a solution of $\mathcal{I}$. Observe that if S contains x_1 or an offspring of w, then $\text{AP-PD}_{\mathcal{N}}(S) = \text{AP-PD}_{\mathcal{N}'}(S)$. Similarly, if S does not contain one of the taxa $x_1, \ldots, x_\ell$ then $\text{AP-PD}_{\mathcal{N}}(S) = \text{AP-PD}_{\mathcal{N}'}(S)$. So, assume now that S does not contain x_1 nor an offspring of w but $x_i \in S$ for some $i \in [\ell]$ with $i > 1$. Define $S' := (S \cup \{x_1\}) \setminus \{x_i\}$. Then,

$$\text{AP-PD}_{\mathcal{N}}(S') \geq \text{AP-PD}_{\mathcal{N}}(S) + \lambda(v_1 x_1) - \lambda(v_i x_i) - \sum_{j=1}^{i-1} \lambda(v_j v_{j+1}), \quad (6.13)$$

which by the observation above is at least $\text{AP-PD}_{\mathcal{N}}(S)$. As $x_1 \in S'$, we have that $\text{AP-PD}_{\mathcal{N}'}(S') = \text{AP-PD}_{\mathcal{N}}(S')$, and so S' is a solution for $\mathcal{I}'$.

Running Time. To find the vertices u and w we iterate over the vertices in Q as vertex u and consider each outgoing edge as a path to u. Therefore, all possible combinations of u and w are found in $\mathcal{O}(m)$ time. Each operation can be executed in constant time. □

We now categorize the vertices in some sets to be able to easier refer to them.

Definition 6.18.

(a) Define A to be the intersection of X with children of ρ.

(b) Define B to be the set of vertices which can be reached from $R \cup (Q \setminus \{\rho\})$.

(c) Define Y to be the side-vetices which are children of ρ and define Y_X to be the intersection of X with the children of Y.

(d) Define Z to be the side-vetices which are not in $B \cup Y$ and define Z_X to be the intersection of X with the children of Z.

(e) Define $a^* := \max_{x \in A} \lambda(\rho x)$ and $x^* := \arg\max_{x \in A} \lambda(\rho x)$.

(f) Define $c^* := \max_{y \in Y_X} \lambda(v_y y)$ and $y^* := \arg\max_{y \in Y_X} \lambda(v_y y)$ where v_y is the parent of y.

In the following, for a taxon $x \in X$, we use an operation *remove* x when we delete x from X and V as well as the incoming edge of x from E. Further, we use an operation *save* x consisting of these steps:

- Reduce k by 1,
- reduce D' by AP-PD$_{\mathcal{N}}(\{x\})$,
- delete x from X,
- remove all edges in $E_{\{x\}}$, and
- identify all vertices with no incoming edge to a single root.

Lemma 6.19. *For a given instance $\mathcal{I}$ of* MAPPD *and a taxon x,*

(a) $\mathcal{I}$ has a solution S with $x \in S$ if and only if $S \setminus \{x\}$ is a solution for the instance after x is saved, and

(b) $\mathcal{I}$ has a solution S with $x \notin S$ if and only if S is a solution for the instance after x is removed.

Proof. To see the first claim, let $\mathcal{N}'$ be the network after saving x. Any edge in $\mathcal{N}'$ affected by some $S \subseteq X \setminus \{x\}$ has a corresponding edge in $E(\mathcal{N}) \setminus E_{\{x\}}$ with the same offspring, from which it follows that AP-PD$_{\mathcal{N}}(S \cup \{x\}) \geq$ AP-PD$_{\mathcal{N}'}(S) + D'$. Conversely, if $x \in S \subseteq X$ then any edge in $E_S \setminus E_{\{x\}}$ has a corresponding edge in $\mathcal{N}'$ affected by $S \setminus \{x\}$, from which it followed that AP-PD$_{\mathcal{N}'}(S \setminus \{x\}) \geq$ AP-PD$_{\mathcal{N}}(S) - D'$.

The second claim follows immediately from the definition of AP-PD$_{\mathcal{N}}(S)$. □

Reduction Rule 6.20. *If $k > |B| + |Y|$ and $a^* > c^*$, then save x^*. If $k > |B| + |Y|$ and $a^* \leq c^*$, then save y^*.*

Lemma 6.21. *Reduction Rule 6.20 is correct and can be applied in $\mathcal{O}(m)$ time.*

Proof. Correctness. Let $v_{1,1}, v_{2,1}, \ldots v_{|Y|,1}$ denote the vertices of Y. Let $v_{j,1}v_{j,2}, \ldots v_{j,\ell_j}$ be the side-vertices on the path from $v_{j,1}$ to a core vertex for each $j \in [|Y|]$. Let $z_{j,i}$ be the leaf child of $v_{j,i}$ for each $j \in [|Y|]$, and $i \in [\ell_j]$. This mapping is unique after Reduction Rule 6.8 has been applied exhaustively. Observe that Z is the set $\{v_{j,i} \mid 2 \leq j \leq |Y|, i \in [\ell_j]\}$.

Similarly, we have $\lambda(v_{j,1}y_{j,1}) \geq \lambda(v_{j,i}y_{j,i}) + \sum_{h=1}^{i-1} \lambda(v_{j,h}v_{j,h+1})$ for each $j \in [|Y|]$, and each $i \in [\ell_j]$. Consequently, we may assume that if $y_{j,i}$ is in a solution for some $i > 1$, then so is $y_{j,1}$.

Furthermore, for any solution that contains $y_{j,i}$ and $y_{j,1}$ for $i > 1$, we can assume the solution contains y^* as otherwise replacing $y_{j,i}$ with y^* gives another solution, because $w(v_{y^*}y^*) \geq \lambda(v_{j,i}y_{j,i})$ where v_{y^*} the parent of y^* and for each $j \in [|Y|]$, and each $i \in [\ell_j]$. Thus, if a solution S contains any element of Z_X we can assume S also contains y^*.

Now, suppose $k > |B| + |Y|$. Then, any solution S contains at least one element of $A \cup Z_X$. This implies that S contains at least one taxon of $A \cup \{y^*\}$ as S contains y^* if it contains any taxon in Z_X. If $a^* > c^*$, then S contains x^*, as otherwise we could replace a taxon from $(A \setminus \{x^*\}) \cup \{y^*\}$. So, in this case, S contains x^*, and we can save x^* by Lemma 6.19. Otherwise, we may assume S contains y^*, as otherwise we can replace an element from A with y^*.

Running Time. We can compute the size of B and Y and find a^* and c^* in $\mathcal{O}(n)$ time. Saving x^* or y^* takes $\mathcal{O}(m)$ time. □

Reduction Rule 6.22. *Let $x_1, \ldots, x_{|A|}$ be the taxa in A such that $\lambda(\rho x_i) \geq \lambda(\rho x_{i+1})$ for each $i \in [|A| - 1]$. If $k > |A|$, then remove $x_{k+1}, \ldots, x_{|A|}$ and their incident edges from $\mathcal{N}$ if $|A| > k$.*

Lemma 6.23. *Reduction Rule 6.22 is correct and can be applied in $\mathcal{O}(n \log n)$ time.*

Proof. Correctness. Clearly, any solution for the instance $\mathcal{I}'$ is also a solution for $\mathcal{I}$. Therefore, let S be a solution for $\mathcal{I}$. If $S \cap \{x_{k+1}, \ldots, x_{|A|}\} = \emptyset$, then S is also a solution for $\mathcal{I}'$. Assume that $x_i \in S$ for some $i \in \{k+1, \ldots, |A|\}$. As $|S| \leq k$ we conclude that there is a taxon x_j for $j \in [k]$ with $x_j \notin S$. Because $\lambda(\rho x_j) \geq \lambda(\rho x_i)$, we conclude that $(S \cup \{x_j\}) \setminus \{x_i\}$ is also a solution for $\mathcal{I}$. Thus, we may assume that $S \cap \{x_{k+1}, \ldots, x_{|A|}\} = \emptyset$.

Running Time. We can sort A in $\mathcal{O}(n \log n)$ time. □

Reduction Rule 6.24. *Let $z_0, \ldots, z_{k+1} \in Y \cup Z$ be vertices with z_i being the parent of z_{i+1} for $i \in [k]_0$. Add an edge $z_{k-1}z_{k+1}$ of weight $\lambda(z_{k-1}z_k) + \lambda(z_kz_{k+1})$. Remove the vertices which are reachable by z_k but not by z_{k+1} and the incident edges from $\mathcal{N}$.*

Lemma 6.25. *Reduction Rule 6.24 is correct and can be applied in $\mathcal{O}(n)$ time.*

Proof. Correctness. Let x be a taxon which is reachable by z_k but not by z_{k+1}. Because we applied Reduction Rules 6.8 and 6.14 exhaustively, each vertex z_i contains at most one child x_i in X and $\lambda(z_ix_i) > \lambda(z_{i+1}x_{i+1}) + \lambda(z_iz_{i+1})$. Consequently, we can assume that if x_i is in a solution then so are $x_0, \ldots, x_{i-1}$. Therefore, x_k can not be in a solution, and an application of Lemma 6.19 is correct.

Running Time. We can find an appropriate vertex z_k in $\mathcal{O}(n)$ time and edit the network in constant time. □

Finally, we have everything to proof this section's main theorem.

Proof of Theorem 6.7. For a given instance $\mathcal{I}$ apply all of the reduction rules exhaustively to receive instance $\mathcal{I}^* = (\mathcal{N}^* = (V^*, E^*), k^*, D^*)$ of MAPPD. We denote with R, Q, and so the respective set in the original instance and with R^*, Q^*, and so the respective set in $\mathcal{I}^*$.

The correctness follows from the correctness of the reticulation rules.

Running time. Observe that Reduction Rules 6.20, 6.22, and 6.24 reduce $|X|$ by 1, while Reduction Rules 6.8 and 6.14 reduce $|X \setminus A|$ by 1. As none of the reduction rules increase $|X|$ or $|X \setminus A|$, these rules are applied at most $|X| + |X \setminus A| \leq 2n$ times in total. For Reduction Rule 6.10, we note that only Reduction Rules 6.8, 6.14, 6.20, 6.22, and 6.24 can create a degree-2 vertex. Thus, every application of Reduction Rule 6.10 occurs after one of these other rules, and thus Reduction Rule 6.10 is applied at most $2 \cdot |X|$ times.

For Reduction Rule 6.16, we observe that each application of this rule reduces the number of side-paths starting at a non-root vertex of length of at least 2. None of the reduction rules increase this measure, and so the number of application of Reduction Rule 6.16 is bounded by the number of side-paths in the original network, which is $\mathcal{O}(\text{e-ret}_{\mathcal{N}})$ by Observation 6.13.

So, the total number of applications of all reduction rules is $\mathcal{O}(|X| + \text{e-ret}_{\mathcal{N}})$. Observe that a single application of any reduction rule is done in $\mathcal{O}(m + n \log n)$ time by Lemmas 6.9, 6.11, 6.15, 6.17, 6.21, 6.23, and 6.25. We conclude that applying all rules exhaustively takes $\mathcal{O}((|X| + \text{e-ret}_{\mathcal{N}}) \cdot (m + n \log n) \cdot \log m \cdot \log \max_\lambda)$ time, which can be summarized as $\mathcal{O}(m^2 \log^2 m \cdot \log \max_\lambda)$.

Size. Each application of Reduction Rule 6.16 increases the number of reticulation-edges by one. Therefore, observe that with e-ret$_{\mathcal{N}}$ we refer to the parameter in the original network.

Observe that none of the reduction rules (except for Reduction Rule 6.16) change the number of reticulations or core vertices in the network. Reduction Rule 6.16 may turn some core vertices into reticulations but otherwise does not create new core vertices or reticulations. Therefore, we have $|R^*|+|Q^*| = |R|+|Q| \in \mathcal{O}(\text{e-ret}_{\mathcal{N}})$ by Observation 6.13.

As Reduction Rule 6.16 is exhaustively applied, we have that each side-path in $\mathcal{N}^*$ not leaving the root has at most one internal vertex, and this vertex has one leaf child. There are $\mathcal{O}(\text{e-ret}_{\mathcal{N}})$ side-paths in $\mathcal{N}^*$ by Observation 6.13 and the fact that none of the reduction rules increase the number of side-paths not leaving the root. Thus, we have that the total number of vertices reachable from $R^* \cup (Q \setminus \rho)$ is at most $|R^*| + |Q^*| + \mathcal{O}(\text{e-ret}_{\mathcal{N}}) = \mathcal{O}(\text{e-ret}_{\mathcal{N}})$. That is, $|B^*| \in \mathcal{O}(\text{e-ret}_{\mathcal{N}})$.

The size of Y^* is the number of paths from ρ to a core-vertex or a reticulation. There are at most $|Q| + \text{e-ret}_{\mathcal{N}}$ such paths in the original instance. Each application of Reduction Rule 6.16 adds one such path. We saw that $|Q| \in \mathcal{O}(\text{e-ret}_{\mathcal{N}})$ and Reduction Rule 6.16 can be applied $\mathcal{O}(\text{e-ret}_{\mathcal{N}})$ times. We conclude $|Y^*| \in \mathcal{O}(\text{e-ret}_{\mathcal{N}})$.

After Reduction Rule 6.20 has been applied exhaustively, we may conclude that $k^* \leq |B^*| + |Y^*| \in \mathcal{O}(\text{e-ret}_{\mathcal{N}})$. After Reduction Rule 6.22 has been applied exhaustively, there are at most k^* vertices in A^* and after Reduction Rule 6.24 has been applied exhaustively, each path in Z^* has length of most $k^* - 1$ such that $|Y^*| + |Z^*| \leq |Y^*| \cdot k^* \in \mathcal{O}(\text{e-ret}_{\mathcal{N}}{}^2)$. We conclude $|V^*| \in \mathcal{O}(\text{e-ret}_{\mathcal{N}}{}^2)$.

We have $\text{e-ret}_{\mathcal{N}^*} \leq \text{e-ret}_{\mathcal{N}} + |Q^*| + |R^*| \in \mathcal{O}(\text{e-ret}_{\mathcal{N}})$ because $\text{e-ret}_{\mathcal{N}} = |E| - |V|$ in any network. Hence, we conclude $|E^*| = \text{e-ret}_{\mathcal{N}^*} + |V^*| \in \mathcal{O}(\text{e-ret}_{\mathcal{N}}{}^2)$. □

From Theorem 6.7, we have that in polynomial time we can reduce any instance $\mathcal{I}$ of MAPPD to an equivalent instance $\mathcal{I}^* = (\mathcal{N}^* = (V^*, E^*, \lambda^*), k^*, D^*)$ in which $|V^*|$, $|E^*|$, and k^* are all bounded by a polynomial in e-ret$_{\mathcal{N}}$. This does not guarantee a polynomial kernel, as the encoding size of D^* or $\max_{\lambda^*}$ could be much larger than $|V^*|$ or $|E^*|$. Fortunately, we can apply a result of [EKMR17, FT87] to bound these values, as follows.

Let $e_1, \ldots, e_{m^*}$ be an order of the edges after applying all reduction rules. We define $w_i := \lambda^*(e_i)$ for each $i \in [m^*]$ and $W := D^*$. In polynomial time, we can compute positive numbers $w'_1, \ldots, w'_{m^*}$, and W' such that the total encoding length is $\mathcal{O}((m^*)^4) \in \mathcal{O}(\text{e-ret}_{\mathcal{N}}{}^8)$ with $\sum_{i \in S} w_i \geq W$ if and only if $\sum_{i \in S} w'_i \geq W'$ for every $S \subseteq [m^*]$ by [EKMR17, Corollary 2].

We directly conclude the following.

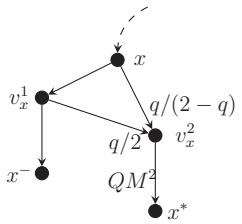


Figure 6.7: Illustration of the leaf-gadget. Omitted edge-weights are 1 and $q(x)$ is abbreviated to q.

Theorem 6.8. MAPPD *admits a polynomial size kernelization algorithm for the number of reticulation-edges* e-ret$_{\mathcal{N}}$.

6.6 Hardness of Max-Net-PD

In this section, we examine another measure of phylogenetic diversity on phylogenetic networks than MAPPD, namely MAX-NET-PD. Recall that in MAX-NET-PD, each reticulation-edge is given an inheritance proportion. Similar to the computation of diversity in GNAP, also in MAX-NET-PD we consider *expected* phylogenetic diversity.

In this section, we present a polynomial-time reduction from UNIT-COST-NAP to MAX-NET-PD, in which the level of the phylogenetic network is 1. Recall that in UNIT-COST-NAP, we are given a phylogenetic tree, a survival probability for every taxon, and two integers k and D, and it is asked whether there is a set S of size at most k taxa such that has an expected phylogenetic diversity of at least D.

By Theorem 3.5, UNIT-COST-NAP is NP-hard even if the phylogenetic tree has a height of 2. We conclude the following.

Theorem 6.9. MAX-NET-PD *is* NP*-hard even if the input network has a level of 1 and the distance between the root and each leaf is 4.*

Proof. By Theorem 3.5, UNIT-COST-NAP is NP-hard on trees with a height of 2. Let $\mathcal{T}$ be an X-tree with a height of 2 for some set of taxa X and let an instance $\mathcal{I} = (\mathcal{T}, \lambda, q, k, D)$ of UNIT-COST-NAP be given.

We define a leaf-gadget which is illustrated in Figure 6.7. Let $x \in X$ be a leaf of $\mathcal{T}$ with survival probability $q(x)$. Add four vertices v_x^1, v_x^2, x^*, and x^-, and edges xv_x^1, xv_x^2, $v_x^1v_x^2$, $v_x^1x^-$, and $v_x^2x^*$. The only reticulation in this gadget is v_x^2 with incoming edges xv_x^2 and $v_x^1v_x^2$. We set inheritance proportions of these reticulation

edges $p(xv_x^2) := q(x)/(2-q(x))$ and $p(v_x^1v_x^2) := q(x)/2$ which are both in $\mathbb{R}_{[0,1]}$ because $q(x) \in \mathbb{R}_{[0,1]}$.

Let $\mathcal{N}$ be the network which results from replacing each leaf of $\mathcal{T}$ with the corresponding leaf-gadget. The leaves of $\mathcal{N}$ are $X' := \{x^*, x^- \mid x \in X\}$. Let d denote the largest denominator of a survival probability $q(x)$ for some $x \in X$, so that every $q(x)$ is expressible as c'/d' for some pair of integers c, and d such that $d' \leq d$. Let M and Q be large integers, such that M is bigger than $\mathrm{PD}_{\mathcal{T}}(X) \geq |X| \geq k$, and $Q \cdot D$ and $Q \cdot d^{-k}$ are both bigger than 3.

Observe that the number of bits necessary to write M and Q is polynomial in the size of $\mathcal{I}$. We set the weight of edges $e \in E(\mathcal{T})$ in $\mathcal{N}$ to $\lambda'(e) = kQ \cdot \lambda(e)$. For each taxon $x \in X$, set $\lambda'(v_x^2x^*) := Q \cdot M^2$ and $\lambda'(e) := 1$ for each $e \in \{xv_x^1, xv_x^2, v_x^1v_x^2, v_x^1x^-\}$.

Finally, let $\mathcal{I}' := (\mathcal{N}, \lambda', p, k, D' := kQ(M^2 + D))$ be an instance of Max-Net-PD. Each leaf-gadget is a level-1-network. As the leaf-gadgets are connected by a tree, $\mathcal{N}$ is a level-1-network. Recall that the height of the tree $\mathcal{T}$ is 2, and as such the distance between the root and each leaf in in $\mathcal{N}$ is 4.

We first show that $\gamma_Z^p(e) = q(x)$ for the edge e incoming at x, if Z contains x^* but not x^-. Indeed, because $x^- \notin Z$, we conclude $\gamma_Z^p(xv_x^2) = p(xv_x^2) = q(x)/(2-q(x))$ and $\gamma_Z^p(xv_x^1) = \gamma_Z^p(v_x^1v_x^2) = p(v_x^1v_x^2) = q(x)/2$. Subsequently,

$$\begin{aligned}
\gamma_Z^p(e) &= 1 - \left(1 - \gamma_Z^p(xv_x^1)\right)\left(1 - \gamma_Z^p(xv_x^2)\right) \\
&= 1 - \frac{2-q(x)}{2} \cdot \frac{2-2q(x)}{2-q(x)} \\
&= 1 - \frac{2-q(x)}{2-q(x)} \cdot \frac{2-2q(x)}{2} \\
&= 1 - \frac{2-2q(x)}{2} = q(x).
\end{aligned} \tag{6.14}$$

We now show that if $\mathcal{I}$ is a `yes`-instance of unit-cost-NAP, then $\mathcal{I}'$ is a `yes`-instance of Max-Net-PD. Afterward, we show the converse.

Suppose that $\mathcal{I}$ is a `yes`-instance of unit-cost-NAP and that $S \subseteq X$ is a solution of $\mathcal{I}$, that is $|S| \leq k$ and $\mathrm{PD}_{\mathcal{T}}(S) \geq D$. Let $S' := \{x^* \mid x \in S\}$ be a subset of X'. Clearly, $|S'| = |S| \leq k$. Because $\mathcal{T}$ does not contain reticulation edges and $\gamma_Z^p(e) = q(x)$ with e being the edge incoming at x, we conclude that

$$\text{Net-PD}_{\mathcal{N}}(S') \geq kQ \cdot \mathrm{PD}_{\mathcal{T}}(S) + k \cdot \lambda'(v_x^2x^*) \geq kQ \cdot (D + M^2) = D'. \tag{6.15}$$

Hence, S' is a solution of $\mathcal{I}'$.

Conversely, let S' be a solution of $\mathcal{I}'$. We define $S^- = S \cap \{x^- \mid x \in X\}$ and $S^* = S \cap \{x^* \mid x \in X\}$. Towards a contradiction, assume that S^- is non-empty.

Then however, using $3 < Q \cdot D$,

$$\begin{aligned}\text{Net-PD}_{\mathcal{N}}(S') &\leq \sum_{x^- \in S^-} (\lambda'(v_x^1 x^-) + \lambda'(x v_x^1)) + |S^*| \cdot (QM^2 + 3) + \sum_{e \in E(\mathcal{T})} \lambda'(e) \\ &\leq 2 \cdot |S^-| + |S^*| \cdot (QM^2 + 3) + kQM \\ &\leq 2 + (k-1)(QM^2 + 3) + kQM \\ &< kQM^2 - QM^2 + kQM + 3k \\ &< k(QM^2 + 3) < k(QM^2 + QD) = D'\end{aligned}$$

contradicts that S' is a solution. Therefore, we conclude that $S' \subseteq \{x^* \mid x \in L\}$ and we assume that the size of S' is k. Define $S := \{x \mid x^* \in S'\}$. Subsequently, with Equation (6.14) we conclude

$$\begin{aligned}kQ(M^2 + D) = D' &\leq \text{Net-PD}_{\mathcal{N}}(S') \\ &= k \cdot QM^2 + \sum_{x \in S} \Bigg(\underbrace{\frac{q(x)}{2} + \frac{q(x)}{2} + \frac{q(x)}{2 - q(x)}}_{\leq 3} \Bigg) + kQ \cdot \text{PD}_{\mathcal{T}}(S).\end{aligned}$$

It follows that $\text{PD}_{\mathcal{T}}(S) \geq \frac{1}{kQ} \cdot (kQ(M^2 + D) - kQM^2 - 3k) = D - 3/Q$.

It remains to show that $\text{PD}_{\mathcal{T}}(S)$ cannot take any values between $D - 3/Q$ and D, and therefore $\text{PD}_{\mathcal{T}}(S) \geq D - 3/Q$ implies $\text{PD}_{\mathcal{T}}(S) \geq D$. To this end, let c_x, and d_x be the unique positive integers such that $q(x) = c_x/d_x$ for each taxon $x \in X$. Then, $q(x)$ and $(1 - q(x))$ are multiples of $1/d_x$, by construction. It follows that for any edge e of the tree $\mathcal{T}$ that $\gamma'_S(e) = (1 - \prod_{x \in \text{off}(e) \cap S}(1 - q(x)))$ is a multiple of $1/(\prod_{x \in S} d_x)$. As all edge-weights are integers, we also have that $\text{PD}_{\mathcal{T}}(S)$ is a multiple of $1/(\prod_{x \in S} d_x)$. It follows that either $\text{PD}_{\mathcal{T}}(S) \geq D$ or $D - \text{PD}_{\mathcal{T}}(S) \geq 1/(\prod_{x \in S} d_x)$. Because $d_x \leq d$ for any x, this difference is at least $d^{-k} > 3/Q$. It follows that if $\text{PD}_{\mathcal{T}}(S) \geq D - 3/Q$ then in fact $\text{PD}_{\mathcal{T}}(S) \geq D$.

We conclude $\text{PD}_{\mathcal{T}}(S) \geq D$. And therefore with $|S| = |S'| = k$ we follow that S is a solution of $\mathcal{I}$. Hence, $\mathcal{I}$ is a `yes`-instance of UNIT-COST-NAP. □

6.7 Discussion

In this chapter, we considered two problems, MAX-ALL-PATHS-PD and MAX-NET-PD, in maximizing phylogenetic diversity in networks, and analyzed them within the framework of parameterized complexity.

We showed that MAPPD is W[2]-hard parameterized by k, the size of the solution. We further were able to show an equivalence between MAPPD parameterized by k

and Item-Weighted Partial Set Cover parameterized by the size of the solution. Thus, establishing the exact complexity class of Item-Weighted Partial Set Cover, would also establish the exact complexity class of MapPD. On the positive side, we showed that MapPD is FPT when parameterized with the number of reticulations $\text{ret}_{\mathcal{N}}$ and with respect to the treewidth $\text{tw}_{\mathcal{N}}$ of the network. We further showed that MapPD admits a kernelization algorithm with respect to the number of reticulation-edges $\text{e-ret}_{\mathcal{N}}$. Finally, we have shown that Max-Net-PD remains NP-hard on phylogenetic networks with a level of 1.

Because $\text{ret}_{\mathcal{N}}$ is smaller than $\text{e-ret}_{\mathcal{N}}$, it is natural to ask if the kernelization result can also be shown for $\text{ret}_{\mathcal{N}}$. We also ask whether MapPD parameterized by k is W[2]-complete.

For Max-Net-PD we also raise some questions. We have proven that MapPD is FPT when parameterized by the threshold of diversity D and the acceptable loss of diversity $\overline{D}$. Also, we showed that MapPD admits a kernelization algorithm of polynomial size with respect to $\text{e-ret}_{\mathcal{N}}$. It is natural to ask, if these positive results transfer to Max-Net-PD. Further, based on the presented hardness for Max-Net-PD we ask for some improvements. First of all, can Max-Net-PD be solved in pseudo-polynomial time on level-1-networks? Secondly, is Max-Net-PD polynomial time solvable on level-1-networks if we require the network to be ultrametric, i.e. when all root-leaf paths have the same length? Finally, does Max-Net-PD remain W[1]-hard when the parameter is the number of species k to save plus the level of the network?

Chapter 7

Conclusion

We studied several problems concerning the maximization of preserved phylogenetic diversity within the constraint of limited resources, which allow for the selection of only a few taxa. The considered problem definitions model biological processes more realistically compared to the basic problem of maximization phylogenetic diversity, Max-PD. For these problem definitions, within the framework of parameterized algorithms, we presented several lower and upper running time bounds for algorithms.

Before presenting a broader perspective on this work, we will briefly review the different problem definitions and the results obtained. Finally, we will offer an outlook on potential future research directions.

7.1 Summary of Problems and Results

The first problem we considered is Generalized Noah's Ark Problem, in Chapter 3. With GNAP one is able to model different costs for saving taxa and also an uncertainty as to whether an intervention actually saves a taxon.

We showed that GNAP is XP with respect to the number of unique costs plus the number of unique survival probabilities. We further proved that GNAP is W[1]-hard with respect to the number of taxa. We also showed that unit-cost-NAP, the special case of GNAP in which all projects with a positive survival probability have a cost of 1, is NP-hard even on phylogenetic trees with a height of 2. It remains open if GNAP, or even unit-cost-NAP, is strongly NP-hard or if GNAP can be solved in pseudo-polynomial running time.

In Chapter 4 we considered Time-PD and s-Time-PD. These problems are motivated by the concern that some taxa need to be treated earlier than others

because not all taxa go extinct at the same point in time. As it is necessary for a solution of an instance of either of the two problems to indicate how to schedule the available time in order to save all of these taxa, a connection to the field of scheduling is given.

Both problems, Time-PD and s-Time-PD, are FPT with respect to the target diversity D. Further, Time-PD is also FPT when parameterized by the acceptable loss of phylogenetic diversity $\overline{D}$. In contrast, this result does not hold for s-Time-PD, unless P $\neq$ NP. It remains open whether Time-PD is solvable in pseudo-polynomial time.

With Max-PD it is not possible to check whether taxa in a selected set have crucial dependencies to taxa which are not selected. In Optimizing PD with Dependencies, which we studied in Chapter 5, one not only searches for a phylogenetically diverse set of taxa but also for one where in the given food-web each taxon either is a source or has an edge from another saved taxon. Therefore, with PDD it is possible to compute a small set of taxa that has a phylogenetic diversity above a certain threshold and each selected is self-sufficient in the ecological system or feeds on the saved set of taxa.

We presented algorithms with a color-coding approach to show that PDD is FPT when parameterized with the solution size plus the height of the phylogenetic tree. s-PDD—the special case of PDD in which the phylogenetic tree is a star—is W[1]-hard when parameterized by the acceptable loss of phylogenetic diversity. We also considered the structure of the food-web. Here, we proved that PDD remains NP-hard if every connected component of the food-web is a star or a clique. However, PDD is FPT with respect to the distance to co-cluster of the food-web. We showed further that s-PDD is FPT with respect to the treewidth of the food-web. It is open whether PDD is FPT with respect to the size of the solution, or if PDD can be solved in polynomial time if every connected component in the food-web contains at most two vertices.

In some constellations, the evolutionary history of taxa may be explained better with a phylogenetic network than with a phylogenetic tree. Models for generalizing phylogenetic diversity on explicit phylogenetic networks have been defined rather recently [WF18, BSW22]. In Chapter 6, we considered two corresponding problems, Max-All-Paths-PD and Max-Net-PD.

We presented reductions to showcase an equivalence between MapPD and a special case of Set Cover, both parameterized by the size of the solution and thereby establishing a W[2]-hardness of MapPD with respect to k. We further showed that MapPD is FPT when parameterized with the number of reticulations or with the treewidth of the network. Finally, Max-Net-PD remains NP-hard on phylogenetic

networks of level 1. These results have the practical implications that exact solutions can be computed in a reasonable time if the phylogenetic network is relatively treelike. It remains open whether MAPPD admits a kernelization of polynomial size for $\text{ret}_{\mathcal{N}}$. Further, we wonder if MAX-NET-PD is FPT when parameterized by k plus the level of the phylogenetic network.

7.2 A Broader View on Our Results

Let us now provide a broader view of the results of this work. Firstly, we want to observe that the regarded problems can be distinguished between two categories. While BUDGETED NAP, TIME-PD, and S-TIME-PD are primarily extending MAX-PD in a direction to better model interventions of men, PDD, MAPPD, and MAX-NET-PD solely model biological processes.

The algorithms that we presented are predominantly dynamic programming algorithms or use the technique of color-coding. It is rather not surprising to have a lot of dynamic programming approaches when dealing with a tree-structure. In the color-coding algorithms, we mostly used perfect hash-families and sometimes universal sets. While these techniques provide the desired FPT-results, representative sets [CFK$^+$15, Mon85, Chapter 12.3] actually yield faster running times for other problems. We, therefore, ask whether it is possible to apply representative sets to at least improve some of the running times.

We observe that certain hardness results, both NP-hardness and W[i]-hardness, were mostly given even on phylogenetic trees or networks that only have a constant height. On the contrary, it seemed that if a certain algorithm works on trees of height three or four, then the approach also generalizes to arbitrary heights. As phylogenetic trees tend to have significant height in practice, this is a rather positive observation because some algorithmic ideas are not strongly impeded by tall trees.

The UNIT-COST-NAP problem presented itself as an interesting problem for reductions. This is for example utilized in Theorem 6.9. We have been able to prove that UNIT-COST-NAP is NP-hard in Theorem 3.5. However, we did not succeed in presenting a pseudo-polynomial running-time algorithm for UNIT-COST-NAP, nor in showing that UNIT-COST-NAP is strongly NP-hard. If UNIT-COST-NAP was strongly NP-hard, then we could also exclude pseudo-polynomial running-time algorithms for GNAP and MAX-NET-PD restricted to level-1-networks, unless P = NP.

We now have a look into some specific parameters considered in this work. The parameterization by the acceptable loss of phylogenetic diversity, $\overline{D}$, is to the best of our knowledge introduced by us. Considering the fact that we are probably interested in preserving the majority of available phylogenetic diversity, we may assume $\overline{D}$ to be

relatively small in real-world instances. Considering $\overline{D}$ as a parameter proved to be fruitful, as Time-PD and MapPD are FPT with respect to $\overline{D}$. Both these algorithms use the technique of color-coding. Theorem 4.2, in which we prove that Time-PD is FPT when parameterized with $\overline{D}$, is arguably the most technically advanced algorithm in this thesis.

Most of the considered problems are W[1]-hard with respect to the size of the solution. This results from the fact that these problems are generalizations of Knapsack or Set Cover, which are known to be W[1]-hard for the solution size. Further, most of the problems can be solved with trivial $\mathcal{O}(2^{|X|})$ running-time algorithms. One can wonder if any improvements can be made to these algorithms. At least for MapPD, we know that the algorithm running in $\mathcal{O}(2^{|X|})$ time is tight under SETH, by Corollary 6.4.

7.3 Research Ideas Based on This Work

In the remainder of the discussion, we want to delve a bit into general ideas for further research. Within this thesis, we presented many algorithmic ideas and discussed their correctness and running time upper bounds. Given the theoretical nature of this work, it would be valuable to evaluate the practical efficiency of these algorithms. Therefore, one would need to implement some of the algorithmic ideas to evaluate their practical efficiency. For instance, the branching algorithm for MapPD (Theorem 6.5) and the algorithms for the parameterization with D (Theorems 4.1 and 5.3) or with $\overline{D}$ (Theorems 4.2 and 6.4) would be recommendable. Implemented algorithms could do both, give further insight into the complexity of the problems, and also act as a bridge between theory and practical decision-making in conservation projects.

In some real-world preservation actions, decision-makers choose to preserve natural reservoirs instead of specific taxa, for example be due to political pressure. In Optimizing Diversity with Coverage (ODC), one may select regions for protection in which all taxa are preserved [MSS07]. ODC is NP-hard by a reduction from Set Cover [MSS07] and a generalization has been studied from an approximation point-of-view [BS08, BS11]. It would be interesting to see whether ODC is FPT for reasonable parameters.

Naturally, one could ask to combine the aspects of the individual problems. Given the presented individual hardness results for the problems, this attempt seems rather not so fruitful—and extremely technical. Nevertheless, the consideration of ecological constraints, as done in PDD, is indeed important to consider in conservation interventions. It could present an interesting study to combine these viability con-

straints with phylogenetic networks. Do the resulting problems become much harder than PDD?

In Chapter 6, we considered two existent definitions of phylogenetic diversity on phylogenetic networks. We have not discussed how well these problems model biological processes. One of the most important tasks in this field now would be to assess which variants of phylogenetic diversity on networks are biologically most accurate. This could of course depend on the type of species considered and in particular on the type of reticulate evolutionary events. Even if the maximization problem cannot be solved efficiently, having a good measure of phylogenetic diversity can still be of great practical use by measuring how diverse a given set of species is.

Finally, we observe that we have considered problems in maximizing phylogenetic diversity. When it comes to the preservation of taxa we are arguably very interested in the different features of a selected set of taxa and therefore rather in feature or functional diversity. Computing functional diversity, however, is even impossible in cases [MPC+18] and so we use phylogenetic diversity as somewhat fitting albeit imperfect proxy of functional diversity [WDS13]. We wonder if in special cases parameterized algorithms could be a tool to handle the intractability. Further, it has been reported that maximizing phylogenetic diversity is only marginally better than selecting a random set of species when it comes to maximizing the functional diversity of the surviving species [MPC+18]. The situation could be different, however, when some constraints of this thesis, especially the biological constraints considered in the last two chapters, are incorporated. Here, investigating the following two questions seems fruitful: First, do randomly selected viable species sets have a higher functional diversity than randomly selected species? Second, do viable sets with maximal phylogenetic diversity have a higher functional diversity than randomly selected viable sets?

Bibliography

[AB04] Arthur O. L. Atkin and Daniel J. Bernstein. Prime sieves using binary quadratic forms. *Mathematics of Computation*, 73:1023–1030, January 2004. (Cited on p. 51)

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009. (Cited on p. 32)

[AMO95] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows: theory, algorithms and applications*. Prentice Hall, 1995. (Cited on pp. 150, 152)

[AYZ95] Noga Alon, Raphael Yuster, and Uri Zwick. Color-Coding. *Journal of the Association for Computing Machinery (JACM)*, 42(4):844–856, 1995. (Cited on p. 40)

[BBC23] BBC. Biodiversity: Almost half of animals in decline, research shows, 2023. BBC: `https://www.bbc.com/news/uk-northern-ireland-65681648`, visited on June 3rd, 2024. (Cited on p. 17)

[BH80] Terry Beyer and Sandra Mitchell Hedetniemi. Constant Time Generation of Rooted Trees. *SIAM Journal on Computing*, 9(4):706–712, 1980. (Cited on p. 133)

[Bil13] Alain Billionnet. Solution of the Generalized Noah's Ark Problem. *Systematic Biology*, 62(1):147–156, 2013. (Cited on pp. 21, 59)

[Bil17] Alain Billionnet. How to Take into Account Uncertainty in Species Extinction Probabilities for Phylogenetic Conservation Prioritization. *Environmental Modeling & Assessment*, 22(6):535–548, 2017. (Cited on pp. 21, 59)

[BJK14] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM Journal on Discrete Mathematics*, 28(1):277–305, 2014. (Cited on p. 39)

[Blä03] Markus Bläser. Computing small partial coverings. *Information Processing Letters*, 85(6):327–331, 2003. (Cited on p. 172)

[BS08] Magnus Bordewich and Charles Semple. Nature Reserve Selection Problem: A Tight Approximation Algorithm. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 5(2):275–280, 2008. (Cited on p. 200)

[BS11] Magnus Bordewich and Charles Semple. Budgeted Nature Reserve Selection with diversity feature loss and arbitrary split systems. *Journal of Mathematical Biology*, 64:69–85, 02 2011. (Cited on p. 200)

[BSS09] Magnus Bordewich, Charles Semple, and Andreas Spillner. Optimizing phylogenetic diversity across two trees. *Applied Mathematics Letters*, 22(5):638–641, 2009. (Cited on pp. 149, 151)

[BSS23a] Matthias Bentert, Jannik Schestag, and Frank Sommer. On the Complexity of Finding a Sparse Connected Spanning Subgraph in a Non-Uniform Failure Model. In *Proceedings of the 18th International Symposium on Parameterized and Exact Computation (IPEC 2023)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2023. (Cited on p. X)

[BSS23b] Matthias Bentert, Jannik Schestag, and Frank Sommer. On the complexity of finding a sparse connected spanning subgraph in a non-uniform failure model. *arXiv preprint arXiv:2308.04575*, 2023. (Cited on p. X)

[BSW22] Magnus Bordewich, Charles Semple, and Kristina Wicke. On the complexity of optimising variants of phylogenetic diversity on phylogenetic networks. *Theoretical Computer Science*, 917:66–80, 2022. (Cited on pp. 24, 25, 161, 162, 164, 165, 166, 168, 178, 198)

[BV04] Mukul Subodh Bansal and Vadlamudi China Venkaiah. Improved Fully Polynomial time Approximation Scheme for the 0-1 Multiple-choice Knapsack Problem. *International Institute of Information Technology Tech Report*, pages 1–9, 2004. (Cited on p. 43)

[CDL+16] Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On Problems as Hard as CNF-SAT. *ACM Transactions on Algorithms (TALG)*, 12(3):1–24, 2016. (Cited on p. 168)

[CDRG+18] Alyssa R. Cirtwill, Giulio Valentino Dalla Riva, Marilia P. Gaiarsa, Malyon D. Bimler, E. Fernando Cagua, Camille Coux, and D. Matthias Dehling. A review of species role concepts in food webs. *Food Webs*, 16:e00093, 2018. (Cited on pp. 122, 160)

[CFK+15] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. (Cited on pp. 32, 38, 39, 40, 98, 137, 138, 155, 199)

[CKvHM16] Olga Chernomor, Steffen Klaere, Arndt von Haeseler, and Bui Quang Minh. *Split Diversity: Measuring and Optimizing Biodiversity using Phylogenetic Split Networks*, pages 173–195. Springer International Publishing, Cham, 2016. (Cited on p. 24)

[CME17] Eileen Crist, Camilo Mora, and Robert Engelman. The interaction of human population, food production, and biodiversity protection. *Science*, 356(6335):260–264, 2017. (Cited on p. 17)

[CNP+22] Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan M.M. Van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. *ACM Transactions on Algorithms (TALG)*, 18(2):1–31, 2022. (Cited on p. 30)

[Cro97] Rossiter H. Crozier. Preserving the information content of species: Genetic Diversity, Phylogeny, and Conservation Worth. *Annual Review of Ecology and Systematics*, 28(1):243–268, 1997. (Cited on pp. 18, 19, 59, 121)

[DF95a] Rodney G. Downey and Michael R. Fellows. Fixed-Parameter Tractability and Completeness I: Basic Results. *SIAM Journal on Computing*, 24(4):873–921, 1995. (Cited on pp. 36, 47)

[DF95b] Rodney G. Downey and Michael R. Fellows. Fixed-Parameter Tractability and Completeness II: On Completeness for W[1].

Theoretical Computer Science, 141(1-2):109–131, 1995. (Cited on pp. 36, 41, 92, 139, 168)

[DF13] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. (Cited on pp. 32, 36, 38, 41, 168)

[Die12] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012. (Cited on p. 28)

[DJ22] Katy Daigle and Julia Janicki. Extinction crisis puts 1 million species on the brink, 2022. Reuters: `https://www.reuters.com/lifestyle/science/extinction-crisis-puts-1-million-species-brink-2022-12-23/8`, visited on June 3rd, 2024. (Cited on p. 17)

[DLS14] Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Kernelization Lower Bounds Through Colors and IDs. *ACM Transactions on Algorithms (TALG)*, 11(2):1–20, 2014. (Cited on pp. 41, 137)

[Dus99] Pierre Dusart. The k^{th} prime is greater than $k(\ln k + \ln \ln k - 1)$ for $k \geq 2$. *Mathematics of Computation*, 68(225):411–415, 1999. (Cited on p. 51)

[EKMR17] Michael Etscheid, Stefan Kratsch, Matthias Mnich, and Heiko Röglin. Polynomial kernels for weighted problems. *Journal of Computer and System Sciences*, 84:1–10, 2017. (Cited on pp. 43, 73, 74, 184, 192)

[Fai92] Daniel P. Faith. Conservation evaluation and phylogenetic diversity. *Biological Conservation*, 61(1):1–10, 1992. (Cited on pp. 18, 19, 59, 176)

[FFHV11] Michael R. Fellows, Guillaume Fertin, Danny Hermelin, and Stéphane Vialette. Upper and lower bounds for finding connected motifs in vertex-colored graphs. *Journal of Computer and System Sciences*, 77(4):799–811, 2011. (Cited on p. 137)

[FG06] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. (Cited on p. 32)

[FGR12] Michael R. Fellows, Serge Gaspers, and Frances A. Rosamond. Parameterizing by the Number of Numbers. *Theory of Computing Systems*, 50(4):675–693, 2012. (Cited on p. 62)

[FJ15] Michael Fuchs and Emma Yu Jin. Equality of Shapley value and fair proportion index in phylogenetic trees. *Journal of Mathematical Biology*, 71:1133–1147, 2015. (Cited on p. 161)

[FR12] Daniel P. Faith and Zoe T. Richards. Climate Change Impacts on the Tree of Life: Changes in Phylogenetic Diversity Illustrated for Acropora Corals. *Biology*, 1(3):906–932, 2012. (Cited on p. 19)

[FSW11] Beáta Faller, Charles Semple, and Dominic Welsh. Optimizing Phylogenetic Diversity with Ecological Constraints. *Annals of Combinatorics*, 15(2):255–266, 2011. (Cited on pp. 22, 23, 122, 123, 137, 140, 144, 145, 149, 155, 157)

[FT87] András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987. (Cited on pp. 42, 184, 192)

[GBP09] Philippe Gambette, Vincent Berry, and Christophe Paul. The structure of level-k phylogenetic networks. In *Proceedings of the Annual Symposium on Combinatorial Pattern Matching*, pages 289–300. Springer, 06 2009. (Cited on pp. 185, 186)

[GCJW+15] Pille Gerhold, James F. Cahill Jr., Marten Winter, Igor V. Bartish, and Andreas Prinzing. Phylogenetic patterns are not proxies of community assembly mechanisms (they are far better). *Functional Ecology*, 29(5):600–614, 2015. (Cited on p. 19)

[GJ78] Michael R. Garey and David S. Johnson. "Strong" NP-Completeness Results: Motivation, Examples, and Implications. *Journal of the ACM*, 25(3):499–508, 1978. (Cited on p. 93)

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completenes*. W. H. Freeman, 1979. (Cited on pp. 32, 50)

[GJS76] Michael R. Garey, David S. Johnson, and Ravi Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2):117–129, 1976. (Cited on pp. 88, 93)

[GKLS24] Jaroslav Garvardt, Christian Komusiewicz, Ber Lorke, and Jannik Schestag. Protective and Nonprotective Subset Sum Games: A Parameterized Complexity Analysis. In *Proceedings of the 8th International*

Conference on Algorithmic Decision Theory (ADT 2024). Springer, 2024. (Cited on p. X)

[GLLK79] Ronald Lewis Graham, Eugene Leighton Lawler, Jan Karel Lenstra, and AHG Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of Discrete Mathematics*, volume 5, pages 287–326. Elsevier, 1979. (Cited on p. 86)

[GRR19] Frank Gurski, Carolin Rehs, and Jochen Rethmann. Knapsack problems: A parameterized point of view. *Theoretical Computer Science*, 775:93–108, 2019. (Cited on pp. 117, 141, 173, 175)

[GRSW23] Jaroslav Garvardt, Malte Renken, Jannik Schestag, and Mathias Weller. Finding degree-constrained acyclic orientations. In *Proceedings of the 18th International Symposium on Parameterized and Exact Computation (IPEC 2023)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2023. (Cited on p. X)

[Har13] Klaas Hartmann. The equivalence of two Phylogenetic Diodiversity measures: the Shapley value and Fair Proportion index. *Journal of Mathematical Biology*, 67:1163–1170, 2013. (Cited on p. 19)

[HB06] Daniel H. Huson and David Bryant. Application of Phylogenetic Networks in Evolutionary Studies. *Molecular Biology and Evolution*, 23(2):254–267, 2006. (Cited on pp. 23, 24)

[HH24] Klaus Heeger and Danny Hermelin. Minimizing the Weighted Number of Tardy Jobs is W[1]-hard. *arXiv preprint arXiv:2401.01740*, 2024. (Cited on pp. 86, 88, 119)

[HHS23] Klaus Heeger, Danny Hermelin, and Dvir Shabtay. Single Machine Scheduling with Few Deadlines. In *Proceedings of the 18th International Symposium on Parameterized and Exact Computation (IPEC 2023)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2023. (Cited on p. 87)

[Hic20] Jason Hickel. Quantifying national responsibility for climate breakdown: an equality-based attribution approach for carbon dioxide emissions in excess of the planetary boundary. *The Lancet Planetary Health*, 4(9):e399–e404, 2020. (Cited on p. 17)

[HKPS21] Danny Hermelin, Shlomo Karhi, Michael Pinedo, and Dvir Shabtay. New algorithms for minimizing the weighted number of tardy jobs on a single machine. *Annals of Operations Research*, 298:271–287, 2021. (Cited on pp. 86, 87, 115)

[HKS08] Claus-Jochen Haake, Akemi Kashiwada, and Francis Edward Su. The Shapley value of phylogenetic trees. *Journal of Mathematical Biology*, 56(4):479–497, 2008. (Cited on p. 19)

[HRS10] Daniel H. Huson, Regula Rupp, and Celine Scornavacca. *Phylogenetic Networks: Concepts, Algorithms and Applications*. Cambridge University Press, 2010. (Cited on pp. 24, 161)

[HS06] Klaas Hartmann and Mike Steel. Maximizing Phylogenetic Diversity in Biodiversity Conservation: Greedy Solutions to the Noah's Ark Problem. *Systematic Biology*, 55(4):644–651, 2006. (Cited on pp. 20, 59, 92, 121)

[HS07] Klaas Hartmann and Mike Steel. Phylogenetic diversity: from combinatorics to ecology. *Reconstructing Evolution: New Mathematical and Computational Advances.*, 2007. (Cited on p. 59)

[HWVW95] Christopher J. Humphries, Paul H. Williams, and Richard I. Vane-Wright. Measuring biodiversity value for conservation. *Annual review of Ecology and Systematics*, 26(1):93–111, 1995. (Cited on p. 18)

[IPCC23] IPCC 2023; Core Writing Team, H. Lee and J. Romero (eds.). Climate Change 2023: Synthesis Report. Contribution of Working Groups I, II and III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change. 2023. (Cited on p. 17)

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. (Cited on p. 39)

[ITC+07] Nick J.B. Isaac, Samuel T. Turvey, Ben Collen, Carly Waterman, and Jonathan E.M. Baillie. Mammals on the EDGE: Conservation Priorities Based on Threat and Phylogeny. *PLOS ONE*, 2(3):1–7, 03 2007. (Cited on p. 19)

[JS23a] Mark Jones and Jannik Schestag. How Can We Maximize Phylogenetic Diversity? Parameterized Approaches for Networks. *archive.org*, 2023. (Cited on p. VIII)

[JS23b] Mark Jones and Jannik Schestag. How Can We Maximize Phylogenetic Diversity? Parameterized Approaches for Networks. In *Proceedings of the 18th International Symposium on Parameterized and Exact Computation (IPEC 2023)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2023. (Cited on pp. VIII, 139, 159)

[JS24] Mark Jones and Jannik Schestag. Maximizing Phylogenetic Diversity under Time Pressure: Planning with Extinctions Ahead. *arXiv preprint arXiv:2403.14217*, 2024. (Cited on pp. VIII, 20, 22, 139, 159)

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a Symposium on the Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972. (Cited on pp. 39, 41, 42, 92, 141)

[KLB+17] Vikas Kumar, Fritjof Lammers, Tobias Bidon, Markus Pfenninger, Lydia Kolter, Maria A. Nilsson, and Axel Janke. The evolutionary History of Bears is characterized by Gene flow across Species. *Scientific Reports*, 7(1):46487, 2017. (Cited on p. 24)

[KLMS23] Christian Komusiewicz, Simone Linz, Nils Morawietz, and Jannik Schestag. On the Complexity of Parameterized Local Search for the Maximum Parsimony Problem. In *Proceedings of the 34th Annual Symposium on Combinatorial Pattern Matching (CPM 2023)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2023. (Cited on p. X)

[KPP04] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack Problems*. Springer, 2004. (Cited on pp. 42, 43, 47)

[KS23a] Christian Komusiewicz and Jannik Schestag. A Multivariate Complexity Analysis of the Generalized Noah's Ark Problem. *arXiv preprint arXiv:2307.03518*, 2023. (Cited on p. VII)

[KS23b] Christian Komusiewicz and Jannik 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*, pages 109–121. Springer, 2023. (Cited on pp. VII, IX, 48, 121)

[KS24a] Christian Komusiewicz and Jannik Schestag. Maximizing Phylogenetic Diversity under Ecological Constraints: A Parameterized Complexity Study. *arXiv preprint arXiv:2405.17314*, 2024. (Cited on p. VIII)

[KS24b] Christian Komusiewicz and Jannik 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. (Cited on p. VIII)

[LFS06] Vincent Lacroix, Cristina G. Fernandes, and Marie-France Sagot. Motif Search in Graphs: Application to Metabolic Networks. *IEEE/ACM transactions on computational biology and bioinformatics*, 3(4):360–368, 2006. (Cited on p. 137)

[LHN05] Erez Lieberman, Christoph Hauert, and Martin A. Nowak. Evolutionary dynamics on graphs. *Nature*, 433(7023):312–316, 2005. (Cited on p. 160)

[Lin42] Raymond L. Lindeman. The trophic-dynamic aspect of ecology. *Ecology*, 23(4):399–417, 1942. (Cited on p. 122)

[Lin19] Bingkai Lin. A simple gap-producing Reduction for the parameterized Set Cover Problem. *arXiv preprint arXiv:1902.03702*, 2019. (Cited on p. 168)

[LJ83] Hendrik Willem Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983. (Cited on p. 42)

[LLS15] Michael J. Lynch, Michael A. Long, and Paul B. Stretesky. Anthropogenic development drives species to be endangered: Capitalism and the decline of species. In *Green Harms and Crimes*, pages 117–146. Springer, 2015. (Cited on p. 17)

[LM69] Eugene L. Lawler and J. Michael Moore. A functional equation and its application to resource allocation and sequencing problems. *Management Science*, 16(1):77–84, 1969. (Cited on p. 86)

[Mar20] Luiz Marques. *Capitalism and Environmental Collapse*. Springer, 2020. (Cited on p. 17)

[Max70] William L. Maxwell. On sequencing n jobs on one machine to minimize the number of late jobs. *Management Science*, 16(5):295–297, 1970. (Cited on p. 86)

[May90] Robert M. May. Taxonomy as destiny. *Nature*, 347(6289):129–130, 1990. (Cited on pp. 17, 18)

[MBR+23] Joseph J. Merz, Phoebe Barnard, William E. Rees, Dane Smith, Mat Maroni, Christopher J. Rhodes, Julia H. Dederer, Nandita Bajaj, Michael K. Joy, Thomas Wiedmann, and Rory Sutherland. World scientists' warning: The behavioural crisis driving ecological overshoot. *Science Progress*, 106(3):00368504231201372, 2023. PMID: 37728669. (Cited on p. 17)

[Min86] Michel Minoux. Solving integer minimum cost flows with separable convex cost objective polynomially. *Netflow at Pisa*, pages 237–239, 1986. (Cited on pp. 150, 152)

[MKvH06] Bui Quang Minh, Steffen Klaere, and Arndt von Haeseler. Phylogenetic Diversity within Seconds. *Systematic Biology*, 55(5):769–773, 10 2006. (Cited on p. 20)

[MKvH07] Bui Quang Minh, Steen Klaere, and Arndt von Haeseler. Phylogenetic Diversity on Split Networks. Technical report, 12 2007. (Cited on p. 162)

[Moh01] Bojan Mohar. Face Covers and the Genus Problem for Apex Graphs. *Journal of Combinatorial Theory, Series B*, 82(1):102–117, 2001. (Cited on pp. 41, 144, 148)

[Mon85] Burkhard Monien. How to find long paths efficiently. In *North-Holland Mathematics Studies*, volume 109, pages 239–254. Elsevier, 1985. (Cited on p. 199)

[Moo68] J. Michael Moore. An n job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Science*, 15(1):102–109, 1968. (Cited on p. 86)

[Mor97] Bernard M. E. Moret. *The theory of computation.* Addison-Wesley, 1997. (Cited on p. 50)

[MP08] Georgina M. Mace and Andy Purvis. Evolutionary biology and practical conservation: bridging a widening gap. *Molecular Ecology*, 17(1):9–19, 2008. (Cited on p. 19)

[MPC+18] Florent Mazel, Matthew W. Pennell, Marc W. Cadotte, Sandra Diaz, Giulio Valentino Dalla Riva, Richard Grenyer, Fabien Leprieur, Arne O. Mooers, David Mouillot, Caroline M. Tucker, and William D. Pearse. Prioritizing phylogenetic diversity captures functional diversity unreliably. *Nature Communications*, 9(1):2888, 2018. (Cited on pp. 19, 121, 201)

[MPKvH09] Bui Minh, Fabio Pardi, Steffen Klaere, and Arndt von Haeseler. Budgeted Phylogenetic Diversity on Circular Split Systems. *IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM*, 6:22–9, 04 2009. (Cited on p. 162)

[MSS07] Vincent Moulton, Charles Semple, and Mike Steel. Optimizing phylogenetic diversity under constraints. *Journal of Theoretical Biology*, 246(1):186–194, 2007. (Cited on pp. 20, 22, 121, 122, 200)

[MVB18] Matthias Mnich and René Van Bevern. Parameterized complexity of machine scheduling: 15 open problems. *Computers & Operations Research*, 100:254–261, 2018. (Cited on p. 86)

[NBP19] Rama Krishna Nimmakayala, Surinder K. Batra, and Moorthy P. Ponnusamy. Unraveling the journey of cancer stem cells from origin to metastasis. *Biochimica et Biophysica Acta (BBA)-Reviews on Cancer*, 1871(1):50–63, 2019. (Cited on p. 23)

[Nic23] Taylor Nicioli. New analysis identifies largest threat to thousands of species facing extinction, 2023. CNN: `https://edition.cnn.com/2023/11/08/europe/analysis-europe-plants-animals-face-extinction-scn/index.html`, visited on June 3rd, 2024. (Cited on p. 17)

[Nie06] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and Its Applications. OUP Oxford, 2006. (Cited on p. 32)

[NSS95] Moni Naor, Leonard J. Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 182–191, 1995. (Cited on pp. 40, 144, 175)

[NW92] Kevin C. Nixon and Quentin D. Wheeler. Measures of phylogenetic diversity. *Extinction and Phylogeny*, pages 216–234, 1992. (Cited on p. 18)

[Pap07] Christos H. Papadimitriou. *Computational complexity*. Academic Internet Public, 2007. (Cited on p. 32)

[Par09] Fabio Pardi. *Algorithms on Phylogenetic Trees*. PhD thesis, University of Cambridge Cambridge, 2009. (Cited on pp. 20, 59)

[PF13] Pavol Prokop and Jana Fančovičová. Does colour matter? The influence of animal warning coloration on human emotions and willingness to protect them. *Animal conservation*, 16(4):458–466, 2013. (Cited on p. 18)

[PG05] Fabio Pardi and Nick Goldman. Species Choice for Comparative Genomics: Being Greedy Works. *PLoS Genetics*, 1(6):e71, 2005. (Cited on pp. 19, 85, 121)

[PG07] Fabio Pardi and Nick Goldman. Resource-Aware Taxon Selection for Maximizing Phylogenetic Diversity. *Systematic Biology*, 56(3):431–444, 2007. (Cited on pp. 20, 59, 63, 64, 75, 85, 92, 152, 153)

[Pis95] David Pisinger. A minimal algorithm for the Multiple-Choice Knapsack Problem. *European Journal of Operational Research*, 83(2):394–410, 1995. (Cited on p. 43)

[PLS14] Benjamin Planque, Ulf Lindstrøm, and Sam Subbey. Non-Deterministic modelling of food-web dynamics. *PloS One*, 9(10):e108243, 2014. (Cited on p. 23)

[PZZ+21] Pavol Prokop, Martina Zvaríková, Milan Zvarík, Adam Pazda, and Peter Fedor. The Effect of Animal Bipedal Posture on Perceived Cuteness, Fear, and Willingness to Protect Them. *Frontiers in Ecology and Evolution*, 9:681241, 2021. (Cited on p. 18)

[RM06] David W. Redding and Arne Ø. Mooers. Incorporating Evolutionary Measures into Conservation Prioritization. *Conservation Biology*, 20(6):1670–1678, 2006. (Cited on pp. 19, 161)

[Ros41] Barkley Rosser. Explicit bounds for some functions of prime numbers. *American Journal of Mathematics*, 63(1):211–232, 1941. (Cited on p. 51)

[RWN+17] William J. Ripple, Christopher Wolf, Thomas M. Newsome, Mauro Galetti, Mohammed Alamgir, Eileen Crist, Mahmoud I. Mahmoud, William F. Laurance, and 364 scientist signatories from 184 countries 15. World Scientists' Warning to Humanity: A Second Notice. *BioScience*, 67(12):1026–1028, 11 2017. (Cited on p. 17)

[Sah76] Sartaj K. Sahni. Algorithms for Scheduling Independent Tasks. *Journal of the ACM*, 23(1):116–127, 1976. (Cited on p. 86)

[SGKS22] Jannik Schestag, Niels Grüttemeier, Christian Komusiewicz, and Frank Sommer. On Critical Node Problems with Vulnerable Vertices. In *Proceedings of the 33rd International Workshop on Combinatorial Algorithms (IWOCA 2022)*, pages 494–508. Springer, 2022. (Cited on p. X)

[SGKS24] Jannik Schestag, Niels Gruettemeier, Christian Komusiewicz, and Frank Sommer. On Critical Node Problems with Vulnerable Vertices. *Journal of Graph Algorithms and Applications*, 28(1):1–26, 2024. (Cited on p. X)

[Sho95] Peter W. Shor. A New Proof of Cayley's Formula for Counting Labeled Trees. *Journal of Combinatorial Theory, Series A*, 71(1):154–158, 1995. (Cited on p. 133)

[Sid73] Jeffrey B. Sidney. An extension of Moore's due date algotithm. In *Proceedings of a Symposium on the Theory of Scheduling and its Applications*, pages 393–398. Springer, 1973. (Cited on p. 86)

[SM12] Luis Santamaría and Pablo F. Mendez. Evolution in biodiversity policy – current gaps and future needs. *Evolutionary Applications*, 5(2):202–218, 2012. (Cited on p. 19)

[SNM08] Andreas Spillner, Binh T. Nguyen, and Vincent Moulton. Computing Phylogenetic Diversity for Split Systems. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 5(2):235–244, 2008. (Cited on pp. 122, 162)

[SPSS23] Gregory W. Stull, Kasey K. Pham, Pamela S. Soltis, and Douglas E. Soltis. Deep reticulation: the long legacy of hybridization in vascular plant evolution. *The Plant Journal*, 114(4):743–766, 2023. (Cited on p. 23)

[Ste59] G. Ledyard Stebbins. The role of hybridization in evolution. *Proceedings of the American Philosophical Society*, 103(2):231–251, 1959. (Cited on p. 23)

[Ste05] Mike Steel. Phylogenetic Diversity and the Greedy Algorithm. *Systematic Biology*, 54(4):527–529, 2005. (Cited on pp. 19, 85, 121, 168, 176, 178)

[SWF+20] Manuel Sorge, Mathias Weller, Florent Foucaud, Ondřej Suchý, Pascal Ochem, Martin Vatshelle, and Gerhard J. Woeginger. The Graph Parameter Hierarchy. *URL: https://manyu.pro/assets/parameter-hierarchy.pdf*, 2020. (Cited on p. 144)

[vIJS+24] Leo van Iersel, Mark Jones, Jannik Schestag, Celine Scornavacca, and Mathias Weller. Maximizing Network Phylogenetic Diversity. *arXiv preprint arXiv:2405.01091*, 2024. (Cited on pp. VIII, IX, 162, 163)

[vIKK+09] Leo van Iersel, Judith Keijsper, Steven Kelk, Leen Stougie, Ferry Hagen, and Teun Boekhout. Constructing level-2 phylogenetic networks from triplets. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 6(4):667–681, 2009. (Cited on p. 185)

[VMM+14] Logan Volkmann, Iain Martyn, Vincent Moulton, Andreas Spillner, and Arne O. Mooers. Prioritizing Populations for Conservation Using Phylogenetic Networks. *PloS one*, 9(2):e88945, 2014. (Cited on p. 24)

[VWHW91] Richard I. Vane-Wright, Christopher J. Humphries, and Paul H. Williams. What to Protect?—Systematics and the Agony of Choice. *Biological Conservation*, 55(3):235–254, 1991. (Cited on p. 18)

[WDS13] Marten Winter, Vincent Devictor, and Oliver Schweiger. Phylogenetic diversity and nature conservation: where are we? *Trends in Ecology & Evolution*, 28(4):199–204, 2013. (Cited on pp. 19, 201)

[Wei66] H. Martin Weingartner. Capital budgeting of interrelated projects: survey and synthesis. *Management Science*, 12(7):485–516, 1966. (Cited on pp. 117, 173, 175)

[Wei92] Martin L. Weitzman. On Diversity. *The quarterly Journal of Economics*, 107(2):363–405, 1992. (Cited on p. 18)

[Wei98] Martin L. Weitzman. The Noah's ark problem. *Econometrica*, pages 1279–1298, 1998. (Cited on pp. 20, 59)

[Wes00] Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, 2 edition, 2000. (Cited on p. 28)

[WF18] Kristina Wicke and Mareike Fischer. Phylogenetic diversity and biodiversity indices on phylogenetic networks. *Mathematical Biosciences*, 298:80–90, 2018. (Cited on pp. 24, 161, 165, 198)

[Wil84] Edward O. Wilson. *Biophilia: the human bond with other species*. 1984. (Cited on p. 18)

[WMS21] Kristina Wicke, Arne Mooers, and Mike Steel. Formal Links between Feature Diversity and Phylogenetic Diversity. *Systematic Biology*, 70(3):480–490, 2021. (Cited on p. 18)

[Yap83] Chee K. Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical computer science*, 26(3):287–300, 1983. (Cited on p. 38)