The Computer Journal Advance Access published July 30, 2007

doi:10.1093/comjnl/bxm049

Fixed-Parameter Algorithms in Phylogenetics

Jens Gramm[1], Arfst Nickelsen[2] and Till Tantau[2,*]

[1]*Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Germany*
[2]*Institut für Theoretische Informatik, Universität zu Lübeck, Germany*
**Corresponding author: tantau@tcs.uni-luebeck.de*

We survey the use of fixed-parameter algorithms in the field of phylogenetics, which is the study of evolutionary relationships. The central problem in phylogenetics is the reconstruction of the evolutionary history of biological species, but its methods also apply to linguistics, philology or architecture. A basic computational problem is the reconstruction of a likely phylogeny (genealogical tree) for a set of species based on observed differences in the phenotype like color or form of limbs, based on differences in the genotype like mutated nucleotide positions in the DNA sequence, or based on given partial phylogenies. Ideally, one would like to construct socalled perfect phylogenies, which arise from a very simple evolutionary model, but in practice one must often be content with phylogenies whose 'distance from perfection' is as small as possible. The computation of phylogenies has applications in seemingly unrelated areas such as genomic sequencing and finding and understanding genes. The numerous computational problems arising in phylogenetics often are NP-complete, but for many natural parametrizations they can be solved using fixed-parameter algorithms.

Keywords: bioinformatics; phylogenetics algorithms; parameterized algorithms

Received 15 March 2006; revised 28 July 2006

1. INTRODUCTION

1.1. Phylogenetics

The word *phylogeny* comes from Greek *phylon*, meaning race, and *geneia*, meaning origin. In phylogenetics one studies how different species are related evolutionary. The basic paradigm is that species spawn new species, for example when part of a species' population adapts to a changing environment. Over time the set of extant species changes as new species emerge and other species become extinct. The ancestral relationship between the species can be depicted by arranging them in a tree, called a *phylogenetic tree* or *phylogeny*, where the leaves are labeled with extant species and where bifurcations correspond to events like adaptations that lead to new species. Interior nodes are labeled with ancestral species or not at all when the ancestral species are unknown or not of interest. A classical example of a phylogeny is the tree of life, a small part of which is shown in Fig. 1, though in the tree of life we see *taxa* instead of species (a taxon is an arbitrary grouping of organisms while the term *species* applies only to the basic building blocks of biodiversity).

Evolutionary processes are not restricted to biology as was already noted by Charles Darwin himself in Chapter 14 of *On the Origin of Species* [1]. In linguistics, languages—instead of species—evolve over time, resulting in a tree of languages. Nodes of the tree are labeled by languages and bifurcations correspond to changes of words or grammar that resulted in new dialects or languages. A small part of the tree of languages is shown in Fig. 1.

Building phylogenies is not an easy task. The problems start with determining the set of taxa since neither for biological species nor for languages it is always clear where we should draw the line. But suppose we have agreed on a set of taxa and the task is to arrange them in a phylogeny, then we only know which taxa there are *now*, but we do not know which taxa there *were* in the past. In rare, fortunate cases we might have access to fossils or to text in dead languages, but normally the path evolution took will be unknown to us.

Even when we know all taxa in the phylogeny, including even possibly extinct early taxa, it is still often subject to debate how they should be arranged. To solve this problem, one idea is to infer the phylogeny by looking at different traits, better known as *characters* in the biological literature, of the taxa. Characters are attributes like the form of the

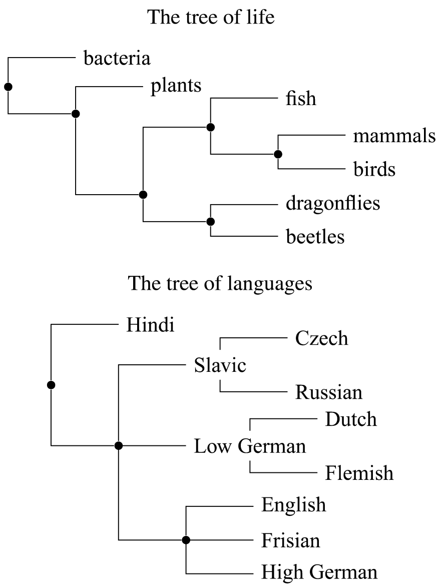


FIGURE 1. Excerpts of two important phylogenies. In the left phylogeny, taxa label only leaf nodes. The shown portion of the tree of languages is taken from [2].

skeleton or, for languages, the way a certain word is spoken. Taxa for which the form of the skeleton is similar or languages for which a word is spoken in a similar way should be in the same subtree of the phylogeny. The joint information from many characters will often leave us with a single phylogeny or at least few possible ones. In biology, principal sources of characters are the phenotype of a taxon, which is roughly 'the way the organisms of the taxon look,' but also genomic information like which genes are present in the organisms of a taxon. In linguistics, characters include pronunciation, word order and more generally grammatical structure.

The construction and study of phylogenetic trees has many applications. First of all, a phylogeny allows us a glimpse of how evolution works and can help us in classifying organisms. Second, we can compare multiple phylogenies built for the same set. For example, one can build two phylogenies of languages, the first based on linguistic traits and the second based on the genetic traits of the speakers of the language. The resulting phylogenies will match in large parts, but the places where they do not match may be especially interesting, indicating for instance that a population may have switched to another language for some reason. A third, rather intriguing application of phylogenies is their use as measures in other applications. One such application is the *haplotype phase determination problem*, presented in detail in Section 7.2. The output for this problem is a set of taxa, but a large number of different sets of taxa are possible outputs *a priori*. The tricky part is not so much *filtering out* the biologically most likely solution, but *measuring* how likely a solution is. An elegant way of doing this is to declare those sets of taxa as 'good solutions' that can be arranged in a phylogeny.

1.2. Computational problems in phylogenetics

The most fundamental problems in phylogenetics are not computational in nature. Deciding what exactly counts as a taxon, choosing a set of characters or deciding which states a taxon is in are not computational problems, but require human experience and judgment. However, once these decisions have been made, numerous computational problems arise that cannot be solved 'by hand' when large amounts of input data need to be processed, as is the case in phylogeny-based haplotyping, for instance. In the following we give an overview of the computational problems that we address in this survey; good starting points for further coverage of computational issues in phylogenetics are [3–6].

A fundamental computational problem in phylogenetics is the construction of a phylogeny for a given set of taxa (detailed mathematical definitions are given later). We are also given a set of characters and we know for each character and each taxon the *state* of the taxon with respect to the character. For example, when the taxa are elephants and mice and the characters are size and color, elephants are in the state 'big' with respect to the character size and 'gray' with respect to the character color, mice are in the states 'small' and 'gray.' State information for taxa and characters is typically arranged in matrices such as the ones shown in Tables 1 and 2.

One has to choose a model of evolution that says which phylogenies are considered good explanations of the observed character–state matrix. A basic model is the following: All taxa sharing a state for some character are descendants of the same taxon. For example, if some mutation causes elephants to be the first 'big' animal, this model insists that all other 'big' animals must be descendants of elephants. One possible way of checking whether this model applies to a phylogeny is to check, whether for each character and each pair of states the path between any two taxa in the first state and the

TABLE 1. A character–state matrix for programming languages.

Language	Scope start	Module start	For-statement	Variable reassignment
Pascal	`begin`	`unit`	Yes	Allowed
Modula	`begin`	`module`	Yes	Allowed
Haskell	`(`	`module`	No	Not allowed
C	{	–	Yes	Allowed
Java	{	`package`	Yes	Allowed
TEX	{	–	No	Allowed

The 'taxa' are the six programming languages on the left. The 'characters' are the symbol used to start a scope, the keyword used to start a module, the question of whether built-in for-loops exist and the question of whether it is permissible to reassign another value to a variable. This matrix cannot be arranged in a perfect phylogeny, but if we remove the 'taxon' TEX, which did not inherit its syntax from another programming language, the remaining rows can.

TABLE 2. Character–state matrix that is an instance of pp.

Species	Hyaline margin	Marginal carina	Premarginal carina
Chelopistes guttatus	0	0	0
Oscvlotes macropoda	0	1	0
Oxylipeurus dentatus	1	1	1
Upupicola upupae	0	0	2
Perineus nigrolimbatus	2	0	2

It is a submatrix of a much larger character-state matrix, compiled by Smith [7], that contains entries for 56 species and 138 characters. In the above matrix, the five rows are five different lice species from the suborder Ischnocera (Phthiraptera). These lice are permanent parasites of many birds and mammals throughout the world. The columns are three characters referring to the form of the head of adult lice. The numbers in the matrix encode the different states of the characters. For example, for the character *marginal carina* the 0-entries mean that the adult marginal carina 'forms a complete thickened band running anteriorly around the preantennal region of the head' and the 1-entries mean that it 'forms a band which is interrupted laterally (partially or completely), medially (dorsally and/or ventrally) or both' [7].

path between any two taxa in the second state do not intersect. Such a phylogeny is called *perfect*. Naturally, the model is rather crude since, after all, not all large animals are related, and the model is regarded as overly simplifying by practitioners; but see the discussion after Definition 2.2 for a justification why we focus on perfect phylogenies nevertheless.

A second set of computational problems arises when it is not possible to arrange taxa in a perfect phylogeny. We then have several options: First, we can lower our standards of what counts as a good phylogeny by allowing a small number of 'backward mutations' in the tree. Second, we can still try to find a perfect phylogeny, but only for a subset of the taxa or for a subset of the characters. Third, we can claim that the data must be in error and try to find a way—as little disruptive as possible—to modify the data such that a perfect phylogeny can be obtained. While this is not advisable in general (we cannot simply claim that elephants can fly, just to fit them into a perfect phylogeny), genomic data is often obtained through laboratory processes in which one cannot avoid a percentage of wrong entries.

Phylogenies need not always be constructed 'from scratch' based on character state data. Rather, we often have access to *partial* phylogenies that are subtrees of the phylogeny sought for. An example is the tree of life: We do not wish to construct this tree based on an enormous character database for millions of taxa. Rather, the problem is to merge many different small phylogenies from the literature into one big phylogeny, appropriately called a *supertree*. For a related problem we also do not construct phylogenies from scratch, but we are given several complete candidate phylogenies obtained through external means and our job is to compute biologically meaningful *distances* between them—a difficult problem all by itself.

New, even more difficult computational problems arise when our data is incomplete, which is often the case. For certain characters and taxa we simply might not know the state: we may not know the translation of a word into a certain language or the genomic sequencing process may have failed to determine the base at a certain nucleotide position. In such cases we do not only need to find a good phylogeny, but we must also fill in the missing entries in a sensible manner. Needless to say that this often introduces a whole new level of complexity.

1.3. Parametrization and phylogenetics

Most computational problems in phylogenetics turn out the be NP-complete, forcing us to look for heuristics, approximation algorithms or fixed-parameter algorithms. The fixed-parameter approach turns out to be especially successful.

The reason for this success is that a number of problematic parameters are, indeed, small in realistic instances for phylogenetic problems—like the number of states per character, the number of characters or our tolerance for errors. The number of states per character, for instance, is at most four (and in many cases even two), whenever genomic data is involved (nature kindly uses only four nucleobases) and the running time of many algorithms is exponential in the number of states per character, but polynomial otherwise. Next, the number of characters in input matrices is large in general (Smith lists 138 different characters for a set of lice species [7]), but it is sometimes possible and necessary to partition the character set into small subsets and apply algorithms only to these small subsets.

In the course of the present paper we will see other examples of parameters that are small in practice, allowing us to construct efficient, exact algorithms for many computational problems arising in phylogenetics.

1.4. Goals and overview

The central goal of this survey is to highlight selected fixed-parameter algorithms from the area of phylogenetics. The chosen examples are intended to illustrate the diversity of computational problems for which fixed-parameter algorithms have been developed within the area of phylogenetics.

We address this survey to readers interested in computational issues and algorithmics and assume no detailed knowledge about biology. We try to give illustrative examples and explanations, as space permits, accessible to non-experts. We assume that you are familiar with fixed-parameter algorithms, but most theorems can be understood without knowledge about parametrized complexity classes—they only come into play in the concluding summary. For an introduction to fixed-parameter theory, see for example the two monographs [8] and [9]. Except for the short proofs of a few easy new results that we include for completeness, no detailed proofs are included in this survey, but we tried to at least sketch some proof ideas of theorems from the literature.

In Section 2, we introduce (one possible version of) the formal problem of constructing a perfect phylogeny and study how the parameters *number of taxa*, *number of characters* and *number of states per character* influence the tractability of the problem. In Section 3, we study ways of measuring the deviation of a given phylogeny from 'perfection.' Section 4 treats problems where the task is to find a phylogeny that is near to perfection with respect to the introduced measures. In Section 5, we look at problems where the task is to compute distances between phylogenies. In Section 6, the problem of merging several partial phylogenies is treated. In Section 7, we consider applications in which the aim is not to construct a phylogeny but where its construction is just a means to an end, namely finding regulatory genomic elements, or determining haplotype phases. In the conclusion, we summarize the complexity-theoretic results in a table and give an outlook.

2. CONSTRUCTION OF PERFECT PHYLOGENIES

In this section we study how difficult it is to construct a perfect phylogeny. First, we define the problem PP (perfect phylogeny) formally, and discuss possible variations. Then, we look at what happens when we fix one of the three central parameters number of taxa, number of characters and number of states per character.

2.1. Formalization of the perfect phylogeny problem

Fix a set C of *characters* like size or color, and for each character $c \in C$ fix is a set Σ_c of states for this character, like $\Sigma_{\text{size}} = \{\text{small, medium, big}\}$. Then the input for the perfect phylogeny problem is a set S of taxa together with one mapping for each taxon $s \in S$, each of which assigns an element of Σ_c to each character $c \in C$.

There are three natural parameters in such inputs:

(i) The number n of taxa.
(ii) The number m of characters.
(iii) The maximum number r of states a character can have.

For computational issues, the names of the characters, the states for each character and even the taxa are not really important. Therefore, we can make the notation simpler by assuming that the set S of taxa is $\{1, \ldots, n\}$, the character set C is $\{1, \ldots, m\}$ and each state set is $\Sigma_i = \{0, \ldots, r-1\}$. It is customary to start the states with 0 so that if there are just two states, then they are 0 and 1. The states of a taxon can now be described by a vector from the set $\Sigma_1 \times \cdots \times \Sigma_m = \{0, \ldots, r-1\}^m$. Thus, the n input taxa are described by length-m vectors of numbers from $\{0, \ldots, r-1\}$. Another way to think about the input is in terms of an $(n \times m)$-matrix with entries from $\{0, \ldots, r-1\}$. Be cautioned that in the biological literature these matrices are sometimes presented in transposed form.

Before we define *perfect* phylogenies, let us first define *phylogenies*.

Definition 2.1 (Phylogeny). *Let A be a matrix describing n taxa.* A phylogeny for the matrix *A is a tree T whose node set V is labeled using a labeling function $l: V \to \{0, \ldots, r-1\}^m$* such that:

(i) *Every row of A, that is, each taxon's state vector, is a label of some node in T.*
(ii) *The labels of the leaves of T are rows of A. (By comparison, the labels of inner nodes correspond to ancestral taxa, which need not, but may, be part of the input.)*

Be cautioned that phylogenetic *trees* may not always be the best way of describing evolutionary relationships, for instance because they do not account for so-called *horizontal gene transfers* in which a gene is transfered between unrelated taxa by a 'mixing' of the genetic material. In a study of conserved loci in bacterial pathogens, Feil *et al.* [10] conclude that for lineages within a species 'over the long term, the impact of relatively frequent recombination is to obliterate the phylogenetic signal in gene trees such that the relationships between major lineages of many bacterial species should be depicted as a network rather than a tree.' Indeed, comparisons of the different genomes that were sequenced during the last decade show that such transfers are quite frequent, threatening the very idea of trying to explain evolutionary history using trees, see the reviews [11] or [12] for an overview and as literature starting points. In the present survey we restrict attention to phylogenetic *trees*, nevertheless, since we should try to understand these first, before tackling the more difficult phylogenetic *networks*. We come back to phylogenetic networks in the outlook at the end of this paper.

Recall that in the evolutionary model behind perfect phylogeny all taxa sharing a state for some character have the same ancestor. This can be formalized as follows:

Definition 2.2 (Perfect phylogeny). *A phylogeny is* perfect *if for every character $c \in C$ and every state $j \in \Sigma_c = \{0, \ldots, r-1\}$, the graph induced by the set of nodes labeled by a state vector $(s_1, \ldots, s_m)$ with $s_c = j$ is connected.*

Definition 2.3 (perfect phylogeny). *The input for the problem* perfect phylogeny (*abbreviated* PP) *is a character–state matrix A for n taxa. The task is to decide whether there exists a perfect phylogeny for A.*

Some remarks on the definition are in order both with respect to its biological relevance and to the chosen mathematical formalization.

Concerning the biological relevance, one can object that real biological character–state matrices rarely admit a perfect phylogeny. For example, Table 2 displays a real-life instance of PP and a perfect phylogeny for this matrix is shown in Fig. 2. However, this table is just a small part of a much larger matrix compiled by Smith [7] and the whole matrix does not admit a perfect phylogeny. Nevertheless, there are several reasons why we should still study perfect phylogenies.

- PP is in some sense the most basic computational problem in phylogeny construction and we would like to understand this problem well before we attack more complicated settings.
- Even if data cannot be arranged in a perfect phylogeny, we may still try to find a phylogeny that is 'as perfect as possible,' see Section 4.
- There are biological settings where the perfect phylogeny model works quite well. Consider *single nucleotide polymorphism sites* (SNP sites) for instance, which are specific base positions in the genome where we observe a variation across the population. The base (state) at such a position might have been adenine originally and a mutation caused it to change to cytosine in part of the population. Such mutations are rare—often SNP sites are hundreds of bases apart. If SNP mutations occur randomly, it is extremely unlikely that the same site will mutate more than once; and having at most one mutation per site exactly defines the evolutionary model of perfect phylogenies. Note, however, that this argument breaks down when the chromosomal crossing over effect has to be taken into account and the perfect phylogeny model for SNPs works only when a small number of SNP sites are considered.

Concerning the mathematical formalization of perfect phylogenies, we chose a broad definition for this survey. We allow an arbitrary tree topology, the input taxa can be found both at the leaves and at inner nodes, and the same label may be found at different nodes. We only insist that there are no superfluous leaves, even though this condition is not strictly necessary either. Other definitions in the literature impose more structure on perfect phylogenies.

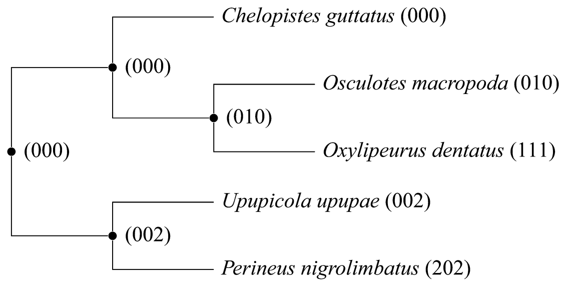
- It is often required that the set of leaf labels equals the set of rows of the input matrix, that is, it is not allowed to place a taxon at inner nodes. A perfect phylogeny in the sense of Definition 2.2 can be turned into a phylogeny with the input taxa at the leaves by adding a pending leaf to all inner nodes that harbor an input taxon.
- It is often convenient to have more control over the tree topology. It is particularly convenient to consider binary trees, which are trees in which every node either has degree one (leaves) or degree three (inner vertices). This can also be achieved easily: Replace all nodes of too high degree by small binary trees with all nodes labeled by the original node's label; and remove all nodes of degree two while joining their adjacent nodes.
- It is sometimes useful to impose the quite natural condition that all labels are distinct. This can be accomplished by contracting subtrees whose nodes all have the same label (and, indeed, sets of nodes that are labeled identically must form a connected subtree in a perfect phylogeny). However, this contraction process may destroy the binary tree property and also the property that input taxa must label leaves.

In a perfect phylogeny there is no designated root node and, in general, it may be debatable which taxon should be considered the 'root.' If, for whatever reason, a root node has been chosen, the phylogeny is called *directed*.

Having defined (the decision version of) the perfect phylogeny problem, the natural question is, how difficult is this problem? Unfortunately, it is NP-complete.

THEOREM 2.1 ([13, 14]). PP *is* NP-*complete.*

This result suggests that in order to tackle the problem we must look at restricted versions. We do so by fixing the different central parameters number n of taxa, number m of characters and the maximum number r of states per character.



Chelopistes guttatus (000)
(000)
Osculotes macropoda (010)
(010)
(000)
Oxylipeurus dentatus (111)
Upupicola upupae (002)
(002)
Perineus nigrolimbatus (202)

FIGURE 2. One possible perfect phylogeny for the character–state matrix from Table 2. The labels assigned to the vertices of the phylogeny are shown in partentheses.

2.2. Number of taxa as the parameter

The first restriction is to limit the number n of taxa in the input. Intuitively, if there are just, say, four taxa, it should not be particularly hard to find out whether we can arrange them in a perfect phylogeny—after all, there are only a fixed number of tree topologies for them.

THEOREM 2.2. PP *can be solved in time* $O(2^n n! \cdot m)$.

Proof. For a fixed binary tree topology T and a one-to-one assignment of elements in S and leaves of T, it can be tested in time $O(nm)$ whether the the inner nodes of T can be labeled in a way such that T is a perfect phylogeny for S. The number of possible binary trees with n distinctly labeled leaves is known to be $1 \cdot 3 \cdot 5 \cdot \cdots \cdot (2n - 5) \leq 2^n(n - 2)!$.

Therefore, enumerating all binary trees for S and testing each for being a perfect phylogeny for S yields the stated running time. □

Theorem 2.2 shows that our intuition was correct and PP is (more or less trivially) fixed-parameter tractable with respect to n. The sketched algorithm is very simple and cannot handle a number n of taxa that is greater than perhaps 10—while in practical situations we typically have over a hundred taxa. More clever exhaustive search algorithms in phylogenetics push the maximum number of taxa that can be handled from about 12 on desktop machines to about 15 on workstations; but what we really would like to find is a fixed-parameter algorithm for the parameter n-based, ideally, on a kernelization algorithm followed by a search tree algorithm, yielding a running time as the one stated in the below open problem.

OPEN PROBLEM 2.1. *Is there a fixed-parameter algorithm for PP with respect to the parameter n with a running time in* $O(c^n + (mr)^{O(1)})$ *for some c close to 1?*

2.3. Number of characters as the parameter

Returning to an arbitrary number of taxa, we now have a look at what happens when we fix the number m of characters. This is justified in an important practical application. As argued by Gusfield in [15], the perfect phylogeny model explains genomic variations well when crossing over effects are not present. This implies that for *short* genomic sequences, the perfect phylogeny model applies, and for longer sequences, we can try to partition the sequence into short intervals and derive perfect phylogenies for these small sets of characters.

Once more, the intuition is that it should be easy to find a perfect phylogeny if there are only, say, three characters and, indeed, Morris, Warnow and Wimer present an algorithm with the following running time:

THEOREM 2.3 ([16]). PP *can be solved in time*

$$O(r^{m+1}m^{m+1} + nm^2).$$

The idea is to show that PP is polynomially equivalent to the problem of triangulating colored graphs. For this latter problem we are given a graph $G = (V, E)$ and a coloring $c : V \to C$, and the task is to find a supergraph G' of G that is properly colored by c and that is triangulated (every cycle of length at least four contains a chord). The number of colors for the triangulated colored graphs problem corresponds to the number of characters in PP.

Using a different approach, Agarwala and Fernández-Baca arrive at the following running time:

THEOREM 2.4 ([17]). PP *can be solved in time*

$$O((r - n/m)^m \cdot rnm).$$

For fixed m, both of the above time bounds are polynomial in n and r. However, neither algorithm shows that the problem is fixed-parameter tractable as we still have m in the exponent and another input parameter in the base. The following theorem of Bodlaender *et al.* shows that it is unlikely that this can be remedied:

THEOREM 2.5 ([18]). *For every t,* PP *with parameter m (number of characters) is* W[t]*-hard.*

Interestingly, this hardness result is also proved by exploiting the equivalence of PP to the problem of triangulating colored graphs.

2.4. Number of states per character as the parameter

The third natural parameter for the perfect phylogeny problem is the number of states per character. Fixed-parameter results for this number are especially important since it is, indeed, small in many applications. Consider, for instance, an input in which characters are nucleotide positions in the genome and the possible states for each character are the four bases adenine, cytosine, guanine and thymine. Then the number of states is four or, if we also take alignment-induced gaps into account by adding a 'no-data' or 'gap' state, five. Even better, in applications such as the phylogeny-based haplotyping presented in Section 7.2, we may assume that at most one mutation per site occurs and, thus, there are only two different states for each character.

The first fixed-parameter algorithm for the parameter r was proposed by Agarwala and Fernández-Baca. It has the following running time:

THEOREM 2.6 ([19]). PP *can be solved in time*

$$O(2^{3r} \cdot (m^3n + m^4)).$$

This result was later improved by Kannan and Warnow.

THEOREM 2.7 ([20]). PP *can be solved in time*

$$O(2^{2r} \cdot m^2n).$$

An $O(m^2n)$ algorithm for the special case $r = 3$ had already been achieved by Dress and Steel [21]. Kannan and Warnow [22] give an $O(mn^2)$ algorithm for $r = 4$.

For the special case $r = 2$ one can make use of a simple, but powerful characterization of matrices that admit a perfect phylogeny. The characterization is in terms of a forbidden induced submatrix and has been rediscovered independently by several authors, among others in [23] and [24].

THEOREM 2.8. *For $r = 2$,* a matrix *A of taxa has a perfect phylogeny if and only if it does not contain the following induced submatrix:*

$$\begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}.$$

Employing this characterization, which will also play a role in Section 4.2, Gusfield devised an algorithm running in linear time.

THEOREM 2.9 ([25]). *For $r = 2$,* PP *can be solved in time $O(mn)$.*

The results of this section can be summed-up as follows: PP with respect to either of the parameters n and r (number of taxa and number of states per character) is in FPT, but with respect to the parameter m (number of characters) it is in XP and hard for W[t] for all t (Table 3 on page 20).

3. MEASURES OF DEVIATION FROM PERFECTION OF PHYLOGENIES

In the previous section we studied perfect phylogenies. In practice, we often have to deal with imperfect phylogenies since, in reality, input matrices only rarely admit a perfect phylogeny. In this case, we may look for a phylogeny that is at least 'near' to being perfect instead. For this, we need to measure how strongly a phylogeny deviates from being a perfect phylogeny.

3.1. Measures based on relaxed evolutionary models

The basic assumption of the evolutionary model underlying perfect phylogenies is that mutations of a character to some state happen only once. We start with two measures that count, in different ways, how often this assumption is violated. Note that the input for these problems is a phylogeny, and not a matrix. In the closely related Section 4.1 we will treat, for each measure, the question of finding some phylogeny for an input matrix that minimizes the distance to perfection.

The first measure is the *penalty* of a phylogeny, due to Fernández-Baca and Lagergren [26].

DEFINITION 3.1 (Length and penalty of a phylogeny). *For an edge of a phylogeny connecting nodes u and v, we define the length of the edge as the Hamming distance of u and v (the number of characters where the states differ). The* length *of a phylogenetic tree T is the sum of lengths taken over all edges of the tree. The* penalty *of a phylogenetic tree T is defined as*

$$\text{penalty}(T) = \text{length}(T) - \sum_{c \in C} (r_c - 1),$$

where r_c is the number of states of character c that are present in the phylogeny.

The idea behind this measure is the following: The length of an edge e connecting taxa u and v is the number of mutations that occurred between u and v. For a perfect phylogeny, a new state is introduced by a mutation only once and therefore every character c contributes exactly $r_c - 1$ to the length of the tree. Hence, the penalty of a tree counts how often the assumption 'each new state is introduced only once by a mutation' is violated. Perfect phylogenies have a penalty 0.

The second measure is the *phylogenetic number*, due to Goldberg *et al.* [27]. For a state j and a character c let $T_{c,j}$ denote the subgraph of the phylogenetic tree T induced by the set of nodes whose labels are in state j for the character c. Then the phylogenetic number is defined as follows:

DEFINITION 3.2 (Phylogenetic number). *The* phylogenetic number *of a phylogeny T is the maximum number of times that any given state arises in T, that is, the maximum number, taken over all characters c and all states j, of connected components in $T_{c,j}$. Phylogenies with phylogenetic number ℓ are called ℓ-phylogenies.*

A 1-phylogeny is the same as a perfect phylogeny. Unlike the penalty, which bounds the total number of violations of the basic evolutionary model, the parameter ℓ does not restrict the total number of violations, but violations may not 'concentrate' at a single state.

A third measure of a similar flavor is the *number of bad states*. It is due to Moran and Snir [28] who study how to get rid of bad states by a minimal number of recolorings (compare Definition 3.6 in the next subsection).

DEFINITION 3.3 (Number of bad states). *Given a phylogeny T and a character c, the character's number of bad states is number of states j for which $T_{c,j}$ is not connected. The* number of bad states *of a phylogeny T is the maximum numbers of bad states taken over all characters.*

Clearly, for a given phylogeny all of the above measures can be computed in polynomial time.

One can easily define further measures of a similar spirit; for instance, we could maximize over the number of bad states per character or sum over the number of times a given state arises in T, and so on, leading to an abundance of possible measures and only few hints as to which measure might be most appropriate in a particular biological setting.

OPEN PROBLEM 3.1. *Do a comparative study of the introduced and similar measures with respect to their biological relevance.*

3.2. Measures based on the modification of input phylogenies

We now introduce measures that are based on the idea that if the input data does not admit a perfect phylogeny, the data must be flawed. One then tries to modify or even remove the taxa of a given phylogeny until a perfect phylogeny is reached. Note, again, that the input is a phylogeny, and not a matrix. The taxa are already arranged in a tree and we only wish to know how the particular input phylogeny needs to be modified to arrive at a perfect phylogeny. In Section 4.2 we study the related, but different, problem of modifying a character-state input matrix so that the resulting matrix admits a perfect phylogeny.

For the first measure of this type one tries to prune a phylogeny until it becomes perfect.

DEFINITION 3.4 (TREE PERFECTION BY TAXA REMOVAL). *The input for* TREE PERFECTION BY TAXA REMOVAL *is a phylogeny T and a number k. The task is to decide whether we can turn the phylogeny T into a perfect phylogeny by repeatedly cutting away leaves such that at most k of the original leaves are removed.*

It is not straightforward how to minimize the number of taxa removals since there are many ways to prune a phylogeny and, indeed, this problem is NP-complete already for $r = 2$ as the following theorem shows.

THEOREM 3.1. *For every $r \geq 2$,* TREE PERFECTION BY TAXA REMOVAL *is* NP-*complete*.

Proof. The problem clearly is in NP. Hardness is shown by reducing VERTEX COVER to it. For a graph $G = (V, E)$ with $|V| = n$ and $|E| = m$, construct a star-shaped phylogeny T with one center node and n leaves, one for each vertex $v \in V$. The taxa have m characters c_e, one for each edge $e \in E$. Each character has two states 0 and 1. The center node is labeled 0^m. The leaf in T corresponding to vertex v in G is labeled with the character–state vector that has state 1 for character c_e if and only if v is an endpoint of e. Now, for each edge there are two taxa (leaves) in the phylogeny for which the state of c_e is 1. At least one of these taxa has to be removed to make the phylogeny perfect, because of the 0^m vector in the 'center.' Therefore, the vertex covers of G correspond exactly to sets of leaves whose removal lets T become perfect. □

Slight modifications of this proof show that the problem remains NP-complete for binary phylogenies, or if one defines phylogenies in a way such that only leaves are labeled with taxa.

OPEN PROBLEM 3.2. IS TREE PERFECTION BY TAXA REMOVAL *fixed-parameter tractable with respect to the parameter k (number of removed taxa)?*

A second measure counts how many characters must be removed (disregarded) so that the phylogeny becomes perfect. This number is much easier to compute.

DEFINITION 3.5 (TREE PERFECTION BY CHARACTER REMOVAL). *The input for* TREE PERFECTION BY CHARACTER REMOVAL *are a phylogeny T and a number k. The task is to decide whether the phylogeny T can be turned into a perfect phylogeny by disregarding k characters.*

THEOREM 3.2. TREE PERFECTION BY CHARACTER REMOVAL *can be solved in polynomial time.*

Proof. A character is either 'in error' (because there is a state such that the set of all taxa of this state for the character is not connected, which can be checked in polynomial time) or the character is 'clean.' We must disregard all erroneous characters and this suffices. □

A third measure, implicitly introduced by Moran and Snir [28], is based on a more fine-grained analysis of the erroneous characters. Instead of just disregarding those characters that violate the connectedness condition, we try to 'fix them' by changing the states at a minimal number of places. Such a change of state may also be regarded as a *recoloring*, since states correspond to colors in equivalent formulations of the perfect phylogeny problem.

DEFINITION 3.6. (Recoloring number). *Given a phylogeny T, the* recoloring number *is the minimal number of state changes (the number of times we need to change a state in some node label) needed to arrive at a perfect phylogeny.*

DEFINITION 3.7 (TREE PERFECTION BY RECOLORING). *The input for* TREE PERFECTION BY RECOLORING *are a phylogeny T and a number k. The task is to decide whether the recoloring number of T is at most k.*

Finding an optimal recoloring for one character is not influenced by recolorings necessary for another character, hence we can compute the recoloring number for each character separately. Hence, the problem reduces to the problem for a single character (called CONVEX RECOLORING OF TREES by Moran and Snir), which Moran and Snir show to be NP-complete. Indeed, Moran and Snir show something even stronger.

THEOREM 3.3 ([28]). TREE PERFECTION BY RECOLORING is NP-*complete, even if we allow only instances where the phylogeny forms a path and where there is only a single character.*

On the other hand, Moran and Snir present an algorithm for computing the recoloring number. Recall that b is the number of bad states (see Definition 3.3) which are the states (or colors) for which some action needs to be taken.

THEOREM 3.4 ([28]). TREE PERFECTION BY RECOLORING *can be solved in time*

$$O\left(\left(\frac{b}{\log b}\right)^{b} \cdot bmn^{4}\right).$$

The above theorem shows that computing the recoloring number is fixed-parameter tractable with respect to the number of bad states.

OPEN PROBLEM 3.3. *With respect to which other parameters is* TREE PERFECTION BY RECOLORING *fixed-parameter tractable?*

4. CONSTRUCTION OF GOOD PHYLOGENIES

In the present section we study algorithms that construct phylogenies that are 'almost' or 'nearly' perfect. To define what counts as a good phylogeny, we use the measures introduced in the previous section. Having fixed a measure, our objective now is to find a phylogeny of minimal measure for a given input matrix. Intuitively, this is a much more difficult problem than the ones studied in the previous section, where we just wanted to compute the measure of a single phylogeny—and often already this seems difficult. However, it is conceptually possible that it is hard to compute a measure for a given phylogeny, but still easy to construct *some* phylogeny of minimal measure for given input matrix.

4.1. Minimizing penalty, phylogenetic number and number of bad states

Let us start with algorithms that find phylogenies minimizing the penalty (number of excess mutations) from Definition 3.1, the phylogenetic number (one plus the maximum of the number of excess mutations per state) from Definition 3.2, or the number of bad states (number of states for which an excess mutation has occurred) from Definition 3.3.

DEFINITION 4.1 (Measure minimization problems). *The input for the problems* PHYLOGENETIC PENALTY MINIMIZATION, PHYLOGENETIC NUMBER MINIMIZATION *and* PHYLOGENETIC BAD STATES MINIMIZATION *is a matrix A of taxa and a number p. The task is to decide whether there exists a phylogeny for A of penalty at most p, with a phylogenetic number of at most p or with at most p bad states.*

Fernández-Baca and Lagergren [26] call phylogenies that minimize the penalty 'near-perfect,' but we use PHYLOGENETIC PENALTY MINIMIZATION for consistency.

All problems are generalizations of PP since when the penalty is 0 (or 1, for the phylogenetic number), the task is simply to decide whether a perfect phylogeny exists. This shows that we cannot hope for a fixed-parameter algorithm for any of these problems with respect to the parameter p alone. If we take the parameter r also into account, two theorems are known about minimizing the penalty.

THEOREM 4.1 ([26]). PHYLOGENETIC PENALTY MINIMIZATION *can be solved in time*

$$O(m^{O(p)}2^{O(p^2r^2)} \cdot n).$$

THEOREM 4.2 ([29]). For $r = 2$, PHYLOGENETIC PENALTY MINIMIZATION *can be solved in time*

$$O(2^{O(p^2)} \cdot nm^2).$$

Theorem 4.2 tells us that for the particularly interesting case of only two states per character there is a fixed-parameter algorithm for finding a good phylogeny with respect to the parameter penalty.

Much less is known about minimizing the phylogenetic number or the number of bad states.

OPEN PROBLEM 4.1. *For which parameters or parameter pairs are* PHYLOGENETIC NUMBER MINIMIZATION *or* PHYLOGENETIC BAD STATES MINIMIZATION *fixed-parameter tractable?*

4.2. Minimizing the modification of input data

The next measures that we considered were based on the idea that one may modify the input to arrive at a perfect phylogeny. Trying to minimize these measures leads to the following problems:

DEFINITION 4.2 (PP BY TAXA REMOVAL). *The input for* PP BY TAXA REMOVAL *is a character–state matrix A of taxa and a number k. The task is to remove at most k taxa (rows) from A such that the resulting matrix admits a perfect phylogeny.*

DEFINITION 4.3 (PP BY CHARACTER REMOVAL). *The input for* PP BY CHARACTER REMOVAL *is a matrix A of taxa and a number k. The task is to remove at most k characters (columns) from A such that the resulting matrix admits a perfect phylogeny.*

Since for the case $r = 2$ we have a characterization of matrices that admit a perfect phylogeny by a forbidden submatrix (see Theorem 2.8) the following problem becomes important for $r = 2$.

DEFINITION 4.4 (ROW DELETION). *For a fixed matrix B, the input for the* ROW DELETION *(B) problem is a matrix A and a number k. The task is to remove at most k rows from A such that the resulting matrix does not contain B as an induced submatrix.*

Combining the results proved in [30] on the row deletion problem and results on the fixed-parameter tractability of the hitting set problem, one gets the following results.

THEOREM 4.3. *For every $r \geq 2$, both* PP BY TAXA REMOVAL *and* PP BY CHARACTER REMOVAL *are* NP-*complete.*

THEOREM 4.4. For $r = 2$, PP BY TAXA REMOVAL *can be solved in time $O(3.30^k + n^4)$ and also in time $O(2.18^k n + n^4)$.*

THEOREM 4.5. *For $r = 2$,* PP BY CHARACTER REMOVAL *can be solved in time $O(1.29^k + m^2)$.*

Proof. For $r = 2$, the PP BY CHARACTER REMOVAL problem is equivalent (by an exchange of the roles of rows and columns) to ROW DELETION $\left(\begin{smallmatrix}0&0&1&1\\0&1&0&1\end{smallmatrix}\right)$. Again for $r = 2$, PP BY TAXA REMOVAL is the row deletion problem for the transposed (4×2)-matrix, but it can also be reduced to n instances of ROW DELETION $\left(\begin{smallmatrix}0&1\\1&0\\1&1\end{smallmatrix}\right)$. Wernicke *et al.* [30] establish a close relationship between ROW DELETION and the d-HITTING SET problem. The task in the d-hitting set problem is to find a hitting set of size at most k for a family of sets of size at most d. On the one hand, this relation can be used to show that PP BY TAXA REMOVAL and PP BY CHARACTER REMOVAL for $r = 2$ are NP-complete. On the other hand, fixed-parameter algorithms for the d-HITTING SET problem with respect to k at once yield fixed-parameter algorithms for ROW DELETION(B) with respect to the parameter k for matrices B with d rows. For 3-HITTING SET, Fernau [31] gives an algorithm with running time $O(2.18^k + n)$, for 4-HITTING SET Niedermeier and Rossmanith [32] give an algorithm with running time $O(3.30^k + n)$. For the 2-HITTING SET problem, which is the well known vertex cover problem, Chen *et al.* [33] give an algorithm with running time $O(1.29^k + kn)$. □

For larger r, where no characterization in terms of forbidden submatrices is known, the complexity of the removal problems is open.

OPEN PROBLEM 4.2. *Are* PP BY TAXA REMOVAL *and* PP BY CHARACTER REMOVAL *with parameter k fixed-parameter tractable for all r?*

DEFINITION 4.5 (PP BY RECOLORING). *The input for* PP BY RECOLORING *is a matrix A of n taxa and a number k. The task is to decide whether A has a phylogeny with recoloring number of at most k.*

Recall that computing the recoloring number of a *given* phylogeny is fixed-parameter tractable with respect to the parameter b (number of bad states), but nothing is known for the problem PP by recoloring.

OPEN PROBLEM 4.3. *How difficult is* PP BY RECOLORING?

5. COMPUTING DISTANCES BETWEEN PHYLOGENIES

In the present section we study how difficult it is to compute the distance between phylogenies, which need not be perfect. Computing such distances is important when several candidate phylogenies are given, obtained either computationally by different methods or compiled from different literature sources. We discuss three classical, well-known editing distance measures as well as a recently introduced distance measure based on planar embeddings of the involved trees.

One way to define a distance between phylogenies is to count the number of modifications necessary to transform one phylogeny into another. Possible modifications include deletion and insertion of taxa or the movement of a subtree of the phylogeny to another place. For different sets of allowed modifications, we get different notions of distance. Three increasingly general modifications have been studied extensively in the literature, see the book chapter of DasGupta *et al.* [34] for an entry point. Usually these operations are considered only on *undirected, binary* phylogenies in which *taxa label only leaves*.

The first operation is the *nearest neighbor interchange*. In a binary phylogeny, every internal edge has four subtrees attached to it (two at the one end of the edge and two at the other end) and the nearest neighbor interchange exchanges two such subtrees. This means that the tree ${}^{\mathrm{A}}_{\mathrm{B}}{>}\!\bullet\!\!-\!\!\bullet\!{<}^{\mathrm{C}}_{\mathrm{D}}$ can be changed into ${}^{\mathrm{A}}_{\mathrm{C}}{>}\!\bullet\!\!-\!\!\bullet\!{<}^{\mathrm{B}}_{\mathrm{D}}$ or into ${}^{\mathrm{A}}_{\mathrm{D}}{>}\!\bullet\!\!-\!\!\bullet\!{<}^{\mathrm{C}}_{\mathrm{B}}$.

The second operation is the *subtree prune and regraft* operation. Here, we are allowed to cut an edge anywhere in the phylogeny and to reattach (regraft) the subtree that we have cut away at some other place. In detail, starting with a phylogeny, we consider some edge connecting two nodes u and v. Let T_u and T_v be the two subtrees connected by the edge. Then the subtree prune and regraft operation allows us to cut the edge between u and v and to reattach the tree T_v at some other place in T_u. To reattach the tree T_v, we split an edge in T_u by adding a new node in the middle and connect the node v to that new node (Fig. 3). The node u, which now has degree two, is removed to make the tree binary again. Although not quite obvious, it is not hard to see that nearest neighbor interchange is a special case of subtree prune and regraft.

The subtree prune and regraft operation models a *horizontal gene transfer*, where a gene is transfered between unrelated taxa by a mixing of their genetic material. As we pointed out in the remark after Definition 2.1, we cannot hope to fully understand evolutionary processes without taking horizontal gene transfer into account. In a survey of the importance of horizontal gene transfer [38] Gogarten, Doolittle and Lawrence point out that 'accumulating prokaryotic gene and genome sequences reveal that the exchange of genetic information through both homology-dependent recombination and horizontal (lateral) gene transfer (HGT) is far more important, in quantity and quality, than hitherto imagined. The traditional

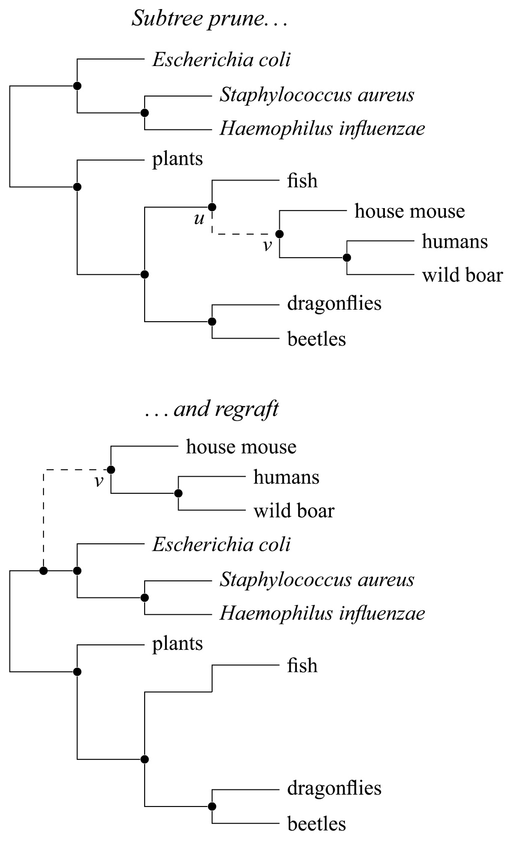


FIGURE 3. Example of how the subtree prune and regraft operation works. In the left phylogeny the edge between u and v is cut and then the tree rooted at v is regrafted at a new position in the right phylogeny. The right phylogeny takes the presence or absence of the gene encoding *N*-acetylneuraminate lyase into account. This gene is present in vertebrates and bacteria but not in the other taxa, suggesting that a horizontal gene transfer took place. The announcement [35] in *Nature* that humans may have acquired over a hundred genes directly from bacteria made newspaper headlines, but the tow *Science* articles *Microbial genes in the human genome: Lateral transfer or gene loss?* [36] and *Are there bugs in our genome?* [37] quickly challenged the findings and suggest other explanations, at least for most genes.

view that prokaryotic evolution can be understood primarily in terms of clonal divergence and periodic selection, must be augmented to embrace gene exchange as a creative force, itself responsible for much of the pattern of similarities and differences we see between prokaryotic microbes.'

The third operation is the *tree bisection and reconnection* operation. This operation is nearly the same as the subtree prune and regraft operation, only we now allow to connect an arbitrary node in T_v to T_u, rather than only the node v. This operation is more general than the subtree prune and regraft operation, but one can simulate a tree bisection and reconnection operation by two subtree prune and regraft operations.

DEFINITION 5.1 (Distance problems). *The input for the three problems* NNI DISTANCE, SPR DISTANCE *and* TBR DISTANCE *are two binary, undirected phylogenies with input taxa labels only at the leaves and a distance d. The task is to decide whether d nearest neighbor interchanges, d subtree prune and regraft operations or d tree bisection and reconnection operations suffice to transform the first phylogeny into the second, respectively.*

Computing the distance between phylogenies turns out to be a hard job. It is known that computing the distance of two phylogenies with respect to either the nearest neighbor interchange operation or the tree bisection and reconnection operation is NP-hard and it is strongly suspected that the same is true for the subtree prune and regraft operation.

THEOREM 5.1 ([39]). NNI DISTANCE *is* NP-*complete.*

The hardness is shown by a rather involved reduction from EXACT COVER BY 3-SETS. Interestingly, the hardness was open for a long time and appeared as an open problem in numerous papers like [40–42], just to name the earlier ones.

OPEN PROBLEM 5.1. *Is* SPR DISTANCE *also NP-complete?* (An NP-completeness proof for SPR distance given in [43] turns out to be incorrect as argued by Allen and Steel [44], but it might be possible to fix the proof.)

THEOREM 5.2 ([43, 44]). TBR DISTANCE *is* NP-*complete*.

THEOREM 5.3 ([44]). TBR DISTANCE *can be solved in time* $O(d^{O(d)} + n^4)$.

No exact time bound is given in [44]; the one stated above is a loose upper bound that we derived by generously bounding the running time of the algorithms given in the paper. As the running time formula suggests, the $O(d^{O(d)} + n^4)$ algorithm uses a kernelization algorithm running in time $O(n^4)$ to reduce the original problem instance to an instance of size at most $O(d)$. This reduced instance can then be solved by brute force.

The kernelization is based on two easy reduction rules: First, if a pendant subtree occurs identically in both trees, then in both trees we replace the subtree by a single leaf that gets a unique new label. Second, if a chain of pendant subtrees occurs identically in both trees, the whole chain can be replaced by a chain of three leafs labeled with three new labels (Fig. 4). The kernelization algorithm simply applies these rules until neither rule can be applied any more. The resulting trees will then have size $O(d)$.

OPEN PROBLEM 5.2. *Are* NNI DISTANCE *or* SPR DISTANCE *with parameter d (distance) also fixed-parameter tractable?*

We conclude this section with a distance measure that was introduced in [45]. It deviates from the above ones in that it is based on planar embeddings of the two trees involved. Given a leaf-labeled tree, a linear ordering on its leaves is called

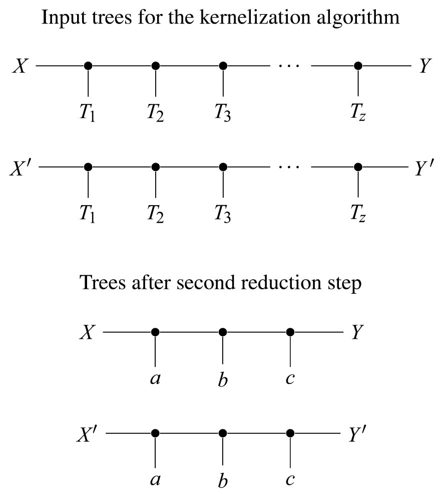

Input trees for the kernelization algorithm
X
Y
T_1
T_2
T_3
T_z
X'
Y'
Trees after second reduction step
a
b
c

FIGURE 4. The second reduction rule in the kernelization algorithm for TBR DISTANCE. Each capital letter is a subtree, each lowercase letter is a label.

suitable if the tree can be embedded into the plane such that its leaves are mapped to a straight line in which the given order is maintained. Given two orderings on the same label set, their *crossing number* is the number of edge crossings when drawing the orderings onto two parallel layers and connecting the corresponding labels by edges (see Fig. 5 for an example). We then obtain a definition of distance for trees as follows:

DEFINITION 5.2 (Crossing distance). *Given two leaf-labeled trees T_1 and T_2 with the same leaf set, their* crossing distance *is the minimal crossing number between two suitable orderings, one with respect to T_1 and one with respect to T_2.*

Note that under this definition trees with different topologies may have distance 0.

DEFINITION 5.3 (Crossing distance). *The input for* 5.3 CROSSING DISTANCE *are two leaf-labeled trees T_1 and T_2 with the same n element leaf set and a distance d. The task is to check whether the crossing distance between T_1 and T_2 is at most d.*

The problem is called TWO-TREE CROSSING MINIMIZATION by Fernau *et al.* [45]. They show that it is NP-complete, but fixed-parameter tractable with respect to parameter *d*.

THEOREM 5.4 ([45]). CROSSING DISTANCE *is* NP-*complete.*

THEOREM 5.5 ([45]). CROSSING DISTANCE *can be solved in time $O(2^{10d} \cdot n^{O(1)})$.*

The problem can be solved by a search tree algorithm that, recursively, identifies 'conflicting' subsets of four leaf labels, that is, four labels for which no suitable orderings without crossing can be found. The algorithm branches recursively for each possible orderings, one in each tree, for identified leaves. In the case in which no size-four conflicting subset is found, a more complex branching into a fixed number of branches can be given.

Unfortunately, because of its high running time the above result merely classifies the problem as fixed-parameter tractable.

OPEN PROBLEM 5.3. *Give a practical fixed-parameter algorithm for computing the crossing distance.*

6. COMBINING PHYLOGENIES

In this section we study approaches to combining several phylogenies into a single phylogeny. Suppose two researchers have accumulated character data for two partially overlapping sets of taxa and both have constructed phylogenies based on their data (see Fig. 6 for an example). A natural question to ask is, how can we combine these two phylogenies into a single phylogeny?

The first approach is to combine the character–state matrices into a *supermatrix* (as it is called in [46]) and to build a phylogeny based on this combined *primary data* or *total evidence* (as it is called in [47]). Another approach,

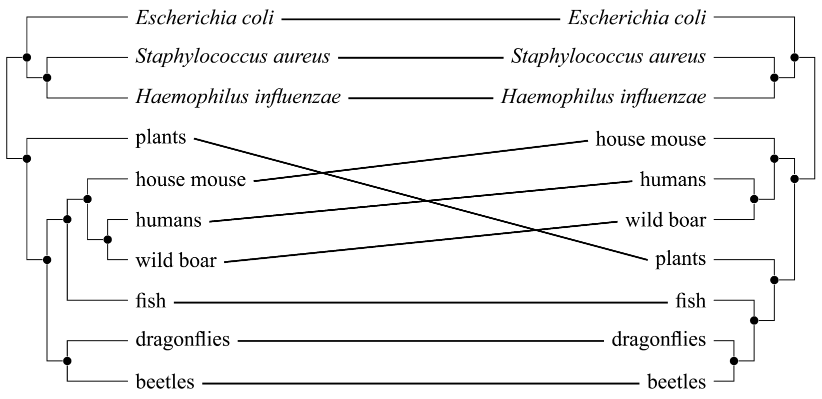


FIGURE 5. Visualization of the crossing number computation for the two phylogenies from Fig. 3. The two phylogenies are drawn in such a way that the taxa lie on two parallel lines. Three crossings result when identical taxa in the different phylogenies are connected.

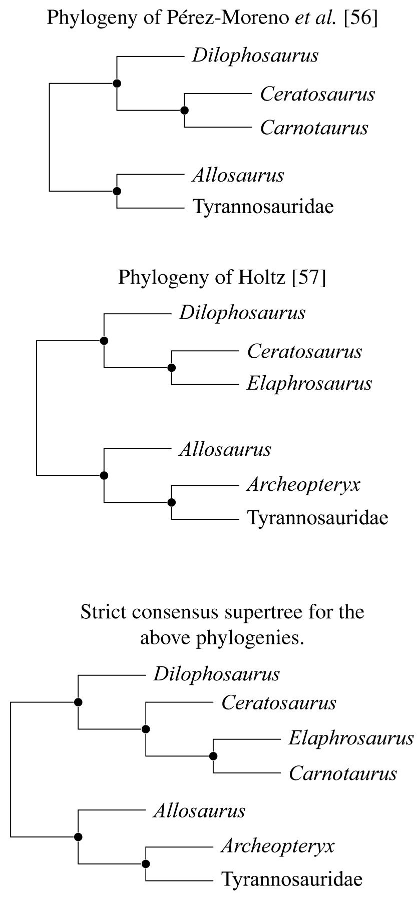


FIGURE 6. (Parts of) Two phylogenies for Dinosauria from two different publication and a strict consensus supertree for them.

which has gained momentum only in recent years, is to ignore the primary data and to build a phylogeny based only on the *topologies of the two phylogenies.* Phylogenies that are based on the topology of other phylogenies rather than on the underlying character–state data are called *supertrees.*

An obvious shortcoming of the supertree approach is that we expect phylogenies based on total evidence to be more exact than phylogenies based only on 'second-hand, indirect data' like tree topologies. To make matters worse, the supertree approach can yield phylogenies that are outright contradictory to the primary data. Nevertheless, over the last years numerous papers have presented supertrees, motivated by a number of arguments that count in favor of the supertree approach:

- The literature contains thousands of phylogenetic studies. When combining published phylogenetic trees to obtain larger trees, it is often hard or impossible to revisit the underlying methods or data. For example, Pisani *et al.* [48] use phylogenies from 124 different publications, some of which date back decades, to construct a supertree for 277 Dinosauria genera.
- In order to increase efficiency, one can try to compute phylogenies in a two-phase process. In a first phase, one computes small trees based on a phylogenetic method of choice. Because the trees are small, one can use time-intensive methods. In a second phase, one combines these trees into one phylogeny.
- Phylogenetic trees can be computed based on different character sets and the task is to combine the resulting trees into a single supertree. Not all data may be available for all taxa of interest, for instance genomic sequences may be available only for a small set of species, making it hard or impossible to construct a supermatrix for the primary character data.

The term 'supertree' stems from the 1986 paper *Consensus supertrees: the synthesis of rooted trees containing overlapping sets of labeled leaves* by Allan Gordon [49]. However, the strict consensus supertrees of Gordon can only be built for conflict-free input phylogenies, which are only rarely available. Today, the term is also used for trees constructed using methods that handle conflicting input phylogenies more gracefully.

In Section 6.1, we investigate strict consensus supertrees, because it is helpful to understand the simple case of no conflicts first before tackling more difficult settings, but also because a strict consensus supertree, in the few cases it exists, is of great interest. The question whether such a strict consensus supertree exists for given input phylogenies is solvable in polynomial time [50] for directed trees, but is NP-hard for undirected trees. This leads us to have a look at the problem from a parametrized point of view.

There is more than one way to build a supertree from conflicting input phylogenies. A first, and rather straightforward idea is to resolve conflicting input phylogenies by deleting a minimal number of the conflicting input trees so that all conflicts are resolved. In the second part of Section 6.1 we present such a fixed-parameter algorithm for the case in which all input trees are unrooted and have four leaves.

Second, we can resolve conflicts by leaving out a bounded number of input taxa from the analysis. In Section 6.2, we consider a version of this problem in which the input trees are rooted and share the same leaf set.

Third, we name a method called *matrix representation with parsimony* (MRP), which was proposed independently by Baum [51], Doyle [52] and Ragan [53]. This method transforms the combination of directed input phylogenies into a maximum parsimony problem on a binary character–state matrix and then computes a maximum parsimony tree for this matrix. One can compute a maximum parsimony tree by applying the fixed-parameter algorithm for PHYLOGENETIC PENALTY MINIMIZATION presented in Section 4 where the parameter is the 'deviation' from being a perfect phylogeny.

Beyond that, it seems that parametrized algorithms tailored to the case of MRP have not been studied so far. But see [54] for a problem formulation slightly changing the optimization goal of MRP and a fixed-parameter result for a constrained case.

For a more detailed discussion and critical appraisal of the different supertree methods, we refer the reader to the monograph edited by Bininda-Emonds [55].

6.1. Combining phylogenies using strict consensus supertrees

For every method, including the strict consensus supertree method, the most basic problem is to decide whether a supertree exists. For the next definitions recall that in a binary phylogeny all nodes have degree one or three.

DEFINITION 6.1 (Strict consensus supertree). *A phylogeny T induces a phylogeny T' if T' can be obtained from T by repeatedly deleting leaves and contracting edges. A phylogeny T is a* strict consensus supertree *of trees $T_1, \ldots, T_t$ if each T_i is induced by T.*

DEFINITION 6.2 (COMPATIBLE UNDIRECTED PHYLOGENIES). *The input for* COMPATIBLE UNDIRECTED PHYLOGENIES (*abbreviated* CUP) *are binary phylogenies $T_1, \ldots, T_t$. The task is to decide whether there is a binary strict consensus supertree for $T_1, \ldots, T_t$.*

Already this basic problem turns out to be hard.

THEOREM 6.1 ([14]). CUP *is* NP-*complete, even if all input trees have four leaves.*

The corresponding problem for *directed* trees is solvable in time $O(n^3)$ for n taxa using an algorithm of Aho *et al.* [50]. Steel [58] raised the question of whether the undirected version is fixed-parameter tractable with respect to the number t of input trees. This parametrization is reasonable since the combination of a small number of possibly large trees is a realistic scenario. Bryant and Lagergren have recently answered Steel's question positively.

THEOREM 6.2 ([59]). CUP *can be solved in time*

$$O(f(t) \cdot n^{O(1)})$$

for some function f.

Proof. First, the input trees can be combined to a *display graph* that has tree-width t, provided CUP has a solution. Second, the problem CUP can be described using a monadic second-order formula on the constructed display graph. Fixed-parameter tractability follows because of two classical results from parametrized complexity theory: determining whether a given input graph has tree-width t is fixed-parameter tractable with respect to parameter t, [60], and evaluating a monadic second-order formula on graphs of tree-width t is also fixed-parameter tractable with respect to the parameter t [61]. □

Unfortunately, both theoretical results on which the fixed-parameter algorithm for CUP is based are, indeed, theoretical and do not have efficient, practical implementations. No one has yet bothered to determine an explicit upper bound on the function f mentioned in the above theorem.

OPEN PROBLEM 6.1. *Give an efficient and practical parametrized algorithm with explicit running time bounds for* CUP *for the parameter t (number of input trees).*

A parametrization of CUP with respect to the maximum size of the input trees does not even lead to a 'theoretical' fixed-parameter algorithm by Theorem 6.1. On the other hand, the problem is fixed-parameter tractable with respect to the total number of n of input taxa since we can try all possible tree topologies over the taxa (see also Theorem 2.2 and Open Problem 2.1).

In practice, multiple phylogenies can only rarely be combined into a strict consensus supertree. Similar to the case of input matrices that do not permit a perfect phylogeny, we must now find ways of resolving the conflicts. Perhaps, the simplest approach is to delete potentially erroneous input trees until a solution can be found. Here, the number of deleted trees is a natural problem parameter.

DEFINITION 6.3 (CUP BY TREE REMOVAL). *The input for* CUP BY TREE REMOVAL *is the same as for* CUP *plus a number k. The task is to remove at most k trees from the input such that the remaining trees are an instance of* CUP.

Theorem 6.1 implies that the above problem is NP-complete for $k = 0$ even for the extreme case that all input trees are *quartet* trees (binary trees with four leaves as in Fig. 7); so it is unlikely that we will make progress on the fixed-parameter tractability of CUP BY TREE REMOVAL. However, in one particular case there is, at least, still hope:

OPEN PROBLEM 6.2. Is CUP BY TREE REMOVAL with parameter k fixed-parameter tractable when we allow only quartets as input and all of them share a common taxon? (Note that a set of quartets that share a common taxon can be thought of as a set of directed triplets.)

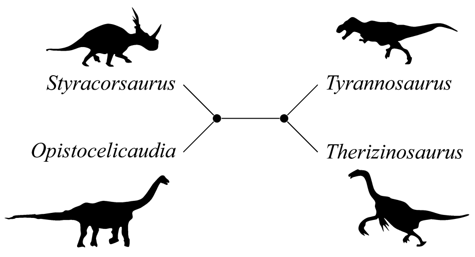


FIGURE 7. A quartet for four Dinosauria genera.

The situation is more favorable when we turn towards the following 'dense' version of the problem:

Definition 6.4 (minimum quartet inconsistency). *The input for* minimum quartet inconsistency *is a set S of n taxa, a set $\mathcal{Q}$ containing a quartet tree for each four element subset of S and a number k. The task is to remove k quartets from $\mathcal{Q}$ so that the remaining quartets have a binary supertree T.*

Theorem 6.3 ([62]). minimum quartet inconsistency *can be solved in time $O(4^k \cdot n + n^4)$.*

The main idea is to employ old results by Bandelt and Dress that allow one to lead 'global' conflicts in the input set back to 'local' conflicts of three quartet trees, and to determine that there are exactly four ways to resolve a local conflict. This results in a search tree algorithm with the stated running time.

Note that the running time in Theorem 6.3 is linear in the input size since there are $O(n^4)$ input quartets for n taxa. The algorithm described in [62] also exhibits how search tree algorithms can be complemented by heuristic strategies to prune the search space beyond the running time guarantee.

Considering arbitrary (non-quartet) input trees, it is neither clear how the notion of 'denseness' should be defined in order to get a result similar to Theorem 6.3, nor how the ideas of the algorithm for minimum quartet inconsistency, which are tailored to quartet trees, could be adapted.

Open Problem 6.3. *Define an appropriate notion of 'denseness' for non-quartet trees. What is the parametrized complexity of the resulting problem?*

6.2. Combining phylogenies using agreement subtrees

Combining phylogenies using strict consensus supertrees is rarely possible in practice, but *always* bound to fail when we wish to combine multiple phylogenies over *identical* leaf sets—a situation that arises in important applications. For example, common heuristic phylogeny reconstruction methods that optimize a maximum parsimony criterion or a maximum likelihood criterion usually produce several optimal or near-optimal trees. Choosing one of the near-optimal trees arbitrary is, well, arbitrary and a 'consensus' of the trees may be preferable. *Ad hoc* methods for finding a consensus like the majority consensus tree method work in polynomial time—the randomized algorithm presented in [63] runs in linear time for instance—but they may yield poorly resolved output trees. In the following, we discuss a more sophisticated version of the consensus problem.

For the rest of this section we consider only directed phylogenies, which no longer need to be binary.

Definition 6.5 (maximum agreement subtree). *The input for* maximum agreement subtree *is a set S of n taxa, directed input trees $T_1, \dots, T_t$ over S, and number k. The task is to find a subset $S' \subseteq S$ of size $n - k$ such that there is a directed phylogeny T over S' such that each of $T_1, \dots, T_t$ induces T.*

Perhaps not surprisingly, this problem is NP-complete. The following theorem shows that the situation is even worse:

Theorem 6.4 ([64]). maximum agreement subtree *is* NP-*complete even for $t = 3$.*

Concerning the maximum degree d of nodes in the trees, the following result is known, which places the problem at least in the class XP with respect to the parameter d.

Theorem 6.5 ([65]). maximum agreement subtree *can be solved in time $O(n^d + tn^3)$.*

For a more complete overview on agreement subtrees we refer to [66]. For us, it is of particular interest that maximum agreement subtree is fixed-parameter tractable with respect to parameter k (number of removed taxa):

Theorem 6.6 ([66]). maximum agreement subtree *can be solved in time $O(2.18^k + tn^3)$ and also in time $O(3^k \cdot tn)$.*

We can restrict attention to 'local' conflicts formed by three taxa, at least one of which has to be removed. This allows us to reduce maximum agreement subtree to 3-hitting set. Applying the best known fixed-parameter algorithm for 3-hitting set [36] yields the first bound. Omitting the time-intensive reduction in the beginning and searching one local conflict at a time yields a $O(3^k \cdot t\,n)$ time algorithm.

The result can be extended to the closely related maximum compatibility tree problem [9]. For input trees with non-identical leaf sets, Berry and Nicolas show that the resulting problem maximum agreement supertree becomes W[2]-hard, even when each input tree has only three leaves. Consequently, a fixed-parameter algorithm corresponding to Theorem 6.6 is out of reach.

7. APPLICATIONS OF PHYLOGENIES

In this final section before the conclusion we present two applications of phylogenetics that are not related to taxonomy. In these applications we are not ultimately interested in a solution phylogeny. Rather, we use a phylogeny or the phylogenetic model to determine something seemingly unrelated. In the first application we use a phylogeny to help in the prediction of regulatory elements of the genome, in the second application we use perfect phylogenies as a measure of the quality of haplotype phase predictions.

7.1. Prediction of regulatory elements using phylogenetic footprinting

The cells of the human body permanently need to produce different proteins like, for instance, hemoglobin, which picks up oxygen in the lungs and transports it through the body.

Every protein is uniquely identified by the sequence of amino acids that, chained together, form this protein; for example, the sequence of hemoglobin starts with valine, histidine, leucine, threonine, proline and contains a total of 146 amino acids. This sequence is encoded in the genome using a sequence of codons (blocks of three bases) where each codon encodes one amino acid. A *gene* is, in essence, just a sequence of codons in the DNA that tells the cell to produce, say, hemoglobin. The human genome has tens of thousands of genes for all the different proteins that need to be produced.

The process of translating the base sequence of a gene into a protein is called *gene expression*. However, cells do not simply express all genes in the genome at all times. Once, say, the cell membrane has been constructed, the expression of the protein(s) for the cell membrane needs to be stopped (or, at least, reduced). For this reason, gene expression is *regulated* by other parts of the genome. Before or after a gene there are base sequences in the genome that are involved in the inhibition or promotion of gene expression, depending on which other proteins are present. These sequences in turn can again be regulated by other regulatory elements in the genome, leading to a highly complex regulatory network. Understanding this network is one of the most challenging, interesting and important tasks of molecular biology.

Phylogenetic footprinting, first proposed by Tagle *et al.* [67], is a method for predicting which regions of the genome are regulatory (involved in the regulatory process). The basic idea relies on the following observation: Suppose we have identified a gene and we expect that there are regulatory elements before and after the gene, but we do not know where they are exactly. Then we expect that regulatory elements, which are as important as the genes themselves for the survival of an individual, will not change (greatly) as mutations occur throughout the genome. If a non-regulatory part mutates, this does not change the chances of survival, but when a mutation occurs inside a gene or a regulatory area, then the individual may not survive. One says that the regulatory elements are *under pressure of selection*, while the surrounding parts of the genome are not. Thus, a possible approach to predicting regulatory elements is to do a sequence alignment of multiple genomic data and to search for parts of the genome that stay (relatively) stable over evolutionary time spans amid parts of the genome that vary.

In phylogenetic footprinting one attempts to improve the prediction using a phylogenetic tree to judge how important a mutation is. If we see only, say, three different sequences in a candidate regulatory region, but the sequences of closely related species vary strongly between the three sequences, we are less likely to believe that the region is regulatory than if related species all share the same sequence inside the regulatory region.

The above ideas lead to a problem called *substring parsimony problem*. To state it formally we first define the *parsimony score*.

Definition 7.1 (Parsimony score). *Recall the notion of the length of a phylogenetic tree from Definition 3.1. Given a partially labeled phylogenetic tree T, the* parsimony score *of the tree is the minimal length of a label completion of T.*

Definition 7.2 (substring parsimony). *The input for* substring parsimony *is a partially labeled phylogeny T in which exactly the leaves are labeled and two integers l and s. The task is to decide whether each leaf label can be replaced by a substring of length l such that the parsimony score of the resulting tree is at most s.*

The substrings of length l that we choose from each leaf are the predicted regulatory elements. Note that in the substring parsimony problem the phylogeny T is fixed and part of the input. The idea is that it is typically already available in the literature or can be computed using one of the method presented in the previous sections.

Blanchette, Schwikowski and Tompa prove the following theorem:

Theorem 7.1 ([68]). substring parsimony *can be solved in time* $O((r^{2l} + m) \cdot ln)$.

The problem can be solved using a dynamic program, although the entries of its 'table' are attached to the nodes of the phylogeny and the 'table' is built from the leaves inward. The table entry for a node u of the phylogeny is a table once more, which stores a number for each state vector of length l, of which there are r^l many. The number stored for a state vector is the best parsimony score that can be achieved for the subtree rooted at u, if u is labeled with the state vector. Building the initial table entries for the leaves is easy (there, all numbers in a table for a leaf are either 0 or infinity) and combining two tables takes time $r^l \cdot r^l$.

The theorem shows that substring parsimony is in FPT with respect to the parameter pair (r, l). The parameter r is 4 in practice, but even for this low value the dominating part of the running time is $r^{2l} = 16^l$, which grows too quickly. Therefore, Blanchette *et al.* develop a number of improvements for the original algorithm and lower the dominating term first to r^l and even further for typical inputs.

In the same paper, Blanchette *et al.* also consider a generalization of substring parsimony. Sometimes regulatory elements may lose their significance when the gene they regulate is no longer important or when another regulatory element takes over. In this case, there is selective pressure only in a subtree of the phylogenetic tree, while in the rest of the tree the parsimony score of the regulatory element will be high.

To handle these losses, Blanchette *et al.* propose to search for substrings that have a low parsimony score but still span a large part of T. To determine the 'size' of a subtree, we do not count the number of elements, but allow a more general measure: We assign an *age* to each edge of T (in [68] the age is called the *length* of the edge, but this term is defined differently in the present paper). Then the *age of a subtree* is the sum over the ages of the edges. The substring parsimony problem *with losses* can now be defined as follows (the below definition is slightly simplified compared with [69]).

Definition 7.3 (Substring parsimony with losses). *The input for the* substring parsimony with losses *problem is a partially labeled phylogenetic tree T in which exactly the leaves are labeled, a labeling function that assigns an age to each edge of T, two integers l and s and a minimum age α. The task is to decide whether one can prune the tree (remove leaves repeatedly) and replace each remaining leaf label by a substring of length l such that the parsimony score of the resulting tree is at most s and the age of the tree is at least* α.

Note that this problem is a generalization of substring parsimony: Setting α to the age of the input tree T enforces that no pruning takes place.

Theorem 7.2 ([68]). substring parsimony with losses *can be solved in time* $O((r^{2l} + m) \cdot ln)$.

A similar dynamic programming approach can be used as for the basic version of the problem.

7.2. Prediction of haplotype phase using perfect phylogenies

The *haplotype phase determination problem* or just *haplotyping problem* arises when one searches for genetic variations of diploid organisms like humans. An example of important genetic variations are *single nucleotide polymorphisms* (SNPs), which we mentioned earlier. They are variations across the population of a single nucleotide in the genome. Thus, some people may have a DNA string where at a specific position on a chromosome the nucleobase is adenine and some other people may have guanine at the same position (Fig. 8). Knowing which nucleobase is present can be important for the prediction of drug response or susceptibility to diseases.

Suppose we wish to determine quickly and inexpensively (for example, in a hospital during a study on drug response) the state of a specific SNP for a person. This can be done, in essence, by sampling cells from a drop of blood, extracting the DNA, using a polymerase chain reaction to increase the amount of DNA present, and then mixing in two primers. The primers can be thought of as two keys, one that will fit when the nucleobase is adenine at the specific SNP site and one that will fit when the nucleobase is guanine. Then, depending on which primer reacted, the result will either glow, say, red or green under fluorescent light (or, frustratingly, not at all, when one of the steps in the experiment was not executed correctly).

Humans are diploid organisms, which means that we have two specimens of each chromosome, one inherited from the mother and one from the father. But, then, it is possible that we inherit one DNA string with adenine at the SNP site and another DNA string with guanine. These two DNA strings are called the *haplotypes*. The fact that we have two haplotypes means that in the experiment the two primers may also *both* glow, which indicates the *heterozygous state* in which we have inherited two haplotypes with different states, or only one may glow, which indicates one of the two *homozygous states* in which both haplotypes agree.

A problem arises when we consider two SNP sites. For this, we do two experiments, each with two special primers. Suppose that we learn from the experiment that both sites are heterozygous. Then we know that, say, at the first position there is adenine on one haplotype and guanine on the other and at the second position there is, say, cytosine on one haplotype and guanine on the other. This information is also called the *genotype* of the person (for these two SNP sites). What we do *not* know is whether the haplotype on which there is guanine at the first position is the same haplotype as the one on which there is guanine at the second position. There might have been a 'switch' from one SNP site to the other, better known as a phase shift. This effect is depicted in Fig. 9.

Determining the two haplotypes of a single genotype is impossible, unless context information is available. If there are h heterozygous SNP sites, then there are 2^{h-1} pairs of haplotypes that explain the observed genotype and each pair is equally likely *a priori*. In order to determine the phase of the haplotypes, we must make assumptions about which haplotypes are more likely than others. For example, we might assume that haplotypes change only rarely (they certainly do not change within a few generations). Then if we are given

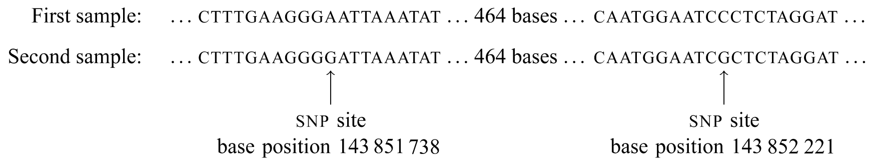

First sample: ... CTTTGAAGGGAATTAAATAT ... 464 bases ... CAATGGAATCCCTCTAGGAT ...
Second sample: ... CTTTGAAGGGGATTAAATAT ... 464 bases ... CAATGGAATCGCTCTAGGAT ...
SNP site
base position 143 851 738
SNP site
base position 143 852 221

FIGURE 8. Parts of two different samples of human DNA base sequences. These sequence are inside the gene PRKAB2, which is located on the human chromosome 1 between base positions 145 093 309 and 145 110 753 in the reference haplotype of the Human Genome Project. It is a regulatory subunit of protein kinase and is highly expressed in skeletal muscle and may have tissue-specific roles. Inside the gene, the base sequences of any two samples are identical except at SNP sites. Two of them are shown, namely first rs6937 and then rs1047140 in the nomenclature of the US National Center for Biotechnology Information [69], from whose genome map the data was obtained.

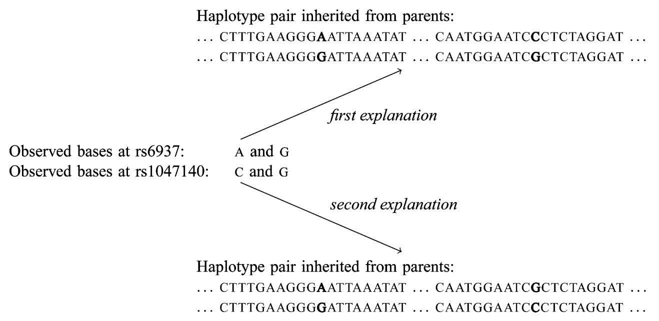

Haplotype pair inherited from parents:
... CTTTGAAGGG**A**ATTAAATAT ... CAATGGAATC**C**CTCTAGGAT ...
... CTTTGAAGGG**G**ATTAAATAT ... CAATGGAATC**G**CTCTAGGAT ...
first explanation
Observed bases at rs6937: A and G
Observed bases at rs1047140: C and G
second explanation
Haplotype pair inherited from parents:
... CTTTGAAGGG**A**ATTAAATAT ... CAATGGAATC**G**CTCTAGGAT ...
... CTTTGAAGGG**G**ATTAAATAT ... CAATGGAATC**C**CTCTAGGAT ...

FIGURE 9. The haplotype phase determination problem arises when we observe two heterozygous SNP sites in the genotype. In this case, there are two different pairs of haplotypes that explain the observed genotype. The haplotype phase determination problem asks us to decide which explanation is correct.

the genotypes of hundreds of persons of the same ethnic group, we can try to find a minimal set of haplotypes such that every observed genotype can be explained by assuming that the person has two haplotypes from the small set. Many statistical methods for haplotype phase determination are based on this parsimony assumption.

In a seminal paper, Gusfield [15] proposed a different idea. Gusfield argues that haplotypes evolve according to the evolutionary model underlying perfect phylogenies: Mutations occur only rarely and there are no back-mutations. Therefore, we should look for a set of haplotypes explaining the genotypes that forms a perfect phylogeny (the taxa being the haplotypes, the SNP sites being the characters, and the nucleobases being the states). The following definitions formalize the problem.

DEFINITION 7.4 (Haplotype, genotype). *A* haplotype *is a state vector. The set Σ_i of permissible states at position i is typically (but need not be) a two element subset of* {A, C, G, T}. *A* genotype *is a sequence of sets, where the i-th set is a subset of size one or two of Σ_i. Two haplotypes* explain *a genotype if the i-th subset of the genotype contains exactly the two states of the i-th positions of the two haplotypes.*

DEFINITION 7.5 (PP HAPLOTYPING). *The input for the* PP HAPLOTYPING *problem is a set of genotypes. The task is to decide whether there exists a set of haplotypes forming a perfect phylogeny such that each genotype can be explained by two haplotypes in the set.*

The PP HAPLOTYPING problem is at least as hard as the PP problem since we can reduce PP to PP HAPLOTYPING by turning each taxon into a 'genotype' whose *i*-th set contains only the *i*-th state of the taxon. Then every set of 'haplotypes' that explains the 'genotypes' contains the original set of taxa. This shows that PP HAPLOTYPING is NP-complete.

The question arises which fixed-parameter results on the PP problem carry over to the more general haplotyping problem. Not too much is known on this since research has almost entirely focused on the case $r = 2$. For this, the following remarkable result is known:

THEOREM 7.3 ([70]). *For $r = 2$,* PP HAPLOTYPING *can be solved in time $O(mn)$.*

OPEN PROBLEM 7.1. *How difficult is* PP HAPLOTYPING *for $r > 2$?*

In practice, the perfect phylogeny haplotyping problem is, unfortunately, not quite the problem that we want to solve. Genotype data that is obtained via the laboratory process sketched earlier will always contain a certain amount of *missing data* caused by impurities or incorrect handling. Such missing data is commonly represented by question mark entries in the genotype input.

DEFINITION 7.6 (INCOMPLETE PP HAPLOTYPING). *The input for* INCOMPLETE PP HAPLOTYPING *is a set of genotypes that may contain question marks for certain characters. The task is to decide whether the question mark entries can be completed in such a way that the resulting set of genotypes is an instance of* PP HAPLOTYPING.

The missing entries add yet another level of complexity. This new problem, which is of great practical interest, is (presumably) no longer fixed-parameter tractable with respect to the central parameter *r*. Indeed, the problem is (presumably) not even in XP as the following theorem shows, which was proved independently by Kimmel and Shamir and also by Gramm, Nierhoff, Sharan and Tantau.

THEOREM 7.4 ([71, 72]). *For every $r \geq 2$,* INCOMPLETE PP HAPLOTYPING *is* NP-*complete*.

The proof in [72] even shows something considerably stronger: the incomplete perfect phylogeny haplotyping problem is still NP-complete for $r = 2$ if we impose a number of restrictions on the phylogeny, like being a path. The result is proved using a reduction for NOT-ALL-EQUAL-3-SAT.

Because of the above result, already for $r = 2$ we have to look for some new parametrizations if we wish to find a fixed-parameter haplotyping algorithm that can deal with missing data. An obvious parametrizations is to consider the total number *q* of question mark entries in the data.

THEOREM 7.5. For $r = 2$, INCOMPLETE PP HAPLOTYPING *can be solved in time $O(3^q \cdot mn)$.*

Proof. There are only $r + \binom{r}{2}$ ways in which a question mark can be completed at any given position (namely by *r* different singleton sets and $\binom{r}{2}$ different two-element sets of states). So we can build all $\left(r + \binom{r}{2}\right)^q$ possible completions and then test for each whether it is an instance for the perfect phylogeny haplotyping problem. For $r = 2$, we know how to solve this in linear time. □

OPEN PROBLEM 7.2. *How difficult is* INCOMPLETE PP HAPLOTYPING *for $r > 2$?*

Unfortunately, the total number q of question marks typically is *not* small in practice. Because of this, a different parameter was studied by Gramm *et al.* in [72], namely the maximal number c of question mark entries *per character*. An analysis of publicly available genotype data shows that, typically, this parameter is reasonably small. The second key idea of the paper is to assume that phylogenies are directed and that they are *paths* (no branching occurs, except possibly at the root). At first sight it may seem strange to consider path phylogenies, but in the human genome for around three-quarters of the genomic loci one finds genotypes where all SNP sites are heterozygous [73]. The only phylogenies that explain such highly heterozygous genotypes are path phylogenies.

THEOREM 7.6 ([72]). *For* $r = 2$, INCOMPLETE PP PATH HAPLOTYPING *can be solved in time*

$$O(3^{O(c^2 \cdot 6^c \cdot c!)} \cdot n^2 m^3).$$

The algorithm starts with a preprocessing phase in which the input is simplified, quite similar to the way a kernelization works. However, the result of the preprocessing is not a problem kernel and its size still depends on the input size rather than solely on the input parameter. Nevertheless, the preprocessing enables us to run a dynamic program that is built in order of increasing so-called *leaf count ranges*.

Currently, an effort is undertaken to implement the algorithm from the above theorem in order to find out whether the algorithm can be applied in practice (it is expected that the algorithm will be more efficient than the worst-case analysis suggests).

OPEN PROBLEM 7.3. *How difficult is* INCOMPLETE PP PATH HAPLOTYPING *for* $r > 2$*?*

OPEN PROBLEM 7.4. *Find a fixed-parameter algorithm for* INCOMPLETE PP HAPLOTYPING *for the parameter c (maximum number of ?-entries per column).*

8. CONCLUSION

Fixed-parameter algorithms are a valuable tool in phylogenetics. We have seen that phylogenetics abounds in computational problems, many of which are NP-complete. Hence we do not expect that efficient exact algorithms will be available for them in the near future, if ever. However, we also saw that many of the computational problems can be solved efficiently and exactly if some of the natural input parameters are reasonably small. Table 3 summarizes how natural input parameters influence the (in)tractability of computational problems in phylogenetics.

In addition to the concrete open problems that we pointed out throughout this survey, in the following we sketch two broader, less concrete prospective directions of future research.

8.1. Future research field: from discrete to stochastic problems

The results presented in this survey refer to problem formulations for discrete input objects and discrete optimization criteria. In computational biology there is a general lack of and a need for fixed-parameter results addressing non-discrete computational problems arising in stochastic analyses. Examples include probabilistic sequence analysis [74] and maximum likelihood analysis.

A concrete stochastic computational problem is the following: The input for MAXIMUM LIKELIHOOD PHYLOGENY is a character–state matrix and transition probabilities for the transitions between character states. The task is to find a phylogeny with the input taxa at the leaves that has a maximal ‘likelihood’ among all such phylogenies. Intuitively, the likelihood of a phylogeny is the sum of the likelihoods that the character states at the leaves were generated given the labeling of the inner nodes. Computing this likelihood is a non-trivial task itself (see for instance [74–76] for details). Only recently it has been shown that MAXIMUM LIKELIHOOD PHYLOGENY is NP-hard [75, 76]. It remains open to address this and related problems with appropriate fixed-parameter algorithms.

8.2. Future research field: from phylogenetic trees to networks

The basic assumption made in this survey, namely that hypotheses on evolutionary history can be represented by trees, is often inappropriate. Phylogenetic trees cannot explain—among other biological effects—the *recombination* effect, where a genomic sequence is combined from two source sequences by taking a prefix from the first and a suffix from the second sequence. The resulting evolutionary history can no longer be represented by a tree; rather, we must use graphs.

Fixed-parameter algorithms might be particularly useful in the study of these graphs since they are not arbitrary, but ‘tree-like.’ They deviate from trees only by a small amount and we propose this extent of deviation (however it is measured) as a natural problem parameter.

There are numerous ways in which we can define ‘tree-likeness’; we just highlight one particular extension of perfect phylogenies to networks. A *phylogenetic network* is a directed acyclic graph whose nodes have in-degree 0 (the root node), in-degree 1 (‘regular’ inner nodes of a phylogeny) or in-degree 2 (the recombination nodes) together with a labeling function that assigns character–state vectors to the nodes. A ‘perfect’ phylogenetic network is defined the same way as a perfect phylogeny, but with an additional rule for recombination nodes: We can split the character–state vector label of recombination nodes into two parts such that the first part is a prefix of the label of one of the parent nodes and the second part is a suffix of the label of the other parent node. The number of recombination nodes is a measure of the network's deviation from being a tree. The computational problem is now to find a phylogenetic network for a given set of taxa that minimizes the number of recombination nodes. This problem is NP-complete [77], but it is open whether it is fixed-parameter tractable with respect to the number of recombination nodes. A partial answer was given by Gusfield *et al.* [78, 79] who show that the problem can be solved efficiently for $r = 2$ if we restrict the phylogenetic networks to so-called *galled trees*. In another direction, the computational problem of finding a phylogenetic network of maximum parsimony is studied in [80] and shown to be fixed-parameter tractable. Many of the problems addressed in this survey can be extended to phylogenetic networks, but almost all of the resulting problems are open.

Concluding this paper, we invite the reader to dive into the presented field, to follow the pointers to literature, to pick up open problems and to come up with exciting new or improved fixed-parameter algorithms.

TABLE 3. Summary of the results on the fixed-parameter tractability of the surveyed problems, sorted according to parameters.

Parameter(s)	Problem	Complexity
Number of taxa (n)	PP	$\in$ FPT
of character (m)	PP	$\in$ XP and W[t]-hard for all t
of states per character (r)	PP	$\in$ FPT
	PP BY TAXA REMOVAL	$\notin$ XP, unless P = NP
	TP BY TAXA REMOVAL	$\notin$ XP, unless P = NP
	PP BY CHARACTER REMOVAL	$\notin$ XP, unless P = NP
	TP BY CHARACTER REMOVAL	$\in$ FPT (trivially)
	PP BY RECOLORING	open
	TP BY RECOLORING	open
	PHYLOGENETIC PENALTIY MINIMIZATION	$\in$ FPT for fixed p
	PHYLOGENETIC NUMBER MINIMIZATION	open
	PHYLOGENETIC BAD STATES MINIMIZATION	open
	PP HAPLOTYPING	open
	INCOMPLETE PP HAPLOTYPING	$\notin$ XP, unless P = NP
	INCOMPLETE PP PATH HAPLOTYPING	$\notin$ XP, unless P = NP
Removals (k) of taxa	PP BY TAXA REMOVAL	$\in$ FPT for $r = 2$
	TP BY TAXA REMOVAL	$\in$ XP, otherwise open
	MAXIMUM AGREEMENT SUBTREE	$\in$ FPT
of characters	PP BY CHARACTER REMOVAL	$\in$ FPT for $r = 2$
	TP BY CHARACTER REMOVAL	$\in$ FPT (trivially)
of trees	CUP BY TREE REMOVAL	$\notin$ XP, unless P = NP
	MINIMUM QUARTET INCONSISTENCY	$\in$ FPT
Bad states (b)	TP BY RECOLORING	$\in$ FPT
State changes (k)	TP BY RECOLORING	$\in$ XP, otherwise open
	PP BY RECOLORING	open
State changes and r (k, r)	PP BY RECOLORING	$\in$ XP, otherwise open
Penalty (p)	PHYLOGENETIC NUMBER MINIMIZATION	$\notin$ XP, unless P = NP
	PHYLOGENETIC BAD STATES MINIMIZATION	$\notin$ XP, unless P = NP
	PHYLOGENETIC PENALTY MINIMIZATION	$\notin$ XP, unless P = NP

Continued

TABLE 3. *Continued*

Parameter(s)	Problem	Complexity
	PHYLOGENETIC PENALTY MINIMIZATION	$\in$ FPT for $r = 2$
Penalty and r (p, r)	PHYLOGENETIC PENALTY MINIMIZATION	$\in$ XP, otherwise open
Distance (d)	NNI DISTANCE	$\in$ XP, otherwise open
	SPR DISTANCE	$\in$ XP, otherwise open
	TBR DISTANCE	$\in$ FPT
	CROSSING DISTANCE	$\in$ FPT
Number of trees (t)	CUP	$\in$ FPT
	MAXIMUM AGREEMENT SUBTREE	$\notin$ XP, unless P = NP
Maximum node degree (d)	MAXIMUM AGREEMENT SUBTREE	$\in$ FPT
Substring length and r (r, l)	SUBSTRING PARSIMONY	$\in$ FPT
	SUBSTRING PARSIMONY WITH LOSSES	$\in$ FPT
?-Entries in matrix (q)	INCOMPLETE PP HAPLOTYPING	$\in$ FPT for $r = 2$
?-Entries per column (c)	INCOMPLETE PP PATH HAPLOTYPING	$\in$ FPT for $r = 2$

The abbreviation TP stands or tree perfection. The '(trivially)' indicates that the problem can be solved in polynomial time, regardless of the parameter.

ACKNOWLEDGEMENTS

Jens Gramm was supported by a grant for the DFG project *Optimal solutions for hard problems in computational biology* (OPAL). Arfst Nickelsen was supported by a grant for the DFG project *Complexity of haplotyping problems.*

REFERENCES

[1] Darwin, C. (1859) *On the Origin of Species by Means of Natural Selection*. Murray, London 1859.

[2] Mateescu, A. and Salomaa, A. (1997) Formal languages: an introduction and a synopsis. In *Handbook of Formal Languages*, Vol. 1. Springer-Velag.

[3] Felsenstein, J. (2004) *Inferring Phylogenies*. Sinauer Associates.

[4] Gusfield, D. (1997) *Algorithm on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press.

[5] Page, R.D.M. and Holmes, E.C. (ed.) (1998) *Molecular Evolution—A Phylogenetic Approach.* Blackwell Science.

[6] Semple, C. and Steel, M. (2003) *Phylogenetics*. Oxford University Press.

[7] Smith, V.S. (2001) Avian louse phylogeny (phthiraptera: Ischnocera): a cladistic study based on morphology. *Zool. J. Linnean Soc.*, **132**, 81–144.

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

[9] Niedermeier, R. (2006) *Invitation to Fixed-Parameter Algorithms*. Oxford University Press.

[10] Feil, E.J. *et al*. (2001) Recombination within natural populations of pathogenic bacteria: short-term empirical estimates and long-term phylogenetic consequences. *Proc. Nat. Acad. Sci. USA*, **98**, 182–187.

[11] Philippe, H. and Douady, C.J. (2003) Horizontal gene transfer and phylogenetics. *Cur. Opin. Microbiol.*, **6**, 498–505.

[12] Grishin, N.V., Wolf, Y.I., Rogozin, I.B. and Koonin, E.V. (2002) Genome trees and the tree of life. *Trends Genet.*, **18**, 472–479.

[13] Bodlaender, H.L., Fellows, M.R. and Warnow, T.J. (1992) Two strikes against perfect phylogeny. In *Proc. of the 19th International Colloquium on Automata, Languages and Programming* (*ICALP*), Vol. 623 of Lecture Notes in Computer Science, pp. 273–283. Springer-Verlag.

[14] Steel, M. (1992) The complexity of reconstructing trees from qualitative characters and subtrees. *J. Classification*, **9**, 91–116.

[15] Gusfield, D. (2002) Haplotyping as perfect phylogeny: conceptual framework and efficient solutions. In *Proc. of the Sixth Annual International Conference on Computational Molecular Biology* (*RECOMB*), pp. 166–75. ACM Press.

[16] McMorris, F.R., Warnow, T.J. and Wimer, T. (1993) Triangulating vertex colored graphs. In *Proc. of the Fourth Symposium on Discrete Algorithms* (*SODA*), Philadelphia, PA, USA, pp. 120–127. SIAM Press.

[17] Agarwala, R. and Fernández-Baca, D. (1996) Simple algorithms for perfect phylogeny and triangulating colored graphs. *Int. J. Foundations Comput. Sci.*, **7**(1), 11–21.

[18] Bodlaender, H.L., Fellows, M.R., Hallett, M.T., Wareham, H.T. and Warnow, T.J. (2000) The hardness of perfect phylogeny, feasible register assignment and other problems on thin colored graphs. *Theoret. Comp. Sci.*, **244**(1–2), 167–188.

[19] Agarwala, R. and Fernández-Baca, D. (1994) A polynomial time algorithm for the perfect phylogeny problem when the number of character states is fixed. *SIAM J. Comput.*, **23**(6), 1216–1224.

[20] Kannan, S. and Warnow, T. (1997) A fast algorithm for the computation and enumeration of perfect phylogenies. *SIAM J. Comput.*, **26**, 1749–1763.

[21] Dress, A. and Steel, M. (1992) Convex tree realizations of partitions. *Appl. Math. Lett.*, **5**(3), 3–6.

[22] Kannan, S. and Warnow, T. (1994) Inferring evolutionary history from DNA sequences. *SIAM J. Comput.*, **23**, 713–737.

[23] Estabrook, G.F., Johnson, C.S., Jr and McMorris, F.R. (1975) An idealized concept of the true cladistic character. *Math. Biosci.*, **23**(5), 263–272.

[24] Meacham, C. A. (1983) Theoretical and computational considerations of the compatibility of qualitative taxonomic characters. *NATO ASI Series*, Springer, **G1** on Numerical Taxonomy.

[25] Gusfield, D. (1991) Efficient algorithms for inferring evolutionary trees. *Networks*, **21**, 19–28.

[26] Fernández-Baca, D. and Lagergren, J. (2003) A polynomial time algorithm for near-perfect phylogeny. *SIAM J.Comput.*, **32**, 1115–1127.

[27] Goldberg, L.A., Goldberg, P.W., Phillips, C., Sweedyk, Z. and Warnow, T. (1996) Minimizing phylogenetic number to find good evolutionary trees. *Discrete Appl. Math.*, **71**, 111–136.

[28] Moran, S. and Snir, S. (2005) Convex recolorings of phylogenetic trees: definitions, hardness results and algorithms. In *Proc. of the Ninth Workshop on Algorithms and Data Structures* (*WADS*)*,* Vol. 3608 of Lecture Notes in Computer Science, pp. 218–232. Springer-Verlag.

[29] Blelloch, G.E., Dhamdhere, K., Halperin, E., Ravi, R., Schwartz, R. and Sridhar, S. (2006) Fixed parameter tractability of binary near-perfect phylogenetic tree reconstruction. In *Proc. of the 33rd International Colloquium on Automata, Languages and Programming* (*ICALP*)*,* Vol. 4051 of Lecture Notes in Computer Science, Springer, pp. 667–678.

[30] Wernicke, S., Alber, J., Gramm, J., Guo, J. and Niedermeier, R. (2004) Avoiding forbidden submatrices by row deletions. In *Proc. of the 31st Annual Conf. on Current Trends in Theory and Practice of Informatics* (*SOFSEM*)*,* Vol. 2832 of Lecture Notes in Computer Science, pp. 349–360. Springer-Verlag.

[31] Fernau, H. (2004) A top-down approach to search trees: Improved algorithmics for 3-hitting set. Technical Report TR04-073, Electronic Colloquium of Computational Complexity (ECCC).**

[32] Niedermeier, R. and Rossmanith, P. (2003) An efficient fixed parameter algorithm for 3-hitting set. *J. Discrete Algorithms*, **1**, 89–102.

[33] Chen, J., Kanj, I.A. and Jia, W. (2001) Vertex cover: further observations and further improvements. *J. Algorithms*, **41**(2), 280–301.

[34] Das Gupta, B., He, X., Jiang, T., Li, M., Tromp, J., Wang, L. and Zhang, L. (1998) Computing distances between evolutionary trees. In Du, D. and Pardales, P. M. *Handbook of Combinatorial Optimization*, **Vol. 2**, pp. 35–76. Kluwer Academic Publishers.

[35] Lander, E.S. *et al*. (International Human Genome Sequencing Consortium). (2001) Initial sequencing and analysis of the human genome. *Nature*, **409**, 860–921.

[36] Salzberg, S.L., White, O., Peterson, J. and Eisen, J.A. (2001) Microbial genes in the human genome: lateral transfer or gene loss. *Science*, **292**, 1903–1906.

[37] Andersson, J.O., Doolittle, W.F. and Nesbø, C.L. (2001) Are there bugs in our genome? *Science*, **292**(5523), 1848–1850.

[38] Gogarten, J.P., Doolittle, W.F. and Lawrence, J.G. (2002) Prokaryotic evolution in light of gene transfer. *Mol. Biol. Evol.*, **19**, 2226–2238.

[39] Das Gupta, B., He, X., Jiang, T., Li, M., Tromp, J. and Zhang, L. (2000) On computing the nearest neighbor interchange distance. In *Discrete Mathematical Problems with Medical Applications,* Vol. 55 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science,* pp. 125–143. AMS Press.

[40] Robinson, D.F. (1971) Comparison of labeled trees with valency three. *J. Combinatorial Theory, Series B*, **11**, 105–119.

[41] Waterman, M.S. and Smith, T.F. (1978) On the similarity of dendrograms. *J. Theoret. Biol.*, **73**, 789–800.

[42] Culik, K., II and Wood, D. (1982) A note on some tree similarity measures. *Inf. Process. Lett.*, **15**, 39–42.

[43] Hein, J., Jiang, T., Wang, L. and Zhang, K. (1996) On the complexity of comparing evolutionary trees. *Discrete Appl. Math.*, **71**, 153–169.

[44] Allen, B. L. and Steel, M. (2001) Subtree transfer operations and their induced metrics on evolutionary trees. *Ann. Combinator.*, **5**, 1–13.

[45] Fernau, H., Kaufmann, M. and Poths, M. (2005) Comparing trees via crossing minimization. In *Proc. of the 25th Conf. on Foundations of Software Technology and Theoretical Computer Science* (*FSTTCS*)*,* Vol. 3821 of Lecture Notes in Computer Science, pp. 457–469. Springer-Verlag.

[46] Sanderson, M.J., Purvis, A. and Henze, C. (1998) Phylogenetic supertrees: assembling the tree of life. *Trends in Ecol. Evol.*, **13**, 105–109.

[47] Kluge, A.G. (1989) A concern for evidence and a phylogenetic hypothesis of relationships among *epicrates* (boidæ, serpents). *Syst. Zool.*, **38**, 7–25.

[48] Pisani, D., Yates, A.M., Langer, M.C. and Benton, M.J. (2006) A genus-level supertree of the Dinosauria. *Proc. Roy. Soc. London, Series B*, **269**, 915–921.

[49] Gordon, A.D. (1986) Consensus supertrees: the synthesis of rooted trees containing overlapping sets of labeled leaves. *J. Classification*, **3**, 31–39.

[50] Aho, A.V., Sagiv, Y., Szymansk, T.G. and Ullman, J.D. (1981) Inferring a tree from lowest common ancestors with an

application to the optimization of relational expressions. *SIAM J. Comput.*, **10**(3), 405–421.

[51] Baum, B.R. (1992) Combining trees as a way of combining data sets for phylogenetic inference, and the desirability of combining gene trees. *Taxon*, **41**, 3–10.

[52] Doyle, J. (1992) Gene trees and species trees: molecular systematics as one-character taxonomy. *Syst. Bot.*, **17**, 144–163.

[53] Ragan, M. (1992) Phylogenetic inference based on matrix representation of trees. *Mol. Phylogenet. Evol.*, **1**, 53–58.

[54] Chen, D., Diao, L., Eulenstein, O., Fernández-Baca, D. and Sanderson, M. J. (2003) Flipping: a supertree construction method. In *Bioconsensus,* Vol. 61 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Sciences*, pp. 135–160. AMS Press.

[55] Bininda-Emonds, O. (ed.) (2004) *Phylogenetic Supertrees*. Kluwer Academic Publishers.

[56] Pérez-Moreno, B.P., Sanz, J.L., Sudre, J. and Sigé, B. (1993) A theropod dinosaur from the lower cretaceous of southern France. *Revue de Pal'eobiologie*, **7**, 173–188.

[57] Holtz, T.R., Jr. (1994) The phylogenetic position of the Tyrannosauridae: implications for theropod systematics. *J. Paleontol.*, **68**, 1100–1117.

[58] Steel, M. August 2003. Personal communication. Open question posed at the Dagstuhl workshop 03311 on fixed parameter algorithms.

[59] Bryant, D. and Lagergren, J. (2006) Compatibility of unrooted trees is FPT. *Theoret. Comp. Sci.*, **351**, 296–302.

[60] Bodlaender, H.L. (1996) A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, **25**(6), 1305–1317.

[61] Courcelle, B. (1990) The monadic second-order logic of graphs I. Recognizable sets of finite graphs. *Inf. Comput.*, **85**(1), 12–75.

[62] Gramm, J. and Niedermeier, R. (2003) A fixed-parameter algorithm for Minimum quartet inconsistency. *J. Comput. Syst. Sci.*, **67**, 723–741.

[63] Amenta, N., Clarke, F. and St. John, K. (2003) A linear-time majority tree algorithm. In *Proc. of the Third Workshop on Algorithms in Bioinformatics* (*WABI*)*,* Vol. 2812 of Lecture Notes in Computer Science, pp. 216–227. Springer-Verlag.

[64] Amir, A. and Keselman, D. (1997) Maximum agreement subtree in a set of evolutionary trees: metrics and efficient algorithm. *SIAM J. Comput.*, **26**(6), 1656–1669.

[65] Farach, M., Przytycka, T.M. and Thorup, M. (1995) On the agreement of many trees. *Inf. Process. Lett.*, **55**(6), 297–301.

[66] Berry, V. and Nicolas, F. (2006) Improved parametrized complexity of maximum agreement subtree and maximum compatible tree problems. *IEEE/ACM Trans. Comput. Biol. Bioinformat.*, **3**(3), 289–302.

[67] Tagle, D.A., Koop, B.F., Goodman, M., Slighton, J.L., Hess, D. and Jones, R. (1988) Embryonic e and g globin genes of a prosimian primate (*Galago crassicaudatus*) nucleotide and amino acid sequences, developmental regulation and phylogenetic footprints. *J. Mol. Biol.*, **203**, 439–455.

[68] Blanchette, M., Schwikowski, B. and Tompa, M. (2002) Algorithms for phylogenetic footprinting. *J. Comput. Biol.*, **9**(2), 211–223.

[69] National center of biotechnology information of the united states. http://www.ncbi.nlm.nih.gov/. Accessed on March 21, 2006.

[70] Ding, Z., Filkov, V. and Gusfield, D. (2005) A lineartime algorithm for the perfect phylogeny haplotyping (PPH) problem. In *Proc. of the Ninth Annual Int. Conf. on Research in Computational Molecular Biology* (*RECOMB*)*,* Vol. 3500 of Lecture Notes in Computer Science, pp. 585–600. Springer-Verlag.

[71] Kimmel, G. and Shamir, R. (2005) The incomplete perfect phylogeny haplotype problem. *J. Bioinformat. Comput. Biol.*, **3**, 359–384.

[72] Gramm, J., Nierhoff, T., Tantau, T. and Sharan, R. (2007) Haplotyping with missing data via perfect path phylogenies. *Discrete Appl. Math.*, **155**, 788–805.

[73] Zhang, J., Rowe, W.L., Clark, A.G. and Buetow, K.H. (2003) Genome wide distribution of high frequency, completely mismatching SNP haplotype pairs observed to be common across human populations. *Am. J. Human Genet.*, **73**, 1073–1081.

[74] Durbin, R., Eddy, S.S., Krogh, A. and Mitchison, G. (1998) *Biological Sequence Analysis—Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press.

[75] Chor, B. and Tuller, T. (2005) Maximum likelihood of evolutionary trees is hard. In *Proc. of the Ninth Annual Int. Conf. on Research in Computational Molecular Biology* (*RECOMB*)*,* Vol. 3500 of Lecture Notes in Computer Science, pp. 296–310.

[76] Chor, B. and Tuller, T. (2005) Maximum likelihood of evolutionary trees: hardness and approximation. *Bioinformatics*, **21**, 97–106.

[77] Wang, L., Zhang, K. and Zhang, L. (2001) Perfect phylogenetic networks with recombination. *J. Comput. Biol.*, **8**: 69–78.

[78] Gusfield, D., Eddhu, S. and Langley, C. (2004) Optimal, efficient reconstruction of phylogenetic networks with constrained recombination. *J. Comp. Syst. Sci.*, **2**, 173–213.

[79] Gusfield, D. (2005) Optimal, efficient reconstruction of root-unknown phylogenetic networks with constrained and structured recombination. *J. Comp. Syst. Sci.*, **70**, 381–398.

[80] Nakhleh, L., Jin, G., Zhao, F. and Mellor-Crummey, J. (2005) Reconstructing phylogenetic networks using maximum parsimony. In *Proc. of the Fourth Fourth International IEEE Computer Society Computational Systems Bioinformatics Conference*, pp. 93–102. IEEE Computer Society Press.