Information Sciences 197 (2012) 77–90

Contents lists available at SciVerse ScienceDirect

Information Sciences

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

Faster computation of the Robinson–Foulds distance between phylogenetic networks ☆

Tetsuo Asano [a], Jesper Jansson [b], Kunihiko Sadakane [c], Ryuhei Uehara [a], Gabriel Valiente [d,*]

[a] School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa 923-1292, Japan
[b] Ochanomizu University, 2-1-1 Otsuka, Bunkyo-ku, Tokyo 112-8610, Japan
[c] National Institute of Informatics, Hitotsubashi 2-1-2, Chiyoda-ku, Tokyo 101-8430, Japan
[d] Algorithms, Bioinformatics, Complexity and Formal Methods Research Group, Technical University of Catalonia, E-08034 Barcelona, Spain

ARTICLE INFO

Article history:
Received 11 August 2010
Received in revised form 15 November 2011
Accepted 29 January 2012
Available online 21 February 2012

Keywords:
Algorithm
Phylogenetic network
Robinson–Foulds distance
Cluster representation
Minimum spread
Leaf-outerplanar phylogenetic network

ABSTRACT

The Robinson–Foulds distance, a widely used metric for comparing phylogenetic trees, has recently been generalized to phylogenetic networks. Given two phylogenetic networks N_1, N_2 with n leaf labels and at most m nodes and e edges each, the Robinson–Foulds distance measures the number of clusters of descendant leaves not shared by N_1 and N_2. The fastest known algorithm for computing the Robinson–Foulds distance between N_1 and N_2 runs in $O(me)$ time. In this paper, we improve the time complexity to $O(ne/\log n)$ for general phylogenetic networks and $O(nm/\log n)$ for general phylogenetic networks with bounded degree (assuming the word RAM model with a word length of $\lceil \log n \rceil$ bits), and to optimal $O(m)$ time for leaf-outerplanar networks as well as optimal $O(n)$ time for level-1 phylogenetic networks (that is, galled-trees). We also introduce the natural concept of the minimum spread of a phylogenetic network and show how the running time of our new algorithm depends on this parameter. As an example, we prove that the minimum spread of a level-k network is at most $k + 1$, which implies that for one level-1 and one level-k phylogenetic network, our algorithm runs in $O((k + 1)e)$ time.

1. Introduction

The Robinson–Foulds distance, introduced in [24], has been the most widely used metric for almost three decades for comparing phylogenetic trees. It measures the dissimilarity between any two given phylogenetic trees by counting the number of so-called *clusters* that are not shared by the two trees, where each set of leaf labels in the subtree rooted at one node constitutes one cluster. In addition to satisfying a number of nice mathematical properties [10,24], the Robinson–Foulds distance between trees can be computed very quickly [10,22]. However, recent evidence indicates that the evolutionary history of life (in particular, the evolutionary history of bacteria) cannot be properly represented as a phylogenetic tree [3,11,25], and *phylogenetic networks* have emerged as the representation of choice for incorporating reticulate evolutionary events such as recombination, hybridization, or lateral gene transfer in an evolutionary history [23].

Essentially, a phylogenetic network is a generalization of a phylogenetic tree in which certain internal nodes are allowed to have more than a single parent. A straightforward extension of the definition of the Robinson–Foulds distance to

☆ A preliminary version of this article appeared in *Proceedings of the 21st Annual Symposium on Combinatorial Pattern Matching*, volume 6129 of *Lecture Notes in Computer Science*, pp. 190–201, Springer-Verlag, 2010.

* Corresponding author.
E-mail addresses: t-asano@jaist.ac.jp (T. Asano), jesper.jansson@ocha.ac.jp (J. Jansson), sada@nii.ac.jp (K. Sadakane), uehara@jaist.ac.jp (R. Uehara), valiente@lsi.upc.edu (G. Valiente).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2012.01.038

phylogenetic networks, along with an algorithm to compute it, was given in [6]. In this paper, we develop new techniques for computing the Robinson–Foulds distance between two input phylogenetic networks more efficiently.

1.1. Basic definitions

A *phylogenetic network* is a connected, rooted, simple, directed acyclic graph with two types of nodes: *tree nodes* (having at most one parent) corresponding to point mutation events, and *hybrid nodes* (with more than one parent) corresponding to reticulate evolutionary events. Without loss of generality, no node has both indegree 1 and outdegree 1. As in the case of phylogenetic trees, nodes with outdegree 0 are called *leaves*, and we require that the leaves in a phylogenetic network are distinctly labeled.

Let $N = (V, E)$ be a phylogenetic network. For any nodes u, $v \in V$, v is called a *descendant of* u if v is reachable from u in N. For convenience, every node is considered to be a descendant of itself. For any $v \in V$, define *the cluster of* v (denoted by $C(v)$) as the set of all leaves that are descendants of v, and let *the cluster collection of* N be the multiset $C(N) = \{C(v): v \in V\}$. If v is a descendant of u with $v \neq u$ then v is a *proper descendant of* u.

Next, the *Robinson–Foulds distance* $d_{RF}(N_1, N_2)$ between two phylogenetic networks N_1, N_2 is defined as the cardinality of the symmetric difference between their two cluster collections, divided by two:

$$d_{RF}(N_1, N_2) = \frac{|C(N_1) \setminus C(N_2)| + |C(N_2) \setminus C(N_1)|}{2}$$

The value of $d_{RF}(N_1, N_2)$ depends on the number of clusters present in one of the two networks but not in the other. Thus, d_{RF} measures the *dissimilarity* between N_1 and N_2. Note that the above definition generalizes the Robinson–Foulds distance between rooted phylogenetic trees from [24].

Example 1. The phylogenetic network in Fig. 1 has the following cluster collection, where trivial clusters (for the leaves) are omitted:

$$C(v_1) = \{v_2, v_{10}, v_{12}, v_{14}, v_{18}, v_{20}, v_{22}, v_{23}, v_{25}, v_{26}, v_{27}, v_{28}, v_{33}, v_{34}, v_{36}, v_{37}, v_{42}, v_{43}, v_{45}, v_{46}, v_{47}\}$$
$$C(v_3) = \{v_{10}, v_{12}, v_{14}, v_{18}, v_{20}, v_{22}, v_{23}, v_{25}, v_{26}, v_{27}, v_{28}, v_{33}, v_{34}, v_{36}, v_{37}, v_{42}, v_{43}, v_{45}, v_{46}, v_{47}\}$$
$$C(v_4) = \{v_{10}, v_{12}, v_{20}, v_{22}, v_{23}, v_{27}, v_{28}, v_{33}, v_{34}, v_{36}, v_{37}, v_{42}, v_{43}, v_{45}, v_{46}, v_{47}\}$$
$$C(v_5) = \{v_{14}, v_{18}, v_{25}, v_{26}, v_{33}, v_{34}, v_{36}, v_{37}, v_{42}, v_{43}, v_{45}, v_{46}, v_{47}\}$$
$$C(v_6) = \{v_{10}, v_{20}, v_{27}, v_{28}, v_{34}, v_{42}, v_{43}, v_{45}, v_{46}, v_{47}\}$$
$$C(v_7) = \{v_{12}, v_{22}, v_{23}, v_{33}, v_{34}, v_{36}, v_{37}, v_{42}, v_{43}, v_{45}, v_{46}, v_{47}\}$$
$$C(v_8) = \{v_{14}, v_{18}, v_{25}, v_{26}\}$$
$$C(v_9) = \{v_{20}, v_{27}, v_{28}, v_{34}, v_{42}, v_{43}, v_{45}, v_{46}, v_{47}\}$$
$$C(v_{11}) = \{v_{22}, v_{23}, v_{33}, v_{34}, v_{36}, v_{37}, v_{42}, v_{43}, v_{45}, v_{46}, v_{47}\}$$
$$C(v_{13}) = \{v_{18}, v_{25}, v_{26}\}$$
$$C(v_{15}) = \{v_{20}, v_{27}, v_{28}\}$$
$$C(v_{16}) = \{v_{22}, v_{23}\}$$
$$C(v_{17}) = C(v_{24}) = \{v_{33}, v_{34}, v_{36}, v_{37}, v_{42}, v_{43}, v_{45}, v_{46}, v_{47}\}$$
$$C(v_{19}) = \{v_{25}, v_{26}\}$$
$$C(v_{21}) = \{v_{27}, v_{28}\}$$
$$C(v_{29}) = C(v_{31}) = \{v_{34}, v_{42}, v_{43}, v_{45}, v_{46}, v_{47}\}$$
$$C(v_{30}) = \{v_{33}, v_{36}, v_{37}\}$$
$$C(v_{32}) = \{v_{36}, v_{37}\}$$
$$C(v_{35}) = C(v_{38}) = \{v_{42}, v_{43}, v_{45}, v_{46}, v_{47}\}$$
$$C(v_{39}) = \{v_{42}, v_{45}\}$$
$$C(v_{40}) = \{v_{43}, v_{46}, v_{47}\}$$
$$C(v_{41}) = \{v_{45}\}$$
$$C(v_{44}) = \{v_{46}, v_{47}\}$$

1.2. Previous Work

According to the definitions above, the Robinson–Foulds distance can be computed easily if the cluster collections of the two networks are known; see Section 5 for details. Fast methods exist for computing the cluster collection of a phylogenetic *tree*; for example, a classic algorithm by Day [10] computes $C(T)$ for any tree T containing n leaves in $O(n)$ time using $O(n\log n)$ bits space. On the other hand, the only known method for computing the cluster collection of a phylogenetic *network* with n leaves, m nodes, and e edges is a simple algorithm by Cardona et al. [6] that performs a breadth-first search from each node v to find its cluster $C(v)$, using a total of $O(me)$ time and $O(nm)$ bits space.

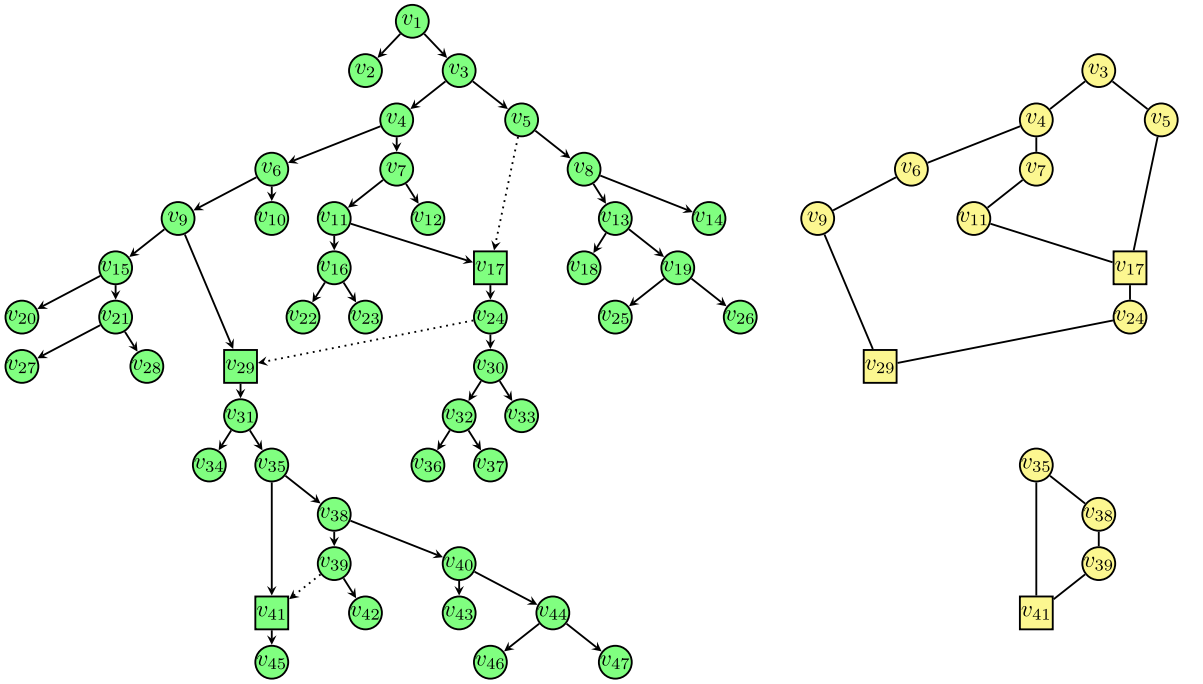

Fig. 1. (Left) An example of a phylogenetic network, adapted from [32]. This is the smallest level-2 phylogenetic network consistent with 1,330 rooted triplets constructed from 21 different isolates of the yeast *Cryptococcus gattii*. (Right) Non-trivial biconnected components.

Two generalizations of the Robinson–Foulds distance between phylogenetic networks are known. In the *tripartitions distance*, introduced by Moret et al. [19] and subsequently improved by Nguyen et al. [20], the descendant leaves of each node in a phylogenetic network are further divided into *strict* and *non-strict* descendants. In the path-multiplicity distance (μ-distance), introduced by Cardona et al. [7] and also studied in [5,6], the descendant leaves of each node are instead represented by the number of paths from the node to each of the leaves. As a result, both the tripartitions distance and the μ-distance allow for finer distinctions than the Robinson–Foulds distance [6].

1.3. New results

We present an improved algorithm for computing the Robinson–Foulds distance between two input phylogenetic networks. For general phylogenetic networks, we first reduce the time complexity of the algorithm of Cardona et al. [6] by following a bottom-up approach similar in spirit to the algorithm proposed in [7] for computing the path-multiplicity representation of a phylogenetic network. Next, by using a compressed representation of the characteristic vectors, we obtain an algorithm for computing the Robinson–Foulds distance between two phylogenetic networks with at most n leaves, m nodes, and e edges each, in $O(ne/\log n)$ time and $O(nm/\log n)$ words, assuming the word RAM model[1] with a word length of $\lceil \log n \rceil$ bits; see [15]. For phylogenetic networks of bounded degree, this reduces to $O(nm/\log n)$ time.

In the case of level-k phylogenetic networks (see Section 4.2 for a definition), we further improve the time complexity by using a representation of each cluster of descendant leaves as an interval of consecutive integers, which allows us to compute the Robinson–Foulds distance between a level-1 and a level-k phylogenetic network in $O((k+1)e)$ time. For this purpose, we introduce a new parameter that we call the *minimum spread* of a phylogenetic network, and prove that every level-k network has minimum spread at most $k+1$. For special cases of bounded-level phylogenetic networks such as leaf-outerplanar phylogenetic networks and galled-trees [13,14], we show that the minimum spread is 1, which means that our algorithm can be implemented to run in optimal $O(e)$ time.

1.4. Organization of the paper

The first part of the paper (Sections 2–4) describes different ways to represent the cluster collection $C(N)$ of a given phylogenetic network N. To be more precise, Section 2 explains the *naive cluster representation* and how to speed up the algorithm of Cardona et al. [6]. Section 3 studies the *cluster representation by characteristic vectors*. Section 4 considers the *cluster representation by interval lists* and defines the *minimum spread* of a phylogenetic network. Sections 4.1 and 4.2

[1] In this paper, we adopt the standard word RAM model; in this model, the number of operations is proportional to the running time. Hence, arguments are simplified in this model.

investigate efficient ways to represent $\mathcal{C}(N)$ when N is a *leaf-outerplanar* or a *level-k* phylogenetic network by deriving upper bounds on the minimum spread for these types of networks.

Next, Section 5 presents an algorithm (Algorithm 3) for computing the Robinson–Foulds distance that takes advantage of our compact cluster representations. In Section 6, empirical results are shown that confirm the theoretical analysis of the algorithm under the different cluster representations. Finally, some open problems are discussed in Section 7.

2. Naive cluster representation

Let $N = (V, E)$ be a given phylogenetic network with n leaves, m nodes, and e edges. Recall that for any $v \in V$, the cluster of v (denoted by $\mathcal{C}(v)$) is the set of all leaf descendants of v, and the cluster collection of N (denoted by $\mathcal{C}(N)$) is the multiset of all clusters in N.

The *naive cluster representation of N* stores the multiset $\mathcal{C}(N)$ explicitly, listing every element in every set $\mathcal{C}(v)$ belonging to $\mathcal{C}(N)$. Cardona et al. [6] described an algorithm for constructing the naive cluster representation of N by doing a breadth-first search from each node in turn to find its set of descendant leaves. Since each of the m breadth-first searches takes $O(e)$ time, the method runs in $O(me)$ time and $O(nm)$ space.

We first observe that the time complexity can be reduced by replacing the m top-down searches by n bottom-up searches. This leads to a significant improvement in many cases because the value of m can be arbitrarily large for a phylogenetic network with n leaves.[2] The following lemma is the basis of such an improvement.

Lemma 1. *Let $v \in V$ be a node of a phylogenetic network $N = (V, E)$. Then, $\mathcal{C}(v) = \{v\}$ if v is a leaf, and $\mathcal{C}(v) = \mathcal{C}(v_1) \cup \cdots \cup \mathcal{C}(v_k)$ if v is an internal node with children $\{v_1, \ldots, v_k\}$.*

Algorithm 1. Compute the naive cluster representation $\mathcal{C}$ of a phylogenetic network N

```
procedure naive_cluster_representation (N, C)
  for each node v of N do
    if v is a leaf then
        C(v) ← {label(v)}
        enqueue(Q, v)
    else
        C(v) ← ∅
  while Q is not empty do
    v ← dequeue(Q)
    mark node v as visited
    for each parent u of node v do
        C(u) ← C(u) ∪ C(v)
        if all children of u are visited then
          enqueue(Q, u)
```

Proof. Trivially, the only descendant of a leaf in N is the leaf itself. For any internal node v and any leaf ℓ in N, if ℓ is a descendant of some child of v then ℓ must be a descendant of v as well; conversely, if ℓ is a descendant of v then there exists a path in N from v to ℓ, and such a path must pass through at least one child of v. □

Lemma 1 suggests a standard bottom-up traversal for computing the naive cluster representation of N in polynomial time. The pseudocode is listed in Algorithm 1. The algorithm computes all clusters in N during a single bottom-up traversal of N with the help of a queue Q that stores nodes temporarily. Initially, Q contains all leaves of N. The main loop dequeues the node v currently first in Q and copies the elements in $\mathcal{C}(v)$ to the (initially empty) set $\mathcal{C}(u)$ for every parent u of v. The last **if**–**then**-statement guarantees that for any internal node u, the leaf descendants of all children of u have always been included in $\mathcal{C}(u)$ before u is enqueued, so that the correct value of $\mathcal{C}(u)$ is passed on upwards towards the root.

Theorem 1. *Let N be a phylogenetic network with n leaves, m nodes, and e edges. The naive cluster representation of N can be computed in $O(ne)$ time using $O(nm)$ bits.*

Proof. Every node is enqueued and dequeued only once, and every parent of each dequeued node v is visited only once from v. The union operation of two subsets of an n-element set, which takes $O(n)$ time, is performed $O(e)$ times. □

[2] Even in the special case of a tree-child time-consistent phylogenetic network (see Section 7 for the definition), although it holds that $m \leqslant (n+4)(n-1)/2$, this bound is tight [6, Proposition 1].

3. Cluster representation by characteristic vectors

The cluster collection of a network can be represented by bit vectors called *characteristic vectors*, formally defined as follows.

Let $N = (V, E)$ be a phylogenetic network with n leaves. A *leaf numbering function* is a bijection from the set of leaf labels in N to the set $\{1, 2, \ldots, n\}$. For any leaf numbering function f and node $v \in V$, the *characteristic vector* for v under f, denoted by $C_f[v]$, is a bit vector of length n such that for any $i \in \{1, 2, \ldots, n\}$, the ith bit equals 1 if and only if the leaf labeled by $f^{-1}(i)$ is a descendant of v in N. Observe that $C_f[r] = 111 \ldots 1$ for the root r of N, and $C_f[\ell]$ contains exactly one 1 for any leaf ℓ of N.

Example 2. Consider the phylogenetic network in Fig. 1. Number the leaves according to the circular ordering v_2, v_{20}, v_{27}, v_{28}, v_{34}, v_{45}, v_{42}, v_{43}, v_{46}, v_{47}, v_{10}, v_{22}, v_{23}, v_{36}, v_{37}, v_{33}, v_{12}, v_{18}, v_{25}, v_{26}, v_{14} along the outer face. This corresponds to a depth-first search of the directed spanning tree obtained by removing one incoming edge (shown dotted in Fig. 1) for each node of indegree 2 in the network, and it yields the characteristic vectors listed in Table 1.

Obviously, given any leaf numbering function f, the cluster representation of N by characteristic vectors under f can be stored explicitly using a total of $m \cdot n$ bits. Furthermore, it can be constructed in $O(ne)$ time by an algorithm analogous to Algorithm 1, implementing the set union operation for two clusters by taking the bitwise OR of their two bit vectors. Our next goal is to find more compact ways of storing the characteristic vectors as well as faster ways of computing them. We first consider arbitrary leaf numbering functions for general phylogenetic networks in Section 3.1 and then focus on some important special classes of phylogenetic networks in Section 4.

3.1. Cluster representation by compressed characteristic vectors

This subsection shows that if one assumes the word RAM model with a word length of $\omega = \lceil \log n \rceil$ bits [15] then the time complexity of Algorithm 1 can be improved by a log n-factor, giving the *cluster representation of N by compressed characteristic vectors*.

The modified version of Algorithm 1 is presented in Algorithm 2. The overall structure of the algorithm is the same as before. The new idea is to fix an arbitrary leaf numbering function f for the given phylogenetic network N, employ a characteristic vector of length n to encode each cluster under f, and exploit the word-level parallelism of bitwise OR operations in the word RAM model to compute all clusters quickly.

We now describe the details of Algorithm 2. In the pseudocode, $x|y$ denotes the bitwise OR of two words x and y, and $x \ll t$ denotes the bitwise shift of a word x to the left by t positions. Define $k = \lceil \frac{2n}{\omega} \rceil$. Divide each characteristic vector $C_f[v]$ into k parts, each of size at most $\lceil \omega/2 \rceil$ bits, and store the parts in the first $\lceil \omega/2 \rceil$ bits of k words $W_1[v], \ldots, W_k[v]$. Then, every set union operation on two clusters may be implemented by k bitwise OR operations on words. For added efficiency, also apply preprocessing to compute and store the outcome of the bitwise OR operation between every possible pair of bit vectors of

Table 1
Characteristic vector representation of the clusters for the phylogenetic network in Fig. 1.

Node	Characteristic vector of the cluster																				
	v_2	v_{20}	v_{27}	v_{28}	v_{34}	v_{45}	v_{42}	v_{43}	v_{46}	v_{47}	v_{10}	v_{22}	v_{23}	v_{36}	v_{37}	v_{33}	v_{12}	v_{18}	v_{25}	v_{26}	v_{14}
v_{21}	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v_{15}	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v_{41}	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v_{39}	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v_{44}	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
v_{40}	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
v_{38}	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
v_{35}	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
v_{31}	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
v_{29}	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
v_9	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
v_6	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
v_{16}	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
v_{32}	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
v_{30}	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
v_{24}	0	0	0	0	1	1	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0
v_{17}	0	0	0	0	1	1	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0
v_{11}	0	0	0	0	1	1	1	1	1	1	0	1	1	1	1	1	0	0	0	0	0
v_7	0	0	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1	0	0	0	0
v_4	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
v_{19}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
v_{13}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
v_8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
v_5	0	0	0	0	1	1	1	1	1	1	0	0	0	1	1	1	0	1	1	1	1
v_3	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
v_1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 2
Compressed characteristic vectors of the phylogenetic network in Fig. 1.

Node	v_2	v_{20}	v_{27}	v_{28}	v_{34}	v_{45}	v_{42}	v_{43}	v_{46}	v_{47}	v_{10}	v_{22}	v_{23}	v_{36}	v_{37}	v_{33}	v_{12}	v_{18}	v_{25}	v_{26}	v_{14}
x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
$\left\lfloor \frac{2(x-1)}{\omega} \right\rfloor + 1$	1	1	1	2	2	3	3	3	4	4	5	5	5	6	6	7	7	7	8	8	9

length $\lceil \omega/2 \rceil$ in a table named *OR*. (The reason why we do not utilize all ω bits of the words for storing characteristic vectors is to prevent the number of entries in the resulting *OR*-table from becoming too large.)

In the initialization stage, immediately before enqueueing a leaf v, exactly one bit in the k words $W_1[v], \ldots, W_k[v]$ (the bit that represents v) will be set to 1 by bit shifting. Which word $W_i[v]$ to apply the bit shift to is determined in accordance with the formula in Lemma 2 below. After Algorithm 2 is finished, the characteristic vector $C_f[v]$ for each node v in N is stored in the k words $W_1[v], \ldots, W_k[v]$.

Lemma 2. *Let C be a bit vector of length n that has been divided into $k = \lceil \frac{2n}{\omega} \rceil$ parts $W_1, W_2, \ldots, W_k$ in such a way that the sum of the lengths of any two consecutive parts W_j and W_{j+1} equals ω and for each $j \in \{1, 2, \ldots, \lfloor k/2 \rfloor\}$, the length of W_{2j} is less than or equal to the length of W_{2j-1}. Then, for $x \in \{1, 2, \ldots, n\}$, position x of C belongs to the part W_i, where $i = \left\lfloor \frac{(x-1)\cdot 2}{\omega} \right\rfloor + 1$.*

Example 3. To obtain the compressed characteristic vectors of the phylogenetic network in Fig. 1 under the leaf numbering function f given by the circular ordering of the leaves from Table 1, first divide each characteristic vector of length $n = 21$ into $k = \lceil 2n/\omega \rceil = 9$ parts according to Lemma 2, where $\omega = \lceil \log n \rceil = 5$; see Table 2. Then, for each node, 9 words are used to store the 9 parts of its characteristic vector. For example, the characteristic vector of node v_7 is 000 01 111 11 011 11 110 00 0, and its compressed characteristic vector is then 017333600. Consider also leaf v_{22}. It is the 12th leaf in the circular ordering (that is, $f(v_{22}) = 12$). According to Lemma 2, the bit representing v_{22} will be packed into the same word as the bits that represent leaves v_{10} and v_{23}.

This yields:

Theorem 2. *Let N be a phylogenetic network with n leaves, m nodes, and e edges. Assuming the word RAM model with a word length of $\omega = \lceil \log n \rceil$ bits, the cluster representation of N by compressed characteristic vectors can be computed in $O(ne/\log n)$ time using $O(nm/\log n)$ words.*

Proof. The *OR*-table may be precomputed in $O(n)$ time and stored in $O(n)$ words since it contains $2^{\lceil \omega/2 \rceil} \cdot 2^{\lceil \omega/2 \rceil} = O(\sqrt{n} \cdot \sqrt{n}) = O(n)$ entries. After the preprocessing, every node is enqueued and dequeued only once, and every parent of each dequeued node v is visited only once from v, as in the proof of Theorem 1. Each one of the $O(e)$ set union operations is implemented by $k = \lceil \frac{2n}{\omega} \rceil = O(n/\log n)$ bitwise OR operations, each of which takes $O(1)$ time to perform by a look-up in the *OR*-table, and therefore takes $O(n/\log n)$ time. The total running time is $O(n + ek) = O(ne/\log n)$. The resulting cluster representation is stored as a compact table with m rows and $O(n/\log n)$ columns using $O(nm/\log n)$ words. □

Notice that the OR-table is precomputed only up to the maximum number of leaves that will fit in a given

Algorithm 2. Compute the cluster representation C by compressed characteristic vectors of a phylogenetic network N

```
procedure compressed_cluster_representation (N, C)
  n ← number of leaves of N
  ω ← ⌈log n⌉
  k ← ⌈2n/ω⌉
  for x ← 0, ..., ⌈ω/2⌉ − 1 do
    for y ← 0, ..., ⌈ω/2⌉ − 1 do
        OR[x, y] ← x|y
  for each node v of N do
    W_1[v], ..., W_k[v] ← 0
    if v is a leaf then
        i ← ⌊((f(v)−1)·2)/ω⌋ + 1
        W_i[v] ← 1 ≪ ⌈(ω·i)/2⌉ − f(v)
        enqueue(Q, v)
  while Q is not empty do
```

```
    v ← dequeue(Q)
    mark node v as visited
    for each parent u of node v do
        for i ← 1, ..., k do
            W_i[u] ← OR[W_i[u], W_i[v]]
        if all children of u are visited then
            enqueue(Q, u)
for each node v of N do
    C[v] ← W_1[v], ..., W_k[v]
```

word. For further details about word-level parallelism and the word RAM model, see [15].

4. Cluster representation by interval lists

Given a phylogenetic network $N = (V, E)$ with n leaves, m nodes, and e edges, and a fixed leaf numbering function f, the cluster of any $v \in V$ can be expressed as the union of a number of intervals on $\{1, 2, \ldots, n\}$. Storing the starting and ending points of these intervals in sorted order for every node in N yields the *cluster representation of N by interval lists*. Below, we consider restricted types of phylogenetic networks that admit "good" leaf numbering functions, in the sense that any resulting cluster contains a small number of intervals. Before investigating suitable leaf numbering functions, we first introduce a few additional definitions.

A maximal consecutive sequence of 1's in a bit vector is called an *interval*. For a given leaf numbering function f and node $v \in V$, let $I_f(v)$ denote the number of intervals in $C_f[v]$ and let the *spread of f* be $I_f = \max_{v \in V} I_f(v)$. The *minimum spread of N* is the minimum value of I_f, taken over all possible leaf numbering functions f.

Intuitively, if the spread of f is small then using interval lists provides an efficient representation of the cluster collection of N. More precisely:

Lemma 3. *Given any leaf numbering function f, the total space needed to store all characteristic vectors under f using the interval list representation is $O(I_f m \log n)$ bits.*

Proof. The starting and ending positions of any interval can be stored in $\lceil 2 \log n \rceil$ bits. There are m nodes and each one has at most I_f intervals, so the total space needed is $O(I_f m \log n)$ bits. □

Lemma 4. *Given any leaf numbering function f, the interval lists for all clusters in N can be computed in $O(I_f \cdot e)$ time.*

Proof. Apply the bottom-up technique from Algorithm 1. To implement the set union operation $C(u) \cup C(v)$, use $O(I_f)$ time to scan the two sorted interval lists for $C_f[u]$ and $C_f[v]$ while merging any intervals that overlap or are immediate neighbors. □

Next, we derive upper bounds on the minimum spread of two special classes of phylogenetic networks called *leaf-outerplanar phylogenetic networks* and *level-k phylogenetic networks*, where k is a non-negative integer. From here on, we only consider phylogenetic networks in which each node has either at most one parent (tree node) or exactly two parents (hybrid node).

4.1. Leaf-outerplanar phylogenetic networks

For any phylogenetic network N, let $\mathcal{U}(N)$ be the undirected graph obtained by replacing every directed edge in N by an undirected edge. A phylogenetic network N is called *leaf-outerplanar* if $\mathcal{U}(N)$ admits a non-crossing layout in the plane with the root and all leaves lying on the outer face. See Fig. 2 for an illustration.

Leaf-outerplanar phylogenetic networks form an important class because they are output by certain popular phylogenetic network construction methods such as Neighbor-Net [4] and QNet [12], and are often used to visualize a set of conflicting phylogenetic trees in a single diagram.

Euler's formula gives the relation $e = O(m)$ for any leaf-outerplanar phylogenetic network. We immediately have:

Lemma 5. *If N is a leaf-outerplanar phylogenetic network then a leaf numbering function f with $I_f = 1$ exists and can be computed in $O(m)$ time.*

Proof. Join the root r and all leaves in N to a new vertex, and call the modified graph N'. Run the linear-time planar embedding algorithm of Chiba et al. [8,21] to construct some planar embedding of N'. This gives a leaf-outerplanar embedding for the original N where r and the leaves of N lie on the outer face. Starting at r, traverse the boundary of the outer face in one direction while letting f assign $1, 2, \ldots, n$ to the leaves of N consecutively in the order that they are visited.

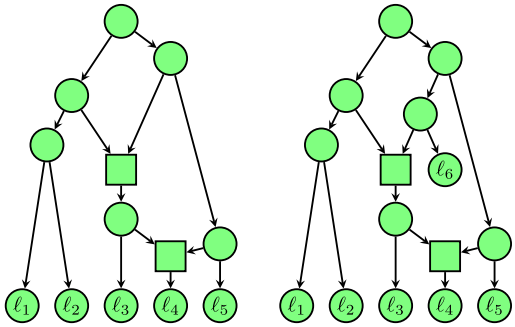

Fig. 2. Two phylogenetic networks. The one on the left is leaf-outerplanar but the one on the right is not.

Consider any node v in N. There are two cases:

- v is a leaf: Trivially, $C_f[v]$ has a single interval.
- v is an internal node: Suppose u, w are children of v such that $C(u) = \{g, \ldots, h\}$, $C(w) = \{k, \ldots, \ell\}$, and $i, \ldots, j \notin C(v)$, but $f(g) \leqslant f(h) < f(i) \leqslant f(j) < f(k) \leqslant f(\ell)$. A path from the root to a leaf in $\{i, \ldots, j\}$ may not pass through v; hence it crosses either the path from v to h or the path from v to k, as shown in Fig. 3. Contradiction.

Therefore, $C_f[v]$ has a single interval for every node v in N, and so $I_f = 1$.
The time complexity to compute f is $O(e) = O(m)$. □

Lemma 5 together with Lemmas 3 and 4 yields:

Theorem 3. *If N is a leaf-outerplanar phylogenetic network with n leaves, m nodes, and e edges, then the cluster representation of N by interval lists can be computed in $O(m)$ time using $O(m \log n)$ bits.*

4.2. Level-k phylogenetic networks

As in the previous subsection, let $\mathcal{U}(N)$ for any phylogenetic network N denote the undirected graph obtained by replacing every directed edge in N by an undirected edge. A *biconnected component* of an undirected graph is a connected subgraph that remains connected after removing any single node and all edges incident to it. For any non-negative integer k, N is a *level-k phylogenetic network* if, for every biconnected component B in $\mathcal{U}(N)$, the subgraph of N induced by the set of nodes in B contains at most k hybrid nodes. For example, the network in Fig. 1 is a level-2 network: the biconnected component shown at the bottom right has one hybrid node, and the one shown at the top right has two hybrid nodes.

Choy et al. [9] introduced the level of a phylogenetic network as a parameter that indicates how tree-like the given network is; the class of level-0 networks equals the class of phylogenetic trees, the class of level-1 networks corresponds to the class of networks in which all underlying cycles are disjoint,[3] the class of level-2 networks contains even more complex networks, etc. It is a useful parameter because several computational problems that are NP-hard for general phylogenetic networks can be solved in polynomial time when the level is bounded [9,16,32,33], and moreover, it is easy to compute the level of any given network.[4]

Lemma 6. *If N is a level-k phylogenetic network then a leaf numbering function f with $I_f \leqslant k + 1$ exists and can be computed in $O(e)$ time.*

Proof. Fix any directed spanning tree T of N. Let f be the leaf numbering function obtained by doing a depth-first search of T starting at the root and assigning the numbers $1, 2, \ldots, n$ to the leaves in the order that they are first visited. Clearly, this takes $O(e)$ time.

We now prove that f has spread $k + 1$. For every node v in N, define $L(T[v])$ as the set of all leaves in the subtree of T rooted at v. The key observation is that the leaves in $L(T[v])$ must be visited consecutively by any depth-first search of T, and thus *form a single interval in $C_f[v]$*. Next, consider any node u in N and let H be the set of hybrid nodes in N that belong to the same biconnected component as u and which are proper descendants of u (the set H may be empty). Then, the set $L(N[u])$ of leaves that are descendants of u in N can be written as:

[3] This particular type of phylogenetic network was first studied by Wang et al. in [34] and later termed *galled-tree* by Gusfield et al. [13,14].
[4] In contrast, other parameters such as the *treewidth* that also measure the tree-likeness of a given (undirected) graph are NP-hard to compute [1].

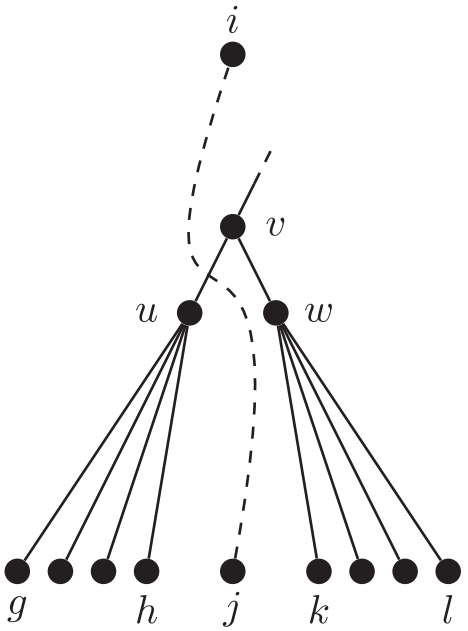

Fig. 3. Illustrating the proof of Lemma 5.

$$L(N[u]) = L(T[u]) \cup \bigcup_{h \in H} L(T[h])$$

Since N is a level-k phylogenetic network, we have $|H| \leqslant k$, which together with the key observation above implies that $C_f[u]$ is the union of at most $k+1$ intervals. It follows that $I_f(u) \leqslant k+1$ for every node u in N. Hence, the spread of f is $I_f = \max_{u \in V} I_f(u) \leqslant k+1$, where V is the set of nodes in N. □

The leaf relabeling scheme invented by Day [10] for phylogenetic trees corresponds to the special case $k = 0$ in Lemma 6 above. Here, $I_f = 1$, so the set of leaves in the subtree rooted at any node of N is represented by a single interval, and $e = O(n)$.

Example 4. Consider again the phylogenetic network in Fig. 1. The leaf numbering in Example 2 gives the interval lists listed in Table 3. The network is level-2 and its spread corresponds to the 3 disjoint intervals (v_{34}, v_{47}), (v_{36}, v_{33}), (v_{18}, v_{14}) of node v_5.

For level-1 networks (that is, galled-trees in the terminology of [13,14]), we can obtain a tighter upper bound on the minimum spread than the one implied by Lemma 6 as follows:

Lemma 7. *If N is a level-1 phylogenetic network then N is leaf-outerplanar.*

Proof. N can be embedded in the plane as follows. Let N' be the graph obtained by, for each biconnected component C_i in $\mathcal{U}(N)$, contracting all nodes and edges in N that belong to C_i into a single node c_i. Then N' is a tree and we may draw N' in the plane. By the definition of a level-1 network, each biconnected component of $\mathcal{U}(N)$ forms a cycle. Reconstruct N from N' by replacing every node c_i by a cycle consisting of the nodes and edges that were contracted in the original N to obtain c_i, while making sure that nothing ends up inside of the cycle. Now, if we consider a large circle enclosing the drawn N, it is clear that every node of N can be reached from the circle without having to cross any edges of N, so the root of N and all leaves of N may be moved to lie on this circle, that is, N is leaf-outerplanar. □

Lemma 7 and Euler's formula imply that $e = O(m)$ for any level-1 phylogenetic network. Also, Lemma 3 in [9] states that for any level-1 phylogenetic network, the total number of nodes m is $O(n)$, so in this case, $e = O(n)$ as well as $m = O(n)$.

Corollary 1. *If N is a level-1 phylogenetic network then a leaf numbering function f with $I_f = 1$ exists and can be computed in $O(n)$ time.*

Proof. Combine Lemmas 5 and 7 to get a leaf numbering function f with $I_f = 1$ in $O(m) = O(n)$ time. □

In summary, we have:

Table 3
Interval list representation of the clusters for the phylogenetic network in Fig. 1.

Node	Interval list	Node	Interval list	Node	Interval list
v_{21}	(v_{27}, v_{28})	v_{29}	(v_{34}, v_{47})	v_7	$(v_{34}, v_{47}), (v_{22}, v_{12})$
v_{15}	(v_{20}, v_{28})	v_9	(v_{20}, v_{47})	v_4	(v_{20}, v_{12})
v_{41}	(v_{45}, v_{45})	v_6	(v_{20}, v_{10})	v_{19}	(v_{25}, v_{26})
v_{39}	(v_{45}, v_{42})	v_{16}	(v_{22}, v_{23})	v_{13}	(v_{18}, v_{26})
v_{44}	(v_{46}, v_{47})	v_{32}	(v_{36}, v_{37})	v_8	(v_{18}, v_{14})
v_{40}	(v_{43}, v_{47})	v_{30}	(v_{36}, v_{33})	v_5	$(v_{34}, v_{47}), (v_{36}, v_{33}), (v_{18}, v_{14})$
v_{38}	(v_{45}, v_{47})	v_{24}	$(v_{34}, v_{47}), (v_{36}, v_{33})$	v_3	(v_{20}, v_{14})
v_{35}	(v_{45}, v_{47})	v_{17}	$(v_{34}, v_{47}), (v_{36}, v_{33})$	v_1	(v_2, v_{14})
v_{31}	(v_{34}, v_{47})	v_{11}	$(v_{34}, v_{47}), (v_{22}, v_{33})$		

Theorem 4. *If N is a level-k phylogenetic network with n leaves, m nodes, and e edges, then the cluster representation of N by interval lists can be computed in $O((k+1)e)$ time using $O((k+1)m\log n)$ bits. In particular, if N is a level-1 phylogenetic network then its cluster representation by interval lists can be computed in $O(n)$ time and $O(n\log n)$ bits.*

Proof. Apply both of Lemmas 3 and 4 to Lemma 6 and Corollary 1, respectively. For the special case of level-1 networks, the complexity becomes $O(e) = O(n)$ time and $O(m\ \log n) = O(n\log n)$ bits. □

5. Computing the Robinson–Foulds distance

We now present the main algorithm of this paper, Algorithm 3, for computing the Robinson–Foulds distance between two input phylogenetic networks N_1, N_2.

The algorithm first constructs the cluster collections $C_1 = C(N_1)$ and $C_2 = C(N_2)$. Next, the cardinality of the symmetric difference of C_1 and C_2 is obtained by radix sorting and then traversing them while using a variable c to count how many elements in C_1 and C_2 are the same. Finally, the algorithm outputs the Robinson–Foulds distance between N_1 and N_2. See [30,31] for further details on simultaneous traversal algorithms on graphs.

The complexity of Algorithm 3 depends on how the cluster collections of N_1 and N_2 are represented, as we demonstrate in Theorem 5 and Corollary 2 below. For $i \in \{1, 2\}$, let m_i and e_i denote the number of nodes and edges of N_i, respectively. Also, let L be the union of the leaf label sets of N_1 and N_2, and define $n = |L|$, $m = \max\{m_1, m_2\}$, and $e = \max\{e_1, e_2\}$.

In the case of general phylogenetic networks, we use the *naive cluster representation* from Section 2 or the *cluster representation by compressed characteristic vectors* from Section 3.1 to implement Algorithm 3. The resulting complexities are given by:

Theorem 5. *Let N_1, N_2 be two phylogenetic networks with n leaf labels and at most m nodes and e edges each. The Robinson–Foulds distance between N_1 and N_2 can be computed in:*

Algorithm 3. Compute the Robinson–Foulds distance between two phylogenetic networks N_1, N_2

```
function robinson_foulds_distance (N1, N2)
  C1 ← cluster_collection(N1)
  C2 ← cluster_collection(N2)
  radix sort C1; radix sort C2
  m1, m2← number of nodes of N1, N2
  i1 ← 1
  i2 ← 1
  c ← 0
  while i1 ⩽ m1 and i2 ⩽ m2 do
    if C1[i1] < C2[i2] then
        i1 ← i1 + 1
    else if C1[i1] > C2[i2] then
        i2 ← i2 + 1
    else
        i1 ← i1 + 1
        i2 ← i2 + 1
        c ← c + 1
  return (m1 + m2 − 2 · c)/2
```

1. $O(n\,e)$ time using $O(nm)$ bits.
2. $O(nm)$ time using $O(nm)$ bits, if N_1 and N_2 have bounded degree.
3. $O(ne/\log n)$ time using $O(nm/\log n)$ words, assuming the word RAM model with a word length of $\lceil \log n \rceil$ bits.
4. $O(nm/\log n)$ time using $O(nm/\log n)$ words, if N_1 and N_2 have bounded degree, assuming the word RAM model with a word length of $\lceil \log n \rceil$ bits.

Proof. Run Algorithm 3, where the first two steps (constructing the cluster collections C_1 and C_2) are implemented as follows. To obtain the first two results, apply Theorem 1 to find the naive cluster representations for C_1 and C_2. For the last two results, take any leaf numbering function f for N_1 and N_2 and then apply Theorem 2 to obtain the cluster representations by compressed characteristic vectors under f.

The complexity to compute C_1 and C_2 depends on Theorems 1 and 2. The radix sort step and remaining operations can be performed in $O(m\,x)$ time, where x denotes the amount of space needed to represent one cluster, that is, $x = O(n)$ bits and $x = O(n/\log n)$ words, respectively. For phylogenetic networks with bounded degree, $e = O(m)$. □

Next, we analyze the complexity when the *cluster representation by interval lists* from Section 4 is employed. The difficulty here is that N_1 might have an efficient leaf numbering function f_1 (that is, with a small spread) and N_2 an efficient leaf numbering function f_2, while f_1 and f_2 are completely different. In this case, we need some way to convert between intervals under f_1 and intervals under f_2 in order to identify clusters that are shared among N_1 and N_2. The proof of the next theorem shows how to do this quickly when at least one of f_1 and f_2 has spread 1:

Theorem 6. *Let N_1 and N_2 be two phylogenetic networks with n leaf labels and at most m nodes and e edges each. Given a leaf numbering function f_1 for N_1 and a leaf numbering function f_2 for N_2 such that $I_{f_1} = 1$, the Robinson–Foulds distance between N_1 and N_2 can be computed in $O(I_{f_2} \cdot e)$ time using $O(I_{f_2} \cdot m \log n)$ bits.*

Proof. Implement the first two steps of Algorithm 3 (constructing the cluster collections C_1 and C_2) as follows:

- Apply Lemmas 3 and 4 to obtain the interval lists for all clusters in N_1 under f_1 and all clusters in N_2 under f_2 using $O(I_{f_2} \cdot m \log n)$ bits and $O(I_{f_2} \cdot e)$ time. By definition, every cluster in N_1 consists of a single interval and every cluster in N_2 consists of at most f_2 intervals. Let C_1 be the multiset of all intervals in N_1.
- Construct the multiset C_2 of all clusters in N_2 that form single intervals under f_1 by modifying a technique from [10], as described next.[5] For each node v in N_2, let $nr\_desc(v)$ denote the number of leaf descendants of v; compute $nr\_desc(v)$ in $O(I_{f_2})$ time for each node v in N_2 simply by summing up the number of elements in its at most I_{f_2} intervals. Then, relabel the leaves in N_2 by the leaf numbering function f_1. For each node v in N_2, define $min(v)$ and $max(v)$ as the leaf descendant of v with the smallest and largest number according to f_1. Do a bottom-up traversal of N_2 in $O(e)$ time to find $min(v)$ and $max(v)$ for every node v in N_2. Observe that for any node v in N_2, if the condition $nr\_desc(v) = max(v) - min(v) + 1$ holds then the set of leaf descendants of v forms a consecutive interval according to f_1 and hence might be a shared cluster between the two networks. Therefore, to construct C_2, initialize $C_2 = \emptyset$ and then, for every node v in N_2, if $nr\_desc(v) = max(v) - min(v) + 1$ then insert the interval $[min(v), max(v)]$ into C_2. Importantly, the multiset C_2 might not contain all clusters of N_2, but will definitely contain those clusters of N_2 that are also clusters of N_1.

After constructing C_1 and C_2, count the number of identical clusters in C_1 and C_2 and compute the Robinson–Foulds distance as specified in Algorithm 3. □

Theorem 6 plus Lemmas 5 and 6, and Corollary 1 imply.

Corollary 2. *Let N_1 and N_2 be two phylogenetic networks with n leaf labels and at most m nodes and e edges each. The Robinson–Foulds distance between N_1 and N_2 can be computed in:*

1. *$O(m)$ time using $O(m \log n)$ bits, if N_1 and N_2 are leaf-outerplanar phylogenetic networks.*
2. *$O((k+1)e)$ time using $O((k+1)m \log n)$ bits, if N_1 is a phylogenetic tree or a level-1 phylogenetic network (that* is, a galled-tree*) and N_2 is a level-k phylogenetic network.*
3. *$O(n)$ time using $O(n \log n)$ bits, if N_1 and N_2 are phylogenetic trees or level-1 phylogenetic networks (that* is, galled-trees*).*

For leaf-outerplanar phylogenetic networks, the running time in Corollary 2 is $O(e) = O(m)$, which is optimal. For cases where N_1 is a level-1 network and the level of N_2 is bounded by a constant, that is, $k = O(1)$, the running time is also (optimal) $O(e)$. Also note that for the special case where both of N_1 and N_2 are phylogenetic trees, $e = O(n)$ holds; then the complexity of our algorithm becomes (optimal) $O(n)$ time and $O(n \log n)$ bits, and this is the same complexity as that of Day's algorithm for computing d_{RF} between two phylogenetic trees [10].

[5] In [10], this technique was used to identify clusters shared by two phylogenetic trees.

6. Empirical results

We have implemented the three cluster representations: naive cluster representation (Algorithm 1), cluster representation by characteristic vectors (Algorithm 2), and cluster representation by interval lists, along with the Robinson–Foulds distance (Algorithm 3) and an additional algorithm for generating random phylogenetic networks with n leaves and m internal nodes, $(m - n + 1)/2$ of which are hybrid. The number of leaves, tree nodes, hybrid nodes, and edges in the random phylogenetic networks are shown in Table 4.

Fig. 4 shows the running time of the Robinson–Foulds distance on random phylogenetic trees and networks for the different representations, averaged over 10 runs (naive cluster representation) or 100 runs (compressed cluster representation). All reported values are user times in seconds on a 2.93 GHz Intel Core i7 Quad-Core processor with 16 GB of memory.

As shown in Fig. 4, there is an improvement by a logarithmic factor in the overall time taken to compute the Robinson–Foulds distance when switching from the naive to the compressed cluster representation. Furthermore, computing the Robinson–Foulds distance is faster when using interval lists instead of characteristic vectors. These empirical results confirm the theoretical analysis done in the previous section.

Table 4
Number of leaves, tree nodes, hybrid nodes, and edges in the random phylogenetic networks used in the experiments.

Leaves	Internal nodes		
	Tree	Hybrid	Edges
100	112	13	237
200	225	26	476
300	337	38	712
400	450	51	951
500	562	63	1187
600	675	76	1426
700	787	88	1662
800	900	101	1901
900	1012	113	2137
1000	1125	126	2376

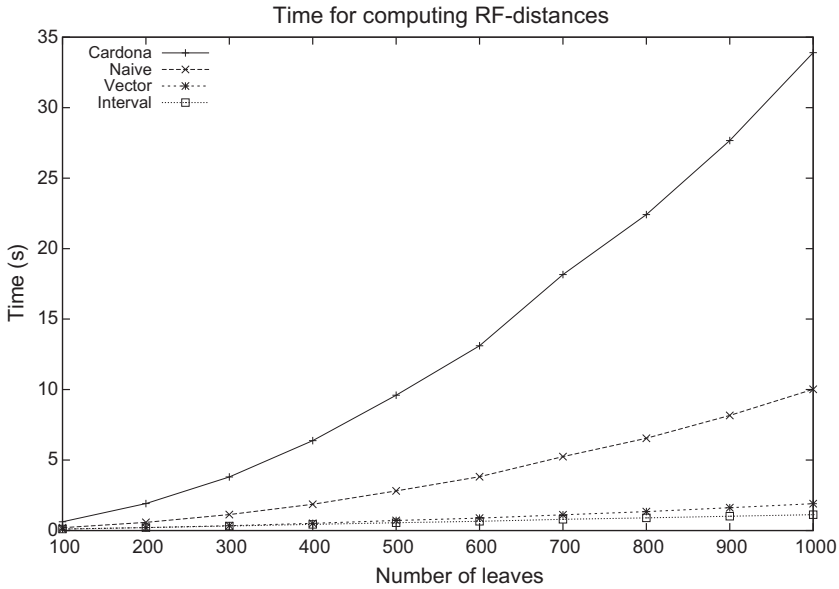

Fig. 4. Time (in seconds) taken to compute the Robinson–Foulds distance between 50×50 pairs of random phylogenetic trees and networks with up to 1000 leaves, using the top-down (Cardona) and bottom-up (Naive) naive cluster representation, the compressed cluster representation by characteristic vectors (Vector), and the compressed cluster representation by interval lists (Interval).

7. Final remarks

We have presented a new, simple algorithm for computing the Robinson–Foulds distance between two given phylogenetic networks N_1, N_2. Our algorithm utilizes word-level parallelism and runs in $O(ne/\log n)$ time for general networks, assuming the word RAM model with a word length of $\lceil \log n \rceil$ bits. For the case of two leaf-outerplanar phylogenetic networks or one level-1 and one level-k phylogenetic network, we can represent clusters as intervals of consecutive integers, and our algorithm runs in $O(m)$ and $O((k+1)e)$ time, respectively. We have also introduced a new parameter, the *minimum spread* of a phylogenetic network, and proved that every level-k network has minimum spread at most $k+1$. For the special case of leaf-outerplanar phylogenetic networks, which includes the class of galled-trees, we have shown that the minimum spread is 1, meaning that our new algorithm can be implemented to run in optimal $O(e)$ time.

We conclude this paper with some open problems:

- Our algorithm computes $d_{RF}(N_1, N_2)$ by comparing the cluster collections $C(N_1)$ and $C(N_2)$. Is there a faster and more direct method to compute $d_{RF}(N_1, N_2)$ without constructing $C(N_1)$ and $C(N_2)$ first?
- How quickly can d_{RF} between two level-k phylogenetic networks with $k > 1$ be computed? It can be done in $O(n\,e)$ time by Theorem 5, but is there any better way that takes advantage of the compactness of the interval list representation for level-k phylogenetic networks guaranteed by Lemma 6? Perhaps hashing can be applied here.
- Can our techniques be extended to efficiently compute the tripartitions distance and the μ-distance mentioned in Section 1.2?
- One application of the Robinson–Foulds distance in phylogenetics is during the construction of a single consensus tree that summarizes a collection of conflicting phylogenetic trees. Here, the dissimilarity for every pair of trees in the given collection may be used as a starting point; see, for instance, [22,28,29]. To cope with huge collections of trees, Pattengale et al. [22] presented a sublinear-time, randomized approximation scheme that returns a close approximation of the Robinson–Foulds distance between every pair of input trees, with high probability. See also [28] for a different randomized method that runs even faster in practice. We wonder if it is possible to adapt the techniques of [22,28] to phylogenetic networks.
- Extra conditions are often imposed on phylogenetic networks to narrow down the output space of reconstruction algorithms [17,18] or to provide a more realistic model of recombination [26,27]. Two such additional conditions are: (1) a phylogenetic network is *time-consistent* when it has a temporal representation [2], that is, an assignment of discrete time stamps to the nodes that increases from parents to tree node children and remains the same from parents to hybrid node children, meaning that the parents of each hybrid node coexist in time and thus, the corresponding reticulate evolutionary event can indeed take place and (2) a phylogenetic network is *tree-child* when every internal node has at least one child that is a tree node [7], meaning that every non-extant species has some extant descendant through mutation alone. Although d_{RF} is not a metric on arbitrary phylogenetic networks, it is a metric on the class of all tree-child time-consistent networks [7, Corollary 1]. A further investigation of what structural restrictions on the networks N_1, N_2 lead to $d_{RF}(N_1, N_2)$ always being a metric is needed, and whether or not such restrictions coincide with biologically meaningful classes of phylogenetic networks.
- Finally, an open problem that is interesting on its own is: What is the computational complexity of computing the minimum spread of an input phylogenetic network?

Acknowledgments

JJ was supported by the Special Coordination Funds for Promoting Science and Technology, Japan, and KAKENHI Grant No. 23700011. TA, RU, GV were supported by the Spanish Government and the EU FEDER program under Project PCI2006-A7-0603.

References

[1] S. Arnborg, D.G. Corneil, A. Proskurowski, Complexity of finding embeddings in a k-tree, SIAM J. Algebra Discr. 8 (2) (1987) 277–284.
[2] M. Baroni, C. Semple, M. Steel, Hybrids in real time, Syst. Biol. 55 (1) (2006) 46–56.
[3] R.G. Beiko, T.J. Harlow, M.A. Ragan, Highways of gene sharing in prokaryotes, Proc. Natl. Acad. Sci. USA 102 (40) (2005) 14332–14337.
[4] D. Bryant, V. Moulton, Neighbor-Net: an agglomerative method for the construction of phylogenetic networks, Mol. Biol. Evol. 21 (2) (2004) 255–265.
[5] G. Cardona, M. Llabrés, F. Rosselló, G. Valiente, A distance metric for a class of tree-sibling phylogenetic networks, Bioinformatics 24 (13) (2008) 1481–1488.
[6] G. Cardona, M. Llabrés, F. Rosselló, G. Valiente, Metrics for phylogenetic networks. I: Generalizations of the Robinson–Foulds metric, IEEE ACM Trans. Comput. Biol. 6 (1) (2009) 46–61.
[7] G. Cardona, F. Rosselló, G. Valiente, Comparison of tree-child phylogenetic networks, IEEE Trans. Comput. Biol. 6 (4) (2009) 552–569.
[8] N. Chiba, T. Nishizeki, S. Abe, T. Ozawa, A linear algorithm for embedding planar graphs using PQ-trees, J. Comput. Syst. Sci. 30 (1) (1985) 54–76.
[9] C. Choy, J. Jansson, K. Sadakane, W.K. Sung, Computing the maximum agreement of phylogenetic networks, Theoret. Comput. Sci. 335 (1) (2005) 93–107.
[10] W.H.E. Day, Optimal algorithms for comparing trees with labeled leaves, J. Classif. 2 (1) (1985) 7–28.
[11] W.F. Doolittle, Phylogenetic classification and the universal tree, Science 284 (5423) (1999) 2124–2128.

[12] S. Grünewald, K. Forslund, A. Dress, V. Moulton, QNet: an agglomerative method for the construction of phylogenetic networks from weighted quartets, Mol. Biol. Evol. 24 (2) (2007) 532–538.
[13] D. Gusfield, S. Eddhu, C.H. Langley, The fine structure of galls in phylogenetic networks, INFORMS J. Comput. 16 (4) (2004) 459–469.
[14] D. Gusfield, S. Eddhu, C.H. Langley, Optimal, efficient reconstruction of phylogenetic networks with constrained recombination, J. Bioinformatics Comput. Biol. 2 (1) (2004) 173–213.
[15] T. Hagerup, Sorting and searching on the word RAM, in: Proc. 15th Annual Symp. Theoretical Aspects of Computer Science, Lect. Notes Comput. Sci, vol. 1373, Springer, 1998.
[16] J. Jansson, N.B. Nguyen, W.-K. Sung, Algorithms for combining rooted triplets into a galled phylogenetic network, SIAM J. Comput. 35 (5) (2006) 1098–1121.
[17] G. Jin, L. Nakhleh, S. Snir, T. Tuller, Maximum likelihood of phylogenetic networks, Bioinformatics 22 (21) (2006) 2604–2611.
[18] G. Jin, L. Nakhleh, S. Snir, T. Tuller, Efficient parsimony-based methods for phylogenetic network reconstruction, Bioinformatics 23 (2) (2007) 123–128.
[19] B.M.E. Moret, L. Nakhleh, T. Warnow, C.R. Linder, A. Tholse, A. Padolina, J. Sun, R. Timme, Phylogenetic networks: modeling, reconstructibility, and accuracy, IEEE Trans. Comput. Biol. 1 (1) (2004) 13–23.
[20] N.B. Nguyen, C.T. Nguyen, W.-K. Sung, Fast algorithms for computing the tripartition-based distance between phylogenetic networks, J. Comb. Optim. 13 (3) (2007) 223–242.
[21] T. Nishizeki, N. Chiba, Planar Graphs: Theory and Algorithms, Dover, 2008.
[22] N.D. Pattengale, E.J. Gottlieb, B.M. Moret, Efficiently computing the Robinson–Foulds metric, J. Comput. Biol. 14 (6) (2007) 724–735.
[23] D. Posada, K.A. Crandall, Intraspecific gene genealogies: trees grafting into networks, Trends Ecol. Evol. 16 (1) (2001) 37–45.
[24] D.F. Robinson, L.R. Foulds, Comparison of phylogenetic trees, Math. Biosci. 53 (1/2) (1981) 131–147.
[25] B.F. Smets, T. Barkay, Horizontal gene transfer: perspectives at a crossroads of scientific disciplines, Nat. Rev. Microbiol. 3 (2005) 675–678.
[26] K. Strimmer, V. Moulton, Likelihood analysis of phylogenetic networks using directed graphical models, Mol. Biol. Evol. 17 (6) (2000) 875–881.
[27] K. Strimmer, C. Wiuf, V. Moulton, Recombination analysis using directed graphical models, Mol. Biol. Evol. 18 (1) (2001) 97–99.
[28] S.-J. Sul, G. Brammer, T.L. Williams, Efficiently computing arbitrarily-sized Robinson*-Foulds distance matrices, in: Proc. 8th Int. Workshop Algorithms in Bioinformatics, Lect. Notes Bioinformatics vol. 5251, Springer, 2008.
[29] S.-J. Sul, T.L. Williams, An experimental analysis of Robinson–Foulds distance matrix algorithms, in: Proc. 16th Ann. European Symposium on Algorithms, Lect. Notes Comput. Sci., vol. 5193, Springer, 2008.
[30] G. Valiente, Efficient algorithms on trees and graphs with unique node labels, in: A. Kandel, H. Bunke, M. Last (Eds.), Applied Graph Theory in Computer Vision and Pattern Recognition, Studies in Computational Intelligencevol, vol. 52, Springer, 2007, pp. 137–149.
[31] G. Valiente, Combinatorial Pattern Matching Algorithms in Computational Biology using Perl and R, Taylor & Francis/CRC Press, 2009.
[32] L. van Iersel, J. Keijsper, S. Kelk, L. Stougie, F. Hagen, T. Boekhout, Constructing level-2 phylogenetic networks from triplets, IEEE ACM Trans. Comput. Biol. 6 (4) (2009) 667–681.
[33] L. van Iersel, S. Kelk, M. Mnich, Uniqueness, intractability and exact algorithms: reflections on level-k phylogenetic networks, J. Bioinformatics Comput. Biol. 7 (4) (2009) 597–623.
[34] L. Wang, B. Ma, M. Li, Fixed topology alignment with recombination, Discr. Appl. Math. 104 (1–3) (2000) 281–300.