Algorithm 4: REVERSE-SEARCH($\pi = p_1 p_2 \ldots p_n$)

```
Let r(π) be a reverse point of π;
Output π;
for each i = 1, 2, ..., r(π) − 1 do
    REVERSE-SEARCH(π[i])
if r(π) ≤ n − 2 and p_{r(π)} < p_{r(π)+2} then
    REVERSE-SEARCH(π[r(π) + 1])
```

is not a child permutation. Based on the above observation, we obtain the enumeration algorithm shown in Algorithm 4. To begin, Algorithm 4 is called with the identity permutation which is the root of the family tree.

By maintaining the reverse point of the current permutation in a traverse of the family tree, we can use a stack to generate each child permutation in O(1) time. To estimate the running time of the algorithm, note that the algorithm can traverse each edge of the family tree in O(1) time. However, the delay time of the algorithm is not bounded by O(1) time for the case that the next permutation is output after deep recursive calls without outputting any permutation. However, by applying the speed-up method proposed by Nakano and Uno [6], we have the following lemma.

Theorem 3 ([9]) *After constructing the root (the identity permutation) in $O(n)$ time, one can enumerate all the permutations in S_n by the reverse search method with a constant time delay. The required working space is $O(n)$.*

Recommended Reading

1. Arimura H, Uno T (2007) An efficient polynomial space and polynomial delay algorithm for enumeration of maximal motifs in a sequence. J Comb Optim 13:243–262
2. Avis D, Fukuda K (1996) Reverse search for enumeration. Discret Appl Math 65(1–3):21–46
3. Goldberg L (1992) Efficient algorithm for listing unlabeled graphs. J Algorithms 13:128–143
4. Goldberg L (1993) Polynomial space polynomial delay algorithms for listing families of graphs. In: Proceedings of the 25th annual ACM symposium on theory of computing, San Diego, pp 218–225
5. Johnson S (1963) Generation of permutations by adjacent transposition. Math Comput 17:282–285
6. Nakano S, Uno T (2004) Constant time generation of trees with specified diameter. In: Proceedings of the 30th workshop on graph-theoretic concepts in computer science (WG 2004), Bad Honnef. LNCS, vol 3353, pp 33–45
7. Savage C (1997) A survey of combinatorial gray codes. SIAM Rev 39(4):605–629
8. Sedgewick R (1977) Permutation generation methods. Comput Surv 9(2):137–164
9. Sekine K, Yamanaka K, Nakano S (2008) Enumeration of permutations. IEICE Trans Fundam Electron Commun Comput Sci J91-A(5):543–549 (in Japanese)
10. Steinhaus H (1964) One hundred problems in elementary mathematics. Basic Books, New York
11. Trotter H (1962) Perm (algorithm 115). Commun ACM 5(8):434–435

Phylogenetic Tree Construction from a Distance Matrix

Jesper Jansson
Laboratory of Mathematical Bioinformatics, Institute for Chemical Research, Kyoto University, Gokasho, Uji, Kyoto, Japan

Keywords

Additive; Dissimilarity matrix; Distance matrix; Phylogenetic tree; Phylogenetic reconstruction; Tree-realizable

Years and Authors of Summarized Original Work

1968; Boesch
1989; Hein
1989; Culberson, Rudnicki
2003; King, Zhang, Zhou

Problem Definition

Let n be a positive integer. A *distance matrix of order n* is a matrix D of size $(n \times n)$ which

satisfies (1) $D_{i,j} > 0$ for all $i, j \in \{1, 2, \ldots, n\}$ with $i \neq j$; (2) $D_{i,j} = 0$ for all $i, j \in \{1, 2, \ldots, n\}$ with $i = j$; and (3) $D_{i,j} = D_{j,i}$ for all $i, j \in \{1, 2, \ldots, n\}$. In the literature, a distance matrix of order n is also called a *dissimilarity matrix of order n*.

Below, all trees are assumed to be unrooted and edge-weighted. For any tree $\mathcal{T}$, the *distance* between two nodes u and v in $\mathcal{T}$ is defined as the sum of the weights of all edges on the unique path in $\mathcal{T}$ between u and v and is denoted by $d^{\mathcal{T}}_{u,v}$. A tree $\mathcal{T}$ is said to *realize* a given distance matrix D of order n if and only if it holds that $\{1, 2, \ldots, n\}$ is a subset of the nodes of $\mathcal{T}$ and $d^{\mathcal{T}}_{i,j} = D_{i,j}$ for all $i, j \in \{1, 2, \ldots, n\}$. Finally, a distance matrix D is called *additive* or *tree-realizable* if and only if there exists a tree which realizes D. See Fig. 1 for an example.

Problem 1 (The Phylogenetic Tree from Distance Matrix Problem)
INPUT: A distance matrix D of order n
OUTPUT: A tree which realizes D and has the smallest possible number of nodes, if D is additive, otherwise *null*

In the time complexities listed below, the time needed to input all of D is not included. Instead, $O(1)$ is charged to the running time whenever an algorithm requests to know the value of any specified entry of D.

Key Results

Several authors have independently shown how to solve the Phylogenetic Tree from Distance Matrix Problem in $O(n^2)$ time. (See [5] for a short survey of older algorithms which do not run in $O(n^2)$ time.)

Theorem 1 ([2, 4, 5, 7, 14]) *There exists an algorithm which solves the Phylogenetic Tree from Distance Matrix Problem in $O(n^2)$ time.*

Although the various existing algorithms are different, it can be proved that:

Theorem 2 ([8,14]) *For any given distance matrix, the solution to the Phylogenetic Tree from Distance Matrix Problem is unique.*

Furthermore, the algorithms referred to in Theorem 1 have optimal running time since any algorithm for the Phylogenetic Tree from Distance Matrix Problem must in the worst case query all $\Omega(n^2)$ entries of D to make sure that D is additive. However, if it is known in advance that the input distance matrix is additive, then the time complexity improves as follows.

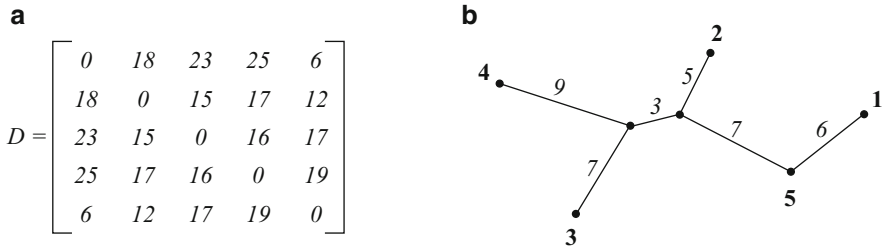
Theorem 3 ([9, 12]) *There exists an algorithm which solves the Phylogenetic Tree from Distance Matrix Problem restricted to additive distance matrices in $O(k\,n \log_k n)$ time, where k is the maximum degree of the tree that realizes the input distance matrix.*

The algorithm of Hein [9] starts with a tree containing just two nodes and then successively inserts each node i into the tree by repeatedly choosing a pair of existing nodes and computing where on the path between them that i should be attached, until i's position has been determined. The same basic technique is used in the $O(n^2)$-time algorithm of Waterman et al. [14] referenced to by Theorem 1 above, but the algorithm of Hein selects paths which are more efficient at discriminating between the possible positions for i. According to [12], the running time of Hein's algorithm is $O(k\,n \log_k n)$.

A lower bound that implies the optimality of Theorem 3 is given by the next theorem.

Theorem 4 ([10]) *The Phylogenetic Tree from Distance Matrix Problem restricted to additive distance matrices requires $\Omega(k\,n \log_k n)$ queries to the distance matrix D, where k is the maximum degree of the tree that realizes D, even if restricted to trees in which all edge weights are equal to* 1.

Independently of [9], Culberson and Rudnicki [5] presented an algorithm for the Phylogenetic Tree from Distance Matrix Problem and claimed it to have $O(k\,n \log_k n)$ time complexity when restricted to additive distance



Phylogenetic Tree Construction from a Distance Matrix, Fig. 1 (**a**) An additive distance matrix D of order 5. (**b**) A tree $\mathcal{T}$ which realizes D. Here, $\{1, 2, \ldots, 5\}$ forms a subset of the nodes of $\mathcal{T}$

matrices and trees in which all edge weights are equal to 1. As pointed out by Reyzin and Srivastava [12], the algorithm actually runs in $\Theta(n^{3/2}\sqrt{k})$ time. See [12] for a counterexample to [5] and a correct analysis. On the positive side, the following special case is solvable in linear time by the Culberson-Rudnicki algorithm:

Theorem 5 ([5]) *There exists an $O(n)$-time algorithm which solves the Phylogenetic Tree from Distance Matrix Problem restricted to additive distance matrices for which the realizing tree contains two leaves only and has all edge weights equal to* 1.

Applications

The main application of the Phylogenetic Tree from Distance Matrix Problem is in the construction of a tree (a so-called *phylogenetic tree*) that represents evolutionary relationships among a set of studied objects (e.g., species or other taxa, populations, proteins, genes, etc.). Here, it is assumed that the objects are indeed related according to a treelike branching pattern caused by an evolutionary process and that their true pairwise evolutionary distances are proportional to the measured pairwise dissimilarities. See, e.g., [1, 6, 7, 14] for examples and many references as well as discussions on how to estimate pairwise dissimilarities based on biological data. Other applications of the Phylogenetic Tree from Distance Matrix Problem can be found in psychology, for example, to describe semantic memory organization [1], in comparative linguistics to infer the evolutionary history of a set of languages [11], or in the study of the filiation of manuscripts to trace how manuscript copies of a text (whose original version may have been lost) have evolved in order to identify discrepancies among them or to reconstruct the original text [1, 3, 13].

In general, real data seldom forms additive distance matrices [14]. Therefore, in practice, researchers consider optimization versions of the Phylogenetic Tree from Distance Matrix Problem which look for a tree that "almost" realizes D. Many alternative definitions of "almost" have been proposed, and numerous heuristics and approximation algorithms have been developed. A comprehensive description of some of the most popular methods for phylogenetic reconstruction from a non-additive distance matrix such as *Neighbor-joining* [16] as well as more background information can be found in, e.g., Chapter 11 of [6]. See also [1] and [15] and the references therein.

Cross-References

- ▸ Distance-Based Phylogeny Reconstruction (Fast-Converging)
- ▸ Distance-Based Phylogeny Reconstruction: Safety and Edge Radius

Acknowledgments JJ was funded by the Hakubi Project at Kyoto University and KAKENHI grant number 26330014.

Recommended Reading

1. Abdi H (1990) Additive-tree representations. In: Dress A, von Haeseler A (eds) Trees and hierarchical structures: proceedings of a conference held at Bielefeld, FRG, Oct 5–9th, 1987. Lecture Notes in Biomathematics, vol 84. Springer, Berlin/Heidelberg, pp 43–59
2. Batagelj V, Pisanski T, Simões-Pereira JMS (1990) An algorithm for tree-realizability of distance matrices. Int J Comput Math 34(3–4):171–176
3. Bennett CH, Li M, Ma B (2003) Chain letters and evolutionary histories. Sci Am 288(6):76–81
4. Boesch FT (1968) Properties of the distance matrix of a tree. Quart Appl Math 26:607–609
5. Culberson JC, Rudnicki P (1989) A fast algorithm for constructing trees from distance matrices. Inf Process Lett 30(4):215–220
6. Felsenstein J (2004) Inferring phylogenies. Sinauer Associates, Sunderland
7. Gusfield DM (1997) Algorithms on strings, trees, and sequences. Cambridge University Press, New York
8. Hakimi SL, Yau SS (1964) Distance matrix of a graph and its realizability. Quart Appl Math 22:305–317
9. Hein J (1989) An optimal algorithm to reconstruct trees from additive distance data. Bull Math Biol 51(5):597–603
10. King V, Zhang L, Zhou Y (2003) On the complexity of distance-based evolutionary tree construction. In: Proceedings of the 14th annual ACM-SIAM symposium on discrete algorithms (SODA 2003), Baltimore, pp 444–453
11. Nakhleh L, Warnow T, Ringe D, Evans SN (2005) A comparison of phylogenetic reconstruction methods on an Indo-European dataset. Trans Philol Soc 103(2):171–192
12. Reyzin L, Srivastava N (2007) On the longest path algorithm for reconstructing trees from distance matrices. Inf Process Lett 101(3):98–100
13. The Canterbury Tales Project. University of Birmingham, Brigham Young University, University of Münster, New York University, Virginia Tech, and Keio University. Website: http://www.petermwrobinson.me.uk/canterburytalesproject.com/
14. Waterman MS, Smith TF, Singh M, Beyer WA (1977) Additive evolutionary trees. J Theor Biol 64(2):199–213
15. Wu BY, Chao K-M, Tang CY (1999) Approximation and exact algorithms for constructing minimum ultrametric trees from distance matrices. J Combin Optim 3(2–3):199–211
16. Saitou N, Nei M (1987) The Neighbor-joining Method: A New Method for Reconstructing Phylogenetic Trees. Mol Biol Evol 4(4):406–425

Planar Directed k-VERTEX-DISJOINT PATHS Problem

Marcin Pilipczuk
Institute of Informatics, University of Bergen, Bergen, Norway
Institute of Informatics, University of Warsaw, Warsaw, Poland

Keywords

Directed graphs; Fixed-parameter tractability; Graph decomposition; Planar graphs; VERTEX-DISJOINT PATHS problem

Years and Authors of Summarized Original Work

1994; Schrijver
2013; Cygan, Marx, Pilipczuk, Pilipczuk

Problem Definition

In the classic VERTEX-DISJOINT PATHS problem, the input consists of an n-vertex graph G and k pairs of terminals $(s_i, t_i)_{i=1}^{k}$, and the question is whether there exist pairwise VERTEX-DISJOINT PATHS $P_1, P_2, \ldots, P_k$ such that for every $1 \leq i \leq k$, the path P_i starts in s_i and ends in t_i. In this entry we are interested in the complexity of this problem restricted to planar directed graphs.

Key Results

An algorithm for the VERTEX-DISJOINT PATHS problem in undirected graphs with running time $f(k)n^3$ for some function f is one of the key ingredients of the minor testing algorithm of Robertson and Seymour [8]. The approach can be summarized as follows: either the input graph has treewidth bounded by a function of k, in which case we can apply standard dynamic