Cross-References

▸ Support Vector Machines

Recommended Reading

1. Agmon S (1954) The relaxation method for linear inequalities. Can J Math 6(3):382–392
2. Block HD (1962) The perceptron: a model for brain functioning. Rev Mod Phys 34:123–135
3. Blum A, Dunagan JD (2002) Smoothed analysis of the perceptron algorithm for linear programming. In: Proceedings of the thirteenth annual symposium on discrete algorithms, San Francisco
4. Cesa-Bianchi N, Gentile C (2006) Tracking the best hyperplane with a simple budget perceptron. In: Proceedings of the nineteenth annual conference on computational learning theory, Pittsburgh
5. Collins M (2002) Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In: Conference on empirical methods in natural language processing, Philadelphia
6. Crammer K, Dekel O, Keshet J, Shalev-Shwartz S, Singer Y (2006) Online passive aggressive algorithms. J Mach Learn Res 7:551–585
7. Crammer K, Singer Y (2002) A new family of online algorithms for category ranking. In: Proceedings of the 25th annual international ACM SIGIR conference on research and development in information retrieval, Tampere
8. Dekel O, Shalev-Shwartz S, Singer Y (2005) The Forgetron: a kernel-based perceptron on a fixed budget. Adv neural Inf Process Syst 18: 259–266
9. Freund Y, Schapire RE (1998) Large margin classification using the perceptron algorithm. In: Proceedings of the eleventh annual conference on computational learning theory, Madison
10. Gentile C (2002) The robustness of the p-norm algorithms. Mach Learn 53(3), 265–299
11. Minsky M, Papert S (1969) Perceptrons: an introduction to computational geometry. MIT, Cambridge
12. Novikoff ABJ (1962) On convergence proofs on perceptrons. In: Proceedings of the symposium on the mathematical theory of automata, New York, vol XII, pp 615–622
13. Rosenblatt F (1958) The perceptron: a probabilistic model for information storage and organization in the brain. Psychol Rev 65:386–407
14. Shalev-Shwartz S, Singer Y (2005) A new perspective on an old perceptron algorithm. In: Proceedings of the eighteenth annual conference on computational learning theory, Bertinoro, 264–278
15. Vapnik VN (1998) Statistical learning theory. Wiley, New York

Perfect Phylogeny (Bounded Number of States)

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

Keywords

Bounded number of states; Character state matrix; Phylogenetic reconstruction; Phylogenetic tree; Perfect phylogeny

Years and Authors of Summarized Original Work

1994; Agarwala, Fernández-Baca
1997; Kannan, Warnow

Problem Definition

Let $S = \{s_1, s_2, \ldots, s_n\}$ be a set of elements called *objects* and let $C = \{c_1, c_2, \ldots, c_m\}$ be a set of functions called *characters* such that each $c_j \in C$ is a function from S to the set $\{0, 1, \ldots, r_j - 1\}$ for some integer r_j. For every $c_j \in C$, the set $\{0, 1, \ldots, r_j - 1\}$ is called the set of *allowed states* of character c_j, and for any $s_i \in S$ and $c_j \in C$, it is said that the *state of* s_i *on* c_j is α, or that the *state of* c_j *for* s_i is α, where $\alpha = c_j(s_i)$. The *character state matrix* for S and C is the $(n \times m)$-matrix in which entry (i, j) for any $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, m\}$ equals the state of s_i on c_j.

In this encyclopedia entry, a *phylogeny for* S is an unrooted tree whose leaves are bijectively labeled by S. For every $c_j \in C$ and $\alpha \in \{0, 1, \ldots, r_j - 1\}$, define the set $S_{c_j,\alpha}$ by $S_{c_j,\alpha} = \{s_i \in S : \text{the state of } s_i \text{ on } c_j \text{ is } \alpha\}$. A *perfect phylogeny for* (S, C) (if one exists) is a phylogeny T for S such that the following holds: for each $c_j \in C$ and pair of allowed states α, β of c_j with $\alpha \neq \beta$, the minimal subtree of T

that connects $S_{c_j,\alpha}$ and the minimal subtree of T that connects $S_{c_j,\beta}$ are vertex disjoint. See Fig. 1 for an example. *The Perfect Phylogeny Problem* (also called *the Character Compatibility Problem* in the literature [2,9]) is the following:

Problem 1 (The Perfect Phylogeny Problem)

INPUT: An $(n \times m)$-character state matrix M for some S and C.

OUTPUT: A perfect phylogeny for (S, C), if one exists; otherwise, *null*.

Below, we define $r = \max_{j \in \{1,2,\ldots,m\}} r_j$ for the input character state matrix M.

Key Results

The following negative result was proved by Bodlaender, Fellows, and Warnow [2] and, independently, by Steel [14]:

Theorem 1 ([2, 14]) *The Perfect Phylogeny Problem is NP-hard.*

On the other hand, certain restrictions of the Perfect Phylogeny Problem can be solved efficiently. One such special case occurs if the number of allowed states of each character is limited. For this case, Agarwala and Fernández-Baca [1] designed a dynamic programming-based algorithm that builds perfect phylogenies on certain subsets of S called *c-clusters* (also referred to as *proper clusters* in [5, 10] and as *character subfamilies* in [6]) in a bottom-up fashion. Each c-cluster G has the property that: (1) G and $S \setminus G$ share at most one state of each character; and (2) for at least one character, G and $S \setminus G$ share no states. The number of c-clusters is at most $2^r m$, and the algorithm's total running time is $O(2^{3r}(nm^3 + m^4))$, i.e., exponential in r. Hence, the Perfect Phylogeny Problem is polynomial-time solvable when the number of allowed states of every character is upper-bounded by $O(\log(m+n))$. Subsequently, Kannan and Warnow [10] presented a modified algorithm with improved running time. They restructured the algorithm of [1] to eliminate one of the three nested loops that steps through all possible c-clusters and added a preprocessing step which speeds up the innermost loop. The resulting time complexity is given by:

Theorem 2 ([10]) *The algorithm of Kannan and Warnow in [10] solves the Perfect Phylogeny Problem in $O(2^{2r} nm^2)$ time.*

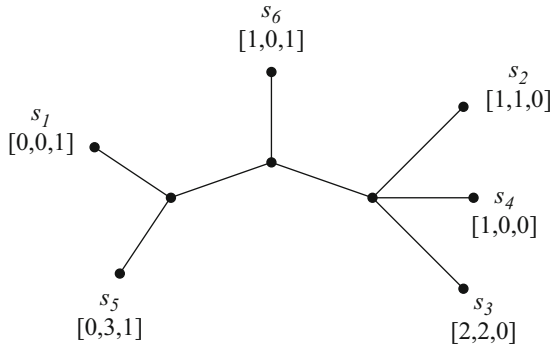
A perfect phylogeny T for (S, C) is called *minimal* if no tree which results by contracting an edge of T is a perfect phylogeny for (S, C). In [10], Kannan and Warnow also showed how to extend their algorithm to enumerate all minimal perfect phylogenies for (S, C) by constructing a directed acyclic graph that implicitly stores the set of all perfect phylogenies for (S, C).

Theorem 3 ([10]) *The extended algorithm of Kannan and Warnow in [10] enumerates the set of all minimal perfect phylogenies for (S, C) so*

a

M	c_1	c_2	c_3
s_1	0	0	1
s_2	1	1	0
s_3	2	2	0
s_4	1	0	0
s_5	0	3	1
s_6	1	0	1

b



Perfect Phylogeny (Bounded Number of States), Fig. 1 (**a**) An example of a character state matrix M for $S = \{s_1, s_2, \ldots, s_6\}$ and $C = \{c_1, c_2, c_3\}$ with $r_1 = 3$, $r_2 = 4$, and $r_3 = 2$, i.e., $r = 4$. (**b**) A perfect phylogeny for (S, C). For convenience, the states of all three characters for each object in S are shown

Perfect Phylogeny (Bounded Number of States), Table 1 The running times of the fastest known algorithms for the Perfect Phylogeny Problem with a bounded number of states

r	Running time	Reference
2	$O(nm)$	[11] together with [7]
3	$\min\{O(nm^2), O(n^2m)\}$	[3,10] together with [9]
4	$\min\{O(nm^2), O(n^2m)\}$	[10] together with [9]
≥ 5	$O(2^{2r}nm^2)$	[10]

that the maximum computation time between two consecutive outputs is $O(2^{2r}nm^2)$.

For small values of r, even faster algorithms are known. Refer to the table in Table 1 for a summary. If $r = 2$, then the problem can be solved in $O(nm)$ time by reducing it to the *Directed* Perfect Phylogeny Problem for Binary Characters (see, e.g., Encyclopedia entry ▸ Directed Perfect Phylogeny (Binary Characters) for a definition of this variant of the problem) using $O(nm)$ time [7,11] and then applying Gusfield's $O(nm)$-time algorithm [7]. If $r = 3$ or $r = 4$, the problem is solvable in $O(n^2m)$ time by another algorithm by Kannan and Warnow [9], which is faster than the algorithm from Theorem 2 when $n < m$. Also note that for the case $r = 3$, there exists an older algorithm by Dress and Steel [3] whose time complexity coincides with that of the algorithm in Theorem 2.

For other special cases of the Perfect Phylogeny Problem that can be solved efficiently, see Encyclopedia entry ▸ Directed Perfect Phylogeny (Binary Characters) or the survey by Fernández-Baca [5].

Applications

Computational evolutionary biology relies on efficient methods for inferring, from some given data, a phylogenetic tree that accurately describes the evolutionary relationships among a set of objects (e.g., biological species, proteins, genes, etc.) assumed to have been produced by an evolutionary process. One of the most widely used techniques for reconstructing a phylogenetic tree is to represent the objects as vectors of character states and look for a tree that clusters objects which have a lot in common. The Perfect Phylogeny Problem can be regarded as the ideal special case of this approach in which the given data contains no errors, evolution is treelike, and each character state can emerge only once in the evolutionary history.

However, data obtained experimentally seldom admits a perfect phylogeny, so various optimization versions of the problem such as *maximum parsimony* and *maximum compatibility* are often considered in practice. These strategies generally lead to NP-complete problems, but there exist heuristics that work well for most inputs. See, e.g., [4, 5, 12] for a discussion. Nevertheless, algorithms for the Perfect Phylogeny Problem may be useful even when the data does not admit a perfect phylogeny, for example, if there exists a perfect phylogeny for $m - O(1)$ of the characters in C. In fact, in one crucial step of their proposed character-based methodology for determining the evolutionary history of a set of related natural languages, Warnow, Ringe, and Taylor [15] consider all subsets of C in decreasing order of cardinality, repeatedly applying the algorithm of [10] until a largest subset of C which admits a perfect phylogeny is found. The ideas behind the algorithms of [1] and [10] have also been utilized and extended by Fernández-Baca and Lagergren [6] in their algorithm for computing *near-perfect phylogenies* in which the constraints on the output have been relaxed in order to permit non-perfect phylogenies whose so-called penalty score is less than or equal to a prespecified parameter q; see [6] for details. (See also [13] for a fixed-parameter tractable algorithm for this problem variant when $r = 2$.)

The motivation for considering a bounded number of states is that characters based on directly observable traits are, by the way they are defined, naturally bounded by some small number (often 2). When biomolecular data is

used to define characters, the number of allowed states is typically bounded by a constant, e.g., $r = 2$ for SNP markers, $r = 4$ for DNA or RNA sequences, or $r = 20$ for amino acid sequences. (see also Encyclopedia entry ▸ Directed Perfect Phylogeny (Binary Characters)). Moreover, characters with $r = 2$ can be useful in comparative linguistics [8].

Open Problems

An open problem is to determine whether the time complexity of the algorithm of Kannan and Warnow [10] can be improved. As noted in [5], it would be interesting to find out if the Perfect Phylogeny Problem is solvable in $O(2^{2r}nm)$ time for any r, or more generally, in $O(f(r)\cdot nm)$ time, where f is a function of r which does not depend on n or m, since this would match the fastest known algorithm for the special case $r = 2$ (see Table 1). Another open problem is to establish lower bounds on the computational complexity of the Perfect Phylogeny Problem with a bounded number of states.

Cross-References

- ▸ Directed Perfect Phylogeny (Binary Characters)
- ▸ Perfect Phylogeny Haplotyping

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

Recommended Reading

1. Agarwala R, 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
2. Bodlaender HL, Fellows MR, Warnow T (1992) Two strikes against perfect phylogeny. In: Proceedings of the 19th international colloquium on automata, languages and programming (ICALP 1992), Vienna. Lecture notes in computer science, vol 623. Springer, Berlin/Heidelberg, pp 273–283
3. Dress A, Steel M (1992) Convex tree realizations of partitions. Appl Math Lett 5(3):3–6
4. Felsenstein J (2004) Inferring phylogenies. Sinauer Associates, Sunderland
5. Fernández-Baca D (2001) The perfect phylogeny problem. In: Cheng X, Du DZ (eds) Steiner trees in industry. Kluwer Academic, Dordrecht, pp 203–234
6. Fernández-Baca D, Lagergren J (2003) A polynomial-time algorithm for near-perfect phylogeny. SIAM J Comput 32(5):1115–1127
7. Gusfield DM (1991) Efficient algorithms for inferring evolutionary trees. Networks 21:19–28
8. Kanj IA, Nakhleh L, Xia G (2006) Reconstructing evolution of natural languages: Complexity and parameterized algorithms. In: Proceedings of the 12th annual international computing and combinatorics conference (COCOON 2006), Taipei. Lecture notes in computer science, vol 4112. Springer, Berlin/Heidelberg, pp 299–308
9. Kannan S, Warnow T (1994) Inferring evolutionary history from DNA sequences. SIAM J Comput 23(4):713–737
10. Kannan S, Warnow T (1997) A fast algorithm for the computation and enumeration of perfect phylogenies. SIAM J Comput 26(6):1749–1763
11. McMorris FR (1977) On the compatibility of binary qualitative taxonomic characters. Bull Math Biol 39(2):133–138
12. Setubal JC, Meidanis J (1997) Introduction to Computational Molecular Biology. PWS Publishing Company, Boston
13. Sridhar S, Dhamdhere K, Blelloch GE, Halperin E, Ravi R, Schwartz R (2007) Algorithms for efficient near-perfect phylogenetic tree reconstruction in theory and practice. IEEE/ACM Trans Comput Biol Bioinform 4(4):561–571
14. Steel MA (1992) The complexity of reconstructing trees from qualitative characters and subtrees. J Classif 9(1):91–116
15. Warnow T, Ringe D, Taylor A (1996) Reconstructing the evolutionary history of natural langauges. In: Proceedings of the 7th annual ACM-SIAM symposium on discrete algorithms (SODA'96), Atlanta, pp 314–322

Perfect Phylogeny Haplotyping

Giuseppe Lancia
Department of Mathematics and Computer Science, University of Udine, Udine, Italy

Keywords

Alleles phasing