Reconstructing the Evolutionary History of Indo-European Languages using Answer Set Programming

Esra Erdem[1], Vladimir Lifschitz[1], Luay Nakhleh[1], and Donald Ringe[2]

[1] Department of Computer Sciences
University of Texas at Austin, Austin, TX 78712, USA
[2] Department of Linguistics
University of Pennsylvania, Philadelphia, PA 19104, USA

Abstract. The evolutionary history of languages can be modeled as a tree, called a phylogeny, where the leaves represent the extant languages, the internal vertices represent the ancestral languages, and the edges represent the genetic relations between the languages. Languages not only inherit characteristics from their ancestors but also sometimes borrow them from other languages. Such borrowings can be represented by additional non-tree edges. This paper addresses the problem of computing a small number of additional edges that turn a phylogeny into a "perfect phylogenetic network". To solve this problem, we use answer set programming, which represents a given computational problem as a logic program whose answer sets correspond to solutions. Using the answer set solver SMODELS, with some heuristics and optimization techniques, we have generated a few conjectures regarding the evolution of Indo-European languages.

1 Introduction

The evolutionary history of languages can be modeled as a tree, called a "phylogeny" (or an "evolutionary tree"), where the leaves represent the extant languages, the internal vertices represent the ancestral languages, and the edges represent the "genetic" relations between the languages. For instance, when we say "French and Italian are both descendants of Latin" we refer to such a tree where French and Italian are denoted by leaves, and Latin is denoted by an internal vertex that is a common ancestor of these two leaves.

Reconstructing phylogenies for various language families is a major endeavor in historical linguistics, but is also of interest to archaeologists, human geneticists, and physical anthropologists. For instance, an accurate reconstruction of the evolutionary history of certain languages can help us answer questions about human migrations, the time that certain artifacts were developed, when ancient people began to use horses in agriculture [4], [5], [11], [15].

Languages not only inherit characteristics from their ancestors but also sometimes borrow them from other languages. In such cases, trees are not a fully adequate model of evolution. Nakhleh *et al.* [7] make this idea precise by defining "perfect phylogenetic networks." They start with a phylogeny built automatically from a dataset describing characteristics of Indo-European languages, and show how some perfect phylogenetic networks can be obtained from it by adding a small number of new edges.

This paper addresses the same computational problem—computing a small set of additional edges that turn a given phylogeny into a perfect phylogenetic network. To solve this problem, we use answer set programming [6], [8], [3]—a new form of declarative programming based on "answer sets" (or stable models) [1], [2]. The idea of answer set programming is to represent a computational problem as a logic program whose answer sets correspond to solutions to the given problem. There are systems specifically designed to compute the answer sets of a logic program. These systems are called answer set solvers. For instance, SMODELS [12] is one of the answer set solvers that are currently available. In the main part of this paper, we assume that the reader has some familiarity with the input language of SMODELS.

In the following, we will first describe the problem mathematically (Section 2) and then formalize it in the language of SMODELS (Section 3). Useful heuristics and optimization techniques will be discussed in Sections 4–6. After that, we will describe the dataset we used to find some explanations to the evolutionary history of the Indo-European languages and present the explanations computed by SMODELS (Section 7). Proofs of theorems can be found in the full version of this paper available at `http://www.cs.utexas.edu/users/esra/ie.pdf`.

2 Problem Description

We describe the problem of computing perfect phylogenetic networks built on a given phylogeny as a graph problem. Therefore, we first introduce some definitions related to graphs.

Recall that a *directed graph (digraph)* is an ordered pair (V, E) where V is a set and E is a binary relation on V. In a digraph (V, E), the elements of V are called *vertices*, and the elements E are called the *edges* of the digraph.

In a digraph, we say that the edge (u, v) is *incident from* u and is *incident into* v. The *out-degree* of a vertex is the number of edges incident from it, and the *in-degree* of a vertex is the number of edges incident into it.

In a digraph (V, E), a *path* from a vertex u to a vertex u' is a sequence $v_0, v_1, \ldots, v_k$ of vertices such that $u = v_0$ and $u' = v_k$, and $(v_{i-1}, v_i) \in E$ for $1 \leq i \leq k$. If there is a path from a vertex u to a vertex v then we say that v is *reachable from* u. If V' is a subset of V, and there exists a path from u to v whose vertices belong to V' then we say that v is *reachable from* u *in* V'.

A *rooted tree* is a digraph with a vertex of in-degree 0, called the *root*, such that every vertex different from the root has in-degree 1 and is reachable from the root. In a rooted tree, a vertex of out-degree 0 is called a *leaf*.

A digraph (V', E') is a *subgraph* of a digraph (V, E) if $V' \subseteq V$ and $E' \subseteq E$.

A *phylogeny* is a triple of the form (V, E, f) where (V, E) is a finite rooted tree and f is a function from $L \times I$ to S, where L is the set of leaves of (V, E), and I, S are finite sets.

We are interested in the problem of turning a phylogeny into a perfect phylogenetic network by adding at most k bidirectional edges. Formally, the problem is defined as follows:

Input: A phylogeny (V, E, f), with $f : L \times I \to S$; a nonnegative integer k.

Output: a function $g : V \times I \to S$ and a symmetric irreflexive binary relation N on V such that

(i) $g|_{L\times I} = f$,
(ii) for all $i \in I$ and $s \in S$, if $V_{is} = \{u \in V : g(u, i) = s\}$ is not empty then $(V, E \cup N)$ has a subgraph with the set V_{is} of vertices that is a rooted tree,
(iii) for every edge $(u, v) \in E$, u is not reachable from v in $(V, E \cup N)$,
(iv) the cardinality of N is at most $2k$.

If (V, E, N, g) satisfies conditions (i)–(iii) then we say that it is a *perfect (phylogenetic) network built on* (V, E, f). The problem described above is essentially the *Minimum Increment to Perfect Phylogenetic Network (MIPPN)* problem [7]. In place of (iv), MIPPN as defined in that paper includes a minimality condition.

Intuitively, the edges of the phylogeny (V, E, f) show the "genetic" relations between languages; each leaf of this phylogeny corresponds to an extant language; the internal vertices represent ancestral languages. The languages are identified by a set of specific observable discrete characteristics, called "(qualitative) characters" (such as grammatical features, unusual sound changes, and cognate classes for different meanings). For every extant language, function f maps every character to a "state"; we say that the leaves of the tree (V, E) are "labeled" by f. Function g extends f to ancestral languages (condition (i)).

Languages can affect each other by transmitting some linguistic properties due to contact. These contacts are not represented on the given phylogeny. They correspond to the elements of N in a perfect network (V, E, N, g) built on (V, E, f). A perfect network explains how every state of every character evolved from its original occurrence in some "root" language (condition (ii)). Languages cannot borrow characteristics from their descendants (condition (iii)). We are only interested in the perfect networks where the number of postulated borrowings is small (condition (iv)), because inheritance of characteristics of a language from its ancestors is far more probable than acquiring them through borrowing. In the case of the phylogeny of Indo-European languages discussed in Section 7 the fact that a small number of edges is sufficient was established in a preliminary analysis done by Tandy Warnow and Donald Ringe.

For instance, consider the phylogeny presented in Figure 1(a) that is reconstructed for 4 extant languages A, B, C, D. There is one character, i.e., $|I| = 1$, and there are two states ($S = \{0, 1\}$). The leaves of the phylogeny are labeled: $f(A, 1) = f(C, 1) = 0$ and $f(B, 1) = f(D, 1) = 1$. A perfect network built on this phylogeny is presented in Figure 1(b). The new bidirectional edge is added to

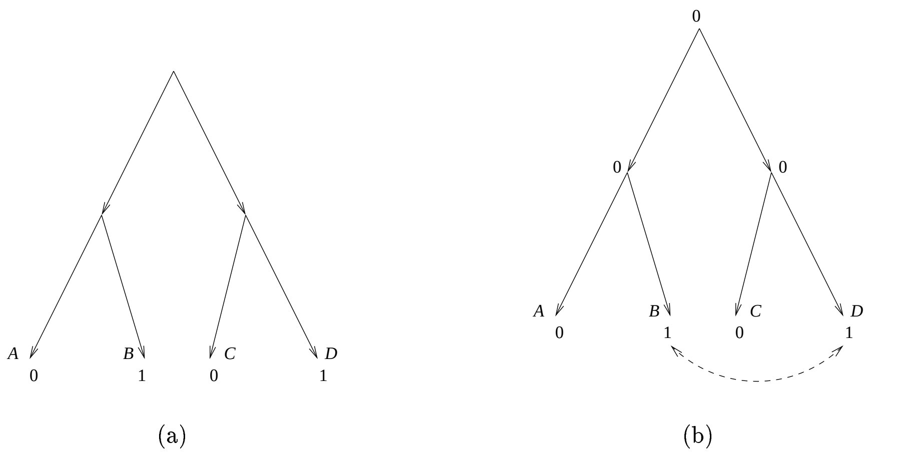


Fig. 1. A phylogeny (a), and a perfect network (b) built on it with $N = \{(B, D), (D, B)\}$.

		Characters	
		0	1
Languages	A	0	0
	B	0	1
	C	1	0
	D	1	1

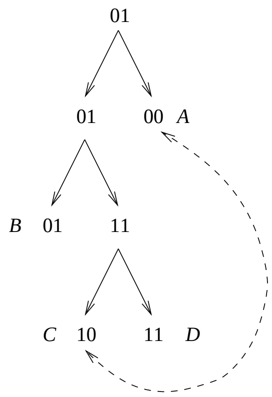


Fig. 2. A perfect network.

make the vertices labeled 1 connected via a rooted tree, i.e., to satisfy condition (ii).

Another example, with two characters ($I = \{0, 1\}$) and two states ($S = \{0, 1\}$), is presented in Figure 2. The new edge is added to make the vertices labeled 0 at Character 1 connected via a rooted tree.

The phylogeny of Indo-European languages described in Section 7 below is a tree with 24 leaves, 370 characters, and 74 states.

```
% a phylogeny for the extant languages A, B, C, D:

% denote A by 1, B by 2, C by 3, D by 4,
%        the parent of A and B by 5,
%        the parent of C and D by 6,
%        the root by 0.

vertex(0..6).

edge(0,5). edge(0,6). edge(5,1).
edge(5,2). edge(6,3). edge(6,4).

state(0;1).

character(0).

f(1,0,0). f(2,0,1). f(3,0,0). f(4,0,1).
```

Fig. 3. Input file describing the phylogeny of Figure 1.

3 Presenting the Problem to SMODELS

A phylogeny is defined by the "domain predicates"[1] `vertex(X)`, `edge(X,Y)`, `state(S)`, `character(C)`, `f(X,C,S)` (expressing that the function f maps the leaf `X` and the character `C` to the state `S`). For instance, the phylogeny of Figure 1 is described to SMODELS by the file presented in Figure 3.[2]

The solutions of the problem are characterized by the atoms of the form `g(X,C,S)` (expressing that the function g maps the vertex `X` and the character `C` to the state `S`) and `new(X,Y)` (expressing that the pairs `(X,Y)` and `(Y,X)` are elements of the set N; `X < Y`). For instance, the output of SMODELS describing the solution presented in Figure 1(b) (with $I = \{0\}$) is:

```
g(0,0,0) g(1,0,0) g(2,0,1) g(3,0,0) g(4,0,1) g(5,0,0) g(6,0,0)
new(2,4)
```

First we describe conditions on the labeling g of the vertices. According to (i), g coincides with f where the latter is defined:

```
g(X,C,S) :- f(X,C,S).
```

Every internal vertex should be labeled by exactly one state for each character:[3]

[1] Intuitively, the domain predicates are the predicates that are used to find all possible variable bindings during grounding. See [13, Definition 8, page 7] for a definition of a domain predicate.

[2] In this file, the expression `vertex(0..6).` has the same meaning as `vertex(0).` `...` `vertex(6).`; the expression `state(0;1).` has the same meaning as `state(0).` `state(1).` [14].

[3] In this rule, for every internal vertex `X` and for every character `C`, the expression `1 {g(X,C,S): state(S)} 1` describes a set of atoms of the form `g(X,C,S)` where `S`

```
1 {g(X,C,S): state(S)} 1 :-
        vertex(X), not leaf(X), character(C).
```

Then, we add at most k pairs of edges between the vertices of this phylogeny:

```
{new(X,Y): vertex(X;Y): X < Y} maxE.
```

(`maxE` represents the value of k).

However, due to (iii), we cannot add such a pair between two vertices if adding it will create a cycle that includes at least one edge of the given phylogeny. To express this condition, we first define the extended set of edges $E \cup N$ by the rules

```
an_edge(X,Y) :- edge(X,Y).
an_edge(X,Y) :- new(X,Y), vertex(X;Y).
an_edge(X,Y) :- new(Y,X), vertex(X;Y).
```

Then we define the binary predicate `directed_path` as the transitive closure of `an_edge`, and impose the constraint

```
:- directed_path(X,Y), edge(Y,X).                        (*)
```

Finally, we need to make sure that (ii) holds. For that we use the following proposition:

Proposition 1 *For any finite digraph (V, E), and any set $V' \subseteq V$, the following conditions are equivalent:*

(a) there exists a subgraph of (V, E) with the set V' of vertices that is a rooted tree,
(b) there exists a vertex $v \in V'$ such that every vertex in V' is reachable from v in V'.

This fact shows that condition (ii) in the statement of the MIPPN problem can be equivalently expressed as follows:

(ii′) for all $i \in I$ and $s \in S$, if $V_{is} = \{u \in V : g(u,i) = s\}$ is not empty then there exists a vertex v in V_{is} such that every vertex in V_{is} is reachable from v in V_{is}.

We consider pairs (i, s) for which some vertex is labeled by state s at character i, so that V_{is} is not empty. After picking an element v of V_{is}, and defining reachability of vertices in V_{is} from v by the predicate `reachable`, we express (ii′) by the constraint

```
:- g(X,C,S), not reachable(X,C,S), character(C),
   state(S), vertex(X).
```

is a state whose cardinality is at least `1` and at most `1`. Such expressions are called “cardinality constraints” [9, page 5].

The answer sets for the program described above correspond to the solutions of the MIPPN problem. We will call it "the basic program."

We can use SMODELS with the basic program to solve small instances of the MIPPN problem. Larger data sets, such as the one described in Section 7, require the use of some heuristics and optimization techniques. Some of these techniques are not complete, that is, do not allow us, generally, to find all solutions. We will discuss them in the following sections.

4 Preprocessing

Sometimes the MIPPN problem for a given phylogeny can be simplified by making its set I of characters smaller. If (V, E, N, g) is a perfect network built on a phylogeny (V, E, f) then, for any subset J of I, $(V, E, N, g|_{V\times J})$ is obviously a perfect network built on $(V, E, f|_{L\times J})$. Proposition 2 below shows, for some special choice of J, that every perfect network built on $(V, E, f|_{L\times J})$ can be extended to a network built on (V, E, f).

Consider a phylogeny (V, E, f), with $f : L \times I \to S$. We say that a character $j \in I$ is *inessential* if there exists a perfect network $(V, E, \emptyset, g)$ built on $(V, E, f|_{L\times\{j\}})$. For instance, in the phylogeny of Figure 2, the first of the two characters is inessential. In the phylogeny of Indo-European languages described in Section 7, 352 characters out of 370 are inessential.

Proposition 2 *Let (V, E, f) be a phylogeny, with $f : L\times I \to S$, and let I' be the set of its inessential characters. There exists a function $g' : V\times I' \to S$ such that, for every perfect network (V, E, N, g) built on $(V, E, f|_{L\times(I\setminus I')})$, $(V, E, N, g\cup g')$ is a perfect network built on (V, E, f).*

This theorem shows that the sets N in solutions to the MIPPN problem for a phylogeny (V, E, f) are identical to the sets N in the solutions to the same problem for the smaller phylogeny $(V, E, f|_{L\times(I\setminus I')})$ (although the functions g in these solutions are, generally, different).

Another way to make a given phylogeny smaller is to check whether it has an internal vertex v such that all leaves descending from v are labeled in the same way. In Figure 4, for instance, this condition holds for the common parent of $A1$ and $A2$. If such a vertex v is found, then we remove all its descendants from the tree, so that v turns into a leaf. The labeling of v in the reduced phylogeny (V', E', f') is the same as the common labeling of the descendants of v in the given phylogeny. For instance, this process turns Figure 4 into the phylogeny of Figure 2.

In our work on the phylogeny of Indo-European languages with 24 leaves, we repeated this process several times, and reduced the given tree to a tree with 15 leaves.

Every solution to the MIPPN problem for the pruned phylogeny can be extended to a solution for the original phylogeny:

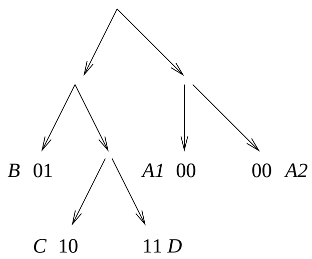


Fig. 4. A phylogeny.

Proposition 3 *Let (V, E, f) be a phylogeny, with $f : L \times I \to S$, and (V', E', f') be the phylogeny obtained from it as described above. Let (V', E', N, g') be a perfect network built on (V', E', f'). Then there exists a function g from $V \times I$ to S with $g|_{V' \times I} = g'$ such that (V, E, N, g) is a perfect network built on (V, E, f).*

However this preprocessing is not complete, i.e., there may be a solution to the MIPPN problem such that its set N of new edges will not be found after pruning vertices. For instance, the perfect network built on the phylogeny of Figure 4 in which new edges connect C with $A1$ and with $A2$ will not be generated after $A1$ and $A2$ are removed from the tree.

5 Partial Perfect Networks and Essential States

The basic program (Section 3) can be improved using "partial" perfect phylogenetic networks—a generalization of the "total" version of this concept defined in Section 2.

Let (V, E, f) be a phylogeny, with $f : L \times I \to S$. A *partial perfect (phylogenetic) network* built on this phylogeny is a quadruple of the form (V, E, N, g), where N is a symmetric irreflexive binary relation on V, and g is a partial mapping of $V \times I$ to S such that the domain of g contains $L \times I$, and conditions (i)–(iii) from Section 2 are satisfied. For instance, a partial perfect network built on the phylogeny of Figure 1 can be defined by

$$N = \{(A, C), (C, A), (B, D), (D, B)\}, \ g = f$$

(Figure 5).

Every partial perfect network can be extended to a perfect network:

Proposition 4 *Let (V, E, f) be a phylogeny, with $f : L \times I \to S$. For any partial perfect network (V, E, N, g) built on this phylogeny there exists an extension g' of g to $V \times I$ such that (V, E, N, g') is a perfect network built on the same phylogeny.*

Proposition 5 below shows, on the other hand, that every perfect network can be obtained by extending a partial perfect network satisfying a certain condition, expressed in terms of "essential states." Let (V, E, f) be a phylogeny, with $f : L \times I \to S$. We say that a state $s \in S$ is *essential* with respect to a character $j \in I$

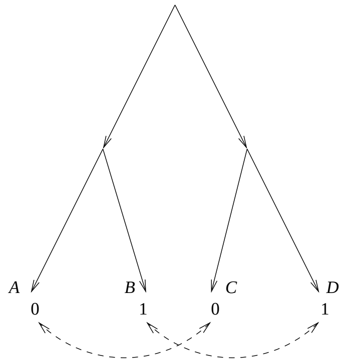


Fig. 5. A partial perfect network built on the phylogeny of Figure 1 with $N = \{(A,C),(C,A),(B,D),(D,B)\},\ g = f$.

if there exist two different leaves l_1 and l_2 in L such that $f(l_1, j) = f(l_2, j) = s$. For instance, in the phylogeny of Figure 6(a), State 3 is essential, and States 1 and 2 are not. There is no need to use inessential states to label internal vertices:

Proposition 5 *Let (V, E, f) be a phylogeny, with $f : L \times I \rightarrow S$. For any perfect network (V, E, N, g') built on this phylogeny there exists a partial mapping g of $V \times I$ to S such that*

- *(V,E,N,g) is a partial perfect network built on the same phylogeny,*
- *g' is an extension of g to $V \times I$, and*
- *$g(v, i)$ is essential with respect to i whenever $v \notin L$.*

For instance, if (V, E, N, g') is the perfect network of Figure 6(a), then the partial perfect network of Figure 6(b) satisfies the conditions of this theorem as (V, E, N, g). In this partial perfect network, inessential states 1 and 2 are not used for labeling internal vertices.

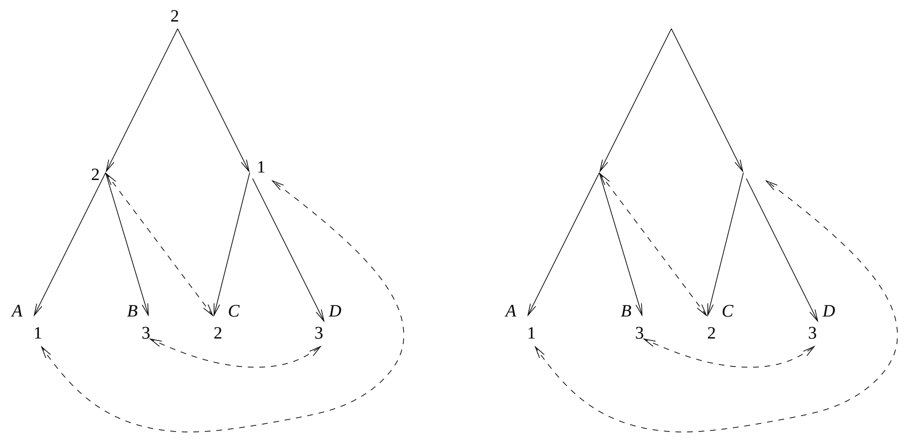
Propositions 4 and 5 show that the problem of computing a perfect network built on a given phylogeny is closely related to the problem of computing a partial perfect network (V, E, N, g) built on this phylogeny such that $g(v, i)$ is essential with respect to i for every internal vertex v and every character i. The sets N in the solutions to this modification of the problem are identical to the sets N in the solutions to the original problem (although the functions g are, generally, different).

In the phylogeny of Indo-European languages described in Section 7, 61 out of the 74 states are not essential with respect to any of its essential characters.

To adapt the basic program (Section 3) to the modification of the problem described above, we replace

```
1 {g(X,C,S): state(S)} 1 :-
        vertex(X), not leaf(X), character(C).
```

with the rule



(a) (b)

Fig. 6. A perfect network (a), and a partial perfect network (b) obtained from it via Proposition 5.

```
{g(X,C,S): essential_state(C,S)} 1 :-
        vertex(X), not leaf(X), character(C).
```

where `essential_state` is defined by the rule

```
essential_state(C,S) :- f(X,C,S), character(C),
        f(X1,C,S), X != X1, vertex(X;X1).
```

6 A Divide-and-Conquer Strategy

Recall that a perfect network built on a phylogeny (V, E, f) is a quadruple (V, E, N, g) satisfying conditions (i)–(iii) from Section 2. If (V, E, N, g) satisfies the first two of these conditions, we will call it an *almost perfect network*.

The divide-and-conquer approach we use is based on the following fact: if, for each $j \in I$, (V, E, N_j, g_j) is an almost perfect network built on $(V, E, f|_{V\times\{j\}})$ then $(V, E, \bigcup_j N_j, \bigcup_j g_j)$ is an almost perfect network built on (V, E, f). In view of this fact, solutions to the MIPPN problem can be generated by finding such networks (V, E, N_j, g_j) for all characters j and checking that $(V, E, \bigcup_j N_j, \bigcup_j g_j)$ satisfies conditions (iii) and (iv).

Proposition 6 below shows that this process can generate every *minimal* solution—every network (V, E, N, g) in which the set N of new edges cannot be replaced by its proper subset without violating condition (ii).

Proposition 6 *Let (V, E, N, g) be a minimal almost perfect network built on (V, E, f). For every $j \in I$, there exists a minimal almost perfect network $(V, E, N_j, g|_{V\times\{j\}})$ built on $(V, E, f|_{L\times\{j\}})$ such that $\bigcup_{j\in I} N_j = N$.*

To compute almost perfect networks (V, E, N_j, g_j) for each character j, we use the basic program without the definition of `directed_path` and without constraint `(*)`. We can find all such networks using the `compute all` $\{\}$ statement of SMODELS. However, there may be more than one such network with the same set N of new edges where the labelings g of the vertices differ; we are only interested in solutions with different sets of new edges. For this reason, we wrote a script that calls SMODELS repeatedly to compute one value of N at a time. To ensure that every next N_j computed by SMODELS is different from the sets computed so far, we add appropriate constraints to the program at every iteration. For instance, if the first call to SMODELS produces `new(1,2), new(3,4)` then the constraint

```
:- new(1,2), new(3,4).
```

will be added to the program when SMODELS is called for the second time, so that SMODELS will now compute an answer set that does not contain {`new(1,2), new(3,4)`}. To compute all minimal almost perfect networks with at most $2k$ new edges, we start with `maxE=0` and increment `maxE` by 1 until it reaches k.

The verification of conditions (iii) and (iv) for the networks $(V, E, \bigcup_j N_j, \bigcup_j g_j)$ generated from the almost perfect networks (V, E, N_j, g_j) is performed by SMODELS programs.

This divide-and-conquer strategy can be extended to partial networks (Section 5) in a straightforward way.

7 The Evolutionary History of the Indo-European Languages

We have applied the computational methods described above to the phylogeny of Indo-European languages that was generated automatically [10] on the basis of a dataset assembled by Donald Ringe and Ann Taylor, who are specialists in Indo-European historical linguistics, with the advice of other specialist colleagues. Figure 7 shows the tree (V, E) of this phylogeny. Its leaves correspond to the following 24 Indo-European languages: Hittite (HI), Luvian (LU), Lycian (LY), Tocharian A (TA), Tocharian B (TB), Vedic (VE), Avestan (AV), Old Persian (PE), Classical Armenian (AR), Ancient Greek (GK), Latin (LA), Oscan (OS), Umbrian (UM), Gothic (GO), Old Norse (ON), Old English (OE), Old High German (OG), Old Irish (OI), Welsh (WE), Old Church Slavonic (OC), Old Prussian (PR), Lithuanian (LI), Latvian (LT), and Albanian (AL). All these languages are “historic,” that is, recorded, and the position of every leaf vertex against the time line in Figure 7 corresponds to the earliest period at which there is substantial attestation of the corresponding language.

The internal vertices of this tree represent “prehistoric” languages, or “protolanguages,” which were reconstructed by comparison of their descendants. For instance, Vertex 38 is proto-Celtic, reconstructed by comparison of Old Irish and Welsh. The position of every internal vertex against the time line corresponds

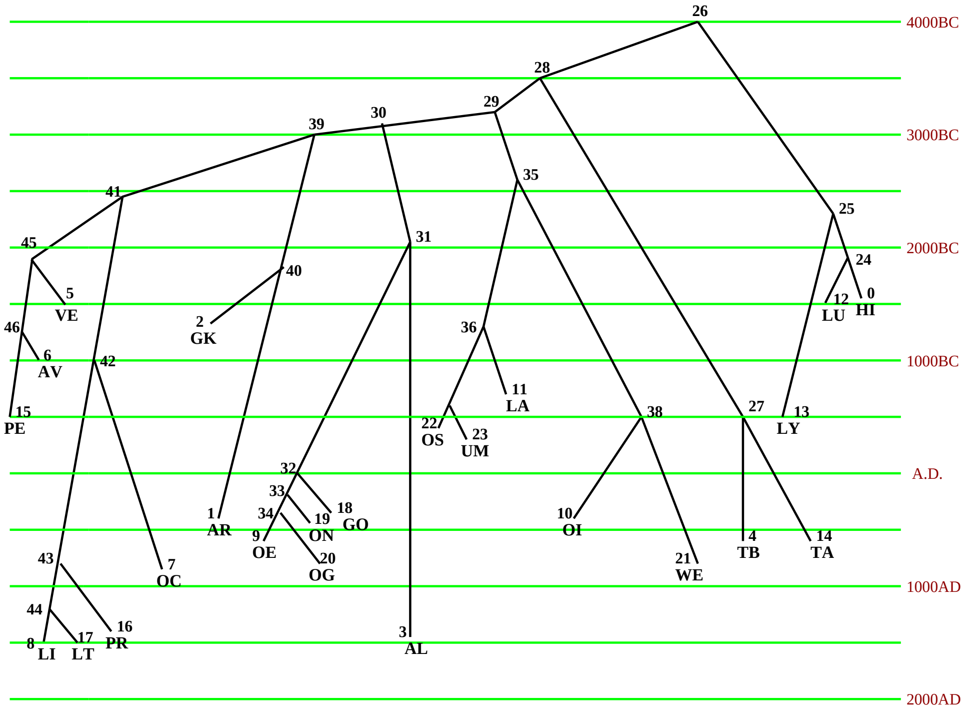


Fig. 7. The phylogeny obtained from the Indo-European dataset.

to the time period when the corresponding protolanguage split up into daughter languages, each spoken by a different speech community.

There are 370 characters in this phylogeny.[4] Out of 370 characters, 22 are phonological characters encoding regular sound changes that have occurred in the prehistory of various languages, 15 are morphological characters encoding details of inflection (or, in one case, word formation), and 333 are lexical characters defined by meanings on a basic word list.

The SMODELS program used to generate perfect networks built on this phylogeny incorporated several domain-specific constraints. One of these constraints prohibits new edges incident with Vertex 24 and its descendants; this constraint is justified by the fact that the labelings of the leaves HI, LU and LY are "disjoint," in their essential parts, from the labelings of the other leaves. Other constraints prohibit contacts between specific pairs of languages which, we know, were spoken at different times. For instance, Old Prussian (Vertex 16) could not be in contact with proto-Celtic (Vertex 38).

Under these constraints, we have found that there are no solutions to the MIPPN problem with fewer than 5 new bidirectional edges. There is only one solution with 5 new edges:

[4] We disregard the 20 characters that take into account multiple character coding and parallel development.

– (32,38) (38,40) (32,43) (18,34) (7,44) .

According to this solution there are 5 borrowings: between proto-Germanic and proto-Celtic (32,38), between proto-Celtic and proto-Greco-Armenian (38,40), between and proto-Germanic and proto-Baltic (32,43), between Gothic and proto-West-Germanic (18,34), and between Old Church Slavonic and proto-East-Baltic (7,44). Each of these contacts is understood to occur at a time prior to the dates assigned to the two languages in the chronology of Figure 7. The approximate times of these contacts are shown in Figure 8.

We have also computed 52 solutions with 6 new edges; 8 of these solutions do not include any borrowings that would be historically implausible:

– (32,38) (31,36) (27,41) (32,43) (18,34) (7,44)
– (32,38) (31,36) (35,40) (32,43) (18,34) (7,44)
– (32,38) (31,36) (36,40) (32,43) (18,34) (7,44)
– (32,38) (31,36) (27,42) (32,43) (18,34) (7,44)
– (32,38) (31,36) (31,40) (32,43) (18,34) (7,44)
– (32,38) (31,36) (27,45) (32,43) (18,34) (7,44)
– (7,27) (32,38) (38,40) (32,42) (18,34) (7,44)
– (32,38) (38,40) (3,7) (32,42) (18,34) (7,44)

In addition to the phylogeny of Figure 7, we considered its modification in which additional internal vertices are introduced—one vertex in the middle of every edge. This extension reflects the possibility of ancestral languages not represented in the original phylogeny, and it was used as the starting point in the work reported in [7]. The calculations described in that paper are limited to the case when new edges are inserted between additional vertices only. Nakhleh *et al.* found 2 solutions with 5 new edges that satisfy this restriction and are historically plausible. To facilitate comparison with that work, we included the same restriction in our program.

For the modified problem, we have computed 11 solutions with 5 new edges; we found among them the 2 solutions from [7], and also one other historically plausible solution. This solution is very close to the 5-edge solution for the phylogeny of Figure 7.

To sum up, the collection of conjectures about the evolutionary history of Indo-European languages generated in our experiments is richer than what is found in [7] in two ways. First, we arrived at several historically plausible perfect networks built on the phylogeny of Figure 7 without additional vertices; this network was not studied in the earlier work at all. Second, we found one new historically plausible perfect network built on the extended phylogeny. The reason why this network could not be generated by Nakhleh *et al.* is that these authors started their computations with a major simplification step, in which every language group (such as Germanic languages or Baltic languages) is reduced to a single vertex, and 16 of the 18 essential characters are considered.

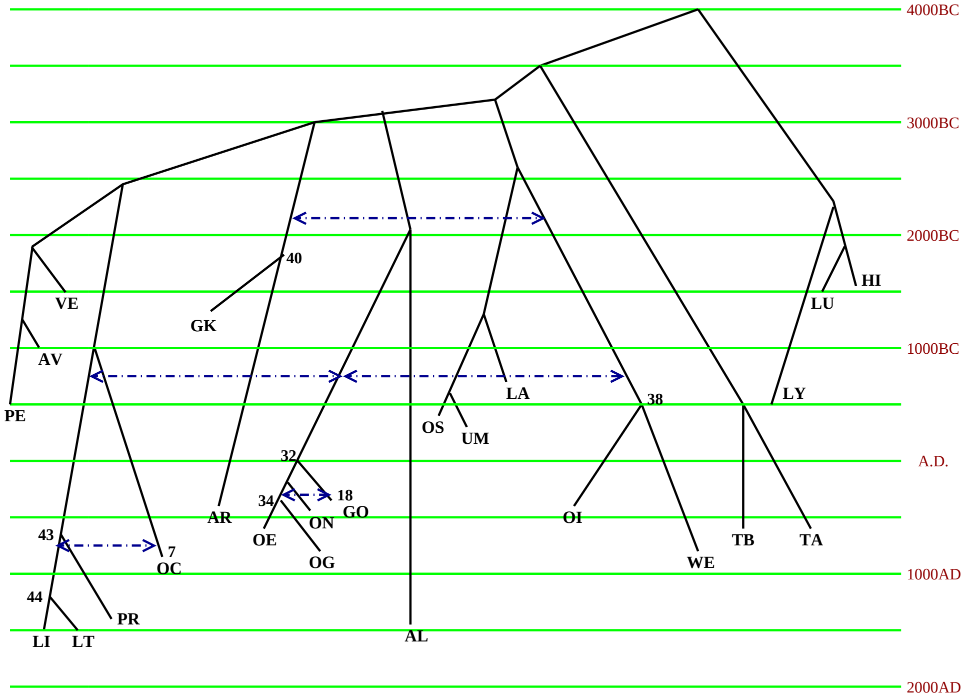


Fig. 8. Contacts between Indo-European languages according to the 5-edge solution.

8 Conclusion

The Minimum Increment to Perfect Phylogenetic Network problem discussed in this paper is a combinatorial search problem well suited to the use of answer set programming. But the basic SMODELS program for the MIPPN problem will produce solutions reasonably fast only if the given phylogeny is small. Several ideas helped us adapt this program to a large phylogeny of Indo-European languages.

One idea is to make the given phylogeny smaller by removing its inessential characters and some of its vertices. This kind of preprocessing is somewhat similar to reducing every language group to a single vertex in [7]. The difference is that our preprocessing is domain-independent—it is defined for any phylogeny. Second, there is no need to use inessential states for labeling the internal vertices of the tree. Finally, a divide-and-conquer strategy allowed us to replace a single run of SMODELS by a series of invocations that solve subproblems of the given problem.

The input program included some domain-specific information about the impossibility of contacts between languages spoken at different times. These constraints helped us further reduce the computation time. Even so, computing the results reported in this paper involved thousands of calls to SMODELS and took more than a week of CPU time. In spite of the presence of several constraints of this kind, most perfect networks computed by our program turned out to be impossible or implausible for historical reasons. To weed out unacceptable solutions, we had to analyze each solution carefully on the basis of the conclusions of earlier research in historical linguistics.

Acknowledgments

We are grateful to Tandy Warnow for useful discussions related to the subject of this paper, and to Selim Erdoğan foruseful comments on a draft of this paper. The first two authors were partially supported by the National Science Foundation under grant IIS-9732744 and by Texas Higher Education Coordinating Board under grant 003658-0322-2001.

References

1. Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert Kowalski and Kenneth Bowen, editors, *Logic Programming: Proc. Fifth Int'l Conf. and Symp.*, pages 1070–1080, 1988.
2. Michael Gelfond and Vladimir Lifschitz. Logic programs with classical negation. In David Warren and Peter Szeredi, editors, *Logic Programming: Proc. Seventh Int'l Conf.*, pages 579–597, 1990.
3. Vladimir Lifschitz. Answer set programming and plan generation. *Artificial Intelligence*, 138:39–54, 2002.
4. V.H. Mair, editor. *The Bronze Age and Early Iron Age Peoples of Eastern Central Asia*. Institute for the Study of Man, Washington, 1998.
5. J.P. Mallory. *In Search of the Indo-Europeans*. Thames and Hudson, London, 1989.
6. Victor Marek and Mirosław Truszczyński. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: a 25-Year Perspective*, pages 375–398. Springer Verlag, 1999.
7. L. Nakhleh, D. Ringe, and T. Warnow. Perfect phylogenetic networks: A new methodology for reconstructing the evolutionary history of natural languages. Submitted for publication, 2002.
8. Ilkka Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25:241–273, 1999.
9. Ilkka Niemelä and Patrik Simons. Extending the Smodel system with cardinality and weight constraints. In Jack Minker, editor, *Logic-Based Artificial Intelligence*. Kluwer, 2000.
10. D. Ringe, T. Warnow, and A. Taylor. Indo-European and computational cladistics. *Transactions of the Philological Society*, 100(1):59–129, 2002.
11. R.G. Roberts, R. Jones, and M.A. Smith. Thermoluminescence dating of a 50,000-year-old human occupation site in Northern Australia. *Science*, 345:153–156, 1990.
12. Patrik Simons, Ilkka Niemelä, and Timo Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138:181–234, 2002.
13. Tommi Syrjänen. Omega-restricted logic programs. In *Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, 2001.
14. Tommi Syrjänen. Lparse 1.0 user's manual[5] 2002.
15. J.P. White and J.F. O'Connell. *A Prehistory of Australia, New Guinea, and Sahul*. Academic Press, New York, 1982.

[5] `http://www.tcs.hut.fi/software/smodels/lparse.ps` .