Reconstructing Evolution of Natural Languages: Complexity and Parameterized Algorithms*

Iyad A. Kanj[1], Luay Nakhleh[2], and Ge Xia[3]

[1] School of Computer Science, Telecommunications and Information Systems,
DePaul University, 243 S. Wabash Avenue, Chicago, IL 60604-2301, USA
ikanj@cs.depaul.edu

[2] Department of Computer Science, Rice University, 6100 Main St., MS 132 Houston,
TX 77005-1892
nakhleh@cs.rice.edu

[3] Department of Computer Science, Lafayette College, Easton, PA 18042, USA
gexia@cs.lafayette.edu

Abstract. In a recent article, Nakhleh, Ringe and Warnow introduced *perfect phylogenetic networks*—a model of language evolution where languages do not evolve via clean speciation—and formulated a set of problems for their accurate reconstruction. Their new methodology assumes *networks*, rather than *trees*, as the correct model to capture the evolutionary history of natural languages. They proved the NP-hardness of the problem of testing whether a network is a perfect phylogenetic one for characters exhibiting at least three states, leaving open the case of binary characters, and gave a straightforward brute-force parameterized algorithm for the problem of running time $O(3^k n)$, where k is the number of bidirectional edges in the network and n is its size. In this paper, we first establish the NP-hardness of the binary case of the problem. Then we provide a more efficient parameterized algorithm for this case running in time $O(2^k n^2)$. The presented algorithm is very simple, and utilizes some structural results and elegant operations developed in this paper that can be useful on their own in the design of heuristic algorithms for the problem. The analysis phase of the algorithm is very elegant using amortized techniques to show that the upper bound on the running time of the algorithm is much tighter than the upper bound obtained under a conservative worst-case scenario assumption. Our results bear significant impact on reconstructing evolutionary histories of languages–particularly from phonological and morphological character data, most of which exhibit at most two states (i.e., are binary), as well as on the design and analysis of parameterized algorithms.

1 Introduction

Languages differentiate and divide into new languages via a process similar to how biological species divide into new species: communities separate (typically

* The first author was supported in part by DePaul University Competitive Research Grant.

D.Z. Chen and D.T. Lee (Eds.): COCOON 2006, LNCS 4112, pp. 299–308, 2006.

geographically), the language changes differently in each of the new communities, and in time people from separate communities can no longer understand each other. While this is not the only means by which languages change, it is this process which is referred to when we say, for example, "French and Italian are both descendants of Latin." The evolution of related languages is mathematically modeled as a rooted tree in which internal nodes represent the ancestral languages and the leaves represent the extant languages.

Reconstructing this process for various language families is a major endeavor within historical linguistics, but is also of interest to archaeologists, human geneticists, and physical anthropologists, for example, because an accurate reconstruction of how certain languages evolved can help answer questions about human migrations, the time that certain artifacts were developed, when ancient people began to use horses in agriculture, the identity of physically European mummies found in China, etc. (see in particular [7,13,18]). Various researchers [2,3,5,14] have noted that if communities are sufficiently separated after they diverge, then the inference of the phylogeny (i.e., evolutionary tree) for the languages can be inferred by comparing the characteristics of the languages (grammatical features, regular sound changes, and cognate classes for different basic meanings), and searching for "perfect phylogenies." However, the problem of determining if a perfect phylogeny exists, and then computing it, is NP-hard [1]. Consequently, efficient techniques for the inference of evolutionary trees for language families were not easily obtained. In the 1990's, various fixed-parameter approaches for the perfect phylogeny problem were developed (although inspired by the biological context rather than the linguistic one). Subsequently, Ringe and Warnow worked together to fully develop the methodology (character encoding and algorithmic techniques) needed to apply these algorithms to the Indo-European language family.

However, while the methodology seemed very clearly heading in the right direction, and even seemed to potentially answer many of the controversial problems in Indo-European evolution (see [10,12,14,15,16]), it became necessary to extend the model to address the problem of how characters evolve when the language communities remain in significant contact. To address this issue, Nakhleh *et al.* introduced the *perfect phylogenetic networks* (PPN) model in which languages do not evolve via a clean speciation process [8,9]. They proved the NP-hardness of the problem of testing whether a network is a perfect phylogenetic one for characters exhibiting at least three states, leaving open the case of binary characters, and gave a straightforward $O(3^k n)$ time parameterized algorithm for the problem [8], where k is the number of bidirectional edges in the network and n is its size.

In this paper we consider the binary case of the problem. This case is of prime interest on its own since it models the problem of reconstructing evolutionary histories of languages, particularly from phonological and morphological character data, most of which exhibit at most two states [6,11,12,14,16,17]. We first prove the NP-hardness of this problem. Then we present a branch-and-bound parameterized algorithm that solves the problem in $O(2^k n^2)$ time. The algorithm employs several interesting structural (network) operations that are very useful in the design of heuristic algorithms for the problem. When analyzed using the standard methods for analyzing parameterized branch-and-bound algorithms, and which usually work under a worst-case scenario assumption, the upper bound obtained on the size of the search tree of the algorithm is $O(3^k)$, matching the upper bound of the trivial brute-force algorithm. This worst-case analysis for a branch-and-search process is usually very conservative— the worst cases can appear very rarely in the entire process, while most other cases permit much better branching and reductions. Instead, we use amortized analysis to show that "expensive" operations can be balanced by efficient ones, and that the actual size of the search tree can be upper bounded by $O(2^k)$. The running time of the algorithm becomes $O(2^k n^2)$. The analysis phase of the algorithm is very elegant illustrating that parameterized algorithms perform much better than their claimed upper bounds, and suggesting that the standard approaches used in analyzing the size of the search tree for parameterized algorithms are very conservative. Most of the proofs in this paper are omitted for lack of space and are available in the technical report **05–007** at the following web address: `http://www.cs.depaul.edu/research/technical.asp`.

2 Inferring Evolutionary Trees

An evolutionary tree, or phylogeny, for a set L of taxa (i.e., species or languages) describes the evolution of the taxa in L from their most recent common ancestor. Each taxon in L corresponds to a leaf in the evolution tree. The Different types of data can be used as input to methods of tree reconstruction; "qualitative character" data, which reflect specific observable discrete characteristics of the taxa under study, are one such type of data. There are several ways of describing qualitative characters: as partitions of the set of taxa into equivalence classes, or as functions that map the taxa to the distinct states. Qualitative characters for languages are grammatical features, unusual sound changes, and cognate classes for different meanings. The assumption of the historical linguistic methodology is that these qualitative characters evolve in such a way that there is no back-mutation (when characters exhibit parallel evolution we can find most of it and exclude those characters). What this means is that when the state of the qualitative character changes in the evolutionary history of the set of languages, it changes to a state which does not exist anywhere else on earth at that time, nor has it appeared earlier. We now formalize this concept mathematically.

Suppose that T is a rooted tree describing the evolution of a set L of languages. Therefore the leaves in T are the languages in L. Suppose that a qualitative character α is defined for each of the languages in L as a function $\alpha : L \to Z$, where Z denotes the set of integers (i.e. each integer represents a possible state for α). That is, α is a labeling to the leaves in T. We say a qualitative character α is compatible (or "convex") on T if we can extend α to every internal node of the tree T, thus defining a qualitative character α', or a labeling to the internal nodes of T, so that for every state, the nodes in T having that specific state induce a connected subgraph of T. (In other words, $\forall z \in Z$, the set of nodes $\{v \in V(T) : \alpha'(v) = z\}$ induces a connected subgraph of T.)

A different way of casting the above problem which is more intuitive is the following. Given a rooted tree T whose leaves are labeled with integers, decide if the internal nodes in T can be labeled so that each set of nodes in T with the same label induces a connected subgraph of T.

Ringe and Warnow postulated that *all* properly encoded qualitative characters for the Indo-European data should be compatible on the true tree, if such a tree existed. Such a tree is called a *perfect phylogeny.* We have the following definition and theorem.

Definition 1. *Let C be a set of qualitative characters defined on a set L of languages. A tree T is a* **perfect phylogeny** *for C and L if every qualitative character in C is compatible on T.*

Theorem 1. *Let T be a phylogenetic tree on a set L of n languages, and assume that each language in L is assigned a state for α. Then we can test the compatibility of α on T in $O(n)$ time.*

The initial analysis of the Indo-European data done by Warnow and Ringe in [16] demonstrated that the IE linguistic data is, nevertheless, “almost perfect”: they found a tree on which the proportion of compatible characters to incompatible characters was enormous. (Even this was quite surprising; the existence of a tree on which a large proportion of characters is compatible is extremely unlikely in biological data analysis.) This suggested that the basic approach was a good one but that the model had to be extended: A tree model is inappropriate and the evolutionary process is better represented as a “network” [8].

3 Phylogenetic Networks Compatibility: Preliminaries and Complexity

This model of how languages evolve on networks references an underlying rooted tree (modeling “genetic descent”) to which bidirectional edges (modeling how linguistic characters can be transmitted through contact) are added. Therefore, the underlying tree is rooted, and the edges of that tree can be naturally oriented from parent to child, whereas the additional edges are by design bidirectional, since contact between language communities can result in the flow of linguistic characters in both directions. This model was formalized in [8] as follows.

Definition 2. *A* phylogenetic network *on a set L of languages is a rooted directed graph $N = (V, E)$ with the following properties:*

(i) $V = L \cup I$, where I denotes added nodes which represent ancestral languages, and L denotes the set of leaves of T.

(ii) E can be partitioned between the edges of a tree $T = (V, E_T)$, and the set of “non-tree” edges or bidirectional edges $E' = E - E_T$. For more convenience in the notation, we will refer to a bidirectional edge by a b-edge. *The edges in T are oriented from parent to child, and hence T is a directed rooted tree.*

(iii) N is "weakly acyclic", i.e., if N contains directed cycles, then those cycles contain only *edges from E'.*

(iv) Every internal node in N has at least two children in T.

Properties (iii) and (iv) above will be referred to as the phylogenetic networks properties*.*

For a phylogenetic network N, we denote by T_N the underlying tree of N. For a node $u \in N$, we denote by $label(u)$ the label of node u, and by $\pi(u)$ the parent of u in T_N. If e is a b-edge between two nodes u and v in the network N, then e has three possible statuses: (1) the edge e can be simply removed denoting that no transfer took place between the two ancestral languages representing u and v, (2) e can be directed from u towards v denoting that the transfer was from the ancestral language representing u to that representing v, or (3) e can be directed from v towards u denoting that the transfer was from the ancestral language representing v to that representing u. If e is directed from u towards v, then the network is transformed as follows. Remove the edge $(\pi(v), v)$ from N, and make u the new parent of v in the resulting network (that is, add the edge (u, v) as a tree edge to the resulting network). Similarly, if e is directed from v towards u, then the edge $(\pi(u), u)$ is removed from N, and the edge (v, u) is added. Note that if there are t b-edges in N, then the t b-edges induce $O(3^t)$ trees based on 3^t different statuses of the t edges. We denote by Γ the set of the trees induced by the t b-edges in N.

An assignment to the statuses of the edges in a network N whose leaves are labeled by a character is said to be *successful* if the character is compatible with the tree induced by this assignment. A *successful labeling* for a compatible tree is a labeling to the nodes of T in which all the nodes with the same label induce a connected subgraph of T.

Note that the order in which the b-edges that are incident on a certain node are assigned can potentially make a difference in the resulting tree.

Definition 3. *Let $N = (V, E)$ be a phylogenetic network on L and Γ be the set of trees induced by all the assignments to the b-edges in N. Let C be a set of characters defined on L, and let $c : L \to Z$ be a character in C. Then c is said to be* compatible *on N if c is compatible on at least one of the trees in Γ. N is called a* Perfect Phylogenetic Network *if all characters in C are compatible on N.*

The Character Compatibility on Phylogenetic Networks problem, denoted henceforth by CCPN, was defined as follows [8].

> **CCPN**
> Given a phylogenetic network $N = (V, E)$ on a set L, and a set of characters C defined on L, decide if N is a perfect phylogenetic network.

This problem was shown to be NP-hard [8] for the case where each character has *at least* three states. We will consider the case of the CCPN problem in which each character has exactly two states. This problem is called the Binary Character Compatibility on Phylogenetic Networks, denoted henceforth by BCCPN. This problem is of prime interest on its own in the field of linguistics (see [6,11,12,14,16,17]).

> **BCCPN**
> Given a phylogenetic network $N = (V, E)$ on a set L, and a set of characters C defined on L such that each character in C has two states (i.e., binary) decide if N is a perfect phylogenetic network.

Remark 1. Deciding if a network N is perfect phylogenetic on a set of characters C reduces to deciding if every character $c \in C$ is compatible on N. Therefore, without loss of generality, we will denote by BCCPN the problem of deciding whether a given binary character c is compatible on N. The mentioning of c becomes irrelevant in this case, and we will simply say N is compatible to denote that the implicit (given) character c is compatible on N.

Theorem 2. BCCPN *is NP-complete.*

Theorem 2 implies that the CCPN problem is NP-complete as well by specialization, giving an alternative, yet different, proof to that in [8].

4 A Parameterized Algorithm for BCCPN

A parameterized problem is a set of pairs of the form (x, k) where x is the input instance and k is a positive integer called the *parameter*. A parameterized problem is said to be *fixed-parameter tractable*, if the problem can be solved in time $f(k)|x|^c$, where f is a computable function of the parameter k, $|x|$ is the input size, and c is a constant independent of k [4]. The area of parameterized algorithms and complexity was introduced mainly in the work of Downey and Fellows [4], and is based on the core observation that for many practical occurrences of intractable problems some parameters remain small, even if the problem instances are large.

Taking the advantage of the fact the the number of b-edges in the phylogenetic network is small [9], the BCCPN problem can be naturally parameterized by the number of b-edges, k, in the phylogenetic network. We call this problem the Parameterized BCCPN problem. It is easy to see that the Parameterized BCCPN problem can be solved in $O(3^k n)$ time, where n is the number of nodes in the phylogenetic network, by enumerating the status of every b-edge in the network, then checking whether the resulting induced tree is compatible. We will significantly improve on this upper bound next. The algorithm we present is a decision algorithm deciding if the network is compatible or not.

Assumption I. Let (N, k) be an instance of Parameterized BCCPN. If there is at most one leaf in N of label 0 (resp. 1), then N is compatible. This is true since if we label all the internal nodes in N with 1 (resp. 0), then every assignment to the b-edges in N is a successful assignment. Since these particular cases can be identified in $O(n)$ time, we will assume henceforth that at any stage of the algorithm, there are at least two leaves of label 0 and at least two leaves of label 1.

Definition 4. *Let N be a phylogenetic network. An internal node s in N is said to be a* splitting node *if there exists a successful assignment to the b-edges in N that results in a compatible tree T, such that there is a valid labeling for the nodes in T with all the nodes in the subtree rooted at s labeled with the same label, and all the other nodes in the tree labeled with the other (different) label.*

Definition 5. *Let N be a phylogenetic network and suppose that s is a splitting node in N. Let A be a successful assignment to the b-edges in N, and let T be the tree induced by A. The assignment A is said to* respect *the splitting node s, if there is a valid labeling for the nodes in T with all the nodes in the subtree rooted at s labeled with the same label, and all the other nodes in the tree labeled with the other (different) label.*

Remark 2. Observe that, if we assume the statements in **Assumption I**, then for any compatible phylogenetic network N there is at least one splitting node in N.

The main algorithm, **Phylogenetic_Compatibility**, which solves the PARAMETERIZED BCCPN problem is given in Figure 2. The algorithm **Phylogenetic_Compatibility** tries every node in N as the splitting node. For each node selected as the splitting node, it calls the subroutine **Is_Compatible** to check whether there exists a successful assignment to N that respects the selected splitting node. Thus, the subroutine **Is_Compatible** works under the assumption that the splitting node is given. The subroutine **Is_Compatible** utilizes the subroutines **Clean**, **Reduce**, and **Merge**, given in Figure 1. These subroutines apply some operations to reduce the network N, and also work under the assumption that the splitting node has been selected.

Proposition 1. *Let N be a phylogenetic network such that none of the operations* **Reduce**, **Clean**, *or* **Merge** *is applicable to N. Then there exist two nodes u and u' in N such that: (1) label(u) = label(u'), (2) (u, u') is a b-edge in N, and (3) all children of u and u' are leaves.*

We call a pair of nodes $\{u, u'\}$ satisfying the three conditions in Proposition 1 a *nice pair*. Proposition 1 establishes the existence of a nice pair in any phylogenetic network N to which none of the operations **Reduce**, **Clean**, or **Merge** is applicable. Now we are ready to present the main algorithm **Phylogenetic_Compatibility** which solves the PARAMETERIZED BCCPN problem. We will assume that

```
Clean($(u, u')$)
Precondition: $label(u) \neq label(u')$ and $(u, u')$ is a b-edge

1. remove the b-edge $(u, u')$ from $N$;

Reduce($u$)
1. if $u$ has two leaf-children with different labels then reject;
2. if all the children of $u$ are leaves and there is no b-edge incident on $u$ then
       if $u$ is marked as the splitting node then
           if there is a leaf in $N$ that is not a child of $u$
               and of the same label as the children of $u$ then reject;
           else accept;
       else
           remove $u$ and its children and replace them with a leaf $l$;
           label $l$ with the same label as the children of $u$;
           add the tree edge $(\pi(u), l)$;
3. if $u$ is unlabeled and has a labeled child $w$ ($w$ could be a leaf) with no
   b-edge incident on $w$ then
     if $w$ is marked as the splitting node then set $label(u) = 1 - label(w)$;
     else set $label(u) = label(w)$;
4. if $u$ is labeled and has an unlabeled child $w$ with no incident b-edge then
     if $w$ is marked as the splitting node then set $label(w) = 1 - label(u)$;
     else set $label(w) = label(u)$;
5. if $u$ is labeled and has at most one leaf-child then
     add two leaves as children to $u$ of the same label as $u$;
6. if $u$ has more than two leaves with the same label then remove all of them
   except two;

Merge($\langle u, u' \rangle$)
Precondition: $label(\pi(u)) \neq label(u) = label(u')$ and $(u, u')$ is a b-edge
1. cut off the tree edge $(\pi(u), u)$ from $N$;
2. remove the b-edge $(u, u')$;
3. identify the two nodes $u$ and $u'$ (i.e., merge the two nodes into one
   new node);
4. let the new node be $w$; set $label(w) = label(u')$ and $\pi(w) = \pi(u')$ (add the
   tree edge $(\pi(u'), w)$);
5. make the children of both $u$ and $u'$ children of $w$;
6. shift all the b-edges that are incident on $u$ and $u'$ to make them incident on
   $w$ without changing the other endpoints of the b-edges;
7. if $u$ or $u'$ is marked as the splitting node then mark the new node $w$ as the
   splitting node;
```

Fig. 1. The subroutine **Merge**

Assumption I is valid before each operation performed by the algorithm and its subroutines. The algorithm is given in Figure 2.

Theorem 3. *The algorithm* **Phylogenetic_Compatibility** *is correct.*

```
Is_Compatible ($N$, $k$)
1. if $k = 0$ and $N$ is not compatible then reject;
2. while Reduce is applicable to a node in $N$ apply it;
3. if any of Clean or Merge is applicable then apply it and go to step 1;
4. let $\{u, u'\}$ be a nice pair in $N$; {* assume without loss of generality that
   $label(u) = label(u') = 1$ *}
     Case 1. Both $\pi(u)$ and $\pi(u')$ are labeled
         remove the b-edge $(u, u')$;
     Case 2. One of $\pi(u)$ and $\pi(u')$ is labeled, say $\pi(u)$. Branch as follows
         first side of the branch: set $label(\pi(u')) = 1$ and remove the
                                   b-edge $(u, u')$;
         second side of the branch: set $label(\pi(u')) = 0$;
     Case 3. (Both $\pi(u)$ and $\pi(u')$ are unlabeled.) Branch as follows
         first side of the branch: set $label(\pi(u)) = 0$;
         second side of the branch: set $label(\pi(u')) = 0$;
         third side of the branch: set $label(\pi(u)) = label(\pi(u')) = 1$ and
                                   remove the b-edge $(u, u')$;
```

```
Phylogenetic_Compatibility
Input: an instance $(N, k)$ of PARAMETERIZED BCCPN where $N$ is a
        phylogenetic network and $k$ is a positive integer
Output: yes/no decision based on whether $N$ is compatible or not
1. for every node $s$ in $N$ do
     1.1. $N' = N$;
     1.2. mark $s$ as the splitting node in $N'$;
     1.3. call Is_Compatible on $(N', k)$;
     1.4. if Is_Compatible returns yes then return yes;
2. return (no);
```

Fig. 2. **Is_Compatible** and **Phylogenetic_Compatibility**

5 Analysis of the Algorithm Is_Compatible

To analyze the running time of the algorithm **Phylogenetic_Compatibility**, and since the algorithm **Phylogenetic_Compatibility** ends up calling the subroutine **Is_Compatible** $O(n)$ times, it suffices to analyze the running time of **Is_Compatible** and multiply it by $O(n)$. The subroutine **Is_Compatible** is a branch-and-bound process, and its execution can be depicted by a search tree. Therefore, the main step in the analysis is deriving an upper bound on the number of leaves in the search tree. The branches performed by the subroutine **Is_Compatible** can be classified into two branches: $(1, 1)$-branches and $(1, 1, 1)$-branches. The latter branch corresponds to an $O(3^k)$ upper bound on the size of the search tree, matching the bound of a trivial brute-force algorithm that enumerates each of the three statuses of every b-edge. Differing from the common analysis techniques based on the worst-case scenario, we use a novel way for analyzing the size of the search tree using amortized techniques, and obtain:

Lemma 1. *Let $\mathcal{T}$ be the search corresponding to the subroutine* **Is_Compatible** *on an instance (N, k). The number of leaves of $\mathcal{T}$ is $O(2^k)$.*

Theorem 4. *The* PARAMETERIZED BCCPN *problem can be solved in time $O(2^k n^2)$ where n is the number of nodes in the network.*

References

1. H. Bodlaender, M. Fellows, and T. Warnow. Two strikes against perfect phylogeny. In *Proceedings of ICALP'92*, LNCS, pages 273–283. Springer Verlag, 1992.
2. A.J. Dobson. Unrooted trees for numerical taxonomy. Unpublished manuscript.
3. A.J. Dobson. Lexicostatistical grouping. *Anthropological Linguistics*, 11:216–221, 1969.
4. R. Downey and M. Fellows. *Parameterized Complexity.* Springer, New York, 1999.
5. H.A. Gleason. Counting and calculating for historical reconstruction. *Anthropological Linguistics*, 1:22–32, 1959.
6. Russell D. Gray and Quentin D. Atkinson. Language-tree divergence times support the anatolian theory of indo-european origin. *Nature*, 426(6965):435–439, November 2003.
7. J.P. Mallory. *In Search of the Indo-Europeans.* Thames and Hudson, London, 1989.
8. L. Nakhleh. *Phylogenetic Networks.* PhD thesis, The University of Texas at Austin, 2004.
9. L. Nakhleh, D. Ringe, and T. Warnow. Perfect phylogenetic networks: A new methodology for reconstructing the evolutionary history of natural languages. *LANGUAGE*, 2005. In press.
10. D. Ringe. Some consequences of a new proposal for subgrouping the IE family. In B.K. Bergen, M.C. Plauche, and A. Bailey, editors, *24th Annual Meeting of the Berkeley Linguistics Society, Special Session on Indo-European Subgrouping and Internal Relations*, pages 32–46, 1998.
11. D. Ringe, T. Warnow, and A. Taylor. Indo-European and computational cladistics. *Transactions of the Philological Society*, 100(1):59–129, 2002.
12. D. Ringe, T. Warnow, A. Taylor, A. Michailov, and L. Levison. Computational cladistics and the position of Tocharian. In V. Mair, editor, *The Bronze Age and early Iron Age peoples of Eastern Central Asia*, pages 391–414. 1998.
13. 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.
14. A. Taylor, T. Warnow, and D. Ringe. Character-based reconstruction of a linguistic cladogram. In J.C. Smith and D. Bentley, editors, *Historical Linguistics 1995, Volume I: General issues and non-Germanic languages*, pages 393–408. Benjamins, Amsterdam, 2000.
15. T. Warnow. Mathematical approaches to comparative linguistics. *Proc. Natl. Acad. Sci.*, 94:6585–6590, 1997.
16. T. Warnow, D. Ringe, and A. Taylor. Reconstructing the evolutionary history of natural languages. Technical Report 95-16, Institute. for Research in Cognitive Science, Univ. of Pennsylvania, 1995.
17. T. Warnow, D. Ringe, and A. Taylor. Reconstructing the evolutionary history of natural languages. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 314–322, 1996.
18. J.P. White and J.F. O'Connell. *A Prehistory of Australia, New Guinea, and Sahul.* Academic Press, New York, 1982.