Detection of Reticulation Events – a Character Based Method

#K C Navada [1] K Chandra Sekaran [2]
1 *Dept of Computer Engg, Manipal Inst of Tech, Manipal, India*
navada.kc@manipal.edu
2 Dept of Computer Engg, National Inst of Tech, Surathkal, India

Abstract

It is observed that during the evolution of species, reticulation is an important event and is very prevalent in several organisms. However, the analytical tools for the representation of these events are still under development. This is primarily because of the difficulty involved in detecting these reticulation events that have taken place during the evolution using the sequences that are currently available in some form. Since it is not possible to get each sequence starting from the root (beginning of the evolution process), inferring the history has to be based on a few assumptions, which are biologically valid. In this paper, we propose an algorithm which, we feel, is a step in this direction. The algorithm detects the sequences which require reticulation events for their formation and returns the count of such sequences.

1. Introduction

Traditionally phylogenetic trees were being used to represent the evolutionary relationship between the species. However, it was observed that in some cases, the relationships could not be represented using a pure tree structure [6] [9]. This is because mutation is not the only event responsible for the evolution. The events such as horizontal gene transfer (HGT), hybridization, recombination all result in non-tree relationships between a node and its parents. A common factor in all these situations is that some of the nodes will have more than one parent – thus resulting in the so called "Phylogenetic Network". Two of the approaches which have drawn quite a bit of attention are the galled tree method [1] and the use of split tree graphs [8] [10]. In this paper, we refer to the presence of any of the above mentioned events which are responsible for the non-tree structure a reticulation event. As is the standard practice in related literatures [1] [2], in this paper also we follow the infinite-site model, which implies the each site mutates not more than once. In other words, there cannot be a back-mutation. In fact, if the back mutations are allowed, the reticulation events can be explained using the ordinary tree structure itself.

One interesting problem in the presence of reticulation events is to determine the number of these reticulation events that have taken place during the evolution process. The algorithm we present attempts to reconstruct the history of evolution top down (that is, from the root which, we assume without any loss of generality, is a sequence of all zeros). The algorithm is based on the following two assumptions:

-First, species, with less number of characters (ones) have come into existence earlier. This is based on the hypothesis that the complexity of the species is continuously increasing. However, this need not be entirely true - reticulation events may sometimes actually result in the formation of a lower complexity species from higher complexity species. However, since these events are relatively rare (as compared to mutation, which can only increase complexity), this assumption is justified.

-Second, sites with higher character frequency (that is, columns with more number of ones) have mutated early during evolution. This is directly implied by the definition infinite-sites model.

2. Definitions

a*) Matrix M*: This is the representation of a set of species (sequences of zeros and ones), with m rows and n columns. Here each row represents a species and each column represents a site.

b*) Root*: The sequence containing all zeros is to be the root of the network. Note that this will not reduce the generality of the problem because, if this is not the case for at any site, then the all the elements along that site may be complemented to ensure this requirement.

c*) Uninformative site*: These are the sites containing all zeros or only one zero or a single one. If a site contains all zeros is obviously uninformative because that site contains no evolutionary information. If a site contains only one zero (it must be in the root), this site has to be invariant in the reticulation event. If a site contains a single one, then it must have been derived from a sequence due to a single mutation event at this site and cannot be the result of a reticulation event.

1-4244-0716-8/06/$20.00 ©2006 IEEE.

d) *Mutation candidate*: A sequence is said to be a mutation candidate if it can be derived from a single node that has already been generated through one or more number of mutations. In fact initially all the sequences are mutation candidates because, any sequence can be derived from the root. However, as the algorithm progresses, the mutations at various sites get distributed among various nodes resulting in nonavailability of all the requited mutated sites in a single node.

e) *Reticulation candidate*: A sequence is said to be a reticulation candidate if it cannot be derived from a single node. In other words, a reticulation candidate is a sequence which is not a mutation candidate.

3. The algorithm

Input: Matrix M containing a root

Output: Number of reticulation events along with the sequences which require reticulation events.

i) Remove all uninformative sites in the set of sequences.

ii) Sort the rows according to the number of ones they contain.

iii) Remove the duplications of sequences.

iv) No_Of_Ones=1
R=0

v) While some sequences still left
{For all sequences with (number of ones = No_Of_Ones) do
{Classify the sequences as mutation candidate or reticulation candidate
For each mutation candidate
{Generate a node
Attach it to its parent node
}
For each reticulation candidate
{Output the sequence
Generate a node
Increment R
}
}
Increment No_Of_Ones
}

vi) Output R

4. Implementation issues

Step (i) removes all the uninformative sites as already defined above. This step reduces time required for the implementation of the step (ii). Requirement of the step (ii) essentially is the consequence of the first assumption mentioned in section 1. Since the presence of duplicated sequences will not serve any purpose they are removed next. Obviously this operation can easily be performed after step (ii). In step (v), the sequences in the matrix M are processed in groups, depending on the number of ones they contain. To determine if a given sequence is a mutation candidate, the network is traversed from the root always choosing the link representing a mutation which is required for the sequence. When no further links are available, if the last node encountered contains all the mutations that have ever taken place during the evolution so far which are required for the sequence in consideration, then it must be a mutation candidate; otherwise, the sequence must be a reticulation candidate. In case of a mutation candidate, generate (possibly a series of) required mutations from the last node encountered, mutating sites with higher frequency first (as per the second assumption above). In case of a reticulation candidate, since its parent is not unique, it is discarded.

5. An Example

As an example, we consider the same example given in [4], pp 381.

The sample consists of eight sequences:

Site 1234
A 0000
B 0101
C 1100
D 0110
E 1111
F 1101
G 1110
H 1001

There are neither uninformative sites nor duplicated sequences in this set of sequences. The step (ii) in the algorithm sorts the sequences, resulting in the following:

A 0000 (-)
B 0101 (2,4)
C 1100 (1,2)
D 0110 (2,3)
H 1001 (1,4)
F 1101 (1,2,4)
G 1110 (1,3,3)
E 1111 (1,2,3,4)

Following step (v), sequences B, C, D are generated with site 2 being mutated first, then the sites 4, 1, and 3 respectively. Then the remaining sequences, being reticulation candidates are generated. Thus the algorithm outputs the number of reticulations as 4, same as is specified in [4].

6. Future work

A related problem to this is to determine the minimum number of reticulation events required to explain the given set of sequences. The method suggested in [3] gives a lower bound, but this bound a very conservative one, being much lower than the actual lower bound. Two relatively better bounds have been proposed in [4]. These methods are dependent on first computing the various local bounds and then combining these results, treating this problem as an optimization problem of an objective function in an integer linear programming problem. Unfortunately, the procedures for computing these bounds computationally expensive as pointed in [2]. Further, the procedures do not keep track of the positions of the reticulation events. It may also be noted that mere determination of the sequences derived out of reticulation events is not sufficient to estimate the number of reticulation events that have taken place, as generation of some of the sequences might require more than one reticulation event as illustrated in the following example containing just four sequences:

A	0000	(-)
B	1001	(1, 4)
C	1100	(1, 2)
D	0101	(2, 4)

The three possible histories are as follows:

i) First A mutates at site 1 giving rise to an interior node, then this node mutates separately at sites 4 and 2 giving rise to sequences B and C respectively. These two sequences then recombine giving another interior node (1, 2, 4). Finally this recombined node gives rise to one more reticulation event with node A producing the sequence D.

ii) First A mutates at site 2 giving rise to an interior node, then this node mutates separately at sites 1 and 4 giving rise to sequences C and D respectively. These two sequences then recombine giving another interior node (1, 2, 4). Finally this recombined node gives rise to one more reticulation event with node A producing the sequence B.

iii) First A mutates at site 4 giving rise to an interior node, then this node mutates separately at sites 1 and 2 giving rise to sequences B and D respectively. These two sequences then recombine giving another interior node (1, 2, 4). Finally this recombined node gives rise to one more reticulation event with node A producing the sequence C.

In the above example, it can be observed that for some of the sequences, more than one reticulation event might be required. In all the three situations one common observation is that the sequence (1, 2, 4) existed some time back in the history (possibly representing an extinct species).

Conclusion

This algorithm must execute faster as compared to the algorithm 3 explained in [4] because of the absence of the recursive call required for its implementation (refer Fig 1 in [2] for details). This has become possible due to the test conducted to decide if a sequence is a mutation candidate or not. However, it is still unclear how close the results are to the actual minimum number of reticulation event requirements.

References

[1] Dan Gusfield, Satish Eddu, Charles Langley "Efficient Reconstruction of Phylogenetic Networks with Constrained Recombination"- IEEE Proceedings of Computational Systems Bioinformatics (CSB'03) 2003

[2] Vineet Bafna, Vikas Bansal "The Number of Recombination Events in a Sample History : Conflict Graph and Lower Bounds IEEE Transactions on Computational Biology and Bioinformatics Vol 1 No 2 April-June 2004, pp 78-90.

[3] R R Hudson, N L Kaplan "Statistical Properties of the Number of Recombination Events in the History of a Sample of DNA Sequences" Genetics Vol 111, 1985, pp 147-164.

[4] Simon R Myers, Robert C Griffiths "Bounds on the Minimum Number of Recombination Events in a Sample History"- Genetics Vol 163, Jan 2003, pp 375-394.

[5] Dan Gusfield "Efficient Algorithms for Inferring Evolutionary Trees" Networks , Vol 21,1991, pp 19-28

[6] L Nakhleh, Jerry Sun, Tandy Warnow, C R Linder, B M E Moret, Anna Tholse "Towards the Development of Computational Tools for Evaluating Phylogenetic Network Reconstruction Methods" Proc Eighth Pacific Symp. Biocomputing (PSB '03), 2003, pp 315-326

[7] Vladimir Makarenkov, Dmytro Kevorkov, Pierre Legendre "Phylogenetic Network Construction Approaches" Applied Mycology and Biotechnology Vol 6, 2006

[8] Andreas W M Dress, Daniel H Huson "Constructing Splits Graphs" IEEE Trasactions on Computational Biology and Bioinformatics, Vol 1, No 3, July-Sept 2004, pp 109-115.

[9] Luay Nakhleh, Li San Wang "Phylogenetic Networks, Trees and Clusters" - The International Workshop on Bioinformatics Research and Applications, LNCS 3515 (2005): 919-926.

[10] Daniel H Huson, Tobias Dezulian, Tobias Klopper, Mike A Steel "Phylogenetic Super Networks from Partial Trees IEEE Transactions on Computational Biology and Bioinformatics, Vol 1, No 4, Oct-Dec 2004, pp 151-158.