Rapid Neighbour-Joining

Martin Simonsen, Thomas Mailund, and Christian N.S. Pedersen

Bioinformatics Research Center (BIRC), University of Aarhus,
C. F. Møllers Allé, Building 1110, DK-8000 Århus C, Denmark
{kejseren,mailund,cstorm}@birc.au.dk

Abstract. The neighbour-joining method reconstructs phylogenies by iteratively joining pairs of nodes until a single node remains. The criterion for which pair of nodes to merge is based on both the distance between the pair and the average distance to the rest of the nodes. In this paper, we present a new search strategy for the optimisation criteria used for selecting the next pair to merge and we show empirically that the new search strategy is superior to other state-of-the-art neighbour-joining implementations.

1 Introduction

The neighbour-joining method by Saitou and Nei [8] is a widely used method for phylogenetic reconstruction, made popular by a combination of computational efficiency combined with reasonable accuracy. With its cubic running time by Studier and Kepler [11], the method scales to hundreds of species, and while it is usually possible to infer phylogenies with thousands of species, tens or hundreds of thousands of species is infeasible. Various approaches have been taken to improve the running time for neighbour-joining. QuickTree [5] was an attempt to improve performance by making an efficient implementation. It outperformed the existing implementations but still performs as a $O\left(n^3\right)$ time algorithm. QuickJoin [6,7], instead, uses an advanced search heuristic when searching for the pair of nodes to join. Where the straight-forward search takes time $O\left(n^2\right)$ per join, QuickJoin on average reduces this to $O\left(n\right)$ and can reconstruct large trees in a small fraction of the time needed by QuickTree. The worst-case time complexity, however, remains $O\left(n^3\right)$, and due to a rather large overhead QuickJoin cannot compete with QuickTree on small data sets. Other approaches, such as "relaxed neighbour-joining" [3,10] and "fast neighbour-joining" [1] modify the optimisation criteria used when selecting pairs to join. The method "relaxed neighbour-joining" has a worst-case $O\left(n^3\right)$ running time while "fast neighbour-joining" has $O\left(n^2\right)$ running time.

In this paper we introduce a new algorithm, RapidNJ, to lower the computing time of canonical neighbour-joining. We improve the performance by speeding up the search for the pair of nodes to join, while still using the same optimisation criteria as the original neighbour-joining method. Worst-case running time remains $O\left(n^3\right)$, but we present experiments showing that our algorithm outperforms both QuickTree, QuickJoin and an implementation of relaxed neighbour-joining on all input sizes.

K.A. Crandall and J. Lagergren (Eds.): WABI 2008, LNBI 5251, pp. 113–122, 2008.

2 Canonical Neighbour-Joining

Neighbour-joining [8,11] is a hierarchical clustering algorithm. It takes a distance matrix D as input, where $D(i,j)$ is the distance between cluster i and j. It then iteratively joins clusters by using a greedy algorithm, which minimises the total sum of branch lengths in the reconstructed tree. Basically the algorithm uses n iterations, where two clusters (i,j) are selected and joined into a new cluster. The two clusters are selected by minimising

$$Q(i,j) = D(i,j) - u(i) - u(j)\,, \tag{1}$$

where

$$u(l) = \sum_{k=0}^{r-1} D(l,k)/(r-2)\,, \tag{2}$$

and r is the number of remaining clusters. When the minimum Q-value $q_{\min} = \min_{0\leq i,j<r} Q(i,j)$ is found, D is updated, by removing the i'th and j'th row and column. A new row and column are inserted with the distances of the new cluster. Distance between the new cluster $a = i \cup j$ and an old cluster k, are calculated as

$$D(a,k) = \frac{D(i,k) + D(j,k) - D(i,j)}{2}\,. \tag{3}$$

The result of the algorithm is a unrooted bifurcating tree where each initial cluster corresponds to a leaf and each join creates an internal node. Finding the pair of clusters to join in each round takes time $O\left(n^2\right)$. The running time of canonical neighbour-joining thus becomes $O\left(n^3\right)$.

3 Rapid Neighbour-Joining

We seek to improve the performance of canonical neighbour-joining by speeding up the search for the pair of clusters to join, while still using the same optimisation criteria as the canonical neighbour-joining method. The overall aim is thus similar to that of QuickJoin, but the approach is different. The RapidNJ algorithm presented in this paper is based on the following observation.

- When searching for $q_{\min}$ in (1), $u(i)$ is constant in the context of row i.

This observation can be used to create a simple upper bound on the values of each row in Q, thereby reducing the search space significantly. The upper bound is dynamically updated based on the Q-values searched. While the algorithm still has a worst-case running time of $O\left(n^3\right)$, our experiments, reported later, show that in practise the algorithm performs much better. To utilize the upper bound, two new data structures, S and I, are used. S is a sorted representation of D, and I maps S to D. Memory consumption is increased due to S and I. The actual increase depends on implementation choices, and can be minimised at the cost of speed. Worst case memory consumption remains $O\left(n^2\right)$.