Dynamic and Parallel Approaches to Optimal Evolutionary Tree Construction

Conference Paper · May 2006

DOI: 10.1109/CCECE.2006.277582 · Source: DBLP

Kazi Zakia Sultana
Montclair State University

DYNAMIC AND PARALLEL APPROACHES TO OPTIMAL EVOLUTIONARY TREE CONSTRUCTION

Anupam Bhattacharjee
Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh.
email: abrbuet@yahoo.com

Kazi Zakia Sultana
Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh.
email: z_sultana00@yahoo.com

Zalia Shams
Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh.
email: setu_18@yahoo.com

Abstract

Phylogenetic trees are commonly reconstructed based on hard optimization problems such as Maximum parsimony (MP) and Maximum likelihood (ML). Conventional MP heuristics for producing phylogenetic trees produce good solutions within reasonable time on small databases (up to a few thousand sequences) while ML heuristics are limited to smaller datasets (up to a few hundred sequences). However, since MP and ML are NP-hard, application of such approaches do not scale large datasets. In this paper, we present a promising divide-and-conquer technique, the ZAZ method, to construct an evolutionary tree. The algorithm has been implemented and tested against five large biological datasets ranging from 5000-7000 sequences and dramatic speedup with significant improvement in accuracy (better than 94%), in comparison to existing approaches, has been obtained. Thus, high quality reconstruction can be obtained for large datasets by using this approach. Moreover, we present here another approach to construct the tree dynamically (when sequences come dynamically with partial information). Finally combining the two approaches, we show parallel approaches to construct the tree when sequences are generated or obtained dynamically.

***Keywords**: Evolutionary tree construction, shared memory model, parallel algorithm, bioinformatics.*

1. Introduction

One of the applications of trees on the field of bioinformatics is *evolutionary tree*. An evolutionary tree is a tree where the leaves represent *species* and internal nodes represent common ancestors. A phylogenetic tree illustrates the evolutionary relationships among group of organisms, or among a family of related nucleic acid or protein sequences. It plays fundamental role in many biological problems such as *multiple sequence alignment*, protein structure and function prediction, and drug design[3]. A well-studied problem in computational biology is the accurate reconstruction of evolutionary trees (also called *phylogenetic trees*) from discrete data sequences[3].

There are two general categories of methods for calculating phylogenetic trees: *distance-based* and *character-based*[1]. The distance-based methods compute a matrix of pairwise distances between sequences in an alignment, and then construct a tree based entirely on the distance computations[2]. *Neighbor-Joining*, *WEIGHBOR*, *BIONJ*, *FASTME* and a latest approach considering maximum-likelihood estimated triplets of sequences belong to this category[4]. The disadvantages of distance-based methods include the inevitable loss of evolutionary information when a sequence alignment is converted to pairwise alignment and bad performance on large datasets. On the other hand, character-based methods examine each column of the alignment separately and look for the tree that best accommodates all the information, such as maximum parsimony (*MP*) or maximum likelihood (*ML*)[5]. *MP* chooses tree that minimizes number of changes required to explain data. *ML* finds a tree that gives the highest likelihood of the observed data. However, the *MP* and *ML* methods are *NP*-hard.

All the methods referred above are categorized as static construction where all sequences are given apriori. But, as the world evolves new species are created every moment. All the earlier methods need to rebuild the tree already constructed. We, in this paper introduce a new category of construction, *dynamic construction* of evolutionary trees, which adjusts the tree with the new data without rebuilding the tree. As, rebuild process needs a lot of time and computation, the new approach gains much speedups with accuracy (≥98%).

Parallel approaches of super tree construction have been studied by Du, Roshan, and Lin [1]. They present algorithm using *Multiple Instruction Multiple Data* (*MIMD)* parallel computing method and applying *recursive DandC* approach with *DCM3* method. Here, we discuss a new parallel approach dealing with *CREW-PRAM* parallel computing model and analyze the load division phase.

So, in this paper, we present

1. A parallel algorithm for evolutionary tree construction.
2. A dynamic approach for constructing or updating evolutionary trees.
3. A combination of parallel and dynamic reconstruction.
4. Theoretical analysis of the algorithm.
5. Experimental result.

2. Preliminaries

The Shared Memory Model consists of a number of processors, each having its own local memory and executing its own local program by communicating through a shared memory unit. Each processor is uniquely identified by an index, called *processor number* or *processor id.* Of the two basic modes of shared memory model, in the *synchronous (parallel random access machine*, *PRAM*) mode, all the processors operate synchronously under the control of a common clock. Among several variations of *PRAM* model based on the simultaneous access of multiple processors to the same location of the global memory, the *concurrent read exclusive write* (*CREW*) *PRAM* model allows simultaneous access for a read instruction only.

3. Dynamic Construction of Small-sized Trees

As the number of species is very large it is costly and time consuming to construct the whole tree again from the scratch when a new sequence arrives even if the best heuristic is used in static construction. Here, we present a dynamic algorithm for evolutionary tree construction. Our procedure assumes that sequences of data come from time to time and the algorithm is capable to update the current tree for the recent data in such a manner that it is very close to the static tree. We follow the experimental method to verify the accuracy of the algorithm.

Let, T , $T^{/}$ be two trees with score S , $S^{/}$ respectively where $S = \sum_{v \in V(T)} height(v)$ and $S^{/} = \sum_{v \in V(T^{/})} height(v)$.

We, as a measure, assume that the tree T differs from $T^{/}$ by $\frac{|S - S^{/}|}{S^{/}} = \overline{accuracy(T)}$

With such measure, we compared the accuracy of our algorithm with *ML*, *MP* methods of static construction and found it with 94% efficiency.

3.1. Pseudocode of the algorithm

Definitions:

H , H_i: Current & previous threshold value[0~1] respectively, initially both are assigned same value[0.5~1].

l: length of sequence

x : Current maximum measure (dissimilarity).

T : The tree where each node b contains

- S_b :a string describing the sequence
- h_b :level distance of the node from the farthest leaf
- Pointer P_b of node points to the nearest node (node with minimum hamming and height difference).

L **:** A link list containing pointers to all isolated nodes and ancestor nodes.

A_p : Previous accuracy assigned zero initially

A_c : Current accuracy assigned zero initially

sm : Function denoting similarity measure

$$sm = \frac{x - Ham\_Dist(seq_1, seq_2)}{x}$$

For three nodes nd_1, nd_2, nd_3

if $sm(nd_1, nd_2) = sm(nd_1, nd_3)$, then nearest of $nd_1 = \min_{2 \le i \le 3}(h(nd_i))$.

Input: String of sequences.

Output: $T^{/}$, The updated tree containing the recent data.

```
ALGORITHM DYNAMIC-ZAZ
BEGIN
1.  $H$ is assigned with the value of $H_i$.
2.  Initialize $x$ to 1.
3.  When a new sequence arrives, make a new $b$ with it.
4.  Search $L$ to find out a node $c$ maximizing $sm(b,c)$.
5.  Insert $b$ in $L$ and update $x$ if necessary.
6.  Based on distance and sequence update $P_b$ of $b$ if necessary.
7.
    7.1. If $x$ is changed in step 5
        7.1.1. All pointers in $L$ are searched to find out two nodes $d$ and $e$ where $sm(d,e)$ is maximum.
        7.1.2. If $sm(d,e) \ge H$ , create an ancestor node $p$ with two children $d$ and $e$. Delete $d$ and $e$ from $L$ and insert $p$ into $L$. The data and pointer fields of $p$ are properly updated.
    7.2. If $x$ is not updated, find out node $f$ pointed by $P_b$.
        7.2.1. If $sm(b, f) \ge H$ ,
            • Create an ancestor node $p$ with two children $b$ and $f$ .
            • Delete $b$ and $f$ from $L$ and insert $p$ in $L$ . The data and pointer fields of $p$ are properly updated.
8.  Repeat Step 7.1.1 and 7.1.2 with all such possible pair maintaining $sm \ge H$ .
```

9. If there are more than one component (forest) in L, decrease H by 0.1.

10. Repeat 7.2.1, 7.2.2, 8, 9 until we get a root of the final evolutionary tree, $T^{/}$.

11. Measure A_c .If $A_c \geq A_p$, increase the value of H_i by 0.1.

END

3.2. Description

With the arrival of a new sequence, we search the linklist of isolated vertices and ancestor nodes, L, to find out the nearest root node (line 3-5). In case of ties, the node with the minimum height difference is taken to minimize total height of the tree imposing high significance on evolutionary time prediction and calculation. P_b of the new node now points to the nearest root node found. P_b of that nearest node may need to be updated. The value of x may also be changed (line 5-6).

Now, if x is not changed, the value of H is checked for the new node with all ancestor and isolated nodes. If it is greater than or equal to the desired threshold, the new node and the nearest node are replaced by a single ancestor node having the former two nodes as its children (7.2.1). If x is changed by the new sequence, two nodes in L which were beyond the range of threshold before, may now fall into the threshold range as the value of sm changes with x.

In such a case, we search for two nodes in L with the similarity between them being maximum among all the pair wise similarity measures. In case of ties; we take the pair with minimum height difference. Continuing such a process the algorithm stops at a point when the tree $T^{/}$ is formed or there is more than one element in L denoting a forest. Now, to form the tree $T^{/}$ we decrease the threshold value stepwise and apply the 7.2.1, 7.2.2, 8, 9 repeatedly until we get the desired tree $T^{/}$ (a single value in L). Then we calculate A_c with given standard value. If $A_c \geq A_p$, we increase H_i by 0.1. Otherwise, we decrease it by 0.1.

3.3. Performance Measurement

3.3.1 Threshold value Trade-offs

Choosing proper value for H needs an interesting tradeoff. If the value of H is high the evolutionary tree tends to have optimum structure but it generates lots of components (i.e., there are many elements remaining in L after step 7). It increases accuracy but simultaneously increases running time because step 8 and 9 needs much more time. Again, if the value of H is lowered, the running time of the algorithm decreases. The more the value of H decreases, the less the number of isolated nodes remains in L. Again, the construction process tends to be static with the increase of H.

A graph obtained from experiments on small data sets (up to 200 species) shows that after a certain limit of threshold value the performance tends to have a constant figure.

3.3.2 Space and Time Complexity

If n denotes the number of sequences obtained, the space complexity of the algorithm is $O(n)$. Total running time of the algorithm is $O(n^3)$. The runtime is not dominating for $n \sim 700$ experimentally. From the performance measurement of the new *Dynamic-ZAZ* algorithm it is obvious that if parallel algorithm is used for construction and the algorithm *Dynamic-ZAZ* is used for a single processor tree construction process, large data sets can be computed optimally with greater speed. One of the major advantages of *Dynamic-ZAZ* is that it needs no particular modeling tool to be implemented.

4. Parallel Algorithm for Construction

In this section we present a generic parallel algorithm for phylogenetic tree construction.

Let, $y =$ maximum value of dissimilarity measure. We cluster the leaves based on an interval *x= [1~y]* (e.g. nodes with sm ranging from *1~x* forms a cluster, nodes from *(x+1) ~2x* forms another cluster and so on). Let n be the maximum number of nodes belonging to some cluster, m be total number of nodes belonging to all clusters and $T(n)$ be the sequential processing time. If we use sequential model for construction, the total run-time of the algorithm we present,

$$T(m,n) = \left\lceil \frac{m}{n} \right\rceil T(n) \quad (1)$$

But if we use parallel models in this purpose, total runtime of the algorithm in parallel model becomes

$$T(m,n) = O(n^3) \quad (2)$$

Now, our target first is to derive a function, f, such that $n = f(x)$. Then we find a value of x to minimize $T(m,n) \approx T(m,x)$ according to Amdahl's law [1].

Let, p is the total number of processes each of which can handle asymptotically at most $n^{/}$ leaves and l is the length of each sequence. To ensure that the process gives the final tree, the following conditions must be satisfied.

$$^{l}c_1 + ^{l}c_2 + \ldots + ^{l}c_x \leq pn^{/} \quad (3)$$

Now from progression and ratio analysis we get that with the increase of hamming distance number of leaves increases approximately (upper bound) $1 + 2/n$ times. Hence from Equation (3) we get that,

$$\frac{(l+2)^x l^x}{l^{x-2}} \leq pn \Rightarrow n \geq l^2 (l+2)^x \quad (4)$$

From Equation (4) and Equation (3) we obtain (we discard the order notation for simplicity and clarity),

$$T(m,x) \geq (ml^2 (l+2)^x [x \lg(l+2) + 2\lg l]) / \lg p \quad (5)$$

Now, for extreme value of x we get,

$$(l+2)^x ml^2 \lg(l+2) + ml^2 (l+2)^x \lg(l+2)[x \lg(l+2) + 2\lg l] - \lg p \le 0$$

To find the value of x we apply Newton-Raphson method ranging from $x = 1 \sim l$ on the above equation. Thus, we get the optimum value of x in $O(\lg l)$ iterations at most.

Getting the optimum value of x, the algorithm now clusters the nodes and applies dynamic supertree construction method inside a cluster and finally converges forming the desired tree. So, what we do is

- First check equation (3) to find out whether we can use *parallel-ZAZ* with the given number of processors.
- If equation (3) is satisfied, we use *parallel-ZAZ* to construct the tree.

```
Algorithm Parallel-ZAZ
BEGIN
  1. Measure dissimilarity and similarity values.
  2. Cluster sequences leased on $x$.
  3. Applying dynamic-ZAZ parallely.
  4. Construct the final tree with the roots found on step 3
     using dynamic-ZAZ.
END
```

From experiments, it is found that the algorithm shows near optimal accuracy comparing to current algorithms (≥92% accuracy). The same measurement as stated in section (3) is used for calculating accuracy. The run time of the algorithm is $O(n^3)$. The algorithm has been tested against large datasets (3000-7000 species) and a dramatic speedup has been observed (check figure section).

5. Conclusion

In this paper we present super tree construction applying parallel algorithms and dynamic construction. We show different tradeoffs and performance measurements in this regard. The new approach establishes a new category of construction with much improvement in accuracy and speed.

References

[1] Zhihua D., and Feng L., Roshan U. W., "Reconstruction of Large Phylogenetic Trees: A Parallel Approach", Electronic Version, 2005.

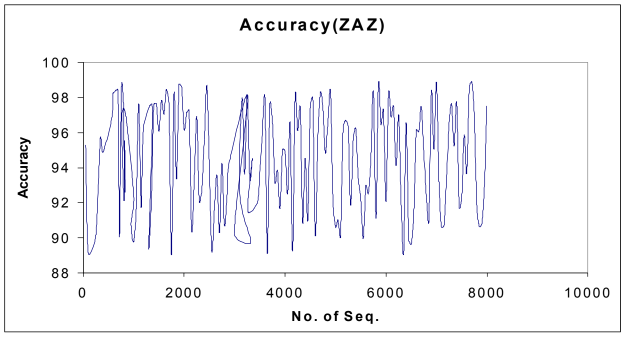
[2] Roshan U., Moret B., Williams T., Warnow T., "Rec-I-DCM3: A Fast Algorithmic Technique for Reconstructing Large Phylogenetic Trees", *Proceedings of the IEEE Computational Systems Bioinformatcis Conference, 2004.*

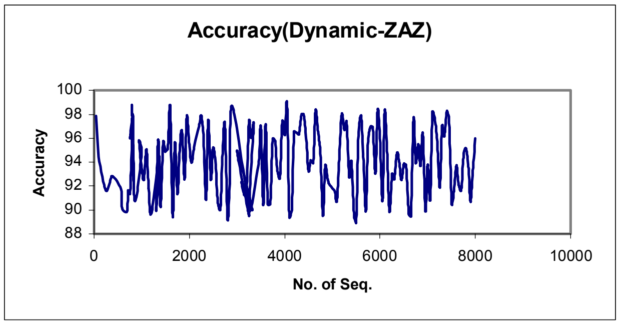
[3] Jones N., Pevzner P., *AN INTRODUCTION TO BIOINFORMATICS ALGORITHMS*, MIT Press, Cambridge, Massachusetts, London, England, 2004.

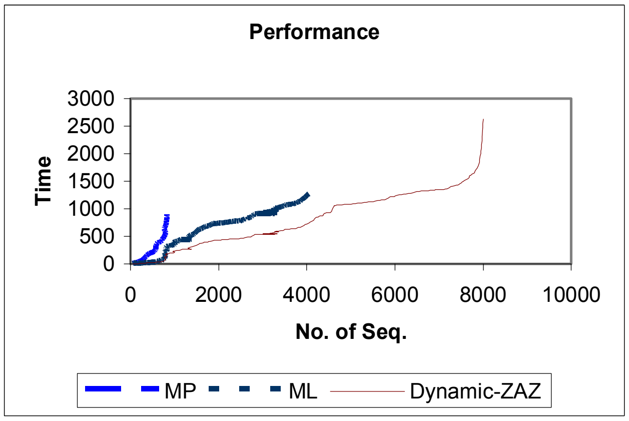
[4] Ranwez, V. and Gascuel, O., "Improvement of distance-based phylogenetic methods by a local maximum likelihood approach using triplets*", Mol. Biol. Evol., 19, 1952–1963, 2002*.

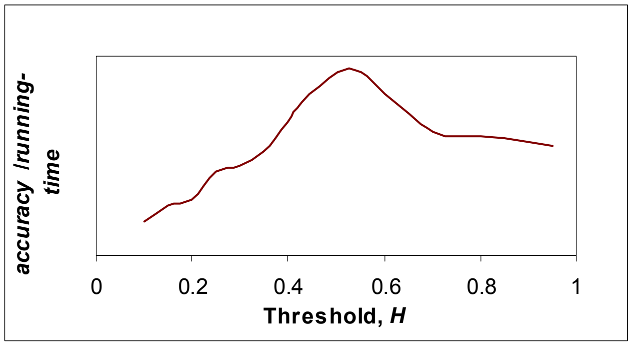
[5] Guindon, S. and Gascuel, O., "A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood" *Syst. Biol., 52, 696–704, 2003.*

Figures


Accuracy(ZAZ)
Accuracy
No. of Seq.


Accuracy(Dynamic-ZAZ)
Accuracy
No. of Seq.


Performance
Time
No. of Seq.
MP
ML
Dynamic-ZAZ


accuracy /running-time
Threshold, H