

Phylogenetic Predictions on Grids

Priyank Raj Katariya, Sathish S Vadhiyar
*Supercomputer Education and Research Centre
 Indian Institute of Science, Bangalore, India*
priyank@rishi.serc.iisc.ernet.in, vss@serc.iisc.ernet.in

Abstract—A phylogenetic or evolutionary tree is constructed from a set of species or DNA sequences and depicts the relatedness between the sequences. Predictions of future sequences in a phylogenetic tree are important for a variety of applications including drug discovery, pharmaceutical research and disease control. In this work, we predict future DNA sequences in a phylogenetic tree using *cellular automata*. Cellular automata are used for modeling neighbor-dependent mutations from an ancestor to a progeny in a branch of the phylogenetic tree. Since the number of possible ways of transformations from an ancestor to a progeny is huge, we use computational grids and middleware techniques to explore the large number of cellular automata rules used for the mutations. We use the popular and recurring neighbor-based transitions or mutations to predict the progeny sequences in the phylogenetic tree. We performed predictions for three types of sequences, namely, *triose phosphate isomerase*, *pyruvate kinase*, and *polyketide synthase* sequences, by obtaining cellular automata rules on a grid consisting of 29 machines in 4 clusters located in 4 countries, and compared the predictions of the sequences using our method with predictions by random methods. We found that in all cases, our method gave about 40% better predictions than the random methods.

Keywords—phylogeny; DNA sequences; predictions; Grids; master-worker;

I. INTRODUCTION

Study of the evolution of different species or organisms is important to biologists since it has practical applications including drug discovery, disease control, and population management [1]. Availability of DNA sequence databases [2], [3] in the last few decades has enabled the study of evolutions at the molecular level. During evolution, a DNA segment consisting of a sequence of purines and pyrimidines (also called base-pairs) changes to a different sequence of base-pairs. *Phylogenetic trees* give a picture of relatedness between various DNA sequences. A branch in a *rooted* phylogenetic tree connecting ancestor and progeny indicates that one sequence (progeny) has evolved from the other (ancestor).

Predictions of future sequences in a phylogenetic tree are important for a variety of applications including designing drugs to combat viruses, developing methodologies to apply corrections to evolutionary paths, and deploying mechanisms to accommodate future sequences. Existing efforts on phylogenetic inference are primarily concerned in the construction of trees from a given set of sequences [4]–[7]. In this paper, we predict future DNA sequences of phylogenetic trees based on modeling the evolutions of the given DNA sequences in the trees. We

predict future sequences by using cellular automata [8], [9] to model mutations of DNA sequences in a phylogenetic tree. We use cellular automata to model neighbor-based mutations of DNA sequences where each base pair in a DNA sequence represents a cell. We represent mutations of base pairs of a DNA sequence in terms of a cellular automata rule that changes the base-pair at a position of a DNA sequence based on the base-pair values at the position and its neighboring positions in the sequence.

Modeling DNA sequence mutations or transformations using cellular automata, where each cell can assume one of four possible states corresponding to four different base-pairs of a DNA sequence, requires exploration of a huge search space and needs a large number of computing cycles. To explore the large number of transformations in a reasonable amount of time, we have developed a computational grid system to conduct simultaneous modeling of different transformations on different ancestor-progeny branches of the phylogenetic tree. After obtaining the cellular automata rules for different time steps in different transformations of different branches, we calculate statistics regarding popular neighbor-dependent mutations for each time step of each branch. We use these statistics on an evolutionary path of the tree to predict the future sequences in the path. We validated our predictions by predicting the sequences of a phylogenetic tree and comparing the predicted and actual sequences of the tree. We performed predictions for three sequences, namely, *triose phosphate isomerase*, *pyruvate kinase*, and *polyketide synthase* sequences, by obtaining cellular automata rules on a grid consisting of 29 machines in 4 clusters located in 4 countries, and compared the predictions of the sequences using our method with predictions by random methods. We found that in all cases, our method gave about 40% better predictions than the random methods.

The rest of the paper is organized as follows. Section II gives a brief background. Section III describes our methodology for obtaining cellular automata rules for neighbor-dependent mutations on different branches of a phylogenetic tree. In Section IV, we describe our methodology of obtaining statistics regarding popular neighbor-dependent transitions and using the statistics to predict future sequences. Section V presents our experiments and results on grid resources. In Section VI, existing efforts on phylogenetic inference are discussed. Section VII presents conclusions and future work.

II. BACKGROUND

In this section, we give brief descriptions of phylogenetic trees, cellular automata and the relationship between cellular automata and DNA evolutions and other necessary background including *PSSM* matrix *molecular clock assumption*.

A. DNA Sequences

DNA is a nucleic acid that contains the genetic instructions for the development and function of living things. The building blocks of the DNA polymer are *nucleotides*, which in turn consist of a phosphate group, a sugar ring group and either a *purine* or a *pyrimidine* base group. Two possible purines are guanine (G) and adenine (A) and the two possible pyrimidines are thymine (T) and cytosine (C). These building blocks of DNA are also known as base pairs of DNA. We can view a DNA strand as a line of cells with each cell having one of the four values (A,G,C or T). Three base pairs in a DNA sequence form one codon. Each codon corresponds to either one of the 20 amino acids or to a control codon (start codon or end codon). A chain of amino acids form a protein which are the basic functional blocks of the organisms. During mutations, some base pairs in a DNA strand change giving rise to a different strand.

B. Phylogenetic Trees

A phylogenetic tree, also called an *evolutionary tree*, is a tree showing the evolutionary interrelationships among various species that are believed to have a common ancestor. The leaves of the tree represent various species or genomic sequences. An internal node of the tree represents an abstract sequence whose existence is presumed. A branch in a phylogenetic tree connecting an ancestor and a progeny indicates that one sequence (progeny) is evolved from the other (ancestor). We use Phylip [10] to construct phylogenetic trees. DNA sequences were downloaded from sequence database at NCBI [3]. The sequences were aligned by ClustalW web interface [11]. The aligned sequences were then input to 'dnamlk' program of Phylip to obtain phylogenetic tree for a given set of sequences. The Phylip output file also contained the lengths of the branches, intermediate sequences and other debugging information.

C. Cellular Automata

A *cellular automaton* is a regular array of identical finite state automata where the next state of an array element is determined solely by its current state and the state of its neighbors. The change of state of the array element is defined by a cellular automata rule [12]. Cellular automata are powerful tools for analyzing DNA mutations. An example of the evolution of one dimensional cellular automata is shown in Figure 1. Each cell can have one of the two possible states - 0 or 1. The neighborhood size is 1, i.e. the state of each cell at the next time step is dependent on the state of that cell, the state of its single left neighbor and the state of its single right neighbor. The

Time steps	Cells						
0	0	1	0	0	0	1	1
1	1	1	1	1	0	0	1
2	0	0	0	1	1	0	0

Figure 1. Evolution of Cellular Automata through time steps

0	0	0	0	0	1	0	1	0	0	1	1
1				0			1			0	
1	0	0	1	0	1	1	1	0	1	1	1
1			1			1				0	

Figure 2. Rule that governs the evolution of cellular automata shown in Figure 1

rule that governs this evolution is depicted in Figure 2¹. To determine the state of the 3rd cell at time step 1, a triplet, 100, is formed using the values of the left neighbor, the 3rd cell and the right neighbor, respectively, in time step 0. This triplet is looked up in the rule given by Figure 2, and the value of the cell is changed to 1 in time step 1. In general, one dimensional cellular automata with P states have $P^{2 \cdot n + 1}$ transitions for a rule and the total number of possible rules are $P^{P^{2 \cdot n + 1}}$.

D. Cellular Automata and Neighbor-based Mutations

There are strong indications that mutations of DNA base-pairs are affected by neighboring base-pairs [13]–[15]. We make an attempt to find out this relationship by modeling DNA as cellular automata where the DNA mutations are governed by the cellular automata rules. The DNA molecule can be viewed as a one-dimensional cellular automata, with four states per cell, corresponding to each of the four base-pairs. Thus, the base of this cellular automata is 4. In this work, we consider only those rules with neighborhood size of 1, i.e. the transition of a base-pair in a DNA sequence during evolution depends on the base-pair and its left and right neighboring base-pairs. Each rule consists of 64 transitions where the left-hand side of a transition contains three base pairs corresponding to a base pair at a position and its left and right neighbors. The right-hand side of each transition can be one of the 4 base-pairs giving rise to 4^{64} rules.

E. Position Specific Scoring Matrix (PSSM)

A Position Specific Scoring Matrix (PSSM) enables the scoring of multiple alignments with sequences. A PSSM is calculated over a set of sequences. It contains five columns corresponding to four base pairs and one for uncertainty base and number of rows equal to the number of bases in the DNA sequences or strands. Entry at row i and column j corresponds to the probability of occurrence of the base corresponding to column j at position i of a strand.

F. Molecular Clock

Molecular clock [16] is an assumption that gives the relation between lengths of the branches in a phylogenetic

¹In our work, we use cellular automata with wrap-around, i.e. the left neighbor of the first cell is the last cell, and the right neighbor of the last cell is the first cell.

tree and the rate of evolution or the time steps taken for the evolutions in the branches, as shown in Equation 1.

$$b_1 = \alpha_1 \cdot t_1; b_2 = \alpha_2 \cdot t_2; \dots; b_n = \alpha_n \cdot t_n \quad (1)$$

In the equation, $b_1, b_2, b_3, \dots, b_n$ are branch-lengths of the n branches in the phylogenetic tree and $t_1, t_2, t_3, \dots, t_n$ are the time steps taken for transformations of the corresponding branches. Branch length is a measure of the difference between the ancestor and progeny of a branch and is obtained along with the phylogenetic tree from the Phylip package. $\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_n$ denote the rates of mutations. According to linear molecular clock assumption, $\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_n$ are related such that

$$b_1 = \alpha_1 \cdot t_1 < b_2 = \alpha_2 \cdot t_2 < \dots < b_n = \alpha_n \cdot t_n \quad (2)$$

and

$$t_1 < t_2 < t_3 < \dots < t_n \quad (3)$$

This assumption is reasonable since greater the branch lengths or greater the difference between the ancestor and progeny sequences, the more the time steps required to transform from an ancestor to a progeny. In our work, we use a modification of a linear molecular clock assumption.

III. SEQUENCE TRANSFORMATIONS FOR A PHYLOGENETIC TREE

In this section, we first describe our algorithm for transformation of an ancestor to progeny sequence of a branch of a phylogenetic tree and formation of cellular automata rules for neighbor-dependent mutations. We also explain the master-worker grid computing architecture, involving distributed grid resources, for exploring the large number of cellular automata rules for the branches.

A. Sequence Transformation on a Branch

Our first step is to derive a sequence transformation methodology that, given an ancestor and a progeny sequence, starts with the ancestor sequence, applies a set of rules for neighbor-based mutations for a certain number of mutation steps to arrive at the progeny sequence. We apply the methodology for each branch for transforming the ancestor to progeny sequence of the branch. We compare a sequence produced during the transformation, with the progeny sequence using a similarity metric defined as the percentage of the number of base pairs in the sequence matching the corresponding base pairs in the progeny sequence. We have developed a program called *sequence transformer* that performs the transformations on a branch. Each sequence in a phylogenetic tree contains a series of *valid regions* where each valid region starts with a *start codon* and ends with a *stop codon*. For sequence transformation on a branch involving an ancestor and a progeny, the sequence transformer first finds the valid regions in the ancestor and progeny sequences, finds the intersection of positions of the valid regions of the two sequences, and forms the *working region* containing the intersection positions. The algorithm tries to transform the working region of the ancestor to the working region of the progeny.

1) *Sequence Transformation Algorithm*: The sequence transformer starts with an ancestor sequence as the current sequence and mutates the base pairs for a certain number of time steps until the current sequence transforms to the progeny sequence. For each time step, the algorithm traverses the current sequence starting from a random position and considers mutation of base pair in each position until it loops back to the starting position. The algorithm builds a *rule table* for each time step containing the neighbor-based transitions or mutations of a base pair with a neighborhood size of 1. The rule table contains a maximum of 64 entries corresponding to the 64 possible left hand sides of the transitions where the left-hand side of each transition contains three base pairs corresponding to a base pair at a position and its left and right neighbors.

The sequence transformer uses a Position Specific Scoring Matrix (PSSM) that is specific for a branch. For each position, the algorithm uses the probabilities in the PSSM for the position to decide if the base pair at the position should be mutated or preserved. The algorithm also uses the PSSM to determine the specific neighbor-dependent mutations for the base pairs that are mutated. To decide if a base pair in a position of the current sequence should be mutated, the algorithm compares the base pair with the base pair in the corresponding position of the progeny sequence. If the base pairs are different, the base pair in the position of the current sequence is mutated. If the two base pairs are equal, the algorithm uses the probabilities for all four possible base pairs for the position from the PSSM to form a Roulette wheel of four sectors. Each sector corresponds to a base pair and the length of a sector is proportional to the probability of the corresponding base pair at the position as given in the PSSM. The algorithm then generates a random number, determines the sector of the Roulette wheel containing the random number, and selects the associated base pair. If the selected base pair is the same as the base pairs in the position of the current and the progeny sequence, the algorithm decides to not mutate the base pair in the position of the current sequence and preserves the base pair.

If the algorithm decides to mutate the base pair in the current position, it forms a window of three base pairs containing the base pair in the position and its immediate left and right neighboring base pairs. To determine the specific mutation for the base pair, the algorithm looks up the rule table to find the transition containing the three base pairs in the window in the left hand side. If the right hand side of the transition is empty, the algorithm uses the PSSM matrix to form a Roulette wheel of four sectors corresponding to the position, generates a random number, and selects the base pair corresponding to the sector containing the random number. The algorithm mutates the base pair in the position to the selected base pair and also fills the right hand side of the transition in the rule table with the selected base pair. If the right hand side of the transition in the rule table contains a base pair, the algorithm mutates the base pair in the current position to the base pair in the rule table. The process is repeated for

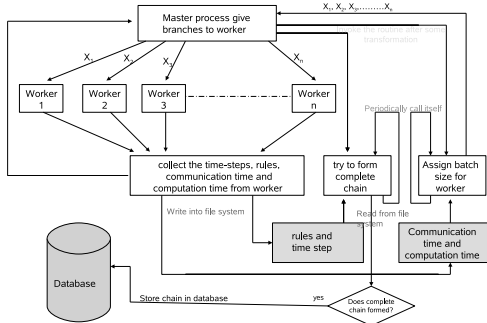


Figure 3. Master-Worker Paradigm

the next position by sliding the window by one position and continued until positions are considered.

2) *Determining PSSM for a Branch*: To determine the PSSM for a branch for formation of cellular automata rules, a set of sequences will have to be considered. In our approach, we automatically divide the phylogenetic tree into different regions or groups of similar sequences using K-Means clustering algorithm [17]. We then form a *group PSSM* for each group of sequences. For formation of cellular automata rules for a branch, our sequence transformer uses the group PSSM of the group containing the progeny sequence of the branch.

B. Multiple Sequence Transformations on Grids

Since our sequence transformer involves randomness in generation of cellular automata rules, each invocation of the sequence transformer for a given branch can give different transformations corresponding to different number of time steps and different sets of rules. The number of such transformations from an ancestor to a progeny sequence can thus be large due to exponential number of possible rules that can be applied at a single time step and the number of time steps. To explore the large number of cellular automata rules for different transformations of a branch and for different branches, we use grid computing middleware and resources.

1) *Master-Worker Paradigm*: A master-worker paradigm is used for invocations of the sequence transformer, formation of cellular automata rules, and insertions of rules into a database. The paradigm is illustrated in Figure 3. The master is responsible for assigning branches to the workers and collecting results from them when they complete their calculations. When a worker completes its calculations, the master is notified of the completion and the worker sends the results back to the master. The master stores the results from the workers into linked lists corresponding to the branches assigned to the workers. The master also periodically calls a *complete chain formation algorithm* to select a subset of neighbor-dependent mutations based on an evolutionary property. The property and the algorithm are described in the next subsection.

A worker takes a branch from the master, forms the working region of the ancestor and progeny sequences in

the branch, fills the unknown base-pairs randomly and invokes the sequence transformer. When the transformation is complete, it sends the results back to the master. The results mainly consist of the number of time steps required for transformations and the rule tables for the time steps containing the neighbor-dependent mutations.

2) *Piecewise Linear Molecular Clock Assumption and Selection of Cellular Automata Rules*: The master process at any point of time has a linked list of outputs from the worker process for each branch. Each node in the linked list corresponds to one invocation of the sequence transformer by a worker for a branch and contains the number of time steps taken for mutations as one of the values. The master process selects only a subset of the nodes in the linked lists such that the time steps corresponding to the nodes satisfy certain evolutionary property and inserts the parameters of the selected nodes in a database. The particular evolutionary property is the assumption of molecular clock related to the rate of evolution.

Let b_1, b_2, \dots, b_n , be the branch lengths of the branches arranged in ascending order and let t_1, t_2, \dots, t_n be the corresponding time steps. According to linear molecular clock assumption, the time steps are related as:

$$t_1 < t_2 < t_3 < \dots < t_n \quad (4)$$

For computation feasibility, we use piecewise linear molecular clock assumption in which two time steps, t_i and t_{i+1} corresponding to branch lengths, b_i and b_{i+1} , are related as:

$$t_{i-1} \leq t_i \leq t_{i-1} \cdot (1 + thres_1) \quad (5)$$

or

$$(1 - thres_2) \cdot t_{i-1} \leq t_i \leq t_{i-1} \quad (6)$$

For our work, we use values of 0.1 and 0.5 for the threshold factors, $thres_1$ and $thres_2$, respectively, shown in Equations 5 and 6, respectively. For a given snapshot of time step values for the branches, our complete chain formation algorithm first tries to satisfy the upper limit condition in Equation 5 for a branch. If it is not able to satisfy the condition, it next tries to satisfy the lower limit condition in Equation 6.

The master process periodically invokes a greedy algorithm, called *complete chain formation* algorithm, with the available snapshot of linked list values, to select only those outputs of sequence transformers that adhere to the piecewise linear molecular clock assumption. The greedy algorithm starts with branches in the phylogenetic tree sorted by their branch lengths. The algorithm finds a node, *prevNode*, in the first linked list having the smallest number of time steps. The *prevNode* is inserted in a *chain*. The algorithm then considers the next linked list and first tries to find a node with the smallest time step value greater than the time step value of *prevNode* such that the absolute difference between the time step values is less than the upper limit threshold. If such a node cannot be found, the algorithm tries to find a node with

greatest time step value smaller than the time step value of *prevNode* such that the absolute difference between the time step values is less than the lower limit threshold. This node now becomes the *prevNode* and is added to the *chain*. The entire procedure is then repeated for all linked lists corresponding to all the branches. A chain containing all the branches in the phylogenetic tree is called *complete chain*. The upper limit threshold is fixed to maximize the chance of finding a complete chain. The lower threshold is used to prevent large deviation from linear molecular clock assumption. If a complete chain can be formed from the current set of linked list values, the master forks another process to insert the parameter values corresponding to the complete chain into a PostgreSQL [18] database. The master also forms additional complete chains from the same set of linked list values by deleting some nodes from the original complete chain and invoking the greedy algorithm, and inserts the complete chains into the database.

3) *Load Balancing*: The master employs round-robin scheduling of tasks to the worker resources. Since the processor speeds and network links in grids can be heterogeneous, the speed of communications between the master and the workers on the distributed resources and the amount of work done by the workers per unit time can vary by large amounts. Hence, the round-robin scheduling can lead to load imbalance among the workers due to heterogeneity and load dynamics on grid resources. We used load balancing techniques to adapt to the grid resource and application dynamics. The master dynamically decides the number of branches or *batch size* to be allocated to a worker at a single step based on the times taken for communications with the worker and computations performed in the worker. To balance the load among the workers, the master periodically monitors the times taken for communication of inputs to and outputs from each worker and for computation performed by the worker for a single branch or *batch size* of 1. It then calculates the communication-to-computation ratio, $ratio_i$, for each worker and finds the minimum of the ratios of the workers, $ratio_{min}$. For each worker i , the master then finds the number of branches to be allocated to worker i or *batch size* for worker i , $batch_i$, at a given iteration as $batch_i = \frac{ratio_i}{ratio_{min}}$. Thus, workers with larger communication-to-computation ratios will be allocated more branches so that the high network latency spent for communicating the branches is amortized by the computations for the branches.

IV. PREDICTIONS OF SEQUENCES

For predictions of sequences in the phylogenetic tree, we use the neighbor-dependent transitions contained in the *rule tables* for the different branches that are stored in the database by the master process. To predict a progeny sequence from an ancestor sequence, we use the rule tables of all the branches in the path from the root to the ancestor of the branch. We denote this path as *history path*. The number of time steps required for obtaining the progeny

from the ancestor is predicted as the average of the time steps of the branches in the history path. To determine the number of time steps in each branch of the history path, we obtain the different time steps for the branch from different complete chains inserted in the database by the master process and calculate the average of the time steps across all the complete chains.

For prediction of the progeny sequence, our prediction algorithm starts with the ancestor sequence, and performs mutations for the predicted number of time steps to obtain the progeny sequence. For a particular time step, the prediction algorithm collects all transitions contained in the rule tables of the branches in the history path for the time step to form a *transition collection*. In each transition, the left-hand side consists of three base pairs, and the right-hand side consists of a base pair or an empty value. A transition with an empty value represents the preservation of the middle base pair in the left-hand side of the transition. For each position in the current sequence at a given time step, the prediction algorithm follows the procedure similar to the sequence transformer procedure described in Section III to decide if the base pair at the position should be mutated or preserved. The prediction algorithm first forms a PSSM using the sequences in the history path, then forms a Roulette wheel for the position from the probabilities for the position in the PSSM, generates a random number, and selects the base pair associated with the sector of the Roulette wheel corresponding to the random number. If the selected base pair is the same as the base pair in the current position, the algorithm decides to not mutate the base pair in the current position and preserves the base pair. If the selected base pair is different, the algorithm decides to mutate the base pair in the current position. This method of preservation of base pairs in some positions of the current sequence is denoted as *explicit preservation*.

If the prediction algorithm decides to mutate the base pair in the current position at the current time step, it forms a sliding window of three base pairs containing the base pair in the position and the base pairs in the left and right neighboring base pairs similar to the procedure in the sequence transformer. The prediction algorithm then calculates the total number of occurrences of those transitions in the *transition collection* for the current time step where the left-hand sides of the transitions consist of the three base pairs in the sliding window. The algorithm then forms a Roulette wheel consisting of five sectors based on the number of occurrences of five distinct transitions. Each sector in the Roulette wheel represents a transition with the left-hand side equal to the three base pairs in the sliding window and the right-hand side equal to one of the four base pairs or an empty value. The size of the sector is proportional to the number of occurrences of the corresponding transition in the *transition collection*. The prediction algorithm then generates a random number between 0 and the total number of occurrences of all transitions, locates the sector containing the random number, and selects the right-hand

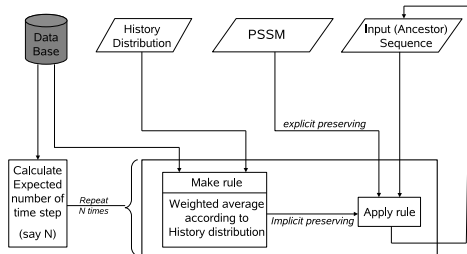


Figure 4. Prediction Model

side of the transition corresponding to the sector. If the selected right-hand side is a base pair, the prediction algorithm mutates the base pair in the current position to the selected base pair. If the selected right-hand is an empty value, the base pair in the current position is not mutated and is preserved. This method of preservation of base pairs is denoted as *implicit preservation*. Thus, popular transitions used in the history path with large number of occurrences have high probability of selection and usage during predictions of sequences. The overall prediction methodology is illustrated in Figure 4.

A. Validation of Predictions

1) *Basic Criteria*: For a prediction of progeny sequence from an ancestor sequence, X , we obtain both the actual progeny sequence, Y , and the predicted progeny sequence by our method, Y' . We then calculate the similarity of ancestor and actual progeny sequences, $similarity(X, Y)$ or *initial similarity*, and actual and predicted progeny sequences, $similarity(Y', Y)$ or *CA similarity*. The basic criteria that our prediction method should satisfy is that $similarity(Y', Y) > similarity(X, Y)$.

2) *Complete Random Prediction Strategy*: We compare our prediction method with a completely random prediction method in which random mutations are applied and random positions are preserved for the predicted number of time steps.

3) *Improved Random Prediction Strategy*: We also compare our prediction method with an improved random prediction method that also applies random mutations. But the number of time steps is estimated based on the lengths of the branches in the history path and using the linear molecular clock assumption. Also, a base pair in a position at a given time step is preserved with a probability $Q = 1 - P$. P represents the probability of mutation of a base pair at a position and is calculated as $P = (1/L)^{(1/t)}$, where L is the length of the sequence and t is the number of time steps.

V. EXPERIMENTS AND RESULTS

In this section, we present the results of our predictions on three different phylogenetic trees corresponding to three different types of sequences, namely, *triose phosphate isomerase*, *pyruvate kinase*, and *polyketide synthase*, using a grid consisting of machines from four countries. The sequences were downloaded from the NCBI database [3] and were aligned using ClustalW web interface [11].

Table I
SEQUENCES AND THEIR PROPERTIES

Sequence name	Sequence length	Number of sequences	Days of Execution	Number of Complete Chains
Triose phosphate isomerase	12248	100	14	200
Pyruvate kinase	4634	300	14	260
Polyketide synthase	3626	115	10	202

Table II
THE GRID INFRASTRUCTURE

Cluster and Location	Number of machines	Specifications
Battlecat cluster, University of Tennessee (UT), USA	8	GNU/Linux 2.6.25, Intel Core 2 Duo 2.13 GHz, 1 Gbps Ethernet
DAS-2, Vrije Universiteit, Netherlands	9	GNU/Linux 2.4.21, Dual PIII 996MHz, 100Mbps Fast Ethernet.
AMD cluster, Indian Institute of Science, India	8	AMD Opteron 246 based 2.21 GHz servers, Fedora Core 4.0, Gigabit Ethernet
Cluster at Faculty of Engineering Kasetsart University, Thailand	4	GNU/Linux 2.6.8, Dual PIII 933 MHz, 100 Mbps Ethernet

For each of the aligned sequence types, a phylogenetic tree was constructed using the Phylip [10] package. The lengths of the aligned sequences, and the number of sequences in the binary phylogenetic tree for the sequence types, are given in Table I. We utilized a grid infrastructure, shown in Table II, consisting of 29 machines distributed in 4 countries for our experiments. The worker processes were executed on all the machines. The master process and the PostgreSQL database for storing the complete chains were started on one of the AMD machines in Indian Institute of Science. The number of days of runs in the grid infrastructure and the number of complete chains for the different types of sequences are also shown in Table I.

Figure 5 shows the similarities between the predicted and the actual progeny sequences using our prediction method (referred as CA method), complete random and improved random methods for different branch lengths of the branches. The figures also show the *initial similarities* between the ancestor and the progeny sequences for the branches. We find that for all cases, our prediction methodology gives much better predictions than random prediction methods. The average improvement in similarity by our method over other methods is 40%. For *pyruvate kinase* and *polyketide synthase* sequences, as shown in Figures 5(a) and 5(b), our prediction method meets the basic criteria where the similarity between our predicted and the actual progeny sequences is greater than the initial similarity between the ancestor and the actual progeny sequences. The average improvement in the similarity by our method over the initial similarity is 40% for *pyruvate kinase* and 9.67% for *polyketide synthase* sequences. This shows that our method is able to predict correctly the evolutionary paths towards the progeny sequence from the ancestor sequence. For some branches of the *pyruvate*

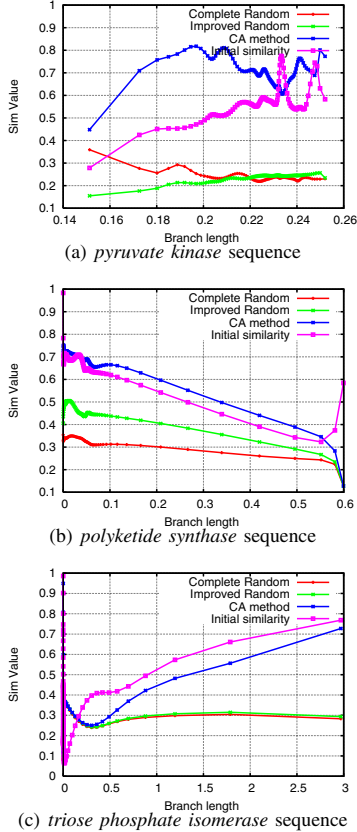


Figure 5. Similarity Values for different Branches

kinase sequence, the similarity of our predicted sequences is about 80% while the initial similarity between the ancestor and the progeny sequence is only 50%. Thus, our prediction method is able to cover 60% of the evolution path from the ancestor to the progeny sequence.

Figure 6 shows the comparison between the similarity values predicted by our sequences and the initial similarity for all branches for the sequences using scatter plots. The line in each graph represents the values corresponding to initial similarity. Points above the line satisfy the basic criteria where the similarity between our predicted sequences with the actual progeny sequences is greater than the similarity between the ancestor and the progeny sequences and points below the line do not satisfy the basic criteria. Figures 6(a) and 6(b) show that our prediction method correctly predicts the evolution path towards the progeny sequence and the similarities of the predicted sequences are greater than the initial similarities for about 83% of the branches in *pyruvate kinase* and *polyketide synthase* sequences. As Figures 5(c) and 6(c) show, the similarities of our predicted sequences are less than the initial similarities for large number of branches for *triose phosphate isomerase* sequences. The initial similarities of the branches of the phylogenetic trees for *triose phosphate isomerase* sequences are less than 0.5 in many cases and have large variations between 0.1 and 0.8. Our prediction

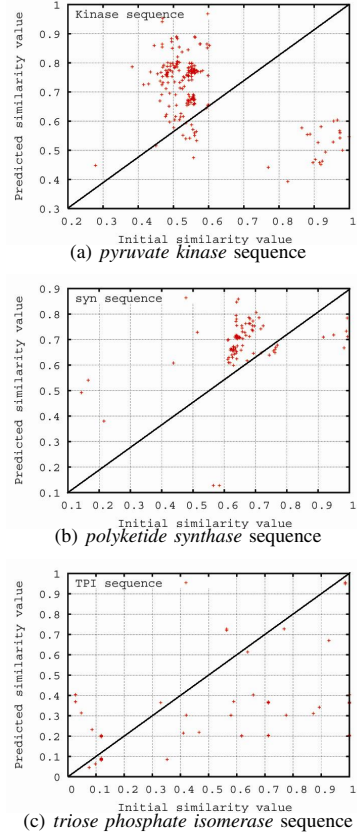


Figure 6. Predicted v/s Initial similarities

method that uses clustering to group similar sequences to determine PSSM and find popular neighbor-based mutations was not able to find similar sequences for grouping. Hence our method gave incorrect predictions for the *triose phosphate isomerase* sequences.

VI. RELATED WORK

There are a number of studies that analyze the impact of neighboring bases on the mutation of a particular base [13], [15]. None of these studies analyze the fine-grain effects of neighboring bases during each step of evolution. Siepel and Haussler [14] incorporate context-dependence in phylogenetic models to improve the quality of phylogenetic trees and estimate the pattern and rates of substitutions on the branches of a phylogenetic tree. Our work uses context-dependence to predict future sequences of the trees. DNA evolution has been modeled as cellular automata in the work by Sirakoulis et. al. [19]. However, the work considers limited cellular automata rules.

Various efforts have performed large-scale phylogenetic analysis using massively parallel processors [4]–[6]. PBPI [4] is a parallel Bayesian phylogenetic inference program that combines likelihood methods and Markov models and uses algorithmic improvements and parallel processing for high performance. Blagojevic et al. [5] ports and optimizes a high performance Maximum Likelihood (ML) program, RAXML, to the Cell Broadband Engine. The work by Ott

et al. [6] ports RAXML to IBM BlueGene/L architecture and performs phylogenetic analysis using both coarse and fine-grained parallelism. All these efforts construct phylogenetic trees for a given group of species. Our effort uses the phylogenetic trees to predict non-existent species.

Large scale phylogenetic tree analysis has also been performed on computational grids [7], [20]. The work by Stewart et. al. [7] had prepared a global grid for studying arthropod evolution. The effort implemented a parallel version of fastDNAmI [21] algorithm on a global grid using a maximum likelihood approach to construct better phylogenetic trees. The work by Joshi and Vadhiyar [20] studies the evolution of HIV sequences using the molecular clock assumption and cellular automata. Our work significantly extends their work to predict future sequences of phylogenetic trees.

VII. CONCLUSIONS AND FUTURE WORK

In this work, we have modeled DNA sequence mutations in the phylogenetic trees using cellular automata, obtained the cellular automata rules for neighborhood-based mutations on branches of the trees, and predicted the future sequences of the trees using the recurring and popular neighborhood-based mutations in the predecessor branches. We explored large number of cellular automata rules on distributed grid resources. We formed cellular automata rules for three types of sequences, namely, *triose phosphate isomerase*, *pyruvate kinase*, and *polyketide synthase*, by executing our paradigm on a grid consisting of 29 machines in 4 clusters located in 4 countries, and compared the predictions of the sequences using our method with predictions by random methods. We found that in all cases, our method gave about 40% better predictions than the random methods. We plan to augment our prediction methods to give good predictions for sequences with very small and very large initial similarities.

ACKNOWLEDGMENT

The authors would like to thank Prof. Jack Dongarra of University of Tennessee, Dr. Thilo Kielmann of Vrije Universiteit, Netherlands and Prof. Putchong Uthayopas and Thara Angskun of ThaiGrid project for providing access to the respective machines.

REFERENCES

- [1] "Understanding Evolution," <http://evolution.berkeley.edu>.
- [2] "HIV Sequence Database," <http://hiv.lanl.gov>.
- [3] "National Center for Biotechnology Information," www.ncbi.nlm.nih.gov/.
- [4] X. Feng, K. Cameron, and D. Buell, "PBPI: a High Performance Implementation of Bayesian Phylogenetic Inference," in *Proceedings of the ACM/IEEE SC 2006 Conference*, 2006, pp. 40–.
- [5] F. Blagojevic, A. Stamatakis, C. Antonopoulos, and D. Nikolopoulos, "RAXML-Cell: Parallel Phylogenetic Tree Inference on the Cell Broadband Engine," in *International Parallel and Distributed Processing Symposium*, 2007, p. 77.
- [6] M. Ott, J. Zola, A. Stamatakis, and S. Aluru, "Large-scale Maximum Likelihood-based Phylogenetic Analysis on the IBM BlueGene/L," in *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, 2007, pp. 1–11.
- [7] C. Stewart, D. Hart, M. Aumuller, R. Keller, M. Muller, H. Li, R. Repasky, R. Sheppard, D. Berry, M. Hess, U. Wossner, and J. Colbourne, "A Global Grid for Analysis of Arthropod Evolution," in *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing*, 2004, pp. 328–337.
- [8] K. Laurio, F. Linakera, and A. Narayanan, "Regular Biosequence Pattern Matching with Cellular Automata," *Information Sciences*, vol. 146, no. Issues 1-4, pp. 89–101, 2002.
- [9] D. Wishart, R. Yang, D. Arndt, P. Tang, and J. Cruz, "Dynamic Cellular Automata: an Alternative Approach to Cellular Simulation," *In Silico Biol.*, vol. 5, no. 2, pp. 139–161, 2005.
- [10] "Phylip Package," <http://evolution.genetics.washington.edu/phylip.html>.
- [11] "ClustalW," <http://www.ebi.ac.uk/clustalw>.
- [12] N. Ganguly, B. Sikdar, A. Deutsch, G. Canright, and P. Chaudhuri, "A Survey on Cellular Automata," Centre for High Performance Computing, Dresden University of Technology, Tech. Rep., December 2003.
- [13] M. Bulmer, "Neighboring Base Effects on Substitution Rates in Pseudogenes," *Molecular Biology and Evolution*, vol. 3, no. 4, pp. 322–329, 1986.
- [14] A. Siepel and D. Haussler, "Phylogenetic Estimation of Context-Dependent Substitution Rates by Maximum Likelihood," *Molecular Biology and Evolution*, vol. 21, no. 3, pp. 468–488, March 2004.
- [15] B. Morton, I. Bi, M. McMullen, and B. Gaut, "Variation in Mutation Dynamics Across the Maize Genome as a Function of Regional and Flanking Base Composition," *Genetics*, vol. 172, no. 1, pp. 569–577, January 2006.
- [16] M. Kimura and T. Ohta, "On the Rate of Molecular Evolution," *Journal of Molecular Evolution*, vol. 1, no. 1, pp. 1–17, 1997.
- [17] A. Jain, M. Murty, and P. Flynn, "Data Clustering: a Review," *ACM Comput. Surveys*, vol. 31, no. 3, pp. 264–323, 1999.
- [18] "PostgreSQL," <http://www.postgresql.org>.
- [19] G. Sirakoulis, I. Karafyllidis, C. Mizas, V. Mardiris, A. Thanailakis, and P. Tsalides, "A Cellular Automaton Model for the Study of DNA Sequence Evolution," *Computers in Biology and Medicine*, vol. 33, no. 5, pp. 439–453, September 2003.
- [20] Y. Joshi and S. Vadhiyar, "Analysis of DNA Sequence Transformations on Grids," *Journal of Parallel and Distributed Computing*, vol. 69, no. 1, pp. 80–90, 2009.
- [21] G. Olsen, H. Matsuda, R. Hagstrom, and R. Overbeek, "fastDNAmL: a Tool for Construction of Phylogenetic Trees of DNA Sequences using Maximum Likelihood," *Computer Applications in the Biosciences*, vol. 10, no. 1, pp. 41–48, 1994.