JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 10, Number 6, 2003

Pp. 803–819

An Eulerian Path Approach to Global Multiple Alignment for DNA Sequences

YU ZHANG[1] and MICHAEL S. WATERMAN[2]

ABSTRACT

With the rapid increase in the dataset of genome sequences, the multiple sequence alignment problem is increasingly important and frequently involves the alignment of a large number of sequences. Many heuristic algorithms have been proposed to improve the speed of computation and the quality of alignment. We introduce a novel approach that is fundamentally different from all currently available methods. Our motivation comes from the Eulerian method for fragment assembly in DNA sequencing that transforms all DNA fragments into a de Bruijn graph and then reduces sequence assembly to a Eulerian path problem. The paper focuses on global multiple alignment of DNA sequences, where entire sequences are aligned into one configuration. Our main result is an algorithm with almost linear computational speed with respect to the total size (number of letters) of sequences to be aligned. Five hundred simulated sequences (averaging 500 bases per sequence and as low as 70% pairwise identity) have been aligned within three minutes on a personal computer, and the quality of alignment is satisfactory. As a result, accurate and simultaneous alignment of thousands of long sequences within a reasonable amount of time becomes possible. Data from an Arabidopsis sequencing project is used to demonstrate the performance.

Key words: multiple sequence alignment, de Bruijn graph, Eulerian path.

1. INTRODUCTION

MULTIPLE SEQUENCE ALIGNMENT IS ROUTINELY USED to find conserved regions in molecular sequences. When a set of related sequences is aligned by multiple sequence alignment algorithms, the hidden commonalities among sequences are revealed. Thus they provide a critical computational tool to retrieve hidden information among huge and exponentially growing genetic data.

A mathematically optimal solution of the multiple sequence alignment problem is achieved by using a dynamic programming algorithm (Sankoff, 1975; Waterman *et al.*, 1976). Because the time and memory cost of the dynamic programming algorithm is exponential to the number of sequences, i.e., $\Theta(L^N)$ where L = sequence length and N = number of sequences, it is impossible to put this method into practical

[1]Department of Mathematics, University of Southern California, 1042 W. 36th Place (DRB 289), Los Angeles, CA 90089-1113.
[2]Department of Biological Sciences, University of Southern California, Los Angeles, CA 90089-1340.

use except on a small set of sequences (see Kececioglu [1993] for extensions of this approach). Many heuristic algorithms achieve reasonably good solutions with limited time and memory usage, and some of them have already been popularly used in modern molecular biology research.

Most currently available algorithms are in one of two classes. The first class is those algorithms using the progressive alignment strategy (Waterman and Perlwitz, 1984; Feng and Doolittle, 1987). A multiple alignment is gradually built up by aligning the closest pair of sequences first and then aligning the next closest pair of sequences, or one sequence with a set of aligned sequences or two sets of aligned sequences. This procedure is repeated until all given sequences are aligned together. The key idea is that the pair of sequences with minimum distance is most likely to having been obtained from the most recent evolutionary divergence and that the pairwise alignment of these two specific sequences provides the most "reliable" information that can be extracted. Many programs based on this method exist. Some of them construct an alignment throughout the entire length of sequences, such as MULTAL (Taylor, 1988), AMULT (Barton and Sternberg, 1987a, 1987b), MSA (Lipman *et al.*, 1989), CLUSTALW (Higgins and Sharp, 1989; Thompson *et al.*, 1994). Other programs try first to identify an ordered series of highly conserved regions, then proceed to align the intervening regions, such as GENALIGN (Martinez, 1988) and ASSEMBLE (Vingron and Argos, 1991). Among these progressive alignment methods, CLUSTALW, probably the best-known and most popular program used currently, builds a guide tree from the pairwise alignment scores and merges subsets of sequences according to the tree. A recent algorithm T-Coffee (Notredame *et al.*, 2000) is similar to CLUSTALW but uses a consistency measure that may reduce the potential errors caused by progressive alignment.

The other class of alignment algorithms uses iterative refinement strategies to improve an initial alignment (Sankoff *et al.*, 1976; Sankoff and Kruskal, 1983). The basic idea for iterative methods is to refine the initial alignment iteratively by local optimization. In HMMT (Eddy, 1995), e.g., the model parameters are reestimated at each iteration. Iteration continues until convergence or reaching the maximal user-defined number of iterations. Currently available programs that apply iterative strategies include DIALIGN (Morgenstein *et al.*, 1996), which uses a local alignment approach to construct multiple alignments based on segment–segment comparisons, where the segments are incorporated into a multiple alignment by an iterative process. The PRRP program (Gotoh, 1996) optimizes an alignment by iteratively dividing the sequences into two groups and then realigning them using a global group-to-group alignment algorithm. SAGA (Notredame and Higgins, 1996) uses a genetic algorithm to select from an evolving population the alignment which optimizes an objective function (OF). An OF named COFFEE (Notredame et al., 1998) is used in SAGA that measures the consistency between the multiple alignment and a library of CLUSTALW pairwise alignments. HMMT (Eddy, 1995) and SAM (Hughey and Krogh, 1996) apply a stochastic iterative strategy that maximizes the probability based on a hidden Markov model (HMM), but they are often used to refine prealigned sequences (Notredame, 2002).

The numerous alignment programs mentioned above have their own advantages and drawbacks. However, there are some common issues for all of the currently available alignment algorithms: 1) robustness under certain conditions, such as gap-rich regions, repeat-rich regions, etc.; 2) local optima problems, especially for iterative methods; 3) time efficiency and memory usage. The time cost for all current algorithms is at best proportional to the square of the number of sequences to be aligned.

Although many heuristic algorithms have been applied to alleviate the huge time and memory expenses, those costs remain a barrier for practical application when confronted with thousands of input sequences, or millions of letters in each sequence, which are not unusual numbers even in bacterial genomes. Certainly, handling hundreds of sequences with lengths in the thousands is a worthy goal.

2. MOTIVATION

This paper presents a novel approach to the global multiple DNA sequence alignment problem using Eulerian paths. We call the method "EulerAlign." The most significant advantages of EulerAlign are its linear time and memory cost with respect to the total size of sequences to be aligned and improved alignment accuracy.

Our motivation comes from the algorithm for fragment assembly in DNA sequencing using the Eulerian superpath approach (EULER) (Idury and Waterman, 1995; Pevzner *et al.*, 2001). In that algorithm, a

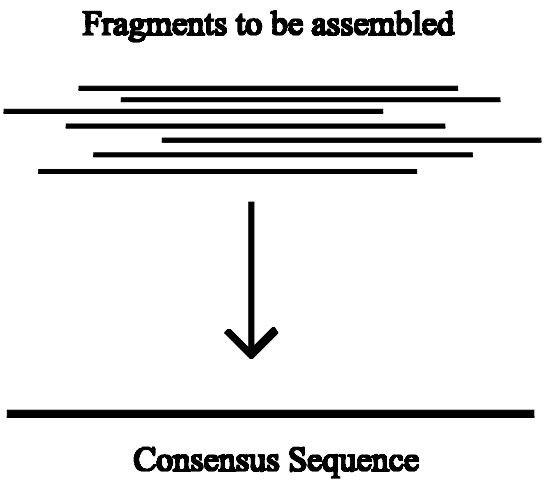

FIG. 1. An extreme version of fragment assembly is multiple sequence alignment.

fragment assembly problem is first reduced to an easy-to-solve Eulerian path problem in the de Bruijn graph, and an Eulerian path is found that corresponds to the consensus sequence of the genome. A significant contribution of the Eulerian approach is that it discards the traditional "overlap-layout-consensus" outline. In another words, it obtains a consensus sequence before knowing an alignment and without doing pairwise alignments.

A global multiple alignment problem is in fact an extreme version of the fragment assembly problem. As shown in Fig. 1, if almost all fragments come from the same region of a genome, then EULER should output a consensus sequence of that region. For multiple sequence alignment, if this consensus sequence is the closest one to all given sequences, one would expect to achieve an accurate alignment using a consensus scoring scheme.

Note that the idea is similar to the star method: given N sequences to be aligned, the star method first computes the alignments of all sequence pairs and picks one sequence among N sequences as the consensus that is closest to all other sequences. EulerAlign attempts to find such a consensus that consists of fragments of the N sequences such that the conserved regions are amplified and noise is suppressed. To understand this, assume all input sequences are derived from a common ancestral sequence; then EulerAlign is used to find this ancestral sequence. By comparing each sequence with the ancestral, it will be able to distinguish between conserved letters and mutations. Consequently, the global multiple alignment will be unambiguously constructed.

A crucial requirement of EULER is the almost "error-free" data that is obtained by an error-correction procedure. In addition, EULER doesn't evaluate the quality of its consensus path until the final quality assessment stage. For the multiple alignment problem, however, people are more interested in aligning distantly related sequences, and for EulerAlign, the consensus sequence obtained must be "accurate." One option is to do error-correction aggressively so all "errors" (differences between sequences) are eliminated, but this is not likely to succeed when aligning distantly related sequences. Instead, EulerAlign applies a somewhat different procedure than does EULER. EulerAlign transforms the initially tangled graph into a directed acyclic graph (DAG), and during the transformation it tries to retain k-tuples common to several sequences. In addition, the weight for each edge is assigned in such a way that the consensus sequence has the maximum weight. By doing these two steps, a consensus-finding problem is transformed into a heaviest path problem.

3. THE ALGORITHM

A brief outline of our approach is as follows: (1) Construct a directed de Bruijn graph using all k-tuples from the sequences to be aligned. (2) Transform the de Bruijn graph to a DAG. (3) Extract a

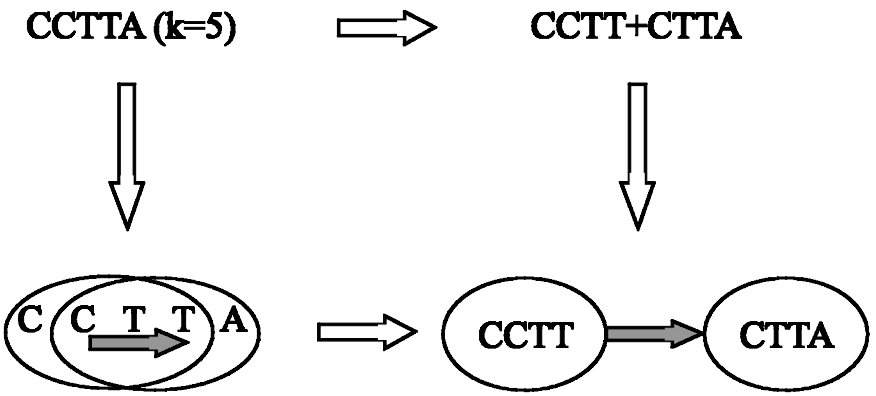

FIG. 2. A k-tuple represented by two vertices and one directed edge connecting them.

consensus path from the DAG according to the weights of edges. (4) Do fast pairwise alignment between the consensus path and each input sequence. (5) Construct the final multiple alignment according to the pairwise alignments. The central part of EulerAlign is the transformation from the constructed de Bruijn graph to a DAG that retains the commonalities among sequences in its edges and allows us to easily construct the consensus path.

We should emphasize that EULER combines all reads and their reverse complements into the graph since, in assembly, fragment orientations are unknown, whereas in our case the orientations are given.

3.1. Graph construction

A k-tuple is k consecutive letters in a sequence. Each k-tuple in given sequences can be transformed into a structure of two vertices connected by a directed edge, where the vertices are the $(k-1)$-tuples contained in the k-tuple and the edge represents the k-tuple itself with the direction from the prefix $(k-1)$-tuple to the suffix $(k-1)$-tuple. For example, a 5-tuple "CCTTA" contains two 4-tuples 'CCTT'+'CTTA' with overlapping letters "CTT." Each 4-tuple is a vertex and the directed edge connecting them represents the 5-tuple "CCTTA." This structure is the basic unit in our de Bruijn graph (Fig. 2).

Our de Bruijn graph is constructed by collecting all k-tuples among the sequences. Each k-tuple structure is added into the graph successively, and if the incoming k-tuple has vertices or an edge identical to the vertices or edges in the already existing graph, then they are merged together. Sequence information is stored in edges. An example is shown Fig. 3. Under this construction, each k-tuple from the sequences corresponds

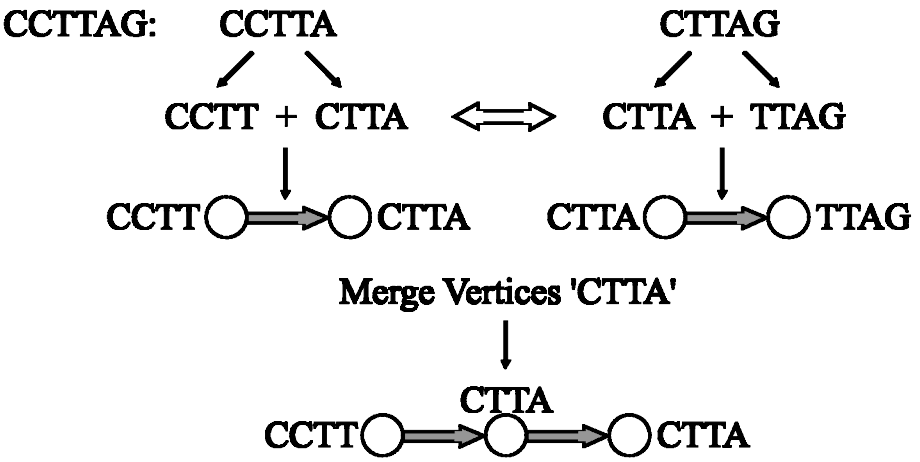

FIG. 3. Construction of the de Bruijn graph for CCTTAG and $k = 5$.

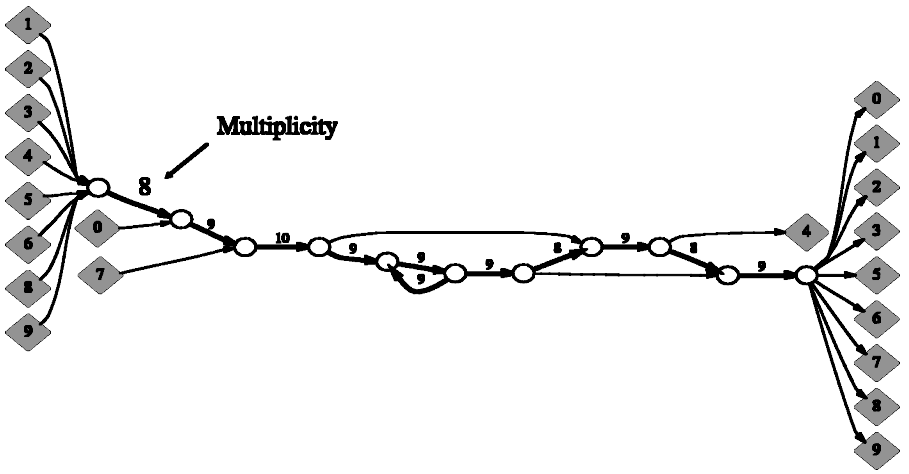

FIG. 4. An example of the initial de Bruijn graph. Branches and loops are due to the differences and repeats among sequences. All thick edges have multiplicities larger than 1. ($N = 10, L = 60bp, k = 6$).

to a unique edge in the graph, and each $(k-1)$-tuple corresponds to a unique vertex. Consequently, each input sequence is represented by a path through this graph.

In order to find the consensus path, which is not necessarily in one of the given sequences, we define the multiplicity of an edge to be the number of sequences containing this edge. Obviously, an edge (of fixed length) with high multiplicity is more likely to represent a consensus (Fig. 4). By defining multiplicity in this way, EulerAlign can find a reasonable consensus path and avoid being biased by repeats.

A consensus path is defined as an acyclic path through the graph. Each edge is visited at most once, and the weight of this path (according to the multiplicities and lengths of edges) is maximized. In another words, finding consensus is a heaviest path problem, which is identical to a shortest path problem with negative weights. A shortest path problem with negative weights can be solved in linear time when the graph is a DAG. Our heuristic is to transform the graph to a DAG such that we can find the consensus in linear time.

3.2. Transforming the graph to a DAG

Because of the random matches, repeats, and mutations in DNA sequences, the initial graph is extremely tangled. A tangle is defined to be a vertex that has more than one incoming or outgoing edge. As the number of incoming or outgoing edges in a tangle increases, so do the chances that the tangle will result in cycles. The goal is to eliminate as many cycles as possible while keeping all the similarity information among sequences untouched (represented by edges with high multiplicities). Note that only cycles are our targets, while tangles are not.

Claim: *Define a* left edge *for vertex* $\mathbf{v}_i$ *to be an edge that points to* $\mathbf{v}_i$*, denoted as* $\mathbf{E}_{\rightarrow \mathbf{v}_i}$*, and a* right edge *for vertex* $\mathbf{v}_i$ *to be an edge that starts from* $\mathbf{v}_i$ *and points to another vertex, denoted as* $\mathbf{E}_{\mathbf{v}_i \rightarrow}$*. If a vertex* $\mathbf{v}_i$ *has two or more left edges* $\{\mathbf{E}^n_{\rightarrow \mathbf{v}_i}\}_{n=1,2,\ldots}$ *that are contained in the same sequence path, then there must exist a cycle in the graph and* $\mathbf{v}_i$ *is a vertex on the cycle.*

Proof. If a path of a sequence contains two left edges $\mathbf{E}^1_{\rightarrow \mathbf{v}_i}$ and $\mathbf{E}^2_{\rightarrow \mathbf{v}_i}$ of $\mathbf{v}_i$, then when walking though this path, $\mathbf{v}_i$ will be visited when visiting $\mathbf{E}^1_{\rightarrow \mathbf{v}_i}$, and $\mathbf{v}_i$ will be visited again when visiting $\mathbf{E}^2_{\rightarrow \mathbf{v}_i}$. Since the path is well connected, it follows that this path contains a cycle with $\mathbf{v}_i$ as a cross vertex. Thus, the graph contains a cycle. ■

Note that the reverse of this claim is not true; i.e., after removing all cycles found by this procedure, the resulting graph is not necessarily a DAG. But a good point is that after removing all such cycles, in our data the acyclic consensus we found was satisfactory.

The basic rules of our transformation are similar to those of EULER (Pevzner *et al.*, 2001):

(1) If a cycle is found by the above procedure, i.e., the corresponding sequence path is of form "$\cdots \mathbf{E}^1_{\rightarrow \mathbf{v}_i} \mathbf{E}^1_{\mathbf{v}_i \rightarrow} \cdots \mathbf{E}^2_{\rightarrow \mathbf{v}_i} \mathbf{E}^2_{\mathbf{v}_i \rightarrow} \cdots$," and the right edges $\mathbf{E}^1_{\mathbf{v}_i \rightarrow}$ and $\mathbf{E}^2_{\mathbf{v}_i \rightarrow}$ are different, then we can make two superedges (an edge that represents two short continuous edges connected by a common vertex) $\mathbf{E}^1_{\rightarrow \mathbf{v}_i \rightarrow}$ and $\mathbf{E}^2_{\rightarrow \mathbf{v}_i \rightarrow}$ to replace $\mathbf{E}^1_{\rightarrow \mathbf{v}_i}$, $\mathbf{E}^1_{\mathbf{v}_i \rightarrow}$, $\mathbf{E}^2_{\rightarrow \mathbf{v}_i}$, $\mathbf{E}^2_{\mathbf{v}_i \rightarrow}$, such that the cycle is eliminated (Fig. 5a).

(2) In a situation similar to (1), but now $\mathbf{E}^1_{\mathbf{v}_i \rightarrow}$ and $\mathbf{E}^2_{\mathbf{v}_i \rightarrow}$ are the same, denoted as $\mathbf{E}_{\mathbf{v}_i \rightarrow}$, we again make two superedges $\mathbf{E}^1_{\rightarrow \mathbf{v}_i \rightarrow}$ and $\mathbf{E}^2_{\rightarrow \mathbf{v}_i \rightarrow}$ by separating $\mathbf{E}_{\mathbf{v}_i \rightarrow}$ such that the cycle is simplified but not eliminated (Fig. 5b). In this case, the sequence information stored in $\mathbf{E}_{\mathbf{v}_i \rightarrow}$ is partitioned into two superedges $\mathbf{E}^1_{\rightarrow \mathbf{v}_i \rightarrow}$ and $\mathbf{E}^2_{\rightarrow \mathbf{v}_i \rightarrow}$ according to which left edge of $\mathbf{v}_i$ they come from, and the multiplicities for superedges $\mathbf{E}^1_{\rightarrow \mathbf{v}_i \rightarrow}$ and $\mathbf{E}^2_{\rightarrow \mathbf{v}_i \rightarrow}$ are computed. This is distinct from EULER, because we know the entire sequences and in most cases it is easy to partition and assign sequences by looking at the corresponding left/right edges.

Because we are doing multiple sequence alignment, many other sequence paths may visit $\mathbf{v}_i$ from $\mathbf{E}^1_{\mathbf{v}_i \rightarrow}$, $\mathbf{E}^2_{\mathbf{v}_i \rightarrow}$ or other edges and leave $\mathbf{v}_i$ in different directions. Thus, one issue concerning our graph transformation is whether the transformations lead to the loss of similarity information and whether the order in which we perform transformations determines whether we will retain the maximum similarity information. A transformation is safe if the transformation does not introduce the loss of similarity information. An example is shown in Fig. 6a. Otherwise the transformation is unsafe, as shown in Fig. 6b. We conclude that under the safe transformation, the desired consensus path will be well maintained. As a result, the current version of EulerAlign attempts to remove all cycles by performing safe transformations first and leaves all unsafe transformations for later.

In Fig. 7 is a de Bruijn graph transformed from the original graph in Fig. 4. Note that a loop in the original graph is removed, and it is a DAG now. The thick path represents our heaviest consensus path.

Intuitively, the total number of possible paths in the graph is decreasing as transformations are performed, and thus we have fewer choices of paths. However, when safe transformations are performed first, the desired consensus path is well maintained, and the number of necessary unsafe transformations to obtain the desired graph is minimized. On the other hand, it is worth mentioning that safe transformations cannot eliminate all cycles. In some situations, no transformation is safe. But we will show by example that, in

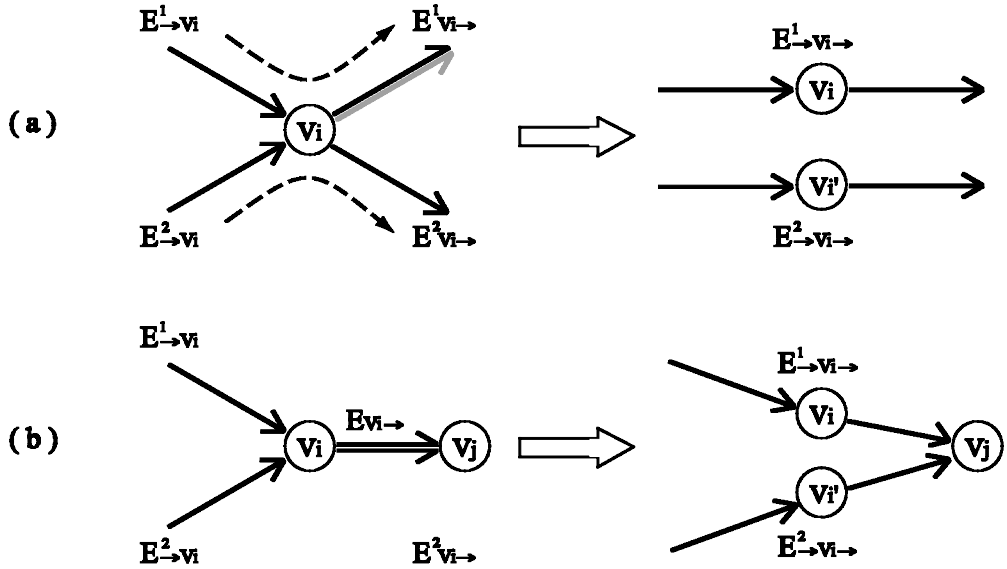

FIG. 5. (**a**) A tangle at $\mathbf{v}_i$ is eliminated by making a copy $\mathbf{v}'_i$ of vertex $\mathbf{v}_i$. (**b**) A tangle at $\mathbf{v}_i$ is eliminated by making a copy $\mathbf{v}'_i$ of vertex $\mathbf{v}_i$, and separating $\mathbf{E}_{\mathbf{v}_i \rightarrow}$. The cycle is not eliminated yet, due to vertex $\mathbf{v}_j$. When this step is repeated several times, the cycle will be finally eliminated.

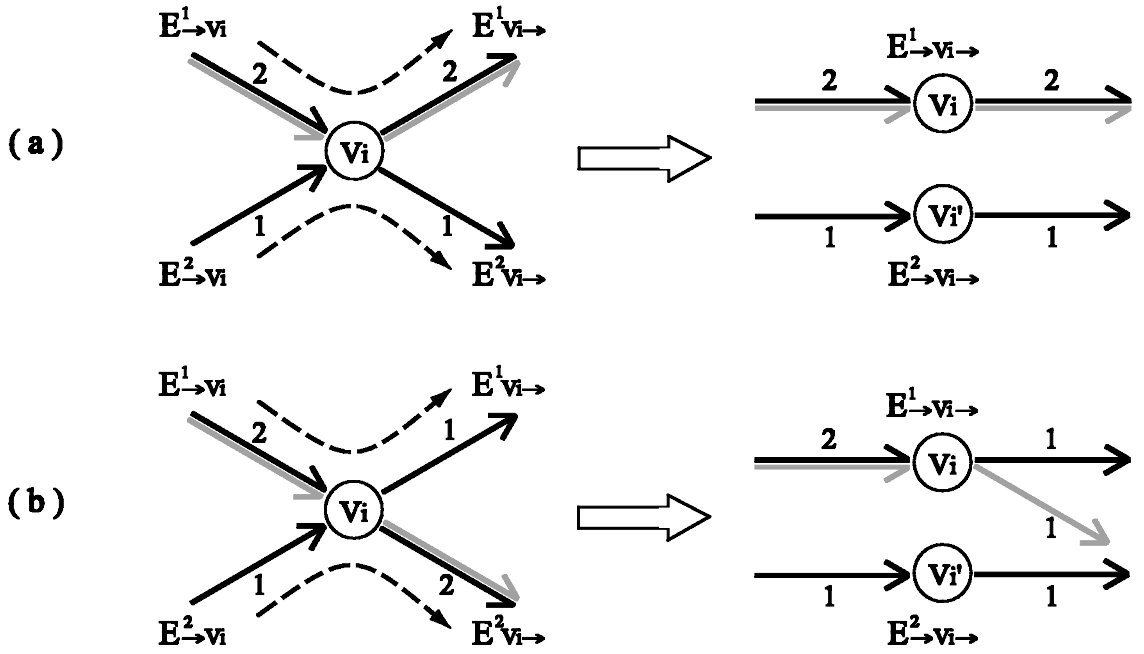

FIG. 6. Examples of safe/unsafe transformation. Dashed lines show the directions of sequence paths. Multiplicities are shown on each edge. (**a**) Different sequences (with different grey scales) share $\mathbf{E}^1_{\rightarrow \mathbf{v}_i}$ and $\mathbf{E}^1_{\mathbf{v}_i \rightarrow}$. After transformation, they remain unchanged. (**b**) After transformation, the similarity information in $\mathbf{E}^2_{\rightarrow \mathbf{v}_i}$ is lost. Thus it is an unsafe transformation.

our experience, the global multiple alignments constructed by EulerAlign are indeed more accurate than of other methods.

3.3. Consensus path and final alignment

After performing the above transformations, we apply a greedy algorithm to find a heaviest path within linear time. The weight for each edge is proportional to its multiplicity and length. Although the greedy algorithm does not guarantee optimal solutions, the results are satisfactory. Note that we can always refer back to the rigorous heaviest path algorithm by arbitrarily removing all edges that complete a cycle (other than the cycles defined by our claim) in linear time and thus guarantee an optimal solution for this version of the problem.

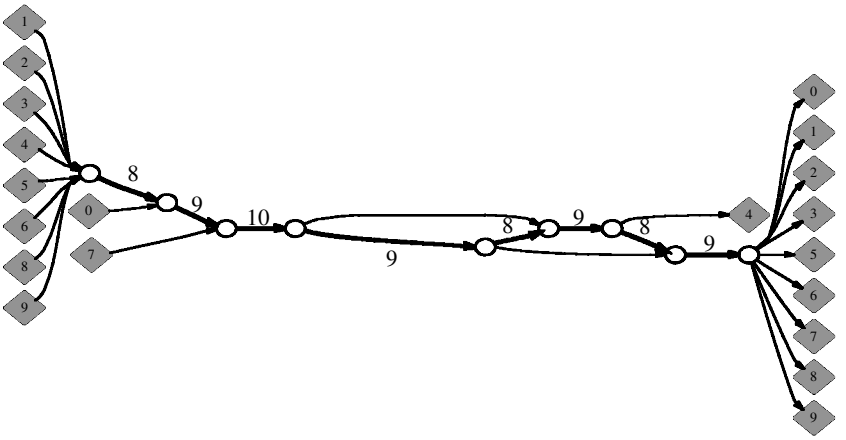

FIG. 7. A DAG achieved by doing safe transformation. The bold path is the desired consensus path, and multiplicities are shown on each edge.

Our next step after obtaining a consensus path is quite straightforward. EulerAlign uses a banded pairwise alignment algorithm (the positional shifts between two candidate letters in two sequences are bounded by a constant) to align the consensus sequence with each input sequence and then combines the alignments to construct the final global multiple alignment. As mentioned before, this alignment attempts to maximize the consensus score, and it is most appropriate for the case that all given sequences are derived from a common sequence. However, the algorithm performs well with sequences derived from an evolutionary process (see results section).

4. PROBABILITY ANALYSIS

Assume all input sequences are derived from a common ancestral sequence $\mathbf{S}_0$. We will show that when the number of input sequences increases, the consensus path found in the graph will be asymptotically "identical" to $\mathbf{S}_0$.

Let N denote the number of sequences to be aligned, L denote the average sequence length, k denote the size of the k-tuple used in construction of the graph, and β denote the mutation rate (by this we mean the probability of a substitution or indel in a sequence at a given position).

Recall that the consensus path is the heaviest path in the graph weighted by the multiplicity and the length for each edge. An edge weight is denoted as $\mathbf{W}(e)$. To be simple, define $\mathbf{W}(e)$ = multiplicity of the edge. Then, with no mutations along the evolution history, all N sequences are exactly identical to $\mathbf{S}_0$, and the multiplicity for any edge is N. With mutations present, there will appear many single edges corresponding to the mutated k-tuples, and the weight of edges in $\mathbf{S}_0$ (denoted as $\mathbf{W}(e \in \mathbf{S}_0)$) will decrease. We will use the Large Deviation Theorem (L.D.T.) for Binomials (see Arratia and Gordon [1989] for an exposition) to estimate the expected $min\{\mathbf{W}(e \in \mathbf{S}_0)\}$ based on the given parameters (N, L, k, β) as defined above. If this expected minimum weight is significantly larger than the expected $max\{\mathbf{W}(e \notin \mathbf{S}_0)\}$, our consensus path should consist of edges presented in $\mathbf{S}_0$ and hence be accurate.

Approximation by L.D.T. Consider a binomial random variable $X \sim B(N, p)$, where p is the probability of success. L.D.T. for Binomials is used to approximate the probability of x or more successes in N independent trials when the specified fraction of successes, $\alpha \equiv x/N$, satisfies $0 < p < \alpha < 1$. Define the odds ratio $r = \frac{p(1-\alpha)}{\alpha(1-p)}$ and the relative entropy $H \equiv H(\alpha, p) \equiv (\alpha)\log(\frac{\alpha}{p}) + (1-\alpha)\log(\frac{1-\alpha}{1-p})$. Then L.D.T. gives the following approximation (Arratia and Gordon, 1989):

$$P(X \geq \alpha N) \sim \frac{1}{1-r} \times \frac{1}{\sqrt{2\pi\alpha(1-\alpha)N}} \times e^{-NH}, \text{ as } N \to \infty. \tag{1}$$

In our case, we define a "success" to be a k-tuple that contains at least one mutation; then the number of mutated k-tuples at the i^{th} position among N sequences is a random variable X_i with a binomial distribution B(N,p), where $p = P$(the k-tuple has mutations) $= 1 - (1-\beta)^k$. For example, if $\beta = 0.05$ and $k = 16$, then $p = 0.56$. Applying Equation (1), it can be shown that for fixed α and i , $P(X_i \geq \alpha N)$ converges to 0 quickly as N increases. Therefore, for fixed sequence length L and assuming the independence of k-tuples in $\mathbf{S}_0$, $P(max\{X_i, \forall i\} \geq \alpha N) = 1 - P(X_i < \alpha N, \forall i) = 1 - P^{L-k+1}(X_1 < \alpha N) = 1 - (1 - P(X_1 \geq \alpha N))^{L-k+1} \to 0$, as N ncreases. So $max\{X_i \mid i =$ all possible positions in $\mathbf{S}_0\}$ is asymptotically less than αN, and thus $min\{\mathbf{W}(e \in \mathbf{S}_0)\} = N - max\{X_i \mid \forall i\}$ is asymptotically greater than $(1-\alpha)N$ as N increases. For example, if $\beta = 0.05$ and $k = 16$, weights of all edges in $\mathbf{S}_0$ are asymptotically greater than $0.44N$.

On the other hand, we estimate $max\{\mathbf{W}(e \notin \mathbf{S}_0)\}$ by the maximum number of identical k-tuples matching by chance, and if it is larger than $min\{\mathbf{W}(e \in \mathbf{S}_0)\}$, our consensus will be biased. Assume the sequences are randomly generated with uniformly distributed letters and all $N \times (L-k+1)$ k-tuples are independent; let a success be a k-tuple identical to a predefined k-tuple, with $p = 4^{-k}$. Under these assumptions, the number of identical k-tuples matching the predefined k-tuple by chance is a binomial random variable $Y \sim B(N(L-k+1), p)$. We usually pick a sufficiently large k so that $4^k \geq NL$. By Equation (1), the expected number of events with at least $\alpha N(L-k+1)$ k-tuples matching by chance can be estimated by $4^k N(L-k+1)P(Y \geq \alpha N(L-k+1)) \propto 4^k\sqrt{N(L-k-1)}e^{-N(L-k+1)H}$. This number converges to 0 quickly as N increases. Therefore, the maximum number of identical k-tuples matching by chance is

asymptotically less than $\alpha N(L-k+1)$ for any $\alpha \geq p$. This implies that $max\{\mathbf{W}(e \notin \mathbf{S}_0)\}$ is asymptotically less than $\alpha N(L-k+1)$. As mentioned, usually we choose k to satisfy $p = 4^{-k} \leq (NL)^{-1}$, so that α can be sufficiently small and the inequality $max\{\mathbf{W}(e \notin \mathbf{S}_0)\} < \alpha N(L-k+1) < min\{\mathbf{W}(e \in \mathbf{S}_0)\}$ holds for large N.

The above computation is over-simplified and many assumptions are unrealistic. But we hope it demonstrates the ability of our heaviest path method to discover the underlying ancestral sequence. Furthermore, the approximation by L.D.T. for Binomials provides a guidance in choosing tuple-size k, for both the lower-bound and the upper-bound.

5. SIMULATION RESULTS

EulerAlign is tested on a SUN UltraSPARC 750MHz (CPU speed) workstation for simulated sequences, ranging from 6 to 500 sequences. Each sequence is from 200 to 1000 bp long, and the similarities between sequences are from 90% to 70%, i.e., the mutation rate per base ranges from 5% to 16%. Substitutions, insertions, and deletions are uniformly and randomly distributed in all sequences, and in some cases according to an evolutionary tree model.

We use three different ways to evaluate alignments:

(1) Sum-of-pairs (SP): One of the typical scoring schema. Although this score will not be appropriate when both closely related and distantly related sequences are present, it is appropriate for the equidistant sequences (type A sequences mentioned below). The SP score is normalized by the number of pairs $\frac{N(N-1)}{2}$.

(2) Aligning Alignment Score (AA): From an ancestral sequence and a phylogenetic tree along which all homologous sequences are derived, one can unambiguously construct the TRUE multiple alignment by following the divergence history along the tree. By using simulated data, we know exactly the ancestral sequence and the tree by which those mutations are generated. Thus, the true multiple alignment is known. By measuring the distance between the true alignment and the alignment obtained by different methods, respectively, one can tell which method gives a better alignment. Aligning Alignment serves as a generalized version of pairwise alignment by regarding each column in one alignment as a "letter" and defining a scoring function for these letters (Waterman and Perlwitz, 1984; Waterman, 1995). The AA score is normalized by N.

(3) Identity (ID): An identity measure between a given alignment $\mathbf{A}$ and the true alignment $\mathbf{A}^*$ mentioned above: $\mathbf{ID} = \frac{\sum_{ijk, i<j} I_{ijk}}{\sum_{ijk, i<j} 1} \times 100\%$, where i and j are sequence indices. Here we consider a sequence pair (i, j). If the k^{th} letter of sequence i matches letter $s \in \{A, C, G, T, -\}$ of sequence j in alignment $\mathbf{A}$ and it also matches s of sequence j (not necessarily at the same position in j) in alignment $\mathbf{A}^*$, then $I_{ijk} = 1$. Otherwise, $I_{ijk} = 0$. The ID score is normalized as a percentage of identities.

We compared EulerAlign with CLUSTALW, one of the best-studied and popular currently available global multiple alignment programs. The test sequences are simulated in two different ways:

(A) All substitutions, insertions and deletions for each sequence $\vec{A}_i$, $i = 1 \cdots N$ are generated randomly and uniformly on a predetermined sequence $\vec{A}_0$, where $\vec{A}_0$ is random. The resulting sequences exactly fit to the consensus alignment model.

(B) First an evolutionary tree, in which the branch lengths are all equal, is generated, and then different mutations, with equal probability on each sequence based on its parent sequence from the tree, are generated. Although this tree is not correct from an evolutionary point of view because of the equal-length branches, it introduces mutation dependence into sequences and creates sequences that should fit the model of CLUSTALW. More importantly, we want to test EulerAlign on nonequidistant sequences.

Tables 1 and 2 compare the results from EulerAlign and CLUSTALW on both types of sequences, with various N, L, and mutation rates. The alignment scoring function is as follows: match = 0; mismatch =

TABLE 1. MULTIPLE ALIGNMENT COMPARISON BETWEEN EULERALIGN AND CLUSTALW ON EQUIDISTANT SEQUENCES (TYPE A)

Sequences			*SP*		*AA*		*ID*		*Time/sec*	
N	*L*	*Mutation*	*Eul*	*Clw*	*Eul*	*Clw*	*Eul*	*Clw*	*Eul*	*Clw*
6	200	5.2%	48	49	19	29	97.4%	93.8%	1	2
10	200	5.2%	53	58	13	37	98.2%	91.7%	1	3
15	200	5.2%	52	66	17	46	97.7%	91.6%	3	4
50	200	5.2%	53	119	14	83	97.9%	85.1%	4	32
6	500	5.2%	120	121	34	55	97.9%	96.0%	3	6
10	500	5.2%	135	148	35	83	97.8%	92.4%	4	14
15	500	5.2%	130	157	41	92	97.5%	92.1%	8	25
50	500	5.2%	141	309	44	230	97.3%	84.1%	11	189
100	500	5.2%	131	396	39	298	97.7%	81.8%	22	643
500	500	5.2%	130	719	38	640	97.8%	64.4%	132	14348
6	200	10.5%	113	103	57	101	90.8%	76.6%	1	1
10	200	10.5%	140	118	91	129	86.0%	70.8%	1	2
15	200	10.5%	133	140	71	149	89.7%	68.2%	3	4
50	200	10.5%	146	188	82	200	86.9%	59.5%	4	29
6	500	10.5%	335	283	191	264	88.3%	77.3%	3	7
10	500	10.5%	334	332	201	317	87.1%	72.0%	3	14
15	500	10.5%	326	339	182	357	88.8%	70.3%	7	25
50	500	10.5%	380	493	268	506	83.3%	61.2%	9	183
100	500	10.5%	380	580	245	606	84.5%	55.9%	19	628
500	500	10.5%	349	751	195	865	88.1%	43.6%	113	14093
6	200	16.4%	232	154	334	282	51.5%	45.6%	1	1
10	200	16.4%	237	163	236	260	64.2%	44.1%	1	2
15	200	16.4%	252	190	289	288	56.8%	43.1%	2	3
50	200	16.4%	262	211	287	332	59.0%	34.0%	3	19
6	500	16.4%	593	389	666	623	59.2%	48.5%	3	6
10	500	16.4%	591	429	634	652	61.2%	43.8%	3	12
15	500	16.4%	621	459	702	709	58.3%	42.4%	5	21
50	500	16.4%	627	541	686	818	59.7%	36.4%	7	129
100	500	16.4%	644	565	726	875	57.7%	32.7%	16	397
500	500	16.4%	628	636	685	1012	60.9%	29.2%	98	8514

1; gapopen = 4; gapextension = 1, an affine distance measure. The relations between mutation rates and sequence similarities are 5.2% $\sim$ 90%, 10.5% $\sim$ 80%, 16.4% $\sim$ 70%.

As shown in the tables, the quality of our alignments for both equidistant sequences (Table 1) and nonequidistant sequences (Table 2) is generally better than CLUSTALW, especially for large N, although type B sequences fit CLUSTALW's model. More significantly, the quality of EulerAlign's alignments is almost invariant with N. When N is small (≤ 10), our SP scores are worse than CLUSTALW, but AA and ID scores are generally better. Finally, the time cost of EulerAlign is approximately linear with respect to N. Figure 8 shows two almost identical alignments obtained by EulerAlign and CLUSTALW.

We also compared EulerAlign with a program using hidden Markov models, SAM (Hughey and Krogh, 1996), for which the simulated equidistant sequences perfectly fit the model. It turns out that SAM outperforms CLUSTALW on equidistant sequences when evaluated by the AA scoring scheme. But EulerAlign works at least as well as SAM, and we believe that EulerAlign will be superior to SAM when the input sequences are more complex. In addition, with some modification, our method can be extended to many other applications, including multiple local alignment and repeat finding, which will be presented in another paper.

TABLE 2. MULTIPLE ALIGNMENT COMPARISON BETWEEN EULERALIGN AND CLUSTALW ON NONEQUIDISTANT SEQUENCES (TYPE B)[a]

Sequences			*SP*		*AA*		*ID*		*Time/sec*	
N	*L*	*Mutation*	*Eul*	*Clw*	*Eul*	*Clw*	*Eul*	*Clw*	*Eul*	*Clw*
6	500	5.2%	110	110	36	61	98.1%	95.2%	3	6
10	500	5.2%	162	172	71	109	96.4%	91.1%	4	13
15	500	5.2%	163	180	60	120	96.6%	90.9%	6	31
50	500	5.2%	138	242	48	160	97.1%	88.8%	11	181
100	500	5.2%	151	399	50	318	97.2%	80.1%	24	631
6	500	10.5%	236	219	91	175	94.4%	84.9%	3	7
10	500	10.5%	380	342	280	375	83.3%	69.2%	4	14
15	500	10.5%	356	340	298	338	83.8%	76.0%	7	25
50	500	10.5%	397	494	283	501	83.2%	63.0%	10	181
100	500	10.5%	345	582	214	610	86.9%	60.3%	18	623
6	500	16.4%	527	362	571	557	65.2%	52.2%	2	6
10	500	16.4%	550	396	629	647	63.9%	47.8%	4	13
15	500	16.4%	583	438	613	635	66.1%	48.8%	4	24
50	500	16.4%	626	540	697	838	59.7%	38.3%	8	119
100	500	16.4%	601	623	649	868	62.6%	39.2%	17	500

[a]"Eul" is EulerAlign, Clw is CLUSTALW, N is the number of sequences to be aligned, L is the average length of each sequence, and "mutation" the mutation rate, corresponding to the similarity between sequences.

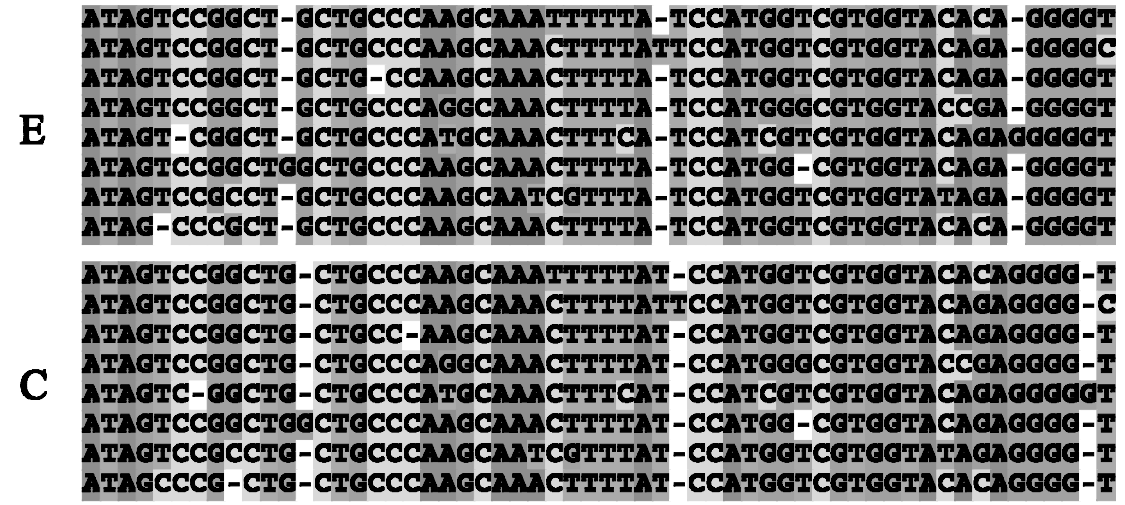

FIG. 8. An example of alignments from EulerAlign (E) and CLUSTALW (C). Total 8 sequences with average length 60 bases and $\sim 90\%$ identities are aligned. The visual tool used is ClustalX (Thompson *et al.*, 1997).

6. COMPUTATIONAL COMPLEXITY

From the results shown in Tables 1 and 2, the time cost for our algorithm is modest. When $N = 500$ and $L = 500$, EulerAlign runs three minutes while CLUSTALW runs $3 \sim 4$ hours. The time cost for EulerAlign is $\sim O(NL)$: (i) for construction and transformation of the graph, it is $\sim O(NL)$, where NL is the total size of sequences, (ii) for finding the heaviest path, it is $\sim O(|\Sigma|NL)$, because each vertex connects at most $2|\Sigma|$ other vertices, where $|\Sigma|$ is the size of the alphabet set Σ, (iii) for banded pairwise alignment, it is $\sim O(NL)$.

In addition, EulerAlign's memory usage is also efficient, because the size of our de Bruijn graph constructed by k-tuples is proportional to the size of input sequences, say, $O(kNL)$, and the memory for banded pairwise alignment is $O(L)$. Consequently, the total memory usage is $O(kNL)$.

7. APPLICATION ON ARABIDOPSIS SEQUENCES

We have applied EulerAlign on the results of resequencing a portion of the Arabidopsis genome from 96 individuals in order to find polymorphism. Data is provided by Prof. M. Nordborg and his group at USC. Each individual genome is sequenced from both the forward and the reverse complement strands, and each letter has a quality value computed by Phred (Ewing and Green, 1998). We implement quality values into EulerAlign by two steps: first, we redefine the weight of edges based on quality values; second, we apply quality values in the scoring matrix of pairwise alignment.

To define the weight of edges from quality values, we need to combine three types of information: edge length, multiplicity, and quality values of all letters in the edge. Recall that the Phred score is defined as $Q = -10 \times \log(\mathbf{P}(\alpha \text{ is incorrect} \mid \alpha \text{ observed}))$, and thus a higher quality value means a more accurate base-call of the letter. Regardless of the genetic variations, sampling sequences from N individuals is equivalent to sequencing one individual N times. Let G denote the individual being sequenced and S_i denote the sequence obtained from the i^{th} sequencing. If a k-tuple is called in n (> 1) sequencing processes from the same position, the probability that the k-tuple is indeed in G should be higher than the probability computed from one sequencing result. Based on this fact, let T_e denote the k-tuple in an edge $\mathbf{e}$ and let $T_e^{i,j}$ denote the actual occurrence of T_e in S_i at position j. We redefine the weight function of an edge $\mathbf{e}$ with multiplicity n by calculating the probability $P_{T_e}^{(n)} = P(T_e \in G \mid T_e \text{ called in n sequences from the same position in G})$ (we assume all k-tuples in edge $\mathbf{e}$ are called from the same position in G, because our graph is an alignment) and the mean value $\bar{P}_{T_e} = \frac{1}{M}\Sigma_{i,j} P(T_e^{i,j} \in G \mid T_e^{i,j} \text{ called})$, where M is the total number of T_e called at any position in any sequence, and finally taking the log ratio $P_{T_e}^{(n)}$ divided by $\bar{P}_{T_e}$. That is, $W(\mathbf{e}) = \log\frac{P_{T_e}^{(n)}}{\bar{P}_{T_e}}$. A larger weight means the k-tuples are more accurate. By assuming the independence of letters, we calculate this log ratio for each letter in T_e individually, where the quality values for each letter from Phred can be directly applied, and then do a summing. Since each edge is constructed from overlapping k-tuples, the overlapping parts among different edges will be counted only once; i.e., during the heaviest path finding process, the weight for each edge is adjusted to disregard the overlapping parts. Following the above formulation, the heaviest path weight is the log ratio of the actual probability that the path is correct divided by the expected probability that the path is correct, and the heaviest path will consist of edges representing the most accurately called k-tuples. We also tried other formulations for the weight functions of edges, such as log likelihood ratio models, and obtained similar results.

When doing pairwise alignment, instead of using an arbitrary scoring matrix, we apply the quality values to compute an average score for aligning two letters. The quality values for each sequence are given, while the quality values for consensus can be computed from the path, or alternatively one can deem the consensus completely correct. The average score of aligning two letters α, β is then computed as $\bar{S}_{\alpha\beta} = \Sigma S_{\alpha'\beta'} P_{\alpha'\beta'}$, where α' and β' are all possible letters, $S_{\alpha'\beta'}$ is a predefined score, and $P_{\alpha'\beta'} = \mathbf{P}(\alpha' \text{ and } \beta' \text{ are true letters} \mid \textit{Observations})$ is computed from both quality values of letter α and β. Gap penalties remain arbitrary in the current version of EulerAlign. Figure 9 shows part of the dataset At_000000072 aligned by EulerAlign.

We computed the alignments of five sequence sets by EulerAlign and CLUSTALW respectively. To apply quality values in CLUSTALW, we replaced DNA letters of different qualities by different protein alphabets. For example, the letter A of quality 40 was replaced by W, whereas A of quality 20 was replaced by V. Then we provided CLUSTALW with a specific score matrix to do the alignment. This work was done by Tina Hu in Prof. M. Nordborg's group at USC. For the sake of consistency, we used a modified version of sum-of-pair scores to measure the alignments computed with quality values. That is, for a match/mismatch, we computed an average match/mismatch score as described before; for an indel, the gap penalty is inversely related to the quality value of the corresponding letter. To test the robustness of EulerAlign, we also computed the alignments without using quality values and compared them using the typical sum-of-pair score. Table 3 shows the results.

FIG. 9. Part of EulerAlign's alignment of At_000000072. The visual tool used is ClustalX (Thompson *et al.*, 1997).

The alignments by CLUSTALW using quality values are worse than those by EulerAlign on all sequence sets. We tested different score matrices and gap penalties but observed similar results. Since each individual genome was sequenced from both the forward and the reverse complement strands, the asymmetric base-calls between two strands may complicate the alignment. In addition, the quality values in each sequence vary greatly. These observations imply the necessity to combine the forward and the reverse complement strands into one sequence before doing alignments.

One may argue that one can simply choose the sequence with the highest average quality value as the consensus. Doing this, in addition to its limited flexibility, gives results that are surprisingly bad compared to using the consensus obtained from the graph, although the two consensus sequences are very similar to each other. To test whether the result is due to the poor quality regions, we cut all sequences at both ends to obtain a set of putative sequences that should align properly with high quality values. Although the result is much better than that when using full length sequences, the conclusion remains unchanged. As shown in Fig. 10, the maximal normalized multiple alignment score is 0.655 (similarity score) by using each sequence as the consensus respectively, whereas we get 0.693 by using our graph consensus. On the

TABLE 3. MULTIPLE ALIGNMENT COMPARISON BETWEEN EULERALIGN AND CLUSTALW ON ARABIDOPSIS SEQUENCES[a]

			With qual		*Without qual*		
Sequences	*N*	*L*	*Eul*	*Clw*	*Eul*	*Clw*	*Time by Eul*
At0072	184	751	214	216	188	194	48 sec
At0076	188	761	234	335	199	210	75 sec
Arab008	181	720	299	431	256	287	39 sec
Arab022	190	701	198	288	159	175	36 sec
Arab042	187	700	182	305	160	174	40 sec

[a](Mismatch, GapOpen, GapExt) = (1, 4, 1); Eul: EulerAlign; Clw: CLUSTALW; time by CLUSTALW: > 40 mins.

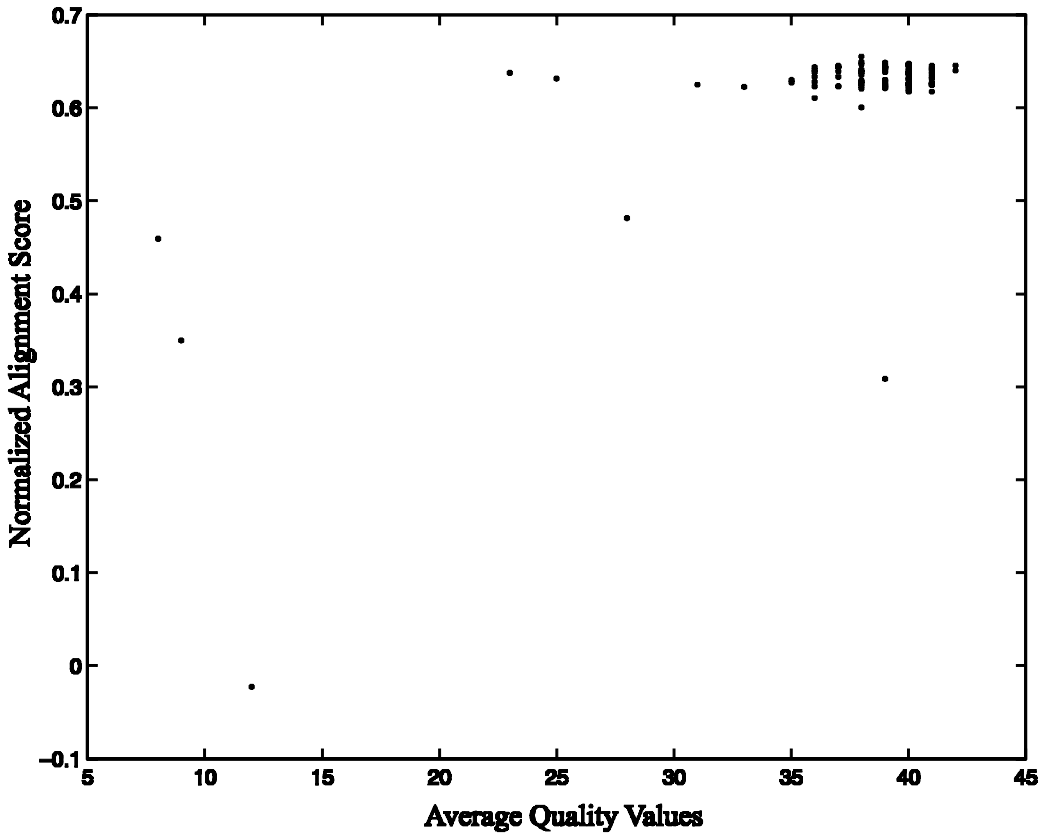

FIG. 10. The normalized sum of pairs score obtained by directly using each sequence as consensus against average quality value of that sequence. Each point represents a sequence. The sequence achieving the best alignment score is not the sequence with the highest average quality value.

other hand, we have the reference sequence from the assembled genome from TIGR, which is regarded as the correct sequence. We then draw two histograms for the similarity between each sequence to the reference and to our consensus. The curve of the similarity score distribution to our consensus turns out to be narrower than that to the reference (Fig. 11), which shows that our consensus is a better estimation of the "center" than is the reference based on the given data.

8. DISCUSSION AND OPEN PROBLEMS

EulerAlign's most significant advantages are its capability to handle a large number of sequences simultaneously and its extraordinary speed. In addition, a consensus sequence that captures the most conserved similarities is obtained before getting an alignment, a distinguishing feature with regard to other alignment algorithms. EulerAlign is very flexible in that additional information can be easily incorporated into the de Bruijn graph. For example, if input sequences have different weights, the weights can be easily incorporated into the multiplicity for edges by modifying the definition. If position-specific information is given, besides doing position-specific pairwise alignment, we can also construct the de Bruijn graph according to this information.

Although our Eulerian approach to global multiple sequence alignment has been successful, there exist some practical problems and we will now discuss them.

Choice of k-tuple size

The choice of tuple-size k affects the quality of the consensus path, because generally one mutation in a sequence will cause a single edge to diverge away from the common edges by $(2k - 1)$ letters. Thus,

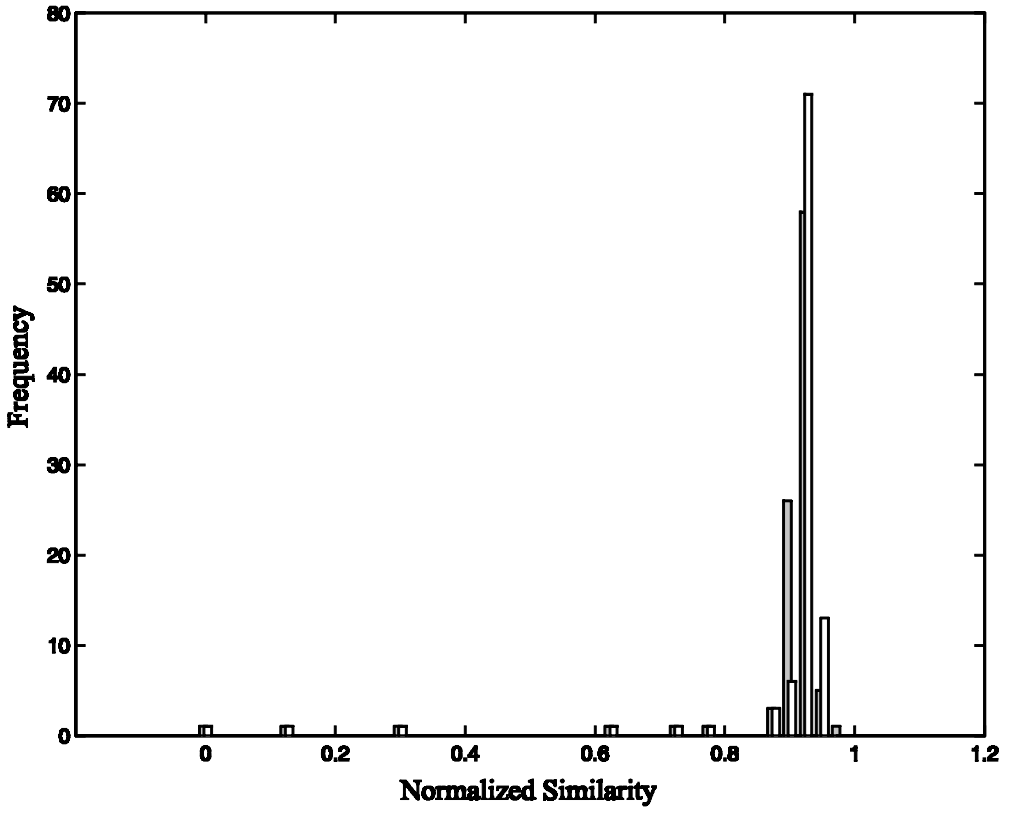

FIG. 11. Distribution of similarity between each sequence to either the reference sequence (grey bar) or our consensus (white bar). The grey bars are slightly shifted to the left for visualization purpose.

the larger k is, the fewer commonalities (or multiplicities for edges) are retained in the graph. Of course, the smaller k is, the more likely it is that a k-tuple is not unique in the sequence.

Currently, EulerAlign arbitrarily chooses a k to construct the de Bruijn graph. For small N and distantly related sequences, a small k (≤ 10) is chosen in order to get high multiplicities for edges. For large N and closely related sequences, a large k (≥ 16) may be proper because a graph constructed using large k is always simpler to solve (fewer random matches result in fewer tangles and cycles) than the one using small k. The upper bound of k can be estimated by L.D.T.

Graph transformation may lose information

As mentioned above, sometimes the safe transformations cannot remove all cycles in the graph, and thus several unsafe transformations may be performed in later stages to remove all remaining cycles. Unsafe transformations inevitably cause the loss of similarity information among sequences that are represented by multiplicities of edges. One reason that a transformation is unsafe comes from our transformation rules. We consider only two left edges at one time, which is not sufficient. One improvement is to consider all left edges of a vertex simultaneously. On the other hand, since we are doing the global sequence alignment, we can perform a banded graph construction; i.e., when merging a k-tuple to an edge, in addition to the requirement of identical k-tuples, their corresponding positions must also be within a window. This feature has been implemented as an option in current version of EulerAlign.

Arbitrary scoring function

In the pairwise alignment step, the scoring function is arbitrarily chosen. This is a common issue for all alignment algorithms that use arbitrary scoring functions, because no scoring function exists that is optimal for all types of input sequences. Consequently, how to choose the optimal scoring functions for different data is a problem. However, because our graph itself is a representation of multiple sequence alignment,

the drawback from an arbitrary scoring function might be solved by using data from the graph; e.g., the existence of many sequence paths sharing a common edge in the graph provides us more confidence to align them together at the corresponding positions.

Our de Bruijn graph is in fact another representation of multiple alignment, in which the alignments of letters are represented by many sequence paths sharing common edges. Since one mutation in sequences will cause a single path to diverge from the consensus by $(2k - 1)$ letters when using a k-tuple, a de Bruijn graph inaccurately represents the multiple alignment. However, the graph representation has its own advantages in searching local commonalities and showing sequence variability. In fact, the Eulerian path approach to the local multiple alignment problem is a natural extension to its global counterpart. We will present the local alignment algorithm in another paper.

NOTE ADDED IN PROOF

After the preparation of our paper, we became aware of a paper "Multiple sequence alignment using partial order graphs" (Lee, C., Grasso, C., and Sharlow, M. 2002. *Bioinformatics*, 18, 452–464). In that paper, a graph structure is embedded in the dynamic programming algorithm to efficiently align a large number of sequences. The authors' method successively reduces the space requirement due to the graph structure applied. A series of pairwise alignments are employed in an arbitrary order under a certain scoring scheme. As with any pairwise method, errors can easily accumulate during the alignment process that results in misalignment. As a comparison, CLUSTALW requires $O(N^2)$ time simply to find a better order to do the pairwise alignments.

ACKNOWLEDGMENTS

We thank Professor Magnus Nordborg at USC for kindly providing the Arabidopsis sequencing data and are grateful to Professor Lei Li at USC and Dr. Haixu Tang at UCSD for many helpful discussions. This research was supported by NIH grant R01 HG02360-01.

REFERENCES

Altschul, S.F., Carroll, R.J., and Lipman, D.J. 1989. Weights for data related by a tree. *J. Mol. Biol.* 207,647–653.

Arratia, R., and Gordon, L. 1989. Tutorial on large deviations for the binomial distribution. *Bull. Math. Biol.* 51, 125–131.

Barton, G.J., and Sternberg, M.J.E. 1987a. Evaluation and improvements in the automatic alignment of protein sequences. *Protein Eng.* 1, 89–94.

Barton, G.J., and Sternberg, M.J.E. 1987b. A strategy for the rapid multiple alignment of protein sequences confidence levels from tertiary structure comparisons. *J. Mol. Biol.* 198, 327–337.

Dayhoff, M.O., Schwartz, R.M., and Orcutt, B.C. 1978. A model of evolutionary change in proteins, *in* DayHoff, M.O., ed., *Atlas of Protein Sequence and Structure*, 345–352, National Biomedical Research Foundation, Washington, DC.

Eddy, S.R. 1995. Multiple alignment using hidden Markov models. *ISMB* 3, 114–120.

Ewing, B., and Green, P. 1998. Base-calling of automated sequencer traces using phred. II. error probabilities. *Genome Res.* 8,186–194.

Feng, D., and Doolittle, R.F. 1987. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.* 25, 351–360.

Gotoh, O. 1996. Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments. *J. Mol. Biol.* 264, 823–838.

Gotoh, O. 1999. Multiple sequence alignment: Algorithms and applications. *Adv. Biophys.* 36, 159–206.

Higgins, D.G., and Sharp, P.M. 1989. Fast and sensitive multiple sequence alignments on a microcomputer. *CABIOS* 5, 151–153.

Hughey, R., and Krogh, A. 1996. Hidden Markov models for sequence analysis: Extension and analysis of the basic method. *CABIOS* 12, 95–107.

Idury, R., and Waterman, M.S. 1995. A new algorithm for DNA sequence assembly. *J. Comp. Biol.* 2, 291–306.

Kececioglu, J.D. 1993. The maximum weight trace problem in multiple sequence alignment. *CPM*, 106–119.

Lipman, D.J., Altschul, S.F., and Kececioglu, J.D. 1989. A tool for multiple sequence alignment. *Proc. Natl. Acad. Sci. USA.* 86, 4412–4415.

Martinez, H.M. 1988. A flexible multiple sequence alignment program. *Nucl. Acids Res.* 16, 1683–1691.

McClure, M.A., Vasi, T.K., and Fitch, W.M. 1994. Comparative analysis of multiple protein-sequence alignment methods. *Mol. Biol. Evol.* 11, 571–592.

Morgenstern, B. 1999. DIALIGN 2: Improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics* 15, 211–218.

Morgenstern, B., Dress, A., and Werner, T. 1996. Multiple DNA and protein sequence alignment based on segment-to-segment comparison. *Proc. Natl. Acad. Sci. USA* 93, 12098-12103.

Notredame, C. 2002. Recent progresses in multiple sequence alignment: A survey. *Pharmacogenomics* 3, 131–144.

Notredame, C., and Higgins, D.G. 1996. SAGA: Sequence alignment by genetic algorithm. *Nucl. Acids Res.* 24, 1515–1524.

Notredame, C., Higgins, D.G., and Heringa, J. 2000. T-Coffee: A novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol.* 302, 205–217.

Notredame, C., Holm, L., and Higgins, D.G. 1998. COFFEE: An objective function for multiple sequence alignments. *Bioinformatics* 14, 407–422.

Pevzner, P.A., Tang, H., and Waterman, M.S. 2001. An Eulerian path approach to DNA fragment assembly. *Proc. Natl. Acad. Sci. USA* 98, 9748–9753.

Sankoff, D. 1975. Minimal mutation trees of sequences. *SIAM J. Appl. Math.* 28, 35–42.

Sankoff, D., Cedergren, R.J., and Lapalme, G. 1976. Frequency of insertion–deletion, transversion, and transition in the evolution of 5S ribosomal RNA. *J. Mol. Evol.* 7, 133–149.

Sankoff, D., and Kruskal, J. 1983. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, Addison-Wesley, Reading, MA.

Smith, R.F., and Smith, T.F. 1990. Automatic generation of primary sequence patterns from sets of related protein sequences. *Proc. Natl. Acad. Sci. USA* 87, 118–122.

Smith, R.F., and Smith, T.F. 1992. Pattern-induced multisequence alignment (PIMA) algorithm employing secondary structure-dependent gap penalties for use in comparative protein modeling. *Protein Eng.* 5, 35–41.

Taylor, W.R. 1988. A flexible method to align a large number of sequences. *J. Mol. Evol.* 28, 161–169.

Thompson, J.D., Higgins, D.G., and Gibson, T.J. 1994. CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucl. Acids Res.* 22, 4673–4680.

Thompson, J.D., Gibson, T.J., Plewniak, F., Jeanmougin, F. and Higgins, D.G. 1997. The ClustalX windows interface: Flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucl. Acids Res.* 24, 4876–4882.

Thompson, J.D., Plewniak, F., and Poch, O. 1999. A comprehensive comparison of multiple sequence alignment programs. *Nucl. Acids Res.* 27, 2682–2690.

Vingron, M., and Argos, P. 1991. Motif recognition and alignment for many sequences by comparison of dot matrices. *J. Mol. Biol.* 218, 33–43.

Vingron, M., and Waterman, M.S. 1996. Alignment networks and electrical networks. *Disc. Appl. Math.* 71, 297–309.

Waterman, M.S. 1995. *Introduction to Computational Biology*, Chapman and Hall, London.

Waterman, M.S., and Jones, R. 1990. Consensus methods for DNA and protein sequence alignment. *Methods Enzymol.* 183, 221–237.

Waterman, M.S., and Perlwitz, M.D. 1984. Line geometries for sequence comparisons. *Bull. Math. Biol.* 46, 567–577.

Waterman, M.S., Smith, T.F., and Beyer, W.A. 1976. Some biological sequence metrics. *Adv. Math.* 20, 367–387.

Address correspondence to:
Yu Zhang
Department of Mathematics
University of Southern California
1042 West 36th Place (DRB 289)
Los Angeles, CA 90089-1113

E-mail: yuzhang@usc.edu

This article has been cited by:

1. Hsun-Wen Chang, Pei-Fang Tsai. 2007. Characterizing the reconstruction and enumerating the patterns of DNA sequences with repeats. *Journal of Combinatorial Optimization* **14**:2-3, 331. [CrossRef]
2. Eva K. Lee, Todd Easton, Kapil Gupta. 2006. Novel evolutionary models and applications to sequence alignment problems. *Annals of Operations Research* **148**:1, 167. [CrossRef]
3. Y. ZHANG, M.S. WATERMAN. 2003. DNA Sequence Assembly and Multiple Sequence Alignment by an Eulerian Path Approach. *Cold Spring Harbor Symposia on Quantitative Biology* **68**:1, 205. [CrossRef]