DNA Sequence Assembly and Multiple Sequence Alignment by an Eulerian Path Approach

Yu Zhang*

Department of Mathematics
University of Southern California
Los Angeles, CA 90089-1113
Phone: 213-821-2231
yuzhang@usc.edu

Michael S.Waterman

Department of Biological Sciences
University of Southern California
Los Angeles, CA 90089-1340
Phone: 213-740-2409
Fax: 213-740-2437
msw@usc.edu

22 May 2003

Running Head:
Eulerian Assembly and Multiple Alignment

Corresponding Author:

Yu Zhang

Mailing Address:

USC, Department of Mathematics
1042 W. 36th Place, DRB289
Los Angeles, CA 90089-1113

Phone: 213-821-2231

Fax: 213-740-2437

E-mail: yuzhang@usc.edu

Abstract

We describe an Eulerian path approach to the DNA fragment assembly that was originated by Idury and Waterman 1995, and then advanced by Pevzner et al. 2001b. This combinatorial approach bypasses the traditional "overlap-layout-consensus" approach and successfully resolved some of the troublesome repeats in practical assembly projects. The assembly results by the Eulerian path approach are accurate, and its computation is significantly more efficient than other assembly programs.

As an extension, we use the Eulerian path idea to address the multiple sequence alignment problem. In particular, we have as a goal aligning thousands of sequences simultaneously, which is computationally exorbitant for all existing alignment algorithms. As a beginning, we focus on DNA sequence alignment. Our method can align hundreds of DNA sequences within minutes with high accuracy, and its computational time is linear to the number of sequences. We demonstrate its performance by alignments of simulated sequences and by an application in a resequencing project of *Arabidopsis thaliana*.

Although having some weaknesses including aligning gap-rich regions, the Eulerian path approach is distinguished from other existing algorithms in solving either fragment assembly or multiple alignment

problems, and hence provides a new perspective in solving these problems.

1 DNA Sequence Assembly

Assembly of short DNA fragments (500-1000bp) generated by shotgun sequencing is a widely used technique for sequencing large genomes, including the human genome. The most popular framework of DNA fragment assembly algorithms in the past 25 years is the "overlap-layout-consensus" approach. All high quality DNA fragments are first compared to each other for possible overlaps; then a layout is made by arranging all DNA fragments into relative positions and orientations according to the overlap information; finally a multiple alignment is computed to obtain a consensus sequence that will be used as the genomic sequence. The main difficulty with this framework in addition to the computation required, comes from the fact that genomic sequences always contain large amount of repeat regions accumulated along their evolutionary history. In particular, repeats that are longer than the fragment length and have $> 98\%$ identity are hard to distinguish from true overlaps, and hence finding a correct path in the layout step is difficult.

Surprisingly, a 15-year-old computational model from DNA arrays provides the basis for a novel approach to assembly. Sequencing by

Hybridization (SBH) is conceptually analogous to DNA fragment assembly by regarding each DNA fragment as a k-tuple. Idury and Waterman (Idury and Waterman 1995) mimicked SBH procedure by breaking each DNA fragment of length n into n-k+1 overlapping k-tuples and hybridized all k-tuples in silicon such that a DNA fragment assembly problem is mapped into a SBH problem. A de Bruijn Graph is constructed in their SBH approach where each edge represents a k-tuple from fragments; two edges share a common vertex if they share a common (k-1)-tuple; identical k-tuples share the same edge. Repeats and sequencing errors make the simple de Bruijn graph extremely tangled and the solution of DNA fragment assembly from such a complex graph remains challenging. Pevzner et al. took the SBH idea and provided some additional features (Pevzner et al. 2001b). Instead of building an overlap graph in the traditional "overlap-layout-consensus" framework, the Eulerian path approach builds a de Bruijn graph that represent all fragments and their relationships in a much simpler way. In addition, the difficulty resulting from the repeats and sequencing errors often are much easier to conquer in the de Bruijn graph structure.

Assume a DNA sequence consists of four unique subsequences that are separated by one triple repeat, as shown in figure 1a. The traditional "overlap-layout-consensus" approach builds an overlap graph by

regarding every fragment as a vertex, and two vertices are connected if two corresponding fragments overlap. Figure 1b shows the overlap graph in our example. The assembly problem is thus finding a path in the overlap graph that visits every vertex exactly once, a Hamilton path problem that is well known NP-complete. On the contrary, the Eulerian path approach breaks all fragments into k-tuples and builds a de Bruijn graph as described above (figure 1c). A conceptually ideal de Bruijn graph is shown in figure 1d, a much simpler representation of repeats than the overlap graph. The most important advantage of this representation is that the assembly problem is now finding a path that visits every edge exactly once, an Eulerian path problem that has linear-time solutions (Fleischner 1990).

The implementation of the Eulerian path approach to DNA fragment assembly problems is called EULER. Traditional algorithms postpone the error correction until the last consensus step, while EULER applies error correction at the beginning of assembly (this is the innovation of Pevzner et al.). Without knowing the finished genomic sequence or even the layout of fragments, error correction is still possible by approximating the k-tuple spectrum and the result is usually accurate enough. After error correction, EULER constructs a de Bruijn graph, and stores the fragment information in corresponding edges. This fragment information is the fundamental difference between EULER

and SBH where such information is unavailable. EULER's superpath idea can successfully solve many repeats by using fragment information. Finally, EULER outputs an Eulerian path from the de Bruijn graph that represents the finished genomic sequence.

1.1 Error Correction

Sequencing errors make the de Bruijn graph a tangle of erroneous edges, thus very difficult to solve. EULER's error correction procedure reduces 97% errors from the original DNA fragments and makes the data almost error-free. The idea is based on an approximation to the spectrum of real genomic sequence, i.e. to find a collection of k-tuples all of which are from the genomic sequence instead of sequencing errors. With sequencing errors, many k-tuples are erroneous and many "true" k-tuples are missing. EULER calls a k-tuple solid if it appears in more than M fragments, otherwise it is a weak k-tuple. The error correction problem is thus to transform the spectrum of the original DNA fragments into the spectrum of a genomic sequence by changing weak k-tuples to solid k-tuples. Without knowing the real genomic sequence, one natural criteria of error correction is to minimize the total number of distinct k-tuples in the spectrum. One error in a fragment will create at most 2k (including the reverse complement part) erroneous k-tuples in the spectrum, or 2d (d<k) if the error appears near either end of a fragment so that at most 2d k-tuples can be

affected. EULER uses a greedy approach to look for error corrections that reduce the number of weak k-tuples by 2k or 2d.

The *Neisseria meningitidis* (NM) sequencing project (Parkhill 2000), one of the most "difficult-to-assemble" and "repeat-rich" bacterial genome completed so far, was used to demonstrate the efficiency of EULER's error correction method. The NM genome contains 2,184,406 nucleotides, with 126 long nearly perfect repeats up to 3,832 bp in length. The sequencing project resulted in 53,263 fragments (coverage 9.7), with 255,631 sequencing errors in total. EULER corrected 97.7% errors and made the original sequencing data almost error-free with 0.11 errors per fragment (Pevzner et al. 2001a).

1.2 Construction of De Bruijn Graph

EULER constructs a de Bruijn graph from the corrected DNA fragments in the following way: given a set of DNA fragments, EULER breaks each fragment and its reverse complement into overlapping k-tuples; each k-tuple represents a directed edge in the graph and the direction of an edge is the direction of reading a k-tuple; tuple positions and fragment indices are stored in the corresponding edges; each k-tuple contains two (k-1)-tuples that will represent vertices connected at two ends of an edge; all edges and vertices are "glued" together if they corresponds to identical k-tuples and (k-1)-tuples. An example

of the de Bruijn graph of two fragments {ATGC, ATGT} is shown in figure 2.

An edge is called single if the edge represents a single k-tuple in the genomic sequence (but this k-tuple may appear in many fragments that cover it), otherwise the edge is called multiple. By regarding each multiple-edge as m parallel single-edges if the edge represents m occurrences of the k-tuple in the genomic sequence, a fragment assembly problem is then to find an Eulerian path that visits each edge exactly once. The algorithm for finding an Eulerian path costs linear time and can detect erroneous edges that will then be discarded. In addition, without knowing orientations, EULER builds all fragments and their reverse complements into the de Bruijn graph, and expects that the graph can be partitioned into two complementary subgraphs, corresponding to reading the sequence in each direction.

1.3 Superpath Transformation

A vertex v is called a source if indegree(v) = 0, or a sink if outdegree(v) = 0. A branching vertex is a vertex that has indegree(v)$\times$outdegree(v) $>$ 1. The de Bruijn graph corresponding to the original fragments of the NM genome has 502,843 branching vertices (k=20), and this number is reduced to 12,175 after error correction. Even with error free data, however, the de Bruijn graph is still very complicated for shotgun se-

quencing projects. EULER uses the fragment information stored in edges to handle this difficulty.

Define a repeat structure a path $P_{v_1 \to v_n} = v_1 \cdots v_n$ in the graph, where indegree(v_1) > 1, outdegree(v_n) > 1, indegree(v_i)=outdegree(v_i) $= 1$, for $1 < i < n$, and outdegree(v_1)=indegree(v_n) $= 1$ if $n > 1$. A repeat structure represents a possible repeat in the assembled sequence. If indegree(v_1) $= p$ and outdegree(v_n) $= q$, then there will be $p \times q$ possible pairings of edges that a path enters the repeat structure from one edge and exits from the other, while the correct pairings corresponding to the assembled sequence are unknown. If a fragment covers the entire repeat, the correct pairings can be detected by following the fragment path. If no fragments covered the entire repeat, the correct pairings will remain unclear and the repeat structure will form a tangle.

A superpath transformation of the de Bruijn graph is then introduced to solve repeat structures. The goal of superpath transformations is to do a series of transformations to the graph so that the final graph contains no multiple-edges. Transformations on multiple-edges should be performed with caution, because fragment information stored in multiple-edges must be partitioned and stored separately into superpaths. For a detailed discussion, please refer to the original paper by

Pevzner et al. 2001.

One feature of the Eulerian superpath approach to the DNA sequence assembly problem is that it uses rather than struggles with the imperfect repeats. By superpath transformation, most imperfect repeats will eventually be separated into different paths. By using the linear-time Eulerian path algorithm, the Eulerian approach has the potential to assemble larger eukaryotic genomes in the future.

2 Multiple DNA Sequence Alignment

The linear-time Eulerian path algorithm has the potential of assembling a large eukaryotic genome of length up to gigabases. A closely related but complementary question is now asked: can the Eulerian path idea be applied to the multiple sequence alignment problem? That is, instead of assembling a long genomic sequence, can we use this idea in aligning a large number of short specific sequences?

Many MSA algorithms have been developed in the past decades. One bottleneck for all of them is the expensive computational cost when aligning extremely long sequences or a huge number of sequences simultaneously. We will demonstrate that the Eulerian path idea can provide almost linear time MSA with accurate solution of this problem. In particular, we present a program called EulerAlign (Zhang

and Waterman, 2003) that uses the Eulerian path method to solve the global MSA of a large number of DNA sequences. An application on a genome resequencing project is then presented to demonstrate its performance.

2.1 Motivation

EulerAlign takes the Eulerian path idea that builds all sequences into a de Bruijn graph, and then extracts the graph information to do the multiple sequence alignment. Recall the process of a sequence assembly project, where genomic sequences are broken into short pieces and these pieces are then randomly cloned to build a library of short DNA fragments. All fragments are sequenced and input into a DNA fragment assembly program to reconstruct the original genomic sequence. A global MSA problem is a special DNA sequencing project (figure 3), where the "genome" is short enough to be sequenced directly, but the "sequencing machine" makes a very high rate of errors.

The Eulerian path approach for the DNA assembly problem requires the input fragment to be "error-free", where EULER did this by an error-correction procedure. In a MSA problem, however, there could be thousands of sequences to be aligned, and the genetic differences among the sequences result in many "errors" compared with sequencing errors. Thus, the error-correction as used in EULER will be help-

ful but not likely to succeed in "correcting" most of them. A MSA problem is to recover the underlying consensus sequence from a large number of divergent sequences, where mutations do not pose a barrier because each sequence covers (almost) the entire consensus sequence. This difference allows us to construct an accurate alignment even with the presence of many mutations.

A consensus sequence is typically obtained from a given alignment by extracting the majority letters (in the simplest case) in each column of the alignment. EulerAlign reverses this procedure by first obtaining a consensus sequence from the graph and then builds the alignment from the consensus. In the de Bruijn graph, each sequence represents a sequence path, and each multiple-edge represents a k-tuple shared by many sequence paths. The letters that are common to a majority of sequences are thus determined by multiple-edges visited by many sequence paths. The multiplicity for an edge is the number of sequence paths visiting the edge. The higher the multiplicity is, the larger the chance that the edge represents a k-tuple in the consensus sequence. Based on this idea, EulerAlign assigns weight to each edge as a function of the edge's multiplicity and length, then uses a heaviest-path algorithm to extract a path with the largest sum of weights. The heaviest path problem (identical to the shortest path problem with negative weights) has linear time solutions for a directed acyclic graph (DAG)

and EulerAlign uses this heaviest path as the consensus to construct the final alignment. The pipeline for EulerAlign is as follows: 1)construct a de Bruijn graph; 2) transform the graph to a DAG; 3) extract a heaviest path as the consensus sequence; 4) do consensus alignment.

2.2 Graph Transformation and Consensus Alignment

The initially constructed graph contains cycles due to repeats and random matches, but an optimal heaviest path is acyclic in the graph. If the true consensus includes repeat regions, an acyclic heaviest path will not accurately represent the true consensus unless the cycles (repeats) are solved. EulerAlign uses superpath transformations defined in EULER to solve these cycles. A superpath transformation captures the sequence path information, and hence can minimize the loss of similarity information stored in edges (multiplicities). When the number of sequences is large, however, it could be very time consuming to remove all cycles by superpath transformations. In addition, an equivalent superpath transformation may not exist in some situations, and thus removing all cycles may cause significant information loss. EulerAlign compromises by applying superpath transformations only on a particular subset of cycles. A cycle is easily detected when a sequence path visits a vertex more than once, and such cycles are called self-cycles because they are entirely involved in one sequence path. Eu-

lerAlign uses the superpath transformation to remove a cycle if and only if the cycle is a self-cycle, and all sequence paths will be individually acyclic after removing all self-cycles. Because the heaviest path itself is acyclic, this procedure is designed to allow the heaviest path to capture most similarity information. A depth-first search in the graph is then performed to eliminate all remaining cycles. Although losing all information stored in the removed cycles, this procedure enables a rigorous heaviest path algorithm to be applied. The time cost for removing all remaining cycles is linear in the size of the graph.

After obtaining an acyclic graph, EulerAlign extracts the heaviest path according to the weight function defined for edges. The algorithm for finding the heaviest path costs linear time proportional to the size of the graph. Finally, EulerAlign applies the classical banded dynamic programming algorithm to align each sequence with the consensus, and builds the final multiple sequence alignment. Positional specific scoring functions derived from "heavy" edges and large potential indels indicated by the position shifts of k-tuples can be used during this process.

2.3 Performance

We have tested EulerAlign on both simulated and real DNA sequences. For simulated sequences, we randomly add substitutions and indels ac-

cording to two models: the equidistance model and the evolutionary model. In the equidistance model, each sequence is independently mutated with a common mutation distribution, and hence on average all sequences are equally similar to each other. In the evolutionary model, all sequences are related to a common ancestral sequence along an evolutionary tree. Mutations in each sequence are generated along the tree and hence are correlated among sequences. The equidistance model perfectly fits the requirement of a consensus alignment, but the evolutionary model reflects a more realistic situation.

Two scoring systems are used to evaluate the performance of EulerAlign: 1) Sum of pairs (SP) score, a popular and simple measure. 2) Aligning alignment (AA) score, comparison of an alignment to the true alignment; by simulating sequences, the true alignment (rather than the mathematically optimal alignment) is known. We used ClustalW (Higgins and Sharp 1989; Thompson et al. 1994), a well-studied and popular MSA software, as the reference. Figure 5 shows the comparison between EulerAlian and ClustalW on sequence sets generated by the evolutionary model with different mutation rates: 5.2% and 16.4%, respectively corresponding to 90% and 70% pairwise sequence similarities. The comparison on the equidistance model is not shown, simply because EulerAlign is designed for that model and hence achieves a better result. The linear growth of the computational time with re-

spect to the number of aligned sequences by EulerAlign is shown in figure 6a, and a significant comparison to the quadratic growth by ClustalW is shown in figure 6b. We used distance scores, and hence the smaller the score the better the result. All tests are done on a SUN UltraSPARC 750MHz workstation.

2.4 Application on *Arabidopsis* Sequences

Arabidopsis thaliana is widely used as a model organism for genetic study in plant biology. As an application on real genomic sequences, we used EulerAlign to construct alignments for several sets of short specific sequences sampled from 96 *Arabidopsis* individuals by PCR experiments with certain primers. These alignments are then used to study the genetic variations and hence evolutionary relationships in the *Arabidopsis* population. Sequence data are kindly provided by M. Nordborg at USC. Presented with base-calling errors, an accurate multiple alignment is crucial for efficiently detecting real genetic variations other than sequencing errors. The main difference between genetic variations and sequencing errors is that sequencing errors are more independently and randomly distributed (although of course a function of position in the sequence fragment).

To reduce base-calling errors, each individual is sequenced from both forward and backward strands, and each base-call has a quality value

assigned by Phred (Ewing and Green 1998). Because the forward and backward pairs represents the same genome region, any discrepancy between them indicates that at least one of two base-calls is wrong. Our experience shows that the base-calls in two strands are asymmetric, that is, base-calls in one strand tends to make certain errors more frequently than in the other strand. All sequence pairs (forward and backward strands) are combined together by Phrap (Green 1994) according to their quality values before doing multiple alignment.

Poor quality values are often assigned as the base-call approaches the end of sequences, and occasionally outlier sequences are generated by wrong PCR-amplification. Regions with low quality values in each sequence are not trimmed or discarded before doing alignment, and hence informative segments can be found even they have low qualities. Since the quality values within and among sequences are highly variant even after the combination of two strands, we must incorporate the quality values into the computation to avoid misalignments due to low quality regions.

We use ClustalW version 1.83 as the reference program. For methods of how quality values are used in EulerAlign and ClustalW, please refer to our previous paper (Zhang and Waterman, 2003). We tested 20 sequence sets, including both good sequence sets and bad sequence

sets where up to 18% sequences are either in very poor quality (< 20 in average after combining both strands) or outliers. The parameters for both EulerAlign and ClustalW are tuned "optimal" by human effort. By optimal we mean the best scoring functions and utilization of quality values for all 20 sequence sets, not individually. All alignments using ClustalW with quality values are done by Tina Hu, at USC.

We used a modified version of sum of pair scores to evaluate alignments from both programs. The scores are adjusted according to the quality value of each letter. For example, the mismatch penalty of two low quality letters is smaller than the penalty of two high quality letters. To test the robustness of EulerAlign, we also computed the alignments by both programs without using quality values, i.e., poor quality sequences and outliers are equally considered with high quality sequences. The sum of pair scores for these alignments are, of course, not adjusted by quality values. Table 1 shows the comparison. We use similarity scores so that the level of pairwise identities of each sequence set can be inferred from the relationship between sequence lengths and scores.

We found that the performance of both programs are comparable after tuning the alignment parameters and using quality values when doing alignment. ClustalW wins in 11 sequences sets whereas EulerAlign

wins in 9. By checking the alignments, we found the major alignment difference by two programs are for the outliers and at either end of sequences which have low quality and less identities. This result is possibly due to the different utilizations of quality values by EulerAlign and ClustalW when doing multiple alignment. On the other hand, without using quality values, both program computed alignments by their default parameters, and EulerAlign outperforms ClustalW in all 20 sequence sets.

It is known that the sum of pair scores or other scoring schema can not always reflect the correctness of a biologically meaningful alignment. An example of mis-alignments, one by ClustalW and one by EulerAlign, are shown in figure 7. We argue that these mis-alignments are due to the improper scoring functions used by each program. Since the alignment parameters have been tuned optimal in both programs (for all 20 sequence sets instead of for each set individually), we conclude that EulerAlign made a more reasonable alignment than ClustalW in this case. Because the real alignments for DNA sequences are unknown, we hope the sum of pair scoring scheme demonstrates, to some extent, the robustness and high performance of EulerAlign. More significantly, EulerAlign computed each alignment of the test sets within 20 seconds and obtained a comparable result to ClustalW, which used 10 minutes on the same machine.

EulerAlign is an efficient alignment tool for mutiple DNA sequence alignment. By incorporating additional information, such as quality values, EulerAlign is able to provide a fast and sensitive way for automated re-sequencing analysis in many biological applications.

3 Discussion

The Eulerian path approach is an efficient and accurate solution to both the DNA fragment assembly problem and the multiple sequence alignment problem. It applies the de Bruijn graph structure that stores the sequence and similarity information simultaneously with economic memory requirements. By breaking sequences into overlapping k-tuples and tracing sequence paths in the corresponding de Bruijn graph, similarity among sequences or even a rough multiple sequence alignment is readily available in a resolution of k consecutive matches. All repeat regions shorter than sequence length are easily detectable by following sequence paths, and the superpath transformation effectively solves these repeat structures. In an assembly problem, an Eulerian path visiting each edge exactly once is extracted in a linear time with respect to the size of fragment sets. In a MSA problem, a heaviest path representing the consensus sequence is obtained in a linear time with respect to the size of aligned sequences. This linearity makes the Eulerian path approach extremely efficient in dealing

with large datasets. Although ad hoc in nature, the Eulerian path approach to the DNA fragment assembly outperforms many other assembly algorithms in the perspective of accuracy. As for the multiple sequence alignment, the Eulerian path approach is at least as good as ClustalW in both simulated and real DNA sequence sets but requires very much less time in computation. Although best fitted to the alignment of sequences within one family, EulerAlign can be extended to align sequences from different families (the algorithm is in progress). In conclusion, we believe that the Eulerian path idea and its combinitorial framework can be fit into many practical problems in computational biology.

4 Acknowledgements

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

References

[1] Ewing B. and Green P. 1998. Base-calling of automated sequencer traces using phred. II. error probabilities. *Genome Research.* **8:** 186-194.

[2] Fleischner H. 1990. *Eulerian Graphs and Related Topics*, Elsevier Science, London.

[3] Green P. 1994. Documentation for Phrap. *http://bozeman.mbt.washington.edu/phrap.docs/phrap.html.*

[4] Higgins D.G. and Sharp P.M. 1989. Fast and sensitive multiple sequence alignments on a microcomputer. *CABIOS.* **5:** 151-153.

[5] Idury R. and Waterman M.S. 1995. A new algorithm for DNA sequence assembly. *J. Comp. Biol.* **2:** 291-306.

[6] Parkhill J., Achtman M., James K.D., Bentley S.D., Churcher C., Klee S.R., Morelli G., Basham D., Brown D., Chillingworth T., Davies R.M., Davis P., Devlin K., Feltwell T., Hamlin N., Holroyd S., Jagels K., Leather S., Moule S., Mungall K., Quail M.A., Rajandream M.A., Rutherford K.M., Simmonds M., Skelton J., Whitehead S., Spratt B.G. and Barrell B.G. 2000. Complete DNA sequence of a serogroup A strain of Neisseria meningitidis Z2491. *Nature (London)*, **404:** 502-506.

[7] Pevzner P.A., Tang H. and Waterman M.S. 2001a. A new approach to fragment assembly in DNA sequencing. *In Proceedings*

of the Fifth International Conference on Computational Biology (RECOMB 2001, Montreal), 256-267.

[8] Pevzner P.A., Tang H. and Waterman M.S. 2001b. An eulerian path approach to DNA fragment assembly. *Proc. Natl. Acad. Sci. USA.* **98:** 9748-9753.

[9] 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. *Nucleic Acids Res.* **24:** 4876-4882.

[10] 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. *Nucleic Acids Res.* **22:** 4673-4680.

[11] Zhang Y. and Waterman M.S. 2003. An Eulerian path approach to global multiple alignment for DNA sequences. *J. Comp. Biol.*, in press.

Figure 1: (a) DNA sequence with a triple repeat R; (b) the overlap graph; (c) construction of the de Bruijn graph by gluing repeats; (d) de Bruijn graph. Reprinted from Pevzner et al. 2001b.

Figure 2: (a) Two DNA fragments and their 3-tuples (for simplicity, their reverse complements are not included); (b) edge and vertex presentation of those 3-tuples; (c) a de Bruijn graph by "gluing" identical edges and vertices.

Figure 3: Analogy between (a) DNA fragment assembly problem and (b) multiple sequence alignment problem. (a) and (b) are similar except the positional distribution of fragments.

Figure 4An example of de Bruijn graph constructed from 6 sequences of 100 bp each. (a) the initially constructed graph has many cycles. (b) After removing self-cycles, the heaviest acyclic path appears as thick edges that are visited by many sequence paths. Diamond vertices correspond to the sequence ends.

Figure 5: SP and AA scores (distance score) with respect to the number of sequences. The squares and triangles indicate different pairwise similarities (square-90%, triangle-70%). Both SP and AA scores are computed from a single alignment test. Solid lines connect points from

EulerAlign, where dashed lines connect points from ClustalW.

Figure 6: Left: Linear time cost (in seconds) by EulerAlign with respect to the number of sequences; three lines correspond to 90%,80% and 70% pairwise similarities. Right: Comparison to the quadratic time cost by ClustalW (dashed lines). The tested numbers of sequences are 10, 15, 50, 100, 250, 500.

Figure 7: Part alignments of sequence set At_000000541 by (a) EulerAlign and (b) ClustalW. In the first part, EulerAlign made a correct alignment whereas ClustalW did not. Alignments by two programs look very different but they are indeed the same region. In the second part, ClustalW made a correct alignment whereas EulerAlign didn't. Sequence order in (a) is adjusted to the same order in (b). The bottom sequence is an outlier. The visualization tool is ClustalX (Thompson et al. 1997).

Table 1: Comparison of alignments by EulerAlign and ClustalW on *Arabidopsis* sequences.

			With Quality		Without Quality	
Set	N	L	EulerAlign	ClustalW	EulerAlign	ClustalW
At_000000166	96	665	355.8	356.4	396.7	390.5
At_000000244	96	677	588.4	588.5	598.6	597.9
At_000000245	84	677	263.6	262.6	322.2	301.1
At_000000296	94	687	480.2	480.1	508.0	504.1
At_000000300	96	541	478.7	477.2	483.6	481.0
At_000000308	95	623	502.9	503.5	524.4	520.4
At_000000325	87	676	262.4	264.6	348.6	330.5
At_000000331	96	716	455.8	455.2	480.2	461.8
At_000000383	93	639	364.6	368.5	410.2	398.7
At_000000397	92	680	277.3	279.2	353.9	327.0
At_000000403	69	644	384.7	381.9	440.7	416.2
At_000000454	95	556	499.3	500.0	504.0	502.7
At_000000459	95	731	630.5	632.7	648.3	643.7
At_000000466	87	719	586.3	585.6	621.8	614.4
At_000000504	95	575	530.8	532.2	536.5	535.3
At_000000541	95	760	413.9	414.8	456.4	447.3
At_000000550	95	760	358.1	358.3	391.8	375.0
At_000000584	87	715	259.7	250.5	344.7	313.6
At_000000603	83	695	338.6	337.7	410.5	405.4
At_000000689	92	724	564.6	560.9	592.6	588.0

N is the number of sequences. L is the average sequence length. Scoring functions are (match, mismatch, gapopen, gapextention) = (1, 0, -4, -1). All scores are normalized by $\frac{N(N-1)}{2}$.

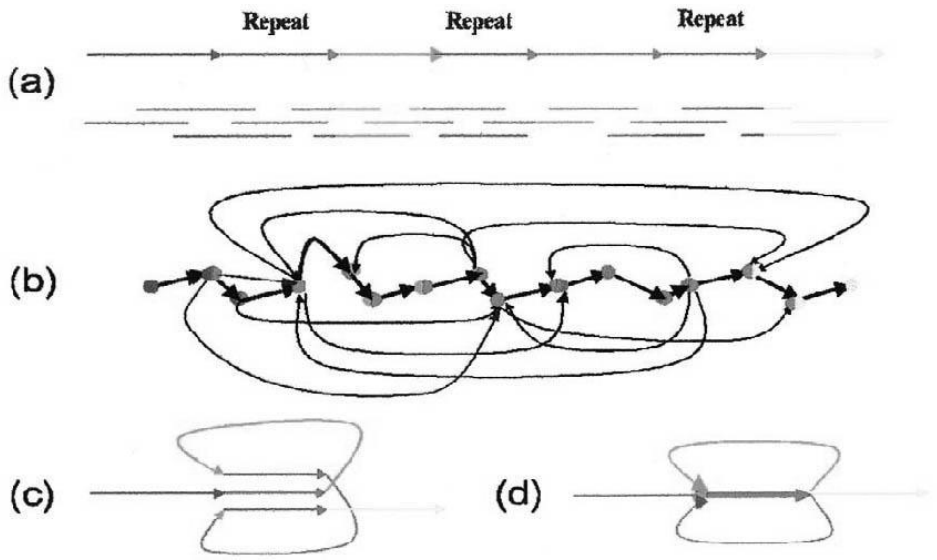


Figure 1:

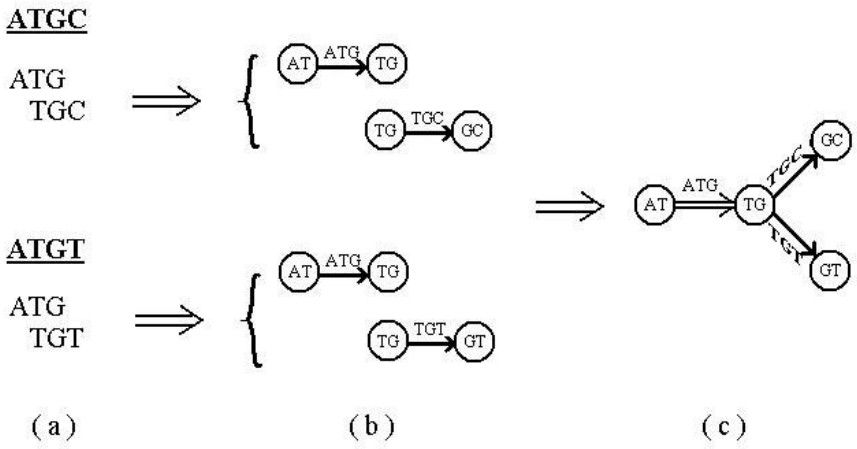

ATGC
ATG
TGC
AT
ATG
TG
TG
TGC
GC
ATGT
ATG
TGT
AT
ATG
TG
TG
TGT
GT
AT
ATG
TG
TGC
GC
TGT
GT
(a)
(b)
(c)

Figure 2:

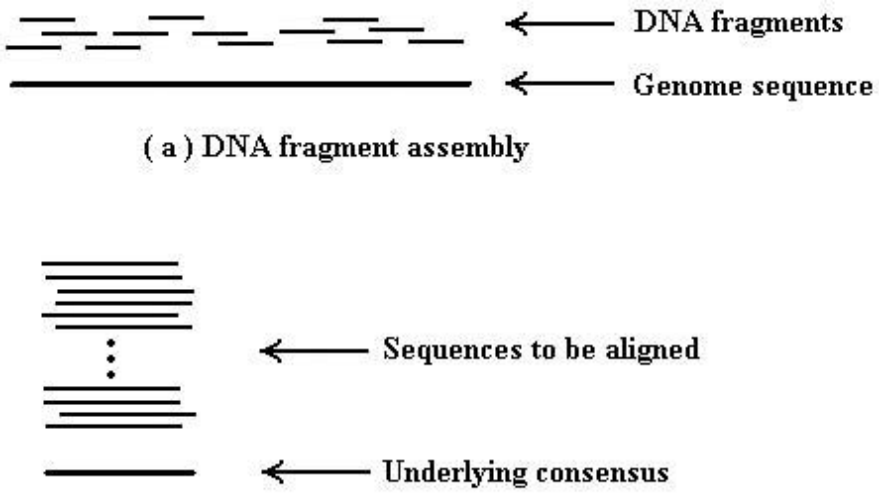
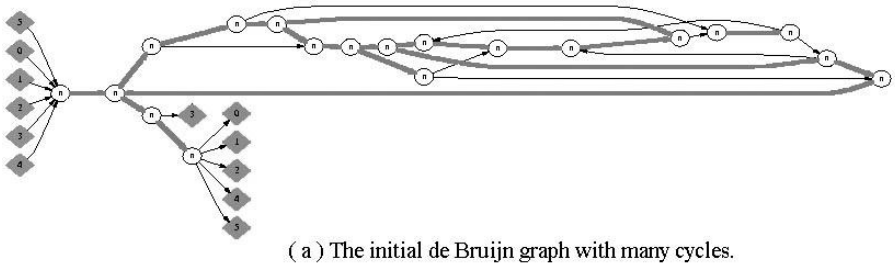
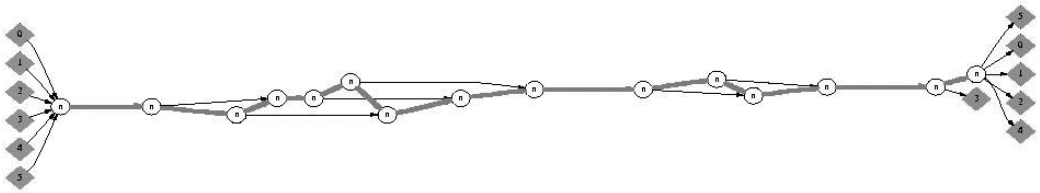

DNA fragments
Genome sequence
(a) DNA fragment assembly
Sequences to be aligned
Underlying consensus
(b) Multiple sequence alignment

Figure 3:



(a) The initial de Bruijn graph with many cycles.



(b) The transformed graph without cycles.

Figure 4:

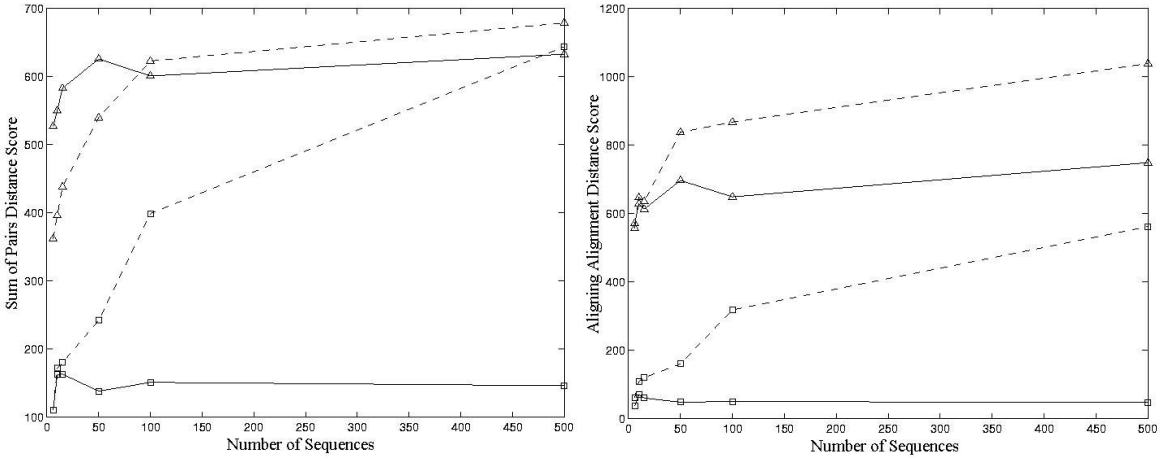

Sum of Pairs Distance Score
Number of Sequences
Aligning Alignment Distance Score
Number of Sequences

Figure 5:

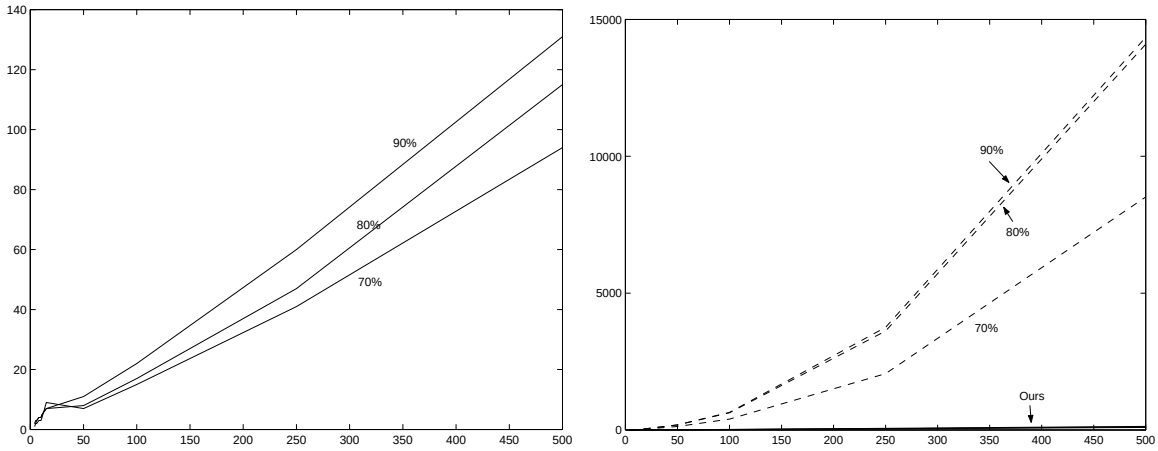
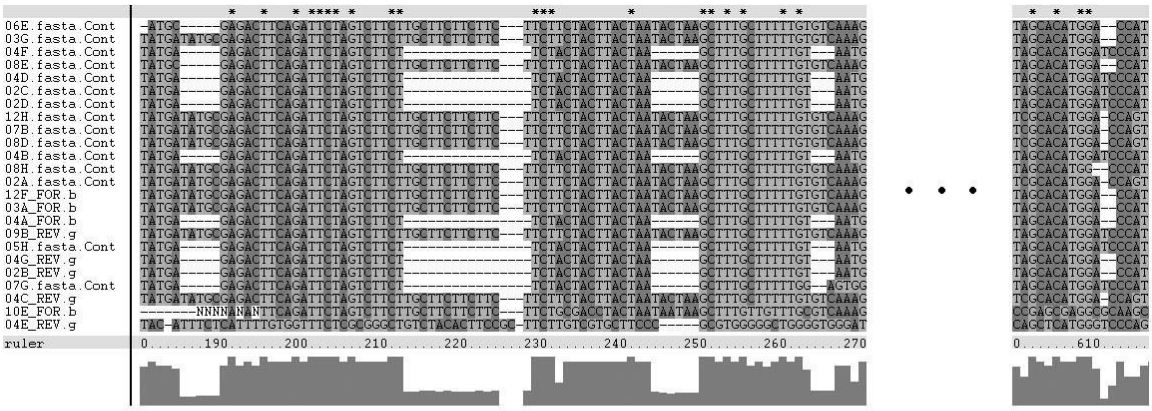
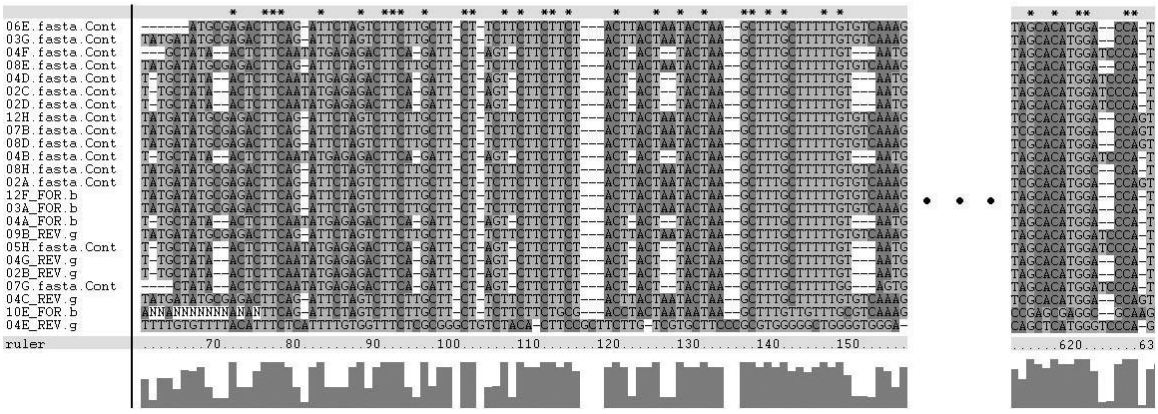

140
120
100
80
60
40
20
0
0
50
100
150
200
250
300
350
400
450
500
90%
80%
70%
15000
10000
5000
0
0
50
100
150
200
250
300
350
400
450
500
90%
80%
70%
Ours

Figure 6:



(a) EulerAlign Alignment



(b) ClustalW Alignment

Figure 7: