

PLUMMER: An Almost Linear Approach to Multiple Whole Genome Alignment *

Jitender S. Deogun, Fangrui Ma, Jingyi Yang

Department of Computer Science and Engineering
University of Nebraska – Lincoln
Lincoln, NE 68588-0115, USA
deogun{fma, jyang}@cse.unl.edu

Abstract

Following advances in biotechnology, many new whole genome sequences are becoming available every year. A lot of useful information can be derived from the alignment and comparison of different genomes. However, most of the current research focuses on pairwise genome alignment, and few available applications can efficiently align multiple genomes. In this paper, we present an optimal approach to multiple whole genome alignment. Our algorithm aligns closely related multiple genomes to find their conserved regions. We present two approaches to align whole genomes. In first approach, called Direct Matching (DM), multiple whole genomes are aligned with their DNA sequences. However, because most parts of prokaryotic genomes are encoded regions, we introduce second method, Functional Matching (FM), to especially align multiple prokaryotic genomes. We present graph theoretic argumentation to prove that our approach has almost linear complexity. In addition, analyses of experimental results for both methods are presented. It may be noted that the FM method generates a greater number of and longer conserved regions than the DM method and thus conveys detailed and accurate information about the conservation and inheritance of genomes.

Keywords: Multiple whole Genome Alignment, Conserved Regions, Prokaryotic Genomes, Cocomparability Graphs, Linear Complexity.

1 Introduction

With advances in biotechnology, more and more whole genome sequences begin to be available. In June 2002, 93 complete genomes were available. At present researchers are actively working on sequencing 284 prokaryotic genomes and 195 eukaryotic genomes [2]. The availability of whole genome sequences is opening great possibilities for understanding the evolutionary process.

Whole genome alignment is relatively new, however, research in this area is advancing fast and several methods have become available in recent years, e.g. PIPMaker [13], and SSAHA [11].

*This research was supported in part by NSF EPSCoR Grant No. EPS-0091900 and NSF Digital Government Grant No. EIA-0091530.

These use different approaches to achieve the objectives, e.g. dynamic programming in DIALIGN [9], hashing in ASSIRC [15], and suffix trees in MUMmer [3]. In this paper, we develop PLUMMER, an almost linear approach to multiple whole genome alignment for finding conserved regions.

Whole genome alignment between pairs of strains from a single species often reveal evidence of extensive regions that abruptly interrupt synteny [12]. The inheritance patterns and diversity within these regions holds significant information regarding the nature of small and large-scale evolutionary events that shaped the genomes. One successful approach for pairwise alignment of the entire genome sequences is MUMmer [3]. The MUMmer software system, developed by Delcher et al., combines suffix tree, *longest increasing subsequence (LIS)*, and Smith-Waterman algorithms to align a pair of whole genomes. Recently, Brudno and Morgenstern [1] proposed an anchored alignment approach. The LIS algorithm [6] is used to find the highest scoring monotonically increasing subsequence of pairwise alignment. The chained seeds are considered as anchor points. The fields between the anchor points are aligned using DIALIGN 2 [10]. However, both these methods are designed for finding pairwise alignment.

Aligning multiple genomes and finding their conserved regions can reveal significant information for bioscientific discovery. Such information can provide a better view of the genetic inheritance and polymorphic relationship among several organisms. To our knowledge, no efficient algorithm has been proposed for multiple whole genome alignment. Many algorithms are only suitable for aligning short sequences, but not sequences at genome level. One approach for multiple sequence alignment is Hidden Markov Models (HMMs) [5]. It is a statistical method that needs a large set of training data to construct species specific models of genomes. Thus, this method is not practical for multiple whole genome sequence alignment. Widely used FASTA and BLAST tools do not give satisfactory results even for two genomes [3]. An alternative way to align multiple genomes is to select one genome, and align all others against it. However, it is difficult to work with, inefficient and unpredictable.

In this paper, we develop a new software system, PLUMMER, for multiple whole genome alignment. Development of PLUMMER is based on a general approach that combines suffix trees and graph theoretic formulation for the alignment of multiple whole genomes. We show that overall complexity of PLUMMER approach is almost linear. We develop two methods, *Direct Matching (DM)* and *Functional Matching (FM)* to find the conserved parts. FM is suitable for aligning prokaryotic genomes. Furthermore, short subsequences between conserved regions can be further aligned with existing multiple sequence alignment tools, like DIALIGN [9].

2 Preliminaries

In this section, we review and introduce concepts related to suffix trees, Maximal Unique Match, and graph theory.

2.1 Suffix Trees and MUMs

A suffix tree τ for an n -character sequence S is a rooted directed tree with exactly n leaf nodes numbered 1 to n . Each internal node, other than the root, has at least two children and each edge

is labeled with a nonempty subsequence of S . No two edges out of a node can have edge labels beginning with the same character. The key feature of the suffix tree is that for any leaf node i , the concatenation of the edge labels on the path from the root to the node i exactly spells out the suffix of S that starts at position i . That is, it represents the subsequence $S[i..n]$ [6].

A suffix tree can also be used to represent suffixes of a set of sequences $\{S_1, S_2, S_3, \dots, S_i\}$. Such a suffix tree is called a *generalized suffix tree*. To handle a set of sequences, a distinct terminal symbol is attached to the end of each sequence, and all sequences are concatenated to form a long sequence. A suffix tree is then built for this long sequence to represent suffixes of all sequences in this set.

A naive algorithm to build a suffix tree takes $O(n^2)$ time, where n is the length of the sequence [6]. However, there are several linear algorithms for building a suffix tree, for example, see Ukkonen [14]. These algorithms can also be used to construct a generalized suffix tree.

Definition 1. A *MUM* is a subsequence that occurs only once in each sequence of a set of multiple sequences. The symbols bounding a MUM in all the sequences must not be the same [3].

A generalized suffix tree can be used to find MUMs among a set of sequences.

2.2 Trapezoidal Graphs

By $G = (V, E)$, we denote a finite, undirected and simple graph. For standard graph theory notations not given here we refer to [7].

Definition 2 ([4]). A *k-level trapezoidal diagram* $\mathcal{D}(G)$ for a graph $G = (V, E)$ assigns to each vertex v of G a collection of intervals

$$\mathcal{T}(v) = \langle [l_v^i, r_v^i] : l_v^i, r_v^i \in \{1, 2, \dots, 2n\}, l_v^i < r_v^i, i \in \{1, 2, \dots, k\} \rangle$$

such that for each $i \in \{1, 2, 3, \dots, k\}$ and any of different vertices $v, w \in V$ the intervals $[l_v^i, r_v^i]$ and $[l_w^i, r_w^i]$ have no endpoints in common. Furthermore, $\{v, w\} \in E$ iff either there is an $i \in \{1, 2, 3, \dots, k\}$ such that $[l_v^i, r_v^i]$ and $[l_w^i, r_w^i]$ have nonempty intersection or there are $i, j \in \{1, 2, 3, \dots, k\}$ such that $l_v^i < r_v^i < l_w^j < r_w^j$ and $l_v^j < r_v^j < l_w^i < r_w^i$.

To understand the above definition, we show the following visualization of a k -level trapezoidal diagram. k parallel horizontal lines L_1, L_2, \dots, L_k are drawn with m points distributed on each line. A vertex $v \in V$ is a polygon Q_v , a *k-trapezoid*, obtained by joining consecutive points in the chain $l_v^1, l_v^2, \dots, l_v^d, r_v^d, r_v^{d-1}, \dots, r_v^1, l_v^1$. $\{v, w\} \in E$ if and only if Q_v and Q_w have nonempty intersection.

Definition 3. A graph G is a *k-dimensional trapezoidal graph* if it has a *k-level trapezoidal diagram*.

Figure 1 gives an example of a k -dimensional trapezoidal graph and the corresponding k -level trapezoidal diagram. In an earlier paper [4], we establish relationship between k -trapezoidal and cocomparability graphs.

Theorem 1 ([4]). *The k-trapezoidal graphs are exactly the cocomparability graphs of partially ordered sets of interval dimension at most k.*

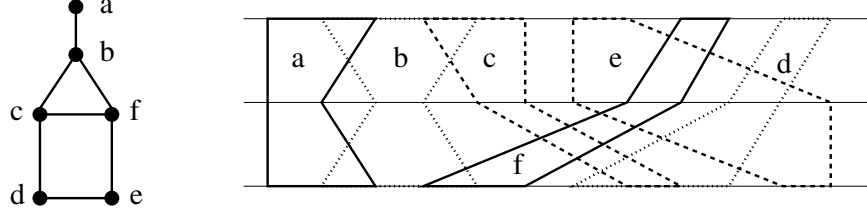


Figure 1: An example of a 3-level trapezoidal diagram and corresponding 3-dimensional trapezoidal graph.

Cocomparability graphs are complements of comparability graphs, and thus a maximum independent set of a k -trapezoidal graph can be found by finding a maximum clique in comparability graphs. The maximum clique problem is NP-complete in general. However, an $O(|V|^3)$ algorithm for finding maximum clique in comparability graphs can be found in [7] and a linear time algorithm was presented recently in [8]. The theorem below follows from [8].

Theorem 2 ([8]). *A maximum clique of a comparability graph can be found in $O(|V| + |E|)$, i.e. linear time.*

2.3 Longest Increasing Subsequence (LIS) of MUMs and MUM Graphs

After all the MUMs of a set of sequences have been identified, each of the MUMs is labeled with an integer from $\{1, 2, 3, \dots, m\}$, according to their positions in the first sequence, where m is the number of MUMs. This label is a unique identifier of a MUM. The MUMs may appear in different order in different sequences according to their positions in the corresponding sequences.

Definition 4. A *MUM sequence* is a permutation of MUM identifiers $\{1, 2, 3, \dots, m\}$, which corresponds to the order of MUMs in a particular sequence.

Definition 5. A *Longest Increasing Subsequence* of a set of *MUMs*, denoted by *LIS-MUMs*, is the largest subset of the MUMs which appear in ascending order in each of the MUM sequences.

To find LIS-MUMs, we introduce the concepts of *MUM diagrams and MUM graphs*, and show that the class of *MUM graphs* is same as the class of *trapezoidal graphs*. We represent a MUM sequence by nodes on a horizontal line, where each node represents a MUM in the sequence. Such a horizontal line is called a *MUM line*. The concepts of MUM diagrams and MUM graphs are defined as follows:

Definition 6. A *k-level MUM diagram* is a diagram depicting the structure of MUMs in different MUM sequences. It consists of k MUM lines. The MUMs with the same identifier on different MUM lines are joined together with line segments resulting in a *MUM chain*, see Figure 2. A MUM chain for MUM x is denoted by C_x .

Definition 7. A *k-dimensional MUM graph*, $G_{MUM} = (V_{MUM}, E_{MUM})$, is the intersection graph of a k -level MUM diagram, where the vertex set, V_{MUM} , is the set of MUM identifiers. Two vertices u and v are adjacent, i.e. $(u, v) \in E_{MUM}$, if and only if C_u and C_v intersect. A 3-level MUM diagram, along with corresponding MUM graph, G_{MUM} , is given in Figure 2.

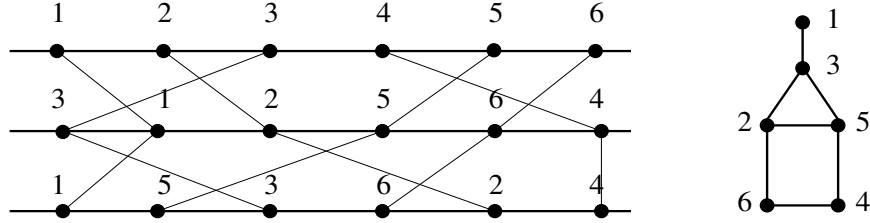


Figure 2: An example of a 3-level MUM diagram and corresponding 3-dimensional MUM graph.

Lemma 3. A k -dimensional MUM graph can be constructed in $O(km^2)$ time from a k -level MUM diagram (or from k MUM sequences), where m is the number of MUMs.

It is easy to see that the MUM diagram is, in fact, a k -level trapezoidal diagram. The proposition below follows immediately.

Proposition 4. The class of k -dimensional MUM graphs is same as the class of k -dimensional trapezoidal graphs.

It follows that a MUM graph, G_{MUM} , is a cocomparability graph. Therefore LIS-MUMs for a set of MUMs can be found by finding a maximum independent set in the MUM graph or equivalently by finding maximum clique in corresponding comparability graphs. The theorem below follows from the Theorems 1, 2 and Lemma 3.

Theorem 5. The time complexity of finding LIS-MUMs for a set of k MUM sequences, each of length m , is $O(km^2)$, $k \geq 3$.

3 The PLUMMER Algorithms

In our approach, the genomes are treated as sequences. A MUM in a set of sequences may indicate a conserved region among the sequences. Therefore, LIS-MUMs of multiple genomes give a maximum set of conserved regions of the group of multiple genomes. The main problem, thus, is to find the LIS-MUMs among several sequences.

In our algorithm, we use Direct Matching (DM) and Functional Matching (FM) methods to find conserved parts from a set of genomes. FM method is especially used for prokaryotic genomes. It only focuses on the conserved parts on the coding regions of the prokaryotic genomes. The main difference between the DM and FM algorithms is that FM adds pre-processing and post-processing to DM. In the following, we describe details of the DM and FM algorithms.

3.1 Direct Matching (DM) Method

The main problem in DM method is to find the LIS-MUMs for k MUM sequences. We first find the MUMs, and use the MUM diagram to define a MUM graph, G_{MUM} . As described in the previous section, the problem of finding LIS-MUMs is solved by finding a maximum independent set of the MUM graph, G_{MUM} . The DM algorithm can be divided into 3 steps.

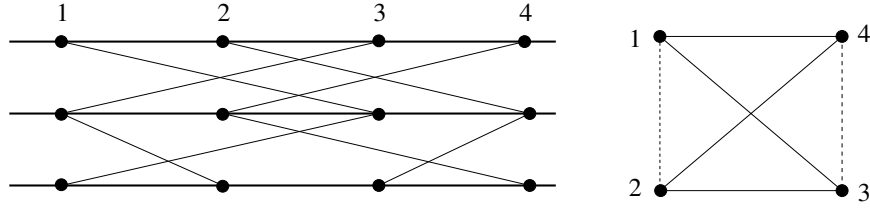


Figure 4: A 3-level MUM diagram and its MUM graph, G_{MUM} . The dashed lines denote $\overline{G_{MUM}}$.

In our example, the maximum clique in $\overline{G_{MUM}}$ (see Figure 4) is $\{1, 2\}$, or $\{3, 4\}$, which is the independent set in G_{MUM} . So the LIS for sequences S_1, S_2 and S_3 is $\{a, d\}$, or $\{bc, e\}$. They are actually the conserved regions if we consider the sequences S_1, S_2 and S_3 as genomes.

This step requires $O(m + e)$ time, where m is the number of MUMs and e is the number of edges in G_{MUM} .

3.2 Functional Matching (FM) Method

In DM method, we identify and align all MUMs. It is possible that a MUM in a genome that is in a functional region is aligned with a MUM in another genome that is not in a functional region. This may result in misleading information. Furthermore, we want to achieve some flexibility in alignment by allowing error in MUMs. The FM method is thus developed to find conserved regions in the coding regions of genomes. This method includes pre-processing of protein sequences, modification of DM algorithm, and post-processing of aligned protein MUMs. The details of the FM method are discussed below.

3.2.1 Pre-processing Step

Proteins can be coded in both directions, forward or backward. For convenience, we call them *forward or backward* proteins.

Definition 9. A *forward Virtual Genome Sequence (VGS)* is obtained by concatenating all the forward protein sequences in a genome. A *backward VGS* is defined similarly.

In the pre-processing step, we construct a forward VGS and a backward VGS for each genome. The relationship between the proteins and genes, along with the positions of genes in the genome, is saved for the post-processing step.

3.2.2 Modification of the DM Algorithm

In the FM method, the forward and backward VGSs are grouped separately to form *forward* and *backward VGS sets*. Since the character set of DNA sequences is different from the character set of protein sequences, the DM program is modified accordingly. Each set of VGSs is aligned just like the genome sequences using the modified DM algorithm.

Definition 10. An *amino acid MUM subsequence (AMS)* is a MUM found in a set of VGSs.

Definition 11. An AMS is called a *forward* (or *backward*) *AMS* if it is found from the forward (or backward) set of VGSs. An AMS is called a *crossing AMS* if some of its occurrences are composed of amino acids from more than one proteins. Otherwise, it is called a *simple AMS*.

LIS-AMSs — LIS of AMSs, are obtained with the modified DM algorithm. LIS-AMSs of a set of forward (or backward) AMSs are called forward (or backward) LIS-AMSs.

3.2.3 Post-processing Step

Definition 12. A *Functional Segment Match (FSM)* is the DNA subsequence corresponding to an AMS in a set of LIS-AMSs.

In this step, the LIS-AMSs are mapped back to their coding genome sequences. Simple forward or backward AMSs can be mapped back easily. But a crossing AMS needs special attention, and must be split into several sub-AMSs based on composing protein sequences. In some applications, it may be meaningful to discard crossing AMSs.

4 Time Complexity Analysis

The time complexity of constructing a suffix tree and finding all MUMs of a set of k genomes is $O(n)$, where n is the total length of all input genome sequences. The next step is to transform a k -level MUM diagram into a k -dimensional trapezoidal graph. The time complexity of defining a k -dimensional trapezoidal graph, G_{MUM} , from a k -level MUM diagram is $O(m^2)$, where m is the number of MUMs and k , the number of sequences, can be treated as a constant. A maximum clique of the complement graph $\overline{G_{MUM}}$ can be found in $O(m + e)$, where e is the number of edges in $\overline{G_{MUM}}$. These cover all the steps in the DM method. As the input graph is G_{MUM} , we conclude that the complexity of the DM algorithm is dominated by $O(m^2)$.

It is commonly accepted that $m \ll n$, the total length of all input genome sequences [3]. Following our empirical results, it may not be unreasonable to assume that $m^2 \leq c * n$, where c is a constant and n is the total length of the group of whole genomes. Therefore, the overall time complexity of DM method can be deemed as linear, i.e. $O(n)$ for finding the conserved regions for multiple whole genome alignment.

To determine the time complexity of FM, we need to analyze only the pre-processing and post-processing steps. It can be easily seen that the pre-processing in FM needs $O(n)$ time. The post-processing can be finished in $O(m^2)$ time, where m now is the number of AMSs. Therefore, the time complexity of FM is same as that of DM.

5 Experimental Results

The software system, PLUMMER, can be used to study properties of conserved regions and inheritance of the microorganisms. It can also identify the regions that differentiate one strain from another. A divide and conquer approach can then be used to locate specific small regions, which can be aligned using an existing multiple alignment tool, such as DIALIGN [9]. In this section, we

discuss some preliminary experimental results of the DM and FM methods. A detailed study of extensive experimental results will be reported in a later paper. The experiments were performed using microorganisms listed in the table below.

<i>Label</i>	<i>Microorganisms</i>	<i>Label</i>	<i>Microorganisms</i>	<i>Label</i>	<i>Microorganisms</i>
A	<i>L. monocytogenes</i>	D	<i>S. pneumoniae</i>	G	<i>E. coli</i> K12
B	<i>L. innocua</i>	E	<i>S. pyogenes</i>	H	<i>E. coli</i> O157H7
C	<i>S. aureus</i>	F	<i>L. lactis</i>	I	<i>E. coli</i> O157H7 EDL933

5.1 Experimental Results for the DM Method

Two types of experiments were performed on different groups of microorganisms. First, we aligned three closely related strains of *E. coli*, and then we aligned four different groups consisting of 3 to 6 less similar genomes.

The experimental results for three closely related strains G, H, and I of *E. coli* are shown in Figure 5. In PLUMMER approach, we impose a threshold of 50 bps on the minimum length of a MUM. The DM algorithm finds 20918 MUMs, and the length of the longest MUM is 2632 bps. As we can observe from Figure 5(a), there are only a few inconsistent or scattered MUMs. Accordingly, the number of LIS-MUMs found is 20667. It should be noted that the number of MUMs changes with the threshold of MUM, as the threshold increases, the total number of MUMs decreases. As shown in Figure 5(b), the LIS-MUMs are perfectly aligned on a smooth diagonal line. The only obvious break between bp positions of 1 and 2 millions indicates that the major difference among three strains is located in this region.

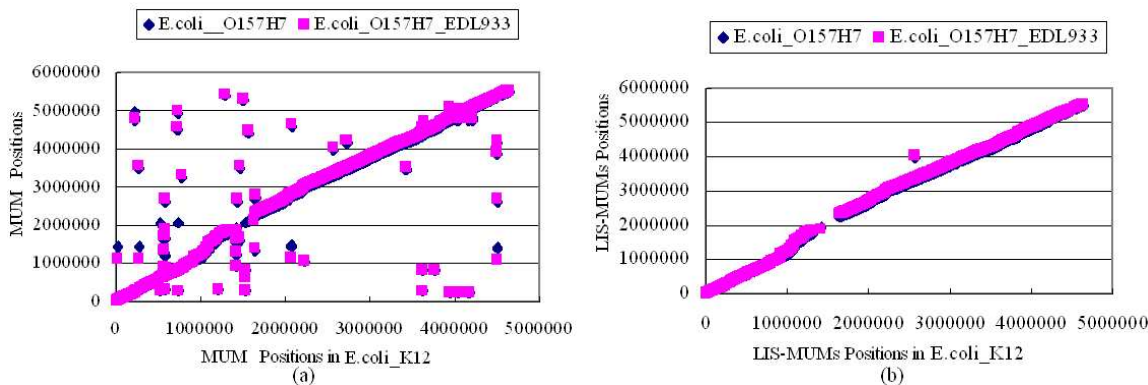


Figure 5: The graph (a) shows the plot of MUMs and (b) that of LIS-MUMs. Each point in the graphs corresponds to a MUM. The x-axis is the positions in *E. coli*_K12, y-axis is the positions in other genomes.

The experimental results for different species and group sizes are summarized in Table 5.1. It is observed that the number of MUMs as well as that of LIS-MUMs decreases quickly as the size of the group increases. The ratio of the number of MUMs generated to that of LIS-MUMs is considerably greater in group of size three than in larger groups. Moreover, the ratio decreases at a decreasing rate. In the first group, many of the MUMs may be random matches. But when the size

Group No.	Group Genomes	Threshold Length	MUM Number	Maximum MUM Length	LIS-MUMs Number
1	A B C	15	10432	151	608
2	A B C D	15	608	88	155
3	A B C D E	15	196	88	116
4	A B C D E F	15	161	88	104

Table 1: Experimental statistics for four experiments on different groups of genomes.

of the group increases, the random subsequences are much less likely to be identified as MUMs. The length of a MUM greatly influences its probability of being an LIS-MUM. That is, the longer a MUM is, more probable it is to be an LIS-MUM.

The graphs in Figure 6 show the alignments of the LIS-MUMs of the groups of size three and four. The trends of alignments are not clear when the size is greater than four. As the number of the genomes in a group increases, the conserved parts become shorter and fewer.

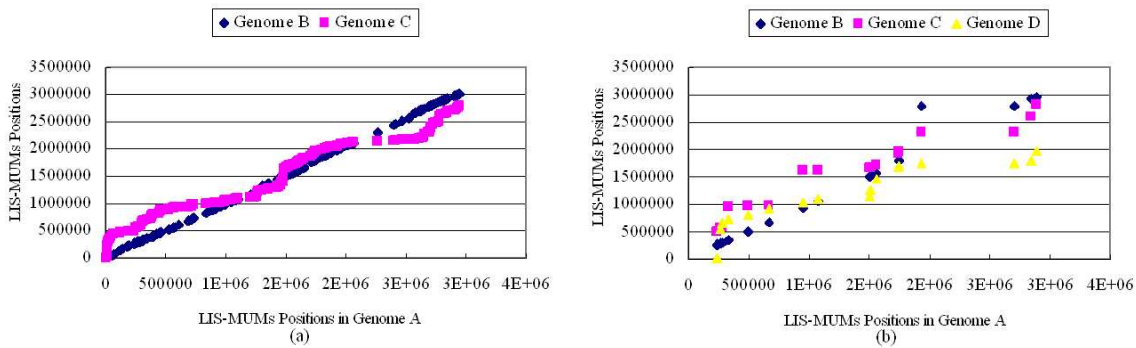


Figure 6: The points in each graph represent LIS-MUMs. The x-axis is the positions in genome A, and the y-axis denotes the positions in other genomes of the group.

5.2 Experimental Results for the FM Method

A group of three prokaryotic genomes A, B, and C was used to perform the experiments with the FM method. The threshold of the AMSs is set to seven amino acids and that of MUMs to 21 bps in order to make the results of the DM and FM experiments comparable. Figure 7 gives a flavor of differences between the DM and FM methods, where a comparison of LIS-MUMs in the DM method with forward FSMs in the FM method is shown.

The number and size of FSMs found with the FM method are much larger than those of MUMs found with the DM method. Figure 7 shows that the curve for FSMs in (b) is much smoother than that for MUMs in (a). From the experiments, it can be seen that the FM method gives better results than the DM method.

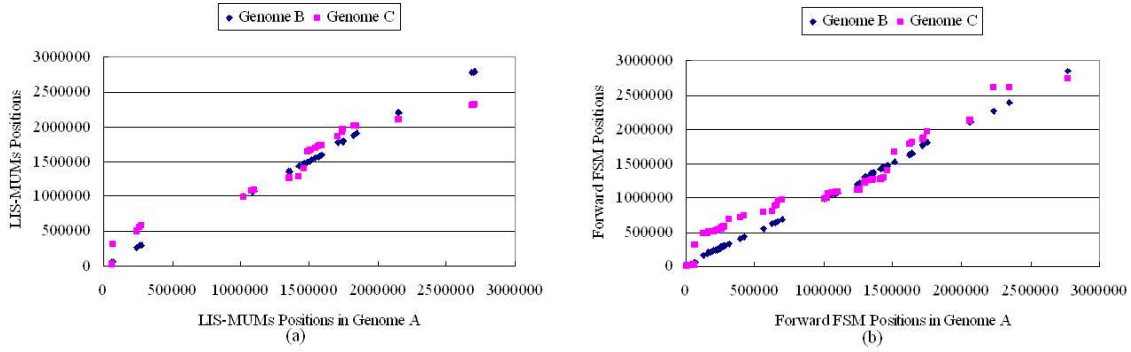


Figure 7: The graph (a) shows the alignment of MUMs in three prokaryotic genomes, and (b) right graph displays the alignment of forward FSMs in the same three genomes.

FM outputs a greater number of and longer FSMs that conveys detailed information about conservation and inheritance of prokaryotic genomes. This is because, different codons can code the same amino acid, which emulates the effect of approximate matching. Since this method only aligns the functional genome parts, it is more efficient, and the results lead to greater biological insight.

6 Conclusions

In this paper, we developed a new software system, PLUMMER, for multiple whole genome alignment. Alignment of multiple whole genome sequences is an important problem, because it can provide significant information about inheritance and polymorphism of multiple whole genomes.

We present an almost linear approach to find conserved regions of multiple whole genome sequences. Development of PLUMMER is based on a general approach that combines suffix trees and graph theoretic formulation for multiple whole genome alignment. We present graph theoretic argumentation to prove that our approach has almost linear complexity. Two methods are developed. First, Direct Matching (DM) method is suitable for alignment of multiple whole genomes with their DNA sequences. Second, Functional Matching (FM) method is suitable for alignment of multiple prokaryotic genomes.

Extensive experiments were conducted to demonstrate effectiveness of PLUMMER system for alignment of multiple whole genomes, but only a small set of experimental results is presented here. Both the DM and FM methods successfully find the conserved parts among a group of multiple whole genomes. The FM method generates greater number of and longer conserved regions as compared to DM method for multiple prokaryotic genomes.

References

- [1] M. Brudno and B. Morgenstern, Fast and sensitive alignment of large genomic sequences. *Proceedings of IEEE Computer Society Bioinformatics Conference, Stanford University, CA*, pp. 138–147 August, 2002
- [2] H. Chuong, Genome Analysis and Genome Comparison. *NIH/NLM/NCBI*, 2002
- [3] A.L. Delcher, S. Kasif, R.D. Fleishmann, J. Peterson, O. White, and S.L. Salzberg, Alignment of whole genomes. *Nuc. Acids. Res.*, **27**, pp. 2369–2376, 1999
- [4] J.S. Deogun, T. Kloks, D. Kratsch, and H. Müller, On the Vertex Ranking Problem for d-trapezoid Graphs and for Circular-arc Graphs. *Discrete Appl. Math.*, **98**, pp. 39–63, 1999.
- [5] S. Eddy, Multiple alignment using hidden Markov models, *Proc. of the third international conf. on Intelligent systems for molecular biology*, pp. 114–120, AAAI Press, 1995
- [6] D. Gusfield, *Algorithms on Strings, Trees, and Sequences—Computer Science and Computational Biology*, Cambridge University Press, British, 1999.
- [7] M.C. Golumbic, *Algorithmic graph theory and perfect graphs*, Academic Press, New York, 1980.
- [8] R.M. McConnell and J.P. Spinrad, Modular decomposition and transitive orientation. *Discrete Mathematics*, **201**, pp. 189–241, 1999.
- [9] B. Morgenstern, K. Frech, A. Dress, T. Werner, DIALIGN: Finding local similarities by multiple sequence alignment. *Bioinformatics*, **14**, pp. 290–294, 1998
- [10] B. Morgenstern, DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, **15**, pp. 211–218, 1999
- [11] Z. Ning, et al, SSAHA: a fast search method for large DNA databases. *Genome Research*, **11(10)**, pp. 1725–9, 2001
- [12] N.T. Perna, G. Plunkett III, V. Burland, B. Mau, J.D. Glasner, D.J. Rose, G.F. Mayhew, P.S. Evans, J. Gregor, H.A. Kirkpatrick, G. Posfai, J. Hackett, S. Klink, A. Boutin, Y. Shao, L. Miller, E.J. Grotbeck, N.W. Davis, A. Lim, E.T. Dimalanta, K.D. Potamouisis, J. Apodaca, T.S. Anantharaman, J. Lin, G. Yen, D.C.Schwartz, R.A. Welch, and F.R. Blattner, Genome sequence of enterohaemorrhagic *Escherichia coli* O157 H7. *Nature*, **409**, pp. 529–533, 2001.
- [13] S. Schwartz, Z. Zhang, K.A. Frazer, et al, PipMaker-A Web Server for Aligning Two Genomic DNA Sequences. *Genome Research*, **10**, pp. 577–586, 2000
- [14] E. Ukkonen, On-line construction of suffix-trees. *Algorithmica*, **14**, pp. 249–260, 1995.
- [15] P. Vincent, et al, A strategy for finding regions of similarity in complete genome sequences, *Bioinformatics*, **14**, pp. 715–725, 1998
- [16] M. Waterman, *Introduction to Computational Biology*, Chapman & Hall, London, 1995.