A New Model for Global Multiple Alignment of Whole Genome Sequences

Yue Ma[1], Cao An Wang

Department of Computer Science, Memorial University of Newfoundland,
St. John's, Newfoundland, Canada, A1B 3X5
[1]Current address: IBM China, 10/F Shui On Plaza, 333 Huai Hai Zhong Road,
Shanghai, China, 200021

{yuema, wang}@cs.mun.ca

Abstract

We propose a new anchor-based model for global multiple alignment of whole genome sequences. The model includes three main phases. Firstly, an enhanced suffix array method is employed to find anchors. Next, a novel chaining strategy, which is based on the dynamic programming technique and the longest common subsequence idea, calculates an anchor-chain for the weighted anchors. Lastly, the progressive multiple alignment method is used to close the gaps between the anchors. The proposed chaining procedure is based on evolutionary theory and can align whole genome sequences not only for close homologs, but also distant species. Combined with the exact suffix array approach, this model can compute partially accurate solutions and generate a high-quality alignment result in terms of computation and biology.

Keyword: multiple genome sequence alignment, anchor-based alignment, enhanced suffix array, multiple heaviest common subsequence.

I. Introduction

With the benefit of advanced biotechnology, large numbers of whole genome sequences have been compiled. To compare whole genome sequences, biologists increasingly need alignment methods that are both efficient enough to handle large numbers of long sequences, and accurate enough to correctly align the conserved biological features of distant species present in the sequences. Aligning whole genome sequences is a fundamentally different problem than aligning short sequences. Recently, intensive research activities have been devoted to this problem. Many available whole genome alignment software systems have been developed [5]. MUMmer [8] detects and aligns every difference between two microbial genomes. DIALIGN is combined with CHAOS [4], which was developed for rapid identification of chains of local pairwise sequence similarities, to handle large and complicated genome sequences. MGA [9] is a widely used program to produce a global multiple alignment for closely related whole genomes. MAUVE [7] is the first alignment system that integrates analysis for large-scale evolutionary events with traditional multiple sequence alignment. AVID [2] is a global alignment program for large genomic regions up to the megabase range and MAVID [3] is developed based on AVID to assemble multiple genome sequences according to a progressive ancestral alignment that incorporates preprocessed constraints. So far, most programs work efficiently in aligning small numbers of closely related genome sequences. Very few genome alignment programs can align distant homologs, and they usually cannot work efficiently for large numbers of genome sequences. Recently, the tools that can align not only close homologs, but also distant ones, are desired by the biologists, because they can test how related the species are to obtain the idea of evolution process, i.e., which species come out first. This idea may lead biological suppositions to the evolution in the future. The tools also can help to test the probability of different species sharing the same ancestor in order to making a prefect evolutionary tree. The program can also help biologist to design primer for PRC (polymorphism chain reaction), which is an important method used for the work of genetics and molecular biology. In order to make up for the lack of methods for aligning distantly related genome sequences, we propose a new strategy with biological reasons: the genome sequences from close homologs are first selected for assembly, and then distantly related genome sequences are appended to the anchor alignment iteratively. Together with our chaining algorithm involving evolutionary theory, this model generates a more accurate and meaningful anchor-chain in terms of computation and biology. It assembles flexible genome sequences and leads to a high-quality alignment result.

II. An Overview of Our Chaining Algorithm

2.1 Our Ideas and Their Origins

The anchor-based alignment approach divides initial large alignment problems into smaller, more manageable ones and combines program speed and sensitivity [4], which is a good solution for whole genome sequence alignment tasks. The procedure of the anchor-based whole genome alignment can be divided into three phases [5]:

1. Computation of all the anchors;
2. Computation of an optimal anchor-chain of collinear non-overlapping anchors: the anchors that form the basis of the alignment;
3. Alignment of the regions between the anchors.

We propose a chaining algorithm as one part of our model in the second phase. The algorithm uses the dynamic programming technique and is based on the standard Longest Common Subsequence idea [6].

The quality of a whole genome alignment method is measured not only by the running efficiency, but also by the biological significance [3] [4]. Therefore, it is important to involve biological ideas to improve the alignment quality and practicality. We place a weight on every anchor in order to find a biologically more correct anchor-chain. We believe that this idea can help our alignment model obtain a more meaningful result. After some helpful talks with biologists, we determined that our weight tends to be related to the length of the anchor. This is based on biological evolutionary theory. If the large-scale sequences are assumed to be whole genome sequences, every anchor can be considered a conserved nucleotide block. According to evolutionary theories such as natural selection, the longer the block is, the more important the evolutionary information and structure it might contain. The reason for this is that only very valuable nucleotide blocks can survive during those significant sequence changes that result from selective pressures. During evolution, there are likely certain important reasons to keep some nucleotide blocks that do not easily change. According to this idea, the longer the block is, the heavier the weight we put on it.

We refer to this chaining procedure as a problem of finding the *Multiple Heaviest Common Subsequence (MHCS)* or the *Multiple Maximum Weight Common Subsequence (MMWCS)*, which is the common subsequence with maximum weight in multiple weighted sequences.

2.2 Computational Complexity

Given a finite sequence $S = < s_1, s_2, ..., s_m >$, a subsequence S' of S is any sequence that consists of S with k terms deleted, for $k \in [0, m]$. Given a set $R = \{S_1, S_2, ..., S_r\}$ of sequences, a *Common Subsequence* is a sequence that is the subsequence of each sequence $S_1, S_2, ..., S_r$ in *R*. In the weight set $W = \{w_1, w_2, ..., w_t\}$, $w_1, w_2, ..., w_t$ are the real numbers associated with each character in those sequences.

Definition 1 *Multiple Maximum Weight Common Subsequence (MMWCS)* problem or *Multiple Heaviest Common Subsequence (MHCS)* problem: Given a multiple sequence set $R = \{S_1, S_2, ..., S_r\}$ with a particular weight *w* assigned to every character of each sequence *S*, what is the common subsequence with the maximum weight, i.e., what is the *MHCS(R)*?

The instance is: given a set of sequence $R = \{S_1, S_2, ..., S_p\}$ and a weight set $W = \{w(x_1), w(x_2), ..., w(x_t)\}$ for the alphabet of *R*, $\Sigma(R)$, whose size $|W| = |\Sigma(R)|$. Clearly $|\Sigma(R)| \le m_1 + m_2 + ... + m_p$, where $m_i = |S_i|$. $\|MHCS(R)\|$ represents the weight of the heaviest common subsequence of *R*..
The decision version of the problem is: given an integer k, a listing of the sequences in R and a listing of the weights in *W*, is $\|MHCS(R)\| \ge k$?

Theorem 1 (COMPLEXITY) The decision version of the *Multiple Heaviest Common Subsequence* problem belongs to NP-Complete.

Proof: First, it is easy to see that $MHCS \in NP$, since a nondeterministic algorithm need only guess a *k* and check in polynomial time whether the weight of the heaviest common subsequence is larger than or equal to *k*, after the weights have been assigned to the alphabets.
Next, we use the restriction technique. We restrict the *MHCS* problem for $\Sigma(R)$ of arbitrary size by allowing only instance with weight 1 in $\Sigma(R)$. Then, the restricted *MHCS* problem becomes the Longest Common Subsequence (LCS) problem for $\Sigma(R)$ of arbitrary size. In other words, the LCS problem is a special case of the *MHCS* problem. When an arbitrary number of sequences are considered, the LCS problem has been proved to be NP-Complete [13].
Therefore, the *Multiple Heaviest Common Subsequence (MHCS)* problem is NP-Complete.

2.3 Algorithm Description

The *MHCS* problem is NP-Complete, which means that no polynomial time algorithm exists for this problem unless P = NP. Moreover, with regard to the theory of parameterized complexity, an approach to attack intractable problems mainly developed by Doweny and Fellows, the fixed alphabet longest common subsequence parameterized in the number of strings (FLCS) has recently been proved to be *W*[1]-hard [14]. Therefore, we can say that, in general, no exact polynomial-time algorithm can find an exact anchor-chain from arbitrary numbers of weighted sequences. However, traditionally, for all the genome alignment programs, the number of the input sequences is forced to be limited to ignore the computational complexity. We limit the numbers of the input genome sequences, therefore, this algorithm can find the result in polynomial time. We propose an algorithm for solving the *MHCS* problem with the idea of extending the dynamic programming technique of the standard longest common subsequence method [12].
The running time of calculating the weight of the *MHCS* is Θ (X_1.length $\cdot$ X_2.length$\cdot \ldots \cdot$ X_k.length) and the running time of printing the *MHCS* is Θ(X_1.length + X_2.length + $\cdots$ + X_k.length) [12].

2.4 Implement and Results

The algorithm is implemented by JAVA and the running results show that different weights assigned to different characters of the input sequences lead to different output solutions [12].

III. The Whole Procedure of Our Model

3.1 Our Ideas and Their Origins

The input of whole genome alignment model is usually assumed to be relatively conserved genome sequences.

In the first phase, we use the enhanced suffix array method to find the conserved blocks among the input genome sequences. Because these conserved blocks are more likely to belong to the global alignment, they are used as anchors for assembling the multiple genome alignment.

In the second phase, we first weigh the anchors based on their lengths. Next, we use our chaining algorithm to find the *heaviest common subsequence* as the anchor-chain. Then, all the anchors are assembled based on this anchor-chain. In our model, we propose a novel alignment method to assemble the anchors. This method will make our model more flexible for different input sequences and user requirements. After consulting with biologists who are currently using sequence alignment tools to help their evolutionary experiments, we realize that a tool for aligning the genome sequences of distantly related species and assembling large numbers of genome sequences are desired. Therefore, we use a different aligning structure to assemble the anchors. For small numbers of closely related genome sequences, this model uses our chaining algorithm to find the anchor-chain and obtain an alignment from all anchors, which is the same as most alignment programs. However, when the inputs are many genome sequences from distantly related species, the model will use a new strategy: it asks users to choose the genome sequences that are from close homologs (i.e. from closely related species). Then, it uses the chaining algorithm to find an anchor-chain from these chosen sequences. Afterward, those unselected anchor sequences append to the anchor alignment iteratively based on the anchor-chain. This idea makes our model suitable for aligning not only closely related genome sequences but also distantly related ones, and it helps our model to align even large numbers of input genome sequences. Moreover, this method will lead to an evolutionary more correct and meaningful anchor-chain. Because the inputs are genome sequences, every anchor found in the first phase consists of nucleotides. For closely related species, these nucleotide blocks are very likely to represent the same or similar traits that are beneficial to evolutionary research. However, for distantly related species, though the constituent nucleotides are the same, these blocks may not represent similar traits. In evolution, the anchors/nucleotide blocks from the closely related species may come from the same ancestor and be very meaningful, but those from the distantly related species may be just a result of unexpected mutation. If the anchor-chain is computed from all the anchor/nucleotide blocks from both closely related and distantly related genome sequences, the procedure will chain the anchors that have the same components together; however, this anchor-chain may only have structural meaning but not any evolutionary meaning. Hence, for genome sequences at any evolutionary distance, our strategy produces an evolutionary more correct anchor-chain that leads to a high-quality alignment result.

In the last phase, gaps between the anchors are further aligned by an existing progressive global multiple alignment tool to generate a detailed sequence alignment.

3.2 Phase 1: Find Multi-MUMs as Anchors

A MUM is defined a maximal unique match decomposition of two genomes in the program MUMmer [8]. In our model, we define the maximal unique match for multiple genomes as multi-MUMs. Because of the assumption that input genome sequences are highly similar, a large number of multi-MUMs are assured to be identified. Aligning Multi-MUMs is the basic step for aligning multiple whole genomes.

Definition 2 A *multi-MUM* is a *maximal unique match* decomposition of multiple genomes. It occurs exactly only once in each sequence of a multiple sequence set and is not contained in any longer such sequence. The two characters bounding a multi-MUM must be mismatches in all the sequences.

In order to find the multi-MUMs from the input genome sequences, we use the enhanced suffix array algorithm [1], which is a suffix array enhanced with a table for longest common prefixes. The enhanced suffix array method requires much less space than the widely used suffix tree method and much less time than other programs for genome analysis task [1].

Employing the enhanced suffix array algorithm to deal with four input sequences S_1 = abeadc, S_2 = edbcaba, S_3 = cabed and S_4 = dcabea, we detect four multi-MUMs: (ab), (c), (d), (e). Each of them is labeled with an integer from $\{1,2,3,\ldots,m\}$, according to their positions in the first input sequence. Obviously, *m* is the number of the multi-MUMs and the integers are assigned as the indices of each of them. The indices are unique identifier of each multi-MUMs. In different input sequences, Multi-MUMs appear in different order according to their positions but the indices are always unique.

Therefore, each input sequence can be represented by the multi-MUMs and the gaps between them on a horizontal line. We use the corresponding index to represent each multi-MUM and ignore the gaps in this step. So, each input sequence can be transformed to a sequence consisting of the indices, which is defined as a *multi-MUM index sequence*.

A *multi-MUM index sequence* for the input sequence S_i is denoted by I_i. In the example, the four multi-MUMs are labeled as: 1=(ab), 2=(e), 3=(d), 4=(c). So the four input sequences can be transformed to four *multi-MUM index sequences*.

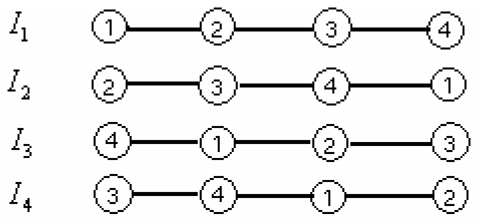

Fig. 1. The *multi-MUM index sequences* of input sequences.

3.3 Phase 2: Find the Multiple Heaviest Common Subsequence as Anchor-chain to Align Anchors

The inputs of this phase are the *multi-MUM index sequences* that we got in phase 1. Based on the evolutionary relationship among the original genome sequences, certain numbers of *multi-MUM index sequences* are chosen to calculate the anchor-chain. We weight the multi-MUMs based on their length, and then use our chaining algorithm to find the *heaviest common subsequence* to be the anchor-chain. Different associated weights will result in different anchor-chains. The weights containing evolutionary information lead to a biologically more meaningful anchor-chain.

In the example here, we choose the first three sequences as the candidate sequences to compute the anchor-chain. “2” is weighed to the multi-MUM 1, “1” is weighed to the multi-MUM 2, “1” is weighed to the multi-MUM 3 and “1” is weighed to the multi-MUM 4. After running our program, we find the heaviest common subsequence of the first three sequences is the multi-MUM 1. Therefore, we choose the multi-MUM 1 to be the anchor-chain and assemble the three anchor-computing sequences according to the selected anchor-chain.

The *multi-MUM index sequences*, which are not selected to calculate the anchor-chain, are aligned according to this chain. That is: place the multi-MUMs, which have the same characters as the anchor-chain, to the anchor-chain column. Based on this procedure, an alignment for all the anchors is assembled.

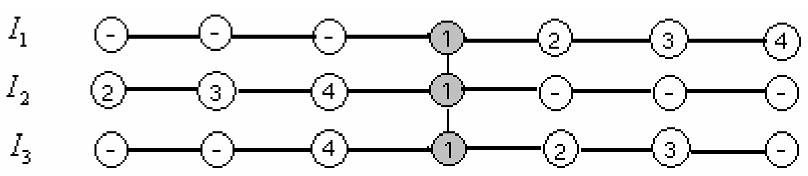

Fig. 2. The alignment of the anchor-computing *multi-MUM index sequences*: Three anchor-computing *multi-MUM index sequences* I_1, I_2, I_3 are aligned according to the selected anchor-chain multi-MUM 1.

In the example, I_4 is appended to the alignment based on the multi-MUM 1. Accordingly, an alignment of all the four *multi-MUM index sequences* is obtained.

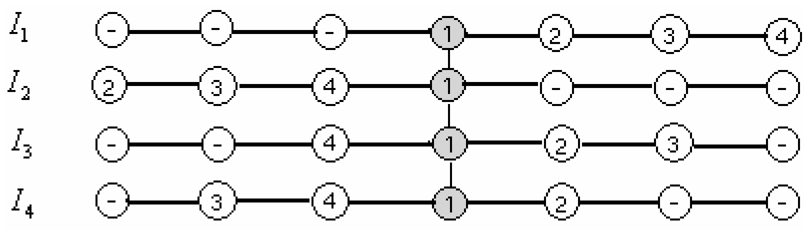

Fig. 3. The alignment of all *the multi-MUM index sequences*: the fourth multi-MUM index sequence I_4 is appended to the alignment of the three anchor-computing multi-MUM index sequences according to the anchor-chain multi-MUM 1.

Therefore, the alignment of the anchors/Multi-MUMs is:

--	--	--	a	b	E	d	c
e	d	c	a	b	--	--	--
--	--	c	a	b	E	d	--
	d	c	a	b	E		

Fig. 4. The alignment of all the anchors

3.4 Phase 3: Close Gaps and Get Detailed Alignment

The progressive global alignment program CLUSTAL W [15] is used to align the gap regions between the anchors to generate detailed alignment. Because the target sequences are whole genomes, which are large-scale sequences, a threshold is set for the maximum length of the gaps to evaluate whether they should be align or not. If the length of a gap is out of the threshold, the gap will be ignored. For our example, the alignment result is:

--	--	--	--	a	b	E	a	d	c
e	d	b	c	a	b	--	a	--	--
--	--	--	c	a	b	E	--	d	--
--	d	--	c	a	b	E	a	--	--

Fig. 5. The alignment result: the progressive global alignment method is used to align the characters in the gaps; together with the aligned anchors, the alignment result is obtained.

3.5 Time Complexity Analysis

In the first phase, a suffix array can be directly constructed in linear time [10]. The *lcp* array and the *ps* array can be obtained from the suffix array in linear time [11]. Hence, in the first phase, constructing a suffix array and computing all the multi-MUMs of input sequences requires linear time: O(n), where n is the total length of all the input genome sequences. With the finite automata algorithm [6], the anchor sequences can be transferred to *multi-MUM index sequence* in linear time. In the second phase, the chaining algorithm for *k multi-MUM index sequences* works in $O(m^k)$ time, where m is the length of the *multi-MUM index sequence*. Then, it takes $O(k'm)$ time for the remaining k' sequences to be appended to the alignment. The running time of the third phase depends on the threshold set by the user.

IV. Conclusions and Future Work

We presented a new anchor-based model for the global multiple alignment of whole genome sequences. Firstly, we proposed a chaining algorithm, which is based on the dynamic programming technique and weighs each anchor by a proper weight according to evolutionary theory. This algorithm finds the *heaviest common subsequence* among the weighted anchor sequences. Though we proved the *MHCS* problem is NP-complete, the algorithm works in polynomial time for limited sequence inputs. We verified that different associated weights lead to different results by implement. Lastly, we described the whole procedure of our alignment method: first, we employed the enhanced suffix array method to find anchors; next, we used our chaining strategy to find the anchor-chain and to generate the alignment of the anchors; finally, we used the progressive multiple alignment tool CLUSTAL W to close the gaps. In the second phase of this procedure, in order to make up for the lack of methods for aligning distantly related genome sequences, we proposed a novel strategy: the genome sequences from close homologs are selected to assemble first, and then distantly related genome sequences are appended to the anchor alignment iteratively. This phase produces a more meaningful and accurate anchor alignment in terms of both computation and biology. It helps our model to assemble more genome sequences at any evolutionary distance. Combined with the exact suffix array approach in the first phase, this model leads to a high-quality alignment result.

Often, a model can be modified and improved. We continue our research on improving and implementing this model to make it more efficient while retaining its accuracy. Furthermore, we plan to do the experiments with the real genomic sequences and compare the result to the existing systems and tools. The testing genomic sequences will be from not only close homologs, but also distant ones, which make more benefits for biological work.

Acknowledgement: This work is supported by NSERC grant OPG0041629.

References

[1] Abouelhoda M.I., Kurtz S., Ohlebusch E., “The enhanced suffix array and its application to genome analysis.” Proceeding of the second workshop on algorithms in Bioinformatics (2002), Lecture Notes in Computer Science.
[2] Bray N., Dubchak I., Pachter L. “AVID: A Global Alignment Program.” Genome Research 13 (2003), pp. 97-102.
[3] Bray N., Pachter L. “MAVID: Constrained Ancestral Alignment of Multiple Sequences.” Genome Research 14 (2004), pp. 693-699.
[4] Brudno M., Morgenstern B. “Fast and sensitive alignment of large genomic sequences.” In Proceedings IEEE Computer Science Bioinformatics Conference (2002), pp. 138-147.

[5] Chain P., Kurtz S., Ohlebusch E., Slezak T., "An applications-focused review of comparative genomics tools: Capabilities, limitations and future challenges." Briefings in Bioinformatics Vol. 4 No.2 (2003), pp. 105-123.
[6] Tomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, Introduction to Algorithms. The MIT Press, 2001.
[7] Darling A., Mau B., Blattner F., Perna N. "Mauve: Multiple Alignment of Conserved Genomic Sequence With Rearrangements." Genome Research (2004).
[8] Arthur L. Delcher, Simon Kasif, Robert D. Fleischmann, Jeremy Peterson, Owen White and Steven L. Salzberg. "Alignment of whole genomes." Nucleic Acids Research Vol.27, No.11 (1999), pp. 2369-2376.
[9] Hohl M., Kurtz S., Ohlebusch E. "Efficient multiple genome alignment." Bioinformatics Vol.18 (2002) S312-S320.
[10] Karkkainen J., Sander P. "Simple Linear Work Suffix array Construction." ICALP, LNCD 2719 (2003), pp. 943-955.
[11] Kasai T., Lee G., Arimura H., Arikawa S., Park K. "Linear-Time Longest-Common-Prefix Computation in Suffix array and its Applications." CPM, LNCS 2089 (2001), pp. 181-192.
[12] Yue Ma "An Anchor-based Model for Global Multiple Alignment of Whole Genome Sequences." M.Sc. thesis. (2005).
[13] David Maier "The Complexity of Some Problems on Subsequences and Supersequences." Journal of the Association for Computing Machinery (1978).
[14] Pietrzak K. "On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems." Journal of Computer and System Sciences 67 (2003), pp. 757-771.
[15] Julie D. Thompson, Desmond G. Higgins, Toby J. Gibson. "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice." Nucleic Acids Research, Vol.22, No. 22 (1994), pp. 4673-4680.

Miss Yue Ma obtained her B.Sc degree in Computer Science from Tongji University, Shanghai China, and M.Sc in Computer Science from Memorial University of Newfoundland, Canada.

Miss Ma is currently working in IBM China Company Limited, Shanghai branch. Her research interest includes design and analysis of algorithms, bioinformatics.

Dr. Cao An Wang is a professor in the department of Computer Science, Memorial University of Newfoundland, Canada. His research interest includes design and analysis of algorithms, computational geometry, bioinformatics, and graph drawing.

Dr. Wang obtained his BA degree in Physics from Peking University, China, and both M.Sc and PhD degrees in Computer Science from the University of Alberta, Canada in 1983 and 1988, respectively.