Synteny mapping seeks an alignment of entire chromosomes or genomes across species. Write a research proposal focused on the problems of creating these maps. What differentiates the creation of syntenic alignments from other sequence alignment algorithms, such as Needleman-Wunsch or FASTA? What technique would you use to create a synteny map between two species?

Swathi Yadlapalli

ABSTRACT

As the number of organisms whose genomes that are mapped have increased the question of how those genomes align assumes greater importance. In this paper I describe an algorithm to align the entire genome sequences of eukaryotic and prokaryotic organisms. I use an efficient data structure called a suffix tree to rapidly align sequences containing millions of nucleotides. The central idea of the algorithm takes two input sequences, either DNA or proteins, and finds all unique subsequences longer than a specified minimum length k that are identical between the two inputs. The algorithm can also find syntenic chromosomal regions, genomic duplications and strain-to-strain comparisons.

DIFFERENCES BETWEEN GENOME AND SEQUENCE ALIGNMENT

There are highly efficient algorithms for alignment of protein sequences, implemented in systems like BLAST [7, 8] and FASTA [9, 10]. There has been vast amount of research done for aligning two sequences. The earliest work done by Needleman and Wunsch [3], Smith and Waterman [4] focused on comparing single proteins or DNA sequences contatining single gene. These algorithms are mostly ineffective in aligning entire genomes. The problem here is of size: single gene sequences may be as long as tens of

thousands of nucleotides, whole genomes are usually millions of nucleotides are larger. The method for aligning large scale DNA sequences deals with millions of nucleotides. The current algorithms either run out of memory or take prolonged time to complete. Also, previous algorithms were designed mainly to discover insertions, deletions and point mutations but not to look for large scale changes that come out in whole genome comparisions, like tandem repeats.

The algorithms for sequence alignments rely on either dynamic programming or hashing techniques. Naïve versions of dynamic programming use $O(N^2)$ (N is the length of shortest sequence compared) space and time, which makes computation almost impossible for sequences of size >= 4MB. Better versions take O(N) space solving the memory problem but still taking longer times. Hashing techniques operate faster on average, but they use a 'match and extend' strategy, the extend part taking $O(N^2)$ time. More complex dynamic programming algorithms can be used for alignment when the error is expected is to be low. For example two similar sequences with atmost E differences can be aligned in time proportional to E times the length of the longer sequence. Sim3 [5] program uses an algorithm which runs in O(N) time when the sequences are highly similar and even very long. However, they do not work for whole genome alignments since the number of differences may be in the order of 10^4 or 10^5 nucleotides. Sim2 [6] uses a BLAST like hashing method to identify k-mer matches, which are extended to

maximal length matches. Then dynamic programming is used to combine them into local alignment chains.

In this paper I propose a method for aligning whole genome sequences. I assume the sequences are closely related, and hence can quickly compare sequences that are millions of nucletides long. I intend to output a base to base alignment of the sequences highlighting the differences such as single nucleotide polymorphisms (SNP), large inserts, repeats and reversals.

ALGORITHM

The algorithm uses a data structure known as suffix Tree which is used to find efficiently all distinct subsequences in a given sequence. There are 3 basic steps in the algorithm: building a suffix tree to find the maximal unique match, finding the longest increasing subsequence and filling the gaps using Smith- Waterman alignment.

Let us assume the input sequences as Genome A and Genome B. The alignment process consists of the following steps.

1. Find a maximal unique match (MUM) decomposition of the 2 genomes. An MUM is a subsequence that occurs uniquely in Genome A and Genome B and is not contained in any longer such sequence. Thus,

MUM is bounded on both sides with mismatches. The assumption is this sequence is almost certain to be part of the golbal alignment.

2. The matches found in the MUM alignment are sorted and the longest possible sets of matches that occur in the same order in both the genomes are extracted. This is done using a variation of the LIS algorithm.
3. The gaps in the alignment are closed by identifying large inserts, SMP, repeats and small mutated regions.
4. The alignment including all the matches in the MUM alignment and also the regions that do not match exactly are output as the final result.

MUM DECOMPOSITION

The goal of this step is to identify maximal unique sequences in both the genomes. The naïve algorithm matches every subsequence in Genome A with Genome B. There are $O(N^2)$ such subsequences (N is the sum of the lengths of the 2 genomes) and each match takes O(N) time using standard pattern matching methods. I can use a suffix tree to store all possible suffixes of an input sequence S. Each suffix can be obtained by traversing the unique path from the root node to leaf node. The simple brute force algorithm to construct suffix trees runs in quadratic time. I use McCreight's algorithm [12] which builds the tree in linear time by using sets of pointers. I construct a suffix tree T for Genome A and then add the suffixes for Genome B to T. Each leaf node in T is labelled to

indicate which suffix it represents in which genome, A or B. MUMs are represented by internal nodes with exactly 2 child nodes, such that the child nodes are leaf node from different genomes. The maximal matches can be identified by the presence of mismatches at their end. Thus, all MUMs can be identified in a single scan of the suffix tree. Because the tree construction and all the subsequent steps take linear time and space, the overall runnig time and space of the system is also linear.

MUM'S SORTING

After finding all the MUM's the next step is to sort them according to their position in Genome A. In the cases of transposition or reversal between the genomes, the B positions are not in ascending order. I employ a variation of the LIS algorithm to find the longest set of MUM's which occur in ascending order in both Genome A and Genome B. For example, if the order of B position is given by the sequence (1, 2, 9, 6, 8, 3, 10), the LIS is (1, 2, 6, 8, 10). The algorithmic variations deal with the lengths of MUM sequences and the fact that they can overlap. This algorithm requires O(K log K) time or O(N) time, where K is the number of MUMs.

CLOSING THE GAPS

After sorting the MUM's the local gaps need to be closed to complete the alignment. A gap is defined as an interruption in the MUM alignment and falls into one of the four classes.

1. SNP interruption: Exactly one base differs between the two sequences. In the simpler case, it is surrounded by an MUM subsequence. In other cases SNP is adjacent to sequences that are not unique. This is dealt by capturing the adjacent sequence and SNP and using the repaeat processing procedure described below.
2. Insertion: A sequence that occurs in one genome but not the other. They can be divided into two classes i) transposition are subsequences that are deleted from one location and inserted elsewhere, they appear in the MUM alignment out of sequence. ii) Simple insertions are subsequences that apppear in only one of the genomes, these can be identified as they don't appear in the MUM alignment.
3. Polymorphic region: Many mutations in a short region. If the regions are sufficiently short, I can use a standard dynamic programming algorithm to do the alignment. For very large regions, I can apply the original matching procedure recursively using a reduced minimum MUM length, if required.
4. Repeat sequence: The sequence is repeated in both the genomes. MUM alignment in these cases outputs intervals which overlaps indicating to the algorithm that a tandem repeat is present. The difference in overlap length

in the two genomes indicates how many additional repeat bases are inserted in one of the genomes.

IMPROVEMENTS

1. In addition to base-to-base comparisons, all matching protein sequences can be clustered. This helps in detecting regions of conserved synteny - multiple proteins from one organism are found in the same order and orientation in another. For each chromosome, all the proteins can be concatenated in order to create mini-proteomes. The algorithm can then be used to align each chromosome to entire genome at the protein level.
2. Algorithmic improvements: i) Reduction of amount of memory used to store suffix trees. Many nodes have more than two children, which reduces the actual memory requirement. ii) Only one sequence is stored in the suffix tree. The second sequence, also called query, is then 'streamed' against the suffix tree, exactly as if it were being added but without actually doing so. We identify where the query sequence would branch off from the tree, thereby finding all matches to the reference sequence. If the branch occurs at a tree position with just a single leaf beneath it, the match is unique in the reference sequence. Iii) Clustering matches together and then find consistent paths within each cluster.

These improvements improve the speed of the original algorithm by three times. Also, they permit the comparison of protein sequence and of multiple sequences from incomplete genomes.

SUMMARY

This paper describes the system for high resolution comparison of complete genome sequences. These techniques have been implemented in the MUMmer [1] system and used to perform complete alignments of 2 pairs of genomes: The first pair were two closely related strains of M.Tuberculosis of 4.4 million nucleotide each, the second pair were two different Micoplasma bacteria with dissimilar lenth. In the former case, the system was very useful at pinpointing the SNP's and significant insertions. In the latter case, where the organisms are not as closely related, the system is still able to align the genomes precisely. They also tested the system on even more distantly related sequences by comparing a syntenic region from the mouse and human genomes. The algorithm makes it possible to detect large-scale relationships between more distantly related organisms, which is becoming more important as more and more genomes are sequenced.

REFERENCES

1. A .L. Delcher, A. Phillippy, J. Carlton, S. L. Salzberg (2002) Fast algorithms for large-scale genome alignment and comparison. Nucleic Acids Research, Vol 30, No. 11, 2478-2483.
2. A. L. Delcher, S. Kasif, R. D. Fleischmann, S. L. Salzberg (1999) Alignment of whole genomes. NAC, Vol 27, No. 11, 2369-2376.
3. S. Needleman, and C. Wunsch. (1970) J. Mol. Biol., 48, 443-453.
4. T. Smith, and M. Waterman. (1981) J. Mol. Biol., 14. 7, 195-197.
5. K. M. Chao, J Zhang, J. Ostell, W. Miller. Comput. Appl. Biosci., 13, 75-80.
6. K. M. Chao, J Zhang, J. Ostell, W. Miller. Comput Appl Biosci., 11, 147-153.
7. S. Altschul, T. Madden, A. Schaffer, J. Zhang, D. Lipman, (1990) J. Mol Biol., 215, 403-410.
8. S. Altschul, T. Madden, A. Schaffer, J. Zhang, D. Lipman, (1997) Nucleic Acid Research, 25, 3389-3402
9. W. R. Pearson (1995) Prot. Sci., 4, 1145-1160.
10. W. R. Pearson, D. J. Lipman (1985) Science, 227, 1435-1441.
11. T. Smith, M. Waterman. (1981) J. Mol. Biol. 147, 195-197.
12. E. M. McCreight. (1976) J ACM, 23, 262-272.