Alignment of whole genomes

Arthur L. Delcher[1,2], Simon Kasif[3], Robert D. Fleischmann[4], Jeremy Peterson[4], Owen White[4] and Steven L. Salzberg[4,*]

[1]Department of Computer Science, Loyola College in Maryland, Baltimore, MD 21210, USA, [2]Celera Genomics, 45 West Gude Drive, Rockville, MD 20850, USA, [3]Department of Electrical Engineering and Computer Science, University of Illinois, Chicago, IL 60607, USA and [4]The Institute for Genomic Research, 9712 Medical Center Drive, Rockville, MD 20850, USA

Received February 2, 1999; Revised and Accepted April 14, 1999

ABSTRACT

A new system for aligning whole genome sequences is described. Using an efficient data structure called a suffix tree, the system is able to rapidly align sequences containing millions of nucleotides. Its use is demonstrated on two strains of *Mycobacterium tuberculosis*, on two less similar species of Mycoplasma bacteria and on two syntenic sequences from human chromosome 12 and mouse chromosome 6. In each case it found an alignment of the input sequences, using between 30 s and 2 min of computation time. From the system output, information on single nucleotide changes, translocations and homologous genes can easily be extracted. Use of the algorithm should facilitate analysis of syntenic chromosomal regions, strain-to-strain comparisons, evolutionary comparisons and genomic duplications.

INTRODUCTION

Since the first successful whole-genome shotgun sequence of *Haemophilus influenzae* (1), the number of organisms whose genomes have been completely sequenced has been increasing rapidly each year. As the number and variety of these genomes increase, it is becoming more common for a project to sequence the genome of an organism that is very closely related to another completed genome. For example, the genomes of *Mycoplasma genitalium* (2) and *Mycoplasma pneumoniae* (3), the third and fifth prokaryotic organisms to be completely sequenced, respectively, are very closely related and share sequence homology across large fractions of their genomes. More recently, there has been tremendous scientific interest in sequencing different strains of the same bacteria. Two strains of *Mycobacterium tuberculosis*, H37Rv (4) and CDC1551 (R.D.Fleischmann *et al*., manuscript in preparation), and two strains of *Chlamydia trachomatis*, serovar D (5) and mouse pneumonitis (Fraser *et al*., manuscript in preparation), will be completely sequenced in the near future; in each case one of the two strains is complete and the other is nearly so. It is clear that the future will see an increasing number of sequencing projects whose target is a strain or species that is closely related to an already-sequenced organism.

When the genome sequence of two closely related organisms becomes available, one of the first questions researchers want to ask is how the two genomes align. There is a large body of research, including many sophisticated algorithms, for aligning two sequences. This vast literature cannot be cited here, but important early work includes Needleman and Wunsch (6) and Smith and Waterman (7) (for recent reviews see 8,9). The focus of most prior research has been on comparing single proteins or genomic DNA sequences containing a single gene. The existing algorithms work extremely well on this task, but in most cases are ineffective in aligning entire genomes. The problem is really one of size: single gene sequences may be as long as tens of thousands of nucleotides, but whole genomes are usually millions of nucleotides or larger. When comparing a 4 Mb sequence such as *M.tuberculosis* to another 4 Mb sequence, many algorithms either run out of memory or take unacceptably long to complete. In addition, previous algorithms were designed primarily to discover insertions, deletions and point mutations, but not to look for the kinds of large-scale changes that can be discovered in whole-genome comparisons, such as differences in tandem repeats and large scale reversals.

In this paper we describe a system for pairwise alignment and comparison of very large scale DNA sequences. The algorithm assumes the sequences are closely related, and using this assumption can quickly compare sequences that are millions of nucleotides in length. It will also be able to compare entire chromosomes as large as human chromosome 1 (i.e., several hundred million basepairs), once such sequences are available, and in the process identify all differences between two different individuals.

The system is specifically designed to perform high resolution comparison of genome-length sequences. It outputs a base-to-base alignment of the input sequences, highlighting the exact differences in the genomes. It will locate all single nucleotide polymorphisms (SNPs), large inserts, significant repeats, tandem repeats and reversals, in addition to identifying the exact matches between the genomes.

We have applied this system to the CDC1551 and H37Rv strains of *M.tuberculosis*, to the two completed *Mycoplasma* genomes, and to two relatively long (225 kb) syntenic sequences from the human and mouse genomes. In the case of tuberculosis, the strains are very closely related, and the system was very useful at pinpointing the SNPs and the relatively small number of

*To whom correspondence should be addressed. Tel: +1 301 315 2537; Fax: +1 301 838 0209; Email: salzberg@tigr.org

significant insertions between these two genomes. (For the context of this discussion, the term SNP is used to mean a sequence that appears in both genomes with a difference of just one base between the two copies. Such polymorphisms may or may not represent mutations that occur in a significant percentage of the population.) In the second case, where the organisms are much less closely related (differing by hundreds of thousands of nucleotides), the system is nonetheless able to align the genomes precisely. We also tested our system on even more distantly related sequences by comparing a syntenic region from the mouse and human genomes. The results of these comparisons are described in the Results and Discussion.

In addition to allowing for comparison between different organisms, the system described here can also be put to a different use. At different stages of any large genomic sequencing project, the assembled sequence will change as gaps are closed, sequencing errors are corrected and additional sequences are completed. Because the finishing stage involves many individuals, it can be difficult for a project leader (or any one person) to get a picture of what has changed each time a genome is reassembled. The program described here can compare two different versions of a genome at different stages of sequencing and highlight precisely what has changed.

The output of the system gives a clear picture at the sequence level of all the differences between two genomes. (The code is freely available; contact the authors by email for details.) To present a more global picture, we have also developed a graphical interface that allows a researcher to scroll along the two genomes being compared and zoom in on areas of interest. (See Figure 6 for an example of what the tool displays.) The next sections describe the computational techniques employed in the system, followed by a demonstration of its use in three different comparisons: complete genomes of two strains of *M.tuberculosis*, complete genomes of two related species of *Mycoplasma*, and related 225 kb regions from mouse chromosome 6 and human chromosome 12.

THE CHALLENGE OF WHOLE GENOME ALIGNMENT

The standard algorithms for sequence alignment rely on either dynamic programming (7,10) or hashing techniques (8,11). Naïve versions of dynamic programming use $O(n^2)$ space and time (where n is the length of the shorter of the two sequences being compared), which makes computation simply unfeasible for sequences of size ≥4 Mb (such as the two *M.tuberculosis* genomes). [For an input with size n, a function X is $O(n^2)$ if, for sufficiently large n and for some constant C independent of n, $X \leq C \cdot n^2$. Informally stated, the $O(n^2)$ notation means that the amount of space and time required for the computation is no more than Cn^2.] Hashing techniques operate faster on average, but they involve a ‘match and extend’ strategy, where the ‘extend’ part also takes $O(n^2)$ time. For dynamic programming, it is possible to reduce the required space to $O(n)$ by taking more time; this solves the memory problem but still leaves one with an unacceptably slow algorithm. Faster algorithms can be developed for specialized purposes, such as a recent system for finding tandem repeats (12). This repeat finder uses a k-tuple hashing algorithm and couples it with a stochastic pattern matching strategy.

More complex dynamic programming methods can be used for alignment when the alignment error is expected to be low. For example, one can align two similar sequences with at most E differences (or errors) in time proportional to E times the length of the longer sequence. The *sim3* program (13) uses a linear time algorithm that works well when the input sequences are highly similar; it runs very fast even on very long sequences. Unfortunately, this class of algorithms does not always work for whole genome alignments, since the ‘errors’ may include multiple large inserts on the order of 10^4 or 10^5 nucleotides. As we demonstrate below, the number of differences may be greater than 100 000 despite the fact that the genomes (in this case *M.genitalium* and *M.pneumoniae*) are closely related and can in fact be aligned with one another.

Another system developed to align long sequences is *sim2* (14). This system uses a BLAST-like hashing scheme to identify exact k-mer matches, which are extended to maximal-length matches. These maximal matches are then combined into local alignment chains by a dynamic programming step. In contrast, our suffix-tree approach directly finds maximal matches that are unique. These matches can then be easily ordered to form the basis of an alignment that can span even very long mismatch regions between the two input genomes.

The system described here was developed in response to our own efforts as part of sequencing strain CDC1551 of *M.tuberculosis*; we realized it was essential to describe all the differences between CDC1551 and the recently completed H37Rv strain (4). The well-known and widely used BLAST (15,16) and FASTA (8,11) systems are not designed to perform large scale alignment of genomes, and our attempts to use these did not produce all the information we needed. It is possible, of course, to align two genomes gene-by-gene, or to align shorter pieces and concatenate all the results. By assuming that the two input sequences are closely related, our algorithm can perform large scale alignments quickly and precisely; the result is a very detailed and inclusive base-to-base mapping between the two sequences.

ALGORITHMIC METHODS

The basis of the algorithm is a data structure known as a suffix tree, which allows one to find, extremely efficiently, all distinct subsequences in a given sequence. The first efficient algorithms to construct suffix trees were given by Weiner (17) and McCreight (18), and this data structure has been studied extensively for more than two decades (9). For the task of comparing two DNA sequences, suffix trees allow us to quickly find all subsequences shared by the two inputs. The alignment is then built upon this information.

Our system uses a combination of three ideas: suffix trees, the longest increasing subsequence (LIS) and Smith–Waterman alignment (7). The novelty of the system derives from the integration of these ideas into a coherent system for large-scale genome alignment. We focus here on the high-level design of the system and exclude some of the low-level algorithmic details; those details can be found in the references.

The inputs to the system are two sequences, which for convenience we refer to as Genome *A* and Genome *B*. Note that any sequences can be provided as input (in fact, we have a modified version of the system that handles protein sequences), but we will use DNA for the purposes of discussion. We assume the sequences to be compared are closely homologous. In particular, we assume that there is a mapping between large subsequences of the two inputs, presumably because they evolved from a common ancestor. The main biological features that the system identifies are as follows. (i) SNPs, defined here as a single mutation ‘surrounded’ by two matching regions on both sides of

```
Genome A:  tcgatcGACGATCGCGGCCGTAGATCGAATAACGAGAGAGCATAAcgactta
Genome B:  gcattaGACGATCGCGGCCGTAGATCGAATAACGAGAGAGCATAAtccagag
```

Figure 1. A maximal unique matching subsequence (MUM) of 39 nt (shown in uppercase) shared by Genome *A* and Genome *B*. Any extension of the MUM will result in a mismatch. By definition, an MUM does not occur anywhere else in either genome.

the mutation. (ii) Regions of DNA where the two input sequences have diverged by more than an SNP. (iii) Large regions of DNA that have been inserted into one of the genomes, for example by transposition, sequence reversal or lateral transfer from another organism. (iv) Repeats, usually in the form of a duplication that has occurred in one genome but not the other. The repeated regions can appear in widely scattered locations in the input sequences. (v) Tandem repeats, regions of repeated DNA that might occur in immediate succession, but with different copy numbers in the two genomes. The copy numbers do not have to be integers; e.g., a repeat could occur 2.5 times in one genome and 4.2 times in the other.

The alignment process consists of the following steps, which are described in more detail in subsequent sections.

(i) Perform a maximal unique match (MUM) decomposition of the two genomes. This decomposition identifies all maximal, unique matching subsequences in both genomes. An MUM is a subsequence that occurs exactly once in Genome *A* and once in Genome *B*, and is not contained in any longer such sequence. Thus, the two character positions bounding an MUM must be mismatches, as shown in Figure 1. The crucial principle behind this step is the following: if a long, perfectly matching sequence occurs exactly once in each genome, it is almost certain to be part of the global alignment. (Note that a similar intuition is behind the hashing method upon which FASTA and BLAST are based.) Thus, we can build the global alignment around the MUM alignment. Because of the assumption that the two genomes are highly similar, we are assured that a large number of MUMs will be identified.

MUMs on both DNA strands are identified; this allows the system to identify sequences from one genome that appear reversed in the other genome.

(ii) Sort the matches found in the MUM alignment, and extract the longest possible set of matches that occur in the same order in both genomes. This is done using a variation of the well-known algorithm to find the LIS of a sequence of integers. Thus, we compute an ordered MUM alignment that provides an easy and natural way to scan the alignment from left to right.

(iii) Close the gaps in the alignment by performing local identification of large inserts, repeats, small mutated regions, tandem repeats and SNPs.

(iv) Output the alignment, including all the matches in the MUM alignment as well as the detailed alignments of regions that do not match exactly.

The system, which is called MUMmer, is packaged as three independent modules: suffix tree construction, sorting and extraction of the LIS, and generation of Smith–Waterman alignments for all the regions between the MUMs. The last step can easily be replaced with another alignment program if a user wishes. In the ensuing sections we elaborate further on each of these steps.

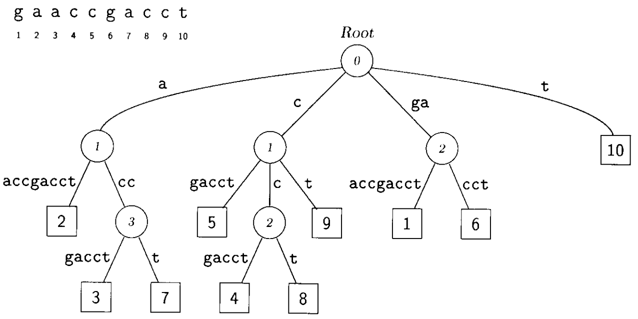

Figure 2. Suffix tree for the sequence gaaccgacct. Square nodes are leaves and represent complete suffixes. They are labeled by the starting position of the suffix. Circular nodes represent repeated sequences and are labeled by the length of that sequence. In this example the longest repeated sequence is acc occurring at positions 3 and 7.

Maximal unique matching subsequence decomposition

As mentioned above, identification of MUMs is the key step in the alignment. By identifying the sequences that occur only once in each genome we can complete the alignment by closing the gaps between the aligned MUMs.

The problem of finding a set of maximal unique matching strings (subsequences) in two very long sequences is by no means computationally trivial. The naïve algorithm for this problem will imply matching every subsequence in Genome *A* with Genome *B*. There are $O(n^2)$ such subsequences (where n is the sum of the lengths of the two genomes), and each match requires approximately $O(n)$ time using standard pattern matching methods.

Fortunately, we can employ an ingenious computational data structure introduced by Weiner (17) called a suffix tree. An example of a suffix tree for the string gaaccgacct is shown in Figure 2.

As the name implies, a suffix tree is a compact representation that stores all possible *suffixes* of an input sequence *S*. A suffix is simply a subsequence that begins at any position in the sequence and extends to the end of the sequence. Each suffix in *S* can be located by traversing a unique path in the tree from the root node to a leaf node. In other words, each leaf node represents a unique suffix. A sequence of length *N* has *N* suffixes, one starting at each sequence position, so the tree must have *N* leaves, and therefore at most *N*–1 internal nodes since each internal node has at least two child nodes. Note that each internal node in a tree corresponds to a repeated sequence in the original genome, where the repeat number equals the number of leaf nodes underneath that node in the tree. [Recently suffix trees have also been used to help discover regulatory elements in genomic yeast sequences (19). For other applications of suffix trees to sequence analysis, see Gusfield (9).]

The simple, brute-force algorithm to construct suffix trees runs in quadratic time; this is no faster than dynamic programming and, as explained above, is impractical for comparing whole genomes. However, it is possible to build a suffix tree in linear time by clever use of sets of pointers (17,18,20,21); our system uses McCreight's (18) algorithm. The total size of the tree is also linear in the sum of the lengths of the genomes in it, since there is exactly one leaf and at most one internal node for each base, and the sizes of these nodes are fixed. Note that the sequence label on each edge can be represented by two integers (its length and starting position in the genome), no matter how long it is. Our particular implementation uses 12 bytes per leaf node, 24 bytes per internal node, plus 1 byte for each base in the genome. More compact representations are possible (22). Because suffix-tree construction and all subsequent steps require no more than linear time and space, the overall running time (and space) required by the system is also linear. As a very generous upper bound, the system as implemented requires no more than 37 bytes per base of the input sequences; thus a comparison of two 100 Mb chromosomes would require <8 gigabytes of memory (and probably far less than that).

MUMmer begins by constructing a suffix tree *T* for genome *A*, and then adding the suffixes for genome *B* to *T*. Adding suffixes from an additional string to a suffix tree is a trivial modification of the construction algorithm for a single string, since the construction is accomplished by adding one suffix at a time to the portion of the tree that has already been constructed. Alternately, we can achieve the same effect by concatenating the two genomes (separated by a dummy character that does not occur in either genome) and constructing a suffix tree from that single concatenated string.

Each leaf node in *T* is labeled to indicate which suffix it represents in which genome, *A* or *B*. The system needs to identify the nodes in the tree that correspond to MUMs. It is not hard to see that every unique matching sequence is represented by an internal node with exactly two child nodes, such that the child nodes are leaf nodes from different genomes. The unique matches that are maximal can be identified by mismatches at their ends. (MUMmer as actually implemented determines whether a match is maximal based on pointers used to construct the suffix tree.) Thus, in a single scan of the suffix tree, all MUMs can be identified.

The main input parameter to the system, besides the genomes themselves, is the length of the shortest MUM that the system will identify. We typically do not want to report short MUMs that are likely to be random matches. For highly similar genomes (as with the two tuberculosis strains), we set this parameter to 50 bp. However, for more distantly related genomes, fewer MUMs of 50 bp might exist, and therefore this parameter can be adjusted. For aligning the two Mycoplasma species, we used a minimum MUM length of just 20 bp.

Sorting the MUMs

After finding all the MUMs, we sort them according to their position in Genome *A*. Now we consider the order of their matching positions in Genome *B*. In some cases, e.g. a transposition or reversal between the genomes, the *B* positions are not in ascending order. See Figure 3 for an illustration. Here we have assigned two integers to each MUM representing the ordinal position of the subsequence in Genomes *A* and *B*. Since we have sorted the MUMs by their *A*-positions, we can depict the alignment as the single sequence of *B*-position integers.

We now employ a variation of the LIS algorithm (9) to find the longest set of MUMs whose sequences occur in ascending order in both Genome *A* and Genome *B*. Essentially, we want the LIS contained in the sequence of *B*-position integers. For instance, if the order of *B* positions is given by the sequence ⟨1, 2, 10, 4, 5, 8, 6, 7, 9, 3⟩, the LIS is ⟨1, 2, 4, 5, 6, 7, 9⟩. The LIS technique allows us to browse the alignment from left to right, as well as 'close the gaps' in the alignment consistently. MUMmer implements a variation of this algorithm that takes into account the lengths of the sequences represented by the MUMs and the fact that they can overlap.

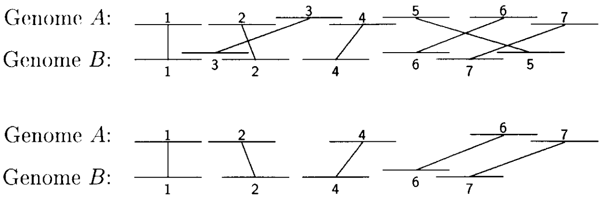

Figure 3. Aligning Genome *A* and Genome *B* after locating the MUMs. Each MUM is here indicated only by a number, regardless of its length. The top alignment shows all the MUMs. The shift of MUM 5 in Genome *B* indicates a transposition. The shift of MUM 3 could be simply a random match or part of an inexact repeat sequence. The bottom alignment shows just the LIS of MUMs in Genome *B*.

1. SNP: exactly one base (indicated by ^) differs between the two sequences. It is surrounded by exact-match sequence.

```
Genome A:  cgtcatgggcgttcgtcgttg
Genome B:  cgtcatgggcattcgtcgttg
                     ^
```

2. Insertion: a sequence that occurs in one genome but not the other.

```
Genome A:  cggggtaaccgc..................cctggtcggg
Genome B:  cggggtaaccgcgttgctcggggtaaccgccctggtcggg
                       ^^^^^^^^^^^^^^^^^^
```

3. Highly polymorphic region: many mutations in a short region.

```
Genome A:  ccgcctcgcctgg.gctggcgcccgctc
Genome B:  ccgcctcgccagttgaccgcgcccgctc
                     ^ ^^ ^^^
```

4. Repeat sequence: the repeat is shown in uppercase. Note that the first copy of the repeat in Genome *B* is imperfect, containing one mismatch to the other three identical copies.

```
Genome A:  cTGGGTGGGACAACGTaaaaaaaaaTGGGTGGGACAACGTc
Genome B:  aTGGGTGGGGCgACGTgggggggggTGGGTGGGACAACGTa
           ^        ^                        ^
```

Figure 4. The four types of gaps in MUM alignment. These examples are drawn from the alignment of the two *M.tuberculosis* genomes.

An example of an initial MUM-alignment and the ordered MUM-alignment that results from applying the LIS algorithm is shown in Figure 3. The longest increasing sequence algorithm requires O (K log K) time, where K is the number of MUMs. It can also be implemented using a simpler dynamic programming algorithm in $O(K^2)$ time. In general, K is much smaller than $N/\log N$ so this step takes $O(N)$ time.

Closing the gaps

Once a global MUM-alignment is found, we deploy several algorithms for closing the local gaps and completing the alignment. A gap is defined as an interruption in the MUM-alignment which falls into one of four classes: (i) an SNP interruption, (ii) an insertion, (iii) a highly polymorphic region or (iv) a repeat. These classes are depicted in Figure 4.

SNP processing. The identification of SNPs is becoming an increasingly important task in DNA sequence analysis, especially as the number of sequences from closely related organisms increases (23,24). SNPs in human DNA appear to be associated with many important health issues, including genetic illnesses and disease susceptibility. SNPs manifest themselves in two ways in the MUM alignment. In the simpler case, the SNP is surrounded by maximal unique matching subsequences. In this case our program easily identifies the SNP, which manifests itself as a simple gap of one base between the MUMs. In some cases, however, an SNP is adjacent to sequences that appear elsewhere in one of the genomes; i.e., the adjacent sequences are not unique. In this case the adjacent sequence plus the SNP is captured and processed by the repeat processing procedure described below.

Insert detection. An insert is defined as a moderately large region that appears in one genome and not the other. Such inserts are easily detected as large gaps in the alignment in one genome and not the other.

Inserts can be divided into two classes. Transpositions are subsequences that have been deleted from one location and inserted elsewhere (with respect to one of the genomes, of course). These are detected during a post-processing step; they appear in the MUM alignment out of sequence. Simple insertions are subsequences that appear in only one of the genomes; these may be the result of lateral transfer, simple deletions or other evolutionary processes. Regardless of the cause, these simple insertions can be identified as such because they do not appear in the MUM alignment.

Polymorphic regions. Gaps in the MUM alignment can be also caused by sequences that have large numbers of differences, but that still should be aligned in the whole genome alignment. Because the number of differences is high, it is less meaningful to define these regions in terms of the SNPs; for example, if the sequence identity is 25%, the sequences might be considered highly homologous although the number of SNPs is triple the number of conserved positions. If these regions are sufficiently small, we align them with a standard dynamic programming algorithm, essentially equivalent to Smith–Waterman (7). This produces an optimal alignment with respect to pre-specified insertion and mutation costs. For very large polymorphic regions, we can apply our entire matching procedure recursively using a reduced minimum MUM length, if desired.

Repeat processing. Repeat sequences do not appear in the MUM alignment because, by definition, the MUM alignment only includes sequences that appear exactly once in each genome. In our comparison of the *M.tuberculosis* strains we found that most repeats were tandem repeats. In every case we found in *M.tuberculosis*, repeat sequences were adjacent to unique sequence, and the MUM on either end of a tandem repeat extended into the repeat itself. As a result, the MUM alignment indicates a gap that is smaller than the length of the tandem repeat. Also note that the MUMs overlap one another.

Figure 5 shows a tandem repeat with different copy numbers in Genomes *A* and *B*. In this example, there are two MUMs: (i) uniqueAAGGAAGG and (ii) AAGGAAGGsequence. Depending on how this region is aligned, the four-base gap could appear anywhere between positions 6 and 14 (as shown) in the alignment. (For consistency, MUMmer always shows the insertion at the rightmost position.) The MUM alignment will indicate that MUM (i) occupies positions 0...13, and MUM (ii) occupies positions 10...25 in Genome *A*. The fact that these two intervals overlap indicates to the algorithm that a tandem repeat is present. The difference in overlap length in the two genomes (the overlap is eight bases in Genome *B* but only four bases in Genome *A*) indicates how many additional repeat bases are inserted in one of the genomes.

```
Genome A:  uniqueAAGGAAGGAAGGsequence
Genome B:  uniqueAAGGAAGG....sequence
           |         |         |
Position:  0         10        20
```

Figure 5. Repeat sequences surrounded by unique sequences. For the purposes of illustration, other characters besides the four DNA nucleotides are used.

RESULTS AND DISCUSSION

Comparing two strains of tuberculosis

We used MUMmer to perform a comparison of two strains of tuberculosis that have recently been sequenced, H37Rv (4) and CDC1551 (R.D.Fleischmann *et al*., manuscript in preparation). H37Rv is a laboratory strain that has been in continuous culture for >90 years, while CDC1551 is a recent clinical isolate that has demonstrated itself to be highly virulent (25). Tuberculosis is known to mutate relatively slowly (26), so despite the length of time that these strains have had to diverge, their genomes are still >99% identical (not counting several large repeat sequences that appear in different copy numbers). Understanding the differences is critical to understanding the different biological behavior of the two strains.

Running the two genomes through MUMmer produced a whole-genome alignment that mapped every base of one genome onto the other. Thus we were able to catalog all SNPs, all insertions of every length, all tandem repeats with different copy numbers and other miscellaneous differences. A detailed description of the biological consequences of this comparison will appear elsewhere (R.D.Fleischmann *et al*., manuscript in preparation).

Our alignment revealed thousands of individual differences between the two genomes, most of which were single base changes. There were several dozen large insertions unique to each genome, many of which contained genes or partial genes. An example is shown in Figure 6, which shows a 15 kb region containing three insertions and five point mutations. Using the display tool illustrated in the figure, we were able not only to identify all the differences, but also to identify which mutations were silent, which ones resulted in a premature stop in one genome, and which of the many differences occurred in intergenic regions. The tool allows the investigator to click on any of the genes, which are linked to a live database, and get a detailed report on that gene. The analysis showed that, of those insertions that occurred in genes, ~75% occurred in hypothetical genes; i.e., genes with no homology to any known gene. Because only 45% of the annotated genes are hypothetical, this result suggests that some of the annotated hypothetical genes are not real. MUMmer's output also makes it easy to find very long identical sequences: the longest such sequence shared between the two genomes is 24 563 bp, and there are 246 MUMs >5000 bp in length.

The time required to generate the alignments was 55 s on a DEC Alpha 4100, broken down as follows: 5 s for suffix tree construction, 45 s for sorting the MUMs and finding the longest increasing sequence and 5 s for generating the Smith–Waterman alignments of the gaps. (Because the sequences are so close to identical, the Smith–Waterman step needs to do very little work.)

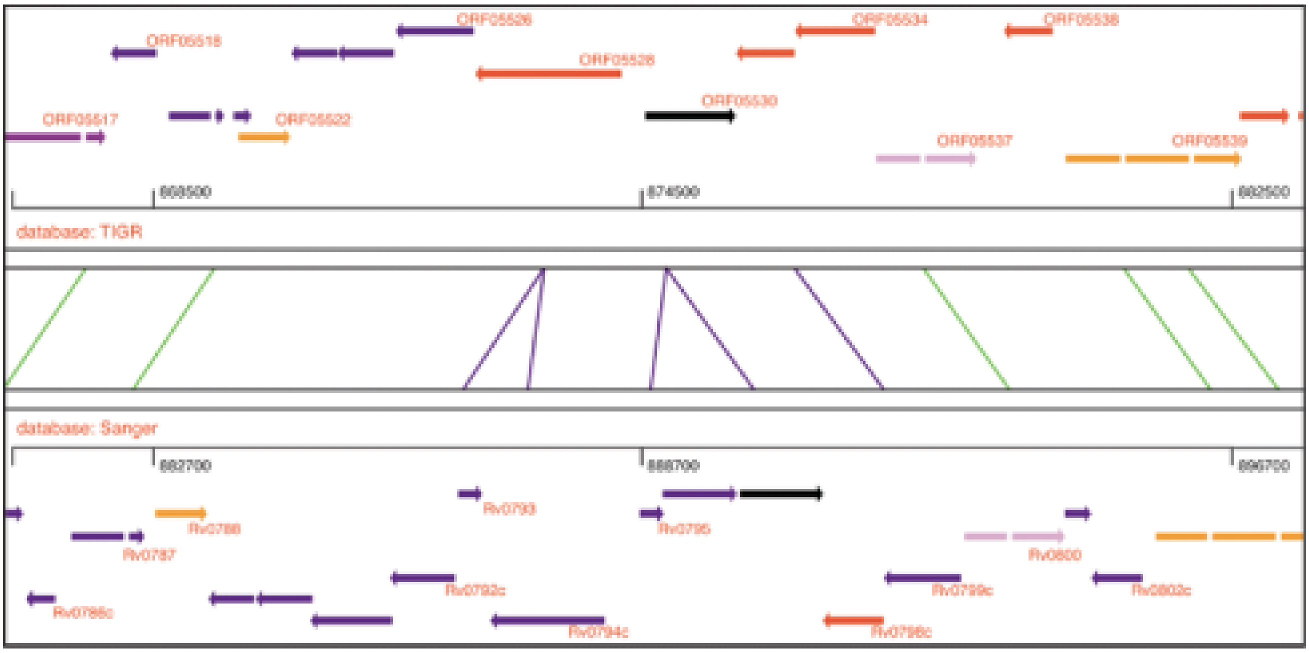

Figure 6. Alignment of *M.tuberculosis* strain CDC1551 (top) and H37Rv (bottom). This alignment was extracted from the graphical display tool developed to scan and zoom in on the output of the genome alignment program. In the view shown, single green lines in the center connect single-base differences between the genomes. Blue v-shaped lines indicate insertions. The first two v-shaped insertions are large insertions in the H37Rv strain, and the third insertion is a very small insertion in CDC1551. This last insertion appears as a line rather than a v-shape due to the resolution of the displayed region. The genes from both genomes are displayed as arrows, with color-coding to indicate the role assigned to each gene. Role assignments and gene IDs are taken from annotation of the TIGR and Sanger genome centers, respectively. Note that both of the large insertions shown here contain genes. White lines (gaps) appearing in the middle of some arrows indicate silent mutations in those genes. Point mutations that change an amino acid are displayed differently; none occur in this region. Either genome can be scrolled independently, and the scale can be adjusted up or down, from viewing of individual bases all the way out to viewing the entire genome on the screen.

Comparing two *Mycoplasma* genomes

The H37Rv and CDC1551 strains of *M.tuberculosis* are highly homologous, containing many subsequences thousands of nucleotides long that are perfect matches. To test the limits of our system, we turned to two bacteria that are 'cousins' but that are much more distantly related. The genome of *M.genitalium* is 580 074 nt in length, while *M.pneumoniae* is 816 394 nt. Clearly there are at least 226 000 nt of additional DNA in *M.pneumoniae*; however, alignments of proteins indicate that nearly all of *M.genitalium* is contained in *M.pneumoniae* (27). The protein alignments further indicate that very large fragments of *M.genitalium* retain the same order and orientation in *M.pneumoniae*. Thus, despite the evolutionary distance between these organisms, we believed that it might be possible to align them using MUMmer.

The system aligned these two genomes quite easily, as it turned out. This was somewhat surprising given previous difficulties at producing such an alignment using alternative methods. Not only did it work, but it worked very fast: the suffix tree portion of the computation took 6.5 s, while sorting and finding the LIS of MUMs took 0.02 s (on a DEC Alpha 4100). Generating the Smith–Waterman alignments of the gaps between the MUMs took 116 s; not surprisingly, this was much slower than for the tuberculosis genome comparison because of the larger amount of sequence that fell into the gaps.

Figure 7 illustrates the alignment at the whole genome level. Three different alignment methods were used in this figure. A FASTA alignment was generated by dividing each genome into 1000 bp segments and using those segments in 'all against all' FASTA (11) searches. [In the comparison of the two *Mycoplasma* that appeared previously (27), the FASTA and BLAST programs were used in a similar manner to compare all genes to all genes, omitting non-coding regions.] This analysis required many hours of compute time. Pairs of sequences that were at least 50% identical over 80% of the match appear as points in the plot. The middle of Figure 7 illustrates a much simpler method, where unique, exactly matching 25mers are plotted; this figure mimics the MUM alignment but uses a fixed length for the MUMs. This is less noisy than the FASTA alignment, but of course it only gives a rough alignment. The bottom of Figure 7 shows the MUM alignment, where each MUM appears as a point. This is the cleanest and most continuous of the alignments, though of course it contains many gaps. As the figure makes clear, the overall alignment of these two genomes is basically a long series of short (mostly 20–30 bp) exact matches strung together in the same order and with the same spacing in both genomes. The longest exactly matching sequence between the two genomes was only 281 bp and there are only 16 shared MUMs >100 bp. In total there are 20 872 bp contained in shared MUMs of ≥15 bp, just 3.6% of the shorter genome's length.

The MUM alignment clearly shows five translocations of *M.genitalium* sequence with respect to *M.pneumoniae*, in agreement with the analysis of Himmelreich *et al.* (27). In the FASTA and 25mer alignments, these translocations are either missing or very difficult to identify amidst the noise.

In addition to the data shown in Figure 7, MUMmer also produces a file containing the complete Smith–Waterman alignment of all the

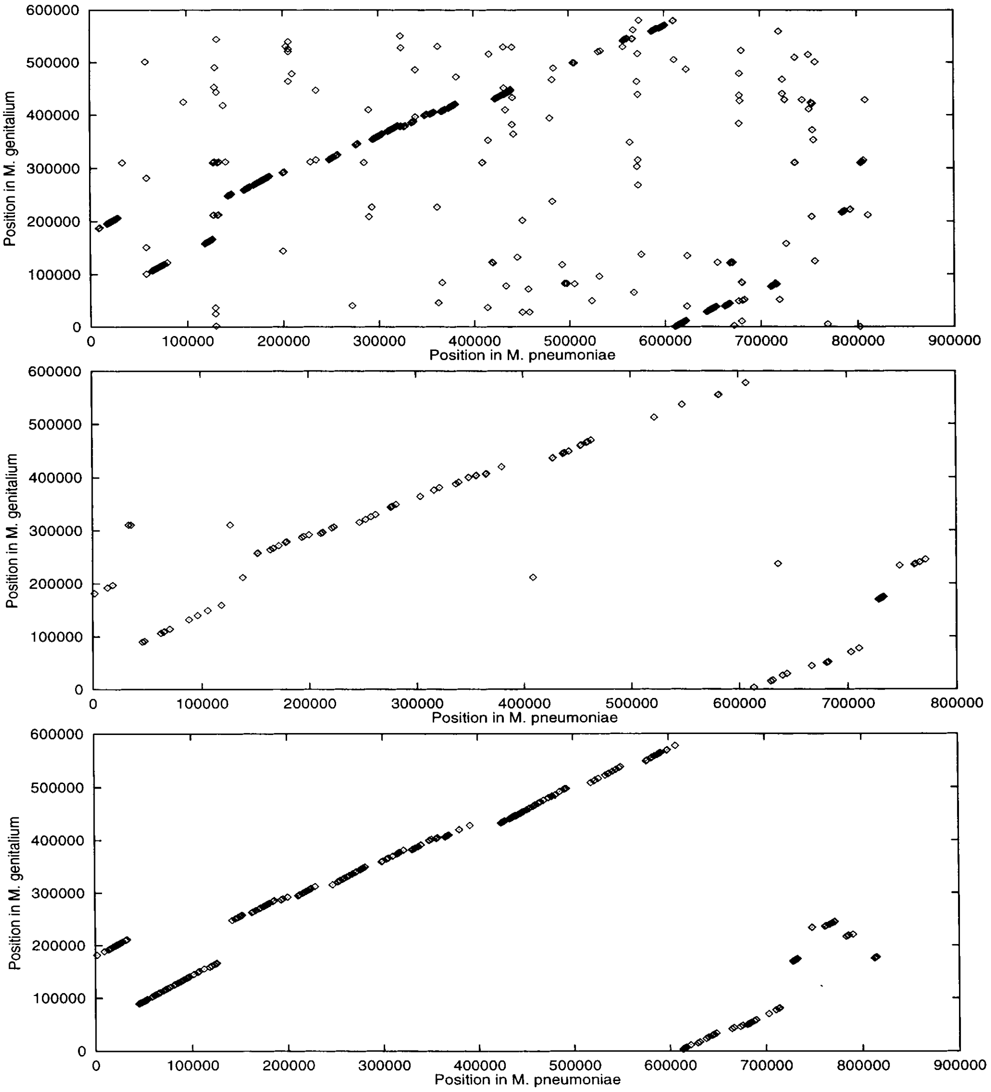

Figure 7. Alignment of *M.genitalium* and *M.pneumoniae* using FASTA (top), 25mers (middle) and MUMs (bottom). In all three plots, a point indicates a 'match' between the genomes. In the FASTA plot a point corresponds to similar genes. In the 25mer plot, each point indicates a 25-base sequence that occurs exactly once in each genome. In the MUM plot, points correspond to MUMs as defined in the main text.

gaps between the MUMs. (See Figure 4 for an illustration of what this output looks like.) Most of these aligned gaps were nearly identical in length, and obviously corresponded to sequences with a shared evolutionary history, but the sequence identity averaged <50%, much lower than in the *M.tuberculosis* comparison. Taken together, this data allows us to map every base of the smaller genome to its corresponding position in the larger genome.

Comparing human and mouse

To test MUMmer on sequences even more distant than the two *Mycoplasmas*, we chose a 222 930 bp subsequence of human chromosome 12p13 (accession no. U47924) that is syntenic to a 227 538 bp contiguous subsequence of mouse chromosome 6 (accession no. AC002397). These sequences were the subject of a recent study by Ansari-Lari *et al*. (28), who used a combination of alignment tools to produce a detailed alignment. These tools included DOTTER (29) for the initial comparison, a modified version of SIM (30) to find good local alignments, percent identity plots (pip) to display the results (31) and a program called CONSERVED (32) to identify segments >50 bp with >60% sequence identity in the SIM alignment. As reported in that study, the nucleotide similarity for the coding portions of the 17 genes in this region ranges from 70 to 92%, while the percent identity for amino acids ranges from 56 to 100%.

Although our alignment does not contain all the details generated and displayed by the combination of methods used in Ansari-Lari *et al*., the overall alignment of the two sequences is easily apparent from the output of our program. The program required 29 s of CPU time to generate the complete alignment, of which 1.6 s was used to build the suffix tree. The alignment using 15mers as the minimum MUM length contained several large gaps corresponding to intergenic regions, as shown in Figure 8. Re-running the program with minimum MUMs of 10 bp reduces the gaps, but in either case one must examine the Smith–Waterman algorithm output to see the complete alignment. The figure also shows other MUMs falling outside the aligned region; these

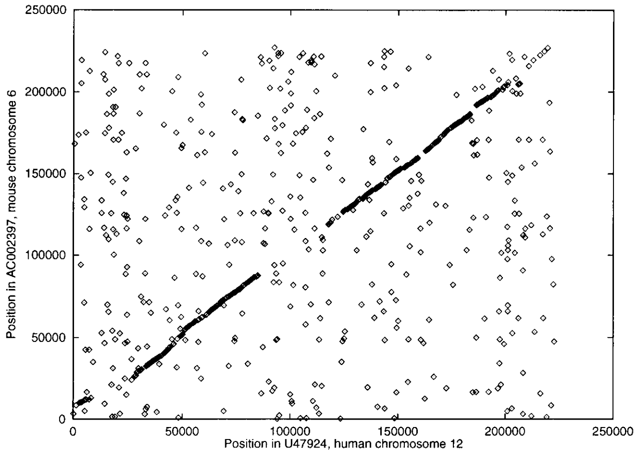

Figure 8. Alignment of a 222 930 bp subsequence of human chromosome 12p13, accession no. U47924, to a 227 538 bp subsequence of mouse chromosome 6, accession no. AC002397. Each point in the plot corresponds to an MUM of ≥15 bp.

are not included in the LIS, but are shown to illustrate the relatively low 'background' of random matches in the 15 bp MUM alignment.

The Smith–Waterman alignments of the regions between the MUMs show that the aligned region stretches almost the entire length of both sequences, containing 220 326 bp of the human sequence and 223 751 of mouse. The total number of bases contained in the MUM portion of the alignment is 14 026, or 6.3% of the human sequence. Only 10 MUMs in the alignment are >50 bp, with the longest being 117 bp.

SUMMARY

This paper describes MUMmer, a new system for high resolution comparison of complete genome sequences. The system was used to perform complete alignments of two pairs of genomes: the first pair were two closely related strains of *M.tuberculosis* of some 4.4 million nucleotides each, while the second pair were two different *Mycoplasma* bacteria that differed in length by almost half the length of the shorter genome. We also compared two sequences of ~225 kb each from the genomes of human and mouse. We expect many more closely related genomic sequences to appear in the coming years, and the system described here should prove useful in aligning those as well. We see no technical problems to using MUMmer for comparing even the longest genomic sequences, as long as sufficient computer memory is available.

ACKNOWLEDGEMENTS

Thanks to Edward Arnold for graphical assistance. S.L.S. is supported in part by the National Human Genome Research Institute at NIH under Grant no. K01-HG00022-1. S.L.S. and A.L.D. are supported in part by the National Science Foundation under Grant no. IRI-9530462 and S.K. is supported by NSF IRI-9529227. R.D.F. is supported in part by National Institute of Allergy and Infectious Diseases at NIH under Grant no. R01-AI40125-01.

REFERENCES

1 Fleischmann,R.D., Adams,M., White,O., Clayton,R., Kirkness,E., Kerlavage,A., Bult,C., Tomb,J.-F., Dougherty,B., Merrick,J., McKenney,K., Sutton,G., FitzHugh,W., Fields,C., Gocayne,J., Scott,J., Shirley,R., Liu,L.-I., Glodek,A., Kelley,J., Weidman,J., Phillips,C., Spriggs,T., Hedblom,E., Cotton,M., Utterback,T., Hanna,M., Nguyen,D., Saudek,D., Brandon,R., Fine,L., Fritchman,J., Fuhrmann,J., Geoghagen,N., Gnehm,C., McDonald,L., Small,K., Fraser,C., Smith,H. and Venter,J.C. (1995) *Science*, **269**, 496–512.
2 Fraser,C.M., Gocayne,J., White,O., Adams,M., Clayton,R., Fleischmann,R., Bult,C., Kerlavage,A., Sutton,G., Kelley,J., Fritchman,J., Weidman,J., Small,K., Sandusky,M., Fuhrmann,J., Nguyen,D., Utterback,T., Saudek,R., Phillips,C., Merrick,J., Tomb,J.-F., Dougherty,B., Bott,K., Hu,P.-C., Lucier,T., Peterson,S., Smith,H., Hutchison,C. and Venter,J.C. (1995) *Science*, **270**, 397–403.
3 Himmelreich,R., Hilbert,H., Plagens,H., Pirkl,E., Li,B.C. and Herrmann,R. (1996) *Nucleic Acids Res*., **24**, 4420–4449.
4 Cole,S.T., Brosch,R., Parkhill,J., Garnier,T., Churcher,C., Harris,D., Gordon,S.V., Eiglmeier,K., Gas,S., Barry,C.E., Tekaia,F., Badcock,K., Basham,D., Brown,D., Chillingworth,T., Connor,R., Davies,R., Devlin,K., Feltwell,T., Gentles,S., Hamlin,N., Holroyd,S., Hornsby,T., Jagels,K. and Barrell,B. (1998) *Nature*, **393**, 537–544.
5 Stephens,R., Kalman,S., Lammel,C., Fan,J., Marathe,R., Aravind,L., Mitchell,W., Olinger,L., Tatusov,R., Zhao,Q., Koonin,E. and Davis,R. (1998) *Science*, **282**, 754–759.
6 Needleman,S. and Wunsch,C. (1970) *J. Mol. Biol*., **48**, 443–453.
7 Smith,T. and Waterman,M. (1981) *J. Mol. Biol*., **147**, 195–197.
8 Pearson,W.R. (1995) *Prot. Sci*., **4**, 1145–1160.
9 Gusfield,D. (1997) *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology.* Cambridge University Press, New York.
10 Waterman,M. (1995) *Introduction to Computational Biology: Maps, Sequences and Genomes.* Chapman & Hall, New York.
11 Lipman,D.J. and Pearson,W.R. (1985) *Science*, **227**, 1435–1441.
12 Benson,G. (1999) *Nucleic Acids. Res*., **27**, 573–580.
13 Chao,K.-M., Zhang,J., Ostell,J. and Miller,W. (1997) *Comput. Appl. Biosci*., **13**, 75–80.
14 Chao,K.-M., Zhang,J., Ostell,J. and Miller,W. (1995) *Comput. Appl. Biosci*., **11**, 147–153.
15 Altschul,S., Gish,W., Miller,W., Myers,E. and Lipman,D. (1990) *J. Mol. Biol*., **215**, 403–410.
16 Altschul,S., Madden,T., Schaffer,A., Zhang,J., Zhang,Z., Miller,W. and Lipman,D. (1997) *Nucleic Acids Res*., **25**, 3389–3402.
17 Weiner,P. (1973) *Linear pattern matching algorithms.* In Proc. 14th IEEE Symp. Switching & Automata Theory. pp. 1–11.
18 McCreight,E.M. (1976) *J. ACM*, **23**, 262–272.
19 Brazma,A., Jonassen,I., Vilo,J. and Ukkonen,E. (1998) *Genome Res*., **8**, 1202–1215.
20 Chen,M.T. and Seiferas,J. (1985) In Apostolico,A. and Galil,Z. (eds), *Combinatorial Algorithms on Words.* Springer-Verlag, New York, pp. 97–107.
21 Ukkonen,E. (1995) *Algorithmica*, **14**, 249–260.
22 Kurtz,S. (1998) *Reducing the space requirement of suffix trees.* Technical Report 98–03, Universität at Bielefeld, Bielefeld, Germany.
23 Weiss,K.N. (1998) *Genome Res*., **8**, 691–697.
24 Taillon-Miller,P., Gu,Z., Li,Q., Hillier,L. and Kwok,P.-Y. (1998) *Genome Res*., **8**, 748–754.
25 Valway,S.E., Sanchez,M.P., Shinnick,T.F.,.Orme,I., Agerton,T., Hoy,D., Jones,J.S, Westmoreland,H. and Onorato,I.M. (1997) *N. Engl. J. Med*., **338**, 633–639.
26 Sreevatsan,S., Pan,X., Stockbauer,K.E., Connell,N.D., Kreiswirth,B.N., Whittam,T.S. and Musser,J.M. (1997) *Proc. Natl Acad. Sci. USA*, **94**, 9869–9874.
27 Himmelreich,R., Plagens,H., Hilbert,H., Reiner,B. and Herrmann,R. (1997) *Nucleic Acids Res*., **25**, 701–712.
28 Ansari-Lari,M., Oeltjen,J.C., Schwartz,S., Zhang,Z., Muzny,D.M., Lu,J., Gorrell,J.H., Chinault,A.C., Belmont,J.W., Miller,W. and Gibbs,R.A. (1998) *Genome Res*., **8**, 29–40.
29 Sonnhammer,E. and Durbin,R. (1995) *Gene*, **167**, GC1–GC10.
30 Huang,X., Hardison,R. and Miller,W. (1990) *Comput. Appl. Biosci*., **6**, 373–381.
31 Schwartz,S., Miller,W., Yang,C.-M. and Hardison R.C. (1991) *Nucleic Acids Res*., **19**, 4663–4667.
32 Oeltjen,J.C., Malley,T.M., Muzny,D.M., Miller,W., Gibbs,R.A. and Belmont,J.W. (1997) *Genome Res*., **7**, 315–329.