

10.1 Introduction

10.1.1 Task Definition

The availability of whole genome sequences opens great new possibilities for understanding the evolutionary process. For example, it provides an opportunity for studying the genetic relationship among closely related microorganisms.

Mouse and human are closely related species. They share a lot of gene pairs. It can be seen in Table “Mouse and Human Share a lot of Gene Pairs” [genBank].

Why whole genome alignment?

Suppose we are given human and mouse genomes, our aim is to extract all the conserved gene pairs between human and mouse. One possible solution is whole genome alignment:

- to find orthologous regions between two genomes,
- to do strain-to-strain or evolutionary comparison,
- to study genomic duplications,
- to analyze syntenic chromosomal regions.

10.1.2 Why not use standard pairwise methods?

1. Different features we are looking at.
 - For standard alignment: we only concern point mutation, insertion and deletion, etc.
 - For genome alignment: we also need to cater for transposition, large insertion/deletion, syntenic blocks, ... etc.
2. Technical concerns:
 - Time and space complexity, ... etc. Aligning whole genomes using standard methods use up a lot of time and space.

10.2 The MUMmer System

10.2.1 Definition of MUM

Though a pair of conserved genes rarely contain the same entire sequence, they share a lot of short common substrings and some of them are indeed unique to this pair of genes. Figure 10.1 is an example of this.

Given genomes A and B: we try to find all common substrings which are unique and are of maximal length. such common substring is known as Maximum Unique Match (MUM). For almost every conserved gene pairs, there exists a MUM which is unique to them.

Definition 10.1 *MUM stands for Maximal Unique Match substring. It is a common substring of the two genomes that is longer than a specific minimum length d such that the common substring is maximal, that is, the substring cannot be extended on either end without incurrng a mismatch and the common substring is unique in both sequences. (By default, $d = 20$.)*

Maximal Unique Match (MUM) have the following two properties:

- Occurs exactly once in both genomes A and B
- Not contained in any longer MUM

Rationale: a significantly long MUM is very likely to be part of the global alignment.

For example, for the sequences in Figure 10.1, There are four MUMs: ctc, gctac, ggtcagctatt, acttaccgc.

```

■ S=acgactcagctactggtcagctattacttaccgc#
■ T=acttctctgctacggtcagctattcacttaccgc$

```

Figure 10.1: An example

Consider another example, For $S = acgat\#$ and $T = cgta\$,$ there are two MUMs: cg and t.

10.2.2 How to find MUMs

Brute-force approach:

1. Input: Two genome sequences $S[1..m_1]$ and $T[1..m_2]$
2. For every position i in S
3. For every position j in T
 - Find the longest common prefix P of $S[i..m_1]$ and $T[j..m_2]$
 - check whether $|P| \geq d$ and whether P is unique in both genomes. If yes, report it as a MUM.

This solution requires at least $O(m_1 m_2)$ time. It's too slow!

Finding MUMs by suffix tree

The key idea in identifying MUMs is to build a suffix tree for genomes S and T .

MUMs can be found in $O(m_1 + m_2)$ time by suffix tree.

1. Build a generalized suffix tree for S and T
2. Mark all the internal nodes that have exactly two leaf children, which represent both suffixes of S and T .
3. For each marked node, WLOG, suppose it represents the i -th suffix S_i of S and the j -th suffix T_j of T . We check whether $S[i-1] = T[j-1]$. If not, the path label of this marked node is an MUM.

Example of finding MUMs

- Consider $S=acgat\#$, $T=cgta\$$
- Step 1: Build the generalized suffix tree for S and T . It can be seen in Figure 10.2.
- Step 2: As shown in Figure 10.3, nodes with path labels cg , g , t are marked since they have exactly two leaf children, which represent both suffixes of S and T .

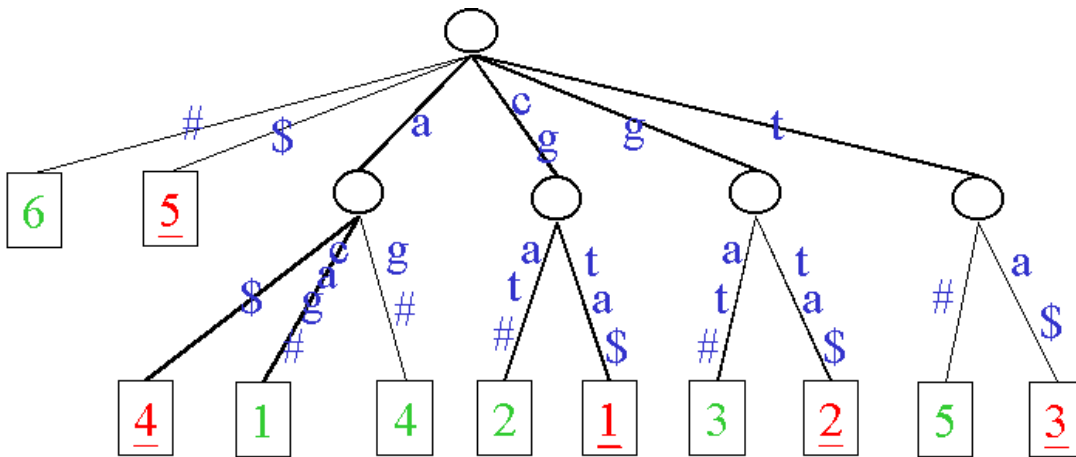


Figure 10.2: Step 1

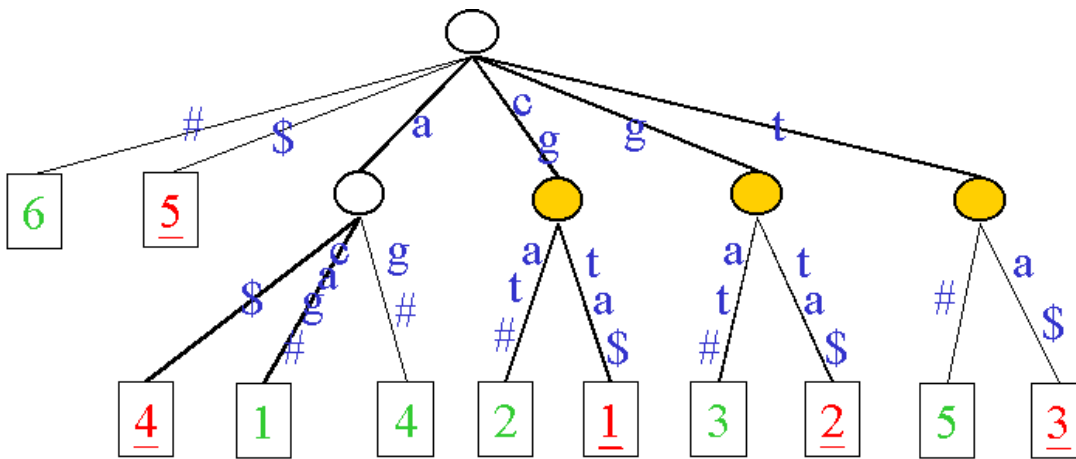


Figure 10.3: Step 2

- Step 3: For each marked node, WLOG, suppose it represents the i -th suffix S_i of S and the j -th suffix T_j of T . We check whether $S[i-1]=T[j-1]$. If not, the path label of this marked node is an MUM. It can be seen in Figure 10.4, the second node we marked with the path label g represents the 3-th suffix $gat\#$ of S and the 2-th suffix $gta\$$ of T . Both $S[3-1]$ and $T[2-1]$ are c , thus the path label g of this marked node is not an MUM. So our output here is cg and t .

Time analysis

- Step 1: Building suffix tree takes $O(m_1 + m_2)$ time.
- Step 2: Mark internal nodes takes $O(m_1 + m_2)$ time.

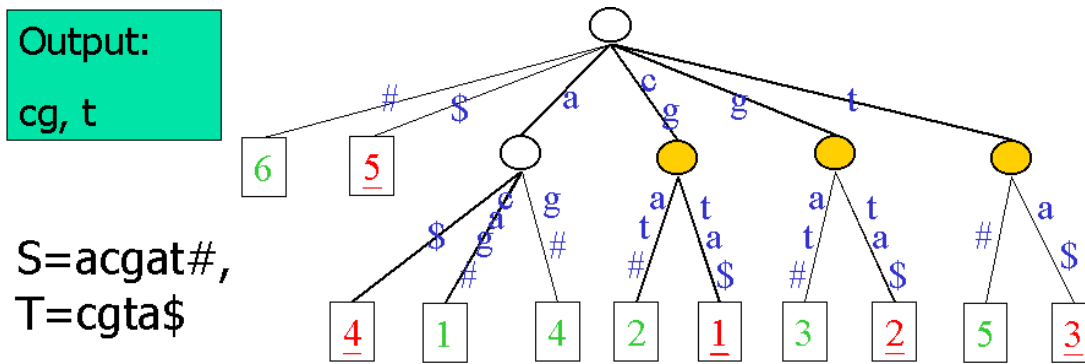


Figure 10.4: Step 3

- Step 3: Extracting MUM also takes $O(m_1 + m_2)$ time.
- In total, $O(m_1 + m_2)$ time.
- Space complexity is linear (exactly one leaf and at most one internal node for each base).
- Main parameter of system: length d of shortest MUM that should be identified (by default, $d = 20$).

When we do experiment, we found that MUMs can cover nearly 100% of the know conserved gene pairs Now we can find MUM in linear time. Is the problem solved? The answer is No. From Table “Mouse and Human Share a lot of Gene Pairs”, we can see that the number of MUMs \gg number of gene pairs. There are many noise. How can we extract the right MUMs?

Two related species should preserve the ordering of most conserved genes. Please see Figure 10.5 for an example.

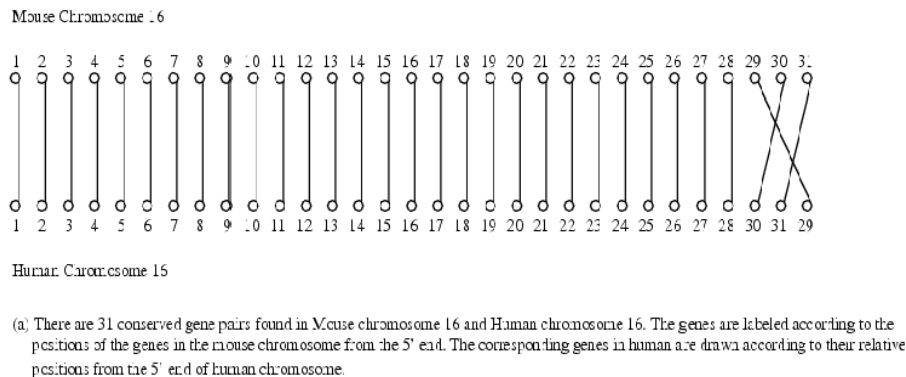


Figure 10.5: Conserved genes in Mouse Chromosome 16 and Human Chromosome 16

Instead of reporting all MUMs to the user, we compute the longest common subsequence (LCS) of all MUMs. Then we report only the MUMs on the LCS.

10.3 MUMmer1 :LCS

10.3.1 Introduction

The LCS (MUMmer1) algorithm [DKFPWS99] arises from a key observation that ordering of most of the conserved genes would be preserved in two related species. In 1999, Delcher et al [DKFPWS99], invented the MUMmer1 while working on *Mycoplasmata tuberculosis* strains. The Tuberculosis genome is known to be up to 99 percent identical.

Unlike MUM, which was proposed earlier, LCS seeks to reduce the number of false positives reported at the cost of reducing the coverage.

The MUMmer1 achieves its objectives by making several assumptions:

10.3.2 Assumptions

1. There exists a single long alignment.
2. MUMs are conserved in the longest common subsequence (LCS).
3. The sequences being compared are highly identical.

The first assumption may be wrong at times, as there may be other shorter but valid alignments which contains other MUMs. The other two assumptions helps to narrows the search space for the MUMs.

10.3.3 Idea

The Figure 10.6 shows an example of LCS. Given 2 Sequences 12345678 and 41325768, we get an LCS of 12568. The idea of LCS is that instead of reporting all MUMs to the user, we compute the LCS of all MUMs, and only report the MUMs on the LCS.

10.3.4 Applications

The LCS does a pairwise alignment and comparison of two large scale DNA sequences, and identifies all differences between them. It outputs a base-to-base alignment of the input sequences, highlighting the exact differences in the genome.

3. Close the gaps in the alignment by performing local identification of large inserts, repeats, small mutated regions, tandem repeats and SNPs.
4. 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 crucial principle behind step 1 is that if a long, perfectly matching sequence occurs exactly once in each genome, it is almost certain to be part of the global alignment.

10.3.7 Analysis of Complexity

STEP 1 - According to section 9.2.2, all MUMs can be computed in $O(n+m)$ time.

STEP 2 - After finding all the MUMs, we sort them according to their position in Genome A, and employ a variation of the LCS algorithm to find the longest set of MUMs whose sequences occur in ascending order contained in both Genome A and B. The LCS technique allows us to browse the alignment from left to right, as well as to "close the gaps" in the alignment consistently. This step requires $O(K \log K)$ time, where K is the number of MUMs. It is much faster than $O(K^2)$ -time simple dynamic programming algorithm since it takes advantage of the fact that all MUMs are distinct. In general, K is much smaller than N .

STEP 3 - Once a global alignment is found, we can 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 the four classes: (i) an SNP, (ii) an insertion, (iii) a highly polymorphic region or (iv) a repeat. Time and space is dependent on the algorithm deployed.

10.3.8 Problems

The strength of LCS lies with the speed at which it process long alignments, and that LCS produce much less false positives than its predecessors. The downside is its assumption (3) stated above, which means LCS overlooks many other shorter MUMs, and that expert knowledge has to be applied to the input sequences in order to yield better performance(in terms of coverage).

10.4 MUMmer2

10.4.1 Algorithmic Improvements

Three significant technical improvements in the core algorithms of MUMmer 2 are listed in the following.

10.4.1.1 Reducing Memory Usage

By employing techniques described by Kurtz [K99] the amount of memory used in the suffix tree was reduced to at most 20 bytes/bp (or amino acid, or other character). The maximum memory usage occurs in the case when each internal node in the suffix tree has only two children. In practice, however, many nodes have more than two children (particularly in the case of polypeptide sequences), which reduces the actual memory requirement.

10.4.1.2 New alternative algorithm for finding exact matches

The original algorithm [DKFPWS99] built a suffix tree containing two input sequences, and then found all maximal unique matches (MUMs) between them. A MUM is a subsequence that occurs in two exactly matching copies, once in each input sequence, and that cannot be extended in either direction. This algorithm is still available in the MUMmer 2 system, but the default algorithm is now a procedure that stores in the suffix tree only one sequence, which is called the reference sequence. The second sequence, which is called the query, is then 'streamed' against the suffix tree, exactly as if it were being added but without actually adding it. This technique was introduced by Chang and Lawler [CL94] and is fully described in [G97]. Using this process one can identify where the query sequence would branch off from the tree, thereby finding all matches to the reference sequence. Wherever a branch occurs at a tree position with just a single leaf beneath it, the match is unique in the reference sequence. By checking the character immediately preceding the start of this match one can determine whether it is a maximal match.

Thus, in time proportional to the length of the query sequence, one can identify all maximal matches between it and a unique string in the reference sequence. Note that these matches are not necessarily unique in the query sequence. Because we stream through the query, outputting matches as we find them, we do not know what sequence will occur later in the query. The advantage of this method is that only one of the two sequences is stored in the suffix tree, reducing the memory requirement by at least half. Further, because of the streaming nature of the algorithm, once the suffix tree has been built for the reference sequence, arbitrarily long, multiple queries can be streamed against it. In fact, the author [DPCS02] have used these programs to compare two assemblies of the entire human genome (each approximately 2.7 billion characters), using each chromosome as a reference and then streaming the other entire genome past it (A.Halpern, personal communication).

10.4.1.3 Cluster matches

A pair of conserved genes are likely corresponding to a sequence of MUMmers that are consecutive and close in both genomes and have sufficient length (Figure 10.7), where the set of such sequences of MUMmers is called cluster.

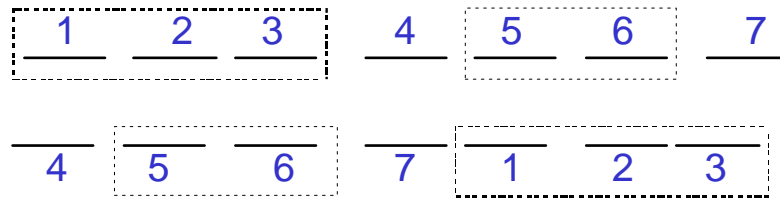


Figure 10.7: sequence of MUMmers

MUMmer1 presumed that two complete sequences were to be aligned, and that no major rearrangements would have occurred between them. Hence, it computed a single longest alignment between the sequences. To facilitate comparisons involving unfinished assemblies and genomes with significant rearrangements, a module has been added that first clusters matches together and then finds consistent paths within each cluster. As a result, the system outputs a series of separate, independent alignment regions. The clustering is performed by finding pairs of matches that are sufficiently close and on sufficiently similar diagonals in an alignment matrix (using thresholds set by the user), and then computing the connected components for those pairs. Within each component, a longest ascending subsequence computation is done to yield the most consistent sequence of matches in the cluster.

10.4.2 Applications of MUMmer 2

MUMmer 2 has been applied to "alignment of incomplete genomes" and "comparative genome annotation", where two new solvers are developed to solve these two problems. They are NUCmer (nucleotide MUMmer) and PROmer (protein MUMmer), where MUMmer is the core algorithm for them. Experiments showed that MUMmer is useful in solving the two problems stated above. Detailed description for these two applications can be found in [DPCS02]

10.5 MUMmer3

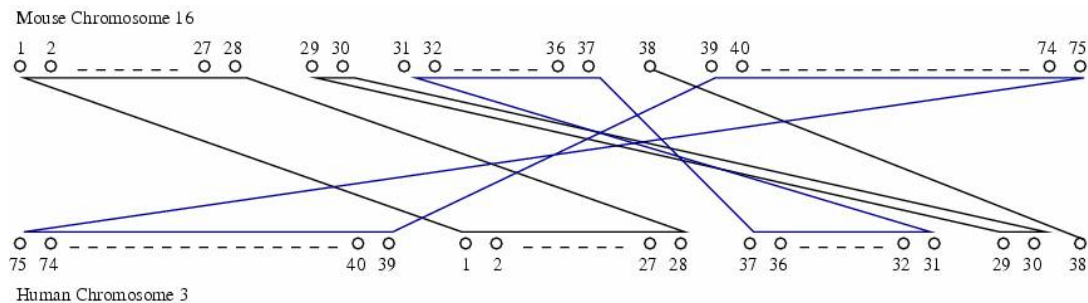
MUMmer 3 uses approximately 17 bytes for each basepair in the reference sequence. Furthermore, the query sequence is "streamed" past the reference suffix tree, so that the memory requirements do not at all depend on the size of the query sequences. Critical to the improvements in MUMmer 3 is a complete

re-write of the core suffix tree library, implemented by Stefan Kurtz [K99] and explained in his various publications. The improvements resulting from the use of this library can be seen in the table below. All statistics are from test runs on a 3.0 GHz Pentium 4 computer running Linux. Resulting output includes both forward and reverse matches.

10.6 Mutation Sensitive Alignment

10.6.1 Introduction

The Mutation Sensitive Alignment (MSA) algorithm [CLSWY03] is based on the observation that if two genomes are closely related, then it is likely that they can be transformed from each other using a few transposition/reversal operations. For example, it can be seen in Figure 10.8 that Mouse Chromosome 16 can be transformed into Human Chromosome 16 in two operations (on sequences 31-37 and 39-75).



(b) There are 31 conserved gene pairs found in Mouse chromosome 16 and Human chromosome 3. The genes are labeled as in (a). Note that the relative ordering of Genes 31 to 37 in human chromosome is exactly the reverse of that in the mouse chromosome. The same situation occurs in Genes 39 to 75.

Figure 10.8: Comparison of Mouse Chromosome 16 and Human Chromosome 16

The MSA algorithm makes use of this observation by looking for subsequences in the two input genome strings that differ by at most k transposition/reversal operations (the authors of the algorithm set $k=4$). It then returns these subsequences as possible conserved genes.

10.6.2 Concepts and Definitions

In this section, we explain the similar subsequence problem (or *k-similar subsequence problem*), which is the basis of the MSA algorithm.

Definition 10.2 Let $A = (a_1, a_2, \dots, a_n)$ and $B = (b_1, b_2, \dots, b_n)$ be the order of the n MUMs in two genome sequences S and T , respectively. A sequence $C = (c_1, c_2, \dots, c_m)$ is a **subsequence** of A if there exists indices (i_1, i_2, \dots, i_m) such that $i_1 < i_2 < \dots < i_m$ and $c_j = a_{i_j}$ for all j . C is a **common subsequence** of S and T if C is a subsequence of both A and B .

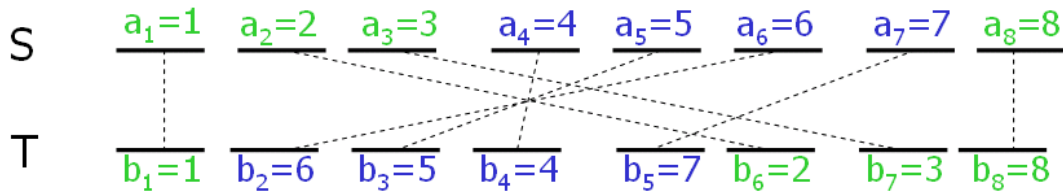


Figure 10.9: Example of Common Subsequence

For example, $(1,2,3,8)$ is a common subsequence of S and T in Figure 10.9. The *weight* of a common subsequence is the total weight of all the MUMs that make up the subsequence. The *maximum weight common subsequence (MWCS)* is the common subsequence with maximum weight.

Definition 10.3 Let k be a non-negative integer. Consider a subsequence X of A and a subsequence Y of B . (X, Y) is a pair of **k -similar subsequences** if X can be transformed into Y by performing k transposition/reversal operations on k disjoint subsequences in X .

Intuitively, a pair of k -similar subsequences each consists of k matching blocks (possibly after a transposition and/or reversal operation) and a remaining common subsequence. This common subsequence is called the *backbone*. Figure 10.10 shows a 2-similar subsequence, with a backbone of $(1,7,8)$.

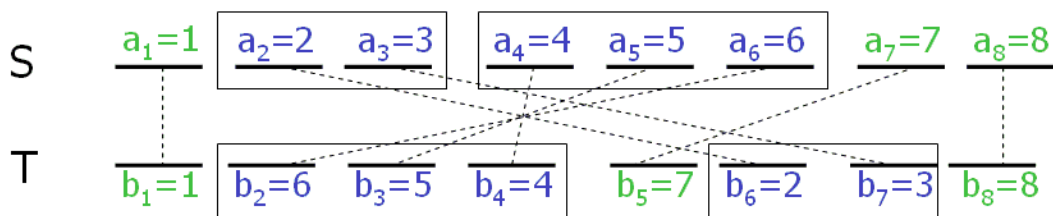


Figure 10.10: A 2-Similar Subsequence

The *Similar Subsequence Problem* has as input two sequences A and B , and a non-negative parameter k . The task is to find a k -similar subsequence of A and B with the greatest weight.

10.6.3 Algorithm Details

The MSA algorithm basically returns the set of greatest-weight k -similar subsequences as output. The premise is that conserved genes between two species are likely to be k -similar for some (small) user-defined value of k .

MSA-Algorithm(Sequence A, B; Integer k)

1. Find all MUMs of A and B (in $O(m_1 + m_2)$ time, as shown in section 9.2.2.)
2. Find the k -similar subsequences of A and B
3. Return the set of k -similar subsequences

The Similar Subsequences Problem that forms the basis of the MSA algorithm in Step 2 has been shown to be NP-Complete [CLSWY03], by reduction from the **MAX-2SAT Problem** [GJ79]. Therefore, an optimal solution to the Similar Subsequences Problem will take exponential time (assuming $P \neq NP$).

One possible brute-force approach to this problem takes $O(n^{2k+1} \log n)$ time. Unfortunately, this is too slow for practical purposes. However, using a heuristic algorithm, this problem can be approximated in $O(n^2(\log n + k))$ time [CLSWY03].

10.6.4 Experimental Results

The performance of MSA was compared to that of MUMmer3 on 15 pairs of mouse and human chromosomes (Table “Mouse and Human Share a lot of Gene Pairs”). Included in this set of data is the known conserved genes between the pairs of chromosomes. This data was obtained from *GenBank* [GENBANK], the largest public DNA sequence database maintained by the National Center for Biotechnology Information (NCBI).

For these experiments, the parameter k for MSA was set at 4, while the parameter gap for MUMmer3 was set at 2000. For each of the chromosome pairs, both techniques were rated on their *coverage* (i.e. the percentage of published genes discovered) and *preciseness* (i.e. the percentage of results returned that matches the published genes). The results are given in Figure 10.11.

It can be seen from these results that MSA provides better coverage as well as better preciseness in general over MUMmer3, of 91.3% and 29.3% respectively. MSA is the current best technique for discovering conserved genes between two closely related genomes.

| Exp. No. | Coverage | | Preciseness | |
|----------|----------|---------|-------------|--------|
| | MUMmer | MSA | MUMmer | MSA |
| 1 | 76.50% | 92.20% | 21.70% | 22.70% |
| 2 | 71.40% | 91.70% | 21.30% | 25.10% |
| 3 | 87.00% | 100.00% | 24.80% | 25.50% |
| 4 | 76.30% | 94.70% | 27.40% | 26.70% |
| 5 | 92.50% | 96.30% | 32.50% | 32.00% |
| 6 | 72.20% | 95.80% | 31.20% | 32.90% |
| 7 | 67.70% | 87.10% | 13.50% | 17.80% |
| 8 | 78.10% | 90.60% | 37.20% | 36.70% |
| 9 | 80.00% | 86.70% | 40.70% | 49.70% |
| 10 | 82.00% | 92.00% | 30.90% | 32.10% |
| 11 | 65.20% | 89.10% | 30.50% | 36.00% |
| 12 | 60.00% | 80.00% | 27.50% | 41.90% |
| 13 | 89.10% | 95.30% | 18.20% | 18.40% |
| 14 | 72.70% | 86.40% | 10.40% | 12.60% |
| 15 | 78.50% | 91.40% | 30.00% | 29.70% |
| average | 76.60% | 91.30% | 26.50% | 29.30% |

Figure 10.11: Experimental Results

References

- [GJ79] M.R. GAREY and D.S. JOHNSON, “Computers and Intractability: A Guide to the Theory of NP-Completeness”, W.H. Freeman and Co., 1979.
- [CLSWY03] H.L. CHAN, T.W. LAM, K. SUNG, P. WONG and S.M. YIU, “Improved Whole Genome Alignment via Mutation-Sensitive Sequence Similarity”, in preparation.
- [GENBANK] “<http://www.ncbi.nlm.nih.gov/Homology>”
- [DKFPWS99] A.L. DELCHER, S. KASIF, R.D. FLEISCHMANN, J. PETERSON, O. WHITE and S.L. SALZBERG, “Alignment of whole genomes”, *Nucleic Acids Research*, 1999, Vol. 27, No.11, 2369–2376.

- [Ara00] The Arabidopsis Genome Initiative (2000) Analysis of the genome sequence of the flowering plant *Arabidopsis thaliana*. *Nature*, 408,796–815.
- [CL94] W.I. CHANG AND E.L. LAWLER (1994) Sublinear expected time approximate string matching and biological applications. *Algorithmica*, 12, 327–344. 11. Perna,N.T., Plunkett,G.,III, Burland,V., Mau,B., Glasner,J.D., Rose,D.J., Mayhew,G.F., Evans,P.S., Gregor,J., Kirkpatrick,H.A. et al. (2001) Genome sequence of enterohaemorrhagic *Escherichia coli* O157:H7. *Nature*, 409, 529–533.
- [DPCS02] A.L. DELCHER, A. PHILLIPPY , J. CARLTON and S.L. SALZBERG , Fast algorithms for large-scale genome alignment and comparison, *Nucleic Acids Research*, *Nuclei Acids Research*, 30(11):2478-2483, 2002.
- [EHWS00] J.A. EISEN, J.F. HEIDELBERG,O. WHITE and S.L. SALZBERG, (2000) Evidence for symmetric chromosomal inversions around the replication origin in bacteria. *Genome Biol.*, 1, 1101–1109.
- [K99] S. KURTZ (1999) Reducing the space requirement of suffix trees. *Software Pract. Experience*, 29, 1149–1171.
- [LKRS99] X. LIN, S. KAUL,S. ROUNSLEY,T.P. SHEA,M.I. BENITO,C.D. TOWN, C.Y. FUJII,T. MASON,C.L. BOWMAN AND M. BARNSTEAD ET AL . (1999) Sequence and analysis of chromosome 2 of the plant *Arabidopsis thaliana*. *Nature*, 402, 761-768.
- [G97] D. GUSFIELD (1997) *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, New York.

Table 10.1: Mouse and Human Share a lot of Gene Pairs

| Mouse Chr No. | Human Chr No. | # of Published Gene Pairs | # of MUMs |
|---------------|---------------|---------------------------|-----------|
| 2 | 15 | 51 | 96,473 |
| 7 | 19 | 192 | 52,394 |
| 14 | 3 | 23 | 58,708 |
| 14 | 8 | 38 | 38,818 |
| 15 | 12 | 80 | 88,305 |
| 15 | 22 | 72 | 71,613 |
| 16 | 16 | 31 | 66,536 |
| 16 | 21 | 64 | 51,009 |
| 16 | 22 | 30 | 61,200 |
| 17 | 6 | 150 | 94,095 |
| 17 | 16 | 46 | 29,001 |
| 17 | 19 | 30 | 56,536 |
| 18 | 5 | 64 | 131,850 |
| 19 | 9 | 22 | 62,296 |
| 19 | 11 | 93 | 29,814 |

Table 10.2: Agenda

| | Coverage | Preciseness |
|------------------------------|-----------------|--------------------------|
| MUM | $\approx 100\%$ | Many false positives |
| LCS (MUMmer1) | Very less | Not many false positives |
| Clustering (MUMmer3) | 76.6% | 26.5% |
| Mutation-Sensitive Alignment | 91.3% | 29.3% |

Table 10.3: Compare MUMmer 3 with MUMmer 2

| | MUMmer 2 | MUMmer 3 |
|--|-----------------|----------------|
| E.coli K12 vs. E.coli O157:H7 | 102 MB / 18 s | 77 MB / 17 s |
| S.cerevisiae vs. S.pombe | 261 MB / 51 s | 204 MB / 47 s |
| A.fumigatus vs. A.nidulans | 578 MB / 128 s | 459 MB / 120 s |
| | NUCmer 2 | NUCmer 3 |
| D.melanogaster arm 2L vs. D.pseudoobscura | 684 MB / 879 s | 485 MB / 835 s |
| | PROmer 2 | PROmer 3 |
| P.falciparum vs. P.yoelii | 752 MB / 1109 s | 522 MB / 975 s |