ALLOWING MISMATCHES IN ANCHORS FOR WHOLE GENOME ALIGNMENT: GENERATION AND EFFECTIVENESS

SM YIU* PY CHAN* TW LAM* WK SUNG† HF TING* PWH WONG‡

*Department of Computer Science
The University of Hong Kong, Hong Kong
E-mail: {smyiu, pychan, twlam, hfting}@cs.hku.hk

†*Department of Computer Science*
National University of Singapore, Singapore
E-mail: ksung@comp.nus.edu.sg

‡*Department of Computer Science*
The University of Liverpool, UK
E-mail: pwong@csc.liv.ac.uk

Recent work on whole genome alignment has resulted in efficient tools to locate (possibly) conserved regions of two genomic sequences. Most of such tools start with locating a set of short and highly similar substrings (called *anchors*) that are present in both genomes. These anchors provide clues for the conserved regions, and the effectiveness of the tools is highly related to the quality of the anchors. Some popular software tools use the exact match maximal unique substrings (EM-MUM) as anchors. However, the result is not satisfactory especially for genomes with high mutation rates (e.g. virus). In our experiments, we found that more than 40% of the conserved genes are not recovered. In this paper, we consider anchors with mismatches. Our contributions include the following.

- Based on the experiments on 35 pairs of virus genomes using three software tools (MUMmer-3, MaxMinCluster, MSS), we show that using anchors with mismatches does increase the effectiveness of locating conserved regions (about 10% more conserved gene regions are located, while maintaining a high sensitivity).
- To generate a more comprehensive set of anchors with mismatches is not trivial for long sequences due to the time and memory limitation. We propose two practical algorithms for generating this anchor set. One aims at speeding up the process, the other aims at saving memory. Experimental results show that both algorithms are faster (6 times and 5 times, respectively) than a straightforward suffix tree based approach.

1. Introduction

Recent research on whole genome alignment allows one to locate conserved regions between two given genomic sequences in an efficient manner. Existing software tools were designed based on the assumption that two regions, if conserved, share a lot of short substrings that are highly similar and unique, though they rarely contain the same sequence. Thus, the first step of these tools is usually to locate a set of such short substrings (called *anchors*). These anchors provide a rough guideline on which portions of the genomes conserved regions can be found. Note that a lot of these anchors may come from noise. The next step is to eliminate the noise and identify the conserved regions. Various techniques

Table 1. Performance of 3 types of anchors on 35 virus pairs.

	EM-MUM		**GAME**		**5-Mismatch Anchors**	
	Coverage	Sensitivity	Coverage	Sensitivity	Coverage	Sensitivity
MUMmer-3	53.0%	66.9%	53.8%	57.4%	62.2%	74.4%
MaxMinCluster	55.4%	66.7%	55.8%	58.5%	63.6%	65.6%
MSS	56.0%	65.9%	60.9%	61.3%	70.6%	82.2%

and heuristics have been proposed for this step (e.g., maximum common subsequence and clustering).

It is obvious that the effectiveness of the software tools are highly dependent on the set of anchors that are identified in the first step. Some popular software tools use maximal substrings that are exactly matched and unique in the two genomes (EM-MUM) as anchors [5, 9]. However, it is found that the amount of conserved regions recovered are not satisfactory, in particular, for genomes with high mutation rates (e.g. virus genomes), thus affecting the final effectiveness of the tools. In Table 1, the first column shows the average result of aligning 35 pairs of virus genomes using EM-MUMs as anchors; we use three different tools namely, MUMmer-3[9], MaxMinCluster[10], and MSS[2] to select the anchors. The performance of the three tools are similar, the coverage ranges from 53% to 56% (i.e., identifying 53% to 56% of conserved gene regions that are known). In fact, we have further investigated the anchors (EM-MUMs) themselves and found that they covered only 66% of the published conserved genes; in other words, any software using EM-MUMs as anchors can achieve a coverage of at most 66%. To improve the coverage, we need better methods to generate better anchors. Another difficulty is that we need to maintain a reasonable sensitivity (refers to the percentage of reported regions that overlap with published conserved gene regions). In this paper, we focus on finding a better set of anchors.

A natural extension for EM-MUM is to allow some mismatches in the maximal unique substrings. In fact, the idea of allowing mismatches in anchors has been explored in a number of research projects[1, 3, 7, 8, 11, 12, 16] and their results also support this extension. Some of these approaches allow mismatches in the anchors based on the statistical background probability of the matching regions[11]. Some tried to incorporate certain biological knowledge when characterizing the type of the mismatches (e.g., DBA[7] and WABA[8]). However, sometimes it is difficult to obtain the appropriate statistical and biological knowledge for the genomes to be aligned. Also, these knowledge may not be general for all cases. Other works take a more general approach. For example, GAME[3], the most recent work using anchors with mismatches, first starts with maximal exact matched substrings, then it tries to extend each of these substrings on the left and the right by allowing mismatches character by character. The extension stops if the percentage of the identical bases drops below a certain threshold. The extended substring is used as an anchor if its length is longer than a pre-set minimum length. From a computational point of view, anchors with a small number of mismatches may be missed in such a generation due to the heuristics nature of the process. We also found that the effectiveness of these anchors fluctuates and may not be significantly better than that of EM-MUM. In Table 1, the second column shows the performance of the three software when using anchors provided by GAME. We can see that out of the three software tools, two give almost no improvement in the coverage when compared to the case in EM-MUM, only one shows a 4.9% increase in coverage. Note that the sensitivity drops in all cases.

On the other hand, we believe that the assumption of having short, unique, and highly similar common substrings in conserved regions is reasonable. In this paper, we propose to generate these unique anchors with x mismatches (called *x-mismatch anchors*, formal definition will be given in Section 2) in a more systematic way. There are two issues involved. The first issue is whether it is necessary to generate a more comprehensive set of x-mismatch anchors in order to achieve higher coverage. In this work, we provide evidence showing that the answer is affirmative. Then, a follow-up question is how one can generate these x-mismatch anchors. This second issue is more difficult than one may expect. While the generation of EM-MUMs can be done in linear time using suffix tree[5], allowing mismatches in the substrings together with the requirement of uniqueness slow down the generation process substantially. The slow down is significant when we want to work on long sequences. For example, the generation time for EM-MUMs for a pair of human-mouse chromosomes with sizes 28M and 14M respectively, is only 5 minutes, however, the generation time for 2-mismatch anchors using a straight-forward approach based on suffix tree requires about 12 hours. We then provide two practical algorithms for generating the x-mismatch anchors. Our contributions are summarized in the following.

- We have compared the effectiveness of three types of anchors: (a) EM-MUM; (b) anchors from GAME; (c) the x-mismatch anchors. We have tested 35 pairs of virus genomes; our evaluation is based on the result of three software tools (MUMmer-3, MaxMinCluster, and MSS). We found that using the x-mismatch anchors, all tools can achieve about 10% increase in coverage (refer to Table 1). More importantly, the improvement in coverage does not imply a decrease in sensitivity. We have also measured the anchors themselves and found that the x-mismatch anchors can achieve 8 - 14% higher coverage than the EM-MUMs, and 8 - 10% higher coverage than the anchors from GAME.

  Besides genomes with high mutation rates, we also tested our anchors on a number of human-mouse chromosome pairs, which are supposed to be more closely related and with lower mutation rates in both DNA and translated protein sequences. The results also show an increase in coverage although the increase for the translated protein sequences is not as significant as in the other case.
- To tackle the problem of anchor generation, we propose two practical algorithms. The first one (called Suffix-Exd) makes use of the suffix tree for locating short substrings (seeds), then performs extension on the seeds to enumerate the anchors. However, in real applications, building a suffix tree for a long sequence requires a large amount of memory, so our second approach (called Hash-Tab) makes use of a hash table to substitute the suffix tree.

  Table 2 compares the running time and memory usage of our approaches with a brute-force approach based on suffix tree using a long human-mouse chromo-

Table 2. Performance of our suggested algorithms for generating 2-mismatch anchors (based on human chromosome 16 of size 28M and mouse chromosome 17 of size 14M).

	Running Time	Memory Usage
Brute-force	12 hr	600M
Suffix-Exd	1.5hr	600M
Hash-Tab	2.6hr	120M

some pair. The results show that our first algorithm runs 6 times faster than the brute-force approach and the second algorithm requires 5 times less memory than the suffix-tree based approach while the running time is still significantly faster than the brute-force approach. We also propose a faster algorithm that makes use of the suffix links to speed up Suffix-Exd for $x \leq 3$.

Remark: The anchor generation problem we studied is related to the approximate string matching problem[4, 13−15]. However, the two problems are not exactly the same. In the approximate string matching problem, we are given a pattern and we want to locate the occurrences of all substrings in a given text that are similar to the given pattern. However, in the anchor generation problem, we are given two long texts and we want to locate all pairs of *maximal* substrings, one in each text such that the two substrings are similar and appear uniquely in the respective text. Also, the algorithms for approximate string matching problem are usually difficult to implement and their practicality for long DNA sequences is still an unknown.

Organization of the paper: The rest of the paper is organized as follows. Section 2 defines the x-mismatch anchors and discusses the effectiveness of these anchors. The x-mismatch anchor generation problem and our proposed generation algorithms are presented in Section 3. Section 4 concludes the paper.

2. The x-Mismatch Anchor and its Effectiveness

In this section, we first formally define an x-mismatch anchor. Then, we compare the effectiveness of these x-mismatch anchors with EM-MUM and anchors from GAME, the most recent work that uses anchors with mismatches.

2.1. *The x-Mismatch Anchor*

Given two genomes, A and B, we define an x-mismatch anchor as follows. We assume that the input genomes are from the positive strand. We use the notations A+and A-to represent the positive and negative strands of A, respectively. Let a and b be two substrings in A and B, respectively. We denote the hamming distance between a and b as $\texttt{Hamm\_dist}(a, b)$.

Definition 2.1. A pair of substrings a and b (a in A and b in B) is an x-mismatch anchor if it satisfies the following.

(1) $\texttt{Hamm\_dist}(a, b) \leq x$. (i.e. At most x mismatches are allowed.)
(2) *Uniqueness*: The substrings a and b appear exactly once in A and B, respectively. (i.e. a appears exactly once in A+or A-, but not both. The same applies to b in B.)

(3) *One-to-one*: The substrings a and b are exact match. Otherwise, $1 \leq \texttt{Hamm\_dist}(a,b) \leq x$ such that there does not exist another substring a' of A with $\texttt{Hamm\_dist}(a',b) \leq x$ and there does not exist another substring b' of B with $\texttt{Hamm\_dist}(a,b') \leq x$.

(4) The first (and the last) characters of a and b must match. (This is to avoid extending two exact matched substrings by $\leq x$ mismatched characters to form another (redundant) x-mismatch anchor.)

(5) *Maximal*: We require the pair (a, b) to be maximal. (i.e. For any (a', b'), if a and b are substrings of a', b' respectively, then (a', b') cannot form an x-mismatch anchor.)

The x-mismatch anchor generation problem is to find all possible pairs (a, b) that are x-mismatch anchors of A and B. In practice, we usually require the anchors to be of length at least L, a user-defined parameter.

2.2. *Effectiveness of x-Mismatch Anchors*

We compare the effectiveness of x-mismatch anchors with that of EM-MUM and the anchors from GAME. We use these anchors as input to three software tools, MUMmer-3, MaxMinCluster, and MSS. The evaluation is based on the set of conserved regions reported by these tools with respect to the set of published conserved genes of the two input genomes. We measure the effectiveness from two aspects: the coverage and the sensitivity. The *coverage* is the percentage of published conserved genes that overlap with the reported regions. Note that high coverage alone may not imply high quality output as an algorithm can simply output every input anchor to achieve the maximum coverage. So, we also measure the percentage of reported regions that overlap with a conserved gene and the percentage is referred as the *sensitivity*. A high quality output is expected to have high coverage and reasonable sensitivity. Note that for the software tools and the generation of anchors from GAME, we set the parameters to be the default values or the values recommended by the authors[a].

Aligning Genomes with High Mutation Rates: We first evaluate the anchors using genomes with high mutation rates. We use nine virus genomes of length from 100K to 180K nucleotides. For these genomes, a number of conserved genes have already been identified by the biological community. These genomes and their corresponding conserved genes were published in Herniou et al.[6]. Since these genomes do not show a high level of similarity, we align the translated protein sequences instead of the DNA sequences of the genomes. We used 35 pairs of virus genomes for experiments as one of the pairs shows an exceptionally high similarity and is excluded from our experiment. Details of the data sets are given in Table 4 of the Appendix.

Findings: We have tried different values for x and minimum anchor length L in the experiments. We found that it is sensible to set $x = 5$ and $L = 13$. Figure 1 shows the

[a]For GAME, we also tried some other values for the parameters and the results are similar.

coverage of the three software tools based on different anchors in 35 test cases. In general, the x-mismatch anchors outperform the other two types of anchors in almost all cases. More precisely (see Table 1), for MUMmer-3, x-mismatch anchors achieve 9% higher coverage than both EM-MUM and the anchors from GAME on average. For MaxMinCluster, x-mismatch anchors achieve 8% higher coverage than both EM-MUM and the anchors from GAME on average. For MSS, x-mismatch anchors achieve 14% higher coverage than EM-MUM and 10% higher coverage than the anchors from GAME on average. Also, x-mismatch anchors can maintain a high sensitivity while achieving a higher coverage.

In fact, we have further investigated the input anchors, we found that the set of x-mismatch anchors covers more conserved genes than the other two types of anchors. On average, 78.7% of published conserved gene regions are found to be overlapped by x-mismatch anchors[b]. For EM-MUM and anchors from GAME, the percentages are relatively lower (only 66% and 68.5%, respectively). Recall that these percentages are roughly the upper bound for the coverage of the software tools. From these figures, we can also see that the effectiveness of x-mismatch anchors seem to be higher.

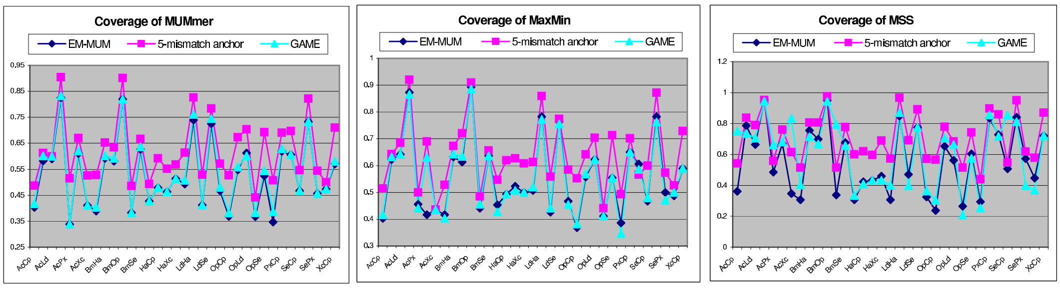

Figure 1. Effectiveness of anchors on 35 virus pairs

Aligning Closely Related Genomes: Besides virus genomes, we have also performed experiments on human-mouse chromosome pairs. Since human and mouse are closely related species, we align the DNA sequences of the genomes in order to see the differences in effectiveness of the anchors. We have used 10 pairs of chromosomes of length from 14M to 65M nucleotides. Details of the data sets are given in Table 5 in the Appendix.

Findings: Note that the sequences are about 100 times longer than those of virus genomes. For GAME, the input anchor sets are too large to be processed by the software tools. On average, the number of anchors from GAME is about 36M while for EM-MUM, there are about 52K anchors and for x-mismatch anchors (Note that we tried a few x values and a few values for setting the minimum anchor length L. It seems reasonable to set $x = 1$ and $L = 20$ as the genomes are closely related), there are about 476K anchors only. The reason for the large volume of anchors in GAME is that it does not require the anchors to be unique

[b] A region is considered to be covered by the set of anchors if the region overlaps with anchors of total length of at least 8.

Table 3. Performance of x-mismatch anchors on 10 human-mouse chromosome pairs.

	EM-MUM		**1-Mismatch Anchors**	
	Coverage	Sensitivity	Coverage	Sensitivity
MUMmer-3	57.5%	31.5%	70.0%	31.4%
MaxMinCluster	72.2%	32.4%	89.9%	32.5%
MSS	87.5%	30.0%	94.6%	30.1%

in the genomes. So, we only compare the x-mismatch anchors with the EM-MUM. The result is shown in Table 3. The x-mismatch anchors also show a significant improvement in terms of coverage while maintaining more or less the same sensitivity as that of EM-MUM. The increase in coverage is about 7 - 17%.

However, as a remark, if the alignment is performed on the translated protein sequences, the improvement is smaller and is of a few percentages (1-6%) by using x-mismatch anchors. The small improvement is due to the fact that the coverage using EM-MUM is already high (about 90%) as the species are closely related. In real applications, we should try to align the translated protein sequences (especially for distant species). So, the results for aligning DNA sequences of the human-mouse chromosome pairs are for reference to illustrate the effectiveness of x-mismatch anchors.

To conclude, using x-mismatch anchors is more effective than EM-MUM and anchors from GAME. In the next section, we will discuss how to generate the set of x-mismatch anchors, especially for long sequences.

3. The Anchor Generation Algorithms

In this section, we propose two practical algorithms, Suffix-Exd and Hash-Tab, for generating x-mismatch anchors given two genomic sequences A and B. By making use of the suffix links, we also show how to speed up Suffix-Exd for the case of $x \leq 3$. To start with, we first present a suffix tree based brute-force approach. Recall that when generating the anchors, we require the length of an anchor to be at least L as very short anchors most likely come from noise. Let A, B be the two given genomes.

The Suffix Tree Based Brute-force Approach: We first build a suffix tree T_{A+} for *A*+, then for each position i of *B*+, we aim at locating all substrings s in *A*+ that satisfy the following. (1) s is of length at least L; (2) s is unique in *A*+; (3) there is a corresponding substring t in *B*+ starting at position i such that $\texttt{Hamm\_dist}(s,t) \leq x$ and (s, t) is maximal. We search the suffix tree T_{A+} in a brute-force manner. Based on the characters at $i, i+1, \ldots$ of *B*+, we search T_{A+}. Since we allow x mismatches, we try all branches at every node and keep track the number of mismatches for each branch with respect to the corresponding substring in *B*+. Output the substring s in the tree if it satisfies the above three conditions.

For each pair (s, t) reported, we check the uniqueness of s and t by searching the suffix trees of *A*-, *B*+, and *B*-. Finally, to satisfy the one-to-one condition (Condition (3) of Definition 2.1), the remaining (s,t) pairs will go through a simple checking procedure (the details will be given in the full paper). Then, repeat the same procedure by building T_{A-} for *A*- and using *B*+ to search for x-mismatch anchors with respect to *B*+ and *A*-.

The brute-force approach is easy to implement, but is too slow, especially for long genomic sequences and large x values. Table 2 shows that it takes 12 hours to enumerate the anchor set for a human-mouse chromosome pair which are of size 28M and 14M.

The Suffix-Exd Approach: In the brute-force approach, for large values of x, a large portion of the tree will be searched and this slows down the searching process. The idea of the Suffix-Exd approach is given in the following lemma based on the pigeon-hole principle.

Lemma 3.1. *Let $s[1..\ell]$ and $t[1..\ell]$ be substrings in the genomes A and B, respectively such that* $\texttt{Hamm\_dist}(s,t) \leq x$. *Then, either* $\texttt{Hamm\_dist}(s[1..\lfloor \ell/2 \rfloor], t[1..\lfloor \ell/2 \rfloor]) \leq \lfloor x/2 \rfloor$ *or* $\texttt{Hamm\_dist}(s[\lfloor \ell/2 \rfloor + 1..\ell], t[\lfloor \ell/2 \rfloor + 1..\ell]) \leq \lfloor x/2 \rfloor$.

Roughly speaking, the above lemma says that if s and t is an x-mismatch anchor, then either the first half or the second half of s and t contain at most $x/2$ mismatches. In other words, there must be substrings (either prefixes or suffices) in s and t of length exactly $\lfloor L/2 \rfloor$ with at most $x/2$ mismatches. (Recall that L is the minimum anchor length.)

So, we can search the suffix tree for these substrings (with fewer mismatches) as seeds in order to avoid searching a large portion of the tree. We then extend from these seeds to locate the anchor set. The details are as follow. For each substring q of length exactly $\lfloor L/2 \rfloor$ in *B*+, we search the suffix tree T_{A+} for substrings p (the *seeds*) such that $\texttt{Hamm\_dist}(p,q) \leq \lfloor x/2 \rfloor$. We call this step the *seed finding step*. Note that we search for shorter, fixed length substrings with fewer mismatches in the suffix tree so as to speed up the process. Then, we extend each (p,q) pair to (p',q') such that p' and q' are maximal, of length $\geq L$, and $\texttt{Hamm\_dist}(p',q') \leq x$. We can then go through the same checking as in the brute-force approach to make sure that p', q' are unique and satisfy the one-to-one condition. Again, we repeat the procedure for T_{A-} and *B*+. From Table 2, we can see that the speed up is about 6 times.

The Hash-Tab Approach: For long sequences, building suffix tree requires a lot of memory. The Hash-Tab approach solves the memory problem as follows. In the seed finding step, instead of using suffix tree, we build a hash table to store the locations of all possible substrings of fixed length in $A+$. Then, for each substring in $B+$, we search the hash table for matching strings in $A+$. To check the uniqueness, building a single suffix tree may not be feasible. So, we can divide the genome into several regions, build multiple suffix trees, then we check all these suffix trees to guarantee the uniqueness. The details will be given in the full paper. The Hash-Tab approach is slower than the Suffix-Exd approach, but it can save a lot of memory. Table 2 shows that the Hash-Tab approach requires 5 times less memory while the running time is still significantly faster than the brute-force approach.

Speeding Up the Suffix-Exd Approach: Recall that in the seed finding step of the Suffix-Exd approach, for each substring q of length exactly $\lfloor L/2 \rfloor$ in *B*+, we search the suffix tree T_{A+} for substrings p such that $\texttt{Hamm\_dist}(p,q) \leq \lfloor x/2 \rfloor$. Assume that we have searched the suffix tree T_{A+} for $p = \alpha u$ where p is a substring in *B*+ and α is a single nucleotide (character), the following lemma shows how to speed up the searching of u by making use of the suffix links in T_{A+}. Let $r = \lfloor x/2 \rfloor$.

Lemma 3.2. *Let $p = \alpha u$ be a substring in B+ and α is a single nucleotide. Let N*

be an internal node in T_{A+} with path label q representing a substring in $A+$ such that $\texttt{Hamm\_dist}(p, q) \leq r$. *Let N' be the node pointed by the suffix link of N and q' be the path label of N'. Then,* $\texttt{Hamm\_dist}(u, q') \leq r$.

From the above lemma, assume that we have finished searching the suffix tree for the substrings p starting at position i in *B*+, if we can keep track of all corresponding locations of N', then we can speed up the searching for substrings starting at position $i+1$. If $r = 1$, we have a simple data structure to do this. In other words, using suffix link, we can easily speed up the seed finding step of Suffix-Exd for $x \leq 3$. The speed up can be shown to be $\lfloor L/2 \rfloor$ times. The details of the proof and the data structure will be given in the full paper.

4. Conclusion

In this paper, we consider the effectiveness and the generation of anchors with mismatches for whole genome alignment. We formally defined an x-mismatch anchor. We then compare the effectiveness of x-mismatch anchors with exact match maximal unique substrings (EM-MUM) and the anchors from GAME (the most recent work that also uses anchors with mismatches) based on a set of experiments on 35 pairs of virus genomes and 10 pairs of human-mouse chromosome pairs using three software tools (MUMmer-3, MaxMinCluster, MSS). The results show that the effectiveness of x-mismatch anchors is higher than the other anchors. We also discuss the issues (time and memory) involved in generating x-mismatch anchors. A straightforward suffix tree based approach uses too much time and memory for long sequences. We propose several practical algorithms to tackle the generation problem. However, designing faster algorithms that use less memory is still a challenging problem and desirable for handling long genomic sequences.

References

1. Brona Brejova, Daniel Brown, and Tomas Vinar. Vector seeds: An extension to spaced seeds allows substantial improvements in sensitivity and specificity. In *Proceedings of the 3rd International Workshop in Algorithms and Bioinformatics (WABI 2003)*, pages 39–54, 2003.
2. HL Chan, TW Lam, WK Sung, Prudence WH Wong, and SM Yiu. A mutation-sensitive approach for locating conserved gene pairs between related species. In *Proceedings of the IEEE Fourth Symposium on Bioinformatics and Bioengineering (BIBE 2004)*, pages 545–552, 2004.
3. J.-H. Choi, H.-G. Cho, and S. Kim. GAME: Genome alignment by match extension. In *Proceedings of the Computational Systems Bioinformatics Conference (CSB 2004)*, 2004. To appear.
4. A. Cobbs. Fast approximate matching using suffix trees. In *Proceedings of the 6th Annual Symposium on Combinatorial Pattern Matching (CPM'95)*, pages 41–54, 1995.
5. A.L. Delcher, A. Phillippy, J. Carlton, and S.L. Salzberg. Fast algorithms for large-scale genome alignment and comparison. *Nucleic Acids Research*, 30(11):2478–2483, 2002.
6. E.A. Herniou, T. Luque, X. Chen, J.M. Vlak, D. Winstanley, J.S. Cory, and D.R. O'Reilly. Use of whole genome sequence data to infer baculovirus phylogeny. *Journal of Virology*, 75(17):8117–8126, 2001.
7. N. Jareborg, E. Birney, and R. Durbin. Comparative analysis of noncoding regions of 77 orthologous mouse and human gene pairs. *Genome Research*, 9:815–824, 2000.
8. J. Kent and M. Zahler. The intronerator: Exploring introns and alternative splicing in c. elegans genomic alignment. *Genome Research*, 10:1115–1125, 2000.

9. S. Kurtz, A. Phillippy, A.L. Delcher, M. Smoot, M. Shumway, C. Antonescu, and S.L. Salzberg. Versatile and open software for comparing large genomes. *Genome Biology*, 5(2), 2004.
10. T. W. Lam, N. Lu, H. F. Ting, Prudence W. H. Wong, and S. M. Yiu. Efficient algorithms for optimizing whole genome alignment with noise. In *Proceedings of the 14th International Symposium on Algorithms and Computation*, pages 364–374, 2003. Also to appear in Bioinformatics.
11. B. Ma, J. Tromp, and M. Li. PatternHunter: Faster and more sensitive homology search. *Bioinformatics*, 18(3):440–445, 2002.
12. B. Morgenstern, K. Frech, D. Dress, and T. Werner. Dialign: Finding local similarities by multiple sequence alignment. *Bioinformatics*, 14:290–294, 1998.
13. G. Navarro and R. Baeza-Yates. A hybrid indexing method for approximate string matching. *Journal of Discrete Algorithms*, 1(1):205–239, 2000.
14. G. Navarro, R. Baeza-Yates, E. Sutinen, and J. Tarhio. Indexing methods for approximate string matching. *IEEE Data Engineering Bulletin*, 24(4):19–27, 2001.
15. E. Ukkonen. Approximate matching over suffix trees. In *Proceedings of the 4th Annual Symposium on Combinatorial Pattern Matching (CPM'93)*, pages 228–242, 1993.
16. P. Vincens, L. Buffat, C. Andre, J.P. Chevrolat, J.F. Boisvieux, and S. Hazout. A strategy for finding regions of similarity in complete genome sequences. *Bioinformatics*, 14:715–725, 1998.

Appendix

Virus Genome	Length (bp)
AcMNPV(Av)	134K
BmNPV(Bm)	128K
OpMNPV(Op)	131K
LdMNPV(Ld)	161K
SeMNPV(Se)	136K
HaSNPV(Ha)	131K
XcGV(Xc)	179K
PxGV(Px)	101K
CpGV(Cp)	124K

Table 4. Details of the 35 baculovirus pairs

Exp. No.	Virus Pair	# of Conserved Genes	Exp. No.	Virus Pair	# of Conserved Genes	Exp. No.	Virus Pair	# of Conserved Genes
1	AcCp	72	13	BmSe	99	25	OpLd	98
2	AcHa	98	14	BmXc	75	26	OpPx	68
3	AcLd	95	15	HaCp	71	27	OpSe	101
4	AcOp	126	16	HaPx	67	28	OpXc	75
5	AcPx	68	17	HaXc	74	29	PxCp	97
6	AcSe	100	18	LdCp	75	30	PxXc	99
7	AcXc	78	19	LdHa	92	31	SeCp	75
8	BmCp	72	20	LdPx	68	32	SeHa	101
9	BmHa	98	21	LdSe	102	33	SePx	68
10	BmLd	93	22	LdXc	77	34	SeXc	76
11	BmOp	122	23	OpCp	76	35	XcCp	107
12	BmPx	68	24	OpHa	95			

Table 5. Details of the 10 human-mouse chromosome pairs

Exp. No.	Mouse Chr. No.	Human Chr. No.	Length of Mouse Chr.	Length of Human Chr.	# of Published Conserved Genes
1	2	15	51M	54M	51
2	7	19	22M	31M	192
3	9	11	51M	47M	101
4	14	8	39M	18M	38
5	15	22	65M	29M	72
6	16	16	63M	26M	31
7	16	22	63M	27M	30
8	17	16	15M	29M	46
9	17	19	31M	40M	30
10	19	11	30M	14M	93