GAME: Genome Alignment by Match Extension

Jeong-Hyeon Choi **Hwan-Gue Cho**
Department of Computer Science
Pusan National University, Korea
`jeochoi@indiana.edu` `hgcho@pusan.ac.kr`

Sun Kim
School of Informatics
Center for Genomics and Bioinformatics
Indiana University, IN 47401, USA
`sunkim@bio.informatics.indiana.edu`
`Phone: 812-856-3009 Fax: 812-856-4764`

Abstract

In this paper, we present a new adaptive genome sequence alignment based on maximal exact match (MEM) anchor. The major problem with the use of MEM anchor is that the number of hits in non-homologous regions increases exponentially when shorter MEM anchors are used to detect more homologous regions. To deal with this problem, we have developed a fast and accurate anchor filtering scheme based on simple match extension. Due to its simplicity and accuracy, all MEM anchors in a pair of genomes can be exhaustively tested and filtered. In addition, by incorporating the translation technique, the aligment quality of our genome alignment algorithm has been further improved. As a result, our genome alignment algorithm outperforms existing algorithms and can align large genomes, e.g., *A. thaliana*, without the typical large memory requirement problem. This is shown using an experiment which compares the performance of PatternHunter, Blastz, MUMmer and our algorithm in aligning all 45 pairs of 10 bacterial genomes. The scalability of our algorithm is shown using another experiment which compares all pairs of 5 chromosomes in *A. thaliana*.

Keywords: whole genome sequence alignment, anchor filtering, maximal exact match

1 Introduction

Recent advances in both sequencing technology and algorithm development for genome sequence software have made it possible to determine the sequence of whole genomes. As a consequence, the number of completely sequenced genomes is increasing rapidly. However, algorithm development for genome annotation has been relatively slow and annotation of completely sequenced genomes inevitably depends on human expert knowledge. The most effective method to understand genome content is to compare multiple genomes, especially when they are close enough to share common subsequences. One important computational method is to align whole genomic sequences.[1] Aligning whole genomes are useful in several ways. For example, sequencing a genome can be much easier and more accurate by aligning its contigs to completely sequenced genomes that are close to the one being sequenced. Another example of use of whole genome alignment is to identify conserved regions among multiple genomes which include not only common genes but regulatory regions and non-coding RNA sequences. A recent paper demonstrated that intergenic regions in multiple Yeast strains can be detected by comparing genomic sequences [7].

[1] In this paper, we will use *genomes* and *genomic DNA sequences* interchangeably.

Work on comparing human and mouse sequences has also demonstrated the possibility of predicting functions and structure of human genes by genome alignment method, e.g., [14].

There has recently been a significant development of whole genome comparison algorithms [1, 2, 4, 8, 10, 14]. These algorithms have been effectively used for several purposes some of which we discussed above. In this paper, we propose a new genome sequence alignment algorithm based on maximal exact match anchors. The main contributions of this paper are:

- We have developed a simple but accurate gap-free anchor filtering scheme. Thanks to its simplicity and accuracy, *all* initial anchors in a pair of entire genomes can be exhaustively tested and filtered. More importantly, its specificity (the ratio of correctly rejecting anchors in non-homologous regions) is almost 1.0. As a result, our alignment algorithm achieves alignment quality better than those from the best performing algorithms.
- The filtering scheme makes it possible to use maximal exact match anchors to align relatively distant genomes. Our alignment algorithm thus allows users to *adjust* the level of sensitivity by using shorter exact match anchors without significant increase in incorrect alignment regions (due to the effective anchor filtering scheme).
- The translation technique increased sensitivity of initial MEM anchors, resulting in better alignment quality.
- Our alignment algorithm does not suffer from typical large memory requirement and can compare large genomes, e.g., all pairs of 5 *A. thaliana* chromosomes with less than 750MB.

2 Anchor-based Genome Alignment

Aligning two genomic sequences requires detecting numerous local alignments – homologous regions such as genes common to two genomes. Due to the large size of genomes, it is not possible to use the dynamic programming technique that requires quadratic amount of time and space in relation to the size of genomes. For this reason, almost all genome sequence alignment algorithms first find anchors and extend or combine them to generate sequence alignments.

There are two common approaches used in detecting anchors, one based on exact matches and the other based on approximate patterns. For those based on exact matches, it is intuitive to use patterns of a fixed length k as anchors, but they are generally too noisy, i.e, too many anchor matches in non-homologous regions. Noise can be significantly reduced by imposing the maximality constraint on exact matches, i.e., only matches that cannot be extended neither left nor right are accepted. These anchors are called *maximal exact matches (MEMs)* of length k, i.e., MEMs longer than k bases. MUMmer [2] came up with the creative idea of using *maximal unique matches (MUMs)* requiring the uniqueness of MEMs. However, as the authors pointed out, MEMs and MUMs are only designed to align two closely related genomes and are not very successful in aligning relatively distant genomes. For this reason, there has recently been growing interest in using approximate patterns as anchors. The most popular approach is to use *gap-free pattern signature*; for example, `1110100110010101111` denotes that a match should occur at positions with `1`'s but a mismatch is allowed at positions with `0`'s. One natural question is why particular signatures are used. In short, assuming some statistical background probability of matching regions, the best or near best performing signatures are computed; a dynamic programming approach is used in PatternHunter [10]. Among various anchor types, the pattern signatures used in PatternHunter, later adopted by Blastz [14], has proved one of the best and has achieved BLAST quality alignment.

In addition, there are anchor types that incorporate biologically important features. For example, WABA [8] is specifically designed to look for variations in synonymous sites (variations in the third base in the same amino acid code) and uses a pair hidden Markov model to combine anchors. GLASS [1] iteratively finds exact matching anchors, but specifically looks for alignments in exon regions – often as short as 50 bases and flanked by much longer regions of poor similarity. Almost all genome alignment algorithms employ one of the anchor types described above.

2.1 Motivation: Increasing sensitivity and specificity of anchors

In the framework of the anchor-based genome alignment algorithms, it is obvious that the performance depends upon the accuracy of anchors. Thus our focus is on increasing the sensitivity and specificity of anchors. To make our discussion simple in this section, we abuse sensitivity and specificity: increasing the number of anchors in homologous regions (increased sensitivity) and decreasing the number of anchors in non-homologous regions (increased specificity). Increasing both sensitivity and specificity is very difficult and the key issue is to balance sensitivity and specificity. For example, decreasing the length k of MEMs will result in more hits in homologous regions, but a lot more hits in non-homologous regions – increased sensitivity with a sharp decrease in specificity. The pattern signature anchors are good, balancing sensitivity and specificity. However they should use predefined optimal signatures that must be computed in advance, thus they are not flexible to adjust the level of sensitivity.

In this paper, we propose *an anchor filtering approach* to improve specificity of anchors. Users can tune parameters of the alignment algorithm, if applicable to the anchor type, to increase sensitivity – independent of increase in specificity. As we pointed out, increasing sensitivity of MEM anchors of length k is achieved by straightforward use of shorter anchors, i.e., smaller value of k. Thus, we are interested in using the MEM-based anchors, which can lead to the development of a new *adaptive* genome alignment algorithm. In the next section, we present a very fast and accurate filtering scheme for MEMs with an empirical analysis.

3 Extended MEM (EMEM): Anchors by Gap-free Extension

The key idea is simply to extend MEMs without computing alignment scores until the percent identity becomes lower than the ones used in existing genome alignment algorithms. The EMEM anchors are generated in two steps:

1. All MEMs are detected using suffix array [11]. As pointed out in [10], use of suffix tree suffers the intrinsic large memory space problem. However, the space problem can be significantly reduced by using suffix array [4].

2. Each MEM anchor is tested with a filter, called *the MEM filter*. More specifically, given a

maximal exact match (MEM) m of length l_m and a threshold T_{pi} for the minimum percent identity, the *gap-free match extension* of m is defined as the extent up to which m can be extended without gaps before its percent identity drops below T_{pi}. Let *gfe(m)* denote the gap-free extension of m and $|\text{gfe}(m)|$ denote its length. Given a threshold T_e for the minimum extension length, *the MEM filter* is defined to filter (discard) any m that fails to extend beyond T_e , i.e., $|gfe(m)| < T_e$.

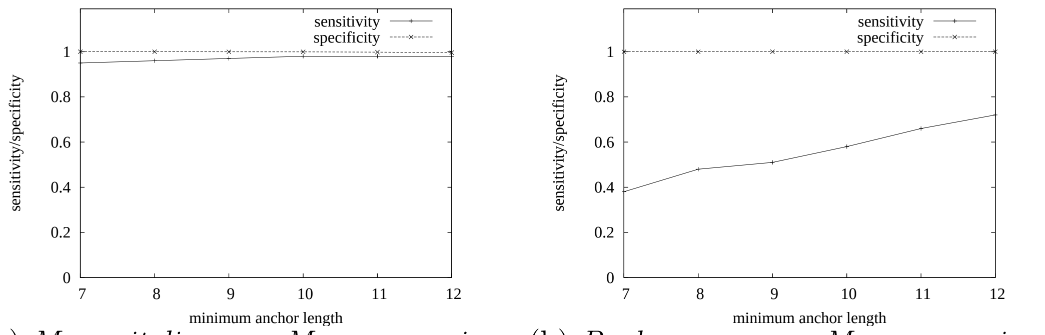
Due to its simplicity, the filtering scheme can further utilize a well developed string pattern matching technique using bit parallelism; multiple characters can be compared simultaneously using the technique, e.g., [9]. Our filtering scheme is accurate (Section 3.1) and runs exceptionally fast, performing about 1.5 million filtering tests per second on a 1.7 Ghz Pentium machine, which makes it possible to test *all* initial anchors exhaustively in a pair of genomes. As a result, our alignment algorithm can align genomes fast and accurately (see Section 6).

3.1 Performance evaluation of the MEM filtering scheme

In this section, we will evaluate the performance of the MEM filtering scheme in terms of its sensitivity and specificity. In addition the performance of three anchor types, MEM, MUM, and EMEM will be compared in terms of the number of hits in COG families. Since MEM anchors are designed to align closely related genomes, we will use two pairs of genomes – *M. genitalium* vs. *M. pneumoniae* (closely related) and *Buchnera sp.* vs. *M. pneumoniae* (relatively distant). For our filtering scheme, we used threshold values, $T_{pi} = 0.5$ and $T_e = 200$.

Performance of the MEM filter: The filtering scheme is general enough to be used for anchors of any type. In this experiment, we have used MEM anchors since it is embedded in our genome alignment algorithm implementation. To compute the sensitivity and specificity of the filtering scheme, we had to classify the MEM anchors (input to the filtering scheme) as correct ones and incorrect ones. In general, this information is not available – this is one of the ultimate goals in bioinformatics. Fortunately, a majority of genes in pairs of genomes are classified in the recent release of the COG database in 2003 as a result of comparing 66 genomes. Thus we have classified the MEM anchors according to the COG family classification, i.e., anchors within matching COG genes are correct hits and the remaining ones are incorrect hits.[2] Evaluation in terms of matching genes is also consistent with Webb Miller's view that the primary use of genome alignment tools is to detect coding regions [12]. Once we classified the MEM anchors as correct and incorrect ones, we could evaluate the performance of our filtering scheme. Given MEMs, detected using suffix array, the filtering test is performed for each MEM. If a MEM passes the test, it is accepted (*positives*). Otherwise, it is rejected (*negatives*). Among positives, we know the ones that are correct (*true positives (TPs)*) and incorrect (*false positives (FPs)*). Among negatives, we know the ones that are correctly rejected (*true negatives (TNs)*) and incorrectly rejected (*false negatives (FNs)*). Then sensitivity is defined as TPs/(TPs+FNs) and specificity as TNs/(TNs+FPs). Figure 1 shows the sensitivity and specificity plots for the two pairs of genomes

[2]We aligned all matching COG genes using `stretcher` from the EMBOSS package, an implementation of the Needleman-Wunsch Algorithm, and extracted, from the alignments, all maximal exact matches which are then compared to the MEM anchors to classify them as correct and in correct ones. See [5] for detail.



(a) *M. genitalium* vs. *M. pneumoniae* (b) *Buchnera sp.* vs. *M. pneumoniae*

Figure 1: Sensitivity and specificity of the anchor filtering scheme.

with varying minimum MEM lengths from 7 to 12. The filtering scheme achieved almost perfect specificity, over 0.999, for all MEM lengths in comparing both pairs of genomes. This is very important since users can use shorter MEMs while still maintaining a very high level specificity. Sensitivity for the *M. genitalium* and *M. pneumoniae* pair was also very high; the lowest number was 0.95 for the MEM length 7. For the relatively distant *Buchnera sp.* vs. *M. pneumoniae* pair (Figure 1 b), sensitivity dropped significantly to 0.38 for the MEM length 7. An interesting observation is that more common genes were detected as the sensitivity of the filtering scheme decreased (Figure 3 b and Figure 1 b). Why does this happen? Sensitivity decreases for shorter MEMs but also as the MEM length decreases, the number of correct anchors increases since the number of total MEMs increases "exponentially". As a result the percentage of common genes detected by EMEM anchors increases for shorter MEMs (Figure 3 b) even though our filtering scheme discards some correct anchors erroneously. For the *Buchnera sp.* vs. *M. pneumoniae* pair, the number of correct EMEM anchors increased; 47, 99, 110, 155, 329, and 409 for MEMs of length 12, 11, 10, 9, 8, and 7 respectively. Thus the performance of our algorithm, in terms of the common genes detected, consistently improves for shorter MEM anchors (see Figure 3 and Section 6). However, it will be desirable to increase sensitivity, while keeping specificity at a very high level, for distantly related genomes. We are currently working on an adaptive filtering scheme to increase sensitivity (see Section 3.2 for more detail). We emphasize the importance of achieving a higher specificity to eliminate anchor hits in non-homologous regions, which will otherwise result in aligning many non-homologous regions.

Performance of the MEM filtered anchors The previous experiment illustrated the performance of the filtering scheme. This experiment compares the performance of three anchor types, MEM, MUM, and EMEM from a different perspective. The performance of the three anchor types with the MEM length of 8 is visualized in Figure 2. As shown, EMEM anchors consistently cover the diagonal regions while there are few outside the diagonal regions. To evaluate the performance more systematically, we measured the ratio of the number of COG families detected by anchor hits to the number of COGs common in the *Buchnera sp.* vs. *M. pneumoniae* pair (relatively distant) with varying minimum MEM lengths from 7 to 12 (Figure 3). Note that the number of common genes detected by MEMs increased for shorter lengths. This shows that

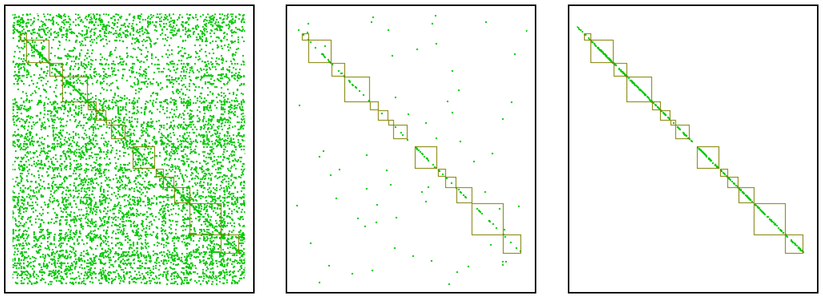


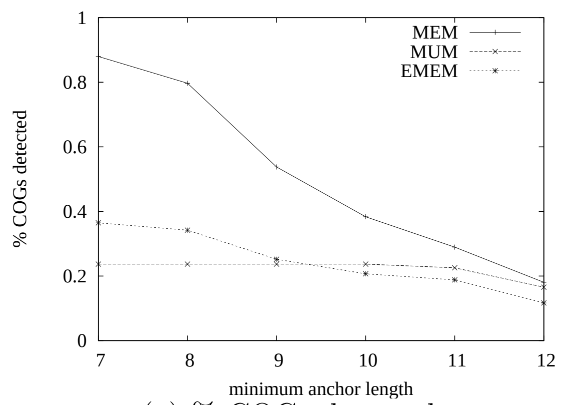
Figure 2: Anchor hits in the matching regions of two genomes *M. genitalium* (501960:516450) vs. *M. pneumoniae* (717850:734750) using MEM (plot a), MUM (plot b), and EMEM (plot c). Dots in the plots denote anchors. Boxes in the plots are the regions where matched COG genes, (atpC, atpD, atpG, atpA, atpH, atpF, atpB, eno, pmsR, phoU, pstB, pstA), are present, so anchors within the boxes are successful ones while anchors outside the boxes are not successful anchors.

exact matches can be used for aligning of relatively distant genomes as an alternative to sophisticated pattern signatures. The number of common genes detected by EMEMs also increased for shorter lengths while those detected by MUMs were steady. In addition to the number of genes detected, we measure the ratio of TP/FP to compare correct anchors vs. incorrect ones. As shown in Figure 3, EMEM anchors can detect common genes with much smaller portion of FPs than MEM and MUM anchors. The ratios of TP/FP for EMEM range from 1.3 to 2.7 while those for MEM and MUM are extremely low; the maximum values are 0.0002 for MEM and 0.0008 for MUM. Experimental data is available on the web [5]. An interesting observation is that MUM outperformed EMEM for the cases of MEM length greater than 9. This suggests that the requirement for gap-free extension $T_{pi} = 0.5$ and $T_e = 200$ can be relaxed for longer MEMs. This is discussed in Section 3.2.

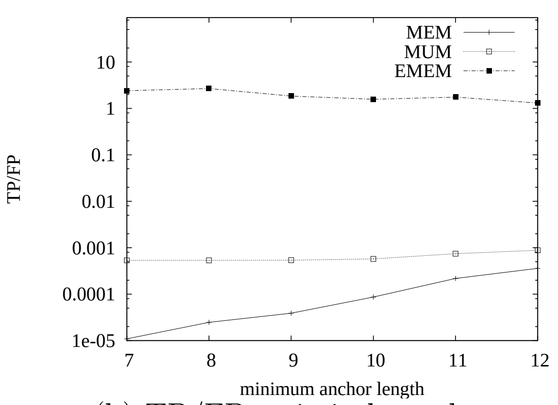
3.2 Why does the MEM filter work?

The MEM filtering scheme considers gap-free extensions only. Blastz also extends matches, while computing alignment scores, first without gaps and later allowing gaps. In contrast, we allow much lower percent identity ($T_{pi} = 0.5$) with a simple scheme of just counting matches as opposed to computing the alignment score. It is not immediately clear why our filtering scheme works as demonstrated in the previous section. We will address this issue in two steps:

1. Anchors in two random sequences are highly unlikely to extend beyond a certain length. Thus we argue that gap-free extensions longer than a certain length are *statistically significant*. Using the random walk model (pp 219–222 in [3]) with $T_{pi} = 0.5$, an exact match of length l_m can be extended on average only to $2 \times l_m$, which is much less than $T_e = 200$. This can also be empirically demonstrated. Plot (a) in Figure 4 shows the "maximum" lengths of EMEMs in random sequences (green line) and the lengths of EMEMs in *M. genitalium* and *M. pneumoniae* (red dots). Drawing a cutoff line at $T_e = 200$, only about 10 among over 26 million matches in the two genomes falls below the maximum line length.



(a) % COGs detected



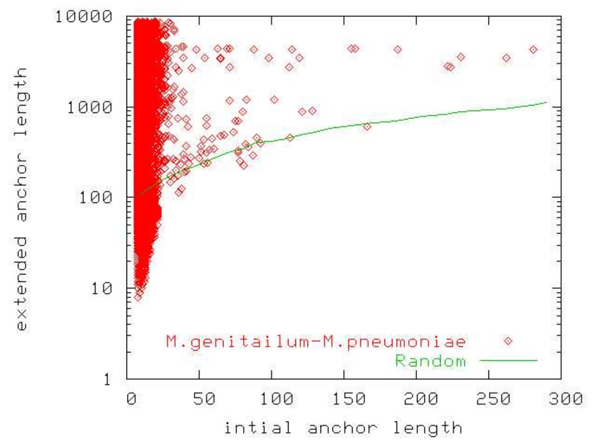
(b) TP/FP ratio in logscale

Figure 3: Performance of MEM, MUM and EMEM anchors for *Buchnera sp.* vs. *M. pneumoniae*. The ratios of the number of COG families detected by anchor hits to the number of COGs common is shown in Plot (a) and TP/FP ratios in logscale in Plot (b). The majority of MUM and MEM anchors are in non-homologous regions while a small portion of EMEM anchors are in non-homologous regions.

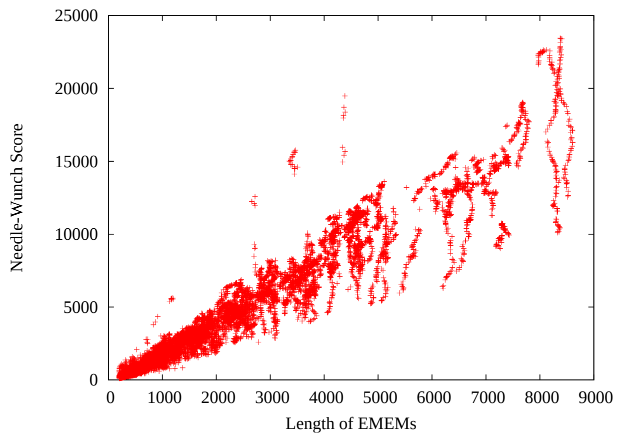
2. There is a very strong correlation between the sequence alignment score and the length of gap-free extension. Plot(b) in Figure 4 shows the correlation between the length of gap-free extensions and the alignment score computed using `stretcher` from the EMBOSS package.

In search of the best parameter values for T_e and T_{pi}, we performed a series of experiments with varying parameter values, 100, 150, 200 for T_e and 0.5, 0.55, 0.6 for T_{pi}, for several pairs of genomes. The parameter values of $T_e = 200$ and $T_{pi} = 0.5$ consistently performed better than others (see [5] for more detail).

An adaptive filtering scheme to increase sensitivity: As shown in Figure 4, a longer maximal exact match m will extend longer by chance until its percent identity drops below $T_{pi} = 0.5$. This is quite intuitive, but we provide a short statistical argument and suggest, as a future study, a parameter setting strategy for T_e and T_{pi} that is adaptive to the length of m. What is shown in the plot(a) of Figure 4 is a conditional probability $Pr(|gfe(m)| > T_e \mid m)$ for a *given* m of length l_m. However, our real interest is how likely the extended gap-free alignment is by chance, i.e., a joint probability $Pr(|gfe(m)| > T_e,\ m)$. Using Bayes rule, $Pr(|gfe(m)| > T_e,\ m) = Pr(|gfe(m)| > T_e \mid m) \times Pr(m)$. Recall that our goal is make $Pr(|gfe(m)| > T_e,\ m)$ very small, i.e., to the value of occurring highly unlikely by chance, say 10^{-10}. Since $Pr(m)$ decreases exponentially for longer l_m, we can relax T_e in $Pr(|gfe(m)| > T_e \mid m)$ for longer l_m, while maintaining the joint probability at the same confidence level, thus a less stringent value for T_e can be used given a longer match m. Indeed, we estimated the value of T_e with a fixed joint probability value using statistics of EMEMs from two random sequences and this confirmed that less stringent values for T_e can be used for longer matches. See [5] for more detail on a work in progress.



(a)



(b)

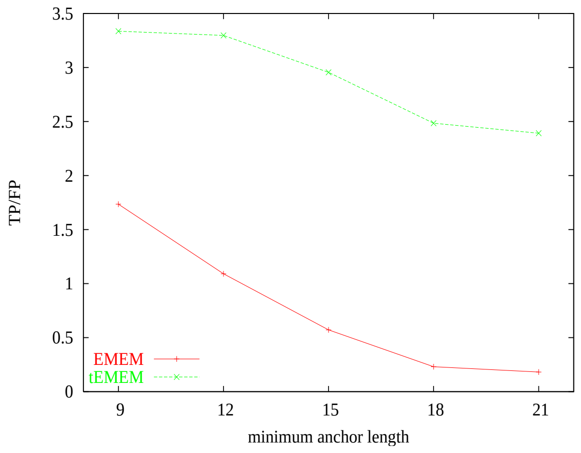
Figure 4: Plot (a) shows the *maximum* lengths of EMEMs in random sequences (green line) and the lengths of EMEMs in *M. genitalium* and *M. pneumoniae* (red dots). Plot (b) shows correlation between the length of EMEMs and the alignment score computed using `stretcher` from EMBOSS.

4 Improving sensitivity of anchors using the translation techniques

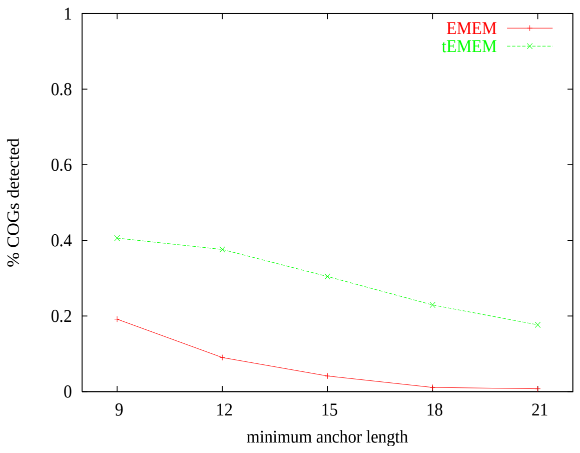
The MEM filtering scheme is to filter out MEMs that are likely to hit non-homologous regions for a *given* set of initial MEM anchors. Increasing the sensitivity of initial MEM anchors will also improve the alignment quality. In this section we will show that the translation technique effectively increases the sensitivity of initial MEM anchors. As described in Section 2, two methods to increase sensitivity of anchors are use of smaller exact patterns and use of pattern signatures to allow mismatches. Here we propose a method to combine both, allowing mismatches in terms of amino acid scoring matrices such as BLOSUM or PAM while shorter patterns can be used as anchors. This method has already been used in a new version of MUMmer [2]. What is new in our method is to combine our novel filtering scheme and translation technique. Use of amino acid scoring matrices has clearly advantage over use of a set of fixed pattern signatures since different matrices can be used for different phylogenetic distance. Let *tMEM*, m, of length k to denote MEM of length k that is translated into amino acid and matched in terms of the translated amino acid sequences and let *tEMEM* to denote tMEMs that pass the filtering test. Below is a procedure to detect tMEMs and tEMEMs of length k for a pair of genomes, G_1 and G_2, for one strand (3-frame translation).

1. Since we do not know exactly where translation starts, we use the standard 6-frame translation, i.e., translation of a DNA sequence into an amino acid sequence is tried at every possible three positions for each of the two DNA strands. Let A_1^1, A_1^2, and A_1^3 denote each of the three frame translation of G_1 respectively, and A_2^1, A_2^2, and A_2^3 denote of G_2 respectively.

2. A suffix array is built using a concatenated string of A_1^1, A_1^2, A_1^3, A_2^1, A_2^2, and A_2^3. Any substring of length $\frac{k}{3}$ or longer becomes tMEM.



(a) TP/FP ratios



(b) Ratios of # COGs detected

Figure 5: The TP/FP ratios and the percent of COG families detected are shown for the *Buchnera sp.* vs. *M. pneumoniae* genome pair. The TP/FP ratio of tEMEM were over 2.0 even for longer lengths, say 21, while the ratio of EMEM were below 1.0, i.e. more FPs than TPs, for the length 12 or longer. The percent of COGs detected also showed the effectiveness of tEMEMs compared to EMEMs for the relatively distant genome pair.

3. For each tMEM, m, a filtering test is performed using DNA sequences, G_1 and G_2, not translated sequences. If m passes the filtering test, it becomes tEMEM.

As shown in Figure 5, the translation technique improves the alignment quality significantly in terms of TP/FP ratios and ratios of the number of COGs detected. One disadvantage of tMEMs is that the number of anchors increases significantly due to the 6 frame translation. However, we will show, in Section 6, that the translation technique can use MEMs of longer length without sacrificing the alignment quality, thus making our algorithm even faster.

5 Algorithm for Genome Alignment by Match Extension

Our alignment algorithm finds maximal matches, filters them by an match extension without gaps, and chains remaining anchors to generate “local” genome alignments. The overview of our algorithm is given below:

1. Initial anchor detection stage: All MEMs or tMEMs anchors are detected using suffix array.
2. Anchor filtering stage: Each anchor is tested by the MEM filter. Any anchor that fails to extend beyond $T_e = 200$ with $T_{pi} = 0.5$ is discarded.
3. Anchor chaining stage: The remaining anchors are combined by computing locally optimal chains.
4. Chain tidying stage: Following a chain of anchors from left to right, each region between two adjacent anchors is aligned by Needleman-Wunsch algorithm [13]. If its alignment score

is below a preset threshold T_{nw}, then the connection between the two anchors is broken. Otherwise it is retained. All connected anchors including regions inbetween are reported as an alignment.

6 Experiments

We evaluated GAME in comparison with Blastz, MUMmer, and PatternHunter. All experiments were performed on a Pentium 2.0 GHz machine with 4GB memory running RedHat Linux 9.0. Two tests were performed, all pairwise comparisons of 10 bacterial genomes from close to distant in terms of phylogeny (*Mycoplasma genitalium, Mycoplasma pneumoniae, Bacillus halodurance, Bacillus subtilis, Staphylococcus aureus, Mycobacterium tuberculosis, Helicobacter pyroli, Escherichia coli, Haemophilus influenza,* and *Methanococcus jannaschii*; the shortest being of 0.56Mb and the longest being of 4.6Mb) and all pairwise comparisons of 5 chromosomes of *A. thaliana*. The performance in terms of the number of COG families detected and runtimes for 45 genome pair comparison are shown in Figure 6. The memory usage of GAME for all pairs ranged only from 25MB to 125MB (see [5] for the actual numbers). For discussion about the result, see the caption of Figure 6 and [5] for more detail.

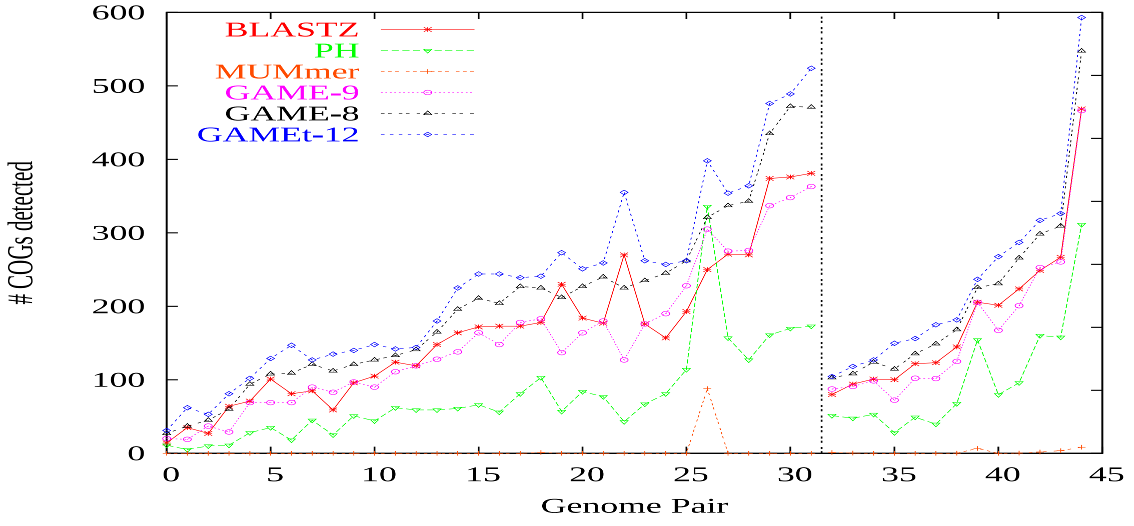
The *A. thaliana* data was used to test the scalability of the algorithms. We performed *all* pairwise comparisons of the five chromosomes, the shortest being 17.5 Mb and the longest being 29.6 Mb. The alignment quality was evaluated in terms of recent segmental duplication data [6]. With the MEM size of 10, $T_e = 200$ and $T_{pi} = 0.5$, GAME compared pairs in time ranging from 32 to 243 minutes with memory use ranging from 497MB to 747MB.[3] The alignment data and statistics with varying parameter settings are available at [5].

7 Conclusion

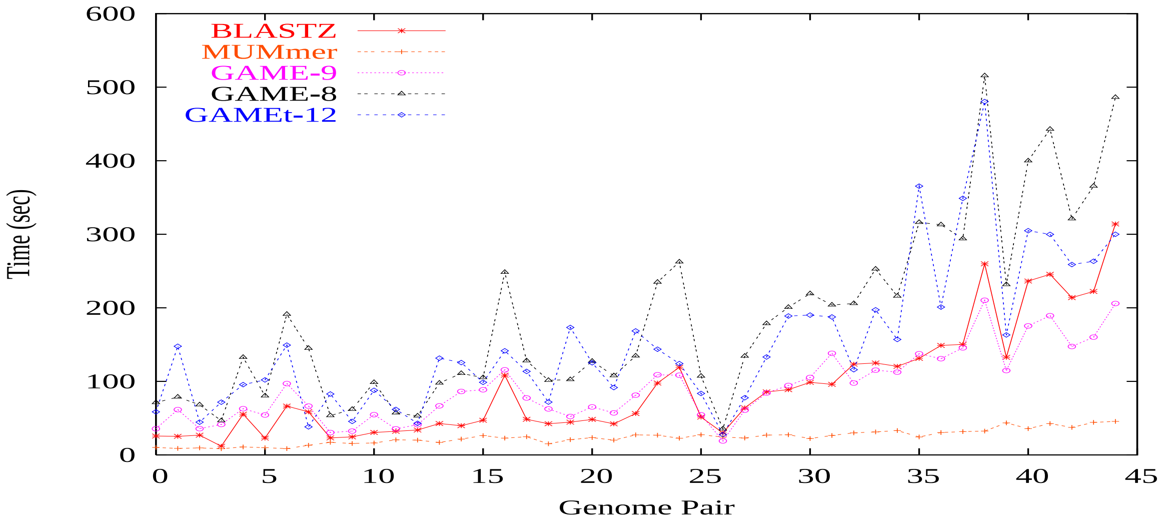
In this paper, we have proposed GAME, a scalable genome sequence alignment algorithm. The MEM based alignment algorithms, in contrast with pattern signature based ones, allow users to adjust the length of MEMs for genome pairs of different phylogenetic distance (Section 2.1). The MEM filtering scheme, the main contribution of our paper, can reliably eliminate noise (anchors in non-homologous regions) which exponentially increases for shorter MEMs. By introducing a fast, accurate filtering scheme and the translation technique for increased sensitivity of initial MEM anchors, GAME was able to demonstrate that the MEM based alignment algorithm can be improved to the level of or even better than sophisticated pattern signature based algorithms such as Blastz, PatternHunter and MUMmer. GAME is scalable enough to compare large genomes, e.g. *A. thaliana*, without compromising alignment quality.

Future work includes an adaptive genome alignment (including adaptive filtering scheme discussed in Section 3.2) and highly scalable genome alignment, e.g, aligning human genomes.

[3]Blastz and MUMmer were not successful to compare any of *A. thaliana* chromosome pairs due to the memory problem. PatternHunter reported its success in comparing *A. thaliana* chromosome 2 and 4, the shortest pair, in 13 minutes.



(a) #COGs detected



(b) Time

Figure 6: The number of COG families detected and runtimes for PatternHunter, Blastz, MUMmer with the MUM option, GAME-8, GAME-9 (MEMs of length 8, and 9 respectively), and GAMEt-12 (use of tMEM of the minimum length 12). The genome pairs are numbered in the increasing order of the number of COGs detected by GAMEt-12 to make the plots more readable. Blastz and GAME-9 were competitive in terms of runtimes and the number of COGs detected. GAME-8 outpeformed others (except GAMEt-12) in all cases except 2 genome pairs and GAMEt-12 outperformed others in all cases. The translation technique improved the performance of GAME significantly in terms of both alignment quality and runtimes; compare the results from GAMEt-12 and GAME-8. The sizes of possibly incorrectly aligned regions (outside matching COG families) for GAME remain almost same for varying length of MEM (see [5] for detail). The runtimes for PatternHunter were not included since it ran on a different machine under MS Windows. GAME achieved high quality genome alignments in reasonable times for all 45 genome pairs. The alignment quality consistently improved for shorter MEM length, say 8 and 9, thus our algorithm allows users to adjust alignment quality for different phylogenetic distance in trade of more computing time.

References

[1] S. Batzoglou, L. Pachter, J. P. Mesirov, B. Berger, and E. S. Lander. Human and mouse gene structure: Comparative analysis and application to exon prediction. *Genome Res.*, 10(7):950–958, 2000.

[2] A. L. Delcher, A. Phillippy, J. Carlton, and S. L. Salzberg. Fast algorithms for large-scale genome alignment and comparison. *Nucl. Acids. Res.*, 30(11):2478–2483, 2002.

[3] W. Ewens and G. Grant. *Statistical Methods in Bioinformatics.* Springer, Reading, MA, 2001.

[4] M. Höhl, S. Kurtz, and E. Ohlebusch. Efficient multiple genome alignment. *Bioinformatics*, 18(1):312S–320, 2002.

[5] http://bio.informatics.indiana.edu/projects/GAME/.

[6] http://www.psb.rug.ac.be/bioinformatics/simillion_pnas02/.

[7] M. Kellis, N. Patterson, M. Endrizzi, B. Birren, and E. Lander. Sequencing and comparison of Yeast species to identify genes and regulatory elements. *Nature*, 423:241–254, 2003.

[8] W. J. Kent and A. M. Zahler. Conservation, regulation, synteny, and introns in a large-scale *C. briggsae*-*C. elegans* genomic alignment. *Genome Res.*, 10(8):1115–1125, 2000.

[9] S. Kim and Y. Kim. A fast multiple string-pattern matching algorithm. In *Proc. of The 17th AoM/IAoM Interantional Conference on Computer Science*, pages 44–49, 1999.

[10] B. Ma, J. Tromp, and M. Li. PatternHunter: faster and more sensitive homology search. *Bioinformatics*, 18(3):440–445, 2002.

[11] U. Manber and G. Myers. Suffix arrays: A new method for on-line string searches. *SIAM J. Comput.*, 22(5):935–948, 1993.

[12] W. Miller. Comparison of genomic DNA sequences: solved and unsolved problems. *Bioinformatics*, 17(5):391–397, 2001.

[13] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48(3):443–53, Mar 1970.

[14] S. Schwartz, W. J. Kent, A. Smit, Z. Zhang, R. Baertsch, R. C. Hardison, D. Haussler, and W. Miller. Human-Mouse Alignments with BLASTZ. *Genome Res.*, 13(1):103–107, 2003.