

Gapped extension for local multiple alignment of interspersed DNA repeats

Todd J. Treangen^{1†}, Aaron E. Darling^{2†}, Mark A. Ragan², Xavier Messeguer¹

¹ Dept. of Computer Science, Polytechnic Univ. of Catalonia, Barcelona, Spain
treangen@lsi.upc.edu,

² ARC Centre of Excellence in Bioinformatics, and Institute for Molecular Bioscience, The
University of Queensland, Brisbane, Australia
a.darling@imb.uq.edu.au,

† These authors contributed equally to this work

Abstract. The identification of homologous DNA is a fundamental building block of comparative genomic and molecular evolution studies. To date, pairwise local sequence alignment methods have been the prevailing technique to identify homologous nucleotides. However, existing methods that identify and align all homologous nucleotides in one or more genomes have suffered poor scalability and limited accuracy. We propose a novel method that couples a gapped extension heuristic with a previously described efficient filtration method for local multiple alignment. During gapped extension, we use the MUSCLE implementation of progressive multiple alignment with iterative refinement. The resulting gapped extensions potentially contain alignments of unrelated sequence. We detect and remove such undesirable alignments using a hidden Markov model to predict the posterior probability of homology. The HMM emission frequencies for nucleotide substitutions can be derived from any strand/species-symmetric nucleotide substitution matrix, and we have developed a method to adapt an arbitrary substitution matrix (i.e. HOXD) to organisms with different G+C content. We evaluate the performance of our method and previous approaches on a hybrid dataset of real genomic DNA with simulated interspersed repeats. Our method outperforms existing methods in terms of sensitivity, positive predictive value, and localizing boundaries of homology. The described methods have been implemented in the free, open-source `procrastAligner` software, available from: <http://alggen.lsi.upc.es/recerca/align/procrastination>

1 Introduction

The importance of accurate homology identification to comparative genomics can not be overestimated[1]. To date, pairwise local sequence alignment methods such as BLAST [2–4] have been the prevailing technique to identify homologous nucleotides. When more than two copies of a homologous sequence element are present in the data, pairwise homology detection methods generate a listing of all possible pairs of homologous elements in the form of pairwise local alignments. Apart from the obvious inefficiency of considering all pairwise homology relationships, a collection of pairwise alignments is not ideal because they are rarely amenable to comparative genomic and phylogenetic analysis without further processing into a multiple alignment.

Local pairwise alignments can be merged to create a multiple alignment by a variety of methods[5–8]. Such methods commonly assume that pairwise homology relationships are transitive, such that if nucleotide a is homologous to nucleotide b , and b is to c , then a must also be homologous to c . Thus, in order to merge pairwise alignments, such methods must tackle the challenging problem of resolving inconsistent transitive homology relationships. Multiple alignment has been demonstrated to be more accurate than pairwise alignment, especially when dealing with a large number of divergent sequences[9, 10]. As the number of homologous sequences grows, we might expect that the number of inconsistent relationships in a collection of pairwise alignments would grow quadratically, whereas a direct multiple alignment method would provide an increasingly accurate alignment. Moreover, highly repetitive regions in the input sequences can cause serious efficiency problems for pairwise methods, as they create $O(r^2)$ pairwise alignments in the presence of a repeat with r copies. Mammalian Alu repeats and IS elements in microbes are two common examples of the overwhelming abundance of repetitive sequence in whole genomes.

Local multiple alignment has the inherent potential to avoid pitfalls associated with pairwise alignment. Although optimal multiple alignment under the SP objective function remains intractable[11], progressive alignment heuristics offer excellent speed and accuracy[12, 13] especially when combined with tree-independent iterative refinement[14], or probabilistic consistency measures[15]. Rather than merging pairwise alignments, why not exploit years of research into multiple alignment heuristics by directly constructing a multiple alignment? We thus present a novel heuristic for directly computing local multiple alignments via gapped extension of chained seed matches.

2 A heuristic for gapped extension of local multiple alignments

Our method for computing local multiple alignments exploits the MUSCLE multiple alignment algorithm to compute gapped extensions of ungapped multi-match seeds (see Figure 1). Gapped alignments arise when extending seeds to fully capture surrounding sequence homology. Our method assumes that a fixed number of nucleotides flanking a seed match are likely to be homologous and computes a global multiple alignment on the flanking region. Our assumption of flanking homology often proves to be erroneous and results in an alignment of unrelated sequences. In the context of *local* multiple alignment, the fundamental problem with such an approach is that current methods for progressive alignment with iterative refinement compute *global* alignments, i.e. they implicitly assume that input sequences are homologous over their entire length. To resolve the problem, we employ a hidden Markov model which detects unrelated regions embedded in the global multiple alignment. Unrelated regions are then removed from the alignment and the local multiple alignment is trimmed to reflect the updated boundaries of homology.

Our method, depicted for an example sequence in Figure 1, has seven primary steps: (1) identify multi-match seeds in the input sequence, (2) chain individual seeds, (3) multiply align of regions between chained seeds, (4) gapped extension of seed chains (5) detect unrelated regions using a hidden Markov model, (6) apply transitive homology relationships, and (7) removal of any unrelated sequence from the final local multiple

alignment. We have previously published Steps 1-2 of our method[16], while steps 3-7 represent a new contribution and are the subject of the present manuscript. Steps 2-7 are applied repeatedly to seeds identified in step 1 to produce local multiple alignments of all homologous nucleotides in the input sequence.

2.1 Chaining multi-match seeds from the input sequence

Given a sequence $\mathcal{S} = s_1, s_2, \dots, s_N$ of length N defined over an alphabet $\{A, C, G, T\}$, our method identifies local multiple alignments on homologous subsequences of \mathcal{S} . Our method first extracts candidate ungapped alignments, or multi-matches, among subsequences in \mathcal{S} , and we denote the set of all such matches as \mathbf{M} . To extract multi-matches from the input sequence, we utilize a palindromic spaced seed pattern[17], which is analyzed at each position in the input sequence. Palindromic spaced seeds offer good efficiency and reasonable sensitivity on a variety of input sequences[16]. We refer the number of matching regions in \mathcal{S} by a given match $M_i \in \mathbf{M}$ as the *multiplicity* of M_i , denoted as $|M_i|$. We refer to each matching region of M_i as a *component* of M_i . Our algorithm has an important limitation on the matches in \mathbf{M} : no two matches M_i and M_j may have the same left-end coordinate, except for the identity case when $i = j$. This constraint has been referred to by others as *consistency* and *transitivity*[18] of matches.

Once a list of multi-matches has been generated, we employ an efficient chaining and filtration algorithm to identify overlapping and nested chains of multi-matches[16]. In order to process each region of sequence $\mathcal{O}(1)$ times, matches are prioritized for chaining in order of decreasing multiplicity. The method chains multi-match seeds of the same multiplicity $|M_i|$ occurring within w characters of each other, thus gaps of up to size w are tolerated. When a multi-match can no longer be chained without including a gap larger than w characters, neighboring *subset* matches within w characters are identified. Each neighboring subset match is then *linked* to the chained match. We refer to the chained match as a *superset* match. Rather than immediately extend the subset match(es), we *procrastinate* and extend the subset match later when it has the highest multiplicity of any match waiting to be extended. When chaining a match with a linked superset, we immediately include the entire region covered by the linked superset match and thus eliminate the need to re-examine sequence already covered by a previously chained match.

2.2 Gapped extension of high-scoring chains

Our new method computes gapped extensions of the chained multi-match seeds. After chaining a multi-match M_i , we perform gapped alignment on all collinear regions located between two adjacent components to generate unextended local multiple alignments. We first evaluate the chain to decide whether expending computational resources on gapped extension will be worthwhile. We can optionally require that two or more seeds be present in the chain and use lower seed weights (k), a technique which has previously been proven successful[2, 19, 20]. To perform a gapped extension in each direction, we use MUSCLE to align dynamically-calculated window of nucleotides (l) to the left and right of the current local multiple alignment. Small values of l restrict

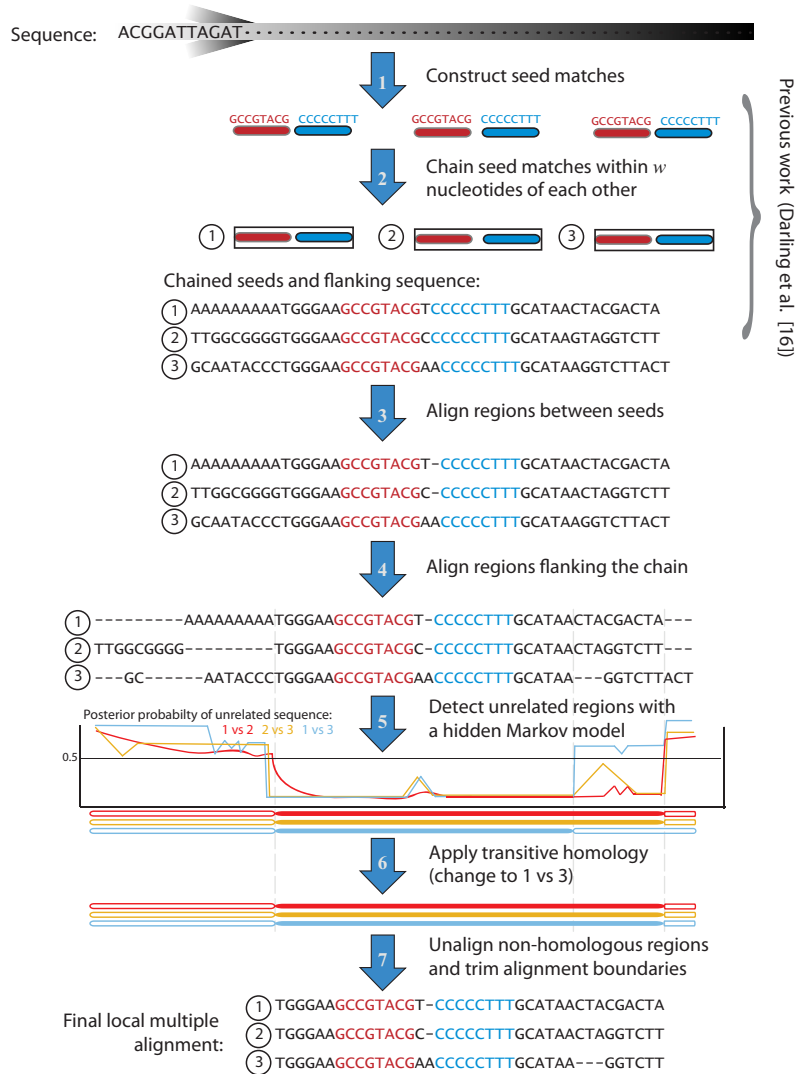


Fig. 1. Overview of the method, starting with an input sequence and ending with a set of local multiple alignments. First we (1) detect multi-matches in the input sequence(s) using palindromic spaced seeds, then we perform (2) chaining and extension of all multi-matches within w nt of each other. In the present example, one chain exists and contains two matches each with three match components labeled 1, 2, and 3. We then perform gapped alignment of the region between chained matches (3). In step (4), we perform a gapped extension by computing a global multiple alignment on the regions to the left and right of each chain component. The resulting alignment may contain unrelated sequence, so in step (5) we apply a hidden Markov model to detect poorly aligned regions indicative of unrelated sequence. Step (6) computes transitive homology relationships to ensure a consistent alignment and aid detection of divergent homologous sequences. Finally, in step (7) we unalign regions found to be non-homologous. If we find after step (2) that the alignment boundaries have been extended, we return to step (4) for another round of chaining.

respectively. Chiamonte *et al.* scaled the resulting $s(x, y)$ values by $\psi = 32.5421$ so the largest was 100, with the rest rounded to the nearest integer. The resulting emission probabilities for the Homologous and Unrelated states are given in Figure 2. HMM probabilities can be derived using any strand/species-symmetric nucleotide substitution matrix, but any particular matrix makes specific assumptions about divergence time, mutation pressures, and sequence composition of the aligned sequences. Genomes can range in G+C content from 30-75%, and at the extremes, a substitution matrix derived on 47.5% G+C sequence (such as HOXD) does not perform well. Previously it has been shown that adapting substitution matrices to the composition of the organisms under comparison can improve sequence alignment accuracy[22]. We have thus developed a method to adapt HMM emission frequencies derived from an arbitrary substitution matrix to organisms with different G+C content (see <http://algggen.lsi.upc.es/recerca/align/procrastination/> for details).

While emission frequencies for nucleotide substitutions can be derived from any strand/species-symmetric nucleotide substitution matrix, the gap-open and extend frequencies can not. To empirically estimate gap-open and extend values for the unrelated state we aligned a 10-kb, 48% G+C content region taken from *E. coli* CFT073 (Accession AF447814.1, coordinates 37,300-38,300) with an unrelated sequence. We generated an unrelated sequence with identical nucleotide composition by reversing the extracted sequence without complementation. We then forced an alignment with MUSCLE and counted the number of gap-open and gap-extend columns in the alignment of unrelated sequences. Gap-open and extend frequencies for the homologous state were estimated by constructing an alignment of 10kb of orthologous sequence shared among a pair of divergent organisms. We aligned the 48%G+C segment between genes *fruR* and *secA* from *E. coli* K12 (Accession U00096.21) and *Y. pestis* CO92 (Accession AL590842.1). We add the empirically derived gap-open and extend frequencies for each state and normalize the emission frequencies to a probability distribution. The resulting emission probabilities are reported in Figure 2.

Using the empirically derived transition and emission probabilities, we apply the posterior HMM decoder implemented in the HMMoC software[23] to compute the posterior probability (p.p.) that each alignment column represents homologous sequence. Columns with a p.p. below 0.5 are considered to be unrelated; use of 0.5 as a p.p. threshold yields a maximum *a posteriori* estimate of homology. We then apply the transitive homology principle to our predictions, resulting in a final set of consistent homology predictions. See Figure 1, steps 5 and 6 for an example. We trim the alignment to exclude all columns beyond the Homologous state. If the original boundaries were improved, we trigger another round of chaining (and consequently another round of extension) in the same direction. When gapped extension fails to improve boundaries in one direction, extension in the other direction is attempted until no further extension is possible.

3 Results

We have previously demonstrated the sensitivity of our chaining method in finding Alu repeats in the human genome[16]. Figure 6 shows part of a local multiple alignment of

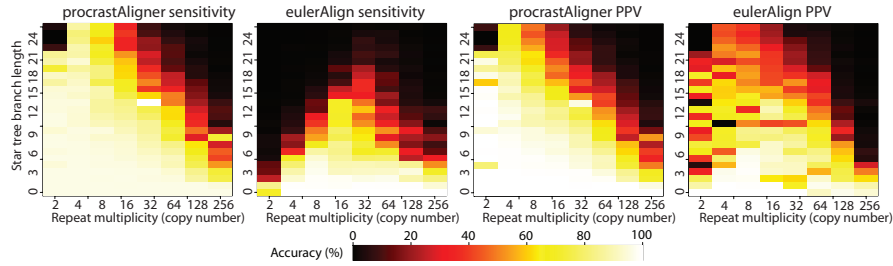


Fig. 3. Accuracy recovering simulated repeat families planted in the *Mycoplasma genitalium* genome. Sum-of-pairs (SP) nucleotide sensitivity and positive predictive value (PPV) of `procrastAligner` and `eulerAlign` were measured for 200 combinations of branch length and multiplicity. Three replicates of each simulation were performed and average accuracy values are shown here. White points indicate perfect alignment of the simulated repeat family. Black points indicate the program completely failed to recover any portion of the repeat family. Average mutations per site can be calculated by multiplying branch length by the fixed substitution rate of 0.09, and indel rate of 0.01. For example, at branch length 20 there are 1.8 substitutions per site and 0.2 indels per site. From the figure, it is apparent that `procrastAligner` performs better at higher mutation rates and multiplicities than `eulerAlign`.

one such Alu family as generated with `procrastAligner`. To highlight the benefits of our proposed heuristic for gapped extension, we compare `procrastAligner`'s performance to the Eulerian path method for local multiple alignment as implemented by `eulerAlign`[8]. The Eulerian path method uses a *de Bruijn* graph for filtration, and goes beyond filtration to compute gapped alignments using banded dynamic programming. To our knowledge, `procrastAligner` and `eulerAlign` represent the only two automated methods to construct local multiple alignments directly from genomic DNA.

3.1 Simulating interspersed repeats

We evaluate accuracy of each method when aligning simulated repeat families that have been inserted into the complete genome of *Mycoplasma genitalium*. The *M. genitalium* genome has been recognized as complex and repeat-rich[24], presenting a biologically relevant and challenging example to evaluate alignment methods. We simulated repeat families of 8 different multiplicities ranging between 2 and 256 (x -axis in Figure 3). Each repeat copy has an average length based on its family's multiplicity ($length = \frac{7680}{multiplicity}$), thus high copy-number repeats are short. Evolution of repeat families was simulated as a marked Poisson process on a star tree topology. The branch lengths were varied between 0 and 24 (y -axis in Figure 3), with the nucleotide substitution rate fixed at 0.09 per unit time, and the indel rate fixed at 0.01 per unit time. Rate heterogeneity among sites was modeled with a gamma distribution ($\theta = 1.0, k = 0.5$). Indel size was Poisson distributed with intensity 3, and insertions and deletions were taken to be equally likely. Each family's ancestral sequence was randomly generated using nucleotide frequencies equal to the composition of *Mycoplasma genitalium* ($A = 0.34, T = 0.34, G = 0.16, C = 0.16$). Insertion sites for repeat copies were chosen

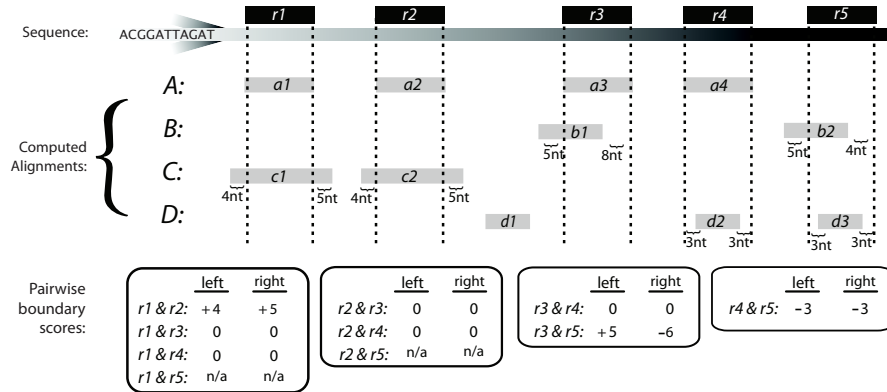


Fig. 4. Pairwise boundary accuracy metric. We define our boundary accuracy metric to be an all-pairs score by comparing the boundaries of all r components of the inserted repeat R to the boundaries predicted by the alignment program. For any pair of components, r_i and r_j , we take the maximum boundary of any local multiple alignments output by the program. The figure shows a multiplicity five interspersed repeat R and four local multiple alignments, A, B, C, D . Boundary predictions can be classified as (1) correct, (2) overprediction, and (3) underprediction, with each discussed in turn: (1) *Correct prediction*. Consider scoring components $r1$ and $r3$. Local multiple alignment A overlaps both $r1$ and $r3$ and no other alignment overlaps both $r1$ and $r3$. The left and right boundaries of alignment A match the boundaries of $r1$ & $r3$ exactly, thus we assign scores of 0 for $r1$ & $r3$. (2) *Overprediction*. Consider scoring components $r1$ and $r2$. These components are overlapped by alignments A and C . Alignment A has perfect boundary predictions for $r1$ & $r2$, while alignment C extends beyond the true boundaries of components $r1$ and $r2$ by 4 nucleotides on the left and 5 nucleotides on the right. Our scoring metric always uses the maximum predicted boundaries for a pair of components, thus the boundary predictions from C are reported for $r1$ & $r2$. (3) *Underprediction*. Consider scoring components $r3$ and $r5$. Alignment B hits both $r3$ and $r5$, but stops short of the right-side boundary by 8nt in $r3$ and 4nt in $r5$. We average the error and record -6 for the right-side of $r3$ & $r5$. Finally, component pairs that are not contained by any computed alignments are not scored, as indicated by n/a .

uniformly at random in the 580kb *M. genitalium* genome, allowing tandem repeats but prohibiting mosaic repeats.

3.2 Alignment accuracy metrics

We used each program to find local multiple alignments in each of the 200 modified *M. genitalium* genomes and recorded alignment accuracy as follows. We calculated Sum-of-Pairs (SP) nucleotide sensitivity as $\frac{TP}{TP+FN}$, where TP is the number of aligned nucleotide pairs in the program's output which are also aligned in the simulated repeat family. FN is the number of aligned nucleotide pairs in the simulated repeat family which are missing from the program's output. This sensitivity measure is identical to the Sum-of-Pairs (SP) accuracy defined by BaliBASE[25]. We calculate the positive predictive value (PPV) as $\frac{TP}{TP+FP}$, where TP is defined as above, and FP is the total number of nucleotide pairs from the program's output where one of the nucleotides are part of the simulated repeat family and the other nucleotide was incorrectly aligned. We also quantify the ability of each aligner to accurately predict the boundaries of the interspersed repeats. For a given pair of repeat components, we calculate accuracy by recording the number of nucleotides between the true boundary and the predicted boundary on both the right and left sides of the repeat. Thus, over-extension gets a positive score, while underextension yields a negative score and perfect boundaries receive a 0 score. See Figure 4 for further details on boundary under/overpredictions.

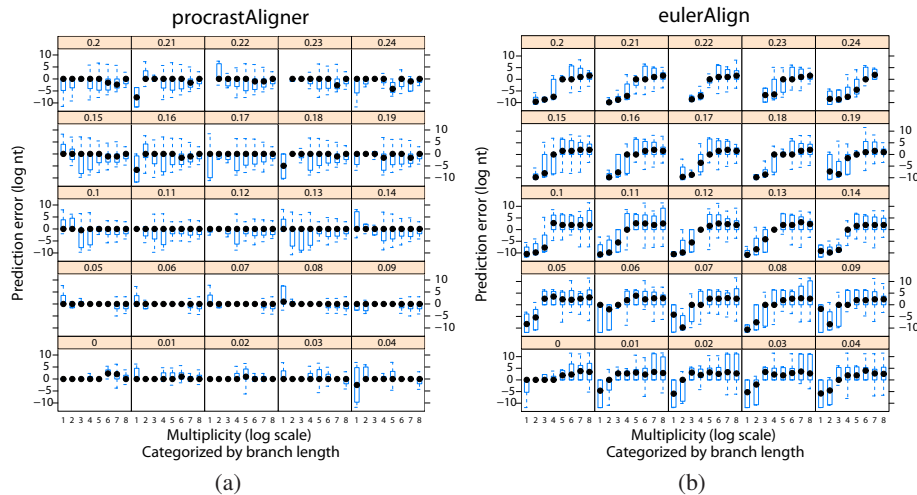


Fig. 5. Boundary prediction performance. All-pairs boundary prediction accuracy of `procrastAligner` and `eulerAlign` were measured for 200 combinations of branch length and multiplicity. Accuracy on each combination is presented as a box-and-whiskers plot using the scoring metric detailed in Section 3.2. Branch lengths range from 0 to 0.24 and increase by intervals of 0.01. The x -axis label represents the multiplicity of the interspersed repeat in \log_2 -scale. i.e. axis label 8 indicates $2^8 = \text{multiplicity } 256$. The y -axis label is the prediction error in \log_2 -scale nucleotides. Values at 0 represent correctly identified repeat boundaries, values greater than 0 represent overpredictions, and values less than 0 represent underpredictions (see Figure 4). In general, `procrastAligner` identifies the true interspersed repeat boundaries more accurately than `eulerAlign`.

3.3 Accuracy when aligning interspersed repeats

We applied `procrastAligner` and `eulerAlign` to the hybrid simulated/real dataset. We ran `procrastAligner` with command-line parameters `--z=15 --w=20`, and `eulerAlign` with `-k 15 -l -i 1000 -v` based on suggestions from the program’s user guide and manual experimentation. Simulations for each of the 200 combinations of branch length and multiplicity were replicated three times and alignments generated in parallel on a 156-node compute cluster. Results of the experiments are reported in Figure 3 and Figure 5. Figure 3 illustrates the sensitivity and PPV of `procrastAligner` and `eulerAlign` on datasets ranging from 0 substitutions and indels per site to 2.16 substitutions and 0.24 indels per site (branch length 24). As mutation rates and repeat multiplicity increase the alignment accuracy decreases for both methods, with accuracy of `eulerAlign` decreasing faster than `procrastAligner`. Surprisingly, `eulerAlign` often fails to align low multiplicity repeats, even when mutation rates are low. Manual experimentation with `eulerAlign` parameters, such as: `-v` (tolerance for mismatches), `-k` (seed k -mer size) from 11 to 15, and `-i` (number of iterations) from 1000 up to 10,000, failed to improve its performance on low-multiplicity repeats. We conjecture that `procrastAligner`’s overall improved accuracy largely derives from its use of spaced seed patterns[16] and tolerance of gaps.

With the `-v` option enabled, the Eulerian path method allows up to 10% mismatches for matching k -mers to seed gapped alignment extensions. While this certainly improves the sensitivity at lower mutation rates, the experimental results presented Figure 3 show that it is inadequate for higher mutation rates. In addition to sensitivity and PPV benchmarks, we also assess how well each aligner recovers the true boundaries of interspersed repeats. Figure 5 illustrates the ability of each program to accurately localize the known boundaries of the simulated interspersed repeats. From the figure, it is apparent that on average, `procrastAligner` predicts the exact repeat boundary for all studied combinations of branch length (repeat degeneracy) and multiplicity (repeat copy number). Moreover, the standard error in `procrastAligner`'s boundary predictions is typically very low, within 4 nucleotides. `eulerAlign`, on the other hand, exhibits more erratic behavior. For low multiplicity repeats it has a strong tendency to underpredict the repeat boundary. At high multiplicities (≥ 32) `eulerAlign` tends to slightly overpredict the boundaries, by including flanking unrelated sequence in the alignment of the interspersed repeat.

Finally, a direct comparison of run time is difficult, due to the differing natures of the local multiple alignment programs. `eulerAlign` runtime depends on the iterations parameter, which controls the total number of local multiple alignments reported, whereas `procrastAligner` reports *all* local multiple alignments in a single run. Despite this, we report the average per-experiment CPU time for each program on the test dataset. `procrastAligner` required on average 55 seconds per experiment, with the longest taking just over two minutes. `eulerAlign` required 1 hour total compute time per experiment on average, which equates to about 4 seconds per iteration. Both programs exhibited similar memory usage, `procrastAligner` requiring on average 50 MB per experiment and `eulerAlign` requiring on average 70 MB per experiment.

4 Discussion

We have presented a sensitive and efficient gapped-extension heuristic for local multiple alignment. We have extended our previous results by converting chains of ungapped multi-matches into gapped local multiple alignments. Our method is based around an efficient heuristic for local multiple alignment, featuring a novel method for gapped extensions joining global multiple alignment with a homology test based on a hidden Markov model. Experimental results demonstrate that the described method offers a level of alignment accuracy exceeding that of previous methods. Accurately predicting homology boundaries has important implications; for example, tools to build repeat family databases can directly use the alignments without the manual curation required in current approaches and also is likely to aid in the evolutionary analysis of transposon proliferation. Further improvement of the alignment methodology will likely require increasingly sensitive methods for seed matching in conjunction with a statistical methodology to assign significance to local multiple alignments. One possible avenue to increase seed matching sensitivity and reduce boundary underpredictions would be merging overlapping seed matches into a shorter, higher-multiplicity match. A second avenue would be use of palindromic seed families instead of using a single seed pattern. With increased seed matching sensitivity comes additional false positive seed hits,

```

0   5   10  15  20  25  30  35  40  45  50  55  60  65  70  75  80  85  90  95  100 105 110
18331 GGAGTTTCAGACCACTGGGGCAACAGGCAAAAC-CCCGTCTTACT--AAAAT--ACAATAAGCTAGCCAGGCGTGGTGGTGCATGTAAACCCAGTACTGGGGA
34336 GGGTTTCAGACCACTGGGGCAACAGGCAAAAC-CCCGTCTTACT--AAAAT--AC-AAAAATAGCTGGGCGTAAAGGCGCACGCCGTAACTCCTAGTACTCGGGG
35889 GGAGTTTCAGACCACTGGGGCAACAGGCAAAAC-CCCGTCTTACT--AAAAT--AC-AAAAATAGCTGGGCGTAAAGGCGCACGCCGTAACTCCTAGTACTCGGGG
40716 GGAGTTTCAGACCACTGGGGCAACAGGCAAAAC-CCCGTCTTACT--AAAAT--AC-AAAAATAGCTGGGCGTAAAGGCGCACGCCGTAACTCCTAGTACTCGGGG
48429 GGAGTTTCAGACCACTGGGGCAACAGGCAAAAC-CCCGTCTTACT--AAAAT--AC-AAAAATAGCTGGGCGTAAAGGCGCACGCCGTAACTCCTAGTACTCGGGG
50769 GGAGTTTCAGACCACTGGGGCAACAGGCAAAAC-CCCGTCTTACT--AAAAT--AC-AAAAATAGCTGGGCGTAAAGGCGCACGCCGTAACTCCTAGTACTCGGGG
77269 GGAGTTTCAGACCACTGGGGCAACAGGCAAAAC-CCCGTCTTACT--AAAAT--AC-AAAAATAGCTGGGCGTAAAGGCGCACGCCGTAACTCCTAGTACTCGGGG
92928 GGAGTTTCAGACCACTGGGGCAACAGGCAAAAC-CCCGTCTTACT--AAAAT--AC-AAAAATAGCTGGGCGTAAAGGCGCACGCCGTAACTCCTAGTACTCGGGG
105233 GGAATTCAGACCACTGGGGCAACAGGCAAAAC-CCCGTCTTACT--AAAAT--AC-AAAAATAGCTGGGCGTAAAGGCGCACGCCGTAACTCCTAGTACTCGGGG
117611 GGAGTTTCAGACCACTGGGGCAACAGGCAAAAC-CCCGTCTTACT--AAAAT--AC-AAAAATAGCTGGGCGTAAAGGCGCACGCCGTAACTCCTAGTACTCGGGG
121584 GGAGTTTCAGACCACTGGGGCAACAGGCAAAAC-CCCGTCTTACT--AAAAT--AC-AAAAATAGCTGGGCGTAAAGGCGCACGCCGTAACTCCTAGTACTCGGGG
131094 GGAGTTTCAGACCACTGGGGCAACAGGCAAAAC-CCCGTCTTACT--AAAAT--AC-AAAAATAGCTGGGCGTAAAGGCGCACGCCGTAACTCCTAGTACTCGGGG
160730 AGAGTTTCAGACCACTGGGGCAACAGGCAAAAC-CCCGTCTTACT--AAAAT--AC-AAAAATAGCTGGGCGTAAAGGCGCACGCCGTAACTCCTAGTACTCGGGG
168253 GGAGTTTCAGACCACTGGGGCAACAGGCAAAAC-CCCGTCTTACT--AAAAT--AC-AAAAATAGCTGGGCGTAAAGGCGCACGCCGTAACTCCTAGTACTCGGGG
174332 GTAGTTTCAGACCACTGGGGCAACAGGCAAAAC-CCCGTCTTACT--AAAAT--AC-AAAAATAGCTGGGCGTAAAGGCGCACGCCGTAACTCCTAGTACTCGGGG
197103 AGAGTTTCAGACCACTGGGGCAACAGGCAAAAC-CCCGTCTTACT--AAAAT--AC-AAAAATAGCTGGGCGTAAAGGCGCACGCCGTAACTCCTAGTACTCGGGG

```

Fig. 6. Alu repeat alignment. Partial view of an Alu repeat alignment output by *procrastAligner* in the *H. sapiens* BAC clone RP11-355H10 (Accession AC010145.10). Each row represents an aligned Alu. Highlighted columns indicate conserved sequence among all 16 copies of the Alu. Start positions are shown to the left, negative values indicate complement strand. Local multiple alignment was generated with *procrastAligner* with parameters: `--z=9 --w=50`.

so a statistical test for rejecting insignificant local alignments will likely be required. Unfortunately exact computation of *p*-values for local multiple alignments remains a daunting challenge, although fast approximation methods for pairwise alignments have shown promise[26] and potentially can be extended to multiple alignments[8, 27].

4.1 Implementation

We have implemented our method in a program, *procrastAligner*, available for Linux, Windows, and Mac OS X. Our open-source implementation is available as C++ source code licensed under the GPL, and can be downloaded from: <http://alggen.lsi.upc.es/recerca/align/procrastination>.

5 Acknowledgments

The authors would like to thank Yu Zhang for providing the *eulerAlign* program. We are grateful to Guillaume Achaz for helpful discussions on the gapped extension algorithm. Accuracy evaluations utilized a compute resource grant from the Australian Partnership for Advanced Computing. AED was supported by NSF grant DBI-0630765. TJT was supported by Spanish Ministry MECD Grant TIN2004-03382 and AGAUR Training Grant FI-IQUC-2005.

References

1. Kumar, S., Filipski, A.: Multiple sequence alignment: In pursuit of homologous DNA positions. *Genome Res.* **17** (2007) 127–135
2. Schwartz, S., Kent, J.W., Smit, A., Zhang, Z., Baertsch, R., Hardison, R.C., Haussler, D., Miller, W.: Human-mouse alignments with blastz. *Genome Res.* **13** (2003) 103–107
3. Pearson, W.R.: Rapid and sensitive sequence comparison with FASTP and FASTA. *Methods Enzymol* **183** (1990) 63–98
4. Ma, B., Tromp, J., Li, M.: PatternHunter: faster and more sensitive homology search. *Bioinformatics* **18** (2002) 440–445

5. Blanchette, M., Kent, W., Riemer, C., Elnitski, L., Smit, A.F., Roskin, K.M., Baertsch, R., Rosenbloom, K., Clawson, H., Green, E., Haussler, D., Miller, W.: Aligning multiple genomic sequences with the threaded blockset aligner. *Genome Res.* **14** (2004) 708–715
6. Raphael, B., Zhi, D., Tang, H., Pevzner, P.: A novel method for multiple alignment of sequences with repeated and shuffled elements. *Genome Res* **14(11)** (2004) 2336–46.
7. Morgenstern, B., French, K., Dress, A., Werner, T.: DIALIGN: finding local similarities by multiple sequence alignment. *Bioinformatics* **14** (1998) 290–294
8. Zhang, Y., Waterman, M.S.: An Eulerian path approach to local multiple alignment for DNA sequences. *PNAS* **102** (2005) 1285–90.
9. Brudno, M., Do, C.B., Cooper, G.M., Kim, M.F., Davydov, E., Program, N.C.S., Green, E.D., Sidow, A., Batzoglou, S.: LAGAN and Multi-LAGAN: efficient tools for large-scale multiple alignment of genomic dna. *Genome Res.* **13** (2003) 721–731
10. Szklarczyk, R., Heringa, J.: Aubergene—a sensitive genome alignment tool. *Bioinformatics* **22** (2006) 1431–1436
11. Wang, L., Jiang, T.: On the complexity of multiple sequence alignment. *J Comput Biol* **1** (1994) 337–348
12. Thompson, J.D., Higgins, D.G., Gibson, T.: CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. *Nucleic Acids Res.* **22** (1994) 4673–80
13. Notredame, C., Higgins, D.G., Heringa, J.: T-Coffee: A novel method for fast and accurate multiple sequence alignment. *J Mol Biol* **302** (2000) 205–217
14. Edgar, R.: MUSCLE: Multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.* **32** (2004)
15. Do, C.B., Mahabhashyam, M.S., Brudno, M., Batzoglou, S.: ProbCons: Probabilistic consistency-based multiple sequence alignment. *Genome Res* **15** (2005) 330–340
16. Darling, A.E., Treangen, T.J., Zhang, L., Kuiken, C., Messeguer, X., Perna, N.T.: Procrastination leads to efficient filtration for local multiple alignment. *Algorithms in Bioinformatics* **4175** (2006) 126–137
17. Choi, P. K., Zeng, F., Zhang, L.: Good spaced seeds for homology search. *Bioinformatics* **20** (2004) 1053–1059
18. Szklarczyk, R., Heringa, J.: Tracking repeats using significance and transitivity. *Bioinformatics* **20 Suppl 1** (2004) I311–I317
19. Altschul, S.F., Madden, T.L., Schffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res* **25** (1997) 3389–3402
20. Kent, W.J.: BLAT—the BLAST-like alignment tool. *Genome Res* **12** (2002) 656–664
21. Chiaromonte, F., Yap, V.B., Miller, W.: Scoring pairwise genomic sequence alignments. *Pac Symp Biocomput* (2002) 115–126
22. Yi-Kuo, Y., Altschul, F.: The construction of amino acid substitution matrices for the comparison of proteins with non-standard compositions. *Bioinformatics* **21** (2005) 902–911
23. Lunter, G.: HMMoC a compiler for hidden Markov models. *Bioinformatics* **23** (2007) 2485–2487
24. Rocha, E.P., Blanchard, A.: Genomic repeats, genome plasticity and the dynamics of *Mycoplasma* evolution. *Nucleic Acids Res* **30** (2002) 2031–2042
25. Thompson, J.D., Plewniak, F., Poch, O.: A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Res* **27** (1999) 2682–90
26. Achaz, G., Boyer, F., Rocha, E.P.C., Viari, A., Coissac, E.: Repseek, a tool to retrieve approximate repeats from large dna sequences. *Bioinformatics* (2006)
27. Prakash, A., Tompa, M.: Statistics of local multiple alignments. *Bioinformatics* **21 Suppl 1** (2005)