Tandem Cyclic Alignment

Gary Benson[⋆]

Department of Biomathematical Sciences
The Mount Sinai School of Medicine
New York, NY 10029-6574
benson@ecology.biomath.mssm.edu

Abstract. We present a solution for the following problem. Given two sequences $X = x_1x_2 \cdots x_n$ and $Y = y_1y_2 \cdots y_m$, $n \leq m$, find the best scoring alignment of $X' = X^k[i]$ vs Y over all possible pairs (k, i), for $k = 1, 2, \ldots$ and $1 \leq i \leq n$, where $X[i]$ is the cyclic permutation of X, $X^k[i]$ is the concatenation of k complete copies of $X[i]$ (k tandem copies), and the alignment must include all of Y and all of X'. Our algorithm allows any alignment scoring scheme with *additive* gap costs and runs in time $O(nm \log n)$. We have used it to identify related tandem repeats in the *C. elegans* genome as part of the development of a multi-genome database of tandem repeats.

1 Introduction

1.1 Problem Description

The problem we solve is the following:

Tandem Cyclic Alignment

Given: Two sequences $X = x_1x_2 \cdots x_n$ and $Y = y_1y_2 \cdots y_m$, $n \leq m$ and an alignment scoring scheme with *additive* gap costs.

Find: The best scoring alignment of $X' = X^k[i]$ vs Y over all possible pairs (k, i), for $k = 1, 2, \ldots$ and $1 \leq i \leq n$, where $X[i]$ is the cyclic permutation of X,

$$X[i] = x_i x_{i+1} \cdots x_n x_1 \cdots x_{i-1},$$

$X^k[i]$ is the concatenation of k complete copies of $X[i]$ (k tandem copies), and the alignment must include all of Y and all of X'.

Let X and Y be two strings over an alphabet Σ. An alignment of X and Y (see section 3.1 for an example) is a pair of equal length sequences $\hat{X}$, $\hat{Y}$ over the

[⋆] Partially supported by NSF grants CCR-9623532 and CCR-0073081.

alphabet $\Sigma \cup \{-\}$ where $-$ is a *gap* character and X, Y are obtained from $\hat{X}$, $\hat{Y}$ by removing the gap characters. An alignment can be interpreted as a sequence Q of *edit operations* [6] that transform X into Y. The allowed operations are 1) insert a symbol into X, 2) delete a symbol in X and 3) replace a symbol in X with a (possibly identical) symbol from Σ. A scoring scheme defines a weight for each possible operation and the alignment score is the sum of the weights assigned to the operations in Q.

There are two widely used classes of scoring schemes, 1) *distance scoring*, in which identical replacement has weight $= 0$, all other operations have weight ≥ 0 and the best alignment has minimum score, and 2) *similarity scoring*, in which "good" replacements have weight > 0, all other operations have weight ≤ 0 and the best alignment has maximum score. Within these classes, scoring schemes are further characterized by the treatment of gap costs. A *gap* is the result of the deletion of one or more consecutive characters in one of the sequences (insertion into the other sequence). *Additive* gap costs assign a constant weight to each of the consecutive characters. Other gap functions have been found useful for biological sequences, including affine gap costs ($\alpha + \beta k$ for a gap of k consecutive characters where α and β are constants) and concave gap costs ($\alpha + \beta f(k)$ where $f()$ is a concave function such as square root). The solution in this paper assumes a scoring scheme with *additive* gap costs. For ease of discussion, we will, for the remainder of the paper, assume distance scoring although the results apply as well to similarity scoring.

Our motivation for this problem arises from an ongoing effort to construct a multi-genome database of tandem repeats (TRDB). A central task is the clustering of tandem repeats into families *i.e.* repeats that occur in different locations in a genome but have identical or very similar underlying patterns. Grouping these repeats will facilitate identification and study of their common properties. Tandem repeat families have been detected in both prokaryotes and eukaryotes, including the *E. coli*, *S. cerevisiae*, *C. elegans* and human genomes.

Clustering requires an effective and consistent means of measuring the similarity or distance between repeats. Standard comparison methods are not easily applied to tandem repeats because they contain *repetitive, approximate copies* of an underlying pattern. In addition, comparison of related repeats often reveals a scrambling of the left to right order of the slightly different internal copies. An accurate comparison method should be insensitive to copy number and copy order and we have therefore chosen to abstract the repeats as either 1) consensus patterns or 2) profiles and then compare them using alignment.

Because repeat copies are adjacent, the *designation of first position* in a consensus or profile *is arbitrary.* This is not just a theoretical abstraction, the number of copies in a repeat is often not a whole number and distinct repeats which are obviously similar often do not start and end at the same relative positions. Therefore, comparison must allow cyclic permutation of one pattern so that its first position can be arbitrarily aligned with any position in the other.

Once families are constructed, we can determine interfamily evolutionary relationships by comparing patterns from different families. In particular, we can determine if one pattern consists of multiple approximate copies of the other, again with the property of cyclic permutation. It is this comparison that Tandem Cyclic Alignment addresses.

1.2 Background

The Tandem Cyclic Alignment problem is a merger of two classes of pairwise alignment problems, 1) *tandem alignment*, in which one of the sequences consists of an indeterminate number of tandem copies of a pattern and 2) *cyclic alignment*, in which cyclic permutation of one of the patterns is allowed. Three related problems from these classes are:

Pattern local, text global tandem alignment. Given a pattern X, a text Y and a scoring scheme for alignment, find the best scoring alignment of $X' = X^k[1]$ vs Y over all $k = 1, 2, \ldots$, where all of Y must occur in the alignment, but where the part of X' aligned with Y need not contain a whole number of copies of $X[1]$. The alignment, rather, may start and end on any index of X.

Pattern and text global tandem alignment. Given a pattern X, a text Y, an index i, $1 \leq i \leq |X|$, and a scoring scheme for alignment, find the best scoring alignment of $X' = X^k[i]$ vs Y, over all $k = 1, 2, \ldots$, where all of Y and all of X' must occur in the alignment.

Cyclic global alignment. Given sequences X and Y and a scoring scheme for alignment, find the best scoring alignment of $X[i]$ vs Y over all possible i, $1 \leq i \leq |X|$ where all of Y and exactly one whole (cyclically permuted) copy of X must occur in the alignment.

The tandem alignment problems are both solved by wraparound dynamic programming (WDP) [8, 3] in $O(mn)$ time when the scoring function has additive or affine gap costs. The cyclic alignment problem can be solved naively in $O(n^2m)$ time by separately computing the alignment of $X[i]$ vs Y for every value of i. Maes [7] presented a $O(nm \log n)$ time solution for scoring schemes with additive gap costs by observing that there exists a set of best scoring alignments, one for each $1 \leq i \leq n$ such that the alignments are pairwise *non-crossing* (below). Landau, Myers and Schmidt [5] gave a $O(n+km)$ algorithm for unit cost differences (edit distance) when the score of the best alignment is bounded by k. Their algorithm, although theoretically efficient, has a large constant factor and is difficult to implement because it requires constructing a suffix tree preprocessed for least common ancestor queries. Schmidt [9] gave a rather complicated $O(nm)$ algorithm for similarity scoring where each insertion/deletion character costs $-s$ and match/mismatch weights are in the interval $[-s, m]$ for fixed positive integer values m and s. This method can not be used to compute general distance scores more efficiently than the Maes algorithm.

It seems natural to adapt the Maes solution to our problem, except for one difficulty: in tandem cyclic alignment, there may be no set of best scoring alignments which are all pairwise non-crossing. What this means is that the number of copies of X used in an alignment can vary depending on the starting position i. (For an example see Section 3.1). We show, though, that no alignment can cross the "same" alignment more than once. This leads to a $O(mn \log n)$ time solution using adaptations of the Maes algorithm and WDP.

The remainder of the paper is organized as follows. In section 2 we give brief descriptions of the non-crossing alignments property, the Maes algorithm and wraparound dynamic programming. In section 3 we give the main theorem about crossing tandem cyclic alignments. In section 4 we then apply this property to obtain our algorithm. Finally, in section 5 we show an example from our analysis applied to tandem repeats from the *C. elegans* genome.

2 Preliminaries

2.1 Non-crossing alignments

When gap costs are additive, a simple non-crossing property of optimal paths in the two dimensional alignment matrix applies [4, 1, 7]. We present one variation appropriate for this paper.

Definition. Two paths A and B in an alignment matrix *cross* if there exist two rows e and f such that in row e all matrix cells in path A are left of all cells in path B and in row f all cells in path B are left of all cells in path A. The paths share one or more common cells where they cross.

Note that *sharing cells* is not the same as crossing.

Property: Given an alignment matrix (see figure 1, left) and four cells q, r, s and t with q left of r in the top row and s left of t in the bottom row, for any optimal scoring path A from q to s, there exists an optimal scoring path B from r to t such that the two paths do not cross.

Proof. By contradiction, suppose all optimal scoring paths from r to t cross A. Let B be one such path. A and B must cross an even number of times. Consider the separate subpaths (labeled A_2 and B_2 in the figure) in which A is first right of B proceeding from the top row.

Claim 1: cost of B_2 is equal or worse than cost of A_2. Otherwise A is not optimal because joining subpaths A_1, B_2 and A_3 is better.

Claim 2: cost of B_2 is better than cost of A_2. Otherwise, joining subpaths B_1, A_2 and B_3 gives a path with score no worse than B, but which does not cross A. Such a path was assumed not to exist.

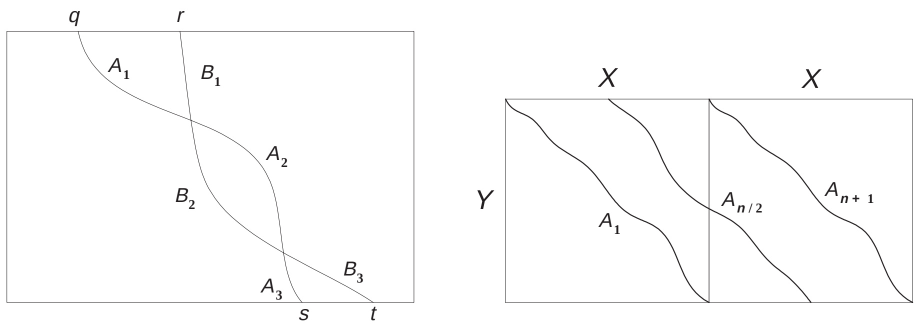

Fig. 1. (left) Alignments do not cross; (right) the Maes algorithm.

Clearly Claims 1 and 2 lead to a contradiction.

2.2 The Maes algorithm for cyclic global alignment

The Maes algorithm [7] capitalizes on the non-crossing property to *bound* the area of the alignment matrix that must be computed for each index i in the alignment of $X[i]$ vs Y.

First the alignment of $X[1]$ vs Y (call it A_1) is computed in $O(nm)$ time. A new matrix is then constructed which uses two concatenated copies of X vs Y (figure 1, right). The alignment A_1 shifted right (call it A_{n+1}) optimally aligns the second copy of X with Y.

A_1 and A_{n+1} bound any alignments which start and end between them. Specifically, they bound the alignment of $X[n/2]$ vs Y (call it $A_{n/2}$). It is easy to see that this procedure can be followed recursively, for a logarithmic number of steps, subdividing X into halves, then fourths, etc. always at the midpoints between bounding alignments. In each step, the alignment score calculations in a matrix cell are computed once, except for matrix cells on a bounding path, where they are computed twice (once for the computation in the interval to the left and once to the right), yielding $O(nm \log n)$ as the overall time of the algorithm.

2.3 Wraparound dynamic programming (WDP)

WDP [8, 3] models the similarity computation of Y with an unrestricted number of copies of X while using an alignment matrix of size nm rather than of size m^2, *i.e.* using only one copy of X. WDP computes in matrix $S[i, j]$ the optimal score that would be obtained by aligning $Y_1 \cdots Y_i$ with $X^* X_1 \cdots X_j$, where X^* indicates zero or more tandem copies of X. The correctness proof hinges on the observation that any optimal scoring alignment will *not* contain a single deletion of $h \geq n$ characters of X ($n = |X|$). This is so because otherwise, another alignment exists, identical except for having a deletion of only $h - n$ characters, and possesing a better score. Since WDP examines all alignments with deletions in X of size $< n$, it produces the optimal scoring alignment.

The technique involves computing *two* passes through each row. In both passes, all cells but the first are treated normally. In the first pass, cell $S[i, 1]$ (corresponding to Y_i and X_1) is given the better of 1) a value derived from the cell $S[i-1, 1]$, the first cell in the row above (corresponding to a deletion of Y_i) and 2) a value derived from cell $S[i-1, n]$, the last cell in the row above (corresponding of a pairing of Y_i and X_1). This later is a wraparound value. In the second pass, $S[i, 1]$ receives the maximum of 1) its current value, and 2) a value derived from $S[i, n]$, the last cell in its row (corresponding to a deletion of X_1). This is also a wraparound value.

3 Crossing Tandem Cyclic Alignments

Here we show that although tandem cyclic alignments may cross, no alignment can cross the "same" alignment more than once. "Same" in this case means an alignment that has been shifted one or more full copies of the pattern left or right, similar to the shifting of the alignment A_1 to become A_{n+1} in the Maes algorithm.

3.1 An example

Let the pattern X and text Y be

$$X = gaccga \quad Y = accgatacgagacccgagaacgagaccg.$$

Then, using an edit distance scoring scheme, (match=0, mismatch, indel=1), the *only* best scoring alignment of $X^k[1]$ vs Y (with a score of 6) uses 5 copies of $X[1]$:

```
           *                *
gaccga gaccga ga-ccga gaccga gaccga
-accga ta-cga gacccga gaacga gaccg-
```

while the *only* best scoring alignment of $X^h[4]$ vs Y (with a score of 8) uses 4 copies of $X[4]$:

```
      *                *
--cgagac cgaga-c cgagac cgaga-c-
accgata- cgagacc cgagaa cgagaccg
```

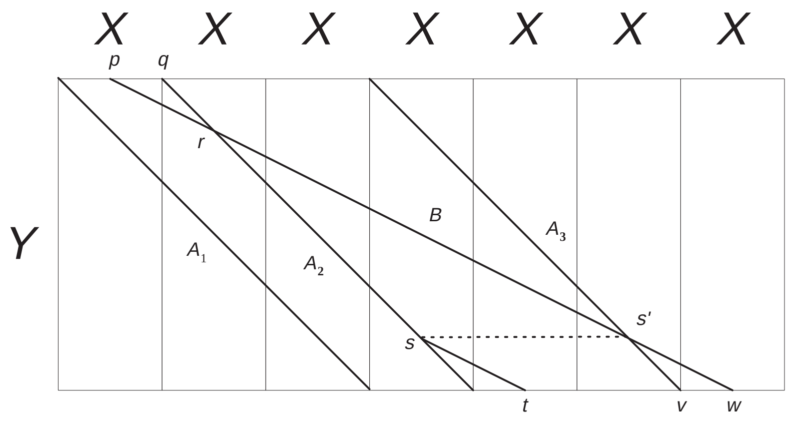

Fig. 2. An illustration of Theorem 1.

Since the alignments use a different number of copies of X, they cross and there is no set of best scoring pairwise non-crossing alignments.

3.2 No alignment crosses the "same" alignment more than once

Theorem 1. *Given two sequences, X and Y and an index i, $1 \leq i \leq n$, let c_i be the number of copies of $X[i]$ in a best scoring alignment of $X' = X^k[i]$ vs Y over all $k = 1, 2, \ldots$, where all of Y and all of X' must be included in the alignment (*i.e. *$c_i = k$ in that best scoring alignment). Then, for any j, $1 \leq j \leq n$ there exists a best scoring alignment of $X' = X^h[j]$ vs Y over all $h = 1, 2, \ldots$ such that $c_j = h$ in that alignment and $|c_i - c_j| \leq 1$.*

In other words, if a best scoring alignment of $X^k[i]$ vs Y uses c copies of $X[i]$, then for any j, there is a best scoring alignment of $X^h[j]$ vs Y which uses one of $\{c-1, c, c+1\}$ copies of $X[j]$.

Proof. Assume that $|c_i - c_j| > 1$. We show a contradiction if $c_j > c_i + 1$. A similar argument holds for $c_j < c_i - 1$. Refer to figure 2. Let A_1 be a best scoring alignment of $X^{c_i}[i]$ vs Y and let B be a best scoring global alignment of $X^{c_j}[j]$ vs Y with smallest c_j and let $c_j > c_i + 1$. Let A_2 be a duplication of A_1 shifted to the right by one copy of $X[i]$ and let A_3 be the rightmost shifted copy crossed by B. (By assumption, A_2 and A_3 are distinct.)

Let r and s' be the points, respectively, where B crosses A_2 and A_3 and let s correspond to the point on A_2 matching s'. Call $x \succ y$ the part of an alignment from point x to point y. Let $s \succ t$ be a duplication of $s' \succ w$ in B shifted to the left. Finally call $cost(x \succ y)$ the alignment score for $x \succ y$ (and recall that we are assuming distance scoring so that smaller cost is better).

Claim 1: $cost(r \succ s') \geq cost(r \succ s)$. Otherwise, piece together $q \succ r$, $r \succ s'$ and $s' \succ v$ to get a better scoring alignment than A_2. But, A_2 is optimal.

Claim 2: $cost(r \succ s') < cost(r \succ s)$. Otherwise, piece together $p \succ r$, $r \succ s$ and $s \succ t$ to get an alignment with score no worse than B, and using less than c_j copies of $X[j]$. But B uses minimal copies.

Claims 1 and 2 produce a contradiction.

4 The Tandem Cyclic Alignment Algorithm

The Tandem Cyclic Alignment problem is solved in three steps. Each step requires first finding a *guide* alignment and then implementing the Maes algorithm using the guide as alignment A_1. Since we are using tandem copies of the pattern, the Maes algorithm will be implemented as *Bounded Wraparound Dynamic Programming* (BWDP) which is described following the outline of the main algorithm:

Step 1: Use pattern and text global WDP (section 1.2) to find the best scoring alignment of $X^k[1]$ vs Y for $k = 1, 2, \ldots$. Call this alignment A. Let the number of copies of X used in A be c. Use A as A_1 in the BWDP version of the Maes algorithm to find the remaining best scoring non-crossing alignments for $X^c[i]$ vs Y for $i = 2, \ldots, n$. Call the best scoring alignment from this step B_c. Save B_c.

Call c^* the number of copies in the solution to the tandem cyclic alignment problem. At this point, we have saved the best from the set of cyclic alignments each of which uses c copies of X, but we do not know if $c = c^*$. However, by Theorem 1, we know that $c^* \in \{c-1, c, c+1\}$.

Step 2: Using A (from step 1) and a copy of A shifted to the *right* one pattern length, find the best scoring alignment of $X^{c+1}[1]$ vs Y using BWDP. Call this alignment A^+ (figure 3). Use A^+ as A_1 in the BWDP version of the Maes algorithm to find the remaining best scoring, non-crossing alignments for $X^{c+1}[i]$ vs Y for $i = 2, \ldots, n$. Call the best scoring alignment from this step B_{c+1}. Save B_{c+1}.

Step 3: Using A (again from step 1) and a copy of A shifted to the *left*, find the best scoring alignment of $X^{c-1}[1]$ vs Y using BWDP. Call this alignment A^-. Use A^- as A_1 in the BWDP version of the Maes algorithm to find the remaining best scoring, non-crossing alignments for $X^{c-1}[i]$ vs Y for $i = 2, \ldots, n$. Call the best scoring alignment from this step B_{c-1}. Save B_{c-1}.

Step 4: Choose the best scoring alignment from B_c, B_{c+1} and B_{c-1}.

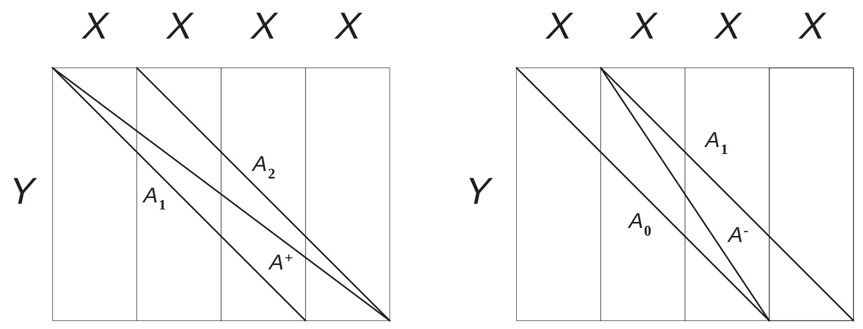

Fig. 3. Finding the guide alignments: A^+ (left) and A^- (right). BWDP achieves the same result using only one copy of X.

Time complexity. Each of the three main steps starts with finding a guide alignment using WDP or BWDP in time $O(nm)$. Then each step finds the remaining alignments using the BWDP version of the Maes algorithm in $O(nm \log n)$ time. The total time is therefore $O(nm \log n)$.

4.1 Bounded Wraparound Dynamic Programming (BWDP)

BWDP is computed in an alignment matrix $W[i, j]$ of size (m+1)(2n+1), *i.e.* it uses *two* copies of X. We are given two alignments L and R as boundaries. We assume that L and R are both alignments of $X^c[j]$ vs Y for a fixed c and different j and that neither crosses outside the pair of "master" bounding alignments $X^c[1]$ vs Y and its duplicate shifted right one copy of X (or alternately $X^c[1]$ vs Y and its duplicate shifted left).

We use L and R to obtain, for each row $i = 0, \ldots, m$ in the matrix, the leftmost, $L[i]$, and the rightmost, $R[i]$, boundary columns between which alignment scores will be computed. Finally, we are given an index k, $L[0] \leq k \leq R[0]$ as the starting column for the alignment. Figure 4, left side, shows the bounded computation as it would appear if we use an unrestricted number of copies of X. Note that for some i, $L[i]$ may be left of the starting position k or $R[i]$ may be right of the ending position $k + cn$. In this case, we contract the boundaries to the appropriate values.

Figure 4, right side, shows the same bounded computation, but this time, in an array which contains only two copies of X. When the boundaries exceed the first two copies of X, the computations wrap around. There is only one question which must be addressed to guarantee that the BWDP result is the same as the unrestricted-copies-of-X result. Can the boundaries collide or cross in the space of only two copies of X (*i.e.* can R catch up with L as they wrap around)?

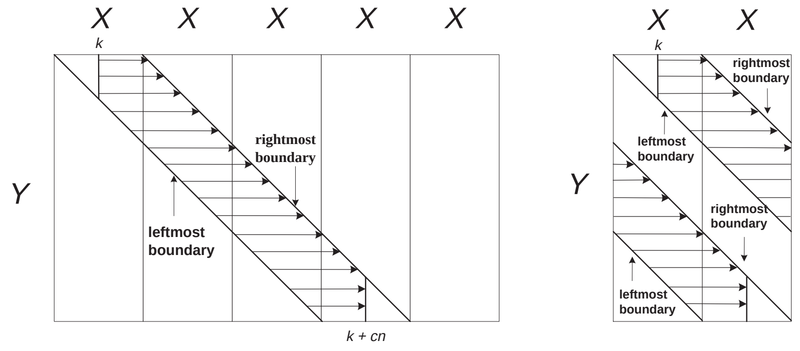

Fig. 4. Bounded wraparound dynamic programming simulates computation with an unrestricted number of copies of X.

Definition: The *width* of the computation space is the maximum difference $R[i] - L[i] + 1$, $i = 0, \ldots, m$ in the unrestricted-copies-of-X computation.

Lemma 2 The maximum width of the computation space is $2n$.

Proof. Note that all the boundaries must lie within the "master" boundaries so it suffices to show the maximum width for the masters. Since the master alignments are duplicates separated by one copy of X, corresponding positions in the alignments are n columns apart, *i.e.* they occur in columns c and $c + n$ (figure 5). Consider a row i in which the alignment moves horizontally in the matrix (a deletion of characters in X). If $L[i]$ is in column c, then $R[i]$ is in column $c+n+h$ where h is the length of the horizontal move. As stated previously

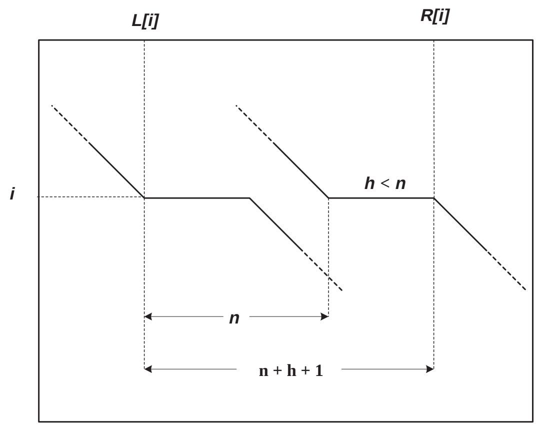

Fig. 5. The maximum width of the computation space is $2n$.

(section 2.3), $h \leq n - 1$, so the maximum possible width of the computation is $2n$.

Corollary 3 The bounding alignments can not collide in the BWDP array which has 2n columns (excluding column zero which is not used after the boundaries wrap around).

5 Application to tandem repeats from the *C. elegans* genome

We implemented the tandem cyclic alignment algorithm and used it to analyze the consensus patterns of tandem repeats found in the *C. elegans* genome. Our goal was to identify pairs of patterns, one of which is a multiple approximate copy of the other. The individual repeats were obtained with the Tandem Repeats Finder (TRF) program [2] which identifies approximately 25,000 tandem repeats in *C. elegans*. From these, we selected nearly 5300 repeats in four groups with nominal pattern sizes of 70 base pairs (bp), 51bp, 35bp, and 17bp. (Repeats within a group had pattern sizes within 3bp of the nominal size.) Each repeat was paired with every repeat from all groups of smaller nominal pattern size (except 70 bp which was not paired with 51 bp and 51 bp which was not paired with 35 bp) and tandem cyclic alignment was run on all pairs.

DNA consists of two strands, one of which is the *reverse complement* of the other. In a reverse complement, the direction of the sequence is reversed, the As and Ts are swapped and the Cs and Gs are swapped. Since similar repeats may have been found as reverse complements, for every pair, we first align the patterns as they appear and then reverse complement one pattern and align them again.

The total number of alignments (including reverse complements) was 16.3 million. On a 500 Mhz PC, the alignments took 17 and 3/4 hours or just over an hour per million alignments.

Figure 6 illustrates an example of the relationships found in this search. It consists of the alignment of the consensus patterns from 3 repeats, from widely scattered genomic locations, that were found to be related. Pattern 1176 is 19 bp long. Four copies are shown (indicated by alternate shading). Pattern 197 is 34 bp long. Two copies are shown. Pattern 8989 is 68 bp long. One copy is shown. For the latter two patterns, only the differences with the top pattern are indicated. A dash ($-$) means that there is no character that corresponds to the character in the top line. This is an insertion (into the top line) or a deletion (from the second or third line).

Notice first that pattern 8989 is almost identical to two copies of pattern 197, differing only in the substitution of A and C. TRF is able to find such closely related patterns (of different sizes) for the same repeat and in fact reported that repeat 8989 also had a pattern of size 34 that was identical to 197.

Next compare patterns 1176 and 197. Pattern 197 consists of two copies of 1176 with 6 differences. Because the two halves of 197 were quite different, TRF did not report a pattern of size 19 (or any other similar size) for repeat 197. The following scenario (highly speculative!) may have occurred. Two identical 19 bp copies existed in an ancestral repeat and one of those copies was extensively mutated, including the deletion of 4 nucloetides. The resulting pair of repeats, now 34 bp long was subsequently transposed to another location in the genome where it duplicated, forming a tandem repeat. Some evidence for this scenario exists in one of the copies of repeat 1176 which contains the adjacent two nucleotide deletion seen in pattern 197.

Without the tandem cyclic alignment algorithm, the relationship between patterns 197 and 1176 would be less clear. Our goal in comparing these patterns is to obtain an *accurate* measure of the distance between them. If we used the pattern local, text global tandem alignment algorithm (section 1.2), the results would depend on the presentation of the text *i.e.* the cyclic permutation in which it appears in the input. If pattern 197 (the text) were presented starting just after a deletion (at the AAT following the two nucleotide deletion for example), then the algorithm would fail to align any characters from pattern 1176 with the positions of the two deleted characters (which do not actually occur in pattern 197). On the other hand, if pattern 197 were presented starting as it does in figure 6 at GCAA, then all the deleted characters will appear in the alignment. The alignment score will be different in these two cases.

Adjustment of alignment score obtained by the pattern local, text global tandem alignment algorithm is possible, but not necessarily straightforward. As an illustration, consider the pattern local, text global alignment (left below) and the tandem cyclic alignment (right below) of text X and pattern Y:

$$X = gggtgg \quad Y = gggt.$$

```
gggt gg        gggt gg--
gggt gg        gggt gggt
```

The former can be transformed into the later by merely adding the deleted characters and the cost for the gap. But, a different situation occurs if the last character of the text is changed to t:

$$\hat{X} = gggtgt.$$

```
gggt gt        gggt g--t
gggt gg        gggt gggt
```

Now the score changes not only by the cost of a gap, but also by the loss of a mismatched pair and the gain of a matched pair. More complicated situations are not difficult to construct. The tandem cyclic alignment algorithm however always gives the correct score without manipulation regardless of the presentation of the pattern or text.

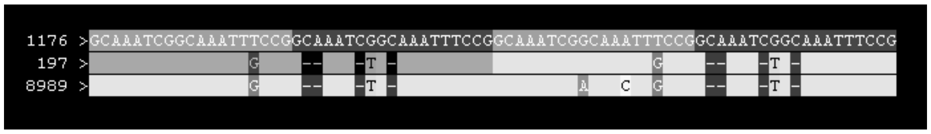
1176 >GCAAATCGGCAAATTTCCGGCAAATCGGCAAATTTCCGGCAAATCGGCAAATTTCCGGCAAATCGGCAAATTTCCG
197 > G -- -T - G -- -T -
8989 > G -- -T - A C G -- -T -

Fig. 6. An example of aligned consensus patterns of different sizes.

6 Conclusion

We have defined a new alignment problem, tandem cyclic alignment and provided an algorithm which solves this problem in $O(nm \log n)$ time for two sequences of length n and m, $n \leq m$ when using any alignment scoring scheme with additive gap costs. The algorithm was used to compare tandem repeats from the *C. elegans* genome in order to identify pairs of repeats with an evolutionary relationship where the consensus pattern of one is a multiple of the consensus pattern of the other. We showed an example of one such relationship which would not be reliably recognized with other alignment algorithms.

References

1. A. Apostolico, M.J. Atallah, L.L. Larmore, and S. Mcfaddin. Efficient parallel algorithms for string editing and related problems. *SIAM J. Comput.*, 19:968–988, 1990.
2. G. Benson. Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Research*, 27:573–580, 1999.
3. V. Fischetti, G. Landau, J. Schmidt, and P. Sellers. Identifying periodic occurrences of a template with applications to a protein structure. In A. Apostolico, M. Crochemore, Z. Galil, and U. Manber, editors, *Proc. 3rd annual Symp. on Combinatorial Pattern Matching, Lecture Notes in Computer Science*, volume 644, pages 111–120. Springer-Verlag, 1992.
4. H. Fuchs, Z. Kedem, and S. Uselton. Optimal surface reconstruction from planar contours. *CACM*, 20:693–702, 1977.
5. G.M. Landau, E.W. Myers, and J.P. Schmidt. Incremental string comparison. *SIAM J. Comput.*, 27:7–82, 1998.
6. V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Phys. Dokl.*, 10:707–710, 1966.
7. M. Maes. On a cyclic string-to-string correction problem. *Information Processing Letters*, 35:73–78, 1990.
8. W. Miller and E. Myers. Approximate matching of regular expressions. *Bulletin of Mathematical Biology*, 51:5–37, 1989.
9. J. Schmidt. All shortest paths in weighted grid graphs and its application to finding all aproximate repeats in strings. *SIAM J. Comput.*, 27:972–992, 1998.