DIMACS Technical Report 2005-39
November 2005

An evaluation of the edit-distance-with-moves similarity metric for comparing genetic sequences

by

Shiri Azenkot[1]
Dept. of Math and Computer Science
Pomona College
Claremont, CA 91711

Tzu-Yi Chen[2]
Dept. of Math and Computer Science
Pomona College
Claremont, CA 91711

Graham Cormode[3]
Bell Laboratories
New Providence, NJ 07974

[1]Research for this report began when this author was a DIMACS/DIMATIA NSF-REU program participant, 2004.
[2]This author is supported in part by the National Science Foundation under Grant No. CCF-0446604. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.
[3]Research for this report began when this author was a DIMACS postdoc.

DIMACS is a collaborative project of Rutgers University, Princeton University, AT&T Labs–Research, Bell Labs, NEC Laboratories America and Telcordia Technologies, as well as affiliate members Avaya Labs, HP Labs, IBM Research, Microsoft Research, Stevens Institute of Technology, Georgia Institute of Technology and Rensselaer Polytechnic Institute. DIMACS was founded as an NSF Science and Technology Center.

ABSTRACT

We describe the first known implementation of an approximation algorithm for the string edit distance with moves similarity metric. This is the first algorithm to consider nontrivial alignment and run in substantially sub-quadratic time [2]. Extensive experimentation demonstrates that the algorithm produces a good approximation for the edit distance with moves, especially on strings of length 500 B to 10 KB. We also found that the algorithm has high potential for use in computational biology. When comparing texts of genetic sequences, our algorithm outperforms the q-grams heuristic in predicting results of the Smith-Waterman algorithm. Finally, we propose additional application areas for our implementation.

1 Introduction

The problem of string similarity takes a number of strings as input and returns a numeric value that represents the degree of similarity among the strings. A related concept is the distance between two strings, where a high numeric value indicates a low degree of similarity. Many similarity metrics have been developed and are used in a variety of applications including signal processing and text retrieval. One of the largest application areas is computational biology, where text comparison of genetic sequences provides important information about sequence functionality and molecular evolution. Genetic sequences are represented as strings and the similarity metric used must account for mutations and evolutionary alterations [4].

Biologists currently use heuristics derived from the Smith-Waterman algorithm, which computes a measure of string similarity based on the number of character insertions, deletions, and replacements (this is known as the edit or Levenshtein distance) [5]. Since both the time and space requirements of Smith-Waterman are quadratic in the length of the strings being compared, more efficient algorithms that produce similar results are highly desired. A new string similarity metric called the edit distance with moves (EDM) was introduced by Cormode and Muthukrishnan [2]. The EDM is an extension of the edit distance similarity metric. Given two strings, the EDM is the smallest number of character deletions, insertions, and replacements and substring moves needed to turn one string into another. Cormode and Muthukrishnan describe an algorithm that runs in $O(n \log n)$ time and computes the approximation of the EDM. This algorithm, called the edit-sensitive parsing (ESP) method, is the first algorithm that considers nontrivial alignment in substantially sub-quadratic time. In this paper we evaluate the claim that the algorithm has high potential for solving problems involving genetic sequences [2].

We describe the first known implementation of the ESP method and investigate (1) how well the algorithm approximates the EDM, and (2) whether it has any applications to problems in computational biology. First, we evaluate the quality of the algorithm's approximation in terms of efficiency and accuracy. We then explore how useful the EDM similarity metric and the ESP method are for comparing genetic sequences, and review possible advantages this approach offers over existing techniques.

Extensive experimentation shows that the ESP method provides a good approximation for the EDM, and outperforms a leading heuristic in predicting Smith-Waterman results. We ran our implementation repeatedly on strings of varying length and degrees of similarity and found that the ESP method has success rates of over 90 percent. Success rates are highest for strings with lengths ranging from 500 B to 10 KB, reaching 97 percent for 1 KB strings. Performance is high for both strings of normal text and sequences of four-character alphabets; the latter simulate genetic sequences. Our results indicate that the ESP method outperforms the q-grams heuristic in predicting the Smith-Waterman algorithm's output, in addition to offering an efficient asymptotic bound on the worst-case running time. We thus show that the EDM is a relevant similarity metric for genetic sequences and the ESP method has potential for use in computational biology.

2 Background

In this section we provide a brief overview of the state-of-the-art techniques used by biologists for genetic sequence comparison. Then we describe key features of the ESP method.

2.1 String similarity in computational biology

Computational biology is currently one of the major application areas for string similarity algorithms. In this field, string similarity algorithms are used to compare the texts of DNA and protein sequences. A good similarity metric must identify common patterns among the sequences while allowing for some "error" brought about by evolution or missing information. Text comparison of genetic sequences is central to solving many biological problems, since sequence determines shape, which in turn determines function. For example, measures of similarity among sequences aid biologists in the construction of phylogenetic trees, which show the evolutionary lineage of organisms [4].

Biologists use the technique of multiple sequence alignment to compare genetic sequences. In this technique, the sequences are lined up against each other so that the maximum number of characters in corresponding positions match. Gaps are inserted in the sequences wherever necessary. The overall score or strength of an alignment is a function of the number of matching characters among the strings with penalties for the number of gaps. While sequence alignment can be done among two or more strings, this article deals with alignments of two sequences only.

Smith-Waterman (SW) is the optimal algorithm for multiple sequence alignment. It utilizes dynamic programming and guarantees an optimal solution. As such, the asymptotic bounds for pairwise alignment for both time and space are $O(mn)$, where m and n are the lengths of the strings being compared [5]. As an attempt to improve upon this bound, many other methods of multiple sequence alignments have developed which use heuristics based on the SW algorithm. BLAST (Basic Local Alignment Search Tool) is a widely used SW heuristic which emphasizes speed over sensitivity [1].

Other string similarity measures have also been implemented as filters for multiple sequence alignment problems. These are used when the goal is to search a database for the sequence that is most closely aligned to some given target sequence. A filter algorithm will use a similarity criterion to eliminate unlikely candidates from the database. The similarity criterion gives a measure of how similar the two strings are, but not necessarily in terms of a multiple sequence alignment score. Q-grams are a notable example of a technique that is implemented in this way [7]. This heuristic is based on the observation that if two strings have an edit distance that is below a certain threshold, they are likely to share many substrings of length q, called q-grams.

Like BLAST and q-grams, most heuristics based on SW sacrifice accuracy for improved efficiency. It is thus an important open problem to find efficient methods of genetic sequence comparison that are more accurate than methods currently in use.

2.2 Algorithm overview

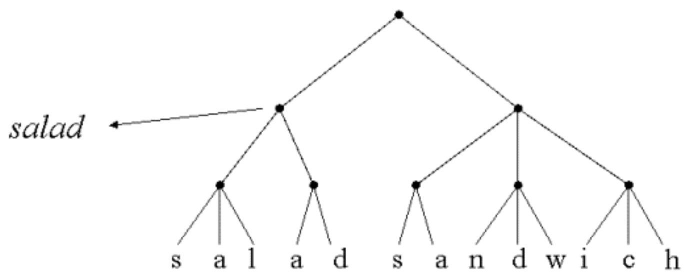

Figure 1: An ESP tree for the string *saladsandwich*.

The ESP method computes a measure of string similarity by dividing the input strings into overlapping substrings of various lengths, and finding the number of substrings shared by the original strings. The set of substrings for a given string is determined by embedding the string into a tree, where each node represents a substring. This process of embedding is the core of the algorithm, and can be applied to many specific instances of the string similarity problem. Cormode and Muthukrishnan describe the ESP method in the context of the string edit distance matching problem [2].

Strings are embedded into trees in a process called edit-sensitive parsing (ESP). The goal of edit-sensitive parsing is to minimize the effects of edit operations on the parse tree. In other words, if two strings x and y have a small edit distance, their trees should appear to be more similar than the strings themselves. ESP uses the deterministic coin tossing method, also known as the local symmetry breaking technique, to construct the parse trees.

An ESP tree is a 2-3 tree, where each character in the original string is represented by a leaf node. Figure 1 shows an example of the ESP tree for the string *saladsandwich*. Every node represents a substring composed of all of the node's leaf descendants. The root node represents the entire original string, since all leaf nodes are its descendants. The process of constructing the tree begins at the bottom level and ends at the root. At each level, the nodes are divided into groups of size two or three, and assigned a common parent node. The grouping process involves several steps.

Before the nodes are grouped, the current level of the tree is divided up into sections, called metablocks. The grouping method used depends on the kind of metablock. The three types of metablocks are:

1. Strings that consist of a repeated symbol.

2. "Long" substrings, at least $\log^* \|\Sigma_(i-1)\|$, and not of type 1.

3. "Short" substrings, shorter than $\log^* \|\Sigma_(i-1)\|$ and not of type 1.

Metablocks of types 1 and 3 are grouped using the same straightforward method. Groups of three are created whenever possible, and remaining nodes are paired up.

Type 2 metablocks are grouped by reducing the alphabet size of the block, marking certain nodes, and splitting the block into groups of two and three according to these marked nodes. The alphabet of a metablock is reduced iteratively until only the characters 0, 1, and 2 remain. At this point the algorithm finds landmarks across the block by identifying local maxima and minima. The spacing of the landmarks should allow for the grouping of pairs and triplets. Nodes are divided so that every group consists of one landmark and one or two of its neighboring nodes. Each group is assigned a common parent node, and the next level of the tree is created.

After the ESP tree is constructed, the algorithm computes the number of occurrences of each substring in the tree. This is called the characteristic vector of the tree. The original string is thus represented in a vector space. The distance between the strings is approximated by computing the vector distance between the characteristic vectors of these strings. In other words, the number of common substrings among the original strings is found and used as an approximation of the edit distance with moves.

The ESP method runs in time $O(n \log n)$, and the EDM approximation is within a factor of $O(\log n \log^* n)$, where n is the length of the larger string in both bounds [2]. This algorithm is the first string pattern matching algorithm to consider nontrivial alignment which runs in substantially sub-quadratic time. Since the embedding process is independent, it can be used to solve a number of string proximity problems including nearest-neighbor, outlier detection, and stream-based approximation of string distances (see [2] for more details).

3 Implementation

In this section we describe our implementation of the ESP method which differs slightly from the original description of the algorithm. We made minor simplifications which we believe have a negligible effect on the algorithm's output and keep the approximation within the proven bound. Additionally, we have added features to accommodate issues that were not previously addressed.

The implementation differs from the original specifications of the ESP method in the details of constructing the ESP tree. As mentioned in section 2.2, before the nodes of a level are grouped, they are divided into three types of metablocks: repeating, short, and long substrings. Our implementation does not distinguish between "long" metablocks of type 2 and "short" metablocks of type 3. The straightforward grouping method is used for metablocks of type 1 only, while metablocks of types 2 and 3 are both grouped using the more complex grouping method, which involves alphabet reduction and landmark selection. This modification was done for the sake of simplicity, and its effect on the algorithm's output is probably negligible. The short strings form relatively few groups, and there is a smaller number possible groupings.

Our implementation also introduces a couple of other new features in the construction process of the ESP tree. These features account for situations that were not explicitly addressed by Cormode and Muthukrishnan [2]. When grouping metablocks of type 2, each node is grouped with its nearest landmark, where preference is given to the landmark on the right. Upon closer inspection of this grouping scheme, ambiguous situations arose. In rare cases, strictly following the grouping guidelines did not yield groups of two and three. To deal with this problem, the implementation of the ESP method creates groups of size one when necessary, so the parse tree formed is not strictly a 2-3 tree. Most nodes have either two or three children, but on rare occasions a node with one child is created. Since these instances are extremely uncommon, their effect is minimal and they can be ignored.

The second new feature of the implementation is incorporated into the process of alphabet reduction. During this process, the current level in the tree is traversed, and each node receives a new label. A label is computed by comparing the current node to its left neighbor. This poses a problem for the first character in a metablock since it doesn't have a left neighbor. The implementation introduces a way of dealing with this. a default value is used to function as the left neighbor of the first node of a metablock. All nodes can then be compared to their left neighbors.

In the subsequent sections we discuss the extensive tests we performed by running this implementation of the ESP method.

4 Quality of Approximation

Our first goal after implementing the edit-sensitive parsing method was to test the quality of the algorithm's approximation. In other words, we wanted to see if its output was a good indicator of the edit distance with moves between the two input strings. If the ESP method yields good results, then it can be used to solve the edit distance with moves problem in a variety of applications.

Below we describe the experiments conducted to test the quality of approximation and their results. First we discuss what criteria we used to measure the quality of approximation and the specific details of how we designed and conducted our tests. This includes where and how we obtained our test cases and what parameters we used. Then we discuss the results of these tests and what they reveal about the algorithm's performance.

4.1 Criteria for a good approximation

The quality of the algorithm's approximation is evaluated in terms of its efficiency and accuracy (see Figure 2 for a visual representation of criteria). The algorithm's efficiency was discussed by Cormode and Muthukrishnan, who prove the asymptotic bound on the runtime is $O(n \log n)$ [2]. As previously mentioned, this bound is a significant improvement over other string similarity algorithms that consider nontrivial alignment. Thus, we consider the criterion of efficiency to be fulfilled. The latter criterion—the accuracy of the algorithm's output—is more complex.

We measure accuracy in two ways: the magnitude of the approximation factor and the relative accuracy of the ESP method. The approximation factor is equal to the output

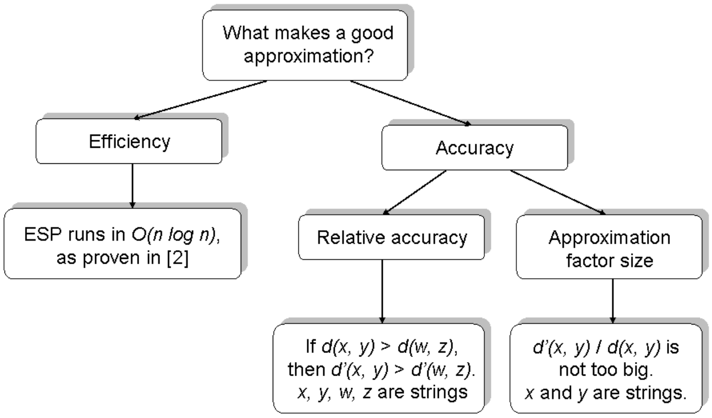

Figure 2: Criteria for evaluating the quality of the ESP method's approximation. The edit distance with moves is defined as $d(x, y)$ and the ESP method's output is defined as $d'(x, y)$.

divided by the actual solution to the problem, and indicates how close the approximation is to the real solution. The distance of the approximation from the actual solution turns out to be unimportant in most applications of the problem, however. In most applications of text comparison, the actual numeric value of the similarity measure is unimportant. What matters is how this numeric value compares to the similarity measures of other sets of strings. We thus focused on the relative accuracy as our guideline, so the output of the ESP method was considered relative to outputs of the algorithm with different inputs. High relative accuracy in our study indicates that the algorithm correctly identified that one pair of strings is more similar than another. Note that similarity among strings is inversely related to their edit distance with moves. Thus, in order to be relatively accurate the algorithm should output a lower number for a pair of strings that are more similar to each other, and a higher number for a pair of strings that are farther removed from each other.

4.2 Tools for assessment

In order to assess the relative accuracy of the algorithm, a reliable measure of the true EDM is needed. With such a measure, it is possible to see whether the algorithm's output corresponds to the EDM values when comparing pairs of input strings. There is no known sub-exponential algorithm that computes the EDM of two arbitrary strings. Thus, we obtained a reasonable estimate of the EDM while generating the pairs of strings we used as input to our program. We define the edit distance with moves to be $k = d(x, y)$, where x and y are strings generated in the following way.

1. Begin with an arbitrary string x.

2. Randomly select one of the four allowed operations: deleting, inserting, or replacing a character, or moving a substring.

3. Randomly select appropriate indexes in the string and perform the operation selected in step 2.

4. Repeat steps 2-3 k times.

5. The modified string is now called y, and $d(x, y) = k$.

According to this definition, $d(x, y)$ is actually an upper bound on the edit distance with moves between x and y; there might be an alternative sequence of operations that transforms x into y in fewer than k steps. However, if the length of x is significantly larger than k, it seems likely that the sequence of operations that has been performed is nearly optimal, and that $d(x, y)$ is fairly accurate. The bottom line is that $d(x, y)$ is substantially more accurate than the ESP method's approximation, referred to as $d'(x, y)$.

To obtain a quantitative measure of the relative accuracy, we implemented a success rate function. This function takes as input a set of experimental results for any number of test cases. Each test case, defined by two strings x and y, is represented by the ESP method's output $d'(x, y)$, and the EDM value $d(x, y)$. The success rate function randomly selects two test cases from the series and checks whether the test case with the higher EDM value produced a higher ESP approximation as well. If x and y are the strings in the first test case, and w and z are the strings in the second test case, the current iteration is successful if $d(x, y) > d(w, z) \Rightarrow d'(x, y) > d'(w, z)$. In other words, the function checks whether the ESP method correctly detected that the pair of strings in one test case is more similar than the pair of strings in the second test case. If it has, this trial is considered a success. This process is done continuously for a given number of iterations (the higher, the better). After the specified number of iterations have been performed, the function returns the proportion of successful trials to the total number of iterations. This value is the success rate of the ESP method for the given data.

4.3 Methods

We tested the ESP method on a collection of strings of varying lengths and degrees of similarity. Since our program gives an approximation of the edit distance with moves between two strings, a test case consists of two strings. One of the two was generated automatically from the other in the process described above. We therefore only had to obtain one string for each test case. An alphabet size of 93, the number of printable characters, was used. Files of this alphabet size included ordinary text files obtained from arbitrary texts from Project Gutenberg (http://www.gutenberg.org). We gathered a collection of strings that varied in length from 500 B to 75 KB, where each string had a unique length.

Each string we obtained was included in numerous test cases, where a test case compared the original string with another string that was unique to that case. We created our test

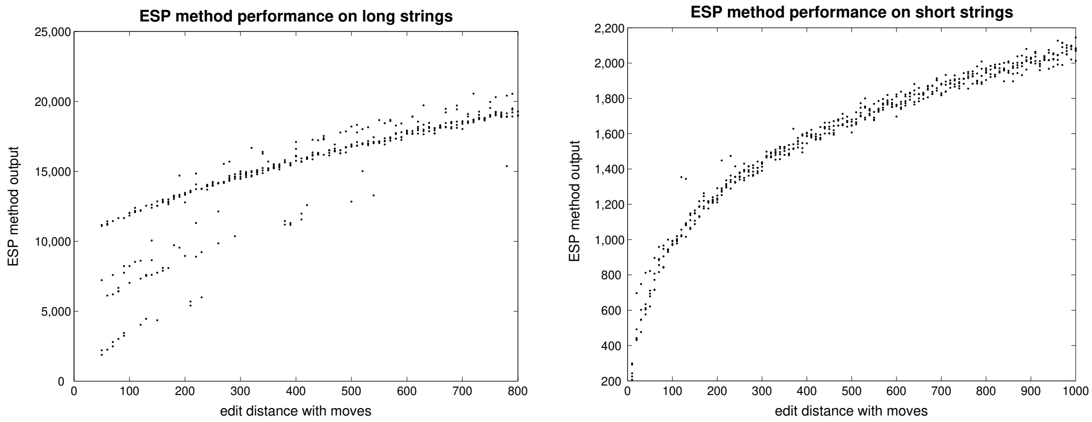

Figure 3: The plot on the left shows the ESP method's performance on strings that are 50 KB long and the plot on the right shows its performance for strings that are 1 KB long. All strings have an alphabet size of 93, the number of printable characters. Each point on the plots represents a test case for two strings x and y. The plots show how $d'(x, y)$ changes as a function of $d(x, y)$

cases in the following way. For a given string of a certain length, we generated new strings with EDM values ranging from 10 to 800. This process was repeated five times, so that we ended up with five test cases for a constant string length, alphabet size, and EDM value. Note that the fact that the same string appeared in so many test cases did not matter since it was paired with a unique string in each case. The string's identity makes little difference, and the relationship between the strings of the pair (which did vary with each case) is the main factor that affecting the performance of the algorithm.

The many test cases we created were used as input to the ESP method to perform a series of tests. A discussion of the results of these tests is provided in the following section.

4.4 Results and Discussion

In this section we describe our experimental findings regarding the accuracy of the ESP method, addressing its approximation factor and relative accuracy.

4.4.1 Relative accuracy

Our results are organized in a series of scatter plots that show the output of the ESP method against the edit distance with moves values for a constant string size. Each point in the scatter plots shown here represents a test case of two input strings, x and y. On the vertical axis we plot $d'(x, y)$ and on the horizontal axis we plot $d(x, y)$. The plots in Figure 3 show our experimental results for 50 KB and 1 KB strings.

The experiments indicate that the ESP method provides a good measure of the edit distance with moves, with success rates mostly over 90 percent. Performance was significantly better on smaller strings. The appearance of the scatter plots suggests that there is higher consistency among smaller strings in our graphs. The computed success rate supported this observation as well. For 1 KB strings, the success rate was 97 percent, while for 50 KB strings, it was 93 percent. The graphs we created exhibit two major trends which are both more pronounced for smaller string lengths. First, the ESP method's output, $d'(x, y)$, increases as a polynomial function of the EDM, $d(x, y)$. The function shows more curvature as the string size decreases. Ideally, we would expect a linear relationship between approximate and actual measures. However, we must also remember that the edit distance $d(x, y)$ is an estimate. As the value of edit distance operations increases, this estimate becomes less reliable. Although this may help account for the polynomial trend, at this point we cannot explain this trend for certain.

The second-and more striking-trend is that several polynomial curves emerge in the graphs. These curves converge as the EDM increases. Larger string sizes, such as the 50 KB strings shown in Figure 3, yield one thick line and two or three additional slimmer lines beneath it. As the string length decreases, the several lines become two defined curves. For all string lengths, the curves eventually converge. This point of convergence, however, also decreases along with the string size. In tests performed on strings that are 1 KB in length, shown in Figure 3, the point of convergence is at an EDM value of about 100. The decreasing point of convergence and the reduction in the number of curves, show that the algorithm yields more consistent and more accurate results for smaller strings. As with the first trend described, the causes of the converging curves are possible directions for future research.

4.4.2 Approximation factor

The experiments indicate that in addition to a higher relative accuracy, the ESP method yields a smaller approximation factor for shorter strings. The graphs that plot the ESP method's approximation against the EDM show how the approximation factor, $d'(x, y)/d(x, y)$, changes as a function of the EDM. The approximation factor decreases as the EDM increases, which is a direct result of the polynomial trend discussed earlier. Comparing these graphs to each other suggests that the approximation factor changes as a function of string length as well. As demonstrated by Figure 3, the approximation factor was significantly higher (much higher than expected) for larger strings. For 50 KB strings, the approximation is over 30, while it is one tenth that amount for 1 KB strings. The decrease in approximation factor is gradual as the strings shorten.

The change in approximation factor as a function of string length can be seen clearly in Figure 4. As with the graphs in Figure 3, we see several curves in Figure 4. In the latter, however, the curves diverge. Although this may seem contradictory, the findings presented in both Figure 3 and Figure 4 are consistent. A set of vertically aligning points in Figure 4 represent the approximation factors for the five trials we obtained for a given EDM (in this case it is 200) and a given string length. The points along a vertical line are spread apart if the ESP method's output was inconsistent for the given set of parameters. This phenomenon

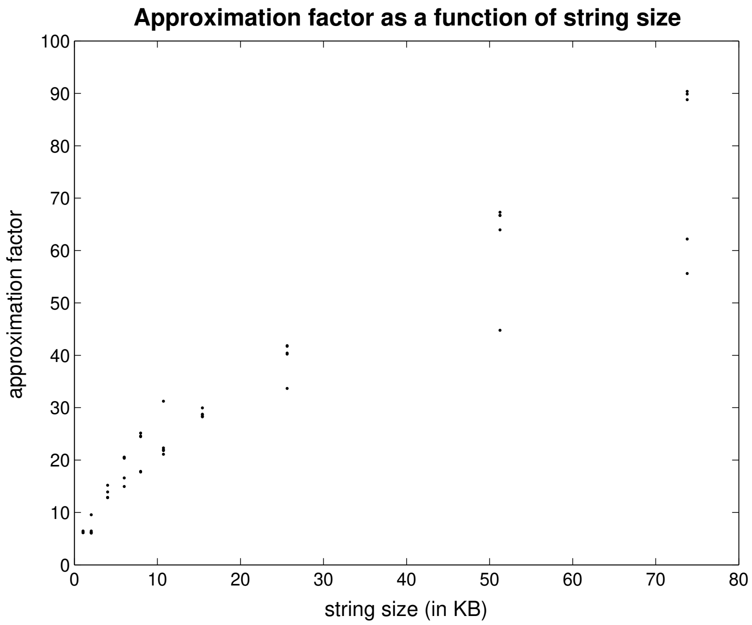

Figure 4: The approximation factor, $d'(x, y)/d(x, y)$ for a test case consisting of the strings x and y, as a function of string length. All strings are over an alphabet of size 93, with $d(x, y) = 200$ for each test case.

is what is illustrated by the converging curves in Figure 3. As we move along a horizontal line in Figure 4, we see the results for strings of growing lengths. Since the larger strings exhibited distinctly separated curves, their approximation factors vary more for a given set of parameters. This is what creates the divergence of the curves in Figure 4.

Most importantly, Figure 4 shows that the ESP method's approximation is bound by some polynomial function ($1/x^n$ such that $0 < n < 1$) . This may not be as tight as the bound given in the original description of the algorithm of $O(\log n \log^* n)$. Perhaps this is due to the fact that our definition of the EDM is not entirely reliable, or to the changes in the approximation factor as a function of the EDM for a given strings length. Despite this discrepancy, the approximation factor appears to be bounded by a sub-linear function and the other results provide sufficient evidence that the ESP method is a good measure of the EDM, especially for smaller strings.

5 Application to Biology

While the first objective of our research focused on the strength of the ESP method in solving the EDM problem, our second objective explores the applicability of the EDM concept itself to the field of computational biology. The main question we address in this section is whether the EDM—and by extension, the output of the ESP method—is relevant in the comparison of genetic sequences. We describe the experiments we conducted in order to determine how the ESP method performs on DNA sequences and how this algorithm compared with current techniques of obtaining similarity measures used by biologists. Two algorithms are used in the analysis: the Smith-Waterman (SW) algorithm and the q-grams heuristic. We consider what advantages, if any, does the ESP method have over these other algorithms.

5.1 Methods

Before comparing the ESP method to other techniques, we wanted to see how well it performs on DNA sequences. We ran the same tests described in section 4.3 on stings of an alphabet size of four, to simulate DNA sequence. These strings were randomly generated, with each character having equal probability of being selected for a given location in the string. Using this set of strings, we created test cases to input to the ESP method as described in Section 4.3.

These test cases were also used as input to existing implementations of the SW and q-gram algorithms. We chose to compare the ESP method with SW in order to see how appropriate the EDM is for measuring sequence similarity in the context of multiple sequence alignment. If the ESP method is a good predicator of the SW algorithm, then it can be relevant to all problems in which SW is used. We chose to compare ESP to q-grams because both provide an approximate measure of string similarity that can be used in conjunction with SW.

Similarity measures were obtained from the SW, q-gram, and ESP algorithms for the same data set. For a quantitative measure of how closely the scores resemble each other, we found the success rates of ESP and q-grams compared with SW. We used the same success rate function described in Section 4.4. We also computed the correlation coefficients between the results of each method. This may not be the ideal statistical tool for this situation, but it provides a quantitative value to be considered as supplement to the other methods of assessment.

We used implementations of the SW and q-gram algorithms available at the SimMetrics Open Source Library [6]. All the implementations of similarity measures in this library outputted a score in the range $[0, 1]$, where 0 is the score for strings that are completely different and 1 is the score for identical strings. In this set of experiments we reused the pairs of four-character strings described above. String similarity measures were obtained for 500 B and 1 KB strings with a four-letter alphabet size to simulate genetic sequences. The pairs of strings we compared were relatively similar to each other, with EDM values ranging from 5 to 100.

5.2 Results and Discussion

5.2.1 ESP method performance on four-character strings

Experimental results on strings of an alphabet size of four were more accurate than those for the larger alphabet. As shown in Figure 5, the approximation did not increase as rapidly as it did for the larger alphabet for increasing EDM values. The curved line in this set of

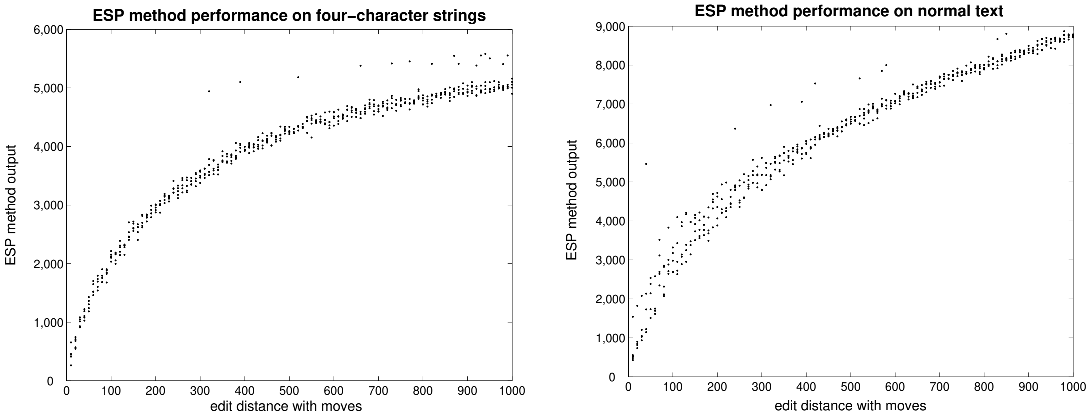

Figure 5: The plot on the left shows the ESP method's performance on strings that are 10 KB in length and have an alphabet size of four. The plot on the right shows the ESP method's performance on strings of the same length, but with an alphabet size of 93.

data resembles a log function more than the polynomial function of the data of the larger alphabet strings. We do see a trace of the multiple curves trend-although in the four letter alphabet most points are along the same line with just a few stray outliners. Furthermore, the four letter alphabet shows more consistent results at lower EDM values, in contrast with the larger alphabet which has yields several curves at lower EDM values. So the relative accuracy is enhanced by the precision of the points to the curve, but negatively affected by the slower growth of the function. In terms of the success rate, these factors cancel each other out: the success rate for strings of length 10 KB for both alphabet sizes is about 96 percent.

The approximation factor of the strings of the four letter alphabet is smaller than that of strings the larger alphabet. Since the curves have different shapes, the difference in approximation factors between strings of the larger and smaller alphabets vary as the EDM increases. As the curve of the graphs of the four letter alphabet strings flattens out, of curve of the graph of the corresponding large alphabet data continues to grow. Thus, the approximation factor of the former will not grow as fast as the latter, and the difference between the two will increase.

Since the four letter alphabet simulates genetic sequences, we can conclude that the ESP method is a good measure of the EDM for genetic sequences, especially when the sequences being compared are relatively similar.

5.2.2 Comparisons of the ESP method with other techniques

Comparing the ESP method's output to other methods will show whether the EDM is an appropriate similarity metric for problems of computational biology. We created scatter plots of the scores of all methods against each other to evaluate (1) how well the ESP method

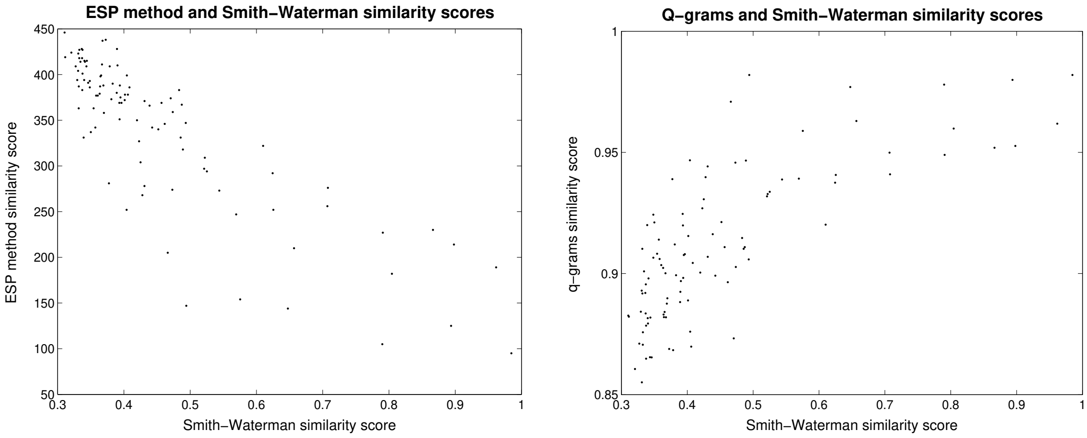

Figure 6: The plot on the left shows the ESP method's output compared to the output of the Smith-Waterman (SW) algorithm, and the lower plot shows the output of the q-grams heuristic compared to the SW algorithm. Similarity scores for SW and q-grams are in the range $[0, 1]$, where higher values indicate more similar (ie, less distant) strings.

predicted the SW similarity measure and (2) how it compared to q-grams. Figure 6 shows the SW output against the ESP method's output, and the SW output against the q-gram algorithm's output for 500 B strings. Note that the relationship between SW and the ESP method is reversed, because lower EDM scores indicate more similar strings. The similarity scores of both q-grams and the ESP method seem to correlate well with the SW scores. The relationship between SW and ESP is fairly linear, while that for SW and q-grams is slightly more curved.

Success rates for the ESP method were higher than those of q-grams against the SW results. The success rate for the ESP method's output is 82 percent, which means that the ESP method correctly identified the less similar of two pairs of strings according to the SW metric 82 percent of the time. The success rate for the q-gram heuristic's output against the SW output was slightly lower, at 78 percent. Thus, the ESP method shows higher accuracy than q-grams for string similarity according to the SW metric.

Correlation coefficients support the findings of the success rate function. The correlation coefficient for the data in Figure 6, between the SW algorithm and the ESP method is -0.83 (the negative correlation is due to the inverse relationship). Meanwhile, the correlation for the data plotted in Figure 6, between q-grams and SW, is 0.75. However, the relationship of the data in this second plot is not quite as linear as the first, so correlation is not the best tool to use for this comparison. While these statistical measures do not produce conclusive results, they support the notion that the ESP method is a potentially good predictor of the SW similarity measure.

We also compared the results of the q-gram algorithm and the ESP method in a scatter plot, shown in Figure 7. The plot shows a very pronounced pattern, with negatively sloped

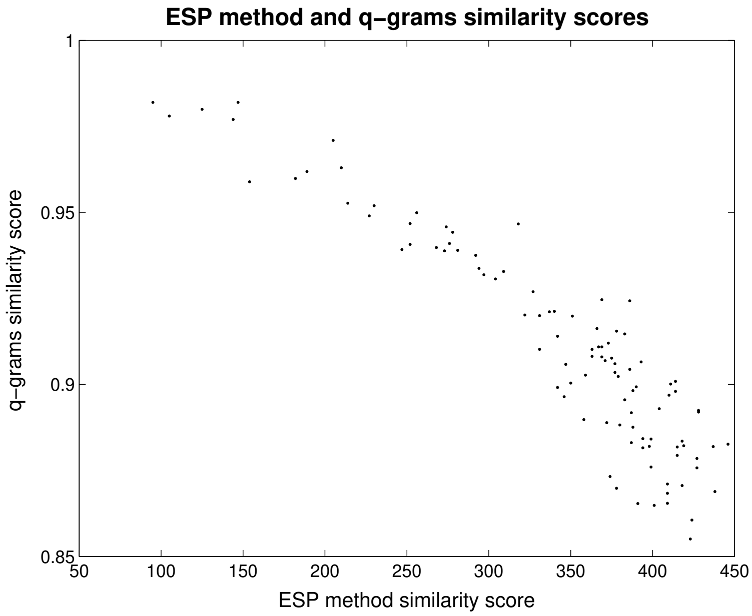

Figure 7: The results of the ESP method and the q-grams heuristic.

curve. The relationship is almost linear, however, and the correlation coefficient is −0.92. This shows that for our test cases the ESP and the q-gram similarity measures were mostly consistent with each other. Our findings thus suggest that if the ESP method was used instead of the q-gram algorithm, it would produce very similar results. One large advantage of the ESP method over the q-gram heuristic is its known asymptotic runtime bound.

Our experiments with 1 KB strings produced similar results. We could not, however, collect much data because the SW algorithm was highly inefficient on these larger strings. This highlights the necessity of finding less costly alternatives in this field, especially when using larger input strings.

6 Conclusion and Future Work

Experimental results of the ESP method's performance pose many open questions in regard to the behavior of the algorithm. We could not fully explain a couple of interesting trends, such as the multiple curves that appear when plotting the ESP method's output against the EDM, and the change in approximation factor as a function of EDM. A better understanding of the algorithm may lead to ways of improving the accuracy of the approximation. One possible direction of improvement is to limit the size of the ESP tree created for each input string by eliminating levels of the tree that are near the root. The closer a node is to the root of a tree, the longer its corresponding substring. It is thus unlikely to find exact substring matches in the other input strings for nodes in the higher levels of the tree, especially for longer strings.

When comparing the ESP method to the SW algorithm and the q-gram heuristic, we found that the ESP method is a good predictor of the SW output. The ESP method's similarity measure was also highly correlated with the q-gram similarity measure. We thus believe that the ESP method can be applied to problems in which the q-gram heuristic is used. Unlike q-grams, the ESP method offers a known asymptotic bound on the worst case running time.

Our comparisons of SW, q-grams, and the ESP method provide a foundation for future research of the ESP method's applications in computation biology. Our experiments were by no means exhaustive; we tested all three algorithms on strings that were only 500 B and 1 KB in length. These strings were all fairly similar, with EDM values ranging from 5 to 100. More tests should be done to compare the SW, q-gram, and ESP method similarity measures for input strings of varying sizes and degrees of similarity. Further testing may show different relationships among the outputs of the three algorithms.

In addition to computational biology, many other possible applications of the EDM similarity metric and the ESP method can be explored. Having shown that the ESP method is a good algorithm for computing the EDM between two strings, the algorithm can be used to test the relevance of the EDM in different contexts. The ESP method itself can also be applied to solving other string similarity problems, in addition to computing the distance between two strings. Cormode and Muthukrishnan outline a broad range of applications for the algorithm, such as string indexing and finding string outliers, for this new algorithm [2]. In addition, we propose that the ESP method can be used in the field of computer security to help decide if an unknown program is a new virus. If the code for the potential virus and existing viruses are represented as strings, finding the EDM between the potential attack and existing viruses provides an appropriate measure of similarity. Substring moves between the two strings represent common subroutines. Low EDM values between the suspicious program and an existing virus indicate that it is very likely that the suspect code is in fact a virus.

References

[1] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *J. Mol. Biol.,* 215(3):403-10, 1990.

[2] G. Cormode and S. Muthukrishnan. The string edit distance matching problem with moves. *Proceedings of the 13th Annual Symposium on Discrete Algorithms,* 667-676, 2002.

[3] D. Gusfield. *Algorithms on strings, trees and sequences: computer science and computational biology.* Cambridge University Press, Cambridge, 1997.

[4] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys,* 33:31-88, 2001.

[5] T. F. Smith and M. S. Waterman. Identification of Common Molecular Subsequences. *J. Mol. Biol.,* 147:195-197, 1981.

[6] SimMetrics Open Source Library. http://www.dcs.shef.ac.uk/ sam/simmetrics.html. University of Sheffield, 2005.

[7] E. Ukkonen. Approximate string-matching with q-grams and maximal matches. *Theoretical Computer Science,* 92:191–211, 1992.