String Editing with Substitution, Insertion, Deletion, Squashing and Expansion Operations

B. J. Oommen

SCS-TR-194, SEPTEMBER 1991

School of Computer Science, Carleton University
Ottawa, Canada, KIS 5B6

STRING EDITING WITH SUBSTITUTION, INSERTION, DELETION, SQUASHING AND EXPANSION OPERATIONS*

B. John Oommen
School of Computer Science
Carleton University
Ottawa ; CANADA : K1S 5B6

ABSTRACT

Let X and Y be any two strings of finite length. The problem of transforming X to Y using the edit operations of substitution, deletion and insertion has been extensively studied in the literature [1, 2, 6-11, 13, 15, 16, 18-21]. The problem can be solved in quadratic time if the edit operations are extended to include the operation of transposition of ***adjacent*** characters [24], but is generally reckoned to be NP-Hard if the transposition of arbitrary characters is permitted. In this paper we consider the problem of editing X to Y when the set of edit operations is extended to include the squashing and expansion operations. Whereas in the squashing operation two (or more) contiguous characters of X can be transformed into a single character of Y, in the expansion operation a single character in X may be expanded into two or more contiguous characters of Y. These operations are typically found in the recognition of cursive script. A quadratic time solution to the problem has been presented. We conjecture that this solution is optimal for the infinite alphabet case. The technique to compute the sequence of edit operations is also presented.

*Partially supported by the Natural Sciences and Engineering Research Council of Canada.

I. INTRODUCTION

In the study of the comparison of text patterns, syllables, sound phonemes and biological macromolecules a question that has interested researchers in that of quantifying the dissimilarity between two strings. A review of such distance measures and their applications is given by Hall and Dowling [2] and Peterson [16]. We recommend to the reader an excellent book edited by Sankoff and Kruskal [18] which discusses in detail the problem of sequence comparison.

The most promising of all distance measures which compare two strings seems to be the one that relates them using various edit operations [18, pp.37-39]. The edit operations most frequently considered are the deletion of a symbol, the insertion of a symbol, and the substitution of one symbol for another [2, 5-11, 13, 15, 16, 18-20]. This distance, referred to as the Generalized Levenshtein Distance(GLD), between two strings is defined as the minimum sum of the edit distances associated with the edit operations required to transform one string to another. Apart from being a suitable index for comparing two strings, this measure is closely related to other numerical and non-numerical measures that involve the strings, such as the Longest Common Subsequence (LCS) [3-6, 12, 14] and the shortest common supersequence [12].

Various algorithms to compute this distance have been proposed. The most straightforward algorithm to achieve this has been independently published by many authors (See [18]) but the algorithm is generally associated with Wagner and Fischer [19]. A faster algorithm for the finite alphabet case has been invented by Masek and Paterson [13]. For the infinite-alphabet case it has been shown that Wagner and Fischer's algorithm is optimal [20]. Related to these algorithms are the ones proposed to compute the LCS of two strings by Hirshberg [3, 4], Hunt and Szymanski [5], and Needleman and Wunsch[14]. Bounds on the complexity of the LCS problem have been given by Aho *et. al.* [1]. In this context it is noteworthy that techniques similar to those described in [19] have been used in the correction of noisy strings, substrings and subsequences [2, 8, 10, 11, 16, 21, 22] both when the transmission channel is unrestricted, and when the channel is restricted to not making consecutive errors [9]. In this case the dictionary is represented as a trie.

All of the above mentioned algorithms consider the editing of one string, say X, to transform it to Y, with the edit process being absolutely unconstrained. Sankoff [17] pioneered the study of constrained string editing. His algorithm is an LCS algorithm which involves a specialized constraint that has its application in the comparison of amino acids sequences. Later in [25] Oommen presented the first known solution to the problem of editing X to Y subject to any general edit constraint which could be arbitrarily complex, so long as it is specified in terms of the number and type of edit operations to be included in the optimal edit transformation. Using the fundamental principles of constrained string editing and considering the properties of a noisy channel which can garble transmitted sequences the first algorithm to correct noisy **subsequences** was presented in

[21]. The accuracy of the algorithm to correct long subsequences with low "signal-to-noise" ratios was demonstrated in [21]. This algorithm too has been recently improved [22]. The application of constrained editing to encryption has also been claimed [23].

It must be observed that in all the above mentioned results the types of edit operations (or garbling operations if the transmission channel is modelled as a garbling mechanism) are the well known substitution, insertion and deletion operations. To our knowledge there is only one paper in the literature which studies the case when the set of edit operations is expanded [24]. In [24], apart from the latter three operations, the set of edit operations has been expanded to also include the transposition operation. In this case the problem can be solved in quadratic time if the edit operations are extended to include the operation of transposition of ***adjacent*** characters [24]. However, it is generally reckoned to be NP-Hard if the transposition of arbitrary characters is permitted (i.e, if the interchanging of x_i and x_j is considered as an edit operation whenever $i \neq j$).

In this paper we consider the problem of editing X to Y when the set of edit operations is extended to include the squashing and expansion operations. Whereas in the squashing operation two (or more) contiguous characters of X can be transformed into a single character of Y, in the expansion operation a single character in X may be expanded into two or more contiguous characters of Y. Thus, in effect, we are now expanding the concept of string editing to consider cases in which two (or more) contiguous characters of one string can be transformed into a single character of the other, or vice versa. From a text processing and formal communication theory point of view, these operations are especially important when the transmission channel is a "keyboard" with bouncy keys. However, we also believe that these extensions will be more applicable in the recognition of cursive script. This is because, in cursive script processing, various squashing and expansion scenarios are encountered : It is not uncommon for the letter "y" to be mistaken for the the combination of the characters "ij" and vice versa, and similarly, it is not uncommon for the letter "w" to be mistaken for the the combination of the characters among which are "ui" or "iu" and vice versa. Similar examples of squashing and expansion are encountered in applications when the demarcation between the boundaries of the individual symbols is not apparent as in the case of the recognition of phoneme sequences [18].

In this paper we present a quadratic time solution to the problem of string editing for the expanded set of edit operations. We conjecture that this solution is optimal for the infinite alphabet case. The technique to compute the optimal sequence of edit operations is also presented. Also, throughout this paper we shall consider the squashing and expansion operations to be such that two contiguous symbols of one string can be transformed into a single symbol of the second. The case when multiple contiguous symbols (more than two) of one string can be transformed into a single symbol of the other can be easily generalized from the principles described here.

I.1 NOTATION

Let **A** be any finite alphabet, and $\mathbf{A}^*$ be the set of strings over **A**. θ is the null symbol, where $\theta \notin \mathbf{A}$. Let $\tilde{\mathbf{A}} = \mathbf{A} \cup \{\theta\}$. $\tilde{\mathbf{A}}$ is referred to as the *Appended Alphabet*. A string $X \in \mathbf{A}^*$ of the form $X = x_1 \ldots x_N$, where each $x_i \in \mathbf{A}$, is said to be of length $|X| = N$. Its prefix of length i will be written as X_i, for $1 \leq i \leq N$. Uppercase symbols represent strings, and lower case symbols, elements of the alphabet under consideration.

Let Z' be any element in $\tilde{\mathbf{A}}^*$, the set of strings over $\tilde{\mathbf{A}}$. The *Compression Operator* $\mathbb{C}$ is a mapping from $\tilde{\mathbf{A}}^*$ to $\mathbf{A}^*$: $\mathbb{C}(Z')$ is Z' with all occurrences of the symbol θ removed from Z'. Note that $\mathbb{C}$ preserves the order of the non-θ symbols in Z'. For example, if $Z' = f\theta o\theta r$, $\mathbb{C}(Z') = for$.

Throughout this paper we shall distinguish between the null string and the null symbol. Whereas μ will be used to represent the empty string, θ will be used to represent the null symbol.

I.2 THE ELEMENTARY EDIT DISTANCES

As mentioned earlier, throughout this paper we shall only consider the case when the squashing and expansion operations involves transforming two contiguous symbols of one string into a single symbol of the other. Bearing this in mind we now define the costs associated with the individual edit operations. If $\mathbf{R}^+$, is the set of nonnegative real numbers, we define the elementary edit distances using five elementary functions $d_s(.,.)$, $d_i(.)$, $d_e(.,.)$, $d_{sq}(.,.)$ and $d_{ex}(.,.)$ defined as :

(i) $d_s(p,q)$ is a map from $\mathbf{A} \times \mathbf{A} \rightarrow \mathbf{R}^+$ and is called the Substitution Map. In particular, $d_s(a,b)$ is the distance associated with substituting b for a, $a,b \in \mathbf{A}$. For all $a \in \mathbf{A}$, $d_s(a,a)$ is generally assigned the value zero, although this is not mandatory.

(ii) $d_i(.)$ is a map from $\mathbf{A} \rightarrow \mathbf{R}^+$ and is called the Insertion Map. The quantity $d_i(a)$ is the distance associated with inserting the symbol $a \in \mathbf{A}$.

(iii) $d_e(.)$ is a map from $\mathbf{A} \rightarrow \mathbf{R}^+$ and is called the Deletion or Erasure Map. The quantity $d_e(a)$ is the distance associated with deleting (or erasing) the symbol $a \in \mathbf{A}$.

(iv) $d_{sq}(.,.)$ is a map from $\mathbf{A}^2 \times \mathbf{A} \rightarrow \mathbf{R}^+$ called the Squashing Map. The quantity $d_{sq}(ab,c)$ is the distance associated with squashing the string ab into a single character c, where $a,b,c \in \mathbf{A}$.

(v) $d_{ex}(.,.)$ is a map from $\mathbf{A} \times \mathbf{A}^2 \rightarrow \mathbf{R}^+$ called the Expansion Map. The quantity $d_{ex}(c,ab)$ is the distance associated with expanding the character c into the string ab, where $a,b,c \in \mathbf{A}$.

I.2 THE SET OF EDIT POSSIBILITIES $\Gamma_{X,Y}$

For every pair (X,Y), $X, Y \in \mathbf{A}^*$, the finite set $\Gamma_{X,Y}$ is defined by means of the compression operator $\mathbb{C}$, as a subset of $\tilde{\mathbf{A}}^* \times \tilde{\mathbf{A}}^*$ as :

$$\Gamma_{X,Y} = \{ (X',Y') \mid (X',Y') \in \tilde{A}^* \times \tilde{A}^* \text{ , and each } (X',Y') \text{ obeys}$$

(i) $\mathfrak{C}(X') = X$, $\mathfrak{C}(Y') = Y$,

(ii) $|X'| = |Y'|$,

(iii) For all $1 \leq i \leq |X'|$, it is not the case that $x'_i = y'_i = \theta$ $\}$. (1)

By definition, if $(X',Y') \in \Gamma_{X,Y}$ then $\text{Max}\ (|X|, |Y|) \leq |X'| = |Y'| \leq |X| + |Y|$.

Viewed from the perspective of the three elementary operations the meaning of the pair $(X',Y') \in \Gamma_{X,Y}$ is interesting. Indeed, each element in $\Gamma_{X,Y}$ corresponds to one way of editing X into Y, using the edit operations of substitution, deletion, and insertion. The edit operations themselves are specified for all $1 \leq i \leq |X'|$ by (x'_i, y'_i), which represents the transformation of x'_i to y'_i. The cases below consider the three edit operations individually:

(i) If $x'_i \in A$ and $y'_i \in A$, it represents the substitution of y'_i for x'_i.

(ii) If $x'_i \in A$ and $y'_i = \theta$, it represents the deletion of x'_i.

(iii) If $x'_i = \theta$ and $y'_i \in A$, it represents the insertion of y'_i.

$\Gamma_{X,Y}$ is an exhaustive enumeration of the set of all the ways by which X can be edited to Y using these three elementary edit operations. However, on examining the individual elements of $\Gamma_{X,Y}$ it becomes clear that each pair contains more information than that. Indeed, ***in each pair,*** there is also information about the various ways by which X can be edited to Y even if the set of edit operations is grown so as to include squashing and expansion. An example would help clarify this.

Consider the case when $(X',Y') = (ab\theta, cde)$. Apart from representing the edit operations described in (i)-(iii) above, it also represents the substitution of 'a' by 'c' and the expansion of 'b' by the character sequence 'de'.

It is interesting to note that the set $\Gamma_{X,Y}$ maps X to Y using edit operations without destroying the order of occurrence of the symbols in X and Y. Also note that the number of elements in the set $\Gamma_{X,Y}$ is given by :

$$|\Gamma_{X,Y}| = \sum_{(k=\max[0,|Y| - |X|])}^{|Y|} \frac{(|X| + k)!}{k!(|Y|-k)!(|X|-|Y|+k)!}$$

Note that $|\Gamma_{X,Y}|$ depends only on $|X|$ and $|Y|$, and not on the actual strings X and Y themselves. Further, observe that the transformation of a symbol $a \in A$ to itself is also considered as an operation in the arbitrary pair $(X',Y') \in \Gamma_{X,Y}$. Finally, it is also interesting to note that the same set of edit operations can be represented by multiple elements in $\Gamma_{X,Y}$. This duplication serves as a powerful tool in the proofs of various analytic results [6, 7, 9, 10, 21, 25].

EXAMPLE I. Let X= f and Y = go. Then,

$$\Gamma_{X,Y} = \{ (f\theta,go), (\theta f,go), (f\theta\theta,\theta go), (\theta f\theta,g\theta o), (\theta\theta f,go\theta) \}.$$

In particular the pair (θf,go) represents the edit operations of inserting the 'g' and replacing the 'f' by an 'o'. It also represents the expansion of 'f' to 'go'. ♦♦♦

Since the Edit Distance between X and Y is the minimum of the sum of the edit distances associated with edit operations required to change X to Y, this distance, written as D(X,Y), has the expression :

$$D(X,Y) = \min_{((X',Y')\in \Gamma_{X,Y})} \sum_{i=1}^{|J'|} [\text{ Distances Associated with the Operations in } (X', Y')] , \quad (2)$$

where, (X',Y') represents J' possible edit operations.

II. THE COMPUTATION OF THE EDIT DISTANCE

Let D(X,Y) be the edit distance associated with transforming X to Y with the edit operations of substitution, insertion, deletion, squashing and expansion. In this section we shall describe how D(.,.,) can be computed. To achieve this, we shall first derive the properties of D(X,Y) which can be derived recursively in terms of the corresponding quantities defined in terms of the prefixes of X and Y, (X_i and X_j respectively) with the assumption that D(μ, μ) is zero. Indeed, in this case we can show the following elementary obvious results which can be proved in the identical way in which the analogous results are proved for the edit distance which entails only the three elementary edit operations [6, 7, 9, 10, 19, 21, 25]. They can also be proved by straightforward enumeration. The proofs of these results are omitted here in the interest of brevity.

LEMMA 0a.

Let $X = X_i = x_1...x_i$ be the prefix of X and Y = μ, the null string. Then, $D(X_i, \mu)$ obeys :

$$D(X_i, \mu) = D(X_{i-1}, \mu) + d_e(x_i).$$ ♦♦♦

LEMMA 0b.

Let X = μ, and $Y_j = y_1...y_j$ be the prefix of Y. Then, $D(\mu, Y_j)$ obeys :

$$D(\mu, Y_j) = D(\mu, Y_{j-1}) + d_i(y_j).$$ ♦♦♦

LEMMA 0c.

Let $X = x_1$ and $Y = y_1$. Then, D(X, Y) obeys :

$$D(X, Y) = \textbf{Min} \left[D(\mu, Y) + d_e(x_1), \quad D(X, \mu) + d_i(y_1), \quad d_s(x_1, y_1) \right]. \qquad \blacklozenge\blacklozenge\blacklozenge$$

LEMMA 0d.

Let $X_i = x_1 \ldots x_i$ with $i \geq 2$ be the prefix of X, and $Y = y_1$, the string consisting of the first character of Y. Then, $D(X_i, Y)$ obeys :

$$\begin{aligned} D(X_i, Y) = \textbf{Min} \Big[& D(X_{i-1}, Y) + d_e(x_i), \\ & D(X_i, \mu) + d_i(y_1), \\ & D(X_{i-1}, \mu) + d_s(x_i, y_1), \\ & D(X_{i-2}, \mu) + d_{sq}(x_{i-1}x_i, y_1) \Big]. \end{aligned} \qquad \blacklozenge\blacklozenge\blacklozenge$$

LEMMA 0e.

Let $X = x_1$ the string consisting of the first character of X and $Y = y_1 \ldots y_j$ be the prefix of Y with $j \geq 2$. Then, $D(X, Y_j)$ obeys :

$$\begin{aligned} D(X, Y_j) = \textbf{Min} \Big[& D(\mu, Y_j) + d_e(x_1), \\ & D(X, Y_{j-1}) + d_i(y_j), \\ & D(\mu, Y_{j-1}) + d_s(x_1, y_j) \\ & D(\mu, Y_{j-2}) + d_{ex}(x_1, y_{j-1}y_j) \Big]. \end{aligned} \qquad \blacklozenge\blacklozenge\blacklozenge$$

We shall now state and prove the main result of our paper.

THEOREM I.

Let $X_i = x_1 \ldots x_i$ and $Y_j = y_1 \ldots y_j$ with $i, j \geq 2$. Also, let $D(X_i, Y_j)$ be the edit distance associated with the transforming X_i to Y_j with the edit operations of substitution, insertion, deletion, squashing and expansion. Then, the following is true :

$$\begin{aligned} D(X_i, Y_j) = \textbf{Min} \Big[& D(X_{i-1}, Y_j) + d_e(x_i), \\ & D(X_i, Y_{j-1}) + d_i(y_j), \\ & D(X_{i-1}, Y_{j-1}) + d_s(x_i, y_j), \\ & D(X_{i-2}, Y_{j-1}) + d_{sq}(x_{i-1}x_i, y_j), \\ & D(X_{i-1}, Y_{j-2}) + d_{ex}(x_i, y_{j-1}y_j) \Big]. \end{aligned}$$

Proof : The proof of the theorem is quite involved and is included in the Appendix. $\blacklozenge\blacklozenge\blacklozenge$

Remarks

A note about the *modus operandus* of the proof of Theorem I is not out of place. From a naive perspective it is possible to consider the techniques applied here as mere applications of dynamic-programming to extensions of a problem that has been widely discussed. This is, in fact, not the case. There is a very fine point in which our proof differs from the proofs currently described in the literature. The fundamental difference is that in the current proof, whenever the set over which the minimization is achieved is grown, it is not merely a single optimization scenario which is encountered. Thus in Case 3 of the proof in the Appendix, there are two possible scenarios by which the minimization can be achieved. The first of the scenarios (See Eq. 10) appears again in the processing of Case 7 (See Eq. 17) and in the processing of Case 9 (See Eq. 21). The second (See Eq. 12) appears again in the processing of Case 5 (See Eq. 15). Thus the **same five terms** appear in their different combinations in various cases encountered in the minimization process. This makes the proof more interesting and a trifle more "intriguing". This is reminiscent of a control system in which various outputs are computed in terms of the **same** state variables by just using different "Output Functions" to evaluate the various combinations of the variables themselves.

III. THE COMPUTATION OF D(X,Y)

To compute D(X,Y) we make use of the fact that this index has the recursive properties given above. The idea is essentially one of computing the distance $D(X_i, Y_j)$ between the prefices of X and Y. The computation of the distances has to be done in a schematic manner, so that any quantity $D(X_i, Y_j)$ is computed before its value is required in any further computation. Just as in the case of the previous string edit algorithms [3-6, 10, 11, 13, 15, 18, 19, 21, 25] this can be actually done in a straightforward manner by tracing the underlying graph, commonly referred to as a ***trellis*** and maintaining an array Z(i,j) defined for all $0 \leq i \leq N$ and $0 \leq j \leq M$ when $|X| = N$ and $|Y| = M$. The quantity Z(i,j) is nothing but $D(X_i, Y_j)$. We will discuss the properties of the our particular trellis subsequently.

Initially the weight associated with the origin, Z(0,0), is assigned the value zero, and the weights associated with the vertices on the axes are evaluated. Thus, Z(i,0) and Z(0,j) are computed using Lemmas 0a and 0b for all $1 \leq i \leq N$ and $1 \leq j \leq M$. The value for Z(1,1) is then computed as a special computation outside any loop. Subsequently, the values for the lines i=1 and j=1 are traversed, and the distances associated with the vertices on these lines are computed using the previously computed values and Lemmas 0d and 0e. Finally, the weights corresponding to strict "interior" values (i.e, whenever $i, j > 1$) of the variables are computed.

The algorithm to compute Z(.,.) is given below.

ALGORITHM GeneralizedDistance

Input : The strings $X = x_1 \ldots x_N$ and $Y = y_1 \ldots y_M$, and the set of elementary edit distances defined using the five elementary functions $d_s(.,.)$, $d_i(.)$, $d_e(.)$, $d_{sq}(.,.)$ and $d_{ex}(.,.)$.

Output : The distance D(X, Y) associated with transforming X to Y using the five edit operations of substitution, insertion, deletion, squashing and expansion.

Method :

$Z(0, 0) \leftarrow 0$

For $i \leftarrow 1$ **to** N **Do**

  $Z(i, 0) \leftarrow Z(i-1, 0) + d_e(x_i)$

For $j \leftarrow 1$ **to** M **Do**

  $Z(0, j) \leftarrow Z(0, j-1) + d_i(y_j)$

$Z(1, 1) \leftarrow$ **Min** $[\, Z(0, 1) + d_e(x_1),\ Z(1, 0) + d_i(y_1),\ Z(0, 0) + d_s(x_1, y_1) \,]$

For $i \leftarrow 2$ **to** N **Do**

  $Z(i, 1) \leftarrow$ **Min** $[\, Z(i-1, 1) + d_e(x_i),\quad Z(i, 0) + d_i(y_1),$
  $Z(i-1, 0) + d_s(x_i, y_1),\quad Z(i-2, 0) + d_{sq}(x_{i-1}x_i\ ,\ y_1) \,]$

For $j \leftarrow 2$ **to** M **Do**

  $Z(1, j) \leftarrow$ **Min** $[\, Z(0, j) + d_e(x_1),\quad Z(1, j-1) + d_i(y_j),$
  $Z(0, j-1) + d_s(x_1, y_j),\quad Z(0, j-2) + d_{ex}(x_1, y_{j-1}y_j) \,]$

For $i \leftarrow 2$ **to** N **do**

  For $j \leftarrow 2$ **to** M **do**

    $Z(i,j) \leftarrow$ **Min** $[Z(i-1, j) + d_e(x_i),\quad Z(i, j-1) + d_i(y_j),$
    $Z(i-1, j-1) + d_s(x_i, y_j),\quad Z(i-2, j-1) + d_{sq}(x_{i-1}x_i\ ,\ y_j),$
    $Z(i-1, j-2) + d_{ex}(x_i\ ,\ y_{j-1}y_j) \,]$

$D(X,Y) \leftarrow Z(N, M)$

END ALGORITHM GeneralizedDistance

Remarks

The computational complexity of algorithms involving the comparison of two strings is conveniently given by the number of symbol comparisons required by the algorithm [1, 20]. In this case, the number of symbol comparisons (or more relevantly the number of invocations of the functions $d_e(.)$, $d_i(.)$, $d_s(.,.)$, $d_{sq}(..,.)$, $d_{ex}(.,..)$) required by ALGORITHM GeneralizedDistance is clearly quadratic. Note that in the interior of the main loop, we will need at most five additions and the computation of the minimum of a fixed (at most five) quantities. We conjecture that this solution is optimal for the infinite alphabet case -- the basis for this conjecture being arguments similar to those discussed in [20] which are yet to be formalized.

We shall illustrate the computation of D(X,Y) by describing the underlying graph that is being traversed and by going through the steps of the algorithm with an example.

III.1 Graphical Representation of the Algorithm and an Example

In the computation of various string similarity and dissimilarity measures, the underlying graph that has to be traversed is commonly called a *trellis*. This trellis is two dimensional in the case of the GLD [2, 6, 13, 15, 18, 19], the Length of the LCS [3-6, 12, 18] and the Length of the Shortest Common Supersequence [12]. Indeed, the same trellis can be traversed using various set operators to yield the Set of the LCSs and the Set of the Shortest Common Supersequences [6]. The trellis becomes essentially three dimensional when one has to compute string probabilities [10], constrained edit distances [25] and correct noisy subsequences [21, 22]. Amazingly enough, although the trellis itself is two dimensional in the former examples, because the graphs are **cycle-free** they can be represented and traversed by merely maintaining single-dimensional structures [4]. Similarly, in the cases of computing string probabilities [10] and constrained edit distances [25] and correcting noisy subsequences [21,22] , although the trellises are three dimensional, they can be represented and traversed using only two dimensional arrays.

Amazingly enough, even though the set of edit operations has been expanded from being merely substitution, insertion and deletion to include squashing and expansion the fundamental properties of the underlying graph traversed remains the same. In this case, the graph G = (V, E), where, V and E are the set of vertices and edges respectively described below :

$$V = \{ \langle i, j \rangle \mid 0 \le i \le N,\ 0 \le i \le M \}, \text{ and,}$$

$$\begin{aligned} E = {} & \{ (\langle i, j \rangle, \langle i+1, j \rangle) \mid 0 \le i \le N-1,\ 0 \le j \le M \} \cup \\ & \{ (\langle i, j \rangle, \langle i, j+1 \rangle) \mid 0 \le i \le N,\ 0 \le j \le M-1 \} \cup \\ & \{ (\langle i, j \rangle, \langle i+1, j+1 \rangle) \mid 0 \le i \le N-1,\ 0 \le j \le M-1 \} \cup \\ & \{ (\langle i, j \rangle, \langle i+2, j+1 \rangle) \mid 0 \le i \le N-2,\ 0 \le j \le M-1 \} \cup \\ & \{ (\langle i, j \rangle, \langle i+1, j+2 \rangle) \mid 0 \le i \le N-1,\ 0 \le j \le M-2 \}. \end{aligned}$$

The graph essentially has arcs whenever a single edit operation can be applied. Indeed, the algorithm describes an efficient quadratic time scheme by which the trellis can be traversed.

For the sake of clarity a pictorial representation of the graph is given in Figure I.

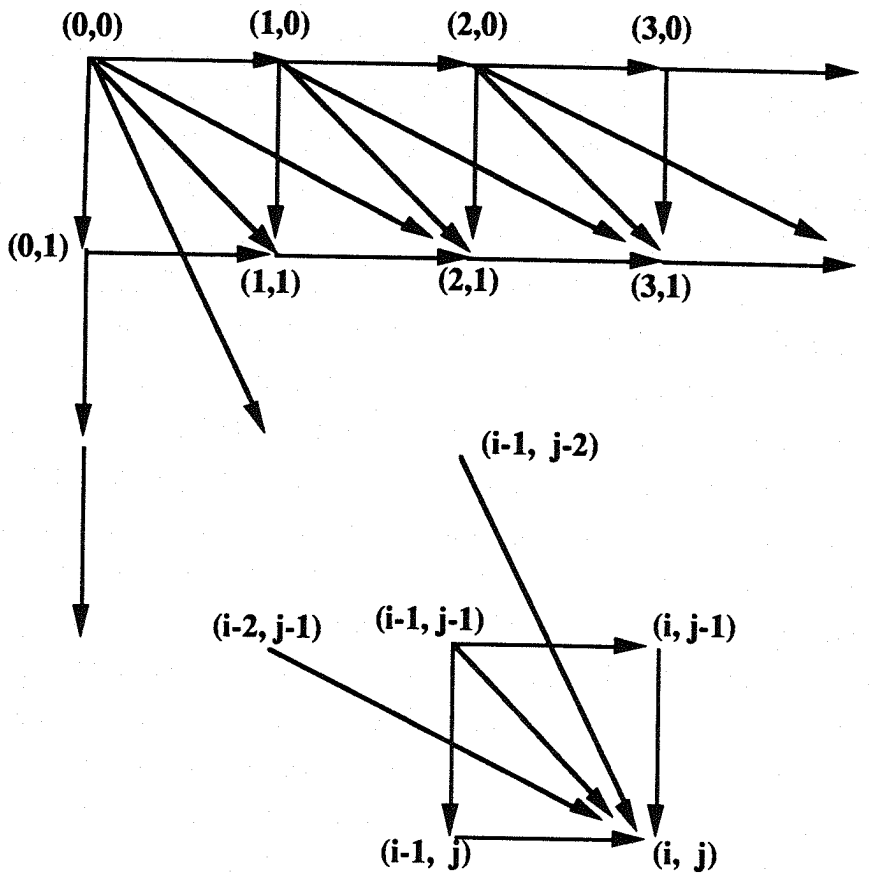

Figure I : The Trellis that has to be traversed in order to compute D(X,Y). Note that the only edges terminating at (i, j) are those starting at (i-2, j-1), (i-1, j-1), (i, j-1), (i-1,j) and (i-1,j-2)

EXAMPLE II.

Let X=ag and Y = bcf. Let us suppose we want to edit X to Y. We shall now follow through the computation of D(ag,bcf) using Algorithm GeneralizedDistance. To begin with, the weight associated with the origin is initialized to be the value zero. The i and j axes are first traversed:

$Z(1, 0) \leftarrow d_e(a)$ $\quad Z(2, 0) \leftarrow d_e(a) + d_e(g)$

$Z(0, 1) \leftarrow d_i(b)$ $\quad Z(0, 2) \leftarrow d_i(b) + d_i(c)$ $\quad Z(0, 3) \leftarrow d_i(b) + d_i(c) + d_i(f)$

The value for Z(1, 1) is next assigned outside of the main loops.

$Z(1, 1) \leftarrow$ **Min** $[\, Z(0, 1) + d_e(a),\ Z(1, 0) + d_i(b),\ Z(0, 0) + d_s(a,b) \,]$

The lines for i = 1 and j = 1 are then traversed:

$Z(2, 1) \leftarrow$ **Min** [$Z(1, 1) + d_e(g)$, $Z(2, 0) + d_i(b)$,
$Z(1, 0) + d_s(g,b)$, $Z(0, 0) + d_{sq}(ag,b)$]

$Z(1, 2) \leftarrow$ **Min** [$Z(0, 2) + d_e(a)$, $Z(1, 1) + d_i(c)$,
$Z(0, 1) + d_s(a,c)$, $Z(0, 0) + d_{ex}(a , bc)$]

$Z(1, 3) \leftarrow$ **Min** [$Z(0, 3) + d_e(a)$, $Z(1, 2) + d_i(f)$,
$Z(0, 2) + d_s(a,f)$, $Z(0, 1) + d_{ex}(a , cf)$]

The strict interior of the trellis for values when both $i, j \geq 2$ are then traversed :

$Z(2, 2) \leftarrow$ **Min** [$Z(1, 2) + d_e(g)$, $Z(2, 1) + d_i(c)$,
$Z(1, 1) + d_s(g, c)$, $Z(0, 1) + d_{sq}(ag, c)$,
$Z(1, 0) + d_{ex}(g , bc)$]

$Z(2, 3) \leftarrow$ **Min** [$Z(1, 3) + d_e(g)$, $Z(2, 2) + d_i(f)$,
$Z(1, 2) + d_s(g, f)$, $Z(0, 2) + d_{sq}(ag, f)$,
$Z(1, 1) + d_{ex}(g, cf)$]

Indeed, the required edit distance D(ag, bcf) is Z(2, 3). ♦♦♦

III.2 Computing the Best Edit Sequence

Just as in all the edit processes studied in the literature, the traversal of the trellis not only yields the information about the distance between the strings X and Y. By virtue of the way the trellis has been traversed the distances between the prefixes of the strings has also been maintained in the process of computation, and thus, the array Z contains information which can be used to compute the best edit sequence which yields the optimal edit distance. This is done by backtracking through the trellis from the array element (N,M) in the reverse direction of the arrows so as to reach the origin, always remembering the path that was used to reach the node which is currently being visited. Thus the actual sequence of edit operations can be printed out in the reverse order.

The technique is well known in dynamic-programming problems and has been used extensively for edit sequences [6, 13, 15, 18, 19] and the LCS problem [3-6, 12, 14]. Without further comment we now present Algorithm ProduceEditOperations, which has as its input the array Z(.,.). To simplify the backtracking, we exclude the possibility of encountering negative values of i and j by rendering Z(.,.) infinite whenever any of the indices are negative.

ALGORITHM ProduceEditOperations

Input : The strings $X = x_1...x_N$ and $Y = y_1...y_M$, the set of elementary edit distances defined as in Algorithm Generalized Distance and the array Z.

Output : The best edit sequence that can transform X to Y using the edit operations of substitution, insertion, deletion, squashing and expansion.

Method :

```
Define Z(i,j ) ← ∞ whenever i < 0 or j < 0.
i ← N
j ← M
While (i ≠ 0 or j ≠ 0 ) Do
    If (Z(i, j) = Z(i-1, j-1) + d_s(x_i, y_j)) Then
        Print ("Substitute" x_i "by" y_j)
        i ← i - 1
        j ← j - 1
    Else
        If (Z(i, j) = Z(i, j-1) + d_i(y_j)) Then
            Print ("Insert" y_j)
            j ← j - 1
        Else
            If (Z(i, j) = Z(i-1, j) + d_e(x_i)) Then
                Print ("Delete" x_i)
                i ← i - 1
            Else
                If (Z(i, j) = Z(i-2, j-1) + d_sq(x_{i-1}x_i, y_j)) Then
                    Print ("Squash" x_{i-1}x_i "into" y_j)
                    i ← i - 2
                    j ← j - 1
                Else
                    If (Z(i, j) = Z(i-1, j-2) + d_ex(x_i, y_{j-1}y_j)) Then
                        Print ("Expand" x_i "into" y_{j-1}y_j)
                        i ← i - 1
                        j ← j - 2
                    EndIf
                EndIf
            EndIf
        EndIf
    EndIf
EndWhile
END ALGORITHM ProduceEditOperations
```

Obviously Algorithm ProduceEditOperations is performed in O(max(M,N)) time.

Remarks

(i) Using the concepts introduced here we can also conceive of measures related to the LCS of two strings in which a substring of one string can be considered to be a single character of the other, as in the case when a single amino acid in a molecular sequence can be decomposed to be a sequence of two (or more) compounds each of which can be represented by single symbols. These measures are indeed Generalized LCS metrics and could possibly be used to study the similarity between biological molecules in which the mutating operations of clustering and fragmentation are considered.

(ii) Throughout this paper we have only considered the case when the types of expansion and squashing errors involve transformations from a single character in one string to two characters in the second. It is easy to visualize the generalization of this when the number of characters involved in the squash/expand operations is a constant K, where $K \geq 2$. The resulting trellis then would have to be traversed in essentially the same way, except that at every internal node, the minimization would involve the computation of 2K+1 quantities. For example, if K is 3, the corresponding minimization in the interior of the trellis would involve the following expression :

$$
\begin{aligned}
Z(i,j) \leftarrow \textbf{Min}\ [\quad & Z(i-1, j-1) + d_s(x_i,y_j), && Z(i, j-1) + d_i(y_j), \\
& Z(i-1, j) + d_e(x_i), && Z(i-2, j-1) + d_{sq}(x_{i-1}x_i , y_j), \\
& Z(i-1, j-2) + d_{ex}(x_i , y_{j-1}y_j) && Z(i-3, j-1) + d_{sq}(x_{i-2}x_{i-1}x_i , y_j), \\
& Z(i-1, j-3) + d_{ex}(x_i , y_{j-2}y_{j-1}y_j)].
\end{aligned}
$$

The algorithm would still be quadratic in the lengths of the strings as long as K is independent of M and N, which is not an unreasonable assumption especially as the types of errors that are caused in a channel are typically not functions of the strings transmitted themselves.

(iii) The use of the GLD to perform the automatic correction of noisy strings [2, 9, 16, 18] and noisy substrings [8] has been discussed in the literature. In the latter results, the errors which were present in the noisy strings were deletions, insertions, and substitutions of individual characters. Furthermore, constrained edit distances have been used very powerfully to correct noisy subsequences [21, 22]. One problem that has been open is that of correcting erroneous strings and substrings in which the channel introduces expansion errors due to "bounce" and squashing errors due to the "coalescing" of characters/phonemes. Such syntactic problems are found in the recognition of cursive script. From our experience, we believe that the generalized edit distance defined and computed in this paper can be used to perform the correction of noisy strings which contain all the above types of errors. We are currently investigating this possibility.

IV CONCLUSIONS

Let X and Y be any two strings of finite length. The problem of transforming X to Y using the edit operations of substitution, deletion and insertion has been extensively studied in the literature [1, 2, 6-11, 13, 15, 16, 18-21]. In [24], the set of edit operations was expanded to include the transposition operation, and the problem was solved in quadratic time when the transposition of *adjacent* characters was permitted [24]. We believe that the problem is NP-Hard if the transposition of arbitrary characters in the strings is permitted. In this paper we have considered the problem of editing X to Y when the set of edit operations is extended to include the squashing and expansion operations. In the squashing operation two (or more) contiguous characters of X can be transformed into a single character of Y, and in the expansion operation a single character in X may be expanded into two or more contiguous characters of Y. The case when the number of operations involved in the squash/expansion is two has been thoroughly analyzed, and the case when this number is larger than two has been alluded to. A quadratic time solution to the problem has been presented. We conjecture that this solution is optimal for the infinite alphabet case. We also present the scheme to compute the sequence of edit operations.

The problem of constrained string editing using these five elementary edit operations of substitution, insertion, deletion, squashing and expansion of characters remains unsolved. Also the problem of obtaining bounds for the complexity of editing strings in this setting remain open.

We are currently investigating the use of these generalized edit distances in the pattern recognition of noisy strings and substrings in channels which actually do permit errors of this type.

ACKNOWLEDGEMENTS

I would like to thank John Andrusek and William Lee for their help in preparing the manuscript.

REFERENCES

1. A. V. Aho, D.S. Hirschberg, and J. D. Ulman, Bounds on the complexity of the longest common subsequence problem, *J. Assoc. Comput. Mach.*, 23:1-12 (1976).
2. P. A. V. Hall and G.R. Dowling, Approximate string matching, *Comput. Surveys*, 12:381-402 (1980).
3. D. S. Hirschberg, Algorithms for longest common subsequence problem, *J. Assoc. Comput. Mach.*, 24:664-675 (1977).
4. D. S. Hirschberg, A linear space algorithm for computing maximal common subsequences, *Comm. Assoc. Comput. Mach.*, 18:341-343 (1975).
5. J. W. Hunt and T. G. Szymanski, A fast algorithm for computing longest common subsequences, *Comm. Assoc. Comput. Mach.*, 20:350-353 (1977).
6. R. L. Kashyap and B. J. Oommen, A common basis for similarity and dissimilarity measures involving two strings, *Internat. J. Comput. Math.*, 13:17-40 (1983).
7. R. L. Kashyap and B. J. Oommen, Similarity measures for sets of strings, *Internat. J. Comput. Math.*, 13:95-104 (1983).

8. R. L. Kashyap and B. J. Oommen, The noisy substring matching problem, *IEEE Trans. Software Engg.*, SE-9:365-370 (1983).
9. R. L. Kashyap and B. J. Oommen, An effective algorithm for string correction using generalized edit distances -I. Description of the algorithm and its optimality, *Inform. Sci.*, 23(2):123-142 (1981).
10. R. L. Kashyap, and B. J. Oommen, String Correction Using Probabilistic Methods, *Pattern Recognition Letters*, 147-154 (1984).
11. A. Levenshtein, Binary codes capable of correcting deletions, insertions and reversals, *Soviet Phys. Dokl.*, 10:707-710 (1966).
12. D. Maier, The complexity of some problems on subsequences and supersequences, *J. Assoc. Comput. Mach.*, 25:322-336 (1978).
13. W. J. Masek and M. S. Paterson, A faster algorithm computing string edit distances, *J. Comput. System Sci.*, 20:18-31 (1980).
14. S. B. Needleman and C. D. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two protiens, *J. Mol. Biol.*, 443-453 (1970).
15. T. Okuda, E. Tanaka, and T. Kasai, A method of correction of garbled words based on the Levenshtein metric, *IEEE Trans. Comput.*, C-25:172-177 (1976).
16. J. L. Peterson, Computer programs for detecting and correcting spelling errors, *Comm. Assoc. Comput. Mach.*, 23:676-687 91980).
17. D. Sankoff, Matching sequences under deletion/insertion constraints, *Proc. Nat. Acad. Sci. U.S.A.*, 69:4-6 (1972).
18. D. Sankoff and J. B. Kruskal, *Time Warps,String Edits and Macromolecules: The Theory and practice of Sequence Comparison*, Addison-Wesley (1983).
19. R. A. Wagner and M. J. Fisher, The string to string correction problem, *J. Assoc. Comput. Mach.*, 21:168-173 (1974).
20. C. K. Wong and A. K. Chandra, Bounds for the string editing problem, *J. Assoc. Comput. Mach.*, 23:13-16 (1976).
21. B. J. Oommen, B.J. Recognition of Noisy Subsequences Using Constrained Edit Distances, *IEEE Trans. on Pattern Anal. and Mach. Intel.*, PAMI-9:676-685 (1987).
22. B. J. Oommen and E. T. Floyd, An Improved Algorithm for the Recognition of Noisy Subsequences, *Proceedings of the 1991 IASTED International Symposium on Artificial Intelligence Applications and Neural Networks*, Zurich, 145-147 (1991).
23. J. Golic and M. Mihaljevic, A noisy clock-controlled shift register cryptanalysis concept based on sequence comparison approach, *Proceedings of EUROCRYPT 90,* Aarhus, Denmark, 487-491 (1990).
24. R. Lowrance and R. A. Wagner, An extension of the string to string correction problem, *J. Assoc. Comput. Mach.*, 22:177-183 (1975).
25. Oommen, B.J., Constrained string editing, *Information Sciences*, 40: 267-284 (1987).

APPENDIX

Proof of Theorem I.

Let $X_i = x_1...x_i$ and $Y_j = y_1...y_j$ and $D(X_i, Y_j)$ be the edit distance associated with the transforming X_i to Y_j with the edit operations of substitution, insertion, deletion, squashing and expansion. Then, we are to prove that :

$$\begin{aligned} D(X_i, Y_j) = \textbf{Min} \Big[\quad & D(X_{i-1}, Y) + d_e(x_i), \\ & D(X_i, Y_{j-1}) + d_i(y_j), \\ & D(X_{i-1}, Y_{j-1}) + d_s(x_i, y_j), \\ & D(X_{i-2}, Y_{j-1}) + d_{sq}(x_{i-1}x_i, y_j), \\ & D(X_{i-1}, Y_{j-2}) + d_{ex}(x_i, y_{j-1}y_j) \Big]. \end{aligned}$$

The proof of the theorem is by induction on the lengths of the strings X_i and Y_j.

The basis step of the proof involves the proofs of the Lemmas 0a-0e. These, of course, should be proved in the interest of mathematical rigor, but they can be proved by straightforward enumeration and their proofs are simple extensions of the results already found in the literature. Hence they are omitted here in the interest of brevity. We merely proceed to the inductive step.

Let Γ_{X_i,Y_j} be the set of all ways by which X_i can be edited into Y_j defined as in (1) for X_i and Y_j. Consider the distance $D(X_i, Y_j)$ which has the expression :

$$D(X_i, Y_j) = \min_{((X_i', Y_j') \in \Gamma_{X,Y})} \sum_{i=1}^{|J'|} \Big[\text{ Distances Associated with Operations in } (X_i', Y_j') \Big], \tag{2}$$

where, $(X_i', Y_j') \in \Gamma_{X_i,Y_j}$ represents J' possible edit operations. Throughout this proof[1], we shall assume that the arbitrary element $(X_i', Y_j') \in \Gamma_{X_i,Y_j}$ is of length L and is of the form given as :

$X_i = x_{i1}'x_{i2}'...x_{iL}'$, and, $Y_j = y_{j1}'y_{j2}'...y_{jL}'$.

---

[1]This notation is not religiously correct. Indeed, the length of the arbitrary element in Γ_{X_i,Y_j} should be $L(X_i', Y_j')$. But this will make an already tedious notation even more cumbersome. We request the reader to permit us this breach in notation with the understanding that he remembers that L is dependent on the element itself.

We partition the set Γ_{X_i,Y_j} into nine mutually exclusive and exhaustive subsets:

$$\Gamma^1_{X_i,Y_j} = \{ (X_i', Y_j') \mid (X_i', Y_j') \in \Gamma_{X_i,Y_j}, \text{ with } x_{iL-1}' = \theta,\ x_{iL}' = \theta,\ y_{jL-1}' = y_{j-1},\ y_{jL}' = y_j \}$$

$$\Gamma^2_{X_i,Y_j} = \{ (X_i', Y_j') \mid (X_i', Y_j') \in \Gamma_{X_i,Y_j}, \text{ with } x_{iL-1}' = \theta,\ x_{iL}' = x_i,\ y_{jL-1}' = y_j,\ y_{jL}' = \theta \}$$

$$\Gamma^3_{X_i,Y_j} = \{ (X_i', Y_j') \mid (X_i', Y_j') \in \Gamma_{X_i,Y_j}, \text{ with } x_{iL-1}' = \theta,\ x_{iL}' = x_i,\ y_{jL-1}' = y_{j-1},\ y_{jL}' = y_j \}$$

$$\Gamma^4_{X_i,Y_j} = \{ (X_i', Y_j') \mid (X_i', Y_j') \in \Gamma_{X_i,Y_j}, \text{ with } x_{iL-1}' = x_i,\ x_{iL}' = \theta,\ y_{jL-1}' = \theta,\ y_{jL}' = y_j \}$$

$$\Gamma^5_{X_i,Y_j} = \{ (X_i', Y_j') \mid (X_i', Y_j') \in \Gamma_{X_i,Y_j}, \text{ with } x_{iL-1}' = x_i,\ x_{iL}' = \theta,\ y_{jL-1}' = y_{j-1},\ y_{jL}' = y_j \}$$

$$\Gamma^6_{X_i,Y_j} = \{ (X_i', Y_j') \mid (X_i', Y_j') \in \Gamma_{X_i,Y_j}, \text{ with } x_{iL-1}' = x_{i-1},\ x_{iL}' = x_i,\ y_{jL-1}' = \theta,\ y_{jL}' = \theta \}$$

$$\Gamma^7_{X_i,Y_j} = \{ (X_i', Y_j') \mid (X_i', Y_j') \in \Gamma_{X_i,Y_j}, \text{ with } x_{iL-1}' = x_{i-1},\ x_{iL}' = x_i,\ y_{jL-1}' = \theta,\ y_{jL}' = y_j \}$$

$$\Gamma^8_{X_i,Y_j} = \{ (X_i', Y_j') \mid (X_i', Y_j') \in \Gamma_{X_i,Y_j}, \text{ with } x_{iL-1}' = x_{i-1},\ x_{iL}' = x_i,\ y_{jL-1}' = y_j,\ y_{jL}' = \theta \}$$

$$\Gamma^9_{X_i,Y_j} = \{ (X_i', Y_j') \mid (X_i', Y_j') \in \Gamma_{X_i,Y_j}, \text{ with } x_{iL-1}' = x_{i-1},\ x_{iL}' = x_i,\ y_{jL-1}' = y_{j-1},\ y_{jL}' = y_j \}$$

By their definitions, we see that the above nine sets are mutually exclusive. Further, since the corresponding elements of (X_i', Y_j') cannot be θ simultaneously, it is clear that every pair in Γ_{X_i,Y_j} must be in one of the above sets. Hence these nine sets partition Γ_{X_i,Y_j}. Rewriting (2) we obtain :

$$D(X_i, Y_j) = \min_{1 \le k \le 9} \ \min_{((X_i', Y_j') \in \Gamma^k_{X_i,Y_j})} \ \sum_{i=1}^{|J'|} [\text{Distances Associated with Operations in } (X_i', Y_j')], \tag{3}$$

where, (X_i', Y_j') represents J' possible edit operations.

We shall now consider each of the nine terms in (3) individually.

Case 1.

Consider the first term in (3). In every pair in $\Gamma^1_{X_i,Y_j}$ we know that the last two elements of each string in the pair are :

$$x_{iL-1}' = \theta,\ x_{iL}' = \theta,\ y_{jL-1}' = y_{j-1},\ y_{jL}' = y_j.$$

Hence,

$$\min_{((X_i', Y_j') \in \Gamma^1_{X_i,Y_j})} \ \sum_{i=1}^{|J'|} [\text{Distances Associated with Operations in } (X_i', Y_j')]$$

$$= \min_{((X_i', Y_j') \in \Gamma^1_{X_i, Y_j})} \sum_{i=1}^{|J'|} [\text{Distances Associated with Operations in } (X'_{iL-1}, Y'_{jL-1})] + d_i(y'_{jL}). \tag{4}$$

For every element in $\Gamma^1_{X_i,Y_j}$ there is a unique element in $\Gamma_{X_i,Y_{j-1}}$ and vice versa, where $\Gamma_{X_i,Y_{j-1}}$ is the set of all ways by which X_i can be edited into Y_{j-1} defined as in (1) for X_i and Y_{j-1}. This unique element is obtained by merely reducing the length of the strings X'_i and Y'_j by unity. By the inductive hypothesis the first term in (4) is exactly $D(X_i, Y_{j-1})$. Since $y'_{jL} = y_j$, this tells that (4) simplifies to :

$$D(X_i, Y_{j-1}) + d_i(y_j). \tag{5}$$

Case 2.

Consider the second term in (3). In every pair in $\Gamma^2_{X_i,Y_j}$ we know that the last two elements of each string in the pair are :

$$x'_{iL-1} = \theta, \; x'_{iL} = x_i, \; y'_{jL-1} = y_j, \; y'_{jL} = \theta.$$

Hence,

$$\min_{((X_i', Y_j') \in \Gamma^2_{X_i, Y_j})} \sum_{i=1}^{|J'|} [\text{Distances Associated with Operations in } (X'_i, Y'_j)]$$

$$= \min_{((X_i', Y_j') \in \Gamma^2_{X_i, Y_j})} \sum_{i=1}^{|J'|} [\text{Distances Associated with Operations in } (X'_{iL-1}, Y'_{jL-1})] + d_e(x'_{iL}). \tag{6}$$

For every element in $\Gamma^2_{X_i,Y_j}$ there is a unique element in Γ_{X_{i-1},Y_j} and vice versa, where Γ_{X_{i-1},Y_j} is the set of all ways by which X_{i-1} can be edited into Y_j defined as in (1) for X_{i-1} and Y_j, and this unique element is again obtained by merely reducing the length of the strings X'_i and Y'_j by unity. Again, by the inductive hypothesis, the first term in (6) is $D(X_{i-1}, Y_j)$. Since $x'_{iL} = x_i$, (6) simplifies to :

$$D(X_{i-1}, Y_j) + d_e(x_i). \tag{7}$$

Case 3.

Consider the third term in (3). From its definition, we know that in every pair in $\Gamma^3_{X_i,Y_j}$ the last two elements of each string in the pair are :

$$x'_{iL-1} = \theta,\ x'_{iL} = x_i,\ y'_{jL-1} = y_{j-1},\ y'_{jL} = y_j.$$

Hence we are to compute :

$$\min_{((X'_i, Y'_j) \in \Gamma^3_{X_i,Y_j})} \sum_{i=1}^{|J'|} [\text{Distances Associated with Operations in } (X'_i, Y'_j)] \qquad (8)$$

In this case the inductive hypothesis leads to two distinct possibilities for the minimization to be achieved because the growing of (X'_{iL-2}, Y'_{jL-2}) to (X'_i, Y'_j) represents two unique sequences of edit operations given as Case 3.1 and 3.2 respectively.

Case 3.1 In this case the minimum is obtained as :

$$\min_{((X'_i, Y'_j) \in \Gamma^3_{X_i,Y_j})} \Big[\sum_{i=1}^{|J'|} [\text{Distances Associated with Operations in } (X'_{iL-2}, Y'_{jL-2})] + d_i(y'_{jL-1}) + d_s(x'_{iL}, y'_{jL}) \Big]. \qquad (9)$$

Since the inductive hypothesis is assumed true for the prefices of X_i and Y_j the first and second terms in the minimization can be coalesced in the computation. In this case it can be seen that for every element in $\Gamma^3_{X_i,Y_j}$ there is a unique element in $\Gamma_{X_{i-1},Y_{j-1}}$ and vice versa, where $\Gamma_{X_{i-1},Y_{j-1}}$ is the set of all ways by which X_{i-1} can be edited into Y_{j-1}. This unique element is obtained by merely reducing the lengths of the strings X'_i and Y'_j by unity. Hence, the coalescing of the first two terms of (9) yields $D(X_{i-1}, Y_j)$. Since $x'_{iL} = x_i$, (9) simplifies to :

$$D(X_{i-1}, Y_{j-1}) + d_s(x_i, y_j). \qquad (10)$$

Case 3.2 In this case the minimum is obtained as :

$$\min_{((X'_i, Y'_j) \in \Gamma^3_{X_i,Y_j})} \Big[\sum_{i=1}^{|J'|} [\text{Distances Associated with Operations in } (X'_{iL-2}, Y'_{jL-2})] + d_{ex}(x'_{iL-1}, y'_{jL-1} y'_{jL}) \Big]. \qquad (11)$$

In this case it must be noticed that $x'_{iL-1} = \theta$ which implies that the last non-θ symbol of X_{i-1} is x'_{iL-2}. Thus, for every element in $\Gamma^3_{X_i,Y_j}$ there is a unique element in $\Gamma_{X_{i-1},Y_{j-2}}$ and vice versa, where $\Gamma_{X_{i-1},Y_{j-2}}$ is the set of all ways by which X_{i-1} can be edited into Y_{j-2}. This unique element is obtained by reducing the length of the strings X'_i and Y'_j by two respectively. By the inductive hypothesis the latter minimization yields $D(X_{i-2}, Y_{j-1})$. Since $x'_{iL} = x_i$, and $y'_{jL-1} = y_{j-1}$, $y'_{jL} = y_j$, in this case (9) simplifies to :

$$D(X_{i-2}, Y_{j-1}) + d_{ex}(x_i, y_{j-1}y_j). \tag{12}$$

Similar arguments can be given for each of the remaining cases. The ideas are almost identical, because, in each case we consider the term to be minimized, analyze how the inductive hypothesis can be utilized, and compute the term obtained on utilizing the consequences of the latter. The additional terms that appear thereafter are then written down based on the individual edit operations that are represented. In the interest of brevity, the details of the remaining cases are omitted and the final results in each of the cases is written down.

Case 4.

Consider the fourth term in (3). In every pair in $\Gamma^4_{X_i,Y_j}$ we know that the last two elements of each string in the pair are

$$x'_{iL-1} = x_i,\ x'_{iL} = \theta,\ y'_{jL-1} = \theta,\ y'_{jL} = y_j.$$

Thus we are to evaluate :

$$\min_{((X'_i, Y'_j) \in \Gamma^4_{X_i,Y_j})} \sum_{i=1}^{|J'|} [\text{Distances Associated with Operations in } (X'_i, Y'_j)]$$

$$= D(X_i, Y_{j-1}) + d_i(y'_{jL}),$$
$$= D(X_i, Y_{j-1}) + d_i(y_j). \tag{13}$$

Case 5.

Consider the fifth term in (3). In every pair in $\Gamma^5_{X_i,Y_j}$ we know that the last two elements of each string in the pair are

$$x'_{iL-1} = x_i,\ x'_{iL} = \theta,\ y'_{jL-1} = y_{j-1},\ y'_{jL} = y_j.$$

Thus we are to evaluate :

$$\min_{((X_i', Y_j') \in \Gamma^5_{X_i, Y_j})} \sum_{i=1}^{|J'|} [\text{Distances Associated with Operations in } (X_i', Y_j')]$$

Arguing as in the case of the third term we obtain that this leads to the minimum of two terms which are given by (14) and (15) :

$$D(X_i, Y_{j-1}) + d_i(y_{jL}') = D(X_i, Y_{j-1}) + d_i(y_j). \quad (14)$$

$$D(X_{i-1}, Y_{j-2}) + d_{ex}(x_{iL-1}', y_{jL-1}' y_{jL}') = D(X_{i-1}, Y_{j-2}) + d_{ex}(x_i, y_{j-1} y_j) \quad (15)$$

Case 6.

Consider the sixth term in (3). In every pair in $\Gamma^6_{X_i, Y_j}$ we know that the last two elements of each string in the pair are

$$x_{iL-1}' = x_{i-1}, \; x_{iL}' = x_i, \; y_{jL-1}' = \theta, \; y_{jL}' = \theta.$$

Thus we are to evaluate :

$$\min_{((X_i', Y_j') \in \Gamma^6_{X_i, Y_j})} \sum_{i=1}^{|J'|} [\text{Distances Associated with Operations in } (X_i', Y_j')]$$

Arguing as in the case of the first term we obtain that this leads to the the quantity :

$$D(X_{i-1}, Y_j) + d_e(x_{iL}') = D(X_{i-1}, Y_j) + d_e(x_i). \quad (16)$$

Case 7.

Consider the seventh term in (3). In every pair in $\Gamma^7_{X_i, Y_j}$ we know that the last two elements of each string in the pair are

$$x_{iL-1}' = x_{i-1}, \; x_{iL}' = x_i, \; y_{jL-1}' = \theta, \; y_{jL}' = y_j.$$

Thus we are to evaluate :

$$\min_{((X_i', Y_j') \in \Gamma^7_{X_i, Y_j})} \sum_{i=1}^{|J'|} [\text{Distances Associated with Operations in } (X_i', Y_j')]$$

Arguing as in the case of the third term we can see that this leads to the minimum of two terms which are given by (17) and (18) :

$$D(X_{i-1}, Y_{j-1}) + d_s(x'_{iL}, y'_{jL}) = D(X_{i-1}, Y_{j-1}) + d_s(x_i, y_j). \quad (17)$$

$$D(X_{i-2}, Y_{j-1}) + d_{sq}(x'_{iL-1}x'_{iL}, y'_{jL}) = D(X_{i-2}, Y_{j-1}) + d_{sq}(x_{i-1}x_i, y_j). \quad (18)$$

Case 8.

Consider the eighth term in (3). In every pair in $\Gamma^8_{X_i,Y_j}$ we know that the last two elements of each string in the pair are

$$x'_{iL-1} = x_{i-1},\ x'_{iL} = x_i,\ y'_{jL-1} = y_j,\ y'_{jL} = \theta.$$

Thus we are to evaluate :

$$\min_{((X'_i, Y'_j) \in \Gamma^8_{X_i,Y_j})} \sum_{i=1}^{|J'|} [\text{Distances Associated with Operations in } (X'_i, Y'_j)]$$

Arguing as in the case of the third term we can see that this leads to the minimum of two terms which are given by (17) and (18) :

$$D(X_{i-1}, Y_j) + d_e(x'_{iL}) = D(X_{i-1}, Y_j) + d_e(x_i) \quad (19)$$

$$D(X_{i-2}, Y_{j-1}) + d_{sq}(x'_{iL-1}x'_{iL}, y'_{jL}) = D(X_{i-2}, Y_{j-1}) + d_{sq}(x_{i-1}x_i, y_j). \quad (20)$$

Case 9.

Finally, consider the ninth term in (3). In every pair in $\Gamma^9_{X_i,Y_j}$ we know that the last two elements of each string in the pair are

$$x'_{iL-1} = x_{i-1},\ x'_{iL} = x_i,\ y'_{jL-1} = y_{j-1},\ y'_{jL} = y_j.$$

Thus we are to evaluate :

$$\min_{((X'_i, Y'_j) \in \Gamma^9_{X_i,Y_j})} \sum_{i=1}^{|J'|} [\text{Distances Associated with Operations in } (X'_i, Y'_j)]$$

Arguing as in the first two cases we can see that this leads to :

$$D(X_{i-1}, Y_{j-1}) + d_s(x'_{iL}, y'_{jL}) = D(X_{i-1}, Y_{j-1}) + d_s(x_i, y_j). \quad (21)$$

Combining (5), (7), (10), (12-21) the theorem is proved. ♦♦♦

School of Computer Science, Carleton University
Recent Technical Reports

SCS-TR-154 **Ideal List Organization for Stationary Environments**
B. John Oommen and David T.H. Ng, March 1989

SCS-TR-155 **Hot-Spot Contention in Binary Hypercube Networks**
Sivarama P. Dandamudi and Derek L. Eager, April 89

SCS-TR-156 **Some Issues in Hierarchical Interconnection Network Design**
Sivarama P. Dandamudi and Derek L. Eager, April 1989

SCS-TR-157 **Discretized Pursuit Linear Reward-Inaction Automata**
B.J. Oommen and Joseph K. Lanctot, April 1989

SCS-TR-158 (revised) **Parallel Fractional Cascading on a Hypercube Multiprocessor**
Frank Dehne, Afonso Ferreira and Andrew Rau-Chaplin, May 1989 (Revised April 1990)

SCS-TR-159 **Epsilon-Optimal Stubborn Learning Mechanisms**
J.P.R. Christensen and B.J. Oommen, June 1989

SCS-TR-160 **Disassembling Two-Dimensional Composite Parts Via Translations**
Doron Nussbaum and Jörg-R. Sack, June 1989

SCS-TR-161 (revised) **Recognizing Sources of Random Strings**
R.S. Valiveti and B.J. Oommen, January 1990
Revised version of SCS-TR-161 "On the Data Analysis of Random Permutations and its Application to Source Recognition", published June 1989

SCS-TR-162 **An Adaptive Learning Solution to the Keyboard Optimization Problem**
B.J. Oommen, R.S. Valiveti and J. Zgierski, October 1989

SCS-TR-163 **Finding a Central Link Segment of a Simple Polygon in O(N Log N) Time**
L.G. Alexandrov, H.N. Djidjev, J.-R. Sack, October 1989

SCS-TR-164 **A Survey of Algorithms for Handling Permutation Groups**
M.D. Atkinson, January 1990

SCS-TR-165 **Key Exchange Using Chebychev Polynomials**
M.D. Atkinson and Vincenzo Acciaro, January 1990

SCS-TR-166 **Efficient Concurrency Control Protocols for B-tree Indexes**
Ekow J. Otoo, January 1990

SCS-TR-167 **A Hierarchical Stochastic Automaton Solution to the Object Partitioning Problem**
B.J. Oommen, January 1990

SCS-TR-168 **Adaptive List Organizing for Non-stationary Query Distributions. Part I: The Move-to-Front Rule**
R.S. Valiveti and B.J. Oommen, January 1990

SCS-TR-169 **Trade-Offs in Non-Reversing Diameter**
Hans L. Bodlaender, Gerard Tel and Nicola Santoro, February 1990

SCS-TR-170 **A Massively Parallel Knowledge-Base Server using a Hypercube Multiprocessor**
Frank Dehne, Afonso Ferreira and Andrew Rau-Chaplin, April 1990

SCS-TR-171 **Parallel Processing of Quad Trees on the Hypercube (and PRAM)**
Frank Dehne, Afonso Ferreira and Andrew Rau-Chaplin, April 1990

SCS-TR-172 **A Note on the Load Balancing Problem for Coarse Grained Hypercube Dictionary Machines**
Frank Dehne and Michel Gastaldo, May 1990

SCS-TR-173 **Self-Organizing Doubly-Linked Lists**
R.S. Valiveti and B.J. Oommen, May 1990

SCS-TR-174 **A Presortedness Metric for Ensembles of Data Sequences**
R.S. Valiveti and B.J. Oommen, May 1990

SCS-TR-175 **Separation of Graphs of Bounded Genus**
Ljudmil G. Aleksandrov and Hristo N. Djidjev, May 1990

SCS-TR-176 **Edge Separators of Planar and Outerplanar Graphs with Applications**
Krzystof Diks, Hristo N. Djidjev, Ondrej Sykora and Imrich Vrto, May 1990

SCS-TR-177 **Representing Partial Orders by Polygons and Circles in the Plane**
Jeffrey B. Sidney and Stuart J. Sidney, July 1990

SCS-TR-178 **Determining Stochastic Dependence for Normally Distributed Vectors Using the Chi-squared Metric**
R.S. Valiveti and B.J. Oommen, July 1990

SCS-TR-179 **Parallel Algorithms for Determining K-width-Connectivity in Binary Images**
Frank Dehne and Susanne E. Hambrusch, September 1990

SCS-TR-180 **A Workbench for Computational Geometry (WOCG)**
P. Epstein, A. Knight, J. May, T. Nguyen, and J.-R. Sack, September 1990

SCS-TR-181 **Adaptive Linear List Reorganization under a Generalized Query System**
R.S. Valiveti, B.J. Oommen and J.R. Zgierski, October 1990

SCS-TR-182 **Breaking Substitution Cyphers using Stochastic Automata**
B.J. Oommen and J.R. Zgierski, October 1990

SCS-TR-183 **A New Algorithm for Testing the Regularity of a Permutation Group**
V. Acciaro and M.D. Atkinson, November 1990

SCS-TR-184 **Generating Binary Trees at Random**
M.D. Atkinson and J.-R. Sack, December 1990

SCS-TR-185 **Uniform Generation of Combinatorial Objects in Parallel**
M.D. Atkinson and J.-R. Sack, January 1991

SCS-TR-186 **Reduced Constants for Simple Cycle Graph Separation**
Hristo N. Djidjev and Shankar M. Venkatesan, February 1991

SCS-TR-187 **Multisearch Techniques for Implementing Data Structures on a Mesh-Connected Computer**
Mikhail J. Atallah, Frank Dehne, Russ Miller, Andrew Rau-Chaplin, and Jyh-Jong Tsay, February 1991

SCS-TR-188 **Generating and Sorting Jordan Sequences**
Out of print Alan Knight and Jörg-Rüdiger Sack, March 1991

SCS-TR-189 **Probabilistic Estimation of Damage from Fire Spread**
Charles C. Colbourn, Louis D. Nel, T.B. Boffey and D.F. Yates, April 1991

SCS-TR-190 **Coordinators: A Mechanism for Monitoring and Controlling Interactions Between Groups of Objects**
Wilf R. LaLonde, Paul White, and Kevin McGuire, April 1991

SCS-TR-191 **Towards Decomposable, Reusable Smalltalk Windows**
Kevin McGuire, Paul White, and Wilf R. LaLonde, April 1991

SCS-TR-192 **PARASOL: A Simulator for Distributed and/or Parallel Systems**
John E. Neilson, May 1991

SCS-TR-193 **Realizing a Spatial Topological Data Model in a Relational Database Management System**
Ekow J. Otoo and M.M. Allam, August 1991

SCS-TR-194 **String Editing with Substitution, Insertion, Deletion, Squashing and Expansion Operations**
B John Oommen, September 1991