Constrained String Editing*

B. J. OOMMEN

School of Computer Science, Carleton University, Ottawa, K1S 5B6, Canada.

ABSTRACT

Let X and Y be any two strings of finite length. We consider the problem of transforming X to Y using the edit operations of deletion, insertion, and substitution. The optimal transformation is the one which has the minimum edit distance associated with it. The problem of computing this distance and the optimal transformation using no edit constraints has been studied in the literature. In this paper we consider the problem of transforming X to Y using any arbitrary edit constraint involving the number and type of edit operations to be performed. An algorithm is presented to compute the minimum distance associated with editing X to Y subject to the specified constraint. The algorithm requires $O(|X|\,|Y|\min(|X|,|Y|))$ time and space. The technique to compute the optimal transformation is also presented.

I. INTRODUCTION

In the study of the comparison of text patterns, syllables, 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 a recent excellent book edited by Sankoff and Kruskal [18] which discusses in detail the problem of sequence comparison.

The most promising of all the 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, 5–10, 15–19]. 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 the other. Apart from being a suitable index for

*A preliminary version of this paper was presented at the 1984 Mathematical Foundations of Computer Science Symposium in Prague, in September 1984. This work was partially supported by the Natural Sciences and Engineering Research Council of Canada.

0020-0255/86/$03.50

comparing two strings, this measure is closely related to the other numerical and nonnumerical measures that involve the strings, such as the longest common subsequence (LCS) and the shortest common supersequence [7].

Various algorithms to compute this distance have been proposed, the best of which are due to Wagner and Fischer [19] and Masek and Paterson [13]. For the infinite-alphabet case it has been shown that Wagner and Fischer's algorithm is the optimal one [20]. A generalization of [19] has even been used to correct noisy substrings [8]. Closely related to these algorithms are the ones proposed to compute the LCS of two strings by Hirschberg [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].

All of the abovementioned 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. The only algorithm presented in the literature on this problem is due to him. His algorithm is a LCS algorithm which involves a specialized constraint that has its application in the comparison of amino acid sequences.

In this paper we consider the problem of editing X to Y subject to any general edit constraint. This edit constraint can be arbitrarily complex, so long as it is specified in terms of the number and type of the edit operations to be included in the optimal edit transformation. For the sake of clarity we give below some examples of constrained editing.

EXAMPLE I. Here are some typical constrained editing problems.

(a) What is the optimal way of editing X to Y using no more than k insertions?

(b) How can we optimally transform X to Y using exactly k substitutions?

(c) Is it possible to transform X to Y using exactly k_s substitutions, k_i insertions, and k_e deletions? If it is possible, what is the distance between X and Y subject to this constraint?

In this paper we first present a consistent method of specifying any arbitrary edit constraint T. We then discuss the computation of $D_T(X, Y)$, the edit distance between X and Y subject to this constraint. The latter quantity is computed by first evaluating the elements of a three-dimensional array $\cdot W(\cdot,\cdot,\cdot)$ using dynamic-programming techniques, and then combining certain elements of $W(\cdot,\cdot,\cdot)$ to yield $D_T(X, Y)$. Using the array $W(\cdot,\cdot,\cdot)$, the optimal sequence of edit operations required to edit X to Y subject to the constraint T can also be obtained by backtracking. It will be shown that the algorithm requires $O(|X|\,|Y| \min(|X|, |Y|))$ time and space.

Apart from defining and solving the problem of constrained string editing in its most general framework, a major contribution of this paper is that of using the concepts of dynamic programming to compute a quantity that does not inherently possess any known recursively computable properties. The quantity which we are referring to is indeed the constrained edit distance $D_T(X, Y)$. However, we show that unlike that quantity, there is an array $W(\cdot, \cdot, \cdot)$ which is closely related to it and which can be computed using dynamic programming in a fairly straightforward way. This technique is quite analogous to the computations in a control system in which the output is evaluated in terms of state variables which are computable in real time.

1.1. NOTATION

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

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

1.2. THE SET OF ELEMENTARY EDIT DISTANCES $d(\cdot, \cdot)$ AND THE SET $G_{X,Y}$

$d(\cdot, \cdot): \tilde{A} \times \tilde{A} \to \mathbf{R}^+$ is a function whose arguments are a pair of symbols belonging to $\tilde{A}$, the appended alphabet, and whose range is the set of nonnegative real numbers; $d(\theta, \theta)$ is undefined and is not needed. The elementary distance $d(a, b)$ can be interpreted as the distance associated with transforming a to b, for $a, b \in \tilde{A}$. Thus,

(a) $d(x_i, y_j)$ is the distance associated with substituting y_j for x_i, $x_i, y_j \in A$.
(b) $d(x_i, \theta)$ is the distance associated with deleting $x_i \in A$.
(c) $d(\theta, y_j)$ is the distance associated with inserting $y_j \in A$.

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

$$\begin{aligned} G_{X,Y} = \{ (X', Y') \mid (X', Y') \in \tilde{A}^* \times \tilde{A}^*, &\text{ and each } (X', Y') \text{ obeys} \\ &\text{(i)} \quad C(X') = X,\ C(Y') = Y, \\ &\text{(ii)} \quad |X'| = |Y'|, \\ &\text{(iii)} \quad \text{for all } 1 \leqslant i \leqslant |X'|, \\ &\text{it is not the case that } x_i' = y_i' = \theta \}. \end{aligned} \tag{1}$$

By definition, if $(X', Y') \in G_{X,Y}$ then $\max[|X|, |Y|] \leqslant |X'| = |Y'| \leqslant |X| + |Y|$.

The meaning of the pair $(X', Y') \in G_{X,Y}$ is that it 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 $1 \leqslant i \leqslant |X'|$ 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'. Between these two cases, all the symbols in X are accounted for.

(iii) If $x_i' = \theta$ and $y_i' \in A$, it represents the insertion of y_i'. Between cases (i) and (iii) all the symbols in Y are accounted for.

$G_{X,Y}$ is an exhaustive enumeration of the set of all the ways by which X can be edited to Y using the edit operations of substitution, insertion, and deletion without destroying the order of occurrence of the symbols in X and Y.

The number of elements in the set $G_{X,Y}$ is given by

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

Note that $|G_{X,Y}|$ 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 $(X', Y') \in G_{X,Y}$.

EXAMPLE II. Let $X = f$ and $Y = go$. Then,

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

In particular the pair $(\theta f, go)$ represents the edit operations of inserting the g and replacing the f by an o.

Since the generalized Levenshtein distance (GLD) between X and Y is the minimum of the sum of the edit distances associated with the edit operations required to transform X to Y, this distance, written as $D(X, Y)$, has the expression [6,9],

$$D(X,Y) = \min_{(X',Y') \in G_{X,Y}} \sum_{i=1}^{|X'|} d(x_i', y_i'). \tag{2}$$

Since the transformation of a symbol $a \in A$ to itself is considered as a substitution operation, an application in which this is a nonoperation must specify $d(a, a) = 0$ for all $a \in A$.

II. EDIT CONSTRAINTS

II.1. PERMISSIBLE AND FEASIBLE EDIT OPERATIONS

Consider the problem of editing X to Y, where $|X| = N$ and $|Y| = M$. Suppose we edit a prefix of X into a prefix of Y, using exactly i insertions, e deletions (or erasures), and s substitutions. Since the *numbers* of edit operations are specified, this corresponds to editing $X_{e+s} = x_1 \dots x_{e+s}$, the prefix of X of length $e+s$, into $Y_{i+s} = y_1 \dots y_{i+s}$, the prefix of Y of length $i+s$.

To obtain bounds on the magnitudes of the variables i, e, and s, we observe that they are constrained by the lengths of the strings X and Y. Thus, if $r = e + s$, $q = i + s$, and $R = \min[M, N]$, these variables will have to obey the following obvious constraints:

$$\max[0, M-N] \leqslant i \leqslant q \leqslant M,$$
$$0 \leqslant e \leqslant r \leqslant N,$$
$$0 \leqslant s \leqslant \min[M, N].$$

Values of triples (i, e, s) which satisfy these constraints are termed the *feasible values* of the variables. Let

$$\begin{aligned} H_i &= \{ j | \max[0, M-N] \leqslant j \leqslant M \}, \\ H_e &= \{ j | 0 \leqslant j \leqslant N \}, \\ H_s &= \{ j | 0 \leqslant j \leqslant \min[M, N] \}. \end{aligned} \tag{3}$$

H_i, H_e, and H_s are called the set of *permissible* values of i, e, and s. Observe that a triple (i, e, s) is feasible if apart from $i \in H_i$, $e \in H_e$, $s \in H_s$ the following is satisfied:

$$i + s \leqslant M \quad \text{and} \quad e + s \leqslant N. \tag{4}$$

The following theorem specifies the permitted forms of the feasible triples which are encountered in editing X_r, the prefix of X of length r, to Y_q, the prefix of Y of length q.

THEOREM I. *To edit X_r, the prefix of X of length r, to Y_q, the prefix of Y of length q, the set of feasible triples is given by*

$$\{ (i, r-q+i, q-i) | \max[0, q-r] \leqslant i \leqslant q \}.$$

Proof. Consider the constraints imposed on feasible values of i, e, and s. Since we are interested in the editing X_r to Y_q, we have to consider only those triples (i, e, s) in which $i + s = r$ and $e + s = q$. But the number of insertions can take any value from $\max[0, q - r]$ to q. For every value of i in this range, the feasible triple (i, e, s) must have exactly $q - i$ substitutions.

Similarly, since the sum of the number of substitutions and the number of deletions is r, the triple (i, e, s) must have exactly $r - q + i$ deletions. Hence the theorem. ∎

II.2. SPECIFICATION OF EDIT CONSTRAINTS

An edit constraint is specified in terms of the number and type of edit operations that are required in the process of transforming X to Y. It is expressed by formulating the number and type of edit operations in terms of three sets Q_e, Q_i, and Q_s which are subsets of the sets H_e, H_i, and H_s defined in (3). We clarify this using the three constraints given in Example I.

EXAMPLE III.

(a) To edit X to Y performing no more than k insertions, the sets Q_s and Q_e are both equal to $\varnothing$, the null set. Further,

$$Q_i = \{j \mid j \in H_i, j \leqslant k\}.$$

(b) To edit X to Y performing exactly k substitutions, Q_i and Q_e would be null and, $Q_s = \{k\} \cap H_s$. Note that if k is not in H_s, the problem is infeasible.

(c) To edit X to Y performing exactly k_i insertions, k_e deletions, and k_s substitutions yields

$$Q_i = \{k_i\} \cap H_i, \qquad Q_e = \{k_e\} \cap H_e, \quad \text{and} \quad Q_s = \{k_s\} \cap H_s.$$

Note that the problem is infeasible unless $k_i \in H_i$, $k_e \in H_e$, and $k_s \in H_s$.

THEOREM II. *Every edit constraint specified for the process of editing X to Y can be written as a unique subset of H_i.*

Proof. Let the constraint be specified by the sets Q_i, Q_e, and Q_s. Every element $j \in Q_e$ requires the editing to be performed using exactly j deletions. From Theorem I, since $|X| = N$, this requires that the number of substitutions be $N - j$. Further, since $|Y| = M$, the number of insertions is forced to be $M - N + j$.

Similarly, if $j \in Q_s$, the edit transformations must contain exactly j substitutions. Since $|Y| = M$ and $|X| = N$, Theorem I requires that $M - j$ insertions and $N - j$ deletions be performed.

Let

$$Q_e^* = \{M - N + j \mid j \in Q_e\} \quad \text{and} \quad Q_s^* = \{M - j \mid j \in Q_s\}.$$

Clearly, for any arbitrary constraint the number of insertions that are permitted is given by a set of integers which is obtained by intersecting Q_i, Q_e^*, and Q_s^*, which is obviously a subset of H_i. ■

REMARKS.

(1) The set referred to above, which describes the constraint and which is the subset of H_i, will in future be written as T.

(2) The edit constraint can just as easily be written as a subset of H_s or H_e. We have arbitrarily chosen to describe T as a subset of H_i.

(3) Observe that the converse of Theorem II is not true. Verbal descriptions of many edit constraints could lead to the same set T.

EXAMPLE IV. Let $X = for$ and $Y = fa$. Suppose we want to transform X to Y by performing at least 1 insertion, at most 1 substitution, and exactly 2 deletions. Then,

$$Q_i = \{1,2\}, \qquad Q_e = \{2\}, \quad \text{and} \quad Q_s = \{0,1\}.$$

Hence,

$$Q_e^* = \{1\} \text{ and } Q_s^* = \{1,2\}.$$

Thus $T = Q_i \cap Q_e^* \cap Q_s^* = \{1\}$.

Hence the optimal transformations must contain *exactly* one insertion. Some candidate edit transformations are given by the following subset of $G_{X,Y}$:

$$\{(\theta for, f\theta\theta a), (f\theta or, fa\theta\theta), (fo\theta r, f\theta a\theta), (for\theta, f\theta\theta a)\}.$$

Note that every pair in the above subset corresponds to at least 1 insertion, at most 1 substitution, and exactly 2 deletions.

We shall refer to the edit distance subject to the constraint T as $D_T(X,Y)$. By definition, $D_T(X,Y) = \infty$ if $T = \varnothing$. This is merely a simple way of expressing that it is impossible to edit X to Y subject to the constraint T. We shall now consider the computation of $D_T(X,Y)$.

III. W: THE ARRAY OF CONSTRAINED EDIT DISTANCES

Let $W(i,e,s)$ be the constrained edit distance associated with editing X_{e+s} to Y_{i+s} subject to the constraint that exactly i insertions, e deletions, and s substitutions are performed in the process of editing. As before, let $r=e+s$ and $q=i+s$. Let $G_{i,e,s}(X,Y)$ be the subset of the pairs in G_{X_r,Y_q} in which every pair corresponds to i insertions, e deletions, and s substitutions. Since we shall always be referring to the strings X and Y, we refer to this set as $G_{i,e,s}$. Thus, using the notation of (1), and assuming (i,e,s) is feasible for the problem, $W(i,e,s)$ has the expression

$$W(i,e,s)=\min_{(X_r',Y_q')\in G_{i,e,s}}\sum_{j=1}^{|X_r'|} d(x_{rj}',y_{qj}'). \tag{5}$$

We shall derive the recursively computable properties of the array $W(\cdot,\cdot,\cdot)$.

THEOREM III. *Let $W(i,e,s)$ be the quantity defined in (5) for any two strings X and Y. Then,*

$$\begin{aligned} W(i,e,s)=\min[&\{W(i-1,e,s)+d(\theta,y_{i+s})\},\\ &\{W(i,e-1,s)+d(x_{e+s},\theta)\},\\ &\{W(i,e,s-1)+d(x_{e+s},y_{i+s})\}] \end{aligned}$$

for all feasible triples (i,e,s).

The theorem is proved in the appendix.

The computation of the distance $D_T(X,Y)$ from the array $W(i,e,s)$ only involves combining the appropriate elements of the array using T, the set of insertions permitted. This is proved in the following theorem.

THEOREM IV. *The quantity $D_T(X,Y)$ is related to the elements of the array $W(i,e,s)$ as follows:*

$$D_T(X,Y)=\min_{i\in T} W(i,N-M+i,M-i).$$

Proof. The theorem follows directly from Theorem I with the substitution $r=N$ and $q=M$. ∎

REMARK. In an unconstrained editing problem i can take on any value ranging from $\max[0, M-N]$ to M. Thus the unconstrained edit distance (which is commonly known as the generalized Levenshtein distance) $D(X,Y)$ has the expression

$$D(X,Y) = \min_{i \in [\max[0, M-N], M]} W(i, N-M+i, M-i).$$

Using the results of the above two theorems, we now propose a computational scheme for $D_T(X,Y)$.

IV. THE COMPUTATION OF $W(\cdot,\cdot,\cdot)$ AND $D_T(X,Y)$

To compute $D_T(X,Y)$ we make use of the fact that though this index does not itself seem to have any recursive properties, the index $W(\cdot,\cdot,\cdot)$, which is closely related to it, has the interesting properties proved in Theorem III. Algorithm I, which we now propose, computes the array $W(\cdot,\cdot,\cdot)$ for all feasible values of the variables i, e, and s. Subsequently, using the array $W(i,e,s)$ as input, Algorithm II computes $D_T(X,Y)$ by comparing the contributions of the pertinent elements in $W(\cdot,\cdot,\cdot)$ as specified by Theorem IV.

The computation of the array $W(\cdot,\cdot,\cdot)$ has to be done in a systematic manner, so that any quantity $W(i,e,s)$ is computed *before* its value is required in any further computation. This is easily done by considering a three-dimensional coordinate system whose axes are i, e, and s. Initially the weight associated with the origin, $W(0,0,0)$, is assigned the value zero, and the weights associated with the vertices on the axes are evaluated. Thus, $W(i,0,0)$, $W(0,e,0)$, and $W(0,0,s)$ are computed for all permissible values of i, e, and s. Subsequently, the i-e, e-s, and i-s planes are traversed, and the weights associated with the vertices on these planes are computed using the previously computed values. Finally, the weights corresponding to strictly positive values of the variables are computed. To avoid unnecessary computations, at each stage the variables are tested for feasibility. The quantity $D_T(X,Y)$ is evaluated by comparing the weights associated with the triples of the form

$$\{(i, N-M+i, M-i) \mid i \in T\}.$$

The algorithm to compute $W(\cdot,\cdot,\cdot)$ is given below.

ALGORITHM I.

Input: The strings $X = x_1 x_2 \dots x_N$, $Y = y_1 y_2 \dots y_M$, and the set of elementary edit distances defined by $d(\cdot,\cdot)$. Let $R = \min[M,N]$.

Output: The array $W(i,e,s)$ for all feasible values of i, e, and s.
Method:

```
$W(0,0,0) = 0$
for $i = 1$ to $M$ do $W(i,0,0) = W(i-1,0,0) + d(\theta, y_i)$
for $e = 1$ to $N$ do $W(0,e,0) = W(0,e-1,0) + d(x_e, \theta)$
for $s = 1$ to $R$ do $W(0,0,s) = W(0,0,s-1) + d(x_s, y_s)$

for $i = 1$ to $M$ do
  for $e = 1$ to $N$ do
    $W(i,e,0) = \min[W(i-1,e,0) + d(\theta, y_i), W(i,e-1,0) + d(x_e, \theta)]$
  end
end

for $i = 1$ to $M$ do
  for $s = 1$ to $M - i$ do
    $W(i,0,s) = \min[W(i-1,0,s) + d(\theta, y_{i+s}), W(i,0,s-1) + d(x_s, y_{i+s})]$
  end
end

for $e = 1$ to $N$ do
  for $s = 1$ to $N - e$ do
    $W(0,e,s) = \min[W(0,e-1,s) + d(x_{s+e}, \theta), W(0,e,s-1) + d(x_{s+e}, y_s)]$
  end
end

for $i = 1$ to $M$ do
  for $e = 1$ to $N$ do
    for $s = 1$ to $\min[(M-i),(N-e)]$ do
  $W(i,e,s) = \min[\{W(i-1,e,s) + d(\theta, y_{i+s})\},$
                  $\{W(i,e-1,s) + d(x_{e+s}, \theta)\},$
                  $\{W(i,e,s-1) + d(x_{e+s}, y_{i+s})\}]$
    end
  end
end
```

END Algorithm I

We now present Algorithm II, which has for its input the array $W(\cdot,\cdot,\cdot)$, and yields as its output the quantity $D_T(X,Y)$.

ALGORITHM II.

Input: The array $W(\cdot,\cdot,\cdot)$ computed using Algorithm I, and the constraint set T.
Output: The constrained distance $D_T(X,Y)$.

Method:

```
$D_T(X,Y) = \infty$
for all $i \in T$ do
    $D_T(X,Y) = \min[D_T(X,Y), W(i, N-M+i, M-i)]$
end
```

END Algorithm II

REMARKS.

(1) 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, 9, 20]. In this case, the number of symbol comparisons required by Algorithm I has an upper bound of

$$\#(\text{Alg I}) = \begin{cases} \dfrac{(N-M)M(M+1)}{2} + \dfrac{M(M+1)(2M+1)}{6} & \text{if} \quad N > M, \\ \dfrac{(M-N)N(N+1)}{2} + \dfrac{N(N+1)(2N+1)}{6} & \text{otherwise.} \end{cases}$$

Note that for every symbol comparison, we will need *at most* three multiplications and two additions. From the last set of "for" loops it is easy to see that the time required to compute the array $W(\cdot,\cdot,\cdot)$ is $O(MNR)$, where $R = \min[M, N]$. Thus it has a worst-case complexity which is cubic in time. Algorithm II clearly requires linear time. Thus the overall time required to compute $D_T(X,Y)$ is $O(MNR)$. We conjecture that this is a lower bound for the number of symbol comparisons required for any algorithm which attempts to solve the constrained string editing problem.

(2) The array $W(i,e,s)$ contains far more information than is required to merely compute $D_T(X,Y)$. It contains all the weights associated with editing prefixes of X into prefixes of Y. One could therefore use the contents of the array to compute far more complicated indices which concern constrained edit distances involving prefixes of either or both the strings. We illustrate the computation of $D_T(X,Y)$ with an example.

EXAMPLE V. Let $X = aa$ and $Y = bc$. Let us suppose we want to edit X to Y, permitting either two substitutions or exactly one edit operation of each type. It can be seen that the set T defining the constraint is $\{0,1\}$.

We shall now follow through the computation of $D_T(aa, bc)$ using Algorithms I and II. To begin with, the weight associated with the origin is initialized. The i, e, and s axes are then traversed:

$$\begin{aligned} W(1,0,0) &= d(\theta,b), & W(2,0,0) &= d(\theta,b)+d(\theta,c),\\ W(0,1,0) &= d(a,\theta), & W(0,2,0) &= 2d(a,\theta),\\ W(0,0,1) &= d(a,b), & W(0,0,2) &= d(a,b)+d(a,c). \end{aligned}$$

The i-e, i-s, and e-s planes are then traversed:

$$\begin{aligned} W(1,1,0) &= d(\theta,b)+d(a,\theta), \qquad W(1,2,0) = d(\theta,b)+2d(a,\theta),\\ W(2,1,0) &= d(\theta,b)+d(\theta,c)+d(a,\theta),\\ W(2,2,0) &= d(\theta,b)+d(\theta,c)+2d(a,\theta),\\ W(1,0,1) &= \min[d(\theta,b)+d(a,c),d(\theta,c)+d(a,b)],\\ W(0,1,1) &= \min[d(a,\theta)+d(a,b),d(a,\theta)+d(a,c)]. \end{aligned}$$

Finally, the weights for strictly positive values of i, e, and s are computed:

$$\begin{aligned} W(1,1,1) = \min[&\{d(\theta,b)+d(a,\theta)+d(a,c)\},\\ &\{d(\theta,c)+d(a,\theta)+d(a,b)\}]. \end{aligned}$$

Since the terms that must be compared to obtain $D_T(X,Y)$ are those which involve $i=0$ and $i=1$, $D_T(X,Y)$ can now be trivially computed as the minimum of $W(0,0,2)$ and $W(1,1,1)$.

IV.1. COMPUTING THE BEST EDIT SEQUENCE

Once the quantity $D_T(X,Y)$ has been computed, the optimal edit sequence can be obtained by backtracking through the array $W(\cdot,\cdot,\cdot)$ and printing out the actual edit sequence traversed, in the reverse order. The technique is well known in dynamic-programming problems and has been used extensively for edit sequences [13, 15, 19] and longest common subsequences [3, 4, 5, 6]. Without further comment we now present Algorithm III, which has as its input the distance $D_T(X,Y)$ and the optimal element of $W(\cdot,\cdot,\cdot)$, i.e., the element $W(I,E,S)$ which is equal to $D_T(X,Y)$.

To simplify the backtracking, we exclude the possibility of encountering negative values of i, e, and s by rendering $W(i,e,s)$ infinite whenever any of the three indices are negative.

ALGORITHM III.

Input: The indices I, E, and S for which $D_T(X, Y) = W(I, E, S)$

Output: The optimal sequence of edit operations subject to the specified constrained. The sequence is given in the reverse order and in the following notation:

(a) The pair (x_i, y_j) means the substitution of y_j for x_i.
(b) The pair (x_i, θ) means the deletion of x_i.
(c) The pair (θ, y_j) means the insertion of y_j.

Method:

```
$i = I$, $e = E$, $s = S$, where $W(I, E, S) = D_T(X, Y)$.
Define $W(i, e, s) = \infty$ whenever $i < 0$ or $e < 0$ or $s < 0$.

while ($i \neq 0$ or $e \neq 0$ or $s \neq 0$) do
  if ($W(i, e, s) = W(i-1, e, s) + d(\theta, y_{i+s})$)
then
    print $(\theta, y_{i+s})$
    $i = i - 1$
  else if ($W(i, e, s) = W(i, e-1, s) + d(x_{e+s}, \theta)$)
    then
      print $(x_{e+s}, \theta)$
      $e = e - 1$
    else
      print $(x_{e+s}, y_{i+s})$
      $s = s - 1$
    endif
  endif
endwhile
```

END Algorithm III

Obviously Algorithm III is performed in $O(\max(M, N))$ time.

REMARKS.

(1) The entire thrust of this paper has been to study the constrained editing of strings in which the primitive edit operations are substitution, insertion, and deletion of individual characters. The problem of editing strings subject to generalized constraints and using the edit operations of substitution, insertion, deletion, *and the transposition of characters* remains unsolved.

(2) Using the concepts introduced here we can also conceive of constrained measures related to the LCS of two strings. These measures would impose more

generalized constraints than those that have been studied in the literature [17] and could be used to study the similarity between biological molecules which have gross differences in the lengths of their string representations.

(3) 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, for previous research applies dynamic-programming tools to obtaining recursive formulations for $D(X, Y)$. Observe however that the constrained distance between X and Y is not recursively computable in terms of the constrained distanced between the prefixes of the corresponding strings. One of the highlights of this paper is that we have shown that $D_T(X, Y)$ can be computed by evaluating the recursively computable index $W(\cdot, \cdot, \cdot)$. This is reminiscent of a control system in which the output is computed in terms of state variables which are recursively computable.

(4) The use of the generalized Levenshtein distance to perform the automatic correction of noisy strings [9] 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 the substitutions of individual characters. One problem that has been open is that of correcting erroneous strings and substrings in which, apart from the latter errors, the noisy string also has chunks of characters deleted. From our experience, we believe that *constrained* string editing properties of strings can be used to perform the correction of noisy strings which contain all the above types of errors. We are currently investigating this possibility.

V. CONCLUSIONS

In this paper we have considered the problem of editing a string X to a string Y subject to a specified edit constraint. The edit constraint is fairly arbitrary and can be specified in terms of the number and type of edit operations desired in the optimal transformation. The way by which the constraint T can be specified has has been proposed. Also the technique to compute $D_T(X, Y)$, the edit distance subject to the constraint T, has been presented. A final algorithm has been given which has as its input the quantity $D_T(X, Y)$ and outputs the optimal edit transformation subject to the specified constraint.

Given the strings X and Y, $D_T(X, Y)$ and the array of constrained edit distances $W(\cdot, \cdot, \cdot)$ can be computed in $O(|X|\,|Y|\min(|X|, |Y|))$ time. If $D_T(X, Y)$ is given, the optimal edit transformation can be obtained by backtracking through $W(\cdot, \cdot, \cdot)$ in $O(\max(|X|, |Y|))$ time.

The problem of constrained string editing using the elementary edit operations of substitution, insertion, deletion, and the transportation of characters remains unsolved.

We are currently investigating the use of constrained edit distances in the pattern recognition of noisy substrings and boundary segments.

APPENDIX

Proof of Theorem II. It is required to prove that for all feasible values of i, e, and s,

$$\begin{aligned} W(i,e,s) = \min[& \{ W(i-1,e,s) + d(\theta, y_{i+s}) \}, \\ & \{ W(i,e-1,s) + d(x_{e+s}, \theta) \}, \\ & \{ W(i,e,s-1) + d(x_{e+s}, y_{i+s}) \}]. \end{aligned}$$

The proof of the theorem is divided into three main cases:

Case (a): Any two of the three variables i, e, and s are zero.
Case (b): Any one of the three variables i, e, and s is zero.
Case (c): None of the variables i, e, and s are zero.

The most involved of these cases is case (c). Cases (a) and (b) are merely one- and two-parameter subcases respectively of case (c). To avoid repetition, we shall here prove only case (c). Thus, for the rest of the proof, we encounter only strictly positive values for the variables i, e, and s.

Let $r = e + s$, $q = i + s$, $X_r = x_1 \dots x_r$, and $Y_q = y_1 \dots y_q$. By definition,

$$W(i,e,s) = \min_{(X_r', Y_q')} \sum_{j=1}^{|X_r'|} d(x_{rj}', y_{qj}'), \tag{A.1}$$

where (X_r', Y_q') is an arbitrary element of the set $G_{i,e,s}$, with x_{rj}' and y_{qj}' as the jth symbols of X_r' and Y_q' respectively. Let the lengths of the strings X_r' and Y_q' in the arbitrary element be L. Then the last symbols of X_r' and Y_q' are x_{rL}' and y_{qL}' respectively.

We partition the set $G_{i,e,s}$ into three *mutually exclusive and exhaustive* subsets:

$$G_{i,e,s}^1 = \{ (X_r', Y_q') | (X_r', Y_q') \in G_{i,e,s}, x_{rL}' = x_r, y_{qL}' = y_q \},$$

$$G_{i,e,s}^2 = \{ (X_r', Y_q') | (X_r', Y_q') \in G_{i,e,s}, x_{rL}' = x_r, y_{qL}' = \theta \},$$

$$G_{i,e,s}^3 = \{ (X_r', Y_q') | (X_r', Y_q') \in G_{i,e,s}, x_{rL}' = \theta, y_{qL}' = y_q \}.$$

By their definitions, we see that the above three sets are mutually exclusive. Further, since x'_{rL} and y'_{qL} cannot be θ simultaneously, every pair in $G_{i,e,s}$ must be in one of the above sets. Hence these three sets partition $G_{i,e,s}$. Rewriting (A.1), we obtain

$$W(i,e,s) = \min_{k=1,2,3} \min_{(X'_r, Y'_q) \in G^k_{i,e,s}} S', \tag{A.2}$$

where $S' = \sum_{j=1}^{|X'_r|} d(x'_{rj}, y'_{qj})$.

Consider each of the terms in (A.2) individually. In every pair in $G^1_{i,e,s}$, $x'_{rL} = x_r$ and $y'_{qL} = y_q$. Hence,

$$\begin{aligned} &\min_{(X'_r, Y'_q) \in G^1_{i,e,s}} \sum_{j=1}^{|X'_r|} d(x'_{rj}, y'_{qj}) \\ &= \min_{(X'_r, Y'_q) \in G^1_{i,e,s}} \sum_{j=1}^{|X'_r|-1} d(x'_{rj}, y'_{qj}) + d(x_r, y_q). \end{aligned} \tag{A.3}$$

For every element in $G^1_{i,e,s}$ there is a unique element in $G_{i,e,s-1}$ and vice versa. Hence, the first term in the above expression is exactly $W(i,e,s-1)$. Since $r = e+s$ and $q = i+s$,

$$\min_{(X'_r, Y'_q) \in G^1_{i,e,s}} \sum_{j=1}^{|X'_r|} d(x'_{rj}, y'_{qj}) = W(i,e,s-1) + d(x_{e+s}, y_{i+s}). \tag{A.4}$$

Consider the second term in (A.2). In every pair in $G^2_{i,e,s}$, $x'_{rL} = x_r$ and $y'_{qL} = \theta$. Hence,

$$\min_{(X'_r, Y'_q) \in G^2_{i,e,s}} \sum_{j=1}^{|X'_r|} d(x'_{rj}, y'_{qj}) = \min_{(X'_r, Y'_q) \in G^2_{i,e,s}} \sum_{j=1}^{|X'_r|-1} d(x'_{rj}, y_{qj}) + d(x_r, \theta). \tag{A.5}$$

For every element in $G^2_{i,e,s}$ there is a unique element in $G_{i,e-1,s}$ and vice versa. Hence, the first term in the above expression is exactly $W(i,e-1,s)$. Thus,

$$\min_{(X'_r, Y'_q) \in G^2_{i,e,s}} \sum_{j=1}^{|X'_r|} d(x'_{rj}, y'_{qj}) = W(i,e-1,s) + d(x_{e+s}, \theta). \tag{A.6}$$

Consider the third term in (A.2). In every pair in $G^3_{i,e,s}$, $x'_{rL} = \theta$ and $y'_{qL} = y_q$. Hence,

$$\min_{(X'_r, Y'_q) \in G^3_{i,e,s}} \sum_{j=1}^{|X'_r|} d(x'_{rj}, y'_{qj}) = \min_{(X'_r, Y'_q) \in G^3_{i,e,s}} \sum_{j=1}^{|X'_r|-1} d(x'_{rj}, y'_{qj}) + d(\theta, y_q). \tag{A.7}$$

For every element in $G^3_{i,e,s}$ there is a unique element in $G_{i-1,e,s}$ and vice versa. Hence, the first term in the above expression is exactly $W(i-1, e, s)$. As in the above cases,

$$\min_{(X'_r, Y'_q) \in G^3_{i,e,s}} \sum_{j=1}^{|X'_r|} d(x'_{rj}, y'_{qj}) = W(i-1, e, s) + d(\theta, y_{i+s}). \tag{A.8}$$

Resubstituting (A.4), (A.6), and (A.8) in (A.2) proves the theorem. ■

I would like to thank Professor Jeff Sidney of the University of Ottawa for proofreading the paper thoroughly and for the time he spent to improve the readability of the paper.

REFERENCES

1. A. V. Aho, D. S. Hirschberg, and J. D. Ullman, 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 the 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. ACM* 18:341–343 (1975).
5. J. W. Hunt and T. G. Szymanski, A fast algorithm for computing longest common subsequences, *Comm. ACM* 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 (Mar. 1983).
7. R. L. Kashyap and B. J. Oommen, Similarity measures for sets of strings, *Internat. J. Comput. Math.* 13:95–104 (May 1983).
8. R. L. Kashyap and B. J. Oommen, The noisy substring matching problem, *IEEE Trans. Software Engrg.* 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 optimatlity, *Inform. Sci.* 23(2):123–142 (Mar. 1981).

10. R. L. Kashyap and B. J. Oommen, Probabilistic correction of strings, in *Proceedings of the IEEE Transactions on Pattern Recognition and Image Processing*, June 1982, pp. 28–33.
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 proteins, *J. Mol. Biol.*, 1970, pp. 443–453.
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. ACM* 23:676–687 (1980).
17. D. Sankoff, Matching sequences under deletion/insertion constraints, *Proc. Nat. Acad. Sci. U.S.A.* 69:4–6 (Jan. 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. Fischer, 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).

Received 11 April 1985; revised 1 May 1986