The File Transmission Problem

Peter Weiner and Robert W. Tuttle

Research Report #16

This work was partially supported by grants from the Alfred P. Sloan Foundation and the Exxon Education Foundation.

March 1973

The File Transmission Problem

Abstract

The *file transmission problem* is to determine the best way to send a file A (assumed to be a linear string over a finite alphabet) from one computer to another via a transmission line, assuming that the receiving computer has access to another file B called the base file. In addition to sending the characters of A directly, we allow the transmission of a copy command which directs the receiving computer to append a specified, but variable length, substring of characters taken from the base file to the end of the file under construction. The cost of transmission is taken as the sum of the number of characters directly sent and K times the number of copy commands. An *optimal derivation* of A is a minimum-cost sequence of characters and copy commands which allow the receiving computer to construct the file A. We present an algorithm for obtaining an optimal derivation. This algorithm is itself optimal up to a constant factor in that both its run time and storage requirements are linear functions of the lengths of A and B.

I. Introduction

With the proliferation of computer networks, it has become important to find efficient methods for sending files from one computer to another. Often, a *base file*, B, similar to the file to be sent, A, is available to the receiving computer. (Perhaps the base file is a source-language program which we have just modified and which was sent to the remote computer for execution prior to modification.) Instead of sending the characters of A directly, it may be more efficient to send information which allows the receiving computer to construct file A using the base file B.

One possible approach is to send a sequence of edit commands. For example, these commands could be instructions to delete, insert, or modify characters already in the base file. In [1], Wagner and Fischer present an algorithm for finding the best command sequence of this type relative to a completely general cost function. The algorithm also has application to problems other than the file transmission problem, but its utility for our problem is limited by the fact that its running time is proportional to the product of the length of A with the length of B. In the next section we describe an alternate choice of commands to be sent to the receiving computer. After introducing an associated cost function, we present an algorithm for obtaining optimal command sequences. We show that this algorithm is itself optimal up to a constant factor in that both its run time and storage requirements are linear functions of the lengths of A and B.

II. Command Sequences and B-derivations

Both the file to be sent, A, and the base file, B, are taken to be finite length strings of symbols (or letters) over a finite alphabet S. The file A will be denoted as

$$A = A\langle 1\rangle\ A\langle 2\rangle\ \ldots\ A\langle n\rangle,$$

where $A\langle i\rangle$, a symbol in S, is the i^{th} letter of the file, and is said to be located in the i^{th} position. To represent the substring of characters which begins at position i of A and ends at position j, we write $A\langle i{:}j\rangle$. That is, when $i \leq j$, $A\langle i{:}j\rangle = A\langle i\rangle\ \ldots\ A\langle j\rangle$, and $A\langle i{:}j\rangle = \Lambda$, the null string, for $i > j$. Similarly, $B = B\langle 1\rangle\ B\langle 2\rangle\ \ldots\ B\langle m\rangle$ denotes a base file of m characters.

We have in mind a scheme of sending copy commands interspersed among the symbols of A to the receiving computer. These commands will instruct the computer to use specified, but variable length, substrings from the base file, say $B\langle i{:}j\rangle$, in constructing the file A. Consequently, we define a *command*, c, to be either a *letter command*, written as s (a symbol of S), or a *copy command*, written $X(p,\ell)$. The substring to be copied, $B\langle p{:}p+\ell-1\rangle$, is located starting at position p of B and is of length ℓ, so p and ℓ must be selected to satisfy

$$1 \leq p \leq m - \ell + 1 \quad \text{and} \quad 1 \leq \ell \leq m.$$

(The length of the base file, $|B|$, is m.)

A string T' is said to be derived from the string T by the command c, written $T \xrightarrow[c]{} T'$, if

$$T' = \begin{cases} T\ s & \text{if } c = s \\ T\ B\langle p{:}p+\ell-1\rangle & \text{if } c = X(p,\ell). \end{cases}$$

A *B-derivation* of A using B is a finite sequence of commands $C = c_1, c_2, \ldots, c_t$, such that there exist strings $T_0, T_1, \ldots, T_t$, with $T_0 = \Lambda$, $T_t = A$, and

$$T_{i-1} \xrightarrow{c_i} T_i \text{ for } 1 \leq i \leq t.$$

If $C = c_1, c_2, \ldots, c_t$ is a B-derivation of A, then C and $A\langle 1{:}m\rangle$ describe the same string. It is natural to decompose A as $A\langle q_1{:}r_1\rangle\ A\langle q_2{:}r_2\rangle \ldots A\langle q_t{:}r_t\rangle$, where $q_1 = 1$, $r_t = m$, $q_i = r_{i-1} + 1$ for $1 < i \leq t$, and

$$r_i = \begin{cases} q_i & \text{if } c_i \text{ is } s \\ q_i + \ell - 1 & \text{if } c_i \text{ is } X(p,\ell) \end{cases} \qquad 1 \leq i \leq t.$$

We say that the command c_i *generates* $A\langle q_i{:}r_i\rangle$ and that c_i has index q_i.

In order to model the cost of transmitting command sequences from one computer to another, we define the cost of a command to be

$$\gamma(c) = \begin{cases} 1 & \text{if } c = s \\ K \geq 1 & \text{if } c = X(p,\ell). \end{cases}$$

The cost of the B-derivation $C = c_1, c_2, \ldots, c_t$ is obtained by summing the component costs, that is,

$$\gamma(C) = \sum_{i=1}^{t} \gamma(c_i).$$

We are interested in obtaining *optimal derivations*, which are the minimum-cost command sequences that generate the transmitted file.

Remark:

While we wish to allow the cost of a copy command, K, to be arbitrary, reasonable choices for K can be made once a scheme for transmitting copy commands is specified. For example, assuming that S contains an escape character not found in the file A, as well as the digits 0, 1, ..., 9, we could transmit a copy command as the escape character followed by two positive numbers each no greater than m. For this scheme, a reasonable choice of K would be the number of characters required to describe a copy command, $2\lceil \log_{10} m \rceil + 1$.

We are now ready to prove some elementary properties of optimal derivations.

Lemma 1:

Suppose that $C = c_1,\ldots,c_t$ is a B-derivation of the string A and that $C' = c_i,c_{i+1},\ldots,c_j$, $1 \leq i \leq j \leq t$, is a B-derivation of A'. Then if C is an optimal derivation of A, C' must be an optimal derivation of A'.

Proof:

If C' is not optimal, then there must exist another B-derivation of A', C'', with $\gamma(C'') < \gamma(C')$. But this implies that the B-derivation of A obtained by substituting the sequence of commands of C'' for $c_i,c_{i+1},\ldots,c_j$ is of less cost than that of C. Since this is not possible when C is optimal, C' must also be optimal.

QED

Corollary 1.1:

In every optimal B-derivation, every copy command $X(p,\ell)$ has length $\ell \geq K$.

Proof:

A copy command of length $\ell < K$ is not an optimal derivation of the ℓ-long string of symbols it generates.

QED

III. Match Functions

In this section, we define two functions that greatly facilitate the synthesis of an algorithm that calculates optimal derivations. These functions are closely related to problems of pattern matching. In a companion paper [2], we describe efficient algorithms for their computation. In this work, we take as fact the claim that these functions can be calculated in time and space that are linearly dependent on $|A|$ and $|B|$.

Definition:

Let A and B be files of length n and m respectively. Then the *match function* M(i) and the *position function* P(i) are defined for $1 \leq i \leq n$ by the equation

$$A\langle i:i+M(i)-1\rangle = B\langle P(i);P(i)+M(i)-1\rangle$$

and the condition that either $i+M(i)-1 = n$ or $A\langle i:i+M(i)\rangle$ is not a substring of B. If several substrings of B satisfy the equation, P(i) is arbitrarily selected to be the leftmost position of any of these substrings. If $A\langle i\rangle$ does not occur in B, then $M(i) \equiv 0$ and P(i) is undefined.

Thus, M(i) is the length k of the longest substring in B identical to $A\langle i:i+k-1\rangle$, and when $M(i) > 0$, P(i) is a position in B that begins such a substring.

It is interesting to observe that the match function can be thought of as a "superposition of triangles," since whenever $M(i) = k > 1$ for $1 \leq i < n$, $M(i+1) \geq k - 1$. (Note also that when $M(i+1) = M(i) - 1 > 0$, P(i+1) can be selected to be P(i) + 1.)

The match function allows us to provide a corollary of Lemma 1 which is somewhat dual to Corollary 1.1:

Corollary 1.2:

If K + 1 consecutive letter commands occur in an optimal derivation of A, the first having index i (of A), then $M(i) \leq K$.

Proof:

If $M(i) > K$, the K + 1 consecutive letter commands are not an optimal derivation of the (K+1)-long substring they generate.

QED

Whenever a copy command $X(p,\ell)$ of index i appears in a derivation it must be the case that $\ell \leq M(i)$. When $\ell = M(i)$, we call the copy command *maximal*.

It is not hard to exhibit optimal derivations which contain sub-maximal copy commands. The following theorem, however, allows us to restrict attention to derivations which contain only maximal copy commands. Such derivations will be called *normal*.

Theorem 1:

For any files A and B, there exists at least one optimal B-derivation of A which is normal.

Proof:

Since the cost of every B-derivation of A is finite, an optimal derivation certainly exists, say C. If C is a single command it is obviously normal.

Assume then, for purposes of induction, that every optimal

B-derivation containing k commands can be replaced by a normal optimal derivation containing k commands, and suppose that C has k + 1 commands $c_1, c_2, \ldots, c_{k+1}$. Let $\hat{C} = c_1, \ldots, c_k$ be an (optimal) B-derivation of $\hat{A}$. The induction hypothesis allows $\hat{C}$ to be replaced by a normal optimal B-derivation of $\hat{A}$, say $C' = c'_1, \ldots, c'_k$. The B-derivation $\bar{C} = c'_1, \ldots, c'_k, c_{k+1}$ is an optimal B-derivation of A. Every copy command $\bar{C}$ is maximal, except possibly c'_k, and c'_k can only be sub-maximal if c_{k+1} is a (maximal) copy command. If $\bar{C}$ is normal we are done; if not, the pair

$$c'_k, c_{k+1} = X(p, \ell_1),\ X(q, \ell_2)$$

can be replaced by the pair

$$c''_k, c''_{k+1} = X(P(i), M(i)),\ X(q+M(i)-\ell_1, \ell_2-M(i)+\ell_1),$$

where i is the index in A of c'_k. Both c''_k and c''_{k+1} are maximal, and it follows that $C'' = c'_1, c'_2, \ldots, c'_{k-1}, c''_k, c''_{k+1}$ is an optimal B-derivation of A which is normal.

QED

IV. A Dynamic Programming Algorithm

In this section, we assume that we know, for given files A and B, the values of the functions M(i) and P(i), $1 \leq i \leq n$, and we derive an algorithm for calculating optimal (and normal) B-derivations of A.

An intuitive heuristic is to work from left to right on A, sending letter commands until a position i is reached with M(i) > K. After sending a copy command of length M(i), the process can be continued starting M(i) symbol to the right in A. The selection of A = 000101 and B = 001000, with K = 2, demonstrates that this heuristic may generate non-optimal derivations. Similarly it is not hard to find "counter-examples" to the simple right to left "first copy > K" heuristic, as well as a procedure which selects the copy command with largest possible length.

Let C(i) be a function defined for $1 \leq i \leq n$ which yields the cost of all optimal B-derivations of the suffix string $A\langle i{:}n\rangle$. In order to calculate efficiently the values of C(i), we need the following final Corollary to Lemma 1.

Corollary 1.3:

Let $C = c_1, c_2, \ldots, c_t$ be an optimal B-derivation of the string $A\langle i{:}n\rangle$. If c_1 is a letter command, then $c_2, \ldots, c_t$ is an optimal derivation of $A\langle i+1{:}n\rangle$. If c_1 is a copy command $X(p,\ell)$, then $c_2, \ldots, c_t$ is an optimal derivation of $A\langle i+\ell{:}n\rangle$.

Proof:

The proof is similar to that of the previous corollaries and is

omitted.

If $c_1, c_2, \ldots, c_t$ is a derivation of A, then since c_1 must be either a letter command or a copy command, Corollary 3 indicates how to calculate the value of C(i) from the values of C(j), $i < j \leq n$. The correctness of Algorithm 1, below, also follows from this corollary. Note that we define $C(n+1) \equiv 0$, and that we assume that the functions M(i) and P(i) are defined elsewhere.

Algorithm 1:

```
1.  C[N+1] := 0;
2.  FOR I := N STEP -1 UNTIL 1 DO
3.  BEGIN CLETTER := 1 + C[I+1];
4.        CCOPY := K + C[I+M(I)];
5.        C[I] := MIN(CLETTER,CCOPY)
6.  END OF LOOP;
```

In order actually to obtain an optimal derivation of $A\langle i{:}n\rangle$, say $c_1, c_2, \ldots, c_t$, we observe that if C[i] = C[i+1] + 1, then c_1 can be selected to be a letter command. The only other possibility, C[i] = C[i+1], implies that c_1 must be selected to be a copy command. These remarks justify the following algorithm for computing an optimal derivation.

Algorithm 2:

```
1.  I := J := 1;
2.  WHILE I ≤ N DO
3.  BEGIN IF C[I] = C[I+1] THEN
4.  BEGIN COMMAND[J] := "X(P(I),M(I)";
5.        J := J+1; I := I + M(I)
6.  END OF THEN CLAUSE;
7.  ELSE BEGIN COMMAND[J] := "A[I]";
8.        J := J+1; I := I+1
9.  END OF ELSE CLAUSE;
10. END OF WHILE STATEMENT;
```

Both algorithms are easily seen to be of O(N) in both time and space; Algorithm 1 executes statements 3 to 5 exactly N times and since the value of I is always increased within the WHILE statement of Algorithm 2, this statement is executed at most N times.

V. Discussion

Our analysis has implicitly assumed that the cost of transmission is proportional to the number of characters sent. In other situations, it is reasonable to try to minimize the total time between the decision to send the file and the time the file is available to the receiving computer.

When a file is sent directly, this time is determined by the smallest transmission rate among the various secondary storage devices (disks, etc.) and the transmission line itself. Since the file must be pre-processed in order to determine a short command sequence, it must presumably be read from a disk into the processor's memory. Consequently, encoding schemes are meaningful only when the rate of transmission over the line is significantly slower than the disk rates; otherwise it would be faster to send the file without extraneous processing.

Note that the linear time bounds of Algorithms 1 and 2 permit us to evaluate their utility in terms of device rates. If, on the other hand, these algorithms ran in time $O(n^2)$, we would find that for large files the processing time at the sending computer would dominate the total transmission time.

Acknowledgement

The idea to use dynamic programming as in Section IV was suggested by Michael J. Fischer during a presentation of this work to an algorithms seminar at MIT. The authors wish to express their appreciation to Dr. Fischer for this idea. Thanks are also due to both Professors Fischer and Albert R. Meyer for several enlightening discussions, as well as to Andrew Sherman of Yale for a careful reading of a preliminary version of this manuscript.

References

[1] Robert A. Wagner and Michael J. Fischer, The String to String Correction Problem, Unpublished Manuscript.

[2] Peter Weiner, Linear Pattern Matching Algorithms, Research Report 17, Department of Computer Science, Yale University, March 1973.