The Complexity of Some Problems on Subsequences and Supersequences

DAVID MAIER

Princeton University, Princeton, New Jersey

ABSTRACT The complexity of finding the Longest Common Subsequence (LCS) and the Shortest Common Supersequence (SCS) of an arbitrary number of sequences is considered We show that the yes/no version of the LCS problem is NP-complete for sequences over an alphabet of size 2, and that the yes/no SCS problem is NP-complete for sequences over an alphabet of size 5

KEY WORDS AND PHRASES computational complexity, NP-completeness, longest common subsequence, shortest common supersequence

CR CATEGORIES 5 23, 5 39

1. *Definitions*

Given a finite sequence $S = s_1, s_2, ..., s_m$, we define a *subsequence* S' of S to be any sequence which consists of S with between 0 and m terms deleted (e.g. ac, ad, and abcd are all subsequences of abcd). We write $S' < S$ if S' is a subsequence of S. We also say that S is a *supersequence* of S', and write $S > S'$. Given a set $R = \{S_1, S_2, ..., S_p\}$ of sequences, we speak of a *Longest Common Subsequence* of R, LCS(R), as a longest sequence S such that $S < S_i$ for $i = 1, ..., p$. For example, abe = LCS({ababe, cabe, abdde}). Actually, LCS(R) is a set of subsequences, since there may be more than one sequence fitting the definition. Since we will be mainly concerned with the length of any (and every) LCS(R), when we write LCS(R) we will mean a single representative of this set. Similarly, a *Shortest Common Supersequence* of R, SCS(R), is a shortest sequence S' such that $S' > S_i$, $I = 1, ..., p$. For example, SCS({abbb, bab, bba}) = abbab.

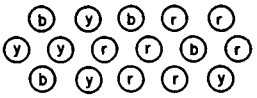
The *yes/no LCS (SCS) problem* is: Given an integer k and a listing of the sequences in R, is $|\text{LCS}(R)| \geq k$ ($|\text{SCS}(R)| \leq k$), where $|S|$ is the number of terms in sequence S? Whenever we refer to the LCS and SCS problems in this paper, we will mean the yes/no versions. We define the *alphabet* of R, $\Sigma(R)$, to be the finite set of values the terms of sequences $S_1, S_2, ..., S_p$ take on. Clearly $|\Sigma(R)| \leq m_1 + m_2 + \quad + m_p$, where $m_i = |S_i|$. We also use $|\ |$ to denote the cardinality of a set; the context will distinguish the usage.

2. *Threading Schemes*

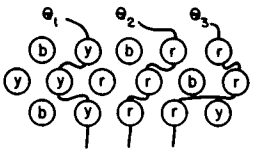
It is convenient to think of the LCS and SCS problems in terms of threading beads. We think of a sequence as a row of beads and the matching process as threading the beads in a certain manner. Suppose we have three sequences S_1 = bybrr, S_2 = yyrrbr, and S_3 = byrry. We represent them as rows of beads:

This work was partially supported by the National Science Foundation under Grant DCR-74-21939

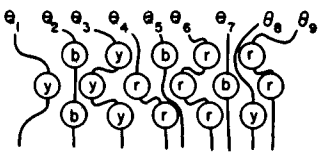
Author's address. Department of Electrical Engineering and Computer Science, Princeton University, Princeton, NJ 08540



For the LCS problem we demand that each thread have exactly one bead from each row, and all the beads on a thread be the same. We stipulate that no two threads may cross. We wish to know if k threads can be used. In our example, k must be three or less:



For the SCS problem, we relax the threading rules so a thread contains at most one bead from each row. We want to find if k threads are sufficient to thread all the beads. All beads on a thread must still be of the same color and threads may not cross. In our example k must be 9 or more:



We refer to a thread by the terms of each sequence it threads, and also designate the type of thread. For example, θ_1 is a y-thread in the LCS example and θ_7 is a b-thread in the SCS example. We refer to a *threading scheme* Θ as a list of threads $\theta_1, \theta_2, \ldots, \theta_j$ which, in the LCS problem, fulfill the threading rules and, in the SCS problem, fulfill the rules *and* thread all the beads. Given a threading scheme $\Theta = \theta_1, \theta_2, \ldots, \theta_j$ for a set of sequences R, we can obtain a common subsequence or common supersequence, depending on the sort of scheme, by reading off the types of $\theta_1, \theta_2, \ldots, \theta_j$. Note that a threading scheme gives rise to a unique sub- or supersequence, but the reverse is false, as there may be a number of threading schemes giving rise to the same sub- or supersequence.

3. *Applications of Threading Schemes to the LCS Problem*

Computing the LCS has found use in the field of molecular biology in studying amino acid sequences in similar proteins [6, 7, 16, 18]. The LCS and SCS problems may also have application to data compression techniques: A number of very similar files might be stored as the LCS or SCS of the files plus modifications for individual files. The complexity of the LCS and similar problems has been analyzed for the case where $|R| = 2$ [1–4, 8–12, 14, 15, 17, 19–24], R being the set of sequences in question. In this paper the LCS and SCS problems are considered with no bound on the size of R, but with various bounds on the size of $\Sigma(R)$. We shall show the SCS problem to be NP-complete for $|\Sigma(R)| \geq 5$ and the LCS problem to be NP-complete for $|\Sigma(R)| \geq 2$.

All of the proofs of NP-completeness will be by reduction of the *node cover problem* [5, 13]. Given an undirected graph $G = (N, E)$ and an integer k, the node cover problem is to determine if there is an N' contained in N, with $|N'| = k$, such that for every $(x, y) \in E$, either $x \in N'$ or $y \in N'$ (possibly both). We assume the problem is posed as an integer k and a list of edges in E: k; (x_1, y_1); (x_2, y_2); ...; (x_r, y_r). We now prove three theorems, each having successively stronger results. The reason for the redundant theorems is to develop the proof in stages which can be more easily grasped.

THEOREM 1. *The LCS problem is NP-complete for $\Sigma(R)$ of arbitrary size.*

PROOF. Given an instance of the node cover problem on the graph $G = (N, E)$: k; (x_1, y_1); (x_2, y_2); ...; (x_r, y_r), encoded into a string of length n, we find N and assign an arbitrary order $\{v_1, v_2, \ldots, v_t\}$ to N. Clearly, $r, t \leq n$. We construct $r + 1$ sequences of length

at most $2(t-1)$ as shown in Figure 1. The first sequence is the *template* T, which is the sequence $v_1, v_2, \ldots, v_t$. For each edge $e_i = (x_i, y_i)$ in E we construct a sequence S_i. Assume without loss of generality that $x_i = v_j$, $y_i = v_m$, and $j < m$. Then S_i is $v_1, v_2, v_3, \ldots, v_{j-1}, v_{j+1}, \ldots, v_m, \ldots, v_t, v_1, v_2, \ldots, v_j, \ldots, v_{m-1}, v_{m+1}, \ldots, v_t$

CLAIM. *The graph G has a node cover of size k if and only if the set $R = \{T, S_1, S_2, \ldots, S_r\}$ has a common subsequence of size $t-k$.*

PROOF (only if). Let $V' = \{u_1, u_2, \ldots, u_k\}$ be the node cover of size k. Then the sequence T', which is T with the nodes of V' deleted, has size $t-k$. Clearly $T' < T$. For each S_i either $x_i \in V'$ or $y_i \in V'$. If $x_i \in V'$, then T' is a subsequence of the first half of S_i, since every node in T' appears in the first half of S_i, and the elements of T' and the first half of S_i are in the same order. If $y_i \in V'$, then T' is a subsequence of the last half of S_i. So T' is a common subsequence of R with length $t-k$.

(if). Let T' be a common subsequence of R. We notice that for each i, T' cannot contain both x_i and y_i. To see this, let $x_i = v_j$ and $y_i = v_m$, where we have assumed $j < m$. For both v_j and v_m to be in T', in any threading scheme we must have threads running through v_j and v_m in T. Also, we have only one choice of where to run the threads in S_i, since it also has only one occurrence of v_j and v_m. But this cannot be done without crossing threads, since the order of v_j and v_m is different in the two sequences. (See Figure 2.) Let $V' = \{u_1, u_2, \ldots, u_k\}$ be the nodes in T but not in T'. We are given $|T'| = t-k$, so $|V'| = k$. For each S_i, either x_i or y_i is in V'. Hence for every edge (x_i, y_i) in E of graph G, $x_i \in V'$ or $y_i \in V'$. Hence V' is a node cover of size k.

PROOF OF THEOREM 1 CONTINUED. From the claim it is apparent that the minimal node cover of G has size k if and only if LCS(R) has size $t-k$. If the node cover problem has length n, the input for the LCS algorithm is of length $t + 2r(t-1) \leq O(n^2)$. It is not hard to see that the construction can be done in polynomial time. Therefore we have a polynomial reduction of the node cover problem to the LCS problem.

THEOREM 2. *The LCS problem is NP-complete for $|\Sigma(R)| = 3$.*

PROOF. Here we use an encoding for the nodes of the graph. In Theorem 1 the main point is that for each edge $e_i = (x_i, y_i)$ in E, the LCS(R) cannot contain both x_i and y_i, and this point is preserved in the encoding. We also encode the edges, for use in distinguishing among the sequences.

In Figure 3 we see how the codes are laid out. The codes are over the alphabet

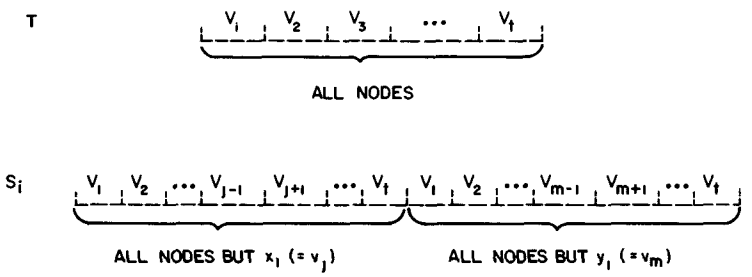


FIG 1

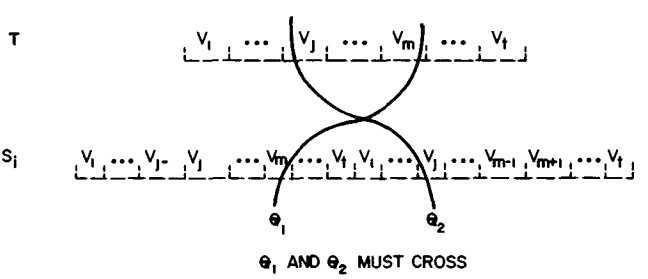


FIG 2

$\{1, 0, *\}$. There are two codes which go into the template, called *codeplates*. The node codeplate $\bar{N}$ consists of $t + 1$ blocks of $4r + 2t$ ones, separated by stars. The edge codeplate $\bar{E}$ has $r + 1$ blocks of $4r + 2t$ zeros, separated by two stars. The code for a node $v_i \in N$ will be denoted $\bar{N}[i]$, and will be obtained by deleting the ith star of $\bar{N}$. We also define a multiple node code for $v_{i_1}, v_{i_2}, \ldots, v_{i_s} \in N$, denoted $\bar{N}[i_1, i_2, \ldots, i_s]$, to be obtained by deleting the i_1st, i_2nd, ..., i_sth stars from $\bar{N}$. $\bar{E}[j]$ and $\bar{E}[j_1, j_2, \ldots, j_s]$ are obtained by analagous deletions of pairs of stars from $\bar{E}$.

Given an instance of the node cover problem on a graph $G = (N, E)$: k; (x_1, y_1); (x_2, y_2); ...; (x_r, y_r), let us construct a set R of $r + 1$ sequences as shown in Figure 4. Our template T will be the edge codeplate $\bar{E}$, followed by the node codeplate $\bar{N}$, followed by another $\bar{E}$. We will distinguish the two $\bar{E}$'s as the left and right $\bar{E}$'s. For the ith edge $e_i = (x_i, y_i)$, let $x_i = v_j$ and $y_i = v_m$, where $\{v_1, v_2, \ldots, v_t\}$ is an ordering of N. We create a sequence S_i which is the code for edge e_i, $E[i]$; the code for v_j, $N[j]$; the edge codeplate, $\bar{E}$; the code for v_m, $\bar{N}[m]$; and a second occurrence of $\bar{E}[i]$. Again, we distinguish the left $\bar{E}[i]$ and the right $\bar{E}[i]$.

CLAIM. *If graph $G = (N, E)$ has a node cover of size k then the set R has a common subsequence of size*

$$(2r + t + 3)(4r + 2t) + (2r + t - k).$$

PROOF. Let $N' = \{u_1, u_2, \ldots, u_k\}$ be a node cover of size k. Let W be the set of all edges $e_i = (x_i, y_i)$ such that $x_i \in N'$. Let U be the set of the rest of the edges in N. Clearly, for every e_i in U, $y_i \in N'$. Let T' be the sequence $\bar{E}[W]$; $\bar{N}[N']$; $\bar{E}[U]$. (See Figure 5.) T' is of length $(2r + t + 3)(4r + 2t) + (2r + t - k)$, since $|W| + |U| = r$. In Figure 6 we show how

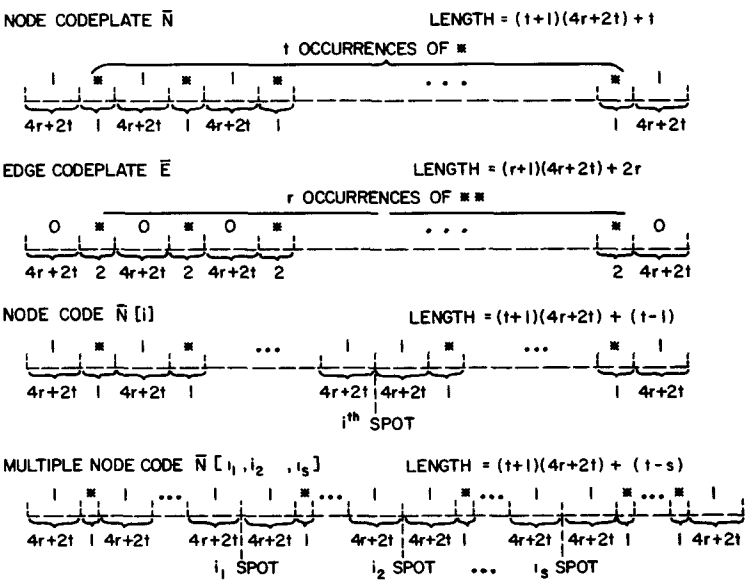


FIG 3

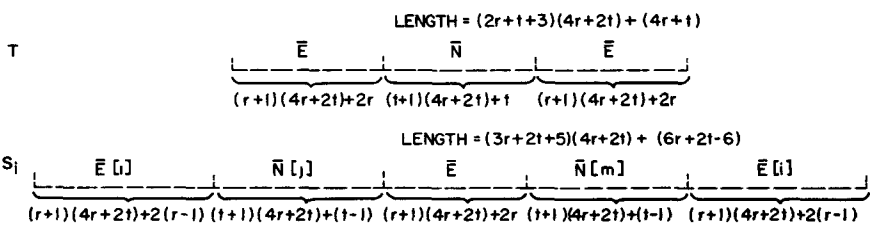


FIG. 4

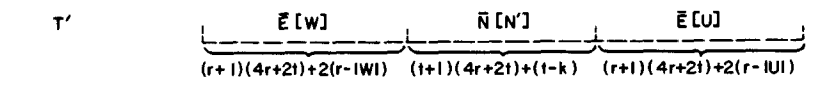


FIG. 5

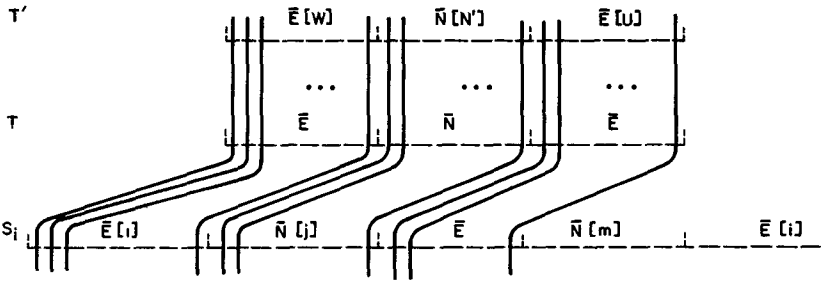


FIG. 6

T' will thread with T and with the sequence S_i for edge $(x_i, y_i) = (v_j, v_m)$, assuming $x_i \in N'$. Corresponding sections of T' and T thread because T' is simply T with some of the stars deleted. T' threads with the left side of S_i as follows: $\bar{E}[W]$ threads with $\bar{E}[i]$ since the only pair of stars missing in $\bar{E}[i]$ is also missing in $\bar{E}[W]$. Similarly, $\bar{N}[N']$ threads with $\bar{N}[j]$ since the jth star of $\bar{N}[N']$ is missing. Finally, $\bar{E}[U]$ threads with $\bar{E}$, since $\bar{E}[U]$ is $\bar{E}$ with some pairs of stars deleted. So T' is a common subsequence of R with the desired length.

CLAIM. *If LCS(R) has length*

$$(2r + t + 3)(4r + 2t) + (2r + t - k),$$

then the graph G has a node cover of size k.

PROOF. We need a preliminary lemma and corollary.

LEMMA. *Given any common subsequence T' of R, there exists a common subsequence T'' of R with $|T''| \geq |T'|$, such that T'' has a threading scheme which threads entire blocks of $4r + 2t$ zeros or ones in the sequences of R. That is, under this threading scheme, in any block of zeros or ones in a sequence, either all the zeros or ones are threaded, or none are threaded.*

PROOF. We will show the lemma holds for the blocks of zeros. The proof is merely a process of pushing threads to the left within blocks of ones and adding more threads. Suppose we have a threading scheme Θ for T', with θ_1 being the leftmost 0-thread, and let $B_0, B_1, B_2, \ldots, B_r$ be the blocks of zeros which θ_1 threads. (See Figure 7(a).) Since θ_1 is the leftmost 0-thread, we can move it so it threads the leftmost zero in each of $B_0, B_1, \ldots, B_r$. (See Figure 7(b).) We then add thread θ_2 running through the second leftmost zero of $B_0, B_1, \ldots, B_r$. Notice that θ_2 can conflict with at most one existing thread. Suppose there were two threads, θ_a and θ_b, which already threaded second leftmost zeros in two different blocks, say B_i and B_j, respectively. Then in sequence S_i, θ_a would be to the left of θ_b, while in S_j, θ_a has to thread to the right of θ_b. This would mean θ_a and θ_b crossed somewhere between S_i and S_j, which is disallowed by our threading rules.

Should we find such a thread conflicting with θ_2, we eliminate it. We continue by adding threads $\theta_3, \theta_4, \ldots, \theta_{4r+2t}$ in the same manner, never decreasing the number of threads in θ. Now we find the next 0-thread to the right of θ_{4r+2t}, call it θ'_1, and repeat the process, realizing that θ'_1 cannot pass through blocks $B_0, B_1, \ldots, B_r$. We continue in this manner until all 0-threads run through blocks which are completely threaded.

We perform a similar process on the 1-threads, and derive our common subsequence T'' from the new threading scheme, with $|T''| \geq |T'|$.

COROLLARY. *There exists an LCS for R with a threading scheme which threads all the zeros and ones in T.*

PROOF First we note that

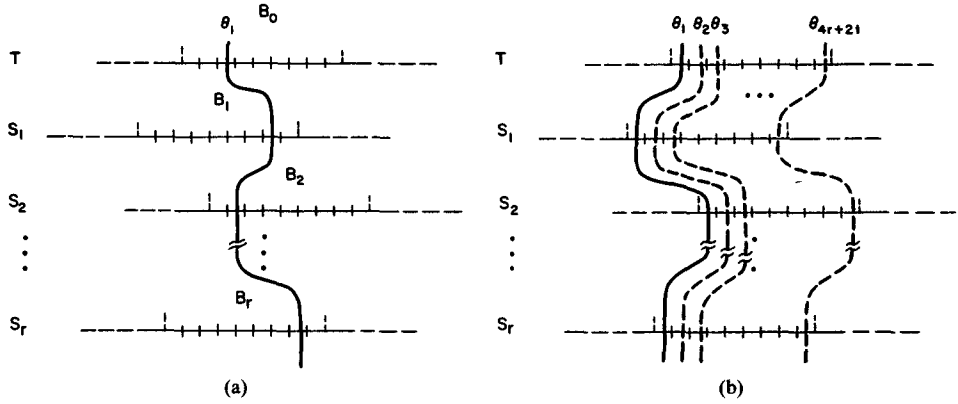


FIG 7

$$\bar{E}[1, 2, ..., r]; \bar{N}[1, 2, ..., t]; \bar{E}[1, 2, ..., r] = 1^{(r+1)(4r+2t)}\, 0^{(t+1)(4r+2t)}\, 1^{(r+1)(4r+2t)}$$

is a common subsequence of R of size $(2r + t + 3)(4r + 2t)$. Suppose T' is an LCS of R. We apply the above lemma to T' to get an LCS T'' with a threading scheme Θ which threads only whole blocks of zeros and ones. Suppose Θ does not thread some block of zeros or block of ones in T, the template. Then the maximum length of T'' is $(2r + t + 2)(4r + 2t) + (4r + t)$, since the blocks have length $4r + 2t$. But this makes T'' shorter than $(2r + t + 3)(4r + 2t)$ and therefore not an LCS of R. Contradiction. So the scheme Θ for T'' must thread all the zeros and ones in T.

PROOF OF CLAIM CONTINUED. Now that we know an LCS T'' for R matches all the zeros and ones in T, we can demonstrate some other properties of T''. First we note that there is no threading scheme Θ for T'' which contains threads θ_a and θ_b threading the left $\bar{E}$ and the right $\bar{E}$ of T and also the $\bar{E}$ of any S_i, since this would prevent the ones in $\bar{N}$ of T from being threaded. (See Figure 8.) Hence for any threading scheme Θ and sequence S_i, it must either be the case that there are $(r + 1)(4r + 2t)$ 0-threads, $\theta_1, \theta_2, ..., \theta_s$, threading all the zeros in the left $\bar{E}$ and the left $\bar{E}[i]$ or $(r + 1)(4r + 2t)$ 0-threads $\theta'_1, \theta'_2, ..., \theta'_s$, threading all the zeros in the right $\bar{E}$ and the right $\bar{E}[i]$, as shown in Figure 9. This means the ith pair of stars in the left or right $\bar{E}$ of T is not threaded, though not necessarily both pairs.

Suppose for a given S_i all these 0-threads run through the left $\bar{E}$ and $\bar{E}[i]$. (See Figure 10.) Then there cannot be threads θ_a and θ_b both passing through $\bar{N}$ of T and then through $\bar{N}[j]$ and $\bar{N}[m]$ of S_i, respectively. This would mean the ith pair of stars of the right $\bar{E}$ in T would go unmatched. But more threads could be run, as shown in Figure 11, by threading the left $\bar{E}$ and $\bar{E}[i]$ together, $\bar{N}$ and $\bar{N}[j]$ together, and the right $\bar{E}$ of T and $\bar{E}$ of S_i together. This would mean the ith star of $\bar{N}$ goes unmatched. A symmetrical situation applies if $\theta'_1, \theta'_2, ..., \theta'_s$ thread the right $\bar{E}$ and $\bar{E}[i]$, with the final result given in Figure 12.

What we see from these constraints on Θ is that for each S_i in R, the ith pair of stars in the left $\bar{E}$ or the right $\bar{E}$ in T goes unmatched and the jth or mth star in $\bar{N}$ goes unmatched. Now if T'', our LCS, is of size $(2r + t + 3)(4r + 2t) + (2r + t - k)$, we know that $(2r + t + 3)(4r + 2t)$ threads match zeros and ones in T, leaving $2r + t - k$ threads for stars. There are $4r + t$ stars in T, and we know that for each S_i in R we can match the ith pair of stars in the left $\bar{E}[i]$, but not in the right $\bar{E}[i]$, or vice versa. So we have $2r$ stars matched in the left and right $\bar{E}$'s of T, leaving $t - k$ stars matched in $\bar{N}$ of T. There are t stars in $\bar{N}$, so k of these stars go unthreaded under the scheme Θ. By letting N' be the set of nodes corresponding to those stars not matched in $\bar{N}$ of T, we can show that N' is a node cover of graph G of size k, by methods similar to Theorem 1.

PROOF OF THEOREM 2 CONTINUED The two claims above suffice to show that LCS(R) has size $(2r + t + 3)(4r + 2t) + (2r + t - k)$ if and only if the graph G has a minimal node cover of size k. Given an input for the node cover problem of length n, we

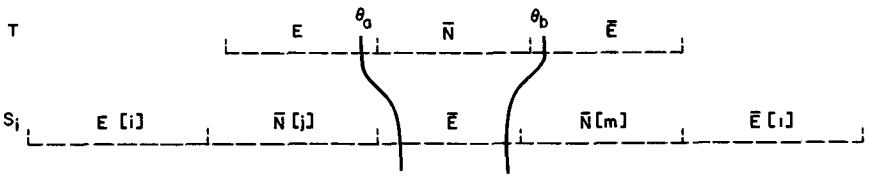


FIG. 8

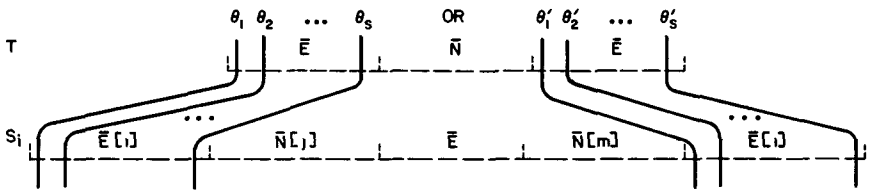


FIG. 9

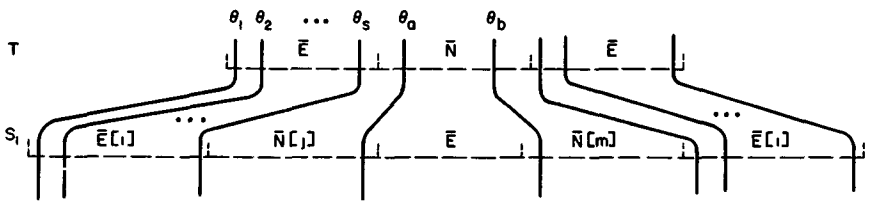


FIG 10

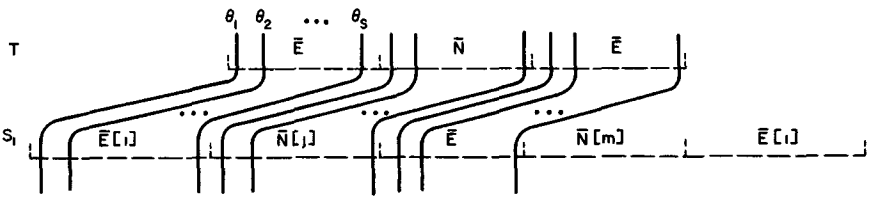


FIG 11

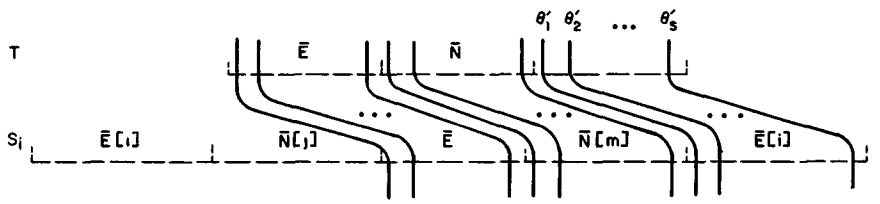


FIG 12

must construct $r + 1$ sequences of length less than or equal to $(3r + 2t + 5)(4r + 2t) + (6r + 2t - 6)$ to use as input for the LCS algorithm, where $r, t \leq n$. So the total length of the input for the LCS problem is $O(n^3)$, and it can be seen that the sequences of R can be generated in polynomial time. Therefore the LCS problem with $|\Sigma(R)| \geq 3$ is NP-complete.

THEOREM 3. *The LCS problem is NP-complete for* $|\Sigma(R)| \geq 2$.

PROOF. The proof of this theorem is essentially that of Theorem 2, but we eliminate the use of stars, replacing them by zeros and ones as appropriate. (See Figure 13). The proof of Theorem 2 now carries through with the changes of stars to zeros and ones, except for one difficulty. The problem arises in the lemma, where the proof becomes invalid with the changes of symbols. We present an alternate lemma which will replace the former lemma and its corollary. All our notation will be the same as in Theorem 2, except as noted.

LEMMA. *There exists an LCS T'' of R with a threading scheme which fully threads all the blocks in T of R.*

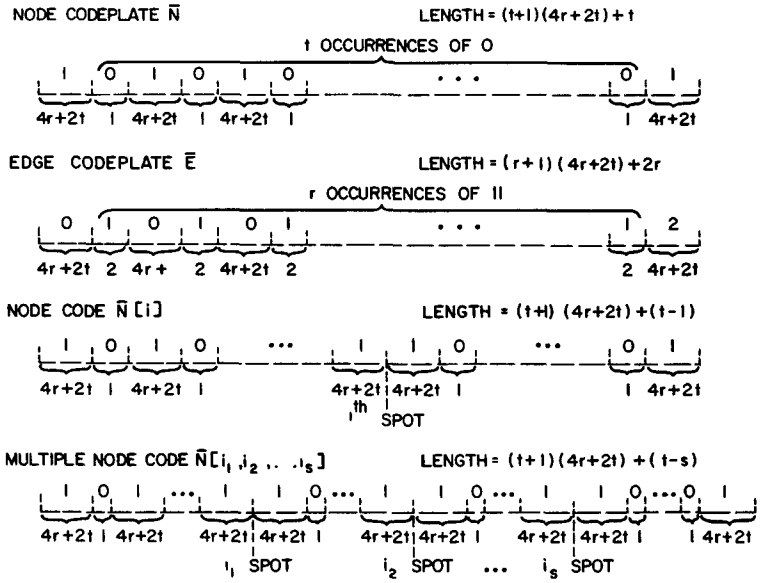


FIG 13

PROOF. Since we now have zeros and ones occurring in places other than in blocks, we will also write of 0-*dividers* (of length 1) and 1-*dividers* (of length 2), these being the zeros and ones which replaced the stars of Theorem 2. Next we note that $\bar{E}[1, 2, \ldots, r]$; $\bar{N}[1, 2, \ldots, t]$; $\bar{E}[1, 2, \ldots, r]$ is a common subsequence of R of length $(2r + t + 3)(4r + 2t) + 2r$; it threads similarly to T' in Figure 6. So any LCS of R must have at least this length, and must therefore thread at least $(2r + t + 2)(4r + 2t) + (2r + t)$ terms of the blocks of T, since there are only $4r + t$ terms in the dividers. In other words, only $2r + t$ terms in blocks of T may go unthreaded, which implies that every block of T must have at least $2r + t$ terms threaded in any scheme for an LCS of R.

Given a threading scheme for an LCS of R, we will show that we can rearrange the threads so that all blocks of T in the left $\bar{E}$ thread completely with entire blocks in S_1, S_2, ..., S_r. A similar approach works for the blocks of $\bar{N}$ and the right $\bar{E}$ of T. We first note that all the sequences in R begin with a 0-block We thread these 0-blocks together, which we can do without decreasing the number of threads. We now look at the next 0-block to the right in the left $\bar{E}$, and attempt to thread it with 0-blocks in S_1, S_2, ..., S_r, and proceed to the right.

We may encounter a hitch, however. Examine Figure 14. We are working on the 0-block B_0 of T, and we know that all the blocks to the left of B_0 thread completely with 0-blocks in the rest of the sequences. The last 0-thread of the previous block we call θ_p. The threads through the 1-divider immediately before B_0 we call θ_a and θ_b, if they exist. The leftmost and rightmost threads in B_0 we call θ_u and θ_v, respectively. Since B_0 must have at least $2r + t$ threads, in each of the S_i some of the threads $\theta_u - \theta_v$ must either pass through or *encompass* (pass to the left and right of) at least one 0-block, since $\bar{N}[x_i]$ and $\bar{N}[y_i]$ have only $t - 1$ zeros apiece.

We can have one of three cases with each of the S_i; these cases are exemplified by sequences S_f, S_g, and S_h Sequence S_f represents the case where there is a block or a portion of a block B_f between θ_v and θ_p (or θ_b instead, if it exists), where no threads from other 0-blocks of T run through B_f. The case for S_g is where all the threads $\theta_u - \theta_v$ pass through a single block B_g, there are no 0-blocks between θ_u and whichever of θ_p or θ_b is closer, and there exist some threads to the right of θ_v in B_g. The case for S_h is that of S_g, except that θ_u and some threads to its right run through 0-dividers instead of B_h.

If we have a case similar to S_f, we simply push all the threads $\theta_u - \theta_v$ into B_f. The case S_g is harder to handle, and will be dealt with in detail. The case for S_h is very much the

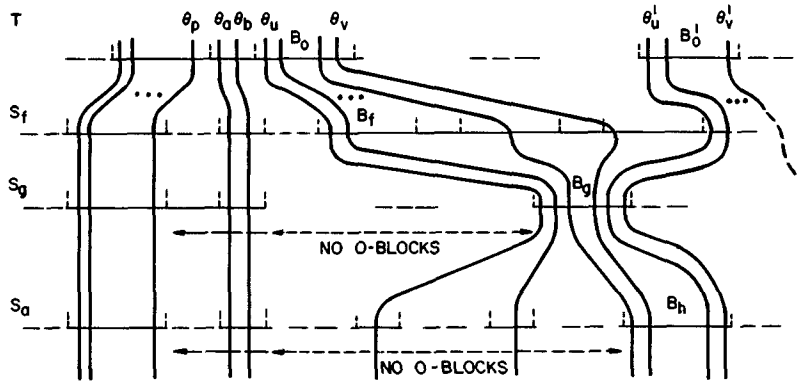


FIG 14

same as the case for S_g, and will not be given. Going back to S_g, consider the thread to the immediate right of θ_v in B_g. This thread is of course to the immediate right of θ_v in T as well, and a moment's careful thought will show that this thread must pass through the next 0-block to the right of B_0 in the left $\bar{E}$. We call this block B_0^1, with leftmost and rightmost threads θ_u^1 and θ_v^1.

The situation for S_g may be further complicated, as we see in Figure 15. Here we have 0-blocks B_0^j, $j = 0, \ldots, s$, in the left $\bar{E}$ of T, with leftmost and rightmost threads θ_u^j and θ_v^j, and 0-blocks B_g^j, $j = 0, \ldots, s$, in S_g (with possibly no S_g^s). The blocks B_0^0, B_0^1, and B_g^0 correspond to B_0, B_0^1, and B_g in Figure 14. The following conditions hold:

(1) B_0^{j-1} and B_0^j are adjacent blocks of T for $j = 1, \ldots, s$.

(2) θ_v^{j-1} and θ_u^j both thread B_g^{j-1} for $j = 1, \ldots, s-1$.

(3) B_g^{j-1} and B_g^j are adjacent 0-blocks in S_g, or they are separated by $\bar{N}[x_g]$, for $j = 1, \ldots, s$. (They cannot be separated by $\bar{N}[y_g]$, for this would make it impossible to thread the 1-blocks in $\bar{N}$ of T.)

(4) B_0^s is the first block to the right of B_0^0 where the threads $\theta_u^s - \theta_v^s$ do not share a block or divider in S_g with any 0-threads to the right of θ_v^s.

Note that B_0^s must occur before $\bar{N}$ of T, since the first 1-block of $\bar{N}$ must have a 1-thread, and our conditions preclude a 1-thread between θ_u^0 and θ_v^s.

Our conditions allow for four cases, as to where θ_v^s runs in S_g.

Case I. The thread θ_v^s runs though the block B_g^s in S_g. (See Figure 16.) We know there are no threads to the right of θ_v^s in B_g^s. So we move threads as follows:

$\theta_u^s - \theta_v^s$ run through B_g^s by themselves.
$\theta_u^{s-1} - \theta_v^{s-1}$ run through B_g^{s-1} by themselves.
$\vdots$
$\theta_u^1 - \theta_v^1$ run through B_g^1 by themselves.
$\theta_u^0 - \theta_v^0$ run through B_g^0 by themselves.

Case II. The thread θ_v^s runs through block B_g^{s-1}. (See Figure 17.) This means all the threads $\theta_u^0 - \theta_v^s$ will fit in the blocks B_g^0 through B_g^{s-1}, plus possibly the 0-dividers in $\bar{N}[x_g]$, of S_g. But this is only $s(4r + 2t) + 2r$ terms. So B_0^0 through B_0^s must have $2r + 2t$ unthreaded terms, which is more than the $2r + t$ allowed. This means our threading scheme cannot correspond to an LCS for R.

Case III. The thread θ_v^s runs through a 0-divider in $\bar{N}[x_g]$, and there is a θ_w between θ_u^s and θ_v^s which threads through B_g^s. (See Figure 18.) Here we proceed as in case I.

Case IV. The thread θ_v^s runs through a 0-divider in $\bar{N}[x_g]$, but there is no block B_g^s between θ_u^s and θ_v^s. (See Figure 19.) The argument from case II can be applied.

We now are able to thread $\theta_u - \theta_v$ through the same block in S_f, S_g, and S_h, with no other threads passing through these blocks. These sequences typify the cases in all the S_i, so we may run $\theta_u - \theta_v$ through blocks by themselves in all the S_i. It then becomes a simple

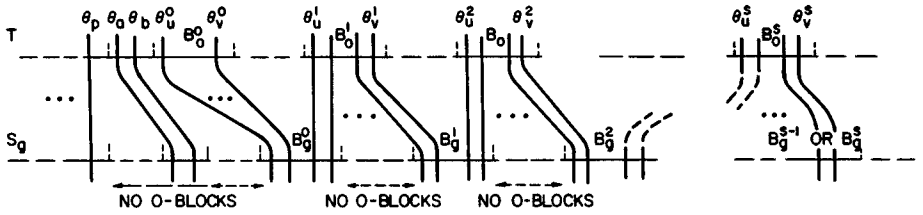


FIG 15

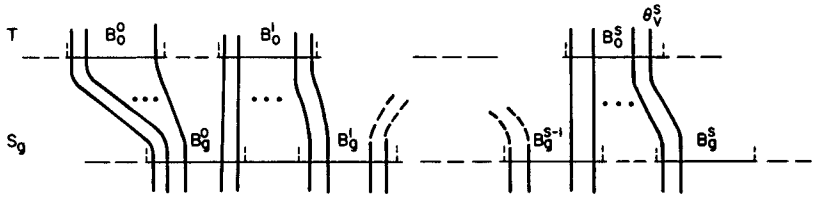


FIG 16

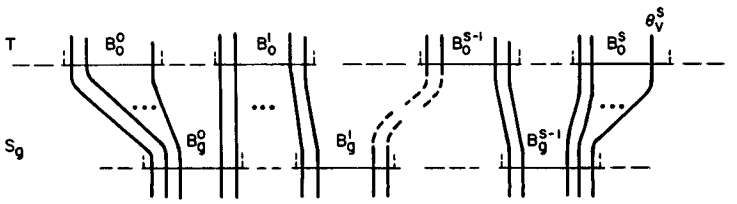


FIG 17

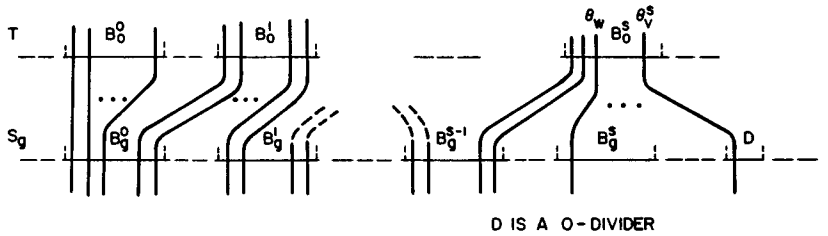


FIG 18

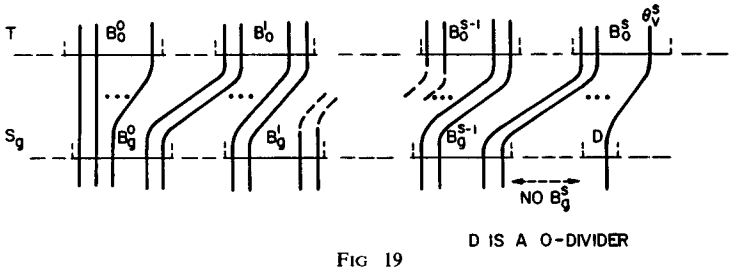


FIG 19

process to add more threads, if necessary, through these blocks to bring the count up to $4r + 2t$. We thus fully thread all the 0-blocks in the left $\bar{E}$ of T. A similar method works for the 1-blocks of $\bar{N}$ and the 0-blocks of the right $\bar{E}$ of T. This leads us to the desired result.

PROOF OF THEOREM 3 CONTINUED. With the above lemma, the proof of Theorem 2, minus the lemma and corollary, is adequate for the proof of Theorem 3.

4 *Applications of Threading Schemes to the SCS Problem*

The proofs we will present dealing with the Shortest Common Supersequence (SCS) problem are fairly similar to those for the LCS theorems; hence we will not go into as much detail. We will prove two theorems, the second stronger than the first, in order to develop the proof in stages.

THEOREM 4. *The SCS problem is NP-complete for $\Sigma(R)$ of arbitrary size*

PROOF. Once again we reduce the node cover problem to the problem at hand. Given a node cover problem on a graph $G = (N, E)$, say, k; (x_1, y_1); (x_2, y_2); ...; (x_r, y_r), we construct a set R of $r + 1$ sequences. The alphabet for R will be as follows. Find the set of nodes in G, $N = \{v_1, v_2, ..., v_t\}$, and make each an alphabet member. The edges of $E = \{e_1, e_2, .., e_r\}$, where $e_i = (x_i, y_i)$, are also members of the alphabet. Finally, star $(*)$ is a member of the alphabet.

We then construct the $r + 1$ sequences. The template T is composed of six sections: $\bar{N}$; $\bar{A}$; $\bar{E}$; $\bar{E}$; $\bar{A}$; $\bar{N}$, in that order. $\bar{N}$ is a list of the nodes in N. $\bar{E}$ is a list of the edges in E, with each edge appearing twice in a row. The section $\bar{A}$ is a sequence of $4c$ stars, where $c = \max(r, t)$. (See Figure 20) For each edge $e_i = (x_i, y_i)$ we construct a sequence S_i which is $e_i e_i$; v_j; $\bar{A}$; v_m; $e_i e_i$, where $x_i = v_j$, $y_i = v_m$, and $\bar{A}$ is $4c$ stars, as before.

CLAIM. *If G has a node cover of size k, then R has a common supersequence of length $8c + 6r + 2t + k$*

PROOF. Let $N' = \{u_1, u_2, ..., u_s\}$ be a node cover of size k. Let $W = \{e_i | x_i \in N'\}$, and $U = \{e_j | e_j$ is not in $W\}$. Clearly if $e_i \in U$, then $y_i \in N'$. We now construct a sequence T' by augmenting T with three sections: $\bar{W}$ at the beginning, $\bar{N}'$ between the two $\bar{E}$'s, and $\bar{U}$ at the end, where $\bar{W}$ is a list of the edges in W, with each edge twice, $\bar{N}'$ is a list of the nodes in N', and $\bar{U}$ is two occurrences of each edge in U. (See Figure 21) T' has length $8c + 6r + 2t + k$. To see that T' is a common supersequence of R, note that for each S_i, we can match $x_i = v_j$ or $y_i = v_m$ to a term in $\bar{N}'$. The rest of the matching follows in a straightforward manner.

CLAIM. *If R has an SCS of length $8c + 6r + 2t + k$, then G has a node cover of size k.*

PROOF. First we note that for any node cover N' of G of size s, there is a supersequence T' of R with length $8c + 6r + 2t + s$, formed in the manner of the T' in the claim above. Further, since $s \leq t$, $8c + 6r + 2t + s \leq 8c + 6r + 3t$, and since N is a node cover of G, any SCS of R has length less than $8c + 6r + 3t$. Let T'' be an SCS of R. We can now prove the following lemma.

LEMMA. *There is a threading scheme Θ'' for T'' such that for all $S_i \in R$, all the stars of $\bar{A}$ of S_i are threaded with the left $\bar{A}$ of T or the right $\bar{A}$ of T.*

PROOF. It is a property of SCS threading schemes that all terms of all sequences get threaded. Suppose for a given S_i, none of the stars in $\bar{A}$ thread with either $\bar{A}$ of T. Then Θ'' has at least $12c + 4r + 2t$ threads, which is more than the $8c + 6r + 3t$ allowed for an SCS. We get the former number by counting the terms in T $(8c + 4r + 2t)$ plus the terms in $\bar{A}$ of S_i, all of which need separate threads. So at least one thread goes through $\bar{A}$ of S_i and one of the $\bar{A}$'s of T. But once we have one common thread, we can shift and add threads to thread all the stars of $\bar{A}$ of S_i and all the stars of either the left or the right $\bar{A}$ of T. (See Figure 22.)

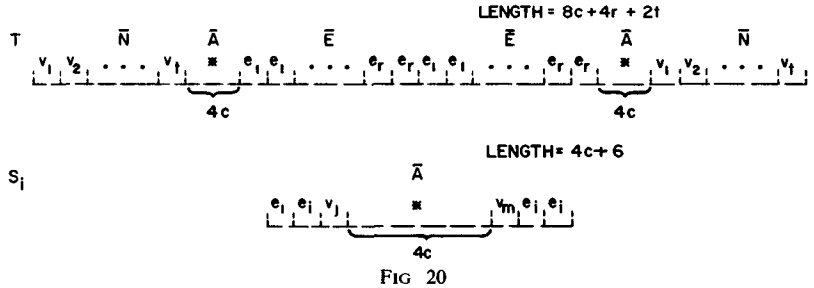


FIG 20

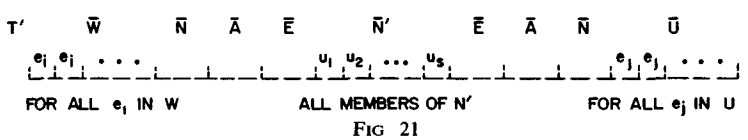


FIG 21

PROOF OF CLAIM CONTINUED. Now it remains to be determined how the rest of S_t is threaded. We will define *extra threads* as those threads in Θ'' threading no term in T. Suppose we have A of S_t threading completely with the left A of T, as shown in Figure 22. Then there are no terms in T or any other sequence to thread with the left pair of e_t's, so they must be on their own extra threads, which we can run around the left end of T. (See Figure 23.) This allows us to thread v_j with the corresponding term in $\bar{N}$ of T. On the right end, it is not difficult to see that we can do no better than running v_m on an extra thread between the $\bar{E}$'s and matching e_te_t to the right $\bar{E}$. If $\bar{A}$ of S_t threads the right $\bar{A}$ of T, we end up with a mirror image scheme.

For each S_t, we must run two extra threads for e_te_t and one extra thread for x_t or y_t (v_j or v_m). If $x_t = x_j$ for some j, and x_t and x_j are on extra threads, these threads can be combined; likewise for $x_t = y_j$, $y_t = x_j$, or $y_t = y_j$ for some j. We see that Θ'' will have $2r$ extra threads for the e_t's and a number of extra threads for some of the members of N, such that x_t or y_t is on an extra thread for each $e_t = (x_t, y_t)$ Since $|T''| = 8c + 6r + 2t + k$ and we have $8c + 4r + 2t$ threads for T, plus $2r$ extra threads for the e_t's, there must be k extra threads for the elements of N. If we let $N' = \{v_j \in N | v_j$ corresponds to an extra thread$\}$, we see that N' is a node cover for G of size k.

PROOF OF THEOREM 4 CONTINUED. The two claims above give us a reduction of the node cover problem to the SCS problem. The length of the sequences for R is polynomial in n, the length of the node cover input. The reduction is polynomial, so the SCS problem is NP-complete for $|\Sigma(R)|$ unbounded.

THEOREM 5. *The SCS problem is NP-complete for* $|\Sigma(R)| = 5$.

PROOF. We use the same notation for our graph and node cover problem as we used in the last theorem. Again, we do an encoding for the nodes and edges, using the alphabet {a, b, 0, 1, *}. The encoding is similar to the encoding used for the LCS problem, except that we insert dividers instead of deleting them. We define the node codeplate $\bar{N}$ as $t + 1$ blocks of $10c$ a's, where $c = \max(r, t)$. Any v_t in N we encode with node code $\bar{N}[i]$, which is obtained by inserting a b between the ith and $(i + 1)$-st blocks of $\bar{N}$. The multiple node code $\bar{N}[i_1, i_2, ..., i_s]$ has a b in the i_1st, i_2nd, ..., i_sth spots. The special case of $\bar{N}[1, 2, ..., t]$ we denote $\bar{N}_s$ and refer to as the *node sink*, since it is a supersequence of all the node codes, as well as the node codeplate. The edge codeplate $\bar{E}$, the edge code $\bar{E}[j]$, and the multiple edge code $\bar{E}[j_1, j_2, ..., j_s]$ are defined similarly with blocks of $10c$ zeros and pairs of ones. (The code $\bar{E}[j]$ is shown in Figure 24.) We call $\bar{E}[1, 2, ..., r]$ the *edge sink* and denote it $\bar{E}_s$. Finally, $\bar{A}$ consists of $10c$ stars.

We define the $r + 1$ sequences of R as follows. The sequences are somewhat similar to the sequences of R in Theorem 4. The template T consists of the following codes in the given order: $\bar{E}$; $\bar{N}_s$; $\bar{A}$; $\bar{E}_s$; $\bar{N}$; $\bar{E}_s$; $\bar{A}$; $\bar{N}_s$; $\bar{E}$. For each $e_t = (x_t, y_t)$ we define S_t as. $\bar{E}[i]$; $\bar{N}[j]$; $\bar{A}$, $\bar{N}[m]$; $\bar{E}[i]$, where $x_t = v_j$ and $y_t = v_m$. (See Figure 25.) Note that T has length $10c(4r + 3t + 9) + (4r + 2t)$.

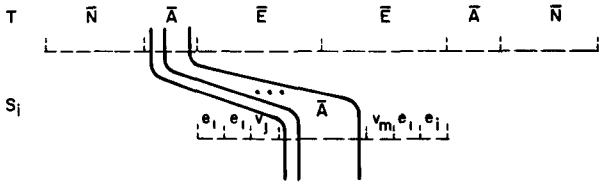


FIG 22

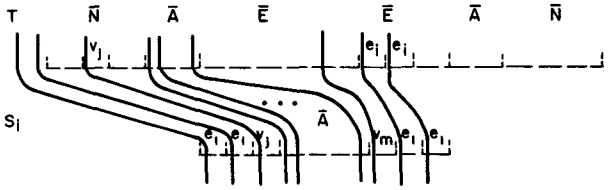


FIG. 23

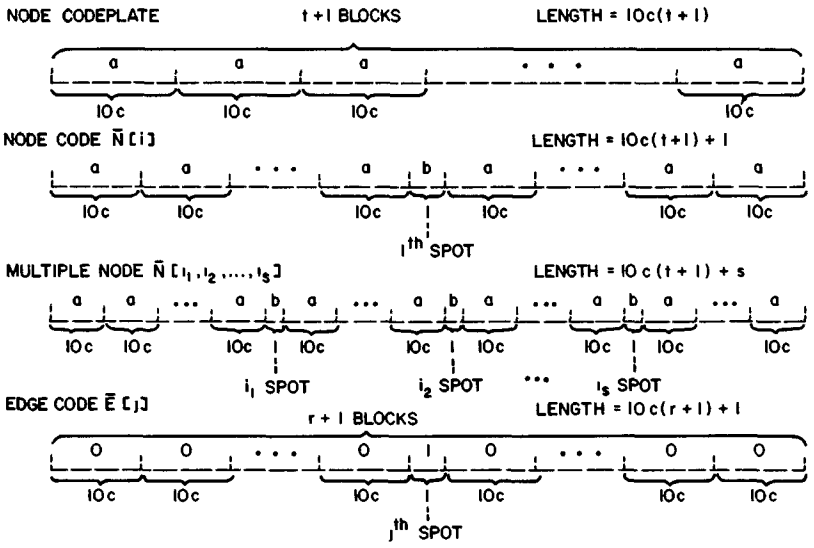


FIG 24

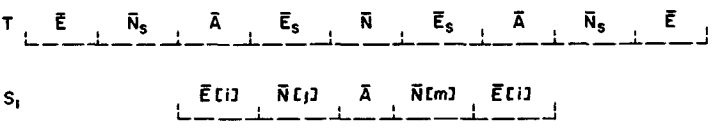


FIG 25

CLAIM. *If G has a node cover of size k, then R has a common supersequence of length*

$$10c(4r + 3t + 9) + (4r + 2t) + (2r + k).$$

PROOF. Given N', a node cover of size k, let W and U be as specified in Theorem 4. Then

$$T' = \bar{E}[W]; \bar{N}_s; \bar{A}; \bar{E}_s; \bar{N}[N']; \bar{E}_s; \bar{A}; \bar{N}_s; \bar{E}[U]$$

is a common supersequence of R of length

$$10c(4r + 3t + 9) + (4r + 2t) + (2r + k).$$

The matching is analogous to that of Theorem 4, with either v_j or v_m threading with $\bar{N}[N']$.

CLAIM. *If R has an SCS of length*

$$10c(4r + 3r + 9) + (4r + 2t) + (2r + k),$$

then G has a node cover of size k.

PROOF. Given any node cover N' for G of size s, we can construct T', a supersequence for R of length

$$10c(4r + 3t + 9) + (4r + 2t) + (2r + s) \leq 10c(4r + 3t + 9) + (4r + 2t) + (2r + t),$$

as above. The right-hand side of the inequality is thus an upper bound on the length of any SCS of R. Let T'' be an SCS of R. We may again observe that there exists a threading scheme Θ'' for T'', such that for any S_i, $\bar{A}$ of S_i is completely threaded with the left $\bar{A}$ or the right $\bar{A}$ of T. If $\bar{A}$ of S_i had no threads in common with the left or right $\bar{A}$ of T, Θ'' would have at least $10c(4r + 3t + 9) + (4r + 2t) + 10c$ threads, which is too many, as $10c > 2r + t$. Once we have one thread common to $\bar{A}$ of S_i and the left or right $\bar{A}$ of T, we can run $10c$ threads through the two $\bar{A}$'s. Figure 26 shows one of the two cases. A similar

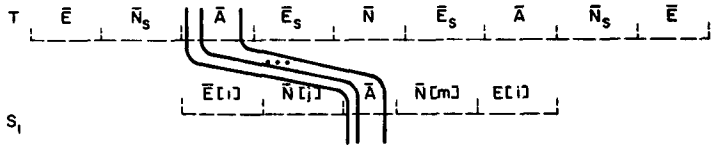


FIG 26

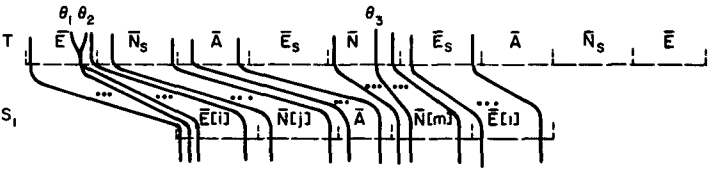


FIG. 27

argument shows that the a-blocks of $\bar{N}[j]$ and $\bar{N}[m]$ and the 0-blocks of the left and right $\bar{E}[i]$ can also be made to thread with whole blocks in T.

Assuming $\bar{A}$ of S_i is threaded as in Figure 26, we have little choice on how to thread the blocks of $\bar{E}[i]$ and $\bar{N}[j]$ They thread with $\bar{E}$ and $\bar{N}_S$ of T, and we need two extra threads, θ_1 and θ_2, to thread the two ones in the left $\bar{E}[i]$. (See Figure 27.) On the right side, we can do no better than to thread $\bar{N}$ and $\bar{N}[m]$, and the right $\bar{E}_S$ and the right $\bar{E}[i]$, with an extra thread θ_3 to thread the b in $\bar{N}[m]$. We have the mirror situation if $\bar{A}$ of S_i is threaded with the right $\bar{A}$ of T.

We may now proceed as in Theorem 4 to show that the nodes of N corresponding to extra threads through $\bar{N}$ of T form a node cover for G of size k.

PROOF OF THEOREM 5 CONTINUED. The reduction of the node cover problem in polynomial time and the NP-completeness follow as usual.

5. *Conclusion*

We have seen that the LCS and SCS problems are NP-complete for alphabet sizes of 2 and 5, respectively. We conjecture that the latter figure might be reduced to 3, by changing the b's and ones to stars in Theorem 5.

The theorems indicate that any method for finding the LCS and SCS of an arbitrary number of sequences is going to be intractable, and hence not useful for data compression schemes. However, we might still ask if there are any good approximation methods which could be used for data compression. Another question of interest is whether there exist low order polynomial reductions directly between the LCS and SCS problems. Here we note that the LCS of a set of sequences does not necessarily give any information about the SCS, since we can always add a sequence to the set which will not change the SCS but which is a common subsequence of all the other sequences.

ACKNOWLEDGMENT. The author wishes to thank Jeff Ullman for suggesting the topics covered in this paper, and for the many discussions we had about them.

REFERENCES

1 AHO, A V, HIRSCHBERG, D S, AND ULLMAN, J D Bounds on the complexity of the longest common subsequence problem *J ACM 23,* 1 (Jan 1976), 1–12

2 AHO, A V, HOPCROFT, J E., AND ULLMAN, J D *The Design and Analysis of Computer Algorithms* Addison-Wesley, Reading, Mass, 1974

3 CHVATAL, V, KLARNER, D A, AND KNUTH, D E Selected combinatorial research problems STAN-CS-72-292, Stanford U, Stanford, Calif, 1972, p 26

4 CHVATAL, V, AND SANKOFF, D Longest common subsequences for two random sequences STAN-CS-75-477, Stanford U, Stanford, Calif., Jan 1975

5 COOK, S A The complexity of theorem proving procedures Proc Third Annual ACM Symp on Theory of Computing, 1971, pp 151–158.

6. DAYHOFF, M.O Computer aids to protein sequence determination. *J. Theoret. Biology 8,* 1 (Jan 1965), 97–112.
7. DAYHOFF, M.O. Computer analysis of protein evolution. *Scientif. Amer. 221,* 1 (July 1969), 86–95
8. FISCHER, M.J., AND PATERSON, M.S. String matching and other products. Tech Memo 41, Proj MAC, M.I.T., Cambridge, Mass, 1974.
9 FREDMAN, M.L. On computing length of the longest increasing subsequences *Discrete Math 11,* 1 (Jan. 1975), 29–36.
10. HIRSCHBERG, D S On finding maximal common subsequences TR-156, Comptr Sci Lab, Princeton U, Princeton, N J, 1974
11. HIRSCHBERG, D S. A linear space algorithm for computing maximal common subsequences *Comm. ACM 18,* 6 (June 1975), 341–343
12 HIRSCHBERG, D S The longest common subsequence problem. Ph D Diss., Princeton U, Princeton, N J., Aug 1975.
13 KARP, R.M Reducibility among combinatorial problems. In *Complexity of Computer Computation,* R E Miller and J.W Thatcher, Eds, Plenum, New York, 1972, pp. 85–103
14 LOWRANCE, R., AND WAGNER, R A An extension of the string-to-string correction problem *J. ACM 22,* 2 (April 1975), 177–183
15. MORRIS, J.H., AND PRATT, V R. A linear pattern-matching algorithm. TR-40, Comptr Ctr., U of California, Berkeley, Calif., June 1970
16 NEEDLEMAN, S B, AND WUNSCH, C S. A general method applicable to the search for similarities in the amino acid sequence of two proteins *J. Molecular Biol 48* (1970), 443–453
17 SANKOFF, D Matching sequences under deletion/insertion constraints *Proc. Nat Acad Sci. USA 69,* 1 (Jan 1972), 4–6
18. SANKOFF, D., AND CEDERGREN, R J A test for nucleotide sequence homology. *J Molecular Biol 77* (1973), 159–164.
19. SELLERS, P H An algorithm for the distance between two finite sequences *J Combin Theory 16* (1974), 253–258
20. SZYMANSKI, T.G. A special case of the maximal common subsequence problem. TR-170, Comptr Sci. Lab, Princeton U, Princeton, N J., Jan. 1975.
21. WAGNER, R.A On the complexity of the extended string-to-string correction problem. Proc Seventh Annual ACM Symp. on Theory of Comptng, 1975, pp 218–223
22. WAGNER, R A, AND FISCHER, M J The string-to-string correction problem *J ACM 21,* 1 (Jan 1974), 168–173
23. WEINER, P Linear pattern matching algorithms Proc 14th Annual Symp on Switching and Automata Theory, 1973, pp 1–11
24. WONG, C.K., AND CHANDRA, A.K Bounds for the string editing problem *J ACM 23,* 1 (Jan. 1976), 13–16.

RECEIVED NOVEMBER 1976; REVISED JUNE 1977