Exemplar Longest Common Subsequence (extended abstract)

Paola Bonizzoni, Gianluca Della Vedova, Riccardo Dondi, Guillaume Fertin, Stéphane Vialette

Exemplar Longest Common Subsequence

Paola Bonizzoni[1], Gianluca Della Vedova[2], Riccardo Dondi[1], Guillaume Fertin[3], and Stéphane Vialette[4]

[1] Dipartimento di Informatica, Sistemistica e Comunicazione, Università degli Studi di Milano-Bicocca Milano - Italy
[2] Dipartimento di Statistica, Università degli Studi di Milano-Bicocca Milano - Italy
[3] LINA - FRE CNRS 2729 Université de Nantes, Nantes Cedex 3, France
[4] LRI - UMR CNRS 8623 Faculté des Sciences d'Orsay, Université Paris-Sud Bât 490, Orsay Cedex, France

Abstract. In the paper we investigate the computational and approximation complexity of the Exemplar Longest Common Subsequence of a set of sequences (ELCS problem), a generalization of the Longest Common Subsequence problem, where the input sequences are over the union of two disjoint sets of symbols, a set of mandatory symbols and a set of optional symbols. We show that different versions of the problem are **APX**-hard even for instances with two sequences. Moreover, we show that the related problem of determining the existence of a feasible solution of the Exemplar Longest Common Subsequence of two sequences is **NP**-hard.
On the positive side, efficient algorithms for the ELCS problem over instances of two sequences where each mandatory symbol can appear totally at most three times or the number of mandatory symbols is bounded by a constant are given.

1 Introduction

Algorithmic studies in comparative genomics have produced powerful tools for the analysis of genomic data which has been successfully applied in several contexts, from gene functional annotation to phylogenomics and whole genome comparison. A main goal in this research field is to explain differences in gene order in two (or more) genomes in terms of a limited number of rearrangement operations.

When there are no duplicates in the considered genomes, the computation of the similarity measure is usually polynomial-time solvable, *e.g.*, number of breakpoints, reversal distance for signed genomes, number of conserved intervals, number of common intervals, maximum adjacency disruption, summed adjacency disruption. It turns out that these methods based on permutations of gene order are useless for larger genomes where several copies of the same gene, or several highly homologous genes may be scattered across the genome. One approach to overcoming this difficulty is based on the concept of *exemplar*: for each genome, an exemplar sequence is constructed by deleting all but one occurrence of each gene family. Another approach is based on *matching* operation: in this two-step procedure, the two genomes are first made *balanced* (the number of occurrences of each gene from the same family must be the same in both genomes) by removing a minimum number of genes and next a one-to-one correspondence (among genes of each family) between genes of the genomes is computed.

Unfortunately, in the presence of duplicates, most similarity measures turn out to be **NP**-hard to compute for both the exemplar and the matching models, so that we generally have to rely on heuristic approaches. We discuss here one such general purpose heuristic

approach (the EXEMPLAR LCS problem) which is basically a constrained string alignment problem that may be of independent interest. The basic idea of the general framework we propose is that for most similarity measures (and for both the exemplar and the matching models), specific common subsequences may correspond to conserved ordered sets of genes. For example, in the exemplar model, a good similarity measure among two genomes with duplications is obtained by looking for the longest common subsequence that must contain at most one occurrence of each letter (or exactly one if no other occurrences of the gene can be found elsewhere along the genomes). This measure suggests to consider a LCS-like problem that deals with two types of letters (*mandatory* and *optional* symbols) to allow greater flexibility in the searching process.

In this paper we will formally define such framework with a simple combinatorial problem that generalizes the well known LCS problem and we will study its computational and approximation complexity. We show that different versions of the problem are **APX**-hard even for instances with two sequences and that even determining if a feasible solution exists or not is **NP**-hard. On a positive side the hardness of the problem can be limited in some cases, in fact we have shown that it is possible to determine efficiently a feasible solution, provided that each symbol appears at most three times totally in the input sequence. Finally we have designed a polynomial-time algorithm for the case where the number of mandatory symbol is at most a constant.

2 The problems

The LONGEST COMMON SUBSEQUENCE problem (shortly LCS) is a well-known problem in Computational Biology. Let $s = s[1], s[2], \ldots, s[m]$ and $t = t[1], t[2], \ldots, t[l]$ be two sequences, s is a subsequence of t if for some $j_1 < j_2 < \ldots < j_m$ $s[h] = t[j_h]$.

Let s_1, s_2 be two sequences, a *longest common subsequence* of s_1 and s_2 is a sequence s of maximum length, such that s is a subsequence of both s_1 and s_2. Let S be a set of sequences, then a *longest common subsequence* of S is a longest possible sequence s such that s is a subsequence of each sequence in S.

A simple way to informally define a subsequence is by using the notion of *threading scheme*. First write the two sequences on two parallel lines, then a threading scheme is a set of lines, each one connecting two identical symbols of two different sequences, such that no two lines are crossing, in this case each line corresponds to one symbol of the subsequence.

Next we give the definition of *longest common subsequence* for a set of sequences.

Problem name	Occurrences mandatory symbols	Occurrences optional symbols
ELCS$(1, \leq 1)$	exactly 1	at most 1
ELCS(1)	exactly 1	unrestricted
ELCS$(\geq 1, \leq 1)$	at least 1	at most 1
ELCS(≥ 1)	at least 1	unrestricted

Table 1. Versions of EXEMPLAR LCS

Given a set of sequences S, the LCS problem asks for a *longest common subsequences* of S. The complexity of LCS problem has been deeply studied in the past. In [7] it is shown that the problem is **NP**-hard even for sequences over an alphabet of size 2. However, when the instance of the problem consists of a fixed number of sequences, the LCS can be solved in polynomial time via dynamic programming algorithms [5, 4]. Next we state formally the EXEMPLAR LCS problem (ELCS). **Input**: a

set S of sequences over alphabet $A_o \cup A_m$, where A_o is the set of *optional* symbols and A_m is the set of *mandatory* symbols. The sets A_o, A_m are disjoint. **Output**: a longest common subsequence of all sequences in S and containing all mandatory symbols.

Given an instance S of ELCS, by *exemplar common subsequence* we mean a feasible solution of ELCS over S. It is possible to define different versions of the problem, according to the number of occurrences of each symbol in the solution, as represented in Table 1. In this paper we will deal with such different versions of ELCS. First notice that ELCS(1) and ELCS(≥ 1) are generalizations of the Longest Common Subsequence problem, where no mandatory symbols are present. Therefore all the hardness results for LCS apply to ELCS(1) and ELCS(≥ 1). Moreover, we will show that the above problems are hard also on instances of only two sequences (while the LCS problem can be solved in polynomial time for any fixed number of sequences).

When dealing with the restriction of ELCS containing only a fixed number of sequences, we will denote such restriction prefixing the problem name with the number of sequences, e.g. 2-ELCS($1, \leq 1$) is the restriction of ELCS($1, \leq 1$) to instances of two sequences.

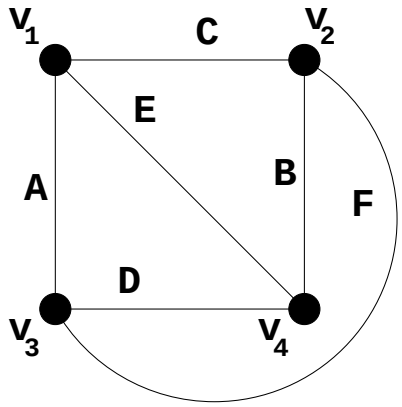
3 The results

Theorem 1 *The 2-ELCS*$(1, \leq 1)$ *problem is* ***APX****-hard even when each symbol appears at most twice in each input sequence.*

Proof. We prove the theorem, describing an L-reduction from Max Independent Set on Cubic Graph to 2-ELCS($1, \leq 1$). Max Independent Set on Cubic Graph is known to be **APX**-hard[1]. Let $G = (V, E)$ be a cubic graph. Then for each vertex v_i there are three edges $e_1(v_i)$, $e_2(v_i)$, $e_3(v_i)$ incident on it. In the reduction each vertex v_i is associated with a symbol v_i of A_o and a symbol x_i in A_m. Each edge is associated with a distinct symbol of A_m. Define a *block* associated with a vertex v_i, as a string consisting of a vertex symbol v_i, the symbols associated with edges incident on v_i in G and the symbols x_i. More precisely, there are two possible blocks associated with v_i, one contained in s_1 and defined as $b_1(v_i) = v_i e_1(v_i) e_2(v_i) e_3(v_i) x_i$, the second contained in s_2 and defined as $b_2(v_i) = e_1(v_i) e_2(v_i) e_3(v_i) v_i x_i$. The two sequences instance of 2-ELCS($1, \leq 1$) are: $s_1 = b_1(v_1) b_1(v_2) \cdots b_1(v_n)$, $s_2 = b_2(v_1) b_2(v_2) \cdots b_2(v_n)$. Observe that the symbols x_i are mandatory, thus they must appear in any feasible solution of 2-ELCS($1, \leq 1$). There is only one of each x_i in both s_1, s_2. More precisely, x_i occurs in blocks $b_1(v_i)$, $b_2(v_i)$ of s_1, s_2 respectively, thus any symbol x_i in a feasible solution of 2-ELCS($1, \leq 1$) over s_1 and s_2 must be taken from $b_1(v_i)$ and $b_2(v_i)$. It follows that if v_i is in a exemplar common subsequence, then the exemplar common subsequence does not contain any symbol of $e_1(v_i) e_2(v_i) e_3(v_i)$ of $b_1(v_i)$ and $b_2(v_i)$. Let s be a feasible solution of 2-ELCS($1, \leq 1$) over s_1, s_2, then s consists of $f_1 x_1 \dots f_i x_i \dots f_n x_n$, where each f_i is either v_i or a subsequence of $e_1(v_i) e_2(v_i) e_3(v_i)$. Observe that each edge symbol is mandatory, which means that it must appear exactly once in a common subsequence. Moreover, an edge symbol encoding edge (v_i, v_j) appears in blocks $b_1(v_i)$ and $b_1(v_j)$ of s_1 and in blocks $b_2(v_i)$ and $b_2(v_j)$ of s_2. Thus a common subsequence takes such edge symbol either from $b_1(v_i)$ and $b_2(v_i)$ or from $b_1(v_j)$ and $b_2(v_j)$.

Let I be the set of vertices appearing in s, then we can show that I is an independent set of G. Assume that symbols $v_i, v_j \in I$. Then (v_i, v_j) is not an edge of G, otherwise the

solution s in f_i and f_j contains symbols v_i and v_j respectively. An immediate consequence is that the edge symbol associated with (v_i, v_j), that can appear only in f_i and f_j, is not contained in s. Since each edge symbol is mandatory, it must appear in any feasible solution of 2-ELCS$(1, \leq 1)$, which is a contradiction.



$s_1 = v_1CAEx_1v_2CFBx2v_3AFDx_3v_4EBDx_4$
$s_2 = CAEv_1x_1CFBv_2x2AFDv_3x_3EBDv_4x_4$

Fig. 1. Reducing the graph K_4

Observe that the length of a feasible solution of 2-ELCS$(1, \leq 1)$ over s_1, s_2 is $|V| + |E| + |I|$, where I is an independent set of G.

On the other side, assume that I is an independent set of G. It is easy to compute a feasible solution of 2-ELCS$(1, \leq 1)$ over s_1, s_2 of size $|V| + |E| + |I|$, retaining only the symbols associated with vertices in I in the exemplar common subsequence. Indeed, note that, since I is an independent set, for each edge $e = (v_i, v_j)$ at least one of v_i, v_j is not in I, hence each symbol associated with e can be retained once in a feasible solution of 2-ELCS$(1, \leq 1)$ over s_1, s_2. □

A similar proof can be given also for 2-ELCS$(\geq 1, \leq 1)$.

Theorem 2 *The 2-ELCS$(\geq 1, \leq 1)$ problem is **APX**-hard even when each symbol appears at most twice in each input sequence.*

Proof. The reduction is similar to the previous one, but for each vertex v_i of the graph, we have four symbols $v_i^a v_i^b v_i^c v_i^d$ and the blocks $b_1(v_i)$ and $b_2(v_i)$ associated with v_i in sequences s_1 and s_2 respectively are defined as follows: $b_1(v_i) = v_i^a v_i^b v_i^c v_i^d e_1(v_i)e_2(v_i)e_3(v_i)x_i$; $b_2(v_i) = e_1(v_i)e_2(v_i)e_3(v_i)v_i^a v_i^b v_i^c v_i^d x_i$.

Again the symbols x_i are mandatory, therefore they must appear in any feasible solution of 2-ELCS$(\geq 1, \leq 1)$ over s_1, s_2. There is only one occurrence of each symbol x_i in both s_1, s_2. More precisely x_i appears in blocks $b_1(v_i)$ and $b_2(v_i)$ of s_1 and s_2. It follows that any symbol x_i in an exemplar common subsequence must be taken from the blocks of s_1, s_2 associated with v_i, that is $b_1(v_i)$ and $b_2(v_i)$. Since each mandatory edge symbols appears twice in each input sequence, it must appear once or twice in a common subsequence.

Clearly if sequence $v_i^a v_i^b v_i^c v_i^d$ is in a feasible solution of 2-ELCS$(\geq 1, \leq 1)$ over s_1, s_2, then this solution does not contain occurrence of symbols of sequence $e_1(v_1)e_2(v_1)e_3(v_1)$ in $b_1(v_i)$ and $b_2(v_i)$. This means that a feasible solution of 2-ELCS$(\geq 1, \leq 1)$ over s_1, s_2 consists of $g_1x_1 \dots g_ix_i \dots g_nx_n$, where each g_i is either a subsequence of $v_i^a v_i^b v_i^c v_i^d$ or g_i is a subsequence of $e_1(v_i)e_2(v_i)e_3(v_i)$. Observe that each edge symbol is mandatory, which means that it must appear exactly once in an exemplar common subsequence. Thus an exemplar common subsequence takes each edge symbol from one of the two blocks where it appears.

Assume that I is an independent set of G, then we claim that there exists a feasible solution s of 2-ELCS$(\geq 1, \leq 1)$ over s_1, s_2 of length $|V| + 3|V| + |I|$. Such a feasible

solution consists of $g_1x_1 \dots g_ix_i \dots g_nx_n$, where each $g_i = v_i^a v_i^b v_i^c v_i^d$ if $v_i \in I$ and $g_i = e_1(v_i)e_2(v_i)e_3(v_i)$ otherwise. Indeed it is immediate to note that the length of sequence s is $|V| + 3(|V| - |I|) + 4|I| = |V| + 3|V| + |I|$ and that it is a common subsequence of s_1 and s_2. Moreover, all mandatory symbols encoding an edge are included in s. W.l.o.g. assume to the contrary that a symbol encoding the edge (v_1, v_2) is not included in s. This fact implies that $g_1 = v_1^a v_1^b v_1^c v_1^d$ and $g_2 = v_2^a v_2^b v_2^c v_2^d$, hence $v_1, v_2 \in I$, contradicting the assumption that I is an independent set of G.

Assume now that there exists a feasible solution s of 2-ELCS($\geq 1, \leq 1$) over s_1, s_2 with length $|V| + 3|V| + |I|$. We can assume that, for each block in s_1, $s2$, either $v_i^a v_i^b v_i^c v_i^d$ or $e_1(v_i)e_2(v_i)e_3(v_i)$ appears as a substring of s. Let Y be the set of blocks for which $v_i^a v_i^b v_i^c v_i^d$ is part of s. Hence the vertices corresponding to Y are an independent set of G. By a trivial counting argument, it is easy to shows that for $|I|$ blocks s includes $v_i^a v_i^b v_i^c v_i^d$. We claim that such blocks encode an independent set. W.l.o.g. assume that $v_1^a v_1^b v_1^c v_1^d$ and $v_2^a v_2^b v_2^c v_2^d$ are included in s, then there is no edge (v_1, v_2) in G, otherwise the mandatory symbol encoding such edge would not be in s. □

A related problem is that, given an instance of 2-ELCS, of determining if a feasible solution exists. In this section we will consider a general version of the 2-ELCS problem.

Notice that both reductions described above hold for instances that are known to admit a feasible solution, therefore they are not sufficient for dealing with the problem. Observe that, since only mandatory symbols are relevant for the existence of a solution, removing all optional symbols does not change the fact that a feasible solution exists or not. Therefore in this section we can assume that both input sequences are made only of mandatory symbols.

Clearly, in order to have a feasible solution, each mandatory symbol must appear in both input sequences. Since it is trivial to verify in polynomial time such property, in the following we can assume that each mandatory symbol appears in both input sequences. The number of occurrences of each mandatory symbol in the instance is a fundamental parameter, in fact we will show that finding a feasible solution can be done in polynomial time for small values of such parameter, but becomes intractable for different values.

Theorem 3 *The problem of determining if a feasible solution exists for an instance of 2-ELCS where each mandatory symbol appears totally at most three times in the input sequences, can be solved in polynomial time.*

Proof. We prove the theorem reducing an instance of 2-ELCS, where each mandatory symbol appears totally at most three times in the input sequences to an instance of 2SAT, that is the restriction of SATISFIABILITY to instances where each clause contains at most two literals. Notice that 2SAT can be solved in polynomial time [2].

For each symbol s, let $occ_1(s)$ (respectively $occ_2(s)$) be the set of positions of the input sequence s_1 (resp. s_2) where the symbol s appears. Clearly both $occ_1(s)$ and $occ_2(s)$ are not empty and $|occ_1(s)| + |occ_2(s)| \leq 3$. For each symbol s there are at most two pairs in $occ_1(s) \times occ_2(s)$, for otherwise $|occ_1(s)| + |occ_1(s)| > 3$, let us associate with each of such pairs a variable $x_{s,i}$, where $i \in \{1, 2\}$ if there are two pairs in $occ_1(s) \times occ_2(s)$ and $i = 1$ if there is only one pair in $occ_1(s) \times occ_2(s)$.

Graphically the possible variables are represented in Fig. 3 with a line connecting two identical symbols belonging to different sequences. The case $|occ_1(s)| + |occ_1(s)| =$

3 is represented by the two leftmost lines and the variables $x_{s,1}$, $x_{s,2}$, while the case $|occ_1(s)| + |occ_1(s)| = 2$ is represented by the rightmost line and the variable $x_{t,1}$. Each truth assignment to the variables can be viewed as picking the lines corresponding to true variables. Let C be the set of clauses of the instance of 2SAT that we are constructing. For each pair $x_{s,1}$, $x_{s,2}$ of variables, the clauses $\neg x_{s,1} \vee \neg x_{s,2}$ and $x_{s,1} \vee x_{s,2}$ are added to C. Moreover, for each symbol s such that there is only one pair in $occ_1(s) \times occ_2(s)$, add the clause $x_{s,1}$ to C (this corresponds to forcing the variable $x_{s,1}$ to be true). The fact that all these clauses are satisfied in any feasible solution of 2SAT, corresponds to pick exactly one of the lines associated with each symbol. Two lines (or two variables) are called *crossing* if they cross in the drawing built as in Fig. 3. More formally, notice that each variable $x_{s,i}$ is associated with an occurrence of s in s_1 (denoted as $s_1(s,i)$) and one occurrence of s in s_2 (denoted as $s_2(s,i)$). A pair $x_{s,i}, x_{t,j}$ of variables is crossing if in s_1 the symbol $s_1(s,i)$ precedes $s_1(t,j)$ and in s_2 the symbol $s_2(s,i)$ does not precede $s_2(t,j)$ or, symmetrically, if in s_1 the symbol $s_1(s,i)$ does not precede $s_1(t,j)$ and in s_2 the symbol $s_2(s,i)$ precedes $s_2(t,j)$. For each pair $x_{s,i}, x_{t,j}$ of crossing variables, the clause $\neg x_{s,i} \vee \neg x_{t,j}$ is added to C.

We can prove that the original instance of 2-ELCS has a feasible solution if and only if the instance of 2SAT is satisfiable, that is there is a truth assignment for all variables such that all clauses in C are evaluated true.

Assume that there is a feasible solution z of the instance of 2-ELCS then, for each symbol s, we pick the lines connecting the symbols retained in z. By definition of common subsequence there cannot be two crossing lines, and exactly one of the lines associated with each symbol must be picked as $|occ_1(s)| + |occ_1(s)| \leq 3$, therefore we have constructed a feasible solution of 2SAT.

Conversely given a truth assignment for all variables that satisfies all clauses in C, it is immediate to note that there are not two crossing lines, and that there is exactly one line for each symbol, therefore it is immediate to construct a feasible solution of 2-ELCS that contains all symbols. □

Notice that the above result holds for all the restrictions of 2-ELCS as no symbol appears twice in both input sequences, therefore it can appear at most once in any solution. Slightly relaxing the constraint on the number of occurrences of each symbol makes the problem hard to solve efficiently, in fact if each mandatory symbol can have three occurrences in each sequence then the problem becomes **NP**-hard, as shown in the following theorem.

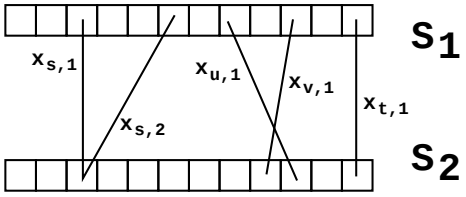


Fig. 2. Reducing 2-ELCS to 2SAT

Theorem 4 *The problem of determining if a feasible solution exists for an instance of 2-ELCS where each mandatory symbol appears at most three times in each input sequence, is **NP**-hard.*

Proof. We will prove the theorem reducing 3SAT to 2-ELCS, with a reduction very similar to the one shown before. Let $C = \{C_1, \ldots, C_k\}$ be a set of clauses, each one consisting of at most three (possibly negated) literals. We construct an instance of 2-ELCS associating a block with each variable. The block of s_1 associated with variable x_i is defined as the symbol x_i, followed by the sequence of clauses containing x_i, followed by the sequence of clauses containing $\neg x_i$, where in each sequence the clauses are ordered according to the index in $\{C_1, \ldots, C_k\}$. In s_2 the block associated with variable x_i is defined as the symbol x_i, followed by the sequence of clauses containing $\neg x_i$, followed by the sequence of clauses containing x_i (again the clauses are ordered according to the index in $\{C_1, \ldots, C_k\}$). For example if x_1 appears negated in C_1 and not negated in C_2, C_3, then the corresponding blocks are $x_1 C_2 C_3 C_1$ (in s_1) and $x_1 C_1 C_2 C_3$ (in s_2). Both sequences s_1 and s_2 consist of the sequence of all blocks associated to the variables of the original instance of 3SAT. All symbols are mandatory, also notice that each symbol appear at most three times in each sequence as each clause contains at most three literals.

Each symbol x_i appears exactly once in each sequence, therefore there is no ambiguity on which occurrence is retained in any exemplar common subsequence. Consequently each symbol retained must correspond to occurrences taken from the same block. Inside the block associated with x_i, retaining the clauses where x_i appears as a positive literal is mutually exclusive with retaining the clauses where x_i appears as a negative literal, by definition of exemplar common subsequence. The first case (that is retaining the clauses where x_i appears as a positive literal) corresponds to setting x_i to true, while the second case corresponds to setting x_i to false. In both case the clauses retained are satisfied by the assignment of variables x_i.

It is immediate to note that any feasible solution must contain all clauses, therefore we have computed a truth assignment of the variables that satisfies all clauses in C, completing the proof. □

The above results have a simple but definitive consequence on the approximability of the 2-ELCS problem where each mandatory symbol appears at most three times in both input sequences, as they rule out any possible polynomial-time approximation algorithm.

Since the problem can be extended to instances consisting of a set of sequences, it is interesting if the above results can be made stronger. In fact, the well-known inapproximability results by Jiang and Li [6] for the LCS problem, immediately apply also to the ELCS(≥ 1) problem, since ELCS(≥ 1) is more general than LCS. A closer inspection of their proofs shows that their result also apply to all versions of ELCS, as the optimal solutions in their reductions contain at most one occurrence of each symbol, hence there cannot be any $O(n^{1-\epsilon})$ ratio polynomial-time approximation algorithm unless **P**=**NP**, even if no mandatory symbol is allowed and all symbols appear at most twice in each sequence.

4 Restricting the problem

We are now interested in determining if a reasonable restriction can lead to an efficient solution. This section is devoted to the restriction of 2-ELCS(1) where the number of mandatory symbols is at most a constant. Our algorithm is based on two phases: the first step consists of guessing the exact ordering of all mandatory symbols in the optimal solution, the second step basically fills in the gaps between each pair of mandatory symbol.

Since each mandatory symbol appears exactly once in a feasible solution, the correct ordering of the mandatory symbol is a permutation of A_m. Since $|A_m|$ is a constant, we can run the second phase for all possible permutations of A_m at the expense of only a constant multiplicative increase in time complexity.

Let s be a permutation of mandatory symbol, the second phase consists of computing a longest common subsequence s^* of $\{s_1, s_2\}$ with the costraint that s is a subsequence of s^*. In the following let us denote by $s[i]$ the i-th character of the sequence s and by $s[i \dots j]$ the substring of s starting with $s[i]$ and ending with $s[j]$. The recurrence equation for $\mathrm{ELCS}[i, j, k]$, that is the length of an optimal solution over the sequences $s_1[1 \dots i]$, $s_2[1 \dots j]$ that is a supersequence of the sequence $s[1] \cdots s[k]$ is:

$$\mathrm{ELCS}[i,j,k] = \max \begin{cases} \mathrm{ELCS}[i-1, j-1, k] + 1 & \text{if } s_1[i] = s_2[j],\ s_1[i] \in A_o \\ \mathrm{ELCS}[i-1, j-1, k-1] + 1 & \text{if } s_1[i] = s_2[j] = s[k] \\ \mathrm{ELCS}[i-1, j, k], \mathrm{ELCS}[i, j-1, k] & \text{if } s_1[i] = s_2[j] \neq s[k],\ s_1[i] \in A_m \\ \mathrm{ELCS}[i-1, j, k], \mathrm{ELCS}[i, j-1, k] & \text{if } s_1[i] \neq s_2[j] \end{cases}$$

The boundary conditions are $\mathrm{ELCS}[0, j, 0] = 0$ and $\mathrm{ELCS}[i, 0, 0] = 0$ for $0 \leq i \leq |s_1|$ and $0 \leq j \leq |s_2|$. The value of the optimal solution can be read in $\mathrm{ELCS}[|s_1|, |s_2|, |s|]$, once the matrix ELCS has been completely filled in. The actual optimal solution can be constructed by standard backtracking techniques [3]. The recurrence equation is for the 2-ELCS(1) problem, but it can be easily modified for the 2-ELCS(≥ 1), by removing the requirement $s_1[i] \in A_o$ in the first condition of the equation, so that a mandatory symbol can be retained more than once.

References

1. P. Alimonti and V. Kann. Some APX-completeness results for cubic graphs. *Theoretical Computer Science*, 237(1–2):123–134, 2000.
2. B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979.
3. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001.
4. K. Hakata and H. Imai. The longest common subsequence problem for small alphabet size between many strings. In *Proc. 3rd International Symp. on Algorithms and Computation (ISAAC)*, pages 469–478, 1992.
5. W. Hsu and M. Du. New algorithms for the LCS problem. *Journal of Computer and System Sciences*, 19:133–152, 1984.
6. T. Jiang and M. Li. On the approximation of shortest common supersequences and longest common subsequences. *SIAM Journal on Computing*, 24(5):1122–1139, 1995.
7. D. Maier. The complexity of some problems on subsequences and supersequences. *Journal of the ACM*, 25:322–336, 1978.