Information Processing Letters 110 (2010) 877–881

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Variants of constrained longest common subsequence

Paola Bonizzoni [a], Gianluca Della Vedova [b], Riccardo Dondi [c,*], Yuri Pirola [a]

[a] DISCo, Università degli Studi di Milano-Bicocca, Milano, Italy
[b] Dipartimento di Statistica, Università degli Studi di Milano-Bicocca, Milano, Italy
[c] Dipartimento di Scienze dei Linguaggi, della Comunicazione e degli Studi Culturali, Università degli Studi di Bergamo, Bergamo, Italy

ARTICLE INFO

Article history:
Received 1 December 2009
Received in revised form 14 July 2010
Accepted 15 July 2010
Available online 17 July 2010
Communicated by J. Torán

Keywords:
Algorithms
Longest common subsequence
Constrained longest common subsequence
Fixed-parameter tractability

ABSTRACT

We consider a variant of the classical Longest Common Subsequence problem called Doubly-Constrained Longest Common Subsequence (DC-LCS). Given two strings s_1 and s_2 over an alphabet Σ, a set C_s of strings, and a function $C_o : \Sigma \to N$, the DC-LCS problem consists of finding the longest subsequence s of s_1 and s_2 such that s is a supersequence of all the strings in C_s and such that the number of occurrences in s of each symbol $\sigma \in \Sigma$ is upper bounded by $C_o(\sigma)$. The DC-LCS problem provides a clear mathematical formulation of a sequence comparison problem in Computational Biology and generalizes two other constrained variants of the LCS problem that have been introduced previously in the literature: the Constrained LCS and the Repetition-Free LCS. We present two results for the DC-LCS problem. First, we illustrate a fixed-parameter algorithm where the parameter is the length of the solution which is also applicable to the more specialized problems. Second, we prove a parameterized hardness result for the Constrained LCS problem when the parameter is the number of the constraint strings ($|C_s|$) and the size of the alphabet Σ. This hardness result also implies the parameterized hardness of the DC-LCS problem (with the same parameters) and its NP-hardness when the size of the alphabet is constant.

1. Introduction

The problem of computing the longest common subsequence (LCS) of two sequences is a fundamental problem in stringology and in the whole field of algorithms [1], as it couples a wide range of applications with a simple mathematical formulation [2]. Applications of variants of LCS range from Computational Biology to data compression, syntactic pattern recognition and file comparison (for instance it is used in the Unix *diff* command) [2].

A few basic definitions are necessary. Given two sequences s and t over a finite alphabet Σ, s is a *subsequence* of t if s can be obtained from t by removing some (possibly zero) characters. When s is a subsequence of t, then t is a *supersequence* of s. Given two sequences s_1 and s_2, the Longest Common Subsequence problem (LCS) asks for a longest possible sequence s that is a subsequence of both s_1 and s_2.

The problem of computing the longest common subsequence of two sequences has been deeply investigated, and polynomial time algorithms are well-known for the problem [3–6]. It is possible to generalize the LCS problem to a set of sequences: in such a case the goal is to compute a sequence that is a subsequence of all input sequences. This problem is NP-hard even on binary alphabet [7] and it is not approximable within factor $O(n^{1-\varepsilon})$, for any constant $\varepsilon > 0$, on arbitrary alphabet [8]. The LCS problem is fixed-parameter tractable (using some naïve algorithms) when the parameters are the number of input strings and the solution length for strings over a fixed alphabet or when the parameters are the alphabet cardinality and the solution length. It does not admit a fixed-parameter algorithm for the other choices of the parameters [9,10].

Computational Biology is a field where several variants of the LCS problem have been introduced for various pur-

* Corresponding author.
E-mail addresses: bonizzoni@disco.unimib.it (P. Bonizzoni), gianluca.dellavedova@unimib.it (G. Della Vedova), riccardo.dondi@unibg.it (R. Dondi), pirola@disco.unimib.it (Y. Pirola).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2010.07.015

poses. For instance, researchers have defined some similarity measures between genome sequences based on constrained forms of the LCS problem. In particular, an LCS-like problem has been studied to deal with two types of symbols (*mandatory* and *optional* symbols) to model the differences in the number of occurrences allowed for each gene [11,12]. Another illustrative example is the definition of *repetition-free longest common subsequence* [12] where, given two sequences s_1 and s_2, a repetition-free common subsequence is a subsequence of both s_1, s_2 that contains at most one occurrence of each symbol. Such a model has been proposed as a similarity measure that can be useful in genome rearrangement analysis, in particular when dealing with the exemplar model [13]. In such a framework we want to compute an exemplar sequence, that is a sequence which contains only one representative (called the exemplar) for each family of duplicated genes inside a genome. In biological terms, the exemplar gene may correspond to the original copy of the gene, from which all other copies originate [13]. Complexity results for the repetition-free longest common subsequence have been recently given in [14].

A different variant of LCS that has been introduced to compare biological sequences is called *constrained longest common subsequence* (C-LCS) [15]. This problem requires to compute, given three sequences s_1, s_2, and s_c, a longest common subsequence s of s_1 and s_2 such that s is a supersequence of s_c. Such a variant of LCS can be useful when comparing two biological sequences (represented by s_1 and s_2) having a known substructure in common (formalized by s_c) which must be conserved by the solution s [15]. The Constrained LCS problem can be solved by some polynomial time algorithms [15–17], but it becomes NP-hard when generalized to a set of input sequences or to a set of constraint sequences [18].

In this paper we are interested to continue the investigation of the computational complexity of the above two measures for comparing sequences, mainly when both constraints characterizing them are required. Indeed, we propose a new similarity measure for the comparison of two biological sequences (genomes) which takes into account the exemplar model (as suggested in [12]) as well as the consensus model commonly used in the comparison of biological structures which is represented by a known substructure common to the two input sequences (*i.e.*, the consensus is given by a constraint set of sequences, as proposed in [15]). We formalize a new problem called *Doubly-Constrained Longest Common Subsequence* (DC-LCS) problem, which extends both the Repetition-Free Longest Common Subsequence problem and the Constrained Longest Common Subsequence problem. Hence, given two input sequences s_1, s_2, the DC-LCS problem asks for the longest common subsequence s of s_1 and s_2 that satisfies two constraints: (i) the number of occurrences of each symbol σ is upper bounded by a quantity $C_o(\sigma)$, and (ii) s is a supersequence of every string of a specified constraint set C_s.

First, we design a fixed-parameter algorithm [9] when the parameter is the length of the solution. Clearly such an algorithm also applies to the more specialized problems. Then we give a parameterized hardness result for the Constrained Longest Common Subsequence problem, when the number of constraint sequences and the size of the alphabet are considered as parameters. The reduction used to prove this hardness result implies an identical result for DC-LCS and also shows that DC-LCS is NP-hard over an alphabet of 3 symbols.

2. Basic definitions

Let s_1, s_2 be two strings over an alphabet Σ. Given a string s, we denote by $s[i]$ the symbol at position i in string s, and by $s[i \dots j]$, the substring of s starting at position i and ending at position j. Given a string s over an alphabet Σ and a symbol σ (not necessarily in Σ), we denote by $occ(\sigma, s)$ the number of occurrences of symbol σ in s. A *string constraint* C_s consists of a set of strings, while an *occurrence constraint* C_o is a function $C_o : \Sigma \to \mathbb{N}$, assigning an upper bound on the number of occurrences of each symbol in Σ. First, consider some variants of the LCS problem.

Problem 1 (*Constrained Longest Common Subsequence* (C-LCS)).
Input: two strings s_1 and s_2, a string constraint C_s.
Output: a longest common subsequence s of s_1 and s_2, so that each string in C_s is a subsequence of s.

The problem admits a polynomial time algorithm when C_s consists of a single string [15–17], while it is NP-hard when C_s consists of an arbitrary number of strings [18]. In the latter case, notice that C-LCS cannot be approximated, since a feasible solution for the C-LCS problem must be a supersequence of all the strings in the constraint C_s and moreover, deciding the existence of a feasible solution is an NP-complete problem [18].

Problem 2 (*Repetition-Free Longest Common Subsequence* (RF-LCS)).
Input: two strings s_1 and s_2.
Output: a longest common subsequence s of s_1 and s_2, so that s contains at most one occurrence of each symbol $\sigma \in \Sigma$.

The problem is APX-hard even when each symbol occurs at most twice in each of the input strings s_1 and s_2 [12]. A positive note is that allowing at most k occurrences of each symbol in each string s_1 and s_2 results in a $\frac{1}{k}$-approximation algorithm [12].

We introduce an even more general version of both the C-LCS and RF-LCS problem, called *Doubly-Constrained Longest Common Subsequence* (DC-LCS) problem.

Problem 3 (*Doubly-Constrained Longest Common Subsequence* (DC-LCS)).
Input: two strings s_1 and s_2, a string constraint C_s, and an occurrence constraint $C_o : \Sigma \to \mathbb{N}$.
Output: a longest common subsequence s of s_1 and s_2, so that each string in C_s is a subsequence of s and s contains at most $C_o(\sigma)$ occurrences of each symbol $\sigma \in \Sigma$.

It is easy to see that the C-LCS problem is the restriction of the DC-LCS problem when $C_o(\sigma) = |s_1| + |s_2|$ for

each $\sigma \in \Sigma$. At the same time, the RF-LCS problem is the restriction of the DC-LCS problem when $C_s = \emptyset$ and $C_o(\sigma) = 1$ for each $\sigma \in \Sigma$. Therefore the DC-LCS problem cannot be approximated within any ratio, since it inherits all hardness properties of C-LCS and RF-LCS.

3. A fixed-parameter algorithm for DC-LCS

Initially we present a fixed-parameter algorithm for the DC-LCS problem when $|C_s| \leqslant 1$ (hence the result also holds for the RF-LCS problem), where the parameter k is the size of a solution of DC-LCS. Later on, we will extend the algorithm to a generic set C_s.

The algorithm is based on the color coding technique [19]. We recall the basic definition of a perfect family of hash functions [20]. Given a set S, a family F of hash functions from S to $\{1, 2, \ldots, k\}$ is called *perfect* if, for any $S' \subseteq S$ of size k, there exists an injective hash function $f \in F$ from S' to the set of labels $\{1, 2, \ldots, k\}$.

Since $|C_s| \leqslant 1$, we denote by s_c the only sequence in C_s and let k be the size of a solution for DC-LCS. Suppose w.l.o.g. that $C_o(\sigma)$ is not bigger than the value $\min\{occ(\sigma, s_1), occ(\sigma, s_2)\}$. Indeed, a solution s of DC-LCS is a subsequence of both s_1 and s_2, hence $occ(\sigma, s)$ is upper bounded by $\min\{occ(\sigma, s_1), occ(\sigma, s_2)\}$.

Given C_o and the sequences s_1 and s_2, we construct a set $\widetilde{\Sigma}$ that contains the pairs (σ, i) for each $\sigma \in \Sigma$ and $i \in \{1, \ldots, C_o(\sigma)\}$. For example, if $s_1 = aaaa\,bbbbb\,cccc\,dddd$, $s_2 = ddddd\,ccc\,bbbb\,aaaa$, and $C_o(a) = C_o(b) = 4$, $C_o(c) = C_o(d) = 3$, then the set $\widetilde{\Sigma}$ is equal to $\{(a,1), (a,2), (a,3), (a,4), (b,1), (b,2), (b,3), (b,4), (c,1), (c,2), (c,3), (d,1), (d,2), (d,3)\}$.

Consider now a perfect family F of hash functions from $\widetilde{\Sigma}$ to the set of labels $\{1, 2, \ldots, k\}$. Given a function $f \in F$, for each $\sigma \in \Sigma$ let $L_f(\sigma)$ be the set of labels associated by f with the pairs $(\sigma, i) \in \widetilde{\Sigma}$, for some i. Let s be a generic sequence over alphabet Σ and let L be a subset of $\{1, \ldots, k\}$. Then s is an *L-colorful* sequence w.r.t. a hash function $f \in F$ iff we can assign to each position i, $1 \leqslant i \leqslant |s|$, a distinct label $l_i \in L$, such that $l_i \in L_f(s[i])$, and each label $l \in L$ is assigned to some position of s. We claim that an L-colorful sequence s w.r.t. a hash function $f \in F$ contains at most $C_o(\sigma)$ occurrences of each symbol $\sigma \in \Sigma$. In fact, the cardinality of the set $L_f(\sigma)$ is upper-bounded by $C_o(\sigma)$, hence there can be at most $C_o(\sigma)$ occurrences of σ in s with distinct labels. Given three sequences s_1, s_2, and s_c, we say that an L-colorful sequence s w.r.t. a hash function f is an L-colorful *solution* (w.r.t. f) if and only if s is a common subsequence of s_1 and s_2 that is also a supersequence of s_c. In other words, an L-colorful solution is a sequence s which is both an L-colorful sequence and a solution of the C-LCS problem. By the definition of L-colorful solution and the claim stated above, any L-colorful solution s is a feasible solution of the DC-LCS problem.

The basic idea of our algorithm is to verify if there exists an L-colorful solution, for some set $L \subseteq \{1, 2, \ldots, k\}$. This task is fulfilled via dynamic programming. Since F is a perfect family of hash functions, and each symbol σ does not occur more than $C_o(\sigma)$ times in s, for each feasible solution s of length k, there exists a hash function $f \in F$ such that s is $\{1, \ldots, k\}$-colorful w.r.t. f. By computing the recurrence for all hash functions in F, we are certain to find a solution of length k, if such a solution exists.

Given a hash function f, we define $V[i, j, h, L]$ which takes value 1 if and only if there exists an L-colorful common subsequence s of $s_1[1 \ldots i]$ and $s_2[1 \ldots j]$, such that s is a supersequence of $s_c[1 \ldots h]$ (notice that s has length equal to $|L|$ or, equivalently, s uses all labels in L). Theorem 3.1 states that $V[i, j, h, L]$ can be computed by the following dynamic programming recurrence which is an extension of the standard equation for the Longest Common Subsequence (LCS) problem [1].

$$V[i, j, h, L] = \max \begin{cases} V[i-1, j, h, L] \\ V[i, j-1, h, L] \\ V[i-1, j-1, h, L \setminus \{\lambda\}] \\ \quad \text{if } s_1[i] = s_2[j] \text{ and} \\ \quad \lambda \in L \cap L_f(s_1[i]) \\ V[i-1, j-1, h-1, L \setminus \{\lambda\}] \\ \quad \text{if } s_1[i] = s_2[j] = s_c[h] \text{ and} \\ \quad \lambda \in L \cap L_f(s_1[i]) \end{cases} \tag{1}$$

The boundary conditions are $V[0, j, h, L] = 0$ and $V[i, 0, h, L] = 0$ if $L \neq \emptyset$, while $V[i, j, 0, \emptyset] = 1$ and $V[i, j, h, \emptyset] = 0$ when $h > 0$. Moreover, as a consequence of the recurrence's definition, $V[i, j, h, L] = 0$ for all $h > |L|$. A feasible solution of length k is $\{1, \ldots, k\}$-colorful w.r.t. f if and only if $V[|s_1|, |s_2|, |s_c|, \{1, \ldots, k\}] = 1$. In this case, a standard backtracking search can reconstruct the actual solution.

Theorem 3.1. *Assume that $f \in F$ is a hash function and let s be a L-colorful solution w.r.t. f. Then Eq. (1) is correct.*

Proof. We will prove the theorem by induction, that is we will prove the correctness of the value of $V[i_a, j_a, h_a, L_a]$ by assuming that of $V[i_b, j_b, h_b, L_b]$ when $i_b \leqslant i_a$, $j_b \leqslant i_a$, $h_b \leqslant h_a$, $L_b \subseteq L_a$, and at least one inequality is strict.

Let s be an optimal L_a-colorful solution for the sequences $s_1[1, \ldots, i_a]$, $s_2[1, \ldots, j_a]$, $s_c[1, \ldots, h_a]$, and let β be the last symbol of s, that is $s = t\beta$, where t is the prefix of s consisting of all but the last character.

Assume that $s_1[i_a] \neq s_2[j_a]$. Then, just as for the recurrences of the standard LCS problem [1], the theorem holds. Assume that $s_1[i_a] = s_2[j_a] = \alpha$ and let $\alpha \neq \beta$. Then, by induction hypothesis, s is an optimal L_a-colorful solution for the sequences $s_1[1, \ldots, i_a - 1]$, $s_2[1, \ldots, j_a]$, $s_c[1, \ldots, h_a]$ and the theorem holds.

Therefore we can assume now that $\alpha = \beta$. Since s is L_a-colorful, we can assign a distinct label l_i in L_a to each position i of s, such that $l_i \in L_f(s[i])$. Let l be the label assigned to the last position of s. By inductive hypothesis there exists an $L \setminus \{l\}$-colorful solution t of $s_1[1, \ldots, i_a - 1]$, $s_2[1, \ldots, j_a - 1]$, $s_c[1, \ldots, j_a]$ (if t is a supersequence of $s_c[1, \ldots, j_a]$) or of $s_1[1, \ldots, i_a - 1]$, $s_2[1, \ldots, j_a - 1]$, $s_c[1, \ldots, j_a - 1]$, hence completing the proof. □

Given a solution s of DC-LCS of size k, let $\widetilde{\Sigma}_s$ be the set of pairs (σ, j), with σ a symbol of s and $1 \leqslant j \leqslant occ(\sigma, s)$. Assume that f is a hash function that maps the set $\widetilde{\Sigma}_s$ to the set of labels $\{1, \ldots k\}$, but is not injective. Then there

is a label $z \in \{1, \ldots, k\}$ that is not assigned to any pair (σ, j). Hence the conditions of the last two cases of our recurrence equation will never hold for $\lambda = z$, which implies that $V[i, j, h, \{1, \ldots, k\}] = 0$ for all values of i, j, h, hence establishing the correctness of our algorithm.

It is immediate to notice that the total number of entries of the matrix $V[\cdot, \cdot, \cdot, \cdot]$ is $|s_1||s_2||s_c|2^k$. Furthermore notice that computing each entry requires at most $O(k)$ time, as case 1 and case 2 of the recurrence require constant time, while case 3 and case 4 require at most $O(k)$ time, since $|L| \leqslant k$. There exists a perfect family of hash functions of size $O(\log|\widetilde{\Sigma}|)2^{O(k)}$ that can be computed in $O(|\widetilde{\Sigma}|\log|\widetilde{\Sigma}|)2^{O(k)}$ time [19]. Eq. (1) is applied for each hash function of the perfect family, hence, since $|\widetilde{\Sigma}| \leqslant |s_1|$, the algorithm has an overall $O(|s_1|\log|s_1|)2^{O(k)} + O(|s_1||s_2||s_c|2^{O(k)}\log|\widetilde{\Sigma}|)$ time complexity.

The algorithm actually computes a longest supersequence of s_c which is a feasible solution of the problem. Assume now that C_s is a generic string set, and let x be an optimal solution of size k of a generic instance of the DC-LCS problem. It is immediate to notice that there exists a minimal common supersequence x_1 of C_s which is a subsequence of x. Clearly $|x_1| \leqslant |x| = k$. By minimality, each symbol of x_1 appears in some sequence in C_s. Moreover the alphabet Σ_1 of symbols appearing in at least one sequence of C_s contains at most k symbols, since otherwise all supersequences of C_s would be longer than k. Consequently there are at most k^k supersequences no longer than k that are taken from the alphabet Σ_1. Our algorithm for a generic C_s enumerates all such supersequences s_c, and applies the algorithm for $|C_s| = 1$ on the new set of constraint sequences made only of s_c, returning the longest feasible solution computed.

The overall time complexity is clearly $k^k T(k, |s_1|, |s_2|)$, where $T(k, |s_1|, |s_2|) = (|s_1|\log|s_1|)2^{O(k)} + O(|s_1||s_2||s_c| \times 2^{O(k)}\log|\widetilde{\Sigma}|)$.

4. W[1]-hardness of C-LCS

In this section we prove that determining if there exists a feasible solution of C-LCS, is not only NP-complete, but also W[1]-hard when the parameter is the number of strings in C_s and the alphabet Σ (see [9] for a description of the consequences of W[1]-hardness).

We reduce the Shortest Common Supersequence (SCS) problem parameterized by the number of input strings and the size of alphabet Σ, which is known to be W[1]-hard [10]. Let $R = \{r_1, \ldots, r_k\}$ be a set of sequences over the alphabet Σ, hence R is a generic instance of the SCS problem. In what follows we denote by l the size of a solution of R.

The input of the C-LCS problem consists of two sequences s_1, s_2, and a string constraint C_s. Let # be a delimiter symbol not in Σ. Pose $C_s = \{\#^l\} \cup R$. Let w be a sequence over Σ such that w contains exactly one occurrence of each symbol in Σ, and let $\mathrm{rev}(w)$ be the reversal of w. Finally, let $s_1 = (w\#)^l$ and $s_2 = (\mathrm{rev}(w)\#)^l$. In the following we call each occurrence of w or of $\mathrm{rev}(w)$ a *block*.

Let t be any supersequence of $\#^l$ that is also a common subsequence of s_1 and s_2. Since in each of those sequences there are l #s, then also t must contain l #s, which in turn implies that by construction of s_1 and s_2, at most one symbol of each block can be in t. Therefore, t contains at most $2l$ symbols. At the same time, let p be a generic sequence over alphabet $\Sigma \cup \{\#\}$ no longer than $2l$, ending with a # and such that no two symbols from Σ appear consecutively in p. Since each symbol of Σ occurs exactly once in w, it is clear that p is a common subsequence of s_1 and s_2. Consequently, the set of all supersequences of $\#^l$ that are also common subsequences of s_1 and s_2 is equal to the set of sequences q that are no longer than $2l$ and such that (i) q contains exactly l #s, (ii) q ends with a #, and (iii) taking two consecutive symbols from q, exactly one of these symbols is equal to #.

An immediate consequence is that there exists a feasible solution of length $2l$ of the instance of C-LCS consisting of the set C_s and the two sequences s_1 and s_2 if and only if there exists a supersequence of length l of the set R of sequences. Indeed, assume that $q = y_1 y_2 \cdots y_l$ is a supersequence of length l of the set R of sequences. Then $p = y_1\#y_2\#\cdots y_l\#$ is a supersequence of q, hence also of each sequence in C_s. Furthermore, by construction p is a subsequence of s_1 and s_2. Assume now that $p = y_1\#y_2\#\cdots y_l\#$ is a solution of the instance of C-LCS made of the set C_s and the two sequences s_1 and s_2. Hence p is a supersequence of each sequence in the set R. The sequences in R are over alphabet Σ (hence they do not contain #), therefore $q = y_1 y_2 \cdots y_l$ is a supersequence of length l of the set R of sequences.

The reduction described is an FPT-reduction [9]. Finally, notice that the W[1]-hardness of C-LCS with parameters $|C_s|$ and $|\Sigma|$ implies the W[1]-hardness of DC-LCS with parameters $|C_s|$ and $|\Sigma|$, since C-LCS is a restriction of the DC-LCS problem.

Moreover, notice that the same reduction can be applied starting from the SCS problem over the binary alphabet, which is NP-hard [21], implying that the DC-LCS problem is NP-hard over a fixed ternary alphabet.

Acknowledgements

We would like to thank the anonymous reviewers whose suggestions have greatly contributed to improving the presentation of the paper. P.B., G.D.V. and Y.P. have been partially supported by FAR grants “Metodi algoritmici innovativi per confrontare strutture combinatorie in biologia computazionale”, and “Metodi algoritmici per l’analisi di strutture combinatorie in bioinformatica”. R.D. has been partially supported by FAR 2009 grant “Algoritmi per il trattamento di sequenze”.

References

[1] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, 2nd edition, MIT Press, 2001.
[2] L. Bergroth, H. Hakonen, T. Raita, A survey of longest common subsequence algorithms, in: Proc. 7th International Symp. on String Processing and Information Retrieval (SPIRE), IEEE Computer Society, Los Alamitos, CA, USA, 2000, pp. 39–48.
[3] M.S. Paterson, V. Dancik, Longest common subsequences, in: Proc. 19th Symp. on Mathematical Foundations of Computer Science (MFCS), 1994, pp. 127–142.
[4] A. Apostolico, Improving the worst-case performance of the Hunt–Szymanski strategy for the longest common subsequence of two strings, Information Processing Letters 23 (1986) 63–69.

[5] A. Apostolico, C. Guerra, The longest common subsequence problem revisited, Algorithmica 18 (1) (1987) 1–11.
[6] A. Apostolico, S. Browne, C. Guerra, Fast linear-space computations of longest common subsequences, Theoretical Computer Science 92 (1) (1992) 3–17.
[7] D. Maier, The complexity of some problems on subsequences and supersequences, Journal of the ACM 25 (1978) 322–336.
[8] T. Jiang, M. Li, On the approximation of shortest common supersequences and longest common subsequences, SIAM Journal on Computing 24 (5) (1995) 1122–1139.
[9] R. Downey, M. Fellows, Parameterized Complexity, Springer-Verlag, 1999.
[10] K. Pietrzak, On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems, Journal of Computer and System Sciences 67 (4) (2003) 757–771.
[11] P. Bonizzoni, G. Della Vedova, R. Dondi, G. Fertin, R. Rizzi, S. Vialette, Exemplar longest common subsequence, IEEE/ACM Trans. on Computational Biology and Bioinformatics 4 (4) (2007) 535–543.
[12] S.S. Adi, M.D.V. Braga, C.G. Fernandes, C.E. Ferreira, F.V. Martinez, M.-F. Sagot, M.A. Stefanes, C. Tjandraatmadja, Y. Wakabayashi, Repetition-free longest common subsequence, Electronic Notes in Discrete Mathematics 30 (2008) 243–248.
[13] D. Sankoff, Genome rearrangement with gene families, Bioinformatics 15 (11) (1999) 909–917.
[14] C.G. Fernandes, C. Ferreira, C. Tjandraatmadja, Y. Wakabayashi, A polyhedral investigation of the LCS problem and a repetition-free variant, in: Proc. 8th Latin American Theoretical Informatics Symposium (LATIN), in: LNCS, vol. 4957, Springer, 2008, pp. 329–338.
[15] Y.-T. Tsai, The constrained longest common subsequence problem, Information Processing Letters 88 (4) (2003) 173–176.
[16] A.N. Arslan, Ö. Egecioglu, Dictionary look-up within small edit distance, International Journal on Foundations of Computer Science 15 (1) (2004) 57–71.
[17] F.Y.L. Chin, A. De Santis, A.L. Ferrara, N.L. Ho, S.K. Kim, A simple algorithm for the constrained sequence problems, Information Processing Letters 90 (4) (2004) 175–179.
[18] Z. Gotthilf, D. Hermelin, M. Lewenstein, L.C.S. Constrained, Hardness and approximation, in: Proc. 19th Symp. on Combinatorial Pattern Matching (CPM), in: LNCS, vol. 5029, Springer, 2008, pp. 255–262.
[19] N. Alon, R. Yuster, U. Zwick, Color-coding, Journal of the ACM 42 (4) (1995) 844–856.
[20] J.P. Schmidt, A. Siegel, The spatial complexity of oblivious k-probe hash functions, SIAM Journal on Computing 19 (5) (1990) 775–786.
[21] K.-J. Räihä, E. Ukkonen, The shortest common supersequence problem over binary alphabet is NP-complete, Theoretical Computer Science 16 (1981) 187–198.