Polynomial-Time Approximation Algorithms for Weighted LCS Problem*

Marek Cygan[1], Marcin Kubica[1], Jakub Radoszewski[1], Wojciech Rytter[1,2], and Tomasz Waleń[1]

[1] Dept. of Mathematics, Computer Science and Mechanics, University of Warsaw, Warsaw, Poland
[cygan,kubica,jrad,rytter,walen]@mimuw.edu.pl
[2] Dept. of Math. and Informatics, Copernicus University, Toruń, Poland

Abstract. We deal with a variant of the well-known Longest Common Subsequence (LCS) problem for weighted sequences. A (biological) weighted sequence determines the probability for each symbol to occur at a given position of the sequence (such sequences are also called Position Weighted Matrices, PWM). Two possible such versions of the problem were proposed by (Amir et al., 2009 and 2010), they are called LCWS and LCWS2 (Longest Common Weighted Subsequence 1 and 2 Problem). We solve an open problem, stated in conclusions of the paper by Amir et al., of the tractability of a log-probability version of LCWS2 problem for bounded alphabets, showing that it is NP-hard already for an alphabet of size 2. We also improve the $(1/|\Sigma|)$-approximation algorithm given by Amir et al. (where Σ is the alphabet): we show a polynomial-time approximation scheme (PTAS) for the LCWS2 problem using $O(n^5)$ space. We also give a simpler $(1/2)$-approximation algorithm for the same problem using only $O(n^2)$ space.

1 Introduction

We consider (biological) *weighted sequences*, in which for each position we know the probability of an occurrence of any symbol from the alphabet Σ (more formal definition follows). Weighted sequences are also referred to in the literature as p-weighted sequences or Position Weighted Matrices (PWM) [2,16]. The notion of weighted sequence was introduced as a tool for motif discovery and local alignment, and is extensively used in computational molecular biology. In particular, binding sites, profiles of protein families and complete chromosome sequences, that have been obtained using a whole-genome shotgun strategy [15,17] can be modelled as weighted sequences [10].

* The first author is supported by grant no. N206 355636 of the Polish Ministry of Science and Higher Education. The third author is supported by grant no. N206 568540 of the National Science Centre. The fourth author is supported by grant no. N206 566740 of the National Science Centre.

Multiple algorithmic results related to combinatorics of weighted sequences, i.e., repetitions, regularities and pattern matching, have already been presented. The basic concepts (including pattern matching, repeats discovery and cover computation) were studied using three different approaches: weighted suffix trees [11], Crochemore partition [7] utilized in [13] and Karp-Miller-Rabin algorithm [8] utilized in [6]. There are also results dealing with: approximate and gapped pattern matching [3,20], property matching [1], swapped matching [19], all-covers and all-seeds problem [18,21], and extracting motifs (repeated motifs, common motifs and all maximal pairs) from weighted sequences [14]. On the practical side, there are recent results concerning massive exact and approximate pattern matching for mapping short weighted sequences to a reference genome [4], also in the parallel setting [12].

Recently Amir et al. [2] extended another well-known string problem, the Longest Common Subsequence problem [5], to weighted sequences. They introduced two versions of the Longest Common Weighted Subsequence problem, LCWS and LCWS2. Despite their similarity the complexity status of these problems is dramatically different, the first has a polynomial time solution, while the latter one is NP-hard. The results from [1,3] are also related to the LCS problem for weighted sequences, however none of the papers considers the LCS problem explicitly (the first one defines and considers weighted Hamming and edit distances, while the second considers a general property matching setting). Moreover, all these papers are limited by the assumption that patterns are ordinary strings.

The main problem considered in this paper is LCWS2. We solve an open problem stated by Amir et al. [2] and show that a log-probability version of LCWS2 is NP-hard for a bounded alphabet, moreover, even for alphabet of size 2 — the proof can be found in Section 3. We also improve the $(1/|\Sigma|)$-approximation algorithm for LCWS2 proposed in [2] by providing a polynomial-time approximation scheme (PTAS) for LCWS2. Note that obtaining a fully polynomial-time approximation scheme (FPTAS) for the LCWS2 problem is not possible, since this would imply tractability of LCWS2. Additionally we give a simpler $(1/2)$-approximation algorithm with smaller space requirements — it uses $O(n^2)$ space instead of $O(n^5)$ space needed by PTAS. Both algorithms are described in Section 4. We start by recalling the definitions of the respective problems, for which we propose a novel, more abstract formulation.

2 Preliminaries

Let Σ be a finite alphabet, $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_K\}$. We will be assuming for simplicity, as in most applications, that $K = |\Sigma| = O(1)$. By Σ^* we denote the set of all words over Σ. By Σ^d we denote the set of words of length d.

Definition 1 (Weighted sequence). *A weighted sequence $X = x_1 x_2 \ldots x_n$ of length $|X| = n$ over an alphabet $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_K\}$ is a sequence of sets of pairs of the form:*

$$x_i = \{(\sigma_j,\ p_i^{(X)}(\sigma_j))\ :\ j = 1, 2, \ldots, K\}.$$

If the considered weighted sequence is unambiguous, we will write p_i instead of $p_i^{(X)}$. Here $p_i(\sigma_j)$ is the occurrence probability of the character σ_j at the position i, these values are non-negative and sum up to 1 for a given i.
By $\mathcal{WS}(\Sigma)$ we denote the set of all weighted sequences over the alphabet Σ.

x_1	x_2	x_3	x_4
$p_1(\mathtt{a}) = 1/3$	$p_2(\mathtt{a}) = 1$	$p_3(\mathtt{a}) = 0$	$p_4(\mathtt{a}) = 1/2$
$p_1(\mathtt{b}) = 1/3$	$p_2(\mathtt{b}) = 0$	$p_3(\mathtt{b}) = 1/2$	$p_4(\mathtt{b}) = 1/4$
$p_1(\mathtt{c}) = 1/3$	$p_2(\mathtt{c}) = 0$	$p_3(\mathtt{c}) = 1/2$	$p_4(\mathtt{c}) = 1/4$

Fig. 1. A weighted sequence $X = x_1x_2x_3x_4$ over the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$

Now we recall the definitions of two versions of the Longest Common Weighted Subsequence problem [2]. We simplify their formulation by introducing an auxiliary notion of *α-subsequence*.

Let $X \in \mathcal{WS}(\Sigma)$. Let $\mathrm{Seq}_d^{|X|}$ be the set of all increasing sequences of d positions in X. For a string $s \in \Sigma^d$ and $\pi \in \mathrm{Seq}_d^{|X|}$, define $\mathcal{P}_X(\pi, s)$ as the probability that the substring on positions corresponding to π in X equals s. More formally, if $\pi = (i_1, i_2, \ldots, i_d)$ then

$$\mathcal{P}_X(\pi, s) = \prod_{k=1}^{d} p_{i_k}^{(X)}(s_k). \tag{1}$$

$$\text{Denote } SUBS(X, \alpha) = \left\{ s \in \Sigma^* : \exists \left(\pi \in \mathrm{Seq}_{|s|}^{|X|} \right) \ \mathcal{P}_X(\pi, s) \geq \alpha \right\}. \tag{2}$$

In other words $SUBS(X, \alpha)$ is the set of deterministic strings which match a subsequence of X with probability at least α. Every $s \in SUBS(X, \alpha)$ is called an *α-subsequence* of X. An α-subsequence of length d is also called an (α, d)-subsequence.

Example 1. Consider the weighted sequence from Fig. 1. The string $\mathtt{a}$ is its α-subsequence for any $\alpha \in (0, 1]$, the string $\mathtt{aba}$ is its $\frac{1}{4}$-subsequence, while the string $\mathtt{aaaa}$ is *not* its α-subsequence for any $\alpha > 0$.

Definition 2 (α-LCWS problem).
Input: *Two weighted sequences $X, Y \in \mathcal{WS}(\Sigma)$ and a cut-off probability α.*
Output: *The longest string $s \in \Sigma^*$ such that*

$$\exists \left(\pi \in \mathrm{Seq}_{|s|}^{|X|},\ \pi' \in \mathrm{Seq}_{|s|}^{|Y|} \right) \ \mathcal{P}_X(\pi, s) \cdot \mathcal{P}_Y(\pi', s) \geq \alpha.$$

Equivalently, s is the longest string in $SUBS(X, \alpha_1) \cap SUBS(Y, \alpha_2)$ for some $\alpha_1 \cdot \alpha_2 \geq \alpha$.

Definition 3 ((α_1, α_2)-LCWS2 problem).
Input: *Two weighted sequences X, Y and two cut-off probabilities α_1, α_2.*
Output: *The longest string $s \in SUBS(X, \alpha_1) \cap SUBS(Y, \alpha_2)$.*

The complexity status of these problems is dramatically different, the α-LCWS problem has a polynomial time solution, while a log-probability version of the (α_1, α_2)-LCWS2 problem is NP-hard.

Theorem 1. *[2]*
(a) *The α-LCWS problem can be solved in $O(n^3)$ time and $O(n^2)$ space. If we are only interested in the length of the output, the problem can be solved in $O(Ln^2)$ time, where L is the length of the solution.*
(b) *The log-probability version of the (α_1, α_2)-LCWS2 problem is NP-hard over unbounded alphabets and admits a $(1/|\Sigma|)$-approximation algorithm (thus the problem itself admits the same approximation algorithm).*

The main problem considered in this paper is LCWS2 (however, we also utilize the polynomial time solution of LCWS problem). We tackle its version with a single cut-off probability, stated in the following Definition 4, which is, by Lemma 1, equivalent to the general version with two parameters.

Definition 4 (α-LCWS2 problem).
Input: *Two weighted sequences $X, Y \in \mathcal{WS}(\Sigma)$ and a cut-off probability α.*
Output: *The longest string $s \in SUBS(X, \alpha) \cap SUBS(Y, \alpha)$.*

The following lemma shows that the (α_1, α_2)-LCWS2 and α-LCWS2 problems are equivalent.

Lemma 1. *The (α_1, α_2)-LCWS2 problem can be reduced in linear time to the $\min(\alpha_1, \alpha_2)$-LCWS2 problem.*

Lemma 1 is a consequence of the following claim.

Claim 2 *Let $X, Y \in \mathcal{WS}(\Sigma)$ and let $\alpha_1, \alpha_2 \in (0, 1]$, $\alpha_1 < \alpha_2$. Then there exist $X', Y' \in \mathcal{WS}(\Sigma')$, where $\Sigma' = \Sigma \cup \{\#\}$ is the original alphabet extended by a symbol $\# \notin \Sigma$, such that for any string s:*

$$s \text{ is a solution to the } (\alpha_1, \alpha_2)\text{-LCWS2 problem for } X \text{ and } Y \Leftrightarrow$$
$$s \text{ is a solution to the } \alpha_1\text{-LCWS2 problem for } X' \text{ and } Y'.$$

In particular, no solution to the α_1-LCWS2 problem for X' and Y' contains the symbol $\#$.

Moreover, $|X'| = |X|$, $|Y'| = |Y|$ and both weighted sequences can be constructed from X and Y in $O(n)$ time.

Proof. First assume that $\alpha_2 < 1$. Let $\gamma = \log_{\alpha_2} \alpha_1$. Recall that $\Sigma = \{\sigma_1, \ldots, \sigma_K\}$. We define weighted sequences X' and Y' over Σ' by the following probabilities:

$$p_i^{(X')}(\sigma_j) = p_i^{(X)}(\sigma_j), \quad p_i^{(X')}(\#) = 0$$
$$p_i^{(Y')}(\sigma_j) = p_i^{(Y)}(\sigma_j)^\gamma, \quad p_i^{(Y')}(\#) = 1 - \sum_{j=1}^{k} p_i^{(Y')}(\sigma_j).$$

It is easy to see that the conditions on the probabilities imposed by Definition 1 are satisfied — observe that $\gamma > 1$ (since $1 > \alpha_2 \geq \alpha_1$). We will prove that the following equality holds for this definition of X' and Y':

$$SUBS(X, \alpha_1) \cap SUBS(Y, \alpha_2) \;=\; SUBS(X', \alpha_1) \cap SUBS(Y', \alpha_1). \qquad (3)$$

Note that the left side of the equality (3) is a subset of Σ^*, while the right side is a subset of $(\Sigma')^*$. We prove (3) by showing two inclusions.

($\subseteq$) Let $s \in SUBS(X, \alpha_1) \cap SUBS(Y, \alpha_2)$, $s = s_1 s_2 \dots s_d$. Then obviously $s \in SUBS(X', \alpha_1)$. Let $\pi \in \mathrm{Seq}_d^{|Y|}$, $\pi = (i_1, i_2, \dots, i_d)$, be a sequence of positions for which $\mathcal{P}_Y(\pi, s) \geq \alpha_2$. Then the same sequence of positions shows that $s \in SUBS(Y', \alpha_1)$:

$$\mathcal{P}_{Y'}(\pi, s) \;=\; \prod_{j=1}^{d} p_{i_j}^{(Y)}(s_j)^{\gamma} \;=\; \mathcal{P}_Y(\pi, s)^{\gamma} \;\geq\; \alpha_2^{\gamma} \;=\; \alpha_1.$$

($\supseteq$) Let $s \in SUBS(X', \alpha_1) \cap SUBS(Y', \alpha_1)$, $s = s_1 s_2 \dots s_d$. First note that $s \in \Sigma^*$, otherwise s would not be an α_1-subsequence of X' (since $p_i^{(X')}(\#) = 0$ for all i). Hence, $s \in SUBS(X, \alpha_1)$. Let $\pi \in \mathrm{Seq}_d^{|Y'|}$, $\pi = (i_1, i_2, \dots, i_d)$, be a sequence of positions for which $\mathcal{P}_{Y'}(\pi, s) \geq \alpha_1$. Then the same sequence of positions shows that $s \in SUBS(Y, \alpha_2)$:

$$\mathcal{P}_Y(\pi, s) \;=\; \prod_{j=1}^{d} p_{i_j}^{(Y')}(s_j)^{1/\gamma} \;=\; \mathcal{P}_{Y'}(\pi, s)^{1/\gamma} \;\geq\; \alpha_2.$$

We are left with the case $\alpha_2 = 1$. If a string $s \in \Sigma^*$ is a 1-subsequence of the sequence Y, it cannot use any position i_j for a letter σ_j such that $p_{i_j}^{(Y)}(s_j) < 1$, hence we set:

$$\begin{aligned}
&p_i^{(X')}(\sigma_j) = p_i^{(X)}(\sigma_j), \quad p_i^{(X')}(\#) = 0,\\
&p_i^{(Y')}(\sigma_j) = 1 \quad \text{for } p_i^{(Y)}(\sigma_j) = 1 \text{ and } p_i^{(Y')}(\sigma_j) = 0 \text{ otherwise},\\
&p_i^{(Y')}(\#) = 1 - \sum_{j=1}^{k} p_i^{(Y')}(\sigma_j).
\end{aligned}$$

It is easy to check that $s \in \Sigma^*$ is a 1-subsequence of the sequence Y iff s is an α_1-subsequence of the sequence Y'. This concludes the proof of the claim. □

3 Integer LCWS2 over a Bounded Alphabet is NP-Hard

We recall the definition of the integer log-probability version of the LCWS2 problem as given by Amir et al. [2]. Define an *I-weighted sequence* X over the alphabet $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_K\}$ as a sequence of sets of pairs of the form:

$$x_i = \{(\sigma_j,\ w_i^{(X)}(\sigma_j)) \ :\ j = 1, 2, \dots, K\}, \quad \text{where } w_i^{(X)}(\sigma_j) \in \mathbb{Z}_+.$$

Let us introduce notations similar to (1) and (2). For an I-weighted sequence X and $s \in \Sigma^d$, define:

$$\mathcal{W}_X(\pi, s) \;=\; \sum_{k=1}^{d} w_{i_k}^{(X)}(s_k) \qquad \text{for } \pi = (i_1, \ldots, i_d) \in \mathrm{Seq}_d^{|X|}.$$

For an I-weighted sequence X and $\alpha \in \mathbb{Z}_+$, denote:

$$SUBS(X, \alpha) \;=\; \left\{ s \in \Sigma^* \;:\; \exists \left(\pi \in \mathrm{Seq}_{|s|}^{|X|} \right) \;\; \mathcal{W}_X(\pi, s) \le \alpha \right\}.$$

Using these notations, the LCIWS2 problem can be stated as follows:

Definition 5 (α-LCIWS2 problem).
Input: *Two I-weighted sequences X, Y and a cut-off value $\alpha \in \mathbb{Z}_+$.*
Output: *The longest string $s \in SUBS(X, \alpha) \cap SUBS(Y, \alpha)$.*

The previously known proof [2] of NP-hardness of the α-LCIWS2 problem depended on the assumption of an unbounded alphabet Σ. We show NP-hardness of α-LCIWS2 over the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.

For this, we perform a reduction of α-LCIWS2 to the following NP-complete problem [9] (the same NP-complete problem was utilized in [2]).

Definition 6 (Partition problem).
Input: *A finite set S, $S \subseteq \mathbb{Z}_+$.*
Binary output: *Is there a subset $S' \subseteq S$ such that $\sum S' = \sum S \setminus S'$.*

We make the reduction from the Partition problem to the LCIWS2 problem as follows. Let $S = \{q_1, q_2, \ldots, q_n\}$ be an instance of the Partition problem. We construct I-weighted sequences $X = x_1 x_2 \ldots x_n$ and $Y = y_1 y_2 \ldots y_n$ over the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ with the following weights of letters from Σ:

$$w_i^{(X)}(\mathtt{a}) = q_i + c, \quad w_i^{(X)}(\mathtt{b}) = c, \qquad w_i^{(Y)}(\mathtt{a}) = c, \quad w_i^{(Y)}(\mathtt{b}) = q_i + c.$$

Here $c > 0$ is an arbitrary positive integer. Finally let $\alpha = \frac{1}{2}\sum S + nc$.

Lemma 3. *The Partition problem for an instance S has a positive answer iff the length of the solution to α-LCIWS2 for X and Y is n.*

Proof. ($\Rightarrow$) Let $\pi = (i_1, i_2, \ldots, i_k)$ be an increasing sequence of positions such that $S' = \{q_{i_1}, q_{i_2}, \ldots, q_{i_k}\}$ is a solution to the Partition problem for S, let $\pi' = (i'_1, i'_2, \ldots, i'_{n-k})$ be the sequence of all remaining positions in S. Then the string $s \in \Sigma^n$ such that $s_{i_j} = \mathtt{a}$ for i_j in π and $s_{i'_j} = \mathtt{b}$ for i'_j in π' is a solution to the α-LCIWS2 problem for X and Y. Indeed, let $id_n = (1, 2, \ldots, n)$. Then:

$$\mathcal{W}_X(id_n, s) \;=\; \sum_i w_i^{(X)}(s_i) \;=\; \sum_{i_j} (q_{i_j} + c) + \sum_{i'_j} c \;=\; \sum S' + nc \;=\; \alpha,$$

$$\mathcal{W}_Y(id_n, s) \;=\; \sum_i w_i^{(Y)}(s_i) \;=\; \sum_{i_j} c + \sum_{i'_j} (q_{i'_j} + c) \;=\; \sum (S \setminus S') + nc \;=\; \alpha,$$

thus $s \in SUBS(X, \alpha) \cap SUBS(Y, \alpha)$. Hence, s is a solution to the α-LCIWS2 problem for X and Y, since it is the longest string in this set.

($\Leftarrow$) Let $s \in \Sigma^n$ be a solution to α-LCIWS2 problem for X and Y. Denote by $\pi = (i_1, i_2, \ldots, i_k)$ and $\pi' = (i'_1, i'_2, \ldots, i'_{n-k})$ the increasing sequences of positions within s containing letters `a` and `b` respectively. Then, for $id_n = (1, 2, \ldots, n)$, the following inequalities must hold:

$$\alpha \geq \mathcal{W}_X(id_n, s) = \sum_i w_i^{(X)}(s_i) = \sum_{i_j}(q_{i_j} + c) + \sum_{i'_j} c = \sum_{i_j} q_{i_j} + nc,$$
$$\alpha \geq \mathcal{W}_Y(id_n, s) = \sum_i w_i^{(Y)}(s_i) = \sum_{i_j} c + \sum_{i'_j}(q_{i'_j} + c) = \sum_{i'_j} q_{i'_j} + nc.$$

By recalling the definition of α and reducing equal addends, we obtain

$$\sum_{i_j} q_{i_j} \leq \frac{1}{2}\sum S \quad \text{and} \quad \sum_{i'_j} q_{i'_j} \leq \frac{1}{2}\sum S. \tag{4}$$

Note that both left sides of the inequalities (4) are non-negative and sum up to $\sum S$. Hence, both inequalities (4) are equalities, and therefore the set S' defined as $S' = \{q_{i_1}, q_{i_2}, \ldots, q_{i_k}\}$ is a solution to the Partition problem for the set S. □

Due to Lemma 3, we have a reduction from the Partition problem to the LCIWS2 problem, and even more, to a decision version of LCIWS2 (asking whether there exists a common subsequence of a given length). We conclude that the decision version of LCIWS2 problem is NP-complete, and moreover:

Theorem 2. *LCIWS2 problem over a* binary alphabet *is NP-hard.*

4 Approximating LCWS2

Previous work on the α-LCWS2 problem [2] contained a $(1/|\Sigma|)$-approximation algorithm. We start this section by presenting a $(1/2)$-approximation algorithm for α-LCWS2 and then proceed to a polynomial-time approximation scheme (PTAS) for this problem. The first algorithm is more space-efficient than the presented general approximation scheme.

4.1 (1/2)-Approximation Algorithm for LCWS2

Let $X, Y \in \mathcal{WS}(\Sigma)$, $n = \max(|X|, |Y|)$, and $\alpha \in (0, 1]$. By $\mathrm{OPT}(X, Y, \alpha)$ we denote the length of the solution to the α-LCWS2 problem for X and Y. In case the length of the solution to α^2-LCWS problem for X, Y is even, we obtain a $(1/2)$-approximation of $\mathrm{OPT}(X, Y, \alpha)$ simply by taking one half of the solution to the α^2-LCWS problem. In the case of odd length, we need to use the solutions to the α'-LCWS problem for all prefixes of X and Y, for different values of α', as shown in the proof of the following theorem.

Theorem 3.
(a) *We can compute a solution to the α-LCWS2 problem for $X, Y \in \mathcal{WS}(\Sigma)$ of length at least $\lfloor \mathrm{OPT}(X, Y, \alpha)/2 \rfloor$ in $O(n^3)$ time and $O(n^2)$ space.*
(b) *There exists a $(1/2)$-approximation algorithm for the α-LCWS2 problem which runs in $O(n^5)$ time and $O(n^2)$ space.*

Proof. (a) Let L be the size of the solution to the α-LCWS2 problem for X and Y and let d be the size of the solution to the α^2-LCWS problem for X and Y. We will show that $\lfloor \frac{d}{2} \rfloor \leq L \leq d$. This suffices to prove point (a), since d can be computed in $O(n^3)$ time and $O(n^2)$ space, see Theorem 1.

The inequality $d \geq L$ is a consequence of the following trivial inclusion:

$$SUBS(X, \alpha) \cap SUBS(Y, \alpha) \ \subseteq \ \{SUBS(X, \alpha_1) \cap SUBS(Y, \alpha_2) \ : \ \alpha_1 \cdot \alpha_2 \geq \alpha^2\},$$

in which the left side is the set of candidates for the solution of α-LCWS2 problem, while the right side are candidates for the solution to α^2-LCWS problem, both for the weighted sequences X and Y.

As for the other inequality, $L \geq \lfloor \frac{d}{2} \rfloor$, let $\pi \in \mathrm{Seq}_d^{|X|}$, $\pi = (i_1, \ldots, i_d)$, and $\pi' \in \mathrm{Seq}_d^{|Y|}$, $\pi' = (i'_1, \ldots, i'_d)$, be the sequences of positions corresponding to the solution $s = s_1 \ldots s_d$ of α^2-LCWS. Thus s satisfies:

$$\mathcal{P}_X(\pi, s) \cdot \mathcal{P}_Y(\pi', s) \ = \ \prod_{j=1}^{d} p_{i_j}^{(X)}(s_j) \cdot \prod_{j=1}^{d} p_{i'_j}^{(Y)}(s_j) \ \geq \ \alpha^2. \tag{5}$$

Let $g = \lfloor \frac{d}{2} \rfloor$. The left side of the inequality (5) can be written as $A \cdot B \cdot C \cdot D$, where:

$$A = \prod_{j=1}^{g} p_{i_j}^{(X)}(s_j), \quad B = \prod_{j=1}^{g} p_{i'_j}^{(Y)}(s_j), \quad C = \prod_{j=g+1}^{d} p_{i_j}^{(X)}(s_j), \quad D = \prod_{j=g+1}^{d} p_{i'_j}^{(Y)}(s_j).$$

Note that at most one of the numbers A, B, C, D can be less than α. Indeed, otherwise the product of these numbers would certainly be less than α^2, since all of them are at most 1. Hence:

$$(A \geq \alpha \ \wedge \ B \geq \alpha) \quad \vee \quad (C \geq \alpha \ \wedge \ D \geq \alpha).$$

Consequently, at least one of the strings $s_1 \ldots s_g$ or $s_{g+1} \ldots s_d$ is a solution to the α-LCWS2 problem, therefore $L \geq \lfloor \frac{d}{2} \rfloor$.

(b) If d is odd we need to additionally check if $L \geq \lceil \frac{d}{2} \rceil$. For this, we iterate over all possibilities of the last positions of the α-subsequence of length L within X and Y and all letters that could be the last letter of the resulting string. For every such choice we obtain an instance of the (α_1, α_2)-LCWS2 problem for some α_1, α_2, which we transform into an instance with a single cut-off probability using Lemma 1. Recall that the alphabet is of a constant size. We have $O(n^2)$ pairs of last positions, so the complexity grows by a quadratic factor. □

4.2 Polynomial-Time Approximation Scheme for LCWS2

Let $X, Y \in \mathcal{WS}(\Sigma)$, $n = \max(|X|, |Y|)$, and $\alpha \in (0, 1]$. We say that an instance (X, Y, α) of the α-LCWS2 problem is a (γ, T)-*power* if all the weights in the sequence X are powers of γ, where $0 < \gamma < 1$ and $\gamma^{T-1} \geq \alpha > \gamma^T$.

Lemma 4. *The α-LCWS2 problem for (γ, T)-power instances can be solved in $O(n^3 T)$ time and space.*

Proof. Without the loss of generality, we can assume, that $m = |Y| \leq |X| = n$. We use the dynamic programming technique. Our approach is a generalisation of the standard LCS algorithm. We have $O(n^3 T)$ states, each described by a tuple (a, b, ℓ, t), where:

- a is the position in the sequence X, $1 \leq a \leq n$;
- b is the position in the sequence Y, $1 \leq b \leq m$;
- ℓ is the length of the subsequence already chosen, $0 \leq \ell \leq m$;
- t is a γ-based logarithm of the product of $p_i(\sigma_j)$ values of the chosen subsequence of X; by the definition of the (γ, T)-power, we only consider integral values of t from the interval $[0, T-1]$.

For a state (a, b, ℓ, t) we store, as $\mathrm{val}(a, b, \ell, t)$, the greatest value β such that there exists a string $z = z_1 z_2 \dots z_\ell \in \Sigma^\ell$ that is a (β, ℓ)-subsequence of $y_1, \dots, y_b$, and there exists a sequence of positions $\pi \in \mathrm{Seq}_\ell^a$ for which

$$\log_\gamma \mathcal{P}_X(\pi, z) = t.$$

If no such (β, ℓ)-subsequence exists, we set $\mathrm{val}(a, b, \ell, t) = 0$.

Intuitively, in a state (a, b, ℓ, t) besides two positions a, b used in the classical LCS algorithm we control the length of the subsequence using the parameter ℓ and with the parameter t we count the number of times the value γ is multiplied in the product of probabilities used in the subsequence of the sequence X.

When computing the value for a state (a, b, ℓ, t), it suffices to consider three options: either drop the a-th position in the sequence X, or drop the b-th position in the sequence Y, or use one of the $|\Sigma|$ letters at the positions a, b in the sequence X and Y respectively. Formally, we have the following recursive formula:

$$\mathrm{val}(a, b, \ell, t) = \max\Big(\mathrm{val}(a-1, b, \ell, t),\ \mathrm{val}(a, b-1, \ell, t), \max_j \Big\{ p_b^{(Y)}(\sigma_j) \cdot \mathrm{val}(a-1, b-1, \ell-1, t - \log_\gamma(p_a^{(X)}(\sigma_j))) \Big\}\Big).$$

We set $\mathrm{val}(a, b, 0, t) = 1$, and for all remaining border cases $\mathrm{val}(a, b, \ell, t) = 0$. Thus we compute all the values $\mathrm{val}(a, b, \ell, t)$ in $O(n^3 T)$ time.

After computing values for all states we look for the greatest value of d for which there exists $t \in [0, T-1]$ such that $\mathrm{val}(|X|, |Y|, d, t) \geq \alpha$. By extending the dynamic programming algorithm in a standard manner we can construct a string $s \in \Sigma^d$ and two sequences of positions $\pi \in \mathrm{Seq}_d^{|X|}, \pi' \in \mathrm{Seq}_d^{|Y|}$ such that $\mathcal{P}_X(\pi, s), \mathcal{P}_Y(\pi', s) \geq \alpha$. □

Lemma 5. *For any $\epsilon > 0$ we can compute in $O(n^4/\epsilon)$ time and space a string which is an $\alpha^{1+\epsilon}$-subsequence of X and an α-subsequence of Y of length at least* $\mathrm{OPT}(X, Y, \alpha)$.

Proof. We assume that $\alpha < 1$, if $\alpha = 1$ we can solve the problem using a reduction to the standard LCS problem in $O(n^2)$ time and space.

We start by scaling and rounding probability distributions of the weighted sequence X. Let $T = \frac{n}{\epsilon}$ and $\gamma = \alpha^{1/T}$. For all i, j we set:

$$p'_i(\sigma_j) \;=\; \gamma^{\lfloor \log_\gamma(p_i^{(X)}(\sigma_j))\rfloor}.$$

Observe that:

$$p_i^{(X)}(\sigma_j) \;=\; \gamma^{\log_\gamma(p_i^{(X)}(\sigma_j))}, \quad p'_i(\sigma_j) \;\geq\; p_i^{(X)}(\sigma_j) \;\geq\; p'_i(\sigma_j)\alpha^{\epsilon/n}.$$

Hence, $p'_i(\sigma_j)$ is greater than $p_i^{(X)}(\sigma_j)$ at most by a factor of $\alpha^{-\epsilon/n}$. Now the conclusion follows from Lemma 4, since with the new weight p' for the sequence X the instance (X, Y, α) is a (γ, T)-power and multiplication of at most n weights differs from α by at most a factor of α^ϵ. Note that the new weight p' is not a probability distribution, as it does not satisfy the equality $\sum_{j=1}^{K} p'(\sigma_j) = 1$, however the algorithm from Lemma 4 does not use this assumption. □

Lemma 6. *Let (X, Y, α) be an instance of the LCWS2 problem. In $O(n^5)$ time and space one can find a string s which is an $(\alpha, d-1)$-subsequence of both X and Y such that no $(\alpha, d+1)$-subsequence of both X and Y exists.*

Proof. We set $\epsilon = 1/n$ and use the algorithm from Lemma 5 for the instance (X, Y, α, ϵ). Thus in $O(n^5)$ time and space we obtain a string $z \in \Sigma^d$ which is an $(\alpha^{1+1/n}, d)$-subsequence of X and an (α, d)-subsequence of Y. Note that no $(\alpha, d+1)$-subsequence of both X and Y exists. Now we remove exactly one character of the string z which has the smallest value of $p_{i_k}^{(X)}(z_k)$. Thus we obtain the final string $s \in \Sigma^{d-1}$ which is an $(\alpha, d-1)$-subsequence of both X and Y, since:

$$\alpha^{(1+1/n)(1-1/d)} \;\geq\; \alpha^{(1+1/n)(1-1/n)} \;=\; \alpha^{1-1/n^2} \;\geq\; \alpha. \quad \square$$

Theorem 4. *For any real value $\epsilon \in (0, 1]$ there exists a $(1-\epsilon)$-approximation algorithm for the LCWS2 problem which runs in polynomial time and uses $O(n^5)$ space. Consequently the LCWS2 problem admits a PTAS.*

Proof. The algorithm works as follows. Using the algorithm from Lemma 6 we find a positive integer d and an $(\alpha, d-1)$-subsequence.

If $d \geq 1/\epsilon$ then we are done since in that case we have $(d-1)/d = 1 - 1/d \geq 1 - \epsilon$ which means that we have found a $(1-\epsilon)$-approximation.

If $d < 1/\epsilon$ then we search for an (α, d)-subsequence using a brute-force approach, i.e., we try all $\binom{|X|}{d}, \binom{|Y|}{d}$ subsets of positions in each sequence and all $|\Sigma|^d$ possible strings of length d, which leads to $n^{O(1/\epsilon)}$ running time. If we find an (α, d)-subsequence we simply return it as the final answer. Otherwise we return the $(\alpha, d-1)$-subsequence found by the algorithm from Lemma 6. Either way when $d < 1/\epsilon$ our answer is optimal. □

Acknowledgements. We thank M. Christodoulakis, M. Crochemore, C. Iliopoulos and G. Tischler for helpful discussions about the weighted LCS problem.

References

1. A. Amir, E. Chencinski, C. S. Iliopoulos, T. Kopelowitz, and H. Zhang. Property matching and weighted matching. *Theor. Comput. Sci.*, 395(2-3):298–310, 2008.
2. A. Amir, Z. Gotthilf, and B. R. Shalom. Weighted LCS. *J. Discrete Algorithms*, 8:273–281, 2010.
3. A. Amir, C. S. Iliopoulos, O. Kapah, and E. Porat. Approximate matching in weighted sequences. In M. Lewenstein and G. Valiente, editors, *CPM*, volume 4009 of *Lecture Notes in Computer Science*, pages 365–376. Springer, 2006.
4. P. Antoniou, C. S. Iliopoulos, L. Mouchard, and S. P. Pissis. Algorithms for mapping short degenerate and weighted sequences to a reference genome. *I. J. Computational Biology and Drug Design*, 2(4):385–397, 2009.
5. L. Bergroth, H. Hakonen, and T. Raita. A survey of longest common subsequence algorithms. In *SPIRE*, pages 39–48, 2000.
6. M. Christodoulakis, C. S. Iliopoulos, L. Mouchard, K. Perdikuri, A. K. Tsakalidis, and K. Tsichlas. Computation of repetitions and regularities of biologically weighted sequences. *Journal of Computational Biology*, 13(6):1214–1231, 2006.
7. M. Crochemore. An optimal algorithm for computing the repetitions in a word. *Inf. Process. Lett.*, 12(5):244–250, 1981.
8. M. Crochemore and W. Rytter. *Jewels of Stringology.* World Scientific, 2003.
9. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.
10. D. Gusfield. *Algorithms on Strings, Trees, and Sequences – Computer Science and Computational Biology.* Cambridge University Press, 1997.
11. C. S. Iliopoulos, C. Makris, Y. Panagis, K. Perdikuri, E. Theodoridis, and A. K. Tsakalidis. The weighted suffix tree: An efficient data structure for handling molecular weighted sequences and its applications. *Fundam. Inform.*, 71(2-3):259–277, 2006.
12. C. S. Iliopoulos, M. Miller, and S. P. Pissis. Parallel algorithms for degenerate and weighted sequences derived from high throughput sequencing technologies. In J. Holub and J. Zdárek, editors, *Stringology*, pages 249–262. Prague Stringology Club, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, 2009.
13. C. S. Iliopoulos, L. Mouchard, K. Perdikuri, and A. K. Tsakalidis. Computing the repetitions in a biological weighted sequence. *Journal of Automata, Languages and Combinatorics*, 10(5/6):687–696, 2005.
14. C. S. Iliopoulos, K. Perdikuri, E. Theodoridis, A. K. Tsakalidis, and K. Tsichlas. Motif extraction from weighted sequences. In A. Apostolico and M. Melucci, editors, *SPIRE*, volume 3246 of *Lecture Notes in Computer Science*, pages 286–297. Springer, 2004.
15. E. W. Myers and Celera Genomics Corporation. A whole-genome assembly of drosophila. *Science*, 287(5461):2196–2204, Mar. 2000.
16. J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, 22:4673–4680, 1994.

17. J. C. Venter and Celera Genomics Corporation. The sequence of the human genome. *Science*, 291:1304–1351, 2001.
18. H. Zhang, Q. Guo, J. Fan, and C. S. Iliopoulos. Loose and strict repeats in weighted sequences of proteins. *Protein and Peptide Letters*, 17(9):1136–1142(7), 2010.
19. H. Zhang, Q. Guo, and C. S. Iliopoulos. String matching with swaps in a weighted sequence. In J. Zhang, J.-H. He, and Y. Fu, editors, *CIS*, volume 3314 of *Lecture Notes in Computer Science*, pages 698–704. Springer, 2004.
20. H. Zhang, Q. Guo, and C. S. Iliopoulos. An algorithmic framework for motif discovery problems in weighted sequences. In T. Calamoneri and J. Díaz, editors, *CIAC*, volume 6078 of *Lecture Notes in Computer Science*, pages 335–346. Springer, 2010.
21. H. Zhang, Q. Guo, and C. S. Iliopoulos. Varieties of regularities in weighted sequences. In B. Chen, editor, *AAIM*, volume 6124 of *Lecture Notes in Computer Science*, pages 271–280. Springer, 2010.