A Multiobjective Approach to the Weighted Longest Common Subsequence Problem

David Becerra, Juan Mendivelso, and Yoan Pinzón

Universidad Nacional de Colombia
Facultad de Ingeniería
Department of Computer Science and Industrial Engineering
Research Group on Algorithms and Combinatorics (ALGOS-UN)
Carrera 30 No. 45-03. Edificio 453. Oficina 207. Bogotá, Colombia
{dcbecerrar,jcmendivelsom,ypinzon}@unal.edu.co

Abstract. Finding the Longest Common Subsequence in Weighted Sequences (WLCS) is an important problem in computational biology and bioinformatics. In this paper, we model this problem as a multiobjective optimization problem. As a result, we propose a novel and efficient algorithm that not only finds a WLCS but also the set of all possible solutions. The time complexity of the algorithm depends primarily on the number of length-1 common subsequences between the two input weighted sequences.

Keywords: longest common subsequence, weighted sequences, multiobjective optimization, bioinformatics

1 Introduction

Algorithmic studies over molecular data have allowed the concomitant development of valuable analysis in biological processes. Specifically, studies on comparative genomics have lead to the development of powerful data analysis tools that have been successfully applied in several contexts from gene functional annotation to phylogenomics and whole genome comparison [4].

Since the publication of the human genome in 2001 [11], weighted sequences, also called position weight matrices [13], have become a major area of research in computational biology. A weighted sequence can be defined as a sequence of character-sets where, at each position of the sequence, each character is associated to a weight (or frequency). Thus, weighted sequences allow a newer and more precise encoding paradigm that allows to model several biological processes. For instance, in a molecular weighted sequence, the characters can represent either nucleotides or amino acids, and the weight can model either the occurrence probability of a character or the stability contributed by the character to a molecular complex [9]. Weighted sequences can be used to represent a variety of sequence lengths from short sequences, like protein binding sites, to much larger sequences such as profiles of protein families or even a complete chromosome sequence [6].

In recent years, different research groups have been modeling several other biological processes through weighted sequences. In [14], weighted sequences were used to propose a simple modeling of the translation of gene expression and regulation. Later, new weighted sequence algorithms were given for DNA approximate matching [2]. Moreover, in [10], a novel data structure called weighted suffix tree was introduced. This structure can be used to compute repeats and covers, as well as to detect the longest common substring. More recently, weighted sequences were used to model gene expression data derived from DNA micro-array analysis [15]. Despite its wide

David Becerra, Juan Mendivelso, Yoan Pinzón: A Multiobjective Approach to the Weighted Longest Common Subsequence Problem, pp. 64–74.
Proceedings of PSC 2012, Jan Holub and Jan Žďárek (Eds.), ISBN 978-80-01-05095-8 © Czech Technical University in Prague, Czech Republic

range of utilities, it is important to consider that, in most sequence comparison methods, the quality of the results are significantly affected by small perturbations in the algorithmic methods and the data. Furthermore, there is a dearth of computational tools to compare sequences beyond a certain length and quality [3].

The Longest Common Subsequence (LCS) of a given set of sequences is one of the most used similarity measures in computational biology. Computing the LCS has been analytically shown to be intractable (NP- hard in the strong sense) even for sequences over a binary alphabet [12]; however, it can be solved for a fixed number of sequences in polynomial time via standard dynamic programming algorithms [7]. Then, the development of adequate algorithms for the variants of the LCS has become an increasing necessity, considering that efficient algorithms are an undeniable requirement for the analysis of high throughput sequencing data.

Particularly, the Weighted Longest Common Subsequence (WLCS) is a similarity measure between weighted sequences. The WLCS of two weighted sequences, X and Y, is the longest common subsequence such that the product of the weights associated to each character in the subsequence is greater or equal to given bounds for X and Y. It was proven that computing the WLCS is NP-hard for unbounded alphabets [1]; opposite to the case of the LCS, the tractability of the problem for a bounded alphabet is still an open problem. To the best of our knowledge, there are only two algorithms to solve the WLCS problem. Specifically, a $(1/|\Sigma|)$-approximation algorithm was presented in [1]; more recently, this algorithm was improved by means of a polynomial-time approximation scheme in [5]. In this paper, we propose an exact algorithm, based on a new concept of dominance, to find the WLCS of two weighted sequences over bounded alphabets (DNA). As an advantage, the proposed algorithm returns not only the longest common subsequence (and its length), but also the set of all dominant common subsequences.

The outline of the paper is as follows. The next section provides some definitions necessary to understand the essential features of this paper. Then, the framework to tackle the WLCS problem as a simple multiobjective problem is presented in §3. In §4, the proposed algorithm is described along with its time complexity analysis and an example. The concluding remarks are drawn in the last section.

2 Preliminaries

Let Σ be an alphabet of cardinality $\sigma = |\Sigma|$ which consists of a set of symbols.

Definition 1 (LONGEST COMMON SUBSEQUENCE). *The* LCS *problem for two input strings x and y consists of finding the longest sequence p such that p is a subsequence of both x and y.*

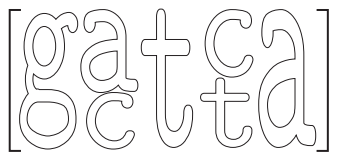
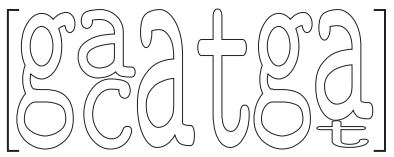
Definition 2 (WEIGHTED SEQUENCE). *A* weighted sequence $X = X[1] \cdots X[n]$ *over the alphabet Σ is a sequence of n sets $X[i]$, $1 \leq i \leq n$. Each set $X[i]$ is comprised of pairs $(s_j, \pi_i^X(s_j))$, where $s_j \in \Sigma$ and $\pi_i^X(s_j)$ is the* occurrence probability *of the character s_j at location i. Additionally, $\sum_j \pi_i^X(s_j) = 1$ for every position $1 \leq i \leq n$. For convenience and brevity, we will refer to $X[i]$ as the set of characters occurring at position i with a probability greater than zero.*

For a finite alphabet $\Sigma = \{s_1, \ldots, s_\sigma\}$, we can view a length-$n$ weighted sequence X as a $|\Sigma| \times n$ matrix A, where $A[j, i] = \pi_i^X(s_j)$. The elements of this matrix represent the occurrence probability of each character in every position of the sequence. Thus,

matrix A is comprised of values within the real interval [0,1]. As an example, two weighted sequences are illustrated in Fig. 1. Notice that for a given character the occurrence probability can be different at each position; however, for a given position, the occurrence probability of all characters must sum 1.

$$X = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{bmatrix} 0_{\mathtt{a}} & 0.6_{\mathtt{a}} & 0_{\mathtt{a}} & 0_{\mathtt{a}} & 1_{\mathtt{a}} \\ 0_{\mathtt{c}} & 0.4_{\mathtt{c}} & 0_{\mathtt{c}} & 0.5_{\mathtt{c}} & 0_{\mathtt{c}} \\ 1_{\mathtt{g}} & 0_{\mathtt{g}} & 0_{\mathtt{g}} & 0_{\mathtt{g}} & 0_{\mathtt{g}} \\ 0_{\mathtt{t}} & 0_{\mathtt{t}} & 1_{\mathtt{t}} & 0.5_{\mathtt{t}} & 0_{\mathtt{t}} \end{bmatrix} \end{array} \qquad Y = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\ \begin{bmatrix} 0_{\mathtt{a}} & 0.5_{\mathtt{a}} & 1_{\mathtt{a}} & 0_{\mathtt{a}} & 0_{\mathtt{a}} & 0.8_{\mathtt{a}} \\ 0_{\mathtt{c}} & 0.5_{\mathtt{c}} & 0_{\mathtt{c}} & 0_{\mathtt{c}} & 0_{\mathtt{c}} & 0_{\mathtt{c}} \\ 1_{\mathtt{g}} & 0_{\mathtt{g}} & 0_{\mathtt{g}} & 0_{\mathtt{g}} & 1_{\mathtt{g}} & 0_{\mathtt{g}} \\ 0_{\mathtt{t}} & 0_{\mathtt{t}} & 0_{\mathtt{t}} & 1_{\mathtt{t}} & 0_{\mathtt{t}} & 0.2_{\mathtt{t}} \end{bmatrix} \end{array}$$

(a)

$X =$  $Y =$ 

(b)

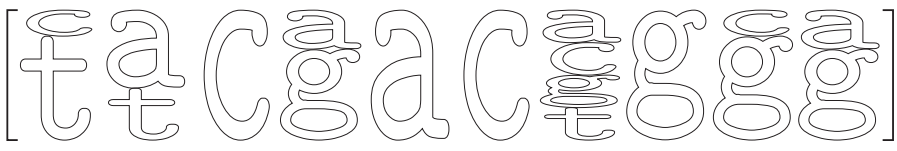
Figure 1. Example of 2 weighted sequences drawn from the alphabet $\Sigma_{\text{DNA}} = \{a, c, g, t\}$ shown in **(a)** as a matrix and in **(b)** as a 2-dimensional pictogram

Definition 3 (SUBSEQUENCE OF A WEIGHTED SEQUENCE). *For a given weighted sequence $X = X[1] \cdots X[n]$, $p = p_{i_1} \cdots p_{i_{|p|}}$ (where $p_{i_j} \in \Sigma$, $1 \leq j \leq |p|$ and $1 \leq i_1 < i_2 < \cdots < i_{|p|} \leq n$) is a* subsequence *of X iff $p_{i_j} \in X[i_j]$ for $1 \leq j \leq |p|$.*

Definition 4 (OCCURRENCE PROBABILITY OF A SUBSEQUENCE). *Let $p = p_{i_1} \cdots p_{i_{|p|}}$ be a subsequence of a weighted sequence $X = X[1] \cdots X[n]$. The* occurrence probability *of the subsequence p with respect to X, denoted as $\pi^X(p)$, is given by $\prod_{j=1}^{|p|} \pi_{i_j}^X(p_{i_j})$.*

See Fig. 2 for an example illustrating this last definition.

$$Z = \begin{array}{c} \begin{matrix} & i_1 & i_2 & i_3 & & & i_4 & & & i_5 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{matrix} \\ \begin{bmatrix} 0_{\mathtt{a}} & 0.6_{\mathtt{a}} & 0_{\mathtt{a}} & 0.3_{\mathtt{a}} & 1_{\mathtt{a}} & 0_{\mathtt{a}} & 0.25_{\mathtt{a}} & 0_{\mathtt{a}} & 0_{\mathtt{a}} & 0.3_{\mathtt{a}} \\ 0.2_{\mathtt{c}} & 0_{\mathtt{c}} & 1_{\mathtt{c}} & 0_{\mathtt{c}} & 0_{\mathtt{c}} & 1_{\mathtt{c}} & 0.25_{\mathtt{c}} & 0_{\mathtt{c}} & 0.2_{\mathtt{c}} & 0_{\mathtt{c}} \\ 1_{\mathtt{g}} & 0_{\mathtt{g}} & 0_{\mathtt{g}} & 0.7_{\mathtt{g}} & 0_{\mathtt{g}} & 0_{\mathtt{g}} & 0.25_{\mathtt{g}} & 1_{\mathtt{g}} & 0.8_{\mathtt{g}} & 0.7_{\mathtt{g}} \\ 0.8_{\mathtt{t}} & 0.4_{\mathtt{t}} & 0_{\mathtt{t}} & 0_{\mathtt{t}} & 0_{\mathtt{t}} & 0_{\mathtt{t}} & 0.25_{\mathtt{t}} & 0_{\mathtt{t}} & 0_{\mathtt{t}} & 0_{\mathtt{t}} \end{bmatrix} \end{array}$$

$Z =$ 

$p =$ `acata` with $i_j = (2,3,4,7,10)$ $\qquad \pi^Z(p) = 0.6 \times 1 \times 0.3 \times 0.25 \times 0.3 = 0.0135$

Figure 2. Example of a subsequence p extracted from a weighted sequence Z. Note that $p' = acata$ with $i_j = (2, 3, 5, 7, 10)$ is also a subsequence of Z but $\pi^Z(p') = 0.045$

Definition 5 (WEIGHTED LONGEST COMMON SUBSEQUENCE PROBLEM[1]). *Let $X = X[1] \cdots X[n]$ and $Y = Y[1] \cdots Y[m]$ be two weighted sequences. For two given constants α_1 and α_2 with $0 < \alpha_1, \alpha_2 \leq 1$, the* weighted longest common subsequence *is*

[1] Also known as the Longest Common Weighted Subsequence Problem with Two Thresholds.

the maximal integer length ℓ such that there is a common subsequence of length ℓ, $p = p_{i_1} \cdots p_{i_\ell}$, for which $\pi^X(p) \geq \alpha_1$ AND $\pi^Y(p) \geq \alpha_2$.

Definition 6 (DOMINANCE BETWEEN TWO COMMON SUBSEQUENCES). *Let $X = X[1] \cdots X[n]$ and $Y = Y[1] \cdots Y[m]$ be two weighted sequences where $n \leq m$. Also, let p and q be two length-h common subsequences of both X and Y where p and q occur in X at indices $(i_{x_1}, \ldots, i_{x_h})$ and $(j_{x_1}, \ldots, j_{x_h})$, respectively. Similarly, p and q occur at Y at indices $(i_{y_1}, \ldots, i_{y_h})$ and $(j_{y_1}, \ldots, j_{y_h})$, respectively. We say that p dominates q, denoted as $p \prec q$, iff:*

i) $\pi^X(p) \geq \pi^X(q)$
ii) $\pi^Y(p) \geq \pi^Y(q)$
iii) $i_{x_h} < j_{x_h}$
iv) $i_{y_h} < j_{y_h}$

Notice that if p dominates q, then the positions of the last character of p, in both X and Y, are lower than those of q. This fact is useful given that a common subsequence that ends at a lower index will have a better chance of being extended; thus, it may lead to longer common subsequences. On the other hand, the occurrence probabilities of p, with respect to X and to Y, are greater or equal to the ones of q. Consequently, if p dominates q, a possible LCS containing p will be at least as good as the LCS containing q.

Definition 7 (MULTIOBJECTIVE OPTIMIZATION PROBLEM – MOOP). *Find a vector $x = [x_1, x_2, \ldots, x_n]^T$ that:*

i) satisfies the r equality constraints $h_i(x) = 0,\ 1 \leq i \leq r$,
ii) is subject to the s inequality constraints $g_i(x) \geq 0,\ 1 \leq i \leq s$, and
iii) optimizes the vector function $f(x) = [f_1(x), \ldots, f_m(x)]^T$.

Then, it is clear that a MOOP problem focuses on searching for the optimal values of the decision variables (vector x) that minimize/maximize the objective function vector $f(x)$ while satisfying the constraints. The vector x is an n-dimensional decision vector or solution and $\mathcal{X}$ is the decision space, *i.e.*, the set of all expressible solutions. The objective vector $z = f(x)$ maps $\mathcal{X}$ into $\Re^m$, where $m \geq 2$ is the number of objectives. The image of $\mathcal{X}$ in the objective space, denoted as $\mathcal{Z}$, is the set of all attainable points (see Fig. 3).

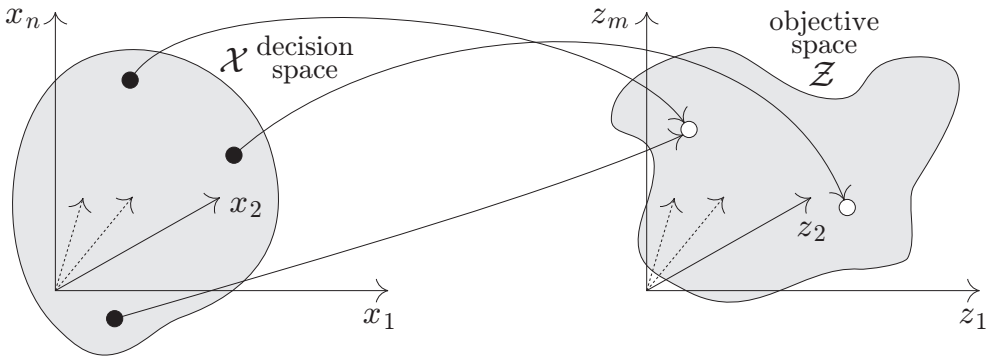


Figure 3. The n-dimensional parameter space maps to the m-dimensional objective space

3 WLCS as a Multiobjective Optimization Problem

In any multiobjective problem, two spaces should be defined: the *decision space* (the set of all expressible solutions), and the *objective space* (the space where the image of the solutions is the set of all attainable solutions). In general, to model any specific problem as a MOOP, three basic sets should be established: a set of objective functions, a set of decision variables and a set of equality/inequality constraints.

Particularly, for the WLCS problem, the length of the LCS should be maximized under two weight restrictions. Moreover, more than one possible direction of extending the substrings must be considered. Given two input weighted sequences, X and Y, and two constants $\alpha_1 > 0$ and $\alpha_2 > 0$, let $p = p_{i_1} \cdots p_{i_h}$ be a length-h common subsequence of X and Y that occurs at indices $(i_{x_1}, \ldots, i_{x_h})$ and $(i_{y_1}, \ldots, i_{y_h})$, respectively. Then, we define the elements of the MOOP model as follows:

– **Decision Space.** A vector with three components will be used as the decision vector. The first component is the list of symbols of the subsequence, *i.e.* $p_{i_1} \cdots p_{i_h}$. The second component is the list of indices at which the subsequence p occurs in X, *i.e.*, $(i_{x_1}, \ldots, i_{x_h})$. Finally, the third component is the list of indices at which the subsequence p occurs in Y, *i.e.*, $(i_{y_1}, \ldots, i_{y_h})$.

– **Objective space.** A vector with four components will be used as the objective function. The first objective, which will be maximized, is the occurrence probability of the subsequence with respect to X, *i.e.*, $\pi^X(p)$. The second objective, which will also be maximized, is the occurrence probability of the subsequence with respect to Y, *i.e.*, $\pi^Y(p)$. The third and fourth objectives, which will be minimized, are the positions of the last symbol of the subsequence in X, *i.e.* i_{x_h}, and in Y, *i.e.*, i_{y_h}, respectively. Then, the dominance relation between two common subsequences can be easily checked (*c.f.* see Definition 6).

– **Feasible regions.** The following constraints establish the feasible regions: the occurrence probabilitities of the subsequence p with respect to X and to Y must be greater or equal to α_1 and α_2, respectively. That is, $\pi^X(p) \geq \alpha_1$ and $\pi^Y(p) \geq \alpha_2$.

The proposed algorithm maximizes the length of the longest common subsequence by means of the objective functions. Its core idea is building different pareto optimum solutions, by using Definition 6, until the LCS is found in one of these optimum sets. Specifically, the algorithm iteratively finds new decision spaces as the result of the concatenation between the former pareto optimum set and the decision space of matches. The new decision space contains all the subsequences of length ℓ, while the former decision space contained all the subsequences of length $\ell - 1$. Then, these solutions are mapped to new objective spaces from which only the new pareto optimum set is extracted. The algorithm will continue until a new decision space can no longer be created (see Fig. 4, for an illustration of this process). In the next section, we describe in detail the proposed algorithm.

4 MiCO: An Algorithm for the WLCS Problem

Given two input weighted sequences, $X = X[1] \cdots X[n]$ and $Y[1] \cdots Y[m]$, and two given constants α_1 and α_2, where $0 < \alpha_1, \alpha_2 \leq 1$, our algorithm computes the

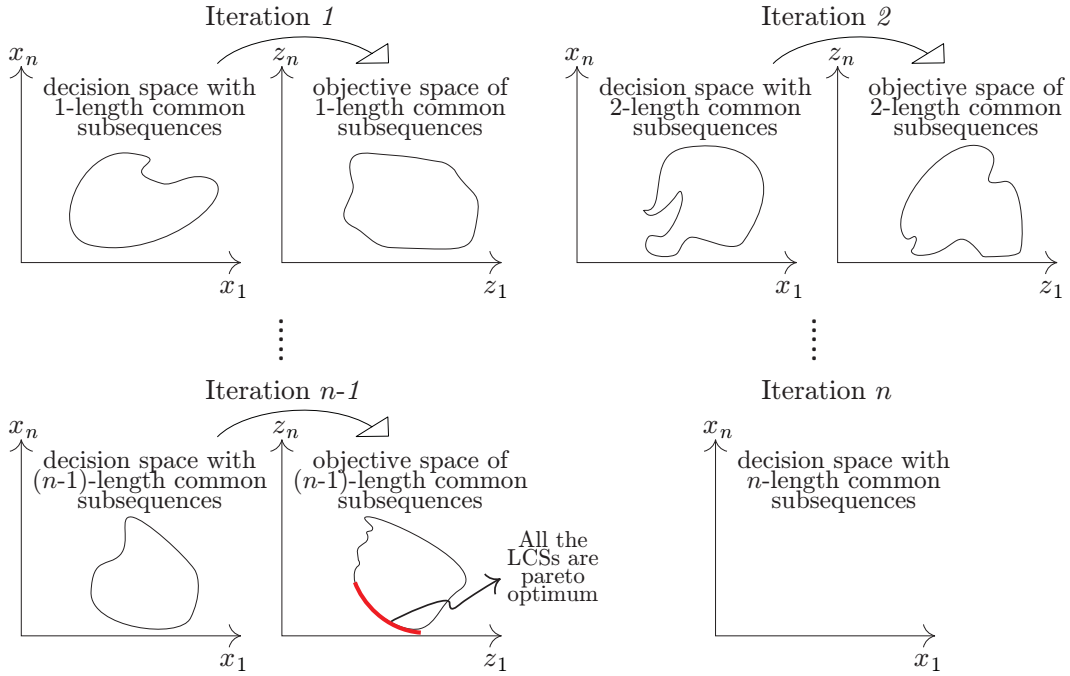


Figure 4. Iterations of the algorithm between the decision and objective spaces

weighted longest common subsequence of X and Y, along with all dominant common subsequences. The algorithm's framework is as follows:

- **Step 1 [computing 1-length common subsequences]:** Find a set Δ of all length-1 common subsequences of X and Y.
- **Step 2 [finding dominant common subsequences]:** Compute the set $\mathcal{D}_i$ of all the length-i common subsequences of X and Y that are not dominated by any other common subsequence. First, the dominant subsequences from Δ are added to $\mathcal{D}_1$. Then, the following two phases are performed for each $\delta \in \Delta$:
  - **Phase 1 [inserting concatenations with δ]:** Updates $\mathcal{D}_i$ by inserting all the new length-i common subsequences, for $i \geq 2$, resulting from the concatenation between each $d \in \mathcal{D}_{i-1}$ and the given δ.
  - **Phase 2 [deleting dominated subsequences]:** Updates $\mathcal{D}_i$ by deleting all the elements that were dominated by the common subsequences inserted during Phase 1.

While Step 1 is straightforward to implement, Step 2 is perhaps the trickiest step and needs a bit more of attention. Both of these steps are described below in more detail.

Step 1: To achieve the goal of this step, we begin by considering a matrix like the matrix shown in Fig. 6**(a)**. In such matrix $X[i]$, $1 \leq i \leq n$, and $Y[j]$, $1 \leq j \leq m$, represent a column and a row, respectively. We now can proceed to compute all the length-1 common subsequences in a *row-wise* fashion from top to bottom, as follows: First, X is traversed (from left to right) and all the positions i for which $\pi_i^X(\gamma) \geq \alpha_1$, for each character γ, are inserted into $List_X[\gamma]$. Thereafter, Y is traversed (from top to bottom) and, for each position j in Y, all the lists $List_X[\gamma]$ for which $\pi_j^Y(\gamma) \geq \alpha_2$ are merged-sorted into a list Γ containing (ℓ, γ), where $\ell \in List_X[\gamma]$. Then, we add

the triplet (j, ℓ, γ) to Δ for every $(\ell, \gamma) \in \Gamma$. Fig. 6**(b–c)** illustrates this operation on our running example.

Step 2: This is an iterative process that is performed once for each element $\delta \in \Delta$ computed in the previous step. Broadly, the main idea is to use Δ to compute the set $\mathcal{D}_i$ of the common subsequences with the following invariant condition of stability: *no element (common subsequence) in $\mathcal{D}_i$ dominates any other element in $\mathcal{D}_i$*. This basically means two things: *i*) a new common subsequence can be inserted into $\mathcal{D}_i$ iff it is not dominated by any other common subsequence currently in $\mathcal{D}_i$, and *ii*) all the common subsequences that are dominated by the new arriving subsequence should be deleted from $\mathcal{D}_i$.

MiCO Algorithm
Input: X, Y, Σ, α_1, α_2
Local Variables: $n \leftarrow |X|$, $m \leftarrow |Y|$, $\Delta \leftarrow \emptyset$
Output: $\mathcal{D}$

```
1  for i ← 1 to n do
2    forall γ ∈ Σ do
3        if π_i^X(γ) ≥ α_1 then    List_X[γ].add(i)
4  for j ← 1 to m do
5    Γ ← ∅
6    forall γ ∈ Σ
7        if π_j^Y(γ) ≥ α_2 then Γ.mergeSort(List_X[γ], γ)
8    forall (ℓ, γ) ∈ Γ do δ ← (j, ℓ, γ), Δ.add(δ)
9        if !Dominate(δ, D_1) then
10           D_1.add(δ)
11 i ← 2
12 forall d ∈ D_{i−1} do
13   forall δ ∈ Δ do
14           if ( Concatenate(d, δ) and !Dominate(dδ, D_i))then
15               D_i.add(dδ)
16               Delete(dδ, D_i)
17   if D_i = ∅ then
18               return D_{i−1}
19 i ← i + 1
```

Figure 5. MiCO Algorithm

The elements of $\mathcal{D}_1$ are the dominant length-1 subsequences extracted from Δ, while the elements of $\mathcal{D}_i$, for $i > 1$, are the dominant length-i subsequences generated by concatenating every $d \in \mathcal{D}_{i-1}$ with the elements $\delta \in \Delta$. When in Step 2 no subsequence is inserted, i.e. $\mathcal{D}_i = \emptyset$, this means that the length of the weighted LCS is $i-1$ and the algorithm returns all the set $\mathcal{D}_{i-1}$. In order to derive our final algorithm we need to define the following procedures:

- *Procedure* Concatenate(d, δ): Returns *true* if subsequence d can be concatenated with subsequence δ, (*i.e.* the positions of the row and the column of the last character of subsequence d are lower than the row and the column of δ, respectively), or *false* otherwise.
- *Procedure* Dominate$(d\delta, \mathcal{D})$: Returns *true* if subsequence $d\delta$ is dominated by any element $d \in \mathcal{D}$, or *false* otherwise.

– *Procedure* Delete($d\delta, \mathcal{D}$): Deletes all the elements of $\mathcal{D}$ that are dominated by d.

The pseudocode for the MiCO algorithm is presented in Fig. 5. The time complexity analysis and an example are presented in §4.1 and §4.2.

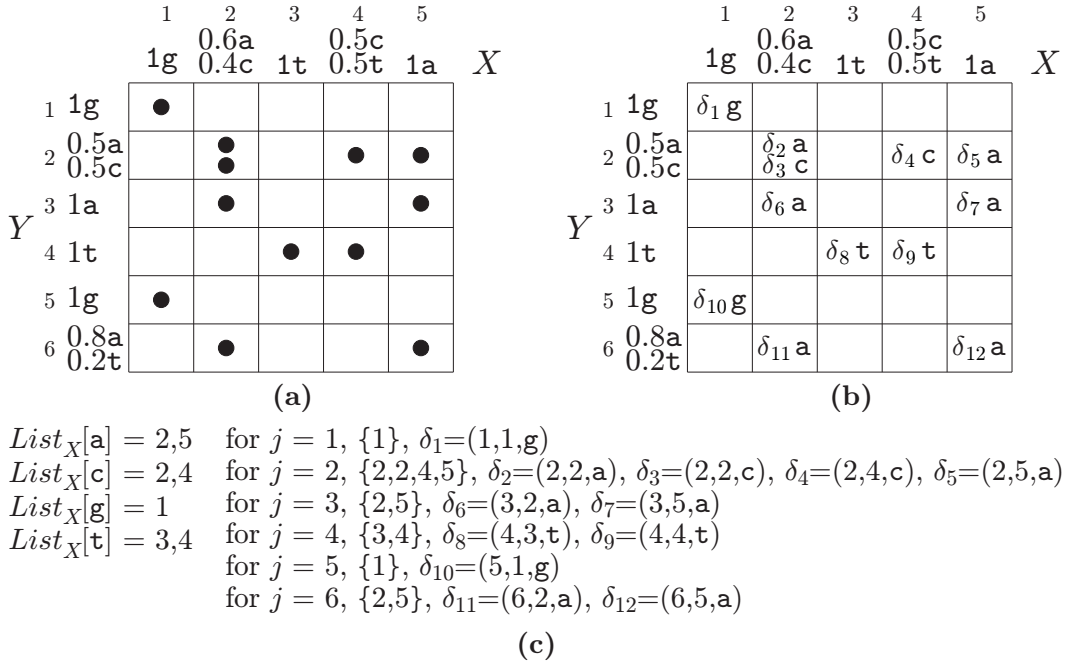


Figure 6. (a) Matrix representation of weighted sequences X and Y, **(b)** computation trace of Step 1, **(c)** resulting length-1 common subsequences for $\alpha_1 = \alpha_2 = 0.2$

4.1 Time Complexity Analysis

The time complexity of Step 1 (see MiCO pseudo-code, lines 1–8) is bounded by $\mathcal{O}(\sigma n + \sigma m) = \mathcal{O}(\sigma m)$. This process can be done in $\mathcal{O}(m \log m)$ time for the case where the sequences are unweighted [8]. However, when this approach is used on weighted sequences, its time complexity becomes $\mathcal{O}(\sigma m \log(\sigma m))$ which is worse than the time complexity of the implementation proposed for Step 1. Step 2, computed in lines 12–19 (see MiCO pseudo-code), can be implemented in $O(\ell|\Delta||\mathcal{D}|)$, where $|\Delta|$ is the number of length-1 common subsequences between the two input sequences, $|\mathcal{D}|$ is the size of the sets resulting from the concatenations, and ℓ is the length of the computed WLCS. Clearly the overall time complexity of the algorithm is determined by Step 2. The space complexity is bounded by O $(|\Delta||\mathcal{D}|)$. It should be remarked that $|\Delta|$ will decrease as the algorithm runs; on the other hand, $|\mathcal{D}|$ will tend to increase during the first stages of the running time and decrease during the last ones.

4.2 Example

For the input weighted sequences X and Y shown in Fig. 1 and two constants $\alpha_1 = 0.2$ and $\alpha_2 = 0.2$, Fig. 6**(c)** shows how to calculate the set of length-1 common subsequences, Δ, using the procedure described in Step 1 (MiCO pseudo-code, lines 1–8). Then, Fig. 7 shows all the iterations during Step 2. The set $\mathcal{D}_1$ is calculated by calling !Dominate($\delta, \mathcal{D}_1$) (MiCO pseudo-code, line 9–10). In the first iteration of the for loops (MiCO pseudo-code, lines 12–13), the set $\mathcal{F}_2$ is built by calling

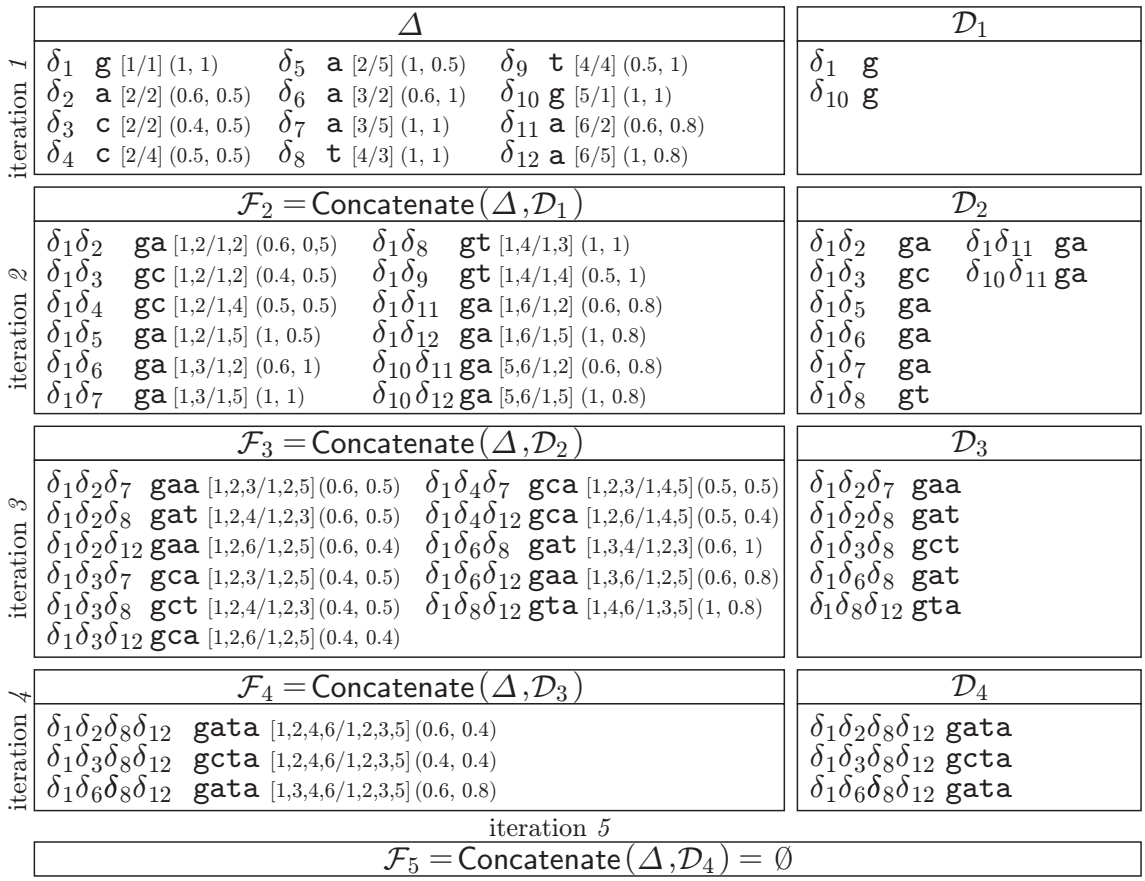


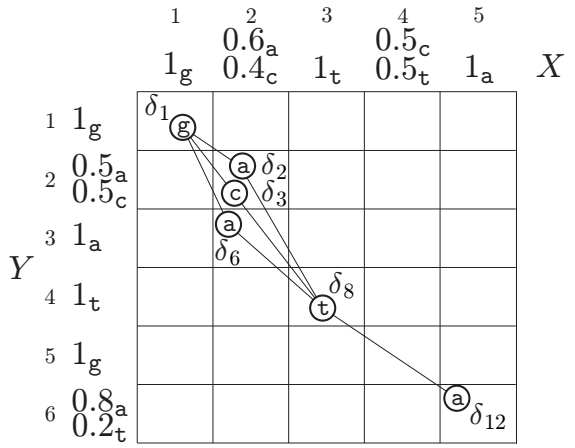
Figure 7. Computation trace of MiCO (Step 2). Each line represents a common subsequence p. The values [/] are the indices where p occurs in Y/X; (,) are $\pi^X(p)$ and $\pi^Y(p)$

Concatenate(d,δ) (MiCO pseudo-code, line 14), and the set $\mathcal{D}_2$ is established by calling !Dominate($d\delta, \mathcal{D}_1$) (MiCO pseudo-code, line 14). The algorithm performs the same procedure for the second and third iterations in which sets $\mathcal{D}_3$ and $\mathcal{D}_4$ are respectively computed. In the fifth iteration, the set $\mathcal{D}_5$ is empty after calling Concatenate(d,δ); thus, the set $\mathcal{D}_4$, consisting of three length-4 common subsequences of X and Y, is returned as the answer. Fig. 8 depicts, in a more schematic form, these longest common subsequences found by the proposed algorithm.

Notice that only the current set $\mathcal{D}$ needs to be stored; we do not need the additional data structure $\mathcal{F}_i$, it is only used for clarity purposes. Furthermore, there are some additional optimizations of the algorithm that were not illustrated in the example. For instance, the size of Δ can be decreased in each iteration and, therefore, we do not need to concatenate the elements $d \in \mathcal{D}_i$, at each iteration i, with all the elements δ in the initial set.

5 Conclusions

In this work we presented a new algorithm for the LCS problem applied to weighted sequences. This algorithm works under a multiobjective perspective, which allows tackling the LCS problem as an optimization problem. The time complexity of the algorithm depends on the total number of matches between the two input weighted



$\pi^X(\delta_1\delta_2\delta_8\delta_{12}) = \pi^X(\texttt{gata}) = 0.6$
$\pi^Y(\delta_1\delta_2\delta_8\delta_{12}) = \pi^Y(\texttt{gata}) = 0.4$

$\pi^X(\delta_1\delta_3\delta_8\delta_{12}) = \pi^X(\texttt{gcta}) = 0.4$
$\pi^Y(\delta_1\delta_3\delta_8\delta_{12}) = \pi^Y(\texttt{gcta}) = 0.4$

$\pi^X(\delta_1\delta_6\delta_8\delta_{12}) = \pi^X(\texttt{gata}) = 0.6$
$\pi^Y(\delta_1\delta_6\delta_8\delta_{12}) = \pi^Y(\texttt{gata}) = 0.8$

Figure 8. WLCS reported by MiCO

sequences, the number of dominant common subsequences detected during the computation and the length of the weighted longest common subsequence.

The main contributions of the paper can be summarized as follows: i) an algorithm that returns all the dominant LCSs between two weighted sequences, and ii) a framework to tackle the LCS problem as a multiobjective optimization problem.

References

1. A. Amir, Z. Gotthilf, and B. Shalom: *Weighted LCS.* Journal of Discrete Algorithms, 8 2010, pp. 273–281.
2. A. Amir, C. Iliopoulos, O. Kapah, and E. Porat: *Approximate matching in weighted sequences.* Lecture Notes on Computer Science, 4009 2006, pp. 365–376.
3. S. Bhowmick, M. Shafiullah, H. Rai, and D. Bastola: *A Parallel Non-Alignment Based Approach to Efficient Sequence Comparison using Longest Common Subsequences.* Journal of Physics: Conference Series, 256 2010, p. 012012.
4. P. Bonizzoni, G. Vedova, R. Dondi, G. Fertin, R. Rizzi, and S. Vialette: *Exemplar Longest Common Subsequence.* IEEE/ACM Transactions on Computational Biology and Bioinformatics, 4 2007, pp. 535–543.
5. M. Cygan, M. Kubica, J. Radoszewski, W. Rytter, and T. Walen: *Polynomial-Time Approximation Algorithms for Weighted LCS Problem*, in Combinatorial Pattern Matching, R. Giancarlo and G. Manzini, eds., vol. 6661 of Lecture Notes in Computer Science, Springer Berlin - Heidelberg, 2011, pp. 455–466.
6. D. Gusfield: *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge University Press, 1997.
7. W. Hsu and M. Du: *New algorithms for the LCS problem.* J. Computer and Systems Sciences, 19 1984, pp. 133–152.
8. J. Hunt and T. Szymanski: *A fast algorithm for computing longest common subsequences.* Communications of the ACM, 20(5) 1977, pp. 350–353.
9. C. Iliopoulos, C. Makris, Y. Panagis, K. Perdikuri, E. Theodoridis, and A. Tsakalidis: *Efficient algorithms for handling molecular weighted sequences.* Exploring New Frontiers of Theoretical Informatics, 155 2004, pp. 265–278.
10. C. Iliopoulos, C. Makris, Y. Panagis, K. Perdikuri, E. Theodoridis, and A. Tsakalidis: *The weighted suffix tree: An efficient data structure for handling molecular weighted sequences and its applications.* Fundamenta Informaticae, 71(2) 2006, pp. 259–277.
11. E. Lander: *Initial sequencing and analysis of the human genome.* Nature, 409 2001, pp. 860–921.
12. D. Maier: *The complexity of some problems on subsequences and supersequences.* Journal of the ACM (JACM), 25(2) 1978, pp. 322–336.

13. C. Makris and E. Theodoridis: *String Data Structures for Computational Molecular Biology.* Algorithms in Computational Molecular Biology: Techniques, Approaches and Applications, 1 2011, pp. 3–27.
14. K. Perdikuri and A. Tsakalidis: *Motif extraction from biological sequences: Trends and contributions to other scientific fields.* Information Technology and Applications, 2005. ICITA 2005. Third International Conference on, 1 2005, pp. 453–458.
15. O. Saetrom, O. Snove Jr, and P. Saetrom: *Weighted sequence motifs as an improved seeding step in microRNA target prediction algorithms.* RNA, 11(7) 2005, p. 995.