The longest common subsequence problem with crossing-free arc-annotated sequences

Guillaume Blin, Minghui Jiang, Stéphane Vialette

▶ **To cite this version:**

Guillaume Blin, Minghui Jiang, Stéphane Vialette. The longest common subsequence problem with crossing-free arc-annotated sequences. 19th edition of the International Symposium on String Processing and Information Retrieval (SPIRE 2012), Oct 2012, Cartagena de Indias, Colombia. pp. 130–142. hal-00713431

HAL Id: hal-00713431
https://hal.science/hal-00713431

Submitted on 1 Jul 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

The longest common subsequence problem with crossing-free arc-annotated sequences

Guillaume Blin[1], Minghui Jiang[2], and Stéphane Vialette[1]

[1] Université Paris-Est, LIGM - UMR CNRS 8049, France
{gblin,vialette}@univ-mlv.fr
[2] Department of Computer Science, Utah State University, USA
mjiang@cc.usu.edu

Abstract. An arc-annotated sequence is a sequence, over a given alphabet, with additional structure described by a - possibly empty - set of arcs, each arc joining a pair of positions in the sequence. As a natural extension of the longest common subsequence problem, Evans introduced the LONGEST ARC-PRESERVING COMMON SUBSEQUENCE (LAPCS) problem as a framework for studying the similarity of arc-annotated sequences. This problem has been studied extensively in the literature due to its potential application for RNA structure comparison, but also because it has a compact definition. In this paper, we focus on the nested case where no two arcs are allowed to cross because it is widely considered the most important variant in practice. Our contributions are three folds: (i) we revisit the nice **NP**-hardness proof of Lin et al. for LAPCS(NESTED, NESTED), (ii) we improve the running time of the FPT algorithm of Alber et al. from $O(3.31^{k_1+k_2}n)$ to $O(3^{k_1+k_2}n)$, where resp. k_1 and k_2 deletions from resp. the first and second sequence are needed to obtain an arc-preserving common subsequence, and (iii) we show that LAPCS(STEM, STEM) is **NP**-complete for constant alphabet size.

1 Introduction

Structure comparison for RNA has become a central computational problem bearing many computer science challenging questions. Indeed, RNA secondary structure comparison is essential for (i) identification of highly conserved structures during evolution (which cannot always be detected in the primary sequence, since it is often unpreserved) which suggest a significant common function for the studied RNA molecules, (ii) RNA classification of various species (phylogeny), (iii) RNA folding prediction by considering a set of already known secondary structures, and (iv) identification of a consensus structure and consequently of a common role for molecules. From an algorithmic point of view, RNA structure comparison was first considered in the framework of ordered trees [12] and, later on, in the one of *arc-annotated sequences* [5]. An *arc-annotated sequence* over some fixed alphabet Σ is a pair (S, P), where S (the *sequence*) is a string of Σ^* and P (the *annotation*) is a set of arcs $\{(i, j) : 1 \leq i < j \leq |S|\}$. In the context of RNA structures, S is a sequence of RNA bases and P represents hydrogen bonds between pairs of elements of S. From a purely combinatorial point of view, arc-annotated sequences are a natural extension of simple sequences. However, using arcs for modeling non-sequential information together with restrictions on the relative positioning of arcs allow for varying restrictions on the structure of arc-annotated sequences. Observe that a (plain) sequence without any arc can be viewed as an arc-annotated sequence with an empty arc set.

Different pattern matching and motif search problems have been considered in the context of arc-annotated sequences among which we can mention finding a *longest arc-annotated subsequence*, finding an *arc-preserving subsequence*, finding a *maximum arc-preserving common subsequence*, and computing the *edit distance for arc-annotated sequences*. The best overview references are [3] and [2].

In an arc-annotated sequence (S, P), two arcs (i_1, j_1) and (i_2, j_2) are crossing, if $i_1 < i_2 < j_1 < j_2$ or $i_2 < i_1 < j_2 < j_1$. An arc (i_1, j_1) is *nested* into an arc (i_2, j_2) if $i_2 < i_1 < j_1 < j_2$. In her pioneering work [4], Evans has introduced a five level hierarchy [1] for arc-annotated sequences that is described as follows: UNLIMITED: no restriction at all, CROSSING: each base is incident to at most one arc, NESTED: each base is incident to at most one arc and no two arcs are crossing, STEM: each base is incident to at most one arc, and given any two arcs one is nested into the other, and PLAIN: there is no arc. This hierarchy is clearly organized according to the following chain of inclusions: PLAIN $\subset$ STEM $\subset$ NESTED $\subset$ CROSSING $\subset$ UNLIMITED.

Let (S_1, P_1) and (S_2, P_2) be two arc-annotated sequences. If $S_1[i] = S_2[j]$ for some pair of integers i and j ($1 \leq i \leq |S_1|$ and $1 \leq j \leq |S_2|$), we refer to $\langle i, j\rangle$ as a *base-match*. If $S_1[i] = S_2[j]$ and $S_1[k] = S_2[l]$ with $(i, k) \in P_1$ and $(j, l) \in P_2$, we refer to the pair $(\langle i, k\rangle\rangle, \langle j, l\rangle)$ as an *arc-match*. A common subsequence T of S_1 and S_2 can be viewed as a set of pairwise disjoint base-matches $M = \{\langle i_k, j_k\rangle : 1 \leq k \leq |T|, 1 \leq i_k \leq |S_1|, 1 \leq j_k \leq |S_2|\}$ such that $\forall 1 \leq k_1 < k_2 \leq |T|$, $i_{k_1} < i_{k_2}$ and $j_{k_1} < j_{k_2}$ (*i.e.* preserving order). The common subsequence T is said to be *arc-preserving* if the arcs induced by M are preserved, *i.e.*, for any distinct $\langle i_{k_1}, j_{k_1}\rangle, \langle i_{k_2}, j_{k_2}\rangle \in M$, $(i_{k_1}, i_{k_2}) \in P_1$ if and only if $(j_{k_1}, j_{k_2}) \in P_2$. Among the many paradigms referring to arc-annotated sequences we focus here on the most natural extension of the longest common subsequence problem, the so-called LONGEST ARC-PRESERVING COMMON SUBSEQUENCE (LAPCS) problem which is defined as follows [4]: Given two arc-annotated sequences (S_1, P_1) and (S_2, P_2), find the longest common subsequence of S_1 and S_2 that is arc-preserving. It is well-known that the LAPCS problem is **NP**-complete [4].

The LAPCS problem is traditionally parameterized by the arc-structure of the two input arc-annotated sequences. We focus on the nested case because it is widely considered the most important variant in practice [10, 11, 1]. We denote by LAPCS(NESTED, NESTED) (resp. LAPCS(STEM, STEM) the LAPCS problem where both arc-annotated sequences are NESTED (resp. STEM). It has been shown in [9] that the LAPCS(NESTED, NESTED) problem is **NP**-complete, even for an unary alphabet. This result has been extended in [8] where it is shown that the LAPCS(STEM, STEM) problem is **NP**-complete. Alber et al. [1] presented two FPT algorithm for the LAPCS(NESTED,NESTED) problem. Given two arc-annotated sequences of maximum length n, their first algorithm decides in $O((3|\Sigma|)^{\ell}\, \ell n)$ time whether the two sequences have an arc-preserving common subsequence of length ℓ, and their second algorithm decides in $O(3.31^{k_1+k_2} n)$ time whether an arc-preserving common subsequence can be obtained by deleting k_1 letters from the first sequence and k_2 letters from the second sequence. Improving the exponential running times of the two algorithms was left as an immediate open question.

[1] Our presentation actually replaces the original CHAIN level with the STEM level due to its importance for practical issues [7].

Moreover, Alber et al. [1] noted that their second algorithm relies on a breadth-first search that is very space-consuming, and asked whether it can be replaced by a simple depth-first search. Our paper makes the following contributions. First, w e revisit the nice **NP**-hardness proof of Lin et al. [11] for the LAPCS(NESTED, NESTED) problem. We point out a problem an provide a simple solution. Second, we improve the running time of the (second) FPT algorithm of Alber et al. [1] from $O(3.31^{k_1+k_2}n)$ to $O(3^{k_1+k_2}n)$. Our algorithm uses the bounded search tree technique, and can be implemented using a simple depth-first search. Third, we show that the LAPCS(STEM, STEM) problem is **NP**-complete for constant alphabet size. The proof is by a tricky modification of [8].

2 LAPCS(NESTED, NESTED) is NP-complete

In this section we prove that the LAPCS(NESTED, NESTED) problem is **NP**-complete even if both arc-annotated sequences are unary. We actually point out a problem in a previous proof by Lin et al. [11] for the same result, and give a simple solution for the correctness of the proof. Our proof is for a large part the same as the proof of Lin et al. [11]. The only difference is that we use larger "*barriers*" of length $\Omega(n)$ each.

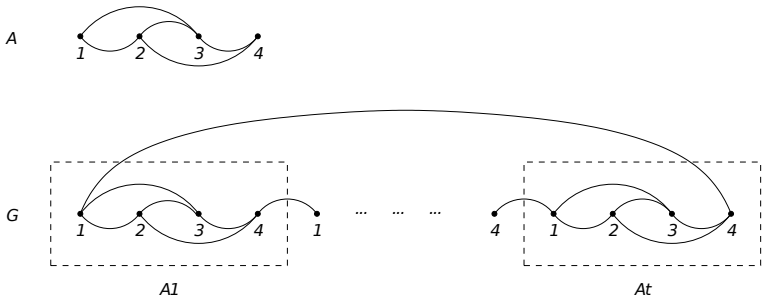

Fig. 1. The counter-example graph G.

Our counter-example graph for the proof of Lin et al. [11] is presented Figure 1. The graph A has 4 vertices v_1, v_2, v_3 and v_4. The graph G has $n = 4t$ vertices, and consists of t copies $A_1, A_2, \ldots, A_t$ of the graph A linked into a circular "list" (for convenience let $A_0 = A_t$ and $A_{t+1} = A_1$) by one additional edge from the vertex v_1 of each A_i to the vertex v_4 of A_{i-1}. One can easily verified that G is cubic, planar, bridgeless, and connected. Moreover, G has a natural two-page book embedding such that each vertex is incident to at least 1 and at most 2 edges on each page, as illustrated in Figure 1. We have the following lemma about the graph G.

Lemma 1. *The maximum cardinality k^* of an independent set in the graph G is $\lfloor \frac{3}{8} n \rfloor$.*

We now turn to pointing out the problem in the proof of Lin et al. [11]. Refer to Figure 2 for the construction of the two arc-annotated sequences P_1 and P_2 based on the graph G according to the reduction of Lin et al. [11]. As illustrated by the dotted lines

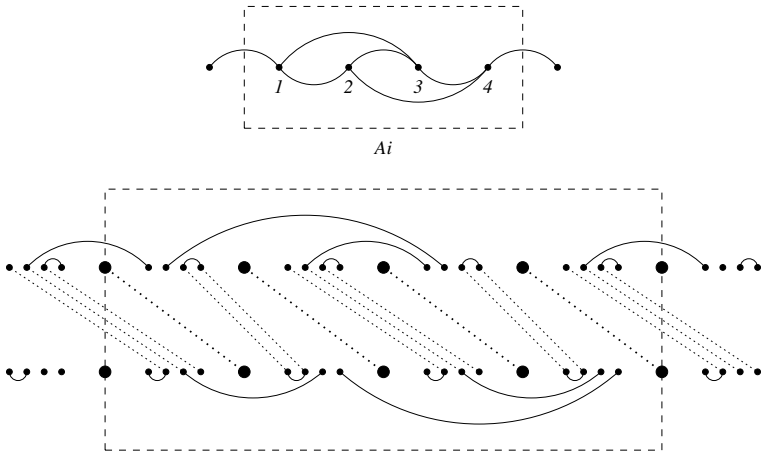

Fig. 2. The two arc-annotated sequences P_1 and P_2 for the graph G. The separating blocks, each of length 8, are illustrated by large dots.

between the two sequences, the two arc-annotated sequences (S_1, P_1) and (S_2, P_2) has an arc-preserving common subsequence of length $\ell = 8n + \frac{3+2}{2}n - 6 = 10n + \frac{1}{2}n - 6$. Lin et al. [11] claimed that every LAPCS can be transformed into a good LAPCS (of the same length). We show that this claim is wrong. Following their proof, the graph G has an independent set of cardinality k if and only if (S_1, P_1) and (S_2, P_2) have a good LAPCS of length $8(n+1) + 2n + k = 10n + k + 8$. Then, by Lemma 1, the maximum length of a good LAPCS of (S_1, P_1) and (S_2, P_2) is at most $\ell_{\text{good}} = 10n + \left\lfloor \frac{3}{8}n \right\rfloor + 8$. Note that for $t > 28$ and correspondingly $n = 4t > 112$, we have $\ell > \ell_{\text{good}}$. This disproves their claim. For a correct proof, we increase the length of each separating block in the reduction from 8 to $s = 4n$. Then, following their proof, the length of an LAPCS is at least $s(n+1) + 2n + k^*$. If a common subsequence has a *far* match $\langle i, j \rangle$ such that $|j - i| \geq n$, then in each sequence there must be at least n unmatched bases on each side of the match. It follows that the length of the common subsequence is at most $s(n+1) + 4n - 2n$, which is less than $s(n+1) + 2n + k^*$. Therefore every match $\langle i, j \rangle$ of an LAPCS must be *near*, i.e., $|j - i| < n$. By the same argument, an LAPCS must include at least one arc from each separating block in each sequence, because otherwise a separating block with no arcs in the LAPCS would have at least $4n/2 = 2n$ unmatched bases. Since all matches must be near, any arc (i_1, i_2) in the LAPCS that comes from a separating block in P_1 must match an arc (j_1, j_2) from the corresponding separating block in P_2 such that either $i_1 \leq j_1 \leq j_2 \leq i_2$ or $j_1 \leq i_1 \leq i_2 \leq j_2$. Then a simple replacement argument shows that all separating blocks are matched completely, and consequently any LAPCS can be transformed into a good LAPCS of the same length.

3 A faster algorithm for the LAPCS(NESTED, NESTED) problem

Theorem 1. *There is an $O(3^{k_1+k_2}n)$-time algorithm for* LAPCS(NESTED,NESTED) *that decides whether an arc-preserving common subsequence of two arc-annotated*

sequences of maximum length n can be obtained by deleting k_1 letters from the first sequence and k_2 letters from the second sequence.

We first observe that the two parameters k_1 and k_2 are not independent. Let n_1 and n_2 be the lengths of the two sequences. Then the problem admits a valid solution only if $n_1 - k_1 = n_2 - k_2$. Without loss of generality, we use a single parameter $k = k_1 + k_2$ for the total number of letters deleted from the two arc-annotated sequences. The running time of our algorithm is thus $O(3^k n)$. For an arc-annotated sequence S and an index i, define $\mathrm{buddy}(S, i) = j$ if $S[i]$ is connected to $S[j]$ by an arc, and $\mathrm{buddy}(S, i) = 0$ otherwise. For an arc-annotated sequence S of length n and two indices $i \leq j$, denote by $S[i, j]$ the subsequence obtained from S by deleting letters $S[1], S[2], \ldots, S[i-1]$ and $S[j+1], S[j+2], \ldots, S[n]$ together with the incident arcs. For an arc-annotated sequence S and three indices $i \leq j \leq k$, denote by $S[i, \bar{j}, k]$ the subsequence obtained from $S[i, k]$ by deleting $S[j]$ and its incident arc (if any).

Algorithm $\mathsf{lapcs}(S, T, k)$
Input: Two arc-annotated sequences S and T, an integer k.
Output: returns k^* – the minimum number of letters that must be deleted from S and T to obtain an arc-preserving common subsequence – if $k^* \leq k$; ∞ otherwise.

The algorithm is recursive. For the base case, the algorithm returns 0 if $S = T$ and $k \geq 0$, and returns ∞ if $S \neq T$ and $k \leq 0$. For the inductive case, the algorithm tries all applicable following cases and returns the minimum value. Let s and t be the lengths of the two sequences S and T, respectively. Put $i = \mathrm{buddy}(S, 1)$ and $j = \mathrm{buddy}(T, 1)$.

Case 1 $S[1] \neq T[1]$.

- Delete $S[1]$, then return $\mathsf{lapcs}(S[2, s], T, k-1) + 1$.
- Delete $T[1]$, then return $\mathsf{lapcs}(S, T[2, t], k-1) + 1$.

Case 2.1 $S[1] = T[1]$, $i = j = 0$.

- Match $S[1] \sim T[1]$, then return $\mathsf{lapcs}(S[2, s], T[2, t], k)$.

Case 2.2 $S[1] = T[1]$, $i > 0$ and $j = 0$.

- Delete $S[1]$, then return $\mathsf{lapcs}(S[2, s], T, k-1) + 1$.
- Delete $T[1]$, then return $\mathsf{lapcs}(S, T[2, t], k-1) + 1$.
- Delete $S[i]$, match $S[1] \sim T[1]$, then return $\mathsf{lapcs}(S[2, \bar{i}, s], T[2, t], k-1) + 1$.

Case 2.3 $S[1] = T[1]$, $i = 0$ and $j > 0$.

- Delete $S[1]$, then return $\mathsf{lapcs}(S[2, s], T, k-1) + 1$.
- Delete $T[1]$, then return $\mathsf{lapcs}(S, T[2, t], k-1) + 1$.
- Delete $T[j]$, match $S[1] \sim T[1]$, then return $\mathsf{lapcs}(S[2, s], T[2, \bar{j}, t], k-1) + 1$.

Case 2.4 $S[1] = T[1]$, $i > 0$ and $j > 0$, $S[i] \neq T[j]$.

- Delete $S[1]$, then return $\mathsf{lapcs}(S[2, s], T, k-1) + 1$.
- Delete $T[1]$, then return $\mathsf{lapcs}(S, T[2, t], k-1) + 1$.

– Delete $S[i]$ and $T[j]$, match $S[1] \sim T[1]$, then return $\mathsf{lapcs}(S[2, \overline{i}, s], T[2, \overline{j}, t], k - 2) + 2$.

Case 2.5.1 $S[1] = T[1]$, $i > 0$ and $j > 0$, $S[i] = T[j]$, $S[2, i-1] = T[2, j-1]$.

– Match $S[1, i] \sim T[1, j]$, then return $\mathsf{lapcs}(S[i+1, s], T[j+1, t], k)$.

Case 2.5.2 $S[1] = T[1]$, $i > 0$ and $j > 0$, $S[i] = T[j]$, $S[i+1, s] = T[j+1, t]$.

– Match $S[1] \sim T[1]$ and $S[i, s] \sim T[j, t]$, then return $\mathsf{lapcs}(S[2, i-1], T[2, j-1], k)$.

Case 2.5.3 $S[1] = T[1]$, $i > 0$ and $j > 0$, $S[i] = T[j]$, $\exists a : S[2, \overline{a}, i-1] = T[2, j-1]$.

– Delete $S[1]$, then return $\mathsf{lapcs}(S[2, s], T, k-1) + 1$.
– Delete $T[1]$, then return $\mathsf{lapcs}(S, T[2, t], k-1) + 1$.
– Delete $S[a]$, match $S[1, \overline{a}, i] \sim T[1, j]$, then return $\mathsf{lapcs}(S[i+1, s], T[j+1, t], k - 1) + 1$.

Case 2.5.4 $S[1] = T[1]$, $i > 0$ and $j > 0$, $S[i] = T[j]$, $\exists b : S[2, i-1] = T[2, \overline{b}, j-1]$.

– Delete $S[1]$, then return $\mathsf{lapcs}(S[2, s], T, k-1) + 1$.
– Delete $T[1]$, then return $\mathsf{lapcs}(S, T[2, t], k-1) + 1$.
– Delete $T[b]$, match $S[1, i] \sim T[1, \overline{b}, j]$, then return $\mathsf{lapcs}(S[i+1, s], T[j+1, t], k - 1) + 1$.

Case 2.5.5 $S[1] = T[1]$, $i > 0$ and $j > 0$, $S[i] = T[j]$, $\exists a : S[i+1, \overline{a}, s] = T[j+1, t]$.

– Delete $S[1]$, then return $\mathsf{lapcs}(S[2, s], T, k-1) + 1$.
– Delete $T[1]$, then return $\mathsf{lapcs}(S, T[2, t], k-1) + 1$.
– Delete $S[a]$, match $S[1] \sim T[1]$ and $S[i, \overline{a}, s] \sim T[j, t]$, then return $\mathsf{lapcs}(S[2, i - 1], T[2, j-1], k-1) + 1$.

Case 2.5.6 $S[1] = T[1]$, $i > 0$ and $j > 0$, $S[i] = T[j]$, $\exists b : S[i+1, s] = T[j+1, \overline{b}, t]$.

– Delete $S[1]$, then return $\mathsf{lapcs}(S[2, s], T, k-1) + 1$.
– Delete $T[1]$, then return $\mathsf{lapcs}(S, T[2, t], k-1) + 1$.
– Delete $T[b]$, match $S[1] \sim T[1]$ and $S[i, s] \sim T[j, \overline{b}, t]$, then return $\mathsf{lapcs}(S[2, i - 1], T[2, j-1], k-1) + 1$.

Case 2.5.7 $S[1] = T[1]$, $i > 0$ and $j > 0$, $S[i] = T[j]$, $S[2, i-1] \neq T[2, j-1]$, $S[i+1, s] \neq T[j+1, t]$, $\forall a : S[2, \overline{a}, i-1] \neq T[2, j-1]$, $\forall a : S[i+1, \overline{a}, s] \neq T[j+1, t]$, $\forall b : S[2, i-1] \neq T[2, \overline{b}, j-1]$, $\forall b : S[i+1, s] \neq T[j+1, \overline{b}, t]$.

– Delete $S[1]$, then return $\mathsf{lapcs}(S[2, s], T, k-1) + 1$.
– Delete $T[1]$, then return $\mathsf{lapcs}(S, T[2, t], k-1) + 1$.
– Delete $S[i]$ and $T[j]$, match $S[1] \sim T[1]$, then return $\mathsf{lapcs}(S[2, \overline{i}, s], T[2, \overline{j}, t], k - 2) + 2$.
– Match $S[1] \sim T[1]$ and $S[i] \sim T[j]$, compute $k' = \mathsf{lapcs}(S[2, i-1], T[2, j-1], k-2) + \mathsf{lapcs}(S[i+1, s], T[j+1, t], k-2)$,, then return k' if $k' \leq k$, or ∞ if $k' > k$.

The correctness of the algorithm is self-evident for the cases from 1 to 2.5.2. To justify the four cases from 2.5.3 and 2.5.6, we have the following easy lemma.

Lemma 2. *For each case from 2.5.3 to 2.5.6, if the condition of the case is met, then there is an optimal solution that corresponds to one of the three branches for that case.*

Finally, the condition for case 2.5.7 ensures that at least two deletions are necessary in each of the two subproblems for $(S[2, i-1], T[2, j-1])$ and $(S[i+1, s], T[j+1, t])$. Thus in the last branch of this case, it is sufficient to set the third parameter to $k-2$ in the two recursions. In terms of time complexity, the seven cases 2.2, 2.3, and 2.5.3–2.5.7 are the worst cases. The six cases 2.2, 2.3, and 2.5.3–2.5.4 correspond to the characteristic polynomial equation $1 = x^{-1} + x^{-1} + x^{-1}$; the last case 2.5.7 corresponds to the characteristic polynomial equation $1 = x^{-1}+x^{-1}+x^{-2}+(x^{-2}+x^{-2})$. Both equations have a unique positive real root $x_0 = 3$.

4 LAPCS(STEM, STEM) for constant alphabet size

The LAPCS(STEM, STEM) problem turns out to be of particular interest for RNA practical issues [7]. This problem has been shown to be **NP**-complete for arbitrarily large alphabets [8]. This section is devoted to investigating the LAPCS(STEM, STEM) problem for constant alphabet size. We first make the easy observation that the LAPCS(STEM, STEM) problem for an alphabet of size 1 admits a polynomial-time exact algorithm by dynamic programming. Unfortunately, this approach cannot be pushed too far. Indeed, we now show that the constant alphabet size assumption is not enough to gain tractability for the LAPCS(STEM,STEM) problem.

Theorem 2. *The* LAPCS(STEM,STEM) *problem is* **NP***-complete for constant alphabet size.*

To prove hardness, we propose a reduction from the **NP**-complete 3-SAT problem [6] which is defined as follows: Given a collection $C_q = \{c_1, c_2, \ldots, c_q\}$ of q clauses, where each clause is the disjunction of 3 literals on a finite set of n boolean variables $V_n = \{x_1, x_2, \ldots, x_n\}$, determine whether there exists a truth assignment to the variables so that each clause has at least one true literal. Let (C_q, V_n) be an arbitrary instance of the 3-SAT problem. For convenience, let L_i^j denote the $j-$th literal of the $i-$th clause (*i.e.* c_i) of C_q. In the following, given a sequence S over an alphabet Σ, let $\circ\mathtt{cc}(i, c, S)$ denote the $i-$th occurrence of the letter c in S. We build two arc-annotated sequences (S_1, P_1) and (S_2, P_2) as follows. An illustration of a full example is given in figures 3 and 4, where $n = 4$ and $q = 3$. For readability reasons, the arc-annotated sequences resulting from the construction have been split into several parts and a schematic overview of the overall placement of each part is provided.

Let S_1 and S_2 be the two sequences defined as follows:

$$S_1 = C_q^1\ S\ C_{q-1}^1 \ldots C_2^1\ S\ C_1^1\ S\ S_M^1\ S\ P_1^1\ S\ P_2^1 \ldots P_{q-1}^1\ S\ P_q^1$$
$$S_2 = C_q^2\ S\ C_{q-1}^2 \ldots C_2^2\ S\ C_1^2\ S\ S_M^2\ S\ P_1^2\ S\ P_2^2 \ldots P_{q-1}^2\ S\ P_q^2$$

where, for all $1 \leq i \leq q$ and $1 \leq k \leq n$,

- $S = 2^\beta$
- $C_i^1 = 9^\delta\ 6^\gamma\ 8^\delta\ 6^\gamma\ X_1^1\ X_2^1 \dots X_n^1\ 6^\gamma\ 8^\delta\ 6^\gamma\ 7^\delta$ with $X_k^1 = 0\ s_j\ 1\ 2^\alpha$ if $x_k = L_i^j$ or $\overline{x_k} = L_i^j$, with $s_1 = 3$, $s_2 = 4$ and $s_3 = 5$; $X_k^1 = 0\ 1\ 2^\alpha$ otherwise;
- $P_i^1 = 6^\gamma\ 6^\gamma\ 9^\delta\ X_n^1 \dots X_{\frac{n}{2}+1}^1\ 8^\delta\ X_{\frac{n}{2}}^1 \dots X_1^1\ 7^\delta\ 6^\gamma\ 6^\gamma$ s.t. $X_k^1 = 1\ 0\ 2^\alpha$;
- $C_i^2 = X_1^2 \dots X_n^2\ 9^\delta\ 6^\gamma\ X_1^2 \dots X_{\frac{n}{2}}^2\ 8^\delta\ X_{\frac{n}{2}+1}^2 \dots X_n^2\ 6^\gamma\ 7^\delta\ X_1^2 \dots X_n^2$ s.t. $\forall 1 \leq j \leq 3$, $\mathtt{occ}(j, X_k^2, C_i^2) = 1\ 0\ s_j\ 2^\alpha$ (resp. $s_j\ 1\ 0\ 2^\alpha$) if $x_k = L_i^j$ (resp. $\overline{x_k} = L_i^j$), with $s_1 = 3$, $s_2 = 4$ and $s_3 = 5$; $\mathtt{occ}(j, X_k^2, C_i^2) = 1\ 0\ 2^\alpha$ otherwise;
- $P_i^2 = (0\ 1\ 2^\alpha)^n\ 7^\delta\ 6^\gamma\ (0\ 1\ 2^\alpha)^{\frac{n}{2}}\ 8^\delta\ (0\ 1\ 2^\alpha)^{\frac{n}{2}}\ 6^\gamma\ 9^\delta\ (0\ 1\ 2^\alpha)^n$.

Moreover, let $S_M^1 = (0\ 1\ 2^\alpha)^n$ and $S_M^2 = (1\ 0\ 2^\alpha)^n$. Notice that, by construction, there is only one occurrence of each $\{3, 4, 5\}$ in C_i^2. Moreover, let $\alpha = 2n + 1$, $\beta = |S_M^1| + \sum_{1 \leq i \leq q}(|C_i^1| + |P_i^1|)$, $\delta = \alpha(n+1)$ and $\gamma = 5\delta + 4$. Let us now define P_1 and P_2. For all $1 \leq i \leq q-1$, (1) add an arc in P_1 between $\mathtt{occ}(k, 0, C_i^1)$ (resp. $\mathtt{occ}(k, 1, C_i^1)$) and $\mathtt{occ}(n-k+1, 0, P_{i+1}^1)$ (resp. $\mathtt{occ}(n-k+1, 1, P_{i+1}^1)$), $\forall 1 \leq k \leq n$ (see Fig. 3.d and 4.b); (2) add an arc in P_2 between $\mathtt{occ}(j * k, 0, C_i^2)$ (resp. $\mathtt{occ}(j * k, 1, C_i^2)$) and $\mathtt{occ}(3n - jk + 1, 0, P_i^2)$ (resp. $\mathtt{occ}(3n - jk + 1, 1, P_i^2)$), $\forall 1 \leq j \leq 3, 1 \leq k \leq n$ (see Fig. 3.c, 4.a and 4.c); (3) add an arc in P_2 between $\mathtt{occ}(k, j, C_i^2)$ and $\mathtt{occ}(d + 1 - k, j, P_i^2)$, $\forall j \in \{7, 8, 9\}$ and $1 \leq k \leq \delta$ (see Fig. 3.c, 4.a and 4.c). Clearly, this construction can be achieved in polynomial-time, and yields to sequences (S_1, P_1) and (S_2, P_2) that are both of type STEM. We now give an intuitive description of the different elements of this construction. Each clause $c_i \in C_q$ is represented by a pair (C_i^1, C_i^2) of sequences. The sequence C_i^2 is composed of three subsequences representing a selection mechanism of one of the three literals of c_i. The pair (S_M^1, S_M^2) of sequences is a control mechanism that will guarantee that a variable x_k cannot be true and false simultaneously. Finally, for each clause $c_i \in C_q$, the pair (P_i^1, P_i^2) of sequences is a propagation mechanism which aim is to propagate the selection of the assignment (*i.e.* true or false) of any literal x_k all over C_q. Notice that all the previous intuitive notions will be detailed and clarified afterwards. In the sequel, we will refer to any such construction as a *snail-construction*. In order to complete the instance of the LAPCS(STEM, STEM) problem, we set $k' = |S_1| - \varepsilon$ with $\varepsilon = q(2(n + 2\delta + 2\gamma + 1)) + n$. (the desired length of the solution). Let (S_1, P_1) and (S_2, P_2) denote the arc-annotated sequences obtained by a snail-construction. We will denote S_d the set of symbols deleted in a solution of the LAPCS problem on (S_1, P_1) and (S_2, P_2) (*i.e.* the symbols that do not belong to the common subsequence). We need some technical lemmas (proofs deferred to Appendix)

Lemma 3. *Any optimal solution of the* LAPCS(STEM, STEM) *problem on* (S_1, P_1) *and* (S_2, P_2) *is of length* $|S_1| - \varepsilon$.

Lemma 4. *In any optimal solution of the* LAPCS(STEM, STEM) *problem on* (S_1, P_1) *and* (S_2, P_2)*, if* $\mathtt{occ}(k, 1, S_M^1)$ *(resp.* $\mathtt{occ}(k, 0, S_M^1)$*) for a given* $1 \leq k \leq n$ *is deleted then,* $\forall 1 \leq j \leq q$, $\mathtt{occ}(k, 1, C_j^1)$ *(resp.* $\mathtt{occ}(k, 0, C_j^1)$*) is deleted.*

The following lemma proves Theorem 2.

Theorem 3. *Given an instance of the problem 3SAT with* n *variables and* q *clauses, there exists a satisfying truth assignment if and only if the* LAPCS(STEM, STEM) *problem for* (S_1, P_1) *and* (S_2, P_2) *is of length* $k' = |S_1| - \varepsilon$.

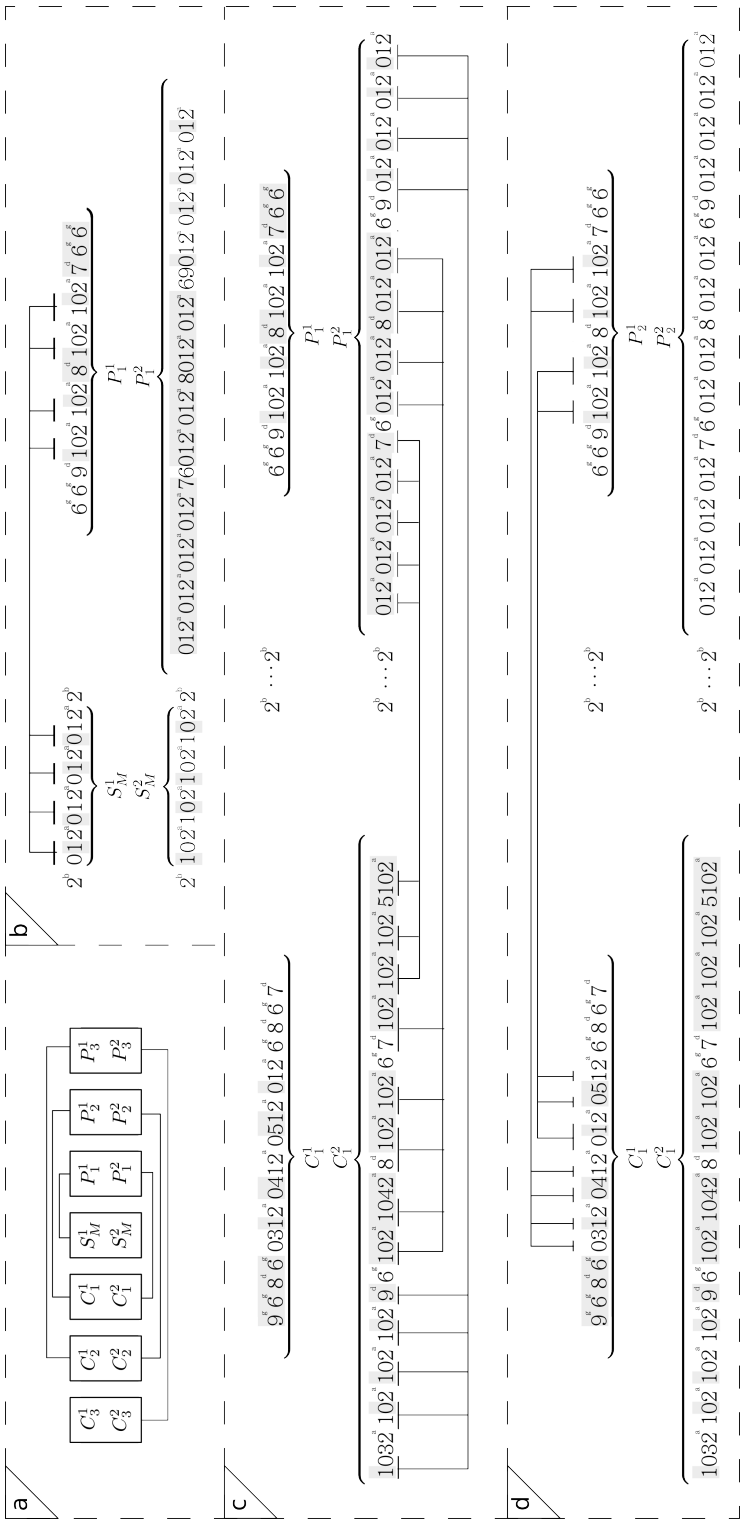

Fig. 3. Considering $C_q = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$. For readability, all the arcs have not been drawn, consecutive arcs are representing by a unique arc with lines for endpoints. Symbols over a grey background may be deleted to obtain an optimal LAPCS. a) A schematic view of the overall arrangement of the components of the two a.a. sequences. b) Description of S_M^1, S_M^2, P_1^1, P_1^2 and the corresponding arcs in P_1. c) Description of C_1^1, C_1^2, P_1^1, P_1^2 and the corresponding arcs in P_2. d) Description of C_1^1, C_1^2, P_2^1, P_2^2 and the corresponding arcs in P_1.

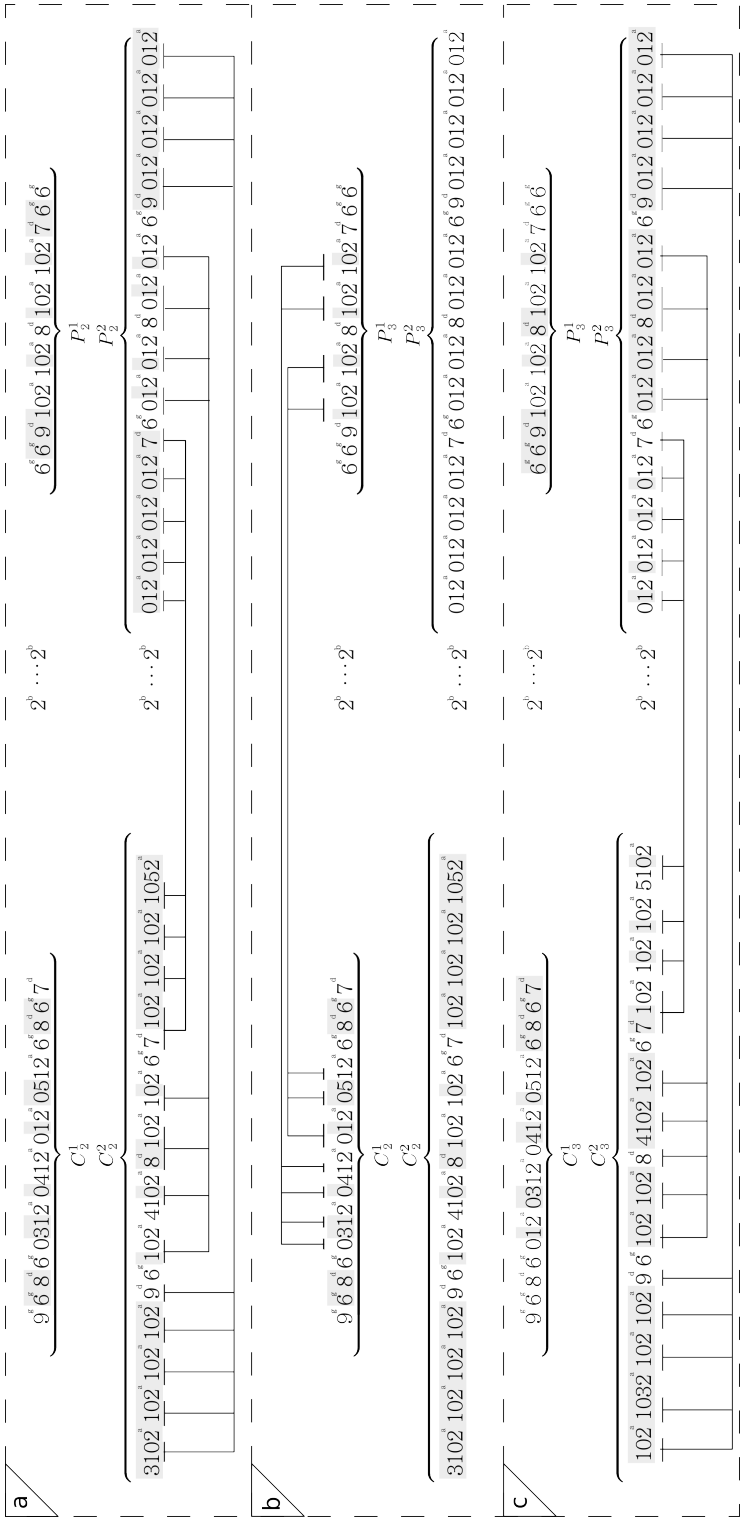

Fig. 4. Considering $C_q = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$. For readability all the arcs have not been drawn, consecutive arcs are representing by a unique arc with lines for endpoints. Symbols over a grey background may be deleted to obtain an optimal LAPCS. a) Description of C_2^1, C_2^2, P_2^1, P_2^2 and the corresponding arcs in P_2. c) Description of C_2^1, C_2^2, P_3^1, P_3^2 and the corresponding arcs in P_1. d) Description of C_3^1, C_3^2, P_3^1, P_3^2 and the corresponding arcs in P_2.

Proof. ($\Rightarrow$) An optimal solution for $C_q = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$ – *i.e.* $x_1 = x_3 = true$ and $x_2 = x_4 = false$ – is illustrated in figures 3 and 4 where any symbol over a grey background have to be deleted. Suppose we have a solution for our 3-SAT instance, that is an assignment of each variable of V_n satisfying each clause of C_q. Let us first list all the symbols to delete in S_1. For all $1 \leq k \leq n$, if $x_k = false$ then delete, $\forall 1 \leq j \leq q$, $\{\mathtt{occ}(k, 0, C_j^1), \mathtt{occ}(k, 1, P_j^1)\}$ and $\mathtt{occ}(k, 0, S_M^1)$; otherwise delete, $\forall 1 \leq j \leq q$, $\{\mathtt{occ}(k, 1, C_j^1), \mathtt{occ}(k, 0, P_j^1)\}$ and $\mathtt{occ}(k, 1, S_M^1)$.

For each L_i^j satisfying c_i with the biggest index j with $1 \leq i \leq q$,

if (1) $j = 1$ then from C_i^1, delete all the symbols 9, the two first substrings of γ symbols 6, the first substring of δ symbols 8, symbols 4 and 5. Moreover, from P_i^1 delete all the symbols 7 and 8, the two last substrings of γ symbols 6 (cf Fig. 3.c);

if (2) $j = 2$ then from C_i^1, delete all the symbols 8, the first and the last substrings of γ symbols 6, symbols 3 and 5. Moreover, from P_i^1 delete all the symbols 7 and 9, the first and the last substrings of γ symbols 6 (cf Fig. 4.a);

if (3) $j = 3$ then from C_i^1, delete all the symbols 7, the two last substrings of γ symbols 6, the last substring of δ symbols 8, symbols 3 and 4. Moreover, from P_i^1 delete all the symbols 8 and 9, the two first substrings of γ symbols 6 (cf Fig. 4.c);

Let us now list all the symbols in S_2 to be deleted. For all $1 \leq k \leq n$, if $x_k = false$ then delete $\mathtt{occ}(k, 0, S_M^2)$; otherwise delete $\mathtt{occ}(k, 1, S_M^2)$. For each L_i^j satisfying c_i with the biggest index j with $1 \leq i \leq q$,

if (1) $j = 1$ then, in C_i^2, delete all the symbols not in $\{6, 7, 8\}$ appearing after $\mathtt{occ}(1, 9, C_i^2)$ (included). Moreover, if $x_k = false$ with $1 \leq k \leq n$ then delete, $\mathtt{occ}(k, 0, C_i^2)$, otherwise delete $\mathtt{occ}(k, 1, C_i^2)$ (cf Fig. 3.c). Moreover, in P_i^2, delete all the symbols not in $\{6, 9\}$ appearing before $\mathtt{occ}(1, 9, P_i^2)$. Moreover, if $x_k = false$ with $1 \leq k \leq n$ then delete, $\mathtt{occ}(3n-k+1, 0, P_i^2)$, otherwise delete $\mathtt{occ}(3n-k+1, 1, P_i^2)$ (cf Fig. 3.c);

if (2) $j = 2$ then, in C_i^2, delete all the symbols 8 and all the symbols appearing before $\mathtt{occ}(1, 9, C_i^2)$ (excluded) or after $\mathtt{occ}(\delta, 7, C_i^2)$ (excluded). Moreover, if $x_k = false$ with $1 \leq k \leq n$ then delete, $\mathtt{occ}(n + k, 0, C_i^2)$, otherwise delete $\mathtt{occ}(n + k, 1, C_i^2)$ (cf Fig. 4.a). Moreover, in P_i^2, delete all the symbols appearing before $\mathtt{occ}(1, 6, P_i^2)$ (excluded) or after $\mathtt{occ}(2\gamma, 6, P_i^2)$ (excluded). Moreover, if $x_k = false$ with $1 \leq k \leq n$ then delete, $\mathtt{occ}(2n - k + 1, 0, P_i^2)$, otherwise delete $\mathtt{occ}(2n - k + 1, 1, P_i^2)$ (cf Fig. 4.a);

if (3) $j = 3$ then, in C_i^2, delete all the symbols not in $\{6, 8, 9\}$ appearing before $\mathtt{occ}(\delta, 7, C_i^2)$ (included). Moreover, if $x_k = false$ with $1 \leq k \leq n$ then delete, $\mathtt{occ}(2n + k, 0, C_i^2)$, otherwise delete $\mathtt{occ}(2n + k, 1, C_i^2)$ (cf Fig. 4.c). Moreover, in P_i^2, delete all the symbols not in $\{6, 7\}$ appearing after $\mathtt{occ}(1, 7, P_i^2)$. Moreover, if $x_k = false$ with $1 \leq k \leq n$ then delete, $\mathtt{occ}(n - k + 1, 0, P_i^2)$, otherwise delete $\mathtt{occ}(n - k + 1, 1, P_i^2)$ (cf Fig. 4.c);

By construction, the natural order of the symbols of S_1 and S_2 allows the corresponding set of undeleted symbols to be conserved in a common arc-preserving common subsequence between (S_1, P_1) and (S_2, P_2). Let us now prove that the length of this last is k'. One can easily check that in this solution, in S_1, n symbols have been deleted from S_M^1 and $\forall 1 \leq i \leq q$, $2\delta+2\gamma+n+2$ symbols from C_i^1 and $2\delta+2\gamma+n$ symbols from P_i^1 have been deleted. Thus, the length of the solution is $|S_1| - [q(2(n + 2\delta + 2\gamma + 1)) + n]$.

($\Leftarrow$) Suppose we have an optimal solution – *i.e.* a set of symbols S_d to delete – for LAPCS of (S_1, P_1) and (S_2, P_2). Let us define the truth assignment of V_n s.t., $\forall 1 \leq i \leq q$, if in C_i^1 symbol 3 is not deleted, then the first literal of clause c_i (*i.e.* L_i^1) is true; if in C_i^1 symbol 4 is not deleted, then the second literal of clause c_i (*i.e.* L_i^2) is true; if in C_i^1 symbol 5 is not deleted, then the third literal of clause c_i (*i.e.* L_i^3) is true;. Let us prove that it is a solution for our 3-SAT instance. By construction, if $L_i^j = x_k$ (resp. $\overline{x_k}$) then in C_i^1, symbol $2 + j$ (*i.e.* 3, 4 or 5) appears between $\mathtt{occ}(k, 0, C_i^1)$ and $\mathtt{occ}(k, 1, C_i^1)$ whereas in C_i^2 it appears after $\mathtt{occ}(k, 1, C_i^2)$ (resp. before $\mathtt{occ}(k, 0, C_i^2)$). Thus, if symbol $2 + j$ (*i.e.* 3, 4 or 5) in C_i^1 is not deleted then $\mathtt{occ}(k, 1, C_i^1)$ (resp. $\mathtt{occ}(k, 0, C_i^1)$) in C_i^1 is deleted if $L_i^j = x_k$ (resp. $\overline{x_k}$). Consequently, according to the proof of Lemma 4, if symbol $2+j$ (*i.e.* 3, 4 or 5) in C_i^1 is not deleted then $\mathtt{occ}(k, 1, C_{i'}^1)$ (resp. $\mathtt{occ}(k, 0, C_{i'}^1)$) in all $C_{i'}^1$, with $1 \leq i' \leq q$ is deleted if $L_i^j = x_k$ (resp. $\overline{x_k}$). Therefore, we can ensure that one cannot obtain L_i^j and $L_{i'}^{j'}$ being true whereas $L_i^j = \overline{L_{i'}^{j'}}$ (that is a variable cannot be simultaneously true and false). By Lemma 3, we can ensure that for any $1 \leq i \leq q$ exactly one of $\{3, 4, 5\}$ is conserved in C_i^1. □

References

1. J. Alber, J. Gramm, J. Guo, and R. Niedermeier. Computing the similarity of two sequences with nested arc annotations. *Theoretical Computer Science*, 312(2-3):337–358, 2004.
2. G. Blin, M. Crochemore, and S. Vialette. *Algorithms in Computational Molecular Biology: Techniques, Approaches and Applications*, chapter Algorithmic Aspects of Arc-Annotated Sequences. Wiley, 2010. To appear.
3. G. Blin, A. Denise, S. Dulucq, C. Herrbach, and H. Touzet. Alignment of RNA structures. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2008. To appear.
4. P. Evans. *Algorithms and Complexity for Annotated Sequences Analysis*. PhD thesis, University of Victoria, 1999.
5. Patricia A. Evans. Finding common subsequences with arcs and pseudoknots. In M. Crochemore and M. Paterson, editors, *Proc. CPM 99*, volume 1645 of *LNCS*, pages 270–280. Springer, 1999.
6. M.R. Garey and D.S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness*. W.H. Freeman, San Francisco, 1979.
7. V. Guignon, C. Chauve, and S. Hamel. An edit distance between rna stem-loops. In Mariano P. Consens and Gonzalo Navarro, editors, *Proc. of SPIRE 2005*, volume 3772 of *LNCS*, pages 335–347, 2005.
8. S. Hamel, G. Blin, and S. Vialette. Comparing RNA structures with biologically relevant operations cannot be done without strong combinatorial restrictions. In S. Fujita and S. Rahman, editors, *Proc. of WALCOM'10*, LNCS, 2010.
9. T. Jiang, G. Lin, B. Ma, and K. Zhang. A general edit distance between RNA structures. *Journal of Computational Biology*, 9(2):371–388, 2002.
10. T. Jiang, G.-H. Lin, B. Ma, and K. Zhang. The longest common subsequence problem for arc-annotated sequences. In R. Giancarlo and D. Sankoff, editors, *Proc. CPM 2000*, volume 1848 of *LNCS*, pages 154–165. Springer, 2000.
11. G. Lin, Z.-Z. Chen, T. jiang, and J. Wen. The longest common subsequence problem for sequences with nested arc annotations. *Journal of Computer and System Sciences*, 65:465–480, 2002.
12. D. Shasha and K. Zhang. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989.

Appendix

Proof (Of Lemma 1). Consider any independent set S in the graph G. Clearly, S includes at most two vertices from each copy A_i of the graph A. Moreover, if S includes exactly two vertices from A_i, then the two vertices must be v_1 and v_4, and consequently S includes at most one vertex from A_{i-1} and at most one vertex from A_{i+1}. Recall that $n = 4t$. Thus the maximum cardinality k^* of an independent set in the graph G is $\left\lfloor \frac{2+1}{2}\, t \right\rfloor = \left\lfloor \frac{3}{8}\, n \right\rfloor$. □

Proof (Of Lemma 2). We first consider case 2.5.3, where $S[1] = T[1]$, $i > 0$ and $j > 0$, $S[i] = T[j]$, and $\exists a : S[2, \overline{a}, i-1] = T[2, j-1]$. Then $i - 1 = j \geq 2$ and $S[1, \overline{a}, i] = T[1, j]$.

Consider any optimal solution in which $S[1]$ and $T[1]$ are not deleted, that is, $S[1] \sim T[1]$ is a match. Then the jth letter of this optimal solution matches $S[i'] \sim T[j']$ for some $i' \geq j$ and $j' \geq j$. If $S[i]$ is not deleted, then the arc $S[1]S[i]$ is matched to the arc $T[1]T[j]$, and consequently at least one letter in $S[2, i-1]$ must be deleted because $S[2, i-1]$ has one more letter than $T[2, j-1]$. In any case, at least one letter in $S[1, i]$ must be deleted, thus $i' > j$. Since $j = i - 1$, it follows that $i' \geq i$. Since we also have $j' \geq j$, it is safe to replace the first j letters of this optimal solution by the j letters in $S[1, \overline{a}, i] = T[1, j]$, which results in an optimal solution that matches $S[1, \overline{a}, i] \sim T[1, j]$ and deletes $S[a]$. This proves the lemma for case 2.5.3. A similar replacement argument proves the lemma for the other three cases. □

Technical lemma needed to prove Lemma 4.

Lemma 5. *In any optimal solution of the* LAPCS(STEM, STEM) *problem on* (S_1, P_1) *and* (S_2, P_2)*, at least one symbol incident to any arc has to be deleted. Moreover, all the substrings of* β *symbols* 2 *– denoted as* S *in the construction – will not be deleted.*

Proof. By contradiction, let us suppose that there exist an optimal solution where one complete substring S has been deleted. Therefore, since S_1 is, by construction, smaller than S_2 the length of this optimal solution is at most $|S_1| - \beta$. Then, in order for this solution to be optimal, one should have $\varepsilon \geq \beta$; a contradiction since $\beta = |S_M^1| + \sum_{1 \leq i \leq q}(|C_i^1| + |P_i^1|) = (2+\alpha)n + q((4\gamma + 4\delta + (2+\alpha)n + 3) + (4\gamma + 3\delta + (2+\alpha)n))$. Thus, in an optimal solution, all the substrings of β symbols 2 should be preserved – contributing $(2q-1)\beta$ to the length of the optimal. This also induces that, $\forall 1 \leq i \leq q$, symbols of C_i^1 will be matched to symbols of C_i^2, symbols of P_i^1 will be matched to symbols of P_i^2 and symbols of S_M^1 will be matched to symbols of S_M^2. □

Technical lemma needed to prove Lemma 3 and Lemma 4.

Lemma 6. *In any optimal solution of the* LAPCS(STEM, STEM) *problem on* (S_1, P_1) *and* (S_2, P_2)*, the alignment of* S_M^1 *and* S_M^2 *contributes* $(\alpha + 1)n$ *to the length of the solution.*

Proof. Indeed, in any optimal solution, all the symbols 2 of S_M^1 should be kept. By contradiction, suppose a full substring of α symbols 2 is not kept. Then the contribution of S_M^1 to this optimal solution is at most $|S_M^1| - \alpha$. Consider now the common subsequence of S_M^1 and S_M^2 where all the symbols 2 are kept. This last is of size $|S_M^1| - 2n$ – which is, by definition, greater than $|S_M^1| - \alpha$; a contradiction. Therefore, in any optimal solution, by construction, for all $1 \leq i \leq n$, either $\circ\mathtt{cc}(i, 0, S_M^1)$ or $\circ\mathtt{cc}(i, 1, S_M^1)$ will be kept in addition to all the symbols 2; leading to a common subsequence of size $(\alpha + 1)n$. □

Technical lemma needed to prove Lemma 3 and Lemma 4.

Lemma 7. *In any optimal solution of the* LAPCS(STEM, STEM) *problem on* (S_1, P_1) *and* (S_2, P_2)*, the alignment of* P_i^1 *and* P_i^2 *contributes* $(2\gamma + \delta + (\alpha + 1)n)$ *to the length of the solution.*

Proof. In any optimal solution, in P_i^1, half of the symbols 6 should be kept. First, note that there are 4γ symbols 6 in P_i^1 and only 2γ such symbols in P_i^2. Thus, in any optimal solution, in P_i^1, at most half of the symbols 6 can be kept. By contradiction to the first assumption, suppose there are, in an optimal solution, less than half of the symbols 6 kept in P_i^1. Then the length of this optimal solution is at most $|P_i^1| - 3\gamma$. Consider now the common subsequence of P_i^1 and P_i^2 where half of the symbols 6 in P_i^1 are kept. This last is of size $|P_i^1| - (2\gamma + 3\delta + (\alpha + 2)n)$ – which is, by definition of γ, greater than $|P_i^1| - 3\gamma$; a contradiction. Therefore, in any optimal solution, since in P_i^1 half of the symbols 6 are kept, by construction, at most one of the substrings 7^δ, 8^δ or 9^δ can be kept. Moreover, in P_i^1, by construction, for all $1 \leq j \leq n$, either $\circ\mathtt{cc}(j, 0, P_i^1)$ or $\circ\mathtt{cc}(j, 1, P_i^1)$ will be kept in addition to all the symbols 2; leading to a common subsequence of size $(2\gamma + \delta + (\alpha + 1)n)$. □

Technical lemma needed to Prove Lemma 3 and Lemma 4.

Lemma 8. *In any optimal solution of the* LAPCS(STEM, STEM) *problem on* (S_1, P_1) *and* (S_2, P_2)*, the alignment of* C_i^1 *and* C_i^2 *contributes* $(2\gamma + 2\delta + (\alpha + 1)n + 1)$ *to the length of the solution.*

Proof. By a similar proof as in Lemma 7, in any optimal solution, in C_i^1 half of the symbols 6 should be kept. Moreover, among all the 4δ symbols 7, 8 and 9, only 2δ may be kept in an optimal solution. Indeed, first, note that there are 2δ symbols 8 in C_i^2 and only δ such symbols in C_i^1. Thus, in any optimal solution, in C_i^1, at most δ of the symbols 8 can be kept. Then, note that by Lemma 7, at most one of the substrings 7^δ, 8^δ

or 9^δ can be kept in an optimal solution and in P_2, for all $1 \leq j \leq \delta$ and $k \in \{7, 8, 9\}$, there is an arc between $\mathtt{occ}(j, k, C_i^2)$ and $\mathtt{occ}(\delta - j + 1, k, P_i^2)$. Moreover, considering all the combinations of alignment of the symbols 6, at most one of $\{3, 4, 5\}$ may be kept. Finally, by construction, for all $1 \leq j \leq n$, either $\mathtt{occ}(j, 0, C_i^1)$ or $\mathtt{occ}(j, 1, C_i^1)$ will be kept in addition to all the symbols 2; leading to a common subsequence of size $(2\gamma + 2\delta + (\alpha + 1)n + 1)$. □

Proof (Of Lemma 3). By construction, S_1 is of length $(2q-1)\beta + q(4\gamma + 4\delta + 3 + (2 + \alpha)n) + (2 + \alpha)n + q(4\gamma + 3\delta + (2 + \alpha)n)$. According to Lemmas 6, 7, 8, any optimal solution is of size $(2q-1)\beta + q(2\gamma + 2\delta + 1 + (1+\alpha)n) + (1+\alpha)n + q(2\gamma + \delta + (1+\alpha)n)$. One can easily verify that any optimal solution is, thus, of size $|S_1| - \varepsilon$. □

Proof (Of Lemma 4). By Lemma 6, for every $1 \leq k \leq n$ only one of $\{\mathtt{occ}(k, 0, S_M^1), \mathtt{occ}(k, 1, S_M^1)\}$ may be conserved between S_M^1 and S_M^2. By Lemma 5, at least one symbol incident to any arc is deleted. Therefore, $\forall 1 \leq k \leq n$ only one of $\{\mathtt{occ}(n - k + 1, 0, SP_i^1), \mathtt{occ}(n - k + 1, 1, P_i^1)\}$ may be conserved.

Let us suppose that for a given $1 \leq k \leq n$, $\mathtt{occ}(k, 1, S_M^1)$ is deleted. Then $\mathtt{occ}(n - k + 1, 0, P_1^1)$ is deleted whereas $\mathtt{occ}(n - k + 1, 1, P_1^1)$ and $\mathtt{occ}(k, 0, S_M^1)$ are conserved.

By construction, since according to the proof of Lemma 7, in P_1^1 half of the symbols 6 and one of the substrings 7^δ, 8^δ or 9^δ are kept, either (1) the two first substrings of γ symbols 6 and the substring of symbols 9, or (2) one of the two first and one of the two last substrings of γ symbols 6 and the substring of symbols 8, or (3) the two last substrings of γ symbols 6 and the substring of symbols 7 are conserved.

Let us first consider that the two first substrings of γ symbols 6 and the substring of symbols 9 are conserved (cf. Fig. 3.c). Since, in P_2, $\forall 1 \leq k \leq \delta$ there is an arc between $\mathtt{occ}(k, 9, C_1^2)$ and $\mathtt{occ}(\delta - k + 1, 9, P_1^2)$, according to Lemma 8, a substring of δ symbols 7 and a substring of δ symbols 8 in C_1^1 have to be conserved. Then one can check that the only solution is to conserve the second substring of δ symbols 8 and the two last substrings of γ symbols 6 of C_1^1 since otherwise one would not be able to conserve all the symbols 2 of C_1^1 (which is required according to Lemma 8). Consequently, the only solution is to conserve, the symbols 2 appearing before $\mathtt{occ}(1, 6, C_1^2)$ (resp. after $\mathtt{occ}(\gamma + 1, 6, P_1^2)$) in C_1^2 (resp. P_1^2). Since by construction, $\forall 1 \leq k \leq n$, there is an arc between $\mathtt{occ}(k, 1, C_1^2)$ (resp. $\mathtt{occ}(k, 0, C_1^2)$) and $\mathtt{occ}(3n - k + 1, 1, P_1^2)$ (resp. $\mathtt{occ}(3n - k + 1, 0, P_1^2)$), in order for $\mathtt{occ}(k, 1, P_1^1)$ to be conserved, one has to conserved $\mathtt{occ}(3n - k + 1, 1, P_1^2)$. Thus, by Lemma 5, $\mathtt{occ}(k, 1, C_1^2)$ has to be deleted and, according to the proof of Lemma 8, $\mathtt{occ}(k, 0, C_1^1)$ has to be conserved.

Let us now consider that one of the two first and one of the two last substrings of γ symbols 6 and the substring of symbols 8 are conserved (cf Fig. 4.a). By a similar reasoning, one can check that the only solution is to conserve, the symbols 2 appearing between $\mathtt{occ}(1, 6, C_1^2)$ and $\mathtt{occ}(\gamma + 1, 6, C_1^2)$ (resp. between $\mathtt{occ}(1, 6, P_1^2)$ and $\mathtt{occ}(\gamma + 1, 6, P_1^2)$) in C_1^2 (resp. P_1^2). Since by construction, $\forall 1 \leq k \leq n$, there is an arc between $\mathtt{occ}(n + k, 1, C_1^2)$ (resp. $\mathtt{occ}(n + k, 0, C_1^2)$) and $\mathtt{occ}(2n - k + 1, 1, P_1^2)$ (resp. $\mathtt{occ}(2n - k + 1, 0, P_1^2)$), in order for $\mathtt{occ}(k, 1, P_1^1)$ to be conserved, one has to

conserved $\mathtt{occ}(2n-k+1,1,P_1^2)$. Thus, by Lemma 5, $\mathtt{occ}(n+k,1,C_1^2)$ has to be deleted and, according to the proof of Lemma 8, $\mathtt{occ}(k,0,C_1^1)$ has to be conserved.

Finally, let us consider that the two last substrings of γ symbols 6 and the substring of symbols 7 are conserved (cf Fig. 4.c). Once again, by a similar reasoning, one can check that the only solution is to conserve, the symbols 2 appearing after $\mathtt{occ}(\gamma+1,6,C_1^2)$ (resp. before $\mathtt{occ}(1,6,P_1^2)$) in C_1^2 (resp. P_1^2). Since by construction, $\forall 1 \leq k \leq n$, there is an arc between $\mathtt{occ}(2n+k,1,C_1^2)$ (resp. $\mathtt{occ}(2n+k,0,C_1^2)$) and $\mathtt{occ}(n-k+1,1,P_1^2)$ (resp. $\mathtt{occ}(n-k+1,0,P_1^2)$), in order for $\mathtt{occ}(k,1,P_1^1)$ to be conserved, one has to conserved $\mathtt{occ}(n-k+1,1,P_1^2)$. Thus, by Lemma 5, $\mathtt{occ}(2n+k,1,C_1^2)$ has to be deleted and, according to the proof of Lemma 8, $\mathtt{occ}(k,0,C_1^1)$ has to be conserved.

With a similar reasoning, by recurrence, since, $\forall 1 \leq i \leq q, 1 \leq k \leq n$, there is an arc in P_1 between $\mathtt{occ}(k,0,C_i^1)$ (resp. $\mathtt{occ}(k,1,C_i^1)$) and $\mathtt{occ}(k,0,P_{i+1}^1)$ (resp. $\mathtt{occ}(k,1,P_{i+1}^1)$), if $\mathtt{occ}(k,0,C_i^1)$ is conserved then $\mathtt{occ}(k,0,P_{i+1}^1)$ is deleted. And therefore, with similar arguments, $\mathtt{occ}(k,0,C_{i+1}^1)$ is conserved. Once more, it is easy to see that this result still holds if $\mathtt{occ}(k,0,C_i^1)$ is conserved. □