ACADEMIC PRESS

Journal of Computer and System Sciences 65 (2002) 465–480

JOURNAL OF COMPUTER AND SYSTEM SCIENCES

www.academicpress.com

The longest common subsequence problem for sequences with nested arc annotations☆

Guohui Lin,[a,1] Zhi-Zhong Chen,[b,2] Tao Jiang,[c,*,3] and Jianjun Wen[c,4]

[a] *Department of Computing Sciences, University of Alberta, Edmonton, Alberta, Canada T6G 2E8*
[b] *Department of Mathematical Sciences, Tokyo Denki University, Hatoyama, Saitama 350-0394, Japan*
[c] *Department of Computer Science, University of California, Riverside, CA 92521, USA*

Received 15 February 2001; received in revised form 04 September 2001

Abstract

Arc-annotated sequences are useful in representing the structural information of RNA and protein sequences. The Longest Arc-Preserving Common Subsequence (LAPCS) Problem has been introduced in Evans (Algorithms and complexity for annotated sequence analysis, Ph.D. Thesis, University of Victoria, 1999) as a framework for studying the similarity of arc-annotated sequences. Several algorithmic and complexity results on the LAPCS problem have been presented in Evans (1999) and Jiang et al. (in: Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching (CPM 2000), Lecture Note in Computer Science, Vol. 1848, 2000, pp. 154–165). In this paper, we continue this line of research and present new algorithmic and complexity results on the LAPCS problem restricted to two nested arc-annotated sequences, denoted as LAPCS(nested, nested). The restricted problem is perhaps the most interesting variant of the LAPCS problem and has important applications in the comparison of RNA secondary structures. Particularly, we prove that LAPCS(nested, nested) is NP-hard, which answers an open question in Evans (1999). We then present a polynomial-time approximation scheme for LAPCS(nested, nested) with an additional *c-diagonal* restriction. An interesting special case, Unary LAPCS(nested, nested),

☆An extended abstract of a preliminary version of this paper appears in *Proceedings of the 28th International Colloquium on Automata, Languages and Programming* (ICALP'01), LNCS 2076, pp. 444–455.

*Corresponding author.

E-mail addresses: ghlin@cs.ualberta.ca (G. Lin), chen@r.dendai.ac.jp (Z.-Z. Chen), jiang@cs.ucr.edu (T. Jiang), wjianju@cs.ucr.edu (J. Wen).

[1] Work partially done while the author was a postdoctoral fellow in the Department of Computer Science, University of Waterloo, and the Department of Computing and Software, McMaster University. Supported in part by NSERC Research Grants OGP0046613 and OGP0046506, and a CITO grant.

[2] Supported in part by the Grant-in-Aid for Scientific Research of the Ministry of Education, Science, Sports and Culture of Japan, under Grant No. 12780241. Work done on visiting at Department of Computer Science, University of California, Riverside, CA 92521, USA.

[3] Supported in part by a UCR startup grant and NSF Grants CCR-9988353 and ITR-0085910.

[4] Supported in part by NSF Grant CCR-9988353.

is also investigated, for which we show the NP-hardness and present a better approximation algorithm than the one for general LAPCS(NESTED, NESTED).

Keywords: Sequence annotation; Longest common subsequence; Book embedding; NP-hardness; Approximation algorithm; Polynomial-time approximation scheme

1. Introduction

Given two sequences S and T over some fixed *alphabet* Σ, sequence T is said to be a *subsequence* of S if T can be obtained from S by deleting some letters (also called *bases*) from S. Notice that the order of the remaining letters of S must be preserved. The *length* of a sequence S, denoted by $|S|$, is the number of letters therein. Given two sequences S_1 and S_2 (over some fixed *alphabet* Σ), the classical LONGEST COMMON SUBSEQUENCE (LCS) problem asks for a longest sequence T that is a subsequence of both S_1 and S_2. Suppose $|S_1| = n_1$ and $|S_2| = n_2$, then a longest common subsequence of S_1 and S_2 can be found by dynamic programming in time $O(n_1 n_2)$ [17,23]. For simplicity, we use $S[i]$ to denote the ith letter in sequence S, and $S[i_1, i_2]$ to denote the substring of S consisting of the i_1th letter through the i_2th letter ($1 \leqslant i_1 \leqslant i_2 \leqslant |S|$).

For any sequence S, an *arc annotation* set (or simply an arc set) P of S is a set of *unordered* pairs of positions in S. Each pair $(i_1, i_2) \in P$, where $1 \leqslant i_1 < i_2 \leqslant |S|$, is said to *connect* the two letters at positions i_1 and i_2 and is called an *arc annotation* (or simply, arc) between the two letters. Such a pair (S, P) of sequence and arc annotation set is referred to as an *arc-annotated sequence* [12]. Observe that a (plain) sequence without any arc can be viewed as an arc-annotated sequence with an empty arc set.

Arc-annotated sequences are useful in describing the secondary and tertiary structures of RNA and protein sequences [2,11,12,15,19,22,25]. For example, one may use arcs to represent bonds between nucleotides in an RNA sequence (see Fig. 1) or contact forces between amino acids in a protein sequence. Therefore, the problem of comparing arc-annotated sequences has applications in the structural comparison of RNA and protein sequences and has received much attention in the literature recently [2,12,15,18,19,25]. In this paper, we follow the LCS approach proposed in [12] and study the LONGEST ARC-PRESERVING COMMON SUBSEQUENCE (LAPCS) PROBLEM for arc-annotated sequences.

Given two arc-annotated sequences S_1 and S_2 with arc sets P_1 and P_2, respectively, if $S_1[i] = S_2[j]$ for some pair of integers i and j, we name the pair $\langle i, j \rangle$ a *base match*; and if $S_1[i_1] = S_2[j_1]$, $S_1[i_2] = S_2[j_2]$, $(i_1, i_2) \in P_1$, and $(j_1, j_2) \in P_2$, for some integers $i_1 < i_2$ and $j_1 < j_2$, we name the pair $\langle (i_1, i_2), (j_1, j_2) \rangle$ an *arc match*. A common subsequence T of S_1 and S_2 induces a bijective mapping from a subset of $\{1, 2, \ldots, n_1\}$ to a subset of $\{1, 2, \ldots, n_2\}$, where $n_1 = |S_1|$ and $n_2 = |S_2|$. Let M denote

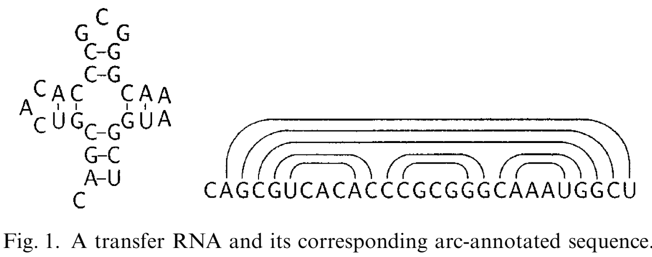


Fig. 1. A transfer RNA and its corresponding arc-annotated sequence.

this mapping and suppose that $M = \{\langle i_\ell, j_\ell \rangle,\ \ell = 1, 2, \ldots, |T|\}$, then we say that T *induces* the base matches $\langle i_\ell, j_\ell \rangle,\ \ell = 1, 2, \ldots, |T|$. The common subsequence T is *arc-preserving* if the arcs induced by the mapping are preserved, i.e. for any $\langle i_{\ell_1}, j_{\ell_1} \rangle,\ \langle i_{\ell_2}, j_{\ell_2} \rangle \in M$:

$$(i_{\ell_1}, i_{\ell_2}) \in P_1 \Leftrightarrow (j_{\ell_1}, j_{\ell_2}) \in P_2;$$

and we say in this case that T induces the arc match $\langle (i_{\ell_1}, j_{\ell_2}), (j_{\ell_1}, j_{\ell_2}) \rangle$ if $(i_{\ell_1}, i_{\ell_2}) \in P_1$ and $(j_{\ell_1}, j_{\ell_2}) \in P_2$ in addition. The LAPCS problem is to find a longest common subsequence of S_1 and S_2 that is arc-preserving (with respect to the given arc sets P_1 and P_2) [12].

It is shown in [12] that the LAPCS problem is NP-hard, if the arc annotation structures are unrestricted. Since in the practice of RNA and protein sequence comparisons arc sets are likely to satisfy some constraints (e.g. bond arcs usually do not cross in the case of RNA, especially transfer RNA, sequences), it is of interest to consider various restrictions on arc structures. The following four natural restrictions on an arc set P have been discussed in the literature [12].

1. No two arcs share an endpoint:

$$\forall (i_1, i_2), (i_3, i_4) \in P, \quad i_1 \neq i_4,\ i_2 \neq i_3, \text{ and } i_1 = i_3 \Leftrightarrow i_2 = i_4.$$

2. No two arcs cross each other:

$$\forall (i_1, i_2), (i_3, i_4) \in P, \quad i_1 \in [i_3, i_4] \Leftrightarrow i_2 \in [i_3, i_4].$$

3. No two arcs nest:

$$\forall (i_1, i_2), (i_3, i_4) \in P, \quad i_1 \leqslant i_3 \Leftrightarrow i_2 \leqslant i_3.$$

4. There are no arcs at all:

$$P = \emptyset.$$

These restrictions are used progressively and inclusively to produce five distinct *levels* of permitted arc structures for sequences in the LAPCS problem:

- UNLIMITED—no restrictions.
- CROSSING—restriction 1.
- NESTED—restrictions 1 and 2.
- CHAIN—restrictions 1, 2 and 3.
- PLAIN—restriction 4.

In the following, we use the notation LAPCS(LEVEL-1, LEVEL-2) [18] to represent the LAPCS problem where the arc structure of sequence S_1 is of level LEVEL-1 and the arc structure of sequence S_2 is of level LEVEL-2. Without loss of generality, we always assume that LEVEL-1 is at the same level of LEVEL-2 or higher than LEVEL-2. Problem LAPCS(UNLIMITED, LEVEL-2) is NP-hard [12] and is not approximable within ratio $n^\varepsilon, \varepsilon \in (0, \frac{1}{4})$ [18], where $n = \max\{n_1, n_2\}$. Problem LAPCS(CROSSING, LEVEL-2) is also NP-hard [12], and is MAX SNP-hard [18], and admits a 2-approximation [18]. If LEVEL-1 is at most NESTED and LEVEL-2 is lower than NESTED, then LAPCS(LEVEL-1, LEVEL-2) is solvable in polynomial time [12,17,18,23]. Prior to this work, the most interesting yet unsolved case is LAPCS(NESTED, NESTED), except that it inherits the 2-approximation algorithm designed for LAPCS(CROSSING, CROSSING) [18].

Notice that problem LAPCS(NESTED, NESTED) effectively models the similarity between two RNA sequences and their secondary structures, and is generally thought of as the most important variant of the LAPCS problem. For example, the arc structure in Fig. 1 is nested. In this paper, we investigate the computational complexity of this problem arm show its NP-hardness, answering an open question in [12]. The hardness result in fact also holds for a rather special case of LAPCS(NESTED, NESTED), denoted by 2-FRAGMENTED LAPCS(NESTED, NESTED), where all base matches induced by an LAPCS are required to have the form $\langle 2i + \frac{1}{2} \pm \frac{1}{2}, 2i + \frac{1}{2} \pm \frac{1}{2} \rangle$. That is, the input sequences are fragmented into pairs of bases and a pair of bases in one sequence are only permitted to match the corresponding pair of bases in the other sequence. In the positive aspect, we present a polynomial-time approximation scheme (PTAS) for a problem slightly more general than 2-FRAGMENTED LAPCS(NESTED, NESTED), called c-DIAGONAL LAPCS(NESTED, NESTED). Here, for any integral constant $c \geqslant 1$, c-DIAGONAL LAPCS(NESTED, NESTED) is the special case of LAPCS(NESTED, NESTED) where base $S_1[i]$ (respectively, $S_2[j]$) is allowed only to match a base in the range $S_2[i-c, i+c]$ (respectively, $S_1[j-c, j+c]$). The c-diagonal restriction has been studied extensively for *sequence alignment* problems in the literature [16,20]. The c-DIAGONAL LAPCS(NESTED, NESTED) problem is relevant in the comparison of conserved RNA sequences where we already have a rough idea about the correspondence between bases in the two sequences. The PTAS is based on an interesting application of the bounded treewidth decomposition technique for planar graphs due to Baker [3].

The rest of the paper is organized as follows. Section 2 proves the hardness results. Section 3 presents a PTAS for c-FRAGMENTED LAPCS(NESTED, NESTED), for any constant $c \geqslant 2$, and then extends it to a PTAS for c-DIAGONAL LAPCS(NESTED, NESTED), for any constant $c \geqslant 1$. Section 3 also presents an efficient exact algorithm for 1-FRAGMENTED LAPCS(CROSSING, CROSSING). Section 4 deals with an interesting special case called UNARY LAPCS(NESTED, NESTED), proves its NP-hardness, and presents a $\frac{4}{3}$-approximation algorithm. Section 5 concludes the paper with some future research topics.

2. Hardness results

In this section, we will prove that problem LAPCS(NESTED, NESTED) is NP-hard. As byproducts, hardness results for several variants are also derived. We begin by reviewing the hardness results of some graph-theoretic problems needed in our proofs.

Lemma 2.1. *The* MAXIMUM INDEPENDENT SET (MIS) *problem restricted to cubic planar graphs is NP-hard* [14]. *On the other hand, it admits a PTAS* [3].

A *book embedding* of a graph consists of an embedding of its vertices along the spine of a book (i.e. a linear ordering of the vertices), and an embedding of its edges onto the pages so that edges embedded on the same page do not cross. (Fig. 3 (upper right) shows an embedding of K_4, the complete graph on 4 vertices, into two pages: the edges above the linear ordering of the vertices are in one page, and the edges below are in the other page.) The objective of the BOOK EMBEDDING PROBLEM is to minimize the number of pages used. The minimum number of pages in which a graph can be embedded is called the *pagenumber* of the graph. Computationally, the BOOK EMBEDDING PROBLEM is hard: it is NP-complete to tell if a graph can be embedded in two pages [10]. The restricted problem of embedding the edges optimally for a fixed vertex ordering is also NP-complete [13]. Nonetheless, we have the following positive results due to Yannakakis.

Lemma 2.2 (Yannakakis [24]). *Graphs with pagenumber* 1 *are exactly the outerplanar graphs. Graphs with pagenumber* 2 *are the subhamiltonian planar graphs*, *i.e. the subgraphs of Hamiltonian planar graphs. There exists an algorithm which embeds a given planar graph in four pages in time linear in the number of edges in the graph. Moreover*, *there exist planar graphs which cannot be embedded in three pages.*

The ultimate goal of this section is to show that LAPCS(NESTED, NESTED) is NP-hard, which answers an open question proposed in [12]. From Lemmas 2.1 and 2.2, we can use the reduction technique in [12,18] to construct a simple reduction showing the NP-hardness of the LAPCS problem for four nested arc-annotated sequences (the definition is a straightforward extension of that of LAPCS(NESTED, NESTED)). Theorem 2.8 below proves the hardness result for two nested arc-annotated sequences. We need the following improvements on Lemmas 2.1 and 2.2 to prove Theorem 2.8.

Lemma 2.3 (Biedl et al. [5]). *The* VERTEX COVER (VC) *problem restricted to cubic triconnected planar graphs is NP-hard.*

Corollary 2.4. *The MIS problem restricted to cubic planar bridgeless connected graphs is NP-hard.*

Lemma 2.5. *Any cubic planar bridgeless graph has a perfect matching* [21], *which can be computed in linear time* [4].

Given a planar graph G and its planar embedding $\mathcal{E}$, any simple cycle C in G partitions the whole embedding plane into two regions. The finite region enclosed by C is called the *interior* of C and denoted by $I(C)$; and the infinite region is called the *exterior* of C and denoted by $E(C)$.

Lemma 2.6. *Cubic planar bridgeless connected graphs are subhamiltonian. Moreover*, *there is a linear-time algorithm that*, *given any cubic planar bridgeless connected graph* G, *finds a Hamiltonian planar supergraph* $H(G)$ *of* G *with maximum degree at most* 5, *and finds a* 2*-page book embedding of* G *such that every vertex has a degree at least one* (*and thus at most two*) *in each page.*

Proof. Given a cubic planar bridgeless connected graph $G = (V, E)$ and its planar embedding $\mathcal{E}$, we construct $H(G)$ in four steps as follows. Firstly, we apply Lemma 2.5 to compute a perfect matching M of G. Let $E' = E \backslash M$ and $G' = (V, E')$. Since every vertex in G' has degree 2, G' is a collection of non-intersecting simple cycles (with respect to the embedding plane). Suppose that G' consists of k cycles. Then $k \leqslant \lfloor \frac{n}{3} \rfloor$ and there is a subset $M' \subset M$ of $k-1$ edges which interconnect these k cycles, that is, graph $G'' = (V, E' \cup M')$ is connected.

Secondly, we orient the cycles in G' as follows. Starting with an arbitrary cycle C_1, orient it in the clockwise direction (i.e. applying the right-hand rule and traversing on the boundary of $I(C_1)$). For each cycle C_2 connected by an edge in M' to C_1, if one of C_1 and C_2 is in the interior of the other, then C_2 is oriented in the same way as C_1, i.e. clockwise; otherwise, it is oriented in the other way, i.e. counterclockwise. By a *Breadth First Search* (BFS) through G'', we can orient all the cycles without conflict. Every edge in E' inherits an orientation from the cycle to which it belongs.

Thirdly, for every edge $(u, v) \in M'$, define the *red predecessor* $rp(u)$ of vertex u, according to the following rules:

(R1) Some cycle C in G' contains both $rp(u)$ and u.
(R2) The edge in M incident to $rp(u)$ lies in the interior of cycle C if and only if edge (u, v) lies in the interior of C.
(R3) Other than $rp(u)$ and u, no vertex on the directed path in C from $rp(u)$ to u satisfies (R2).

Because of the bridgelessness property, $rp(u) \neq u$. Define the *green predecessor* $gp(u)$ of vertex u to be the direct successor of $rp(u)$ in C. Notice that $gp(u)$ may happen to be u. However, clearly, $rp(u) \neq gp(u)$. The red predecessor and the green predecessor of vertex v are similarly defined. It is trivial to notice that for any pair of distinct vertices u and v in a cycle, if they both have red predecessors and green predecessors, then $rp(u) \neq rp(v)$ and $gp(u) \neq gp(v)$.

At the fourth step, for edge $(u, v) \in M'$, connect vertex $rp(u)$ and vertex $rp(v)$ by a *red edge* $(rp(u), rp(v))$ and embed it in the face of $\mathscr{E}$ whose boundary consists of the edge in M incident to u, the (directed) path from $rp(v)$ to u, edge (u, v), the (directed) path from $rp(v)$ to v, and the edge in M incident to vertex v. Similarly, connect vertex $gp(u)$ and vertex $gp(v)$ by a *green edge* $(gp(u), gp(v))$ and embed this edge in the same face (without intersecting the embedded red edge $(rp(u), rp(v))$). See Fig. 2 for an illustration.

Let

$$M_r = \{(rp(u), rp(v)) \mid (u, v) \in M'\},$$

$$M_g = \{(gp(u), gp(v)) \mid (u, v) \in M'\},$$

$$M_d = \{(rp(u), gp(u), rp(v), gp(v)) \mid (u, v) \in M'\},$$

where M_r (or M_g) is a set of artificial red (or green, respectively) edges, and M_d is a subset of E'. We set $H(G) = (V, E \cup M_r \cup M_p)$. Obviously, $(V, (E' \backslash M_d) \cup M_r \cup M_g)$ is a Hamiltonian cycle, denoted by HC, in $H(G)$; graph $H(G)$ has maximum degree 5, as each vertex may be incident, additionally, to a green edge and/or a red edge; and in HC the red edges and the green edges appear alternately.

Next, we construct a 2-page book embedding of graph $H(G)$ as follows. By the definition of red predecessors, any vertex incident to a red edge must be incident to an edge in E which lies in $I(HC)$ and an edge in E which lies in $E(HC)$. Break arbitrarily an edge in HC and embed the resulting Hamiltonian path along the book spine. The edges in E lying in $I(HC)$ are embedded in page 1; and the edges in E

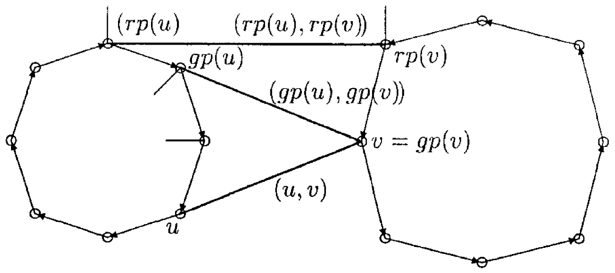


Fig. 2. An illustration: adding a red edge and a green edge.

lying in $E(HC)$ are embedded in page 2. Consider a segment of HC connecting a red edge and a green edge containing no colored edges, i.e. the edges in this segment are all in E. Let us focus on the head vertex of this segment connected to the green edge. If it is incident to two edges in E that both lie in $I(HC)$ (or $E(HC)$, respectively), then alternately embed the edges in the segment, starting from this head, in page 2 and page 1 (or page 1 and page 2, respectively). Since each edge in HC can be embedded in either page, applying the above process for every such segment of HC results in a valid 2-page book embedding of graph $H(G)$ such that every vertex has degree at least 1 in each page. □

Remark 2.7. In the proof of Lemma 2.6, we add $2k \leqslant \lfloor 2n/3 \rfloor$ edges to graph G to obtain a Hamiltonian planar supergraph $H(G)$. We do not need to add so many edges if our goal is only to obtain a Hamiltonian planar supergraph. Rather than using the above procedure, we can obtain a Hamiltonian planar supergraph by adding an edge (u', v') to graph G for every edge $(u, v) \in M'$, where u', v' are the directed predecessors of u, v in (directed) graph G'. In the resulting graph, the maximum degree of a vertex is 4. Therefore, adding (at most) $k \leqslant \lfloor n/3 \rfloor$ edges makes graph G Hamiltonian. A more dextrous analysis with a further improved bound of $\lfloor (n-4)/6 \rfloor$ (edges) is given in [6].

Theorem 2.8. LAPCS(NESTED, NESTED) *is NP-hard.*

Proof. The proof is a reduction from the MIS problem restricted to cubic planar bridgeless connected graphs. Given a cubic planar bridgeless connected graph G, we first use Lemma 2.6 to construct a 2-page book embedding $\mathscr{E}$ of G in linear time such that in either page every vertex has degree at least 1 (and thus at most 2).

In book embedding $\mathscr{E}$, suppose without loss of generality that, the vertices lying along the spine of the book are in the ordering $v_1, v_2, \ldots, v_{n-1}, v_n$. For the first page of the book embedding, construct a segment of 8 letters ccccdbab for each vertex, and for the second page, construct a segment ccccbdba for each vertex. The base sequence for the first page is formed by concatenating the n segments followed by an additional partial segment cccc, which is $(\text{ccccdbab})^n\text{cccc}$; the base sequence for the second page is formed similarly, which is $(\text{ccccbdba})^n\text{cccc}$.

In each page of the book embedding, if there are two edges incident to vertex v_i, then there are two arcs imposed on the corresponding constructed sequence each having a letter b in the ith segment as an endpoint (as for which one, it depends on the book embedding of the two edges to avoid arc crossing, i.e., inherits the non-crossing property of the embedding of edges in one page). In page 1, if there is exactly one edge incident to vertex v_i, then there is an arc in P_1 which has the second letter b in the ith segment (that is $S_1[8i]$) as an endpoint, and in P_1 there are two more arcs $(8i-4, 8i-3)$ and $(8i-2, 8i-1)$, associated with this edge. In page 2, if there is exactly one edge incident to vertex v_i, then there is an arc in P_2 which has the first letter b in the ith segment (that is $S_2[8i-3]$) as an endpoint, and in P_2 there are two more arcs $(8i-2, 8i-1)$ and $(8i, 8i+1)$, associated with this edge. This forms the two nested arc-annotated sequences (S_1, P_1) and (S_2, P_2).

Fig. 3 illustrates a small example of the above construction. The cubic graph in this example is K_4. The graph on the upper right shows a 2-page book embedding satisfying the conditions in Lemma 2.6, where the edges above the linear ordering of the vertices are in page 1, and the others are in page 2. Notice that every vertex has degree either 1 or 2 in each page. For each edge, there is a corresponding arc connecting two bases b, from the segments constructed for the endpoints of the edge, respectively. For instance, arc $(6, 32) \in P_1$ corresponds to edge (v_1, v_4). Basing on the

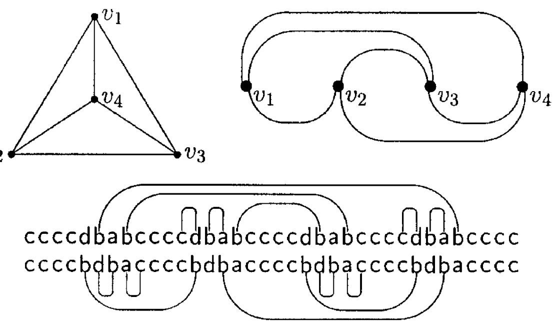

v_1
v_2
v_3
v_4
ccccdbabccccdbabccccdbabccccdbabcccc
ccccbdbaccccbdbaccccbdbaccccbdbacccc

Fig. 3. Mapping a cubic planar bridgeless graph to an instance of LAPCS(NESTED, NESTED).

arc connecting rules in the last paragraph, since vertex v_1 has degree 2 in page 1, the two letters b are both endpoints of some arcs corresponding to graph edges; While since v_2 has degree 1 in page 1, only the second letter b is the endpoint of some graph edge. But, two arcs $(12, 13), (14, 15) \in P_1$ are created to correspond to this situation. Recall that vertex v_i has degree 2 in page 1 (or page 2) if and only if it has degree 1 in page 2 (or page 1, respectively). The lower part of the figure shows the two constructed nested arc-annotated sequences with respect to the book embedding.

Define a base match $\langle i, j \rangle$ with $|i - j| \leqslant 4$ to be a *good* base match; otherwise a *bad* base match. Also call each base match $\langle i, j \rangle$ a $S_1[i]$-*match* (or equivalently $S_2[j]$-*match* because $S_1[i] = S_2[j]$).

One can check that every LAPCS of the above-constructed instance induces no bad base matches, since otherwise it is always possible to replace the first several bad base matches with their endpoints residing in some segment along one sequence with more good matches, contradicting the maximality of the LAPCS. Furthermore, every LAPCS induces no arc match and thus it can be transformed to induce all the $4(n + 1)$ good c-matches of form $\langle i, i \rangle$. Call this kind of LAPCS's *good*. It is not hard to check that for every vertex index i, there are at most two good base matches (but at least one) from the ith segment which could co-exist in a good LAPCS; and if there are two, then they both are b-matches. In other words, two b-matches $\langle 8i - 2, 8i - 3 \rangle$ and $\langle 8i, 8i - 1 \rangle$ are included in a good LAPCS if and only if vertex v_i is in the maximum independent set of graph G. Since no LAPCS can include an arc-match, if we let $v_{j_1}, v_{j_2}, v_{j_3}$ be the three neighbors of v_i in G, then the good LAPCS does not contain 2 good b-matches from segments j_1, j_2, j_3. By the construction, when a good LAPCS includes just one good b-match from segment j, then it can be replaced by either the good a-match or the good d-match, for any j. Therefore, graph G has a maximum independent set of cardinality k if and only if the constructed instance of LAPCS(NESTED, NESTED) has a (good) LAPCS of length exactly $4(n + 1) + 2k + (n - k) = 5n + k + 4$. This proves the theorem. □

In fact, the proof of Theorem 2.8 can be made much simpler by employing a reduction similar to those in [12,18]. The reason why we prefer a more complicated reduction is that this reduction actually shows the NP-hardness of the problem of finding a good LAPCS in two given nested arc-annotated sequences. Here, in a good LAPCS, the allowed forms of base matches are $\langle 2i - 1, 2i - 1 \rangle$, $\langle 2i - 1, 2_i \rangle$, $\langle 2i, 2i - 1 \rangle$, and $\langle 2i, 2i \rangle$. We use 2-FRAGMENTED LAPCS(NESTED, NESTED) to denote the problem of finding a good LAPCS in two given nested arc-annotated sequences. Generally, we can define c-FRAGMENTED LAPCS(NESTED, NESTED) for any positive integer c, in which the two given sequences are chopped into fragments of lengths at most c, and the allowed base matches are those between fragments at the same location. Notice that c-FRAGMENTED LAPCS(NESTED, NESTED) is actually a further restriction of $(c-1)$-DIAGONAL LAPCS(NESTED, NESTED). Therefore, the proof of Theorem 2.8 and the above discussion justify the following:

Theorem 2.9. c-FRAGMENTED LAPCS(NESTED, NESTED) *is NP-hard when* $c \geqslant 2$. c-DIAGONAL LAPCS(NESTED, NESTED) *is NP-hard when* $c \geqslant 1$.

3. c-FRAGMENTED LAPCS and c-DIAGONAL LAPCS

In this section, we investigate the c-FRAGMENTED LAPCS and the c-DIAGONAL LAPCS problems for arc-annotated sequences. We first give a simple efficient algorithm for solving 1-FRAGMENTED LAPCS(CROSSING, CROSSING) (which is equivalent to 0-DIAGONAL LAPCS(CROSSING, CROSSING)) and then PTAS's for c-FRAGMENTED LAPCS(NESTED, NESTED) when $c \geqslant 2$ and for c-DIAGONAL LAPCS (NESTED, NESTED) when $c \geqslant 1$.

*3.1. 1-*FRAGMENTED *LAPCS(*CROSSING, CROSSING*)*

Let $(S_1, P_1), (S_2, P_2)$ be an instance of 1-FRAGMENTED LAPCS(CROSSING, CROSSING). Assume without loss of generality that $n = |S_1| = |S_2|$. Clearly, ignoring the arcs we may compute a classical LCS T of S_1 and S_2 by a linear scan in $O(n)$ time. We construct a graph G associated with T as follows. If T induces base match $\langle i, i \rangle$, then create a vertex v_i. If T induces a pair of base matches $\langle i, i \rangle$ and $\langle j, j \rangle$ and (i, j) is an arc in either P_1 or P_2 but not both, then we impose an edge connecting v_i and v_j in G. It is clear that G has maximum degree 2, and thus every independent set of G one-to-one corresponds to an arc-preserving common subsequence of (S_1, P_1) and (S_2, P_2). Therefore, by computing a maximum independent set of G, which can be done in linear time since G is composed of a collection of disjoint cycles and paths, we get an LAPCS for (S_1, P_1) and (S_2, P_2).

Theorem 3.1. *The* 1-FRAGMENTED LAPCS(CROSSING, CROSSING) *problem is solvable in $O(n)$ time.*

*3.2. c-*FRAGMENTED *LAPCS(*NESTED, NESTED*)*

Before describing the PTAS, we review the notions of *tree decomposition* of a graph and *k-outerplanar graph*, which play important roles in our construction.

A *tree decomposition* of a graph $G = (V, E)$ is a pair $\mathscr{D} = (T, \mathscr{X})$ where $T = (U, F)$ is a tree and $\mathscr{X} = \{X_u \mid u \in U\}$ is a family of $|U|$ subsets of V such that the following hold:

- $\bigcup_{u \in U} X_u = V$.
- For each edge $(v_1, v_2) \in E$, there is a vertex $u \in U$ such that $\{v_1, v_2\} \subseteq X_u$.
- If $u_2 \in U$ is on the path connecting u_1 and u_3 in T, then $X_{u_1} \cap X_{u_3} \subseteq X_{u_2}$.

The *treewidth* associated with this tree decomposition is $tw(G, \mathscr{D}) = \max_{u \in U} |X_u| - 1$. The *treewidth* of G, denoted by $tw(G)$, is the minimum $tw(G, \mathscr{D})$ taken over all tree decompositions $\mathscr{D}$ of G.

The following lemma is widely known in the literature (see, e.g. [1]):

Lemma 3.2. *Given a graph G and a tree decomposition $\mathscr{D}$ of G, we can compute a maximum independent set of G in $O(2^k k^2 |G|)$ time, where $k = tw(G, \mathscr{D})$ and $|G|$ is the total number of vertices and edges in G.*

The notion of k-outerplanar graphs was introduced by Baker [3]. These graphs are defined inductively as follows. 1-*outerplanar graphs* are exactly outerplanar graphs. For $k \geqslant 2$, *k-outerplanar graphs* are those planar graphs G that have a planar embedding such that deleting all vertices on the exterior face of the embedding and all edges incident to them from the embedding yields a $(k-1)$-outerplanar graph.

Lemma 3.3 (Bodlaender [7]). *For $k \geqslant 1$, k-outerplanar graphs have treewidth less than or equal to $3k - 1$.*

By Lemmas 3.2 and 3.3, the MIS problem restricted to k-outerplanar graphs is solvable in linear time. This result was originally due to Baker [3] but based on a different approach.

For any integer $k \geqslant 2$, a *k-cover* of a graph $G = (V, E)$ is a family $\mathscr{F}$ of k subsets of V such that each vertex of G appears in at least $k-1$ subsets in $\mathscr{F}$. The notion of k-covers of graphs was introduced by Chen [9]. The idea behind Baker's PTAS [3] for the MIS problem restricted to planar graphs is to compute a k-cover $\mathscr{F}$ of a given planar graph G such that each subset in $\mathscr{F}$ induces a $(k-1)$-outerplanar subgraph of G. Extending this idea and applying Lemma 3.2, Chen [9] proved the following:

Lemma 3.4 (Chen [9]). *Suppose that $\mathscr{C}$ is a class of graphs $G = (V, E)$ such that given G and an integer $k \geqslant 2$, we can compute a k-cover $\mathscr{F}$ of G and a tree decomposition $\mathscr{D}_U$ of the subgraph G_U of G induced by each subset $U \in \mathscr{F}$ in $T(k, |G|)$ total time such that $tw(G_U, \mathscr{D}_U)$ is bounded from above by a number $f(k)$ independent of G. Then, given a graph G in $\mathscr{C}$ and an integer $k \geqslant 2$, we can compute an independent set of G in $T(k, |G|) + O(2^{f(k)}(f(k))^2 k|G|)$ time whose size is at least $\frac{k-1}{k}$ times the optimum. Consequently, the MIS problem restricted to graphs in $\mathscr{C}$ admits a PTAS.*

Now we are ready to present the PTAS for c-FRAGMENTED LAPCS(NESTED, NESTED).

Theorem 3.5. *Given an instance of c-FRAGMENTED LAPCS(NESTED, NESTED), say (S_1, P_1) and (S_2, P_2), and an integer $k \geqslant 2$, we can compute an arc-preserving common subsequence of (S_1, P_1) and (S_2, P_2) in $O(2^{(3k-4)c^2} k^3 c^5 (|S_1| + |S_2|))$ time whose length is at least $\frac{k-1}{k}$ times the optimum. Consequently, the c-FRAGMENTED LAPCS(NESTED, NESTED) problem admits a PTAS.*

Proof. Given an instance of c-FRAGMENTED LAPCS(NESTED, NESTED), say (S_1, P_1) and (S_2, P_2), and an integer k, we are asked to compute an arc-preserving common subsequence whose length is at least $\frac{k-1}{k}$ times the optimum. Without loss of generality, we assume that $|S_1| = |S_2| = mc$, and let S_i^j denote the jth fragment of length c in S_i. We construct two graphs G and H in the following.

Let V_j, $1 \leqslant j \leqslant m$, denote the set of vertices each corresponding to a base match $\langle (j-1)c + \ell_1, (j-1)c + \ell_2 \rangle$ in the jth fragment (i.e. $S_1[(j-1)c + \ell_1] = S_2[(j-1)c + \ell_2]$), where $1 \leqslant \ell_1, \ \ell_2 \leqslant c$. Clearly, $|V_j| \leqslant c^2$ for all j. Let $V = \bigcup_{1 \leqslant j \leqslant m} V_j$. Two vertices in V are connected via an edge if and only if they are conflicting base matches (i.e. they cross each other or violate the arc-preserving constraint). This forms a graph G. Observe that an independent set in G one-to-one corresponds to an arc-preserving common subsequence of (S_1, P_1) and (S_2, P_2) (with the c-fragmented restriction).

Consider an integer $k \geqslant 2$. We obtain a new graph H from G by merging the vertices in each subset V_j into a super-vertex v_j and keeping at most one edge between each pair of super-vertices. Since both P_1 and P_2 are nested, H is a simple planar graph. Notice that the number of vertices in H is $2m$.

Since H is planar, we can emulate Baker [3] to compute a k-cover $\mathscr{F}_H$ of H in $O(mk)$ time such that each subset in $\mathscr{F}_H$ induces a $(k-1)$-outerplanar subgraph of G. Let $\mathscr{F}_H = \{U_1, \ldots, U_k\}$. For each $U_i \in \mathscr{F}_H$, we can compute a tree decomposition $\mathscr{D}_i = (T_i, \mathscr{X}_i)$ of the subgraph H_i of H induced by U_i with $tw(H_i, \mathscr{D}_i) \leqslant 3k-4$, in $O(|U_i|)$ time [8].

For each $U_i \in \mathscr{F}_H$, let W_i be the subset of V obtained from U_i by replacing each super-vertex $v \in U_i$ with the original vertices merged into v. Similarly, for each $1 \leqslant i \leqslant k$ and each subset $X_{i,j} \in \mathscr{X}_i$, we obtain a subset $Z_{i,j}$ of V from $X_{i,j}$ by replacing each super-vertex $v \in X_{i,j}$ with the original vertices merged into v. One can verify that (1) $\{W_1, \ldots, W_k\}$ is a k-cover of G and (2) each $\mathscr{D}_i = (T_i, \{Z_{i,j} \mid X_{i,j} \in \mathscr{X}_i\})$ is a tree decomposition of the subgraph of G induced by W_i. Note that the treewidth of each $\mathscr{D}_i$ is less than or equal to $(3k-4)c^2$. Moreover, the k-cover $\{W_1, \ldots, W_k\}$ and the tree decompositions $\mathscr{D}_1$ through $\mathscr{D}_k$ can be computed in $O(k|G|) = O(kmc^2)$ total time. So, by Lemma 3.4, given G and k, we can compute an independent set of G in $O(2^{(3k-4)c^2} k^3 c^4 |G|)$ time whose size is at least $\frac{k-1}{k}$ times the optimum. In turn, Given (S_1, P_1), (S_2, P_2), and k, we can compute an arc-preserving common subsequence of (S_1, P_1) and (S_2, P_2) in $O(2^{(3k-4)c^2} k^3 c^5 (|S_1| + |S_2|))$ time whose length is at least $\frac{k-1}{k}$ times the optimum. This finishes the proof of the theorem. □

3.3. *c*-DIAGONAL *LAPCS*(NESTED, NESTED)

The PTAS in Section 3.2 can be easily extended to a PTAS for c-DIAGONAL LAPCS(NESTED, NESTED) as shown below.

Theorem 3.6. *Given an instance of* c-DIAGONAL LAPCS(NESTED, NESTED), *say* (S_1, P_1) *and* (S_2, P_2), *and an integer* $k \geqslant 2$, *we can compute an arc-preserving common subsequence of* (S_1, P_1) *and* (S_2, P_2) *in* $O(2^{(3ck-4)(ck-1)^2} c^9 k^9 (|S_1| + S_2|))$ *time whose length is at least* $\frac{k-1}{k}$ *times the optimum. Consequently, the* c-DIAGONAL LAPCS(NESTED, NESTED) *problem admits a PTAS.*

Proof. Given an instance of c-DIAGONAL LAPCS(NESTED, NESTED), say (S_1, P_1) and (S_2, P_2), and an integer $k \geqslant 2$, we are asked to compute an arc-preserving common subsequence whose length is at least $\frac{k-1}{k}$ times the optimum. Without loss of generality, we assume that $|S_1| = |S_2|$. The algorithm uses the PTAS designed for c-FRAGMENTED LAPCS(NESTED, NESTED) as a subroutine.

Fix a constant b (to be specified later). For every $i \in [1, b]$, let Q_i denote the b-FRAGMENTED LAPCS(NESTED, NESTED) problem for (S_1, P_1) and (S_2, P_2) where the first fragment has length i, each of the others but the last fragment has length b. Let ℓ_i^* denote the length of an LAPCS, say T_i, for problem Q_i. Suppose that R is an LAPCS for the original problem. Then, every base match in R is a legal base match in at least $(b-c+1)$ out of b problems $Q_1, Q_2, \ldots, Q_b$. Therefore, the maximum among $\ell_1^*, \ell_2^*, \ldots, \ell_b^*$ is at least as large as $\frac{b-c+1}{b}|R|$. Notice that by Theorem 3.5 we may compute an arc-preserving common subsequence C_i for problem Q_i in $O(2^{(3b-1)b^2} b^8 (|S_1| + |S_2|))$ time such that the length of C_i is at least as large as $\frac{b}{b+1}\ell_i^*$.

Let C be the longest one among $C_1, \dots, C_b$. By the discussions in the previous paragraph, $|C| \geqslant \frac{b}{b+1} \cdot \frac{b-c+1}{b}|R| = \frac{b-c+1}{b+1}|R|$. By setting $b = ck - 1$, we have $|C| \geqslant \frac{k-1}{k}|R|$. Thus, the algorithm that outputs C on input $(S_1, P_1), (S_2, P_2)$ and k runs in $O(2^{(3ck-4)(ck-1)^2} c^9 k^9 (|S_1| + |S_2|))$ time and hence is a PTAS for the c-DIAGONAL LAPCS(NESTED, NESTED) problem. □

4. UNARY LAPCS(NESTED, NESTED)

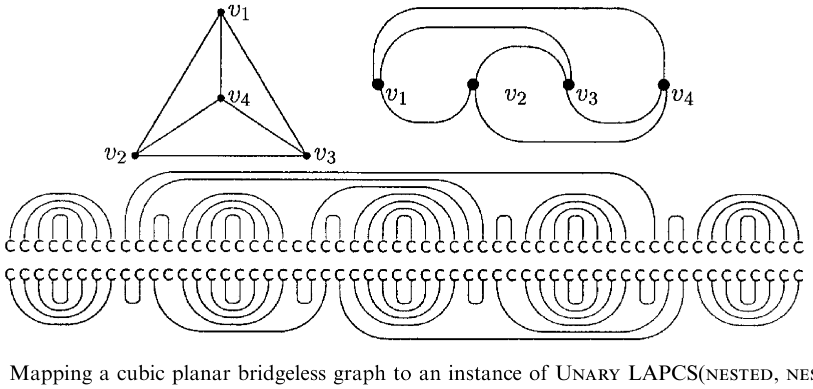
In this section, we consider the UNARY LAPCS(NESTED, NESTED) problem, a special case where all the bases are the same. We show that UNARY LAPCS(NESTED, NESTED) is also NP-hard, and design a $\frac{4}{3}$-approximation algorithm for it.

Theorem 4.1. *The* UNARY LAPCS(NESTED, NESTED) *problem is NP-hard.*

Proof. The reduction is again from the MIS problem for cubic planar bridgeless connected graphs. We also begin with the 2-page book embedding $\mathscr{E}$ of a given cubic planar bridgeless connected graph G, obtained by employing Lemma 2.6. This time, each of the sequences S_1 and S_2 consists of $12n + 8$ bases. The arc sets P_1 and P_2 both contain arcs $\{(12i + j, 12i + 9 - j) \mid 1 \leqslant j \leqslant 4, 0 \leqslant i \leqslant n\}$. But, in addition, P_1 also includes arcs of form $(12i + 11, 12i + 12)$ and P_2 includes arcs of form $(12i + 9, 12i + 10)$.

If there are two edges incident to vertex v_i in page 1 of the book embedding $\mathscr{E}$, then we include two arcs in P_1, each of which has an endpoint at position $12(i-1) + 9$ or $12(i-1) + 10$ in the segment constructed for v_i (as for which arc ends at which position, it depends on the book embedding of the two edges). If there is exactly one edge incident to vertex v_i, then we define an arc in P_1 with position $12(i-1) + 10$ as an endpoint. Similarly, in page 2, if there are two edges incident to vertex v_i, then P_2 has two arcs, each of which has an endpoint $12(i-1) + 11$ or $12(i-1) + 12$. If there is exactly one edge incident to vertex v_i in page 2, then P_2 has an arc with $12(i-1) + 11$ as an endpoint. This forms two nested arc-annotated sequences (S_1, P_1) and (S_2, P_2).

See Fig. 4 for a small example of the construction. The cubic graph in this example again is K_4, and a 2-page book embedding satisfying the conditions in Lemma 2.6 is shown on the right hand side, where the edges above the line of vertices are in page 1, and the others are in page 2. Each edge of the graph corresponds to an arc connecting two bases from the segments constructed for the endpoints of the edge. For instance, arc $(9, 46) \in P_1$ corresponds to edge (v_1, v_4).



v_1
v_4
v_2
v_3
v_1
v_2
v_3
v_4
cccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccc

Fig. 4. Mapping a cubic planar bridgeless graph to an instance of UNARY LAPCS(NESTED, NESTED).

Similarly to the proof of Theorem 2.8, it is not hard to check that every LAPCS can be transformed to include all the matches $\{\langle 12i+j, 12i+j\rangle \mid 1\leqslant j\leqslant 8, 0\leqslant i\leqslant n\}$. Again, call such an LAPCS *good*, and let us focus on a good LAPCS. In such an LAPCS, $(S_1[12i+1, 12i+8], P_1[12i+1, 12i+8])$ serves as separating blocks such that all other base matches must be between $S_1[12i+9, 12i+12]$. and $S_2[12i+9, 12i+12]$. Hence, there are either 2 or 3 co-existing base matches between $S_1[12i+9, 12i+12]$ and $S_2[12i+9, 12i+12]$, for every i. Moreover, when there are 2 base matches we can make them to have the forms $\langle 12i+11, 12i+9\rangle$ and $\langle 12i+12, 12i+10\rangle$, since $\langle (12i+11, 12i+12), (12i+9, 12i+10)\rangle$ is an arc match. Also, 3 base matches are possible if and only if bases $S_1[12i+9]$, $S_1[12i+10]$, $S_2[12i+11]$ and $S_2[12i+12]$ can be matched, since in this case the LAPCS cannot induce the arc match $\langle (12i+11, 12i+12), (12i+9, 12i+10)\rangle$. From this observation, if (v_i, v_j) is an edge in graph G, then it is impossible that there are 3 co-existing base matches between $S_1[12i+9, 12i+12]$ and $S_2[12i+9, 12i+12]$, and there are 3 co-existing base matches between $S_1[12j+9, 12j+12]$ and $S_2[12j+9, 12j+12]$. In other words, two bases of the arc corresponding to an edge in G cannot both enter the LAPCS. Therefore, the vertices v_i satisfying the condition that there are 3 co-existing base matches between $S_1[12i+9, 12i+12]$ and $S_2[12i+9, 12i+12]$ form an independent set of graph G. So, the graph G has an independent set of cardinality k if and only if the constructed instance of Unary LAPCS(nested, nested) has a (good) LAPCS of length $8(n+1)+3k+2(n-k)=10n+k+8$. This proves the theorem. □

The following corollary follows from the above construction.

Corollary 4.2. c-fragmented Unary LAPCS(nested, nested) *when* $c\geqslant 4$ *and* c-diagonal Unary LAPCS(nested, nested) *when* $c\geqslant 3$ *are NP-hard.*

Theorem 4.3. Unary LAPCS(nested, nested) *admits a* $\frac{4}{3}$*-approximation algorithm.*

Proof. Given a pair of Unary sequences with nested arcs (S_1, P_1) and (S_2, P_2), where $|S_1| = n_1 \leqslant n_2 = |S_2|$, an arc-preserving common subsequence T satisfies *left-priority* if for every arc $(i_1, i_2)\in P_1$, T does not contain $S_1[i_2]$ unless it contains $S_1[i_1]$. For the ease of exposition, we call such an arc-preserving common subsequence satisfying left-priority a *Lep-APCS*. A longest Lep-APCS is denoted for short by *Lep-LAPCS*. A key ingredient of our approximation algorithm is a polynomial-time algorithm for computing a Lep-LAPCS, as shown below.

The basic idea is dynamic programming. For every pair of integers i and i', where $1\leqslant i\leqslant i'\leqslant n_1$, let $\tilde{S}_1[i, i']$ denote the subsequence of $S_1[i, i']$ by deleting the bases that are the right endpoints of arcs whose left endpoints are not in the interval $[i, i']$,[5] $\tilde{\imath}$ (or $\tilde{\imath}'$) the index of the base to the right (or left, respectively) of $S_1[i]$ (or $S_1[i']$, respectively) in $\tilde{S}_1[i, i']$, and $P_1[i, i']$ the subset of arcs whose both endpoints are in the interval $[i, i']$.

Let $DP(i, i'; j, j')$ denote the length of a Lep-LAPCS for the subsequences $(\tilde{S}_1(i, i'], P_1[i, i'])$ and $(S_2[j, j'], P_2[j, j'])$, where $1\leqslant i\leqslant i'\leqslant n_1$ and $1\leqslant j\leqslant j'\leqslant n_2$. Here we require that the pair j and j' satisfy that there is no arc in P_2 such that its one endpoint is in the interval $[j, j']$ while the other is not. The solution to our original Lep-LAPCS problem would be stored in $DP(1, n_1; 1, n_2)$. In the following, when we say that $S_1[k]$ (or $S_2[k]$) is *free*, it is not an endpoint of any arc in $P_1[i, i']$ (or $P_2[j, j']$, respectively). The dynamic programming is a two-step computation.

[5]Note: We may assume without loss of generality that $S_1[i]$ and $S_1[i']$ are both in $\tilde{S}_1[i, i']$. Otherwise, if, for example, $S_1[i]$ is not in $\tilde{S}_i[i, i']$, then $\tilde{S}_1[i, i'] = \tilde{S}_1[i+1, i']$.

Step 1: For every $(j_1,j_2)\in P_2$, we do the following two-phase pre-computation.

Phase 1: If $j_2-j_1>1$, then let $j=j_1+1$. Let j' be a position in range $[j_1+1,j_2-1]$ such that $(j,j')\notin P_2$ and no arc in P_2 has exactly one endpoint in range $[j,j']$. The recurrence relation for the computation of entry $DP(i,i';j,j')$ is defined for several cases.

If $S_2[j']$ is free, then in the case that $S_1(i')$ is free,

$$DP(i,i';j,j')=\max\begin{cases}DP(i,i';j,j'-1),\\ DP(i,\bar{\imath}';j,j'),\\ DP(i,\bar{\imath}';j,j'-1)+1.\end{cases}$$

In the other case, i.e. $S_1[i']$ is the right endpoint of some arc in $P_1[i,i']$,

$$DP(i,i';j,j')=\max\begin{cases}DP(i,i';j,j'-1),\\ DP(i,\bar{\imath}';j,j').\end{cases}$$

If $(j'',j')\in P_2$ for some $j''\in[j+1,j'-1]$, then

$$DP(i,i';j,j')=\max_{i\leqslant i''\leqslant i'}\{DP(i,i''-1;j,j''-1)+DP(i'',i';j''j')\}.$$

Phase II: If $S_1[i]$ is free but $S_1[i']$ is not, then

$$DP(i,i';j_1,j_2)=\max\begin{cases}DP(\tilde{\imath},i';j_1,j_2),\\ DP(\tilde{\imath},i';j_1+1,j_2-1)+1,\\ DP(i,\bar{\imath}';j_1,j_2).\end{cases}$$

Similarly, if $S_1[i']$ is free, then no matter whether or not $S_1[i]$ is free,

$$DP(i,i';j_1,j_2)=\max\begin{cases}DP(\tilde{\imath},i';j_1,j_2),\\ DP(\tilde{\imath},i';j_1+1,j_2-1)+1,\\ DP(i,\bar{\imath}';j_1,j_2),\\ DP(i',\tilde{\imath};j_1+1,j_2-1)+1.\end{cases}$$

If $(i,i')\in P_1[i,i']$, then

$$DP(i,i';j_1,j_2)=DP(\tilde{\imath},\bar{\imath}';j_1+1,j_2-1)+2.$$

If neither of $S_1[i]$ and $S_1[i']$ is free, but $(i,i')\notin P_1[i,i']$, then

$$DP(i,i';j_1,j_2)=\max\begin{cases}DP(i,i';j_1+1,j_2-1),\\ DP(\tilde{\imath},i';j_1+1,j_2-1)+1,\\ DP(i,\bar{\imath}';j_1,j_2),\\ DP(\tilde{\imath},i';j_1,j_2).\end{cases}$$

Step 2: Let j' be a position in range $[1,n_2]$ such that $(1,j')\notin P_2$ and no arc in P_2 has exactly one endpoint in range $[1,j']$. The recurrence relation for the computation of entry $DP(i,i';1,j')$ is defined for several cases.

If $S_2[j']$ is free, then in the case that $S_1[i']$ is free,

$$DP(i,i';1,j')=\max\begin{cases}DP(i,i';1,j'-1),\\ DP(i,\bar{\imath}',1,j'),\\ DP(i,\bar{\imath}';1,j'-1)+1.\end{cases}$$

In the other case, i.e. $S_1[i']$ is the right endpoint of some arc in $P_1[i,i']$,

$$DP(i,i';1,j')=\max\begin{cases}DP(i,i';1,j'-1),\\ DP(i,\bar{\imath}';1,j').\end{cases}$$

If $(j'', j') \in P_2$ for some $j'' \in [2, j' - 1]$, then

$$DP(i, i' : 1, j') = \max_{i \leqslant i'' \leqslant i'} \{DP(i, i'' - 1; 1, j'' - 1) + DP(i'', i'; j'', j')\}.$$

Therefore, $DP(1, n_1; 1, n_2)$ can be computed in $O(n_1^3 n_2)$ time, since there are $O(n_1^2 n_2)$ entries and each entry is obtained by checking at most $O(n_1)$ terms. Employing a standard back-tracing technique, a Lep-LAPCS can be recovered in $O(n_1 n_2)$ time. Let n_1^* denote the length of a Lep-LAPCS.

We may similarly define arc-preserving common subsequences satisfying *right-priority*, and denote the longest ones as *Rip-LAPCS*'s. Using the same computation technique as in the above, a Rip-LAPCS and its length can be computed in $O(n_1^3 n_2)$ time too. Let n_2^* denote the length of a Rip-LAPCS.

Let T_0 denote a longest APCS under the constraint that it contains no arc-match at all, and $n_0^* = |T_0|$. Such a T_0 can be computed by deleting all bases which are right endpoints of arcs in P_1 and P_2 from both sequences and then taking the shorter one. Let n^* denote the length of a (genuine) LAPCS T^* for (S_1, P_1) and (S_2, P_2), and m^* denote the number of arc-matches in T^*. If $m^* \leqslant \frac{1}{4} n^*$, then we conclude that $n_0^* \geqslant n^* - m^* \geqslant \frac{3}{4} n^*$. Otherwise, the number of base-matches in T^* which do not form arc-matches is less than $\frac{1}{2} n^*$. Therefore, $\max\{n_1^*, n_2^*\} > \frac{3}{4} n^*$. It follows that by taking the longest one among T_0, a Lep-LAPCS, and a Rip-LAPCS, as the approximate solution, its length is guaranteed to be at least $\frac{3}{4}$ times the optimum. Notice that the overall algorithm runs in $O(n_1^3 n_2)$ time. □

5. Concluding remarks

We have completely answered an open question proposed in [12] on whether or not it is NP-hard to compute an LAPCS for two nested arc-annotated sequences. In particular, we have shown that 2-FRAGMENTED LAPCS(NESTED, NESTED) is already NP-hard. In the positive aspect, we designed a PTAS for c-DIAGONAL LAPCS (NESTED, NESTED) for any integer $c \geqslant 1$. However, does the general LAPCS(NESTED, NESTED) admit a PTAS? We leave it as an open question. A special case, UNARY LAPCS(NESTED, NESTED) has also been investigated. Besides its hardness result, we have presented an approximation algorithm with a ratio better than the best ratio known for the general LAPCS(NESTED, NESTED). Investigating this special case, although may not be of practical interest, could provide some insight into the general problem LAPCS(NESTED, NESTED).

Another interesting research topic would be to design better (than 2) approximation algorithms for LAPCS (CROSSING, CROSSING), which also has interesting applications in protein threading and RNA tertiary structure comparison.

Acknowledgments

The authors thank Therese Biedl, Erik Demaine, Paul Kearney, Bin Ma, and Jinbo Xu for many valuable discussions. In particular, Therese pointed out Lemma 2.3 to us which directly leads to Corollary 2.4 and the discussions with Therese simplified the proof of the first part (subhamiltonian) of Lemma 2.6, which is a key (i.e. 2-page book embedding) to the proof of Theorem 2.8. They also thank the anonymous referee for many helpful comments.

References

[1] S. Arnborg, Efficient algorithms for combinatorial problems on graphs with bounded decomposability: a survey, BIT 25 (1985) 2–23.

[2] V. Bafna, S. Muthukrishnan, R. Ravi, Computing similarity between RNA strings, in: Proceedings of the 6th Annual Symposium on Combinatorial Pattern Matching (CPM'95), Helsinki, Finland, Lecture Note in Computer Science, Vol. 937, 1995, pp. 1–16.

[3] B.S. Baker, Approximation algorithms for NP-complete problems on planar graphs, J. ACM 41 (1994) 153–180.

[4] T.C. Biedl, P. Bose, E.D. Demaine, A. Lubiw, Efficient algorithms for Peterson's matching theorem, in: Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'99), Baltimore, MD, 1999, pp. 130–139.

[5] T.C. Biedl, G. Kant, M. Kaufmann, On triangulating planar graphs under the four-connectivity constraint, Algorithmica 19 (1997) 427–446.

[6] T.C. Biedl, G.-H. Lin, Planar cubic graphs are subhamiltonian, Manuscript, Department of Computer Science, University of Waterloo, March 2000.

[7] H.L. Bodlaender, Planar graphs with bounded treewidth, Technical Report RUU-CS-88-14, Department of Computer Science, Utrecht University, The Netherlands, March 1988.

[8] H.L. Bodlaender, A linear time algorithm for finding tree-decompositions of small treewidth, SIAM J. Comput. 25 (1996) 1305–1317.

[9] Z.-Z. Chen, Efficient approximation schemes for maximization problems on $K_{3,3}$-free or K_5-free graphs, J. Algorithms 26 (1998) 166–187.

[10] F.R.K. Chung, F.T. Leighton, A.L. Rosenberg, Embedding graphs in books: a graph layout problem with applications to VLSI design, SIAM J. Algebraic Discrete Methods 8 (1987) 33–58.

[11] F. Corpet, B. Michot, RNAling program: alignment of RNA sequences using both primary and secondary structures, Comput. Appl. Bio-sci. 10 (1994) 389–399.

[12] P.A. Evans, Algorithms and complexity for annotated sequence analysis, Ph.D. Thesis, University of Victoria, 1999.

[13] M.R. Garey, D.S. Johnson, G.L. Miller, C.H. Papadimitriou, The complexity of coloring circular arcs and chords, SIAM J. Algebraic Discrete Methods 1 (1980) 216–227.

[14] M.R. Garey, D.S. Johnson, L. Stockmeyer, Some simplified NP-complete graph problems, Theoret. Comput. Sci. 1 (1976) 237–267.

[15] D. Goldman, S. Istrail, C.H. Papadimitriou, Algorithmic aspects of protein structure similarity, in: IEEE Proceedings of the 40th Annual Conference of Foundations of Computer Science (FOCS'99), 1999, pp. 512–521.

[16] D. Gusfield, Algorithms on Strings, Trees, and Sequences, Cambridge University Press, New York, 1997.

[17] D.S. Hirschberg, The longest common subsequence problem, Ph.D. Thesis, Princeton University, 1975.

[18] T. Jiang, G.-H. Lin, B. Ma, K. Zhang, The longest common subsequence problem for arc-annotated sequences, in: Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching (CPM 2000), Lecture Note in Computer Science, Vol. 1848, 2000, pp. 154–165. Full paper accepted by Journal of Discrete Algorithms.

[19] H. Lenhof, K. Reinert, M. Vingron, A polyhedral approach to RNA sequence structure alignment, in: Proceedings of the Second Annual International Conference on Computational Molecular Biology (RECOMB'98), New York, NY, 1998, pp. 153–159.

[20] M. Li, B. Ma, L. Wang, Near optimal multiple sequence alignment within a band in polynomial time, in: ACM Proceedings of the 32nd Annual Symposium on Theory of Computing (STOC'00), Portland, OR, 2000, pp. 425–434.

[21] J. Peterson, Die theorie der regulären graphs (the theory of regular graphs), Acta Math. 15 (1891) 193–220.

[22] D. Sankoff, Simultaneous solution of the RNA folding, alignment, and protosequence problems, SIAM J. Appl. Math. 45 (1985) 810–825.

[23] R.A. Wagner, M.J. Fischer, The string-to-string correction problem, J. ACM 21 (1974) 168–173.

[24] M. Yannakakis, Embedding planar graphs in four pages, J. Comput. System Sci. 38 (1989) 36–67.

[25] K. Zhang, L. Wang, B. Ma, Computing similarity between RNA structures, in: Proceedings of the 10th Annual Symposium on Combinatorial Pattern Matching (CPM'99), Warwick, UK, Lecture Note in Computer Science, Vol. 1645, 1999, pp. 281–293.