Comparing RNA Structures: Towards an Intermediate Model Between the Edit and the Lapcs Problems

Guillaume Blin[1], Guillaume Fertin[2], Gaël Herry[2], and Stéphane Vialette[3]

[1] IGM-LabInfo - UMR CNRS 8049 - Université de Marne-la-Vallée
77 454 Marne-la-Vallée Cedex 2 - FRANCE - gblin@univ-mlv.fr
[2] LINA - FRE CNRS 2729 - Université de Nantes
44322 Nantes Cedex 3 - FRANCE - {fertin,herry}@lina.univ-nantes.fr
[3] LRI - UMR CNRS 8623 - Université Paris-Sud
91405 Orsay Cedex - FRANCE - vialette@lri.fr

Abstract. In the recent past, RNA structure comparison has appeared as an important field of bioinformatics. In this paper, we introduce a new and general intermediate model for comparing RNA structures: the Maximum Arc-Preserving Common Subsequence problem (or Mapcs). This new model lies between two well-known problems – namely the Longest Arc-Preserving Common Subsequence (Lapcs) and the Edit distance. After showing the relationship between Mapcs, Lapcs, Edit, and also the Maximum Linear Graph problem, we will investigate the computational complexity landscape of Mapcs, depending on the RNA structure complexity.

Keywords: RNA structures, arc-annotated sequences, motif extraction

1 Introduction

In computational biology, the understanding of biological mechanisms is frequently induced by sequences comparison. However, in the context of RiboNucleic Acid molecules (RNA), one cannot focus only on sequences. Indeed, it is now clearly established that the conformation of an RNA molecule partially determines its function and therefore, RNA comparison has certainly to take into account both the sequence and the structure. From a combinatorial point of view, an RNA molecule may be described by the sequence of its bases *i.e.*, a single strand composed of the nucleotides A, C, G and U (also called the *primary structure*), together with the set of hydrogen bonds that connect pairs of bases. Those pairings induce a specific conformation, usually called *secondary structure* if it can be drawn planarly, and *tertiary structure* otherwise.

At a theoretical level, the RNA structure comparison problem has been addressed with different paradigms allowing flexibility on the comparison criteria [1, 5–7, 10, 13, 15]. Nevertheless, they all rely on the concept of *structure comparison*. In order to compare two RNA structures, one has to consider a set Δ of operations on bases and/or hydrogen bonds. Given such a set Δ, comparing two

RNA structures usually reduces to finding a series of operations of Δ – an *edit script* – that transforms one structure into the other. Providing a cost for each operation of Δ allows us to evaluate the *cost* $\mathcal{C}$ of any edit script by summing the cost of all operations of the edit script. Then, referring to the standard *parsimony criterion*, the goal is to find the edit script transforming one structure into the other that minimizes the total cost $\mathcal{C}$.

Existing paradigms mainly differ in the set of allowed operations: some consider only operations that can behave separately on bases and on hydrogen bonds, whereas others take into account operations that can act separately or simultaneously on bases and hydrogen bonds. In the past few years, essentially due to the increase of the number of determined RNA structures, their comparison has become all the more important. Unfortunately, for most paradigms, comparing such structures turns out to be an intractable problem. Nevertheless, recent research [11, 16, 18] has shown that relaxing the constraints on the preservation of the primary structure makes the RNA structure comparison problem tractable for more general cases including some types of tertiary structures. However, a certain gap lies between simplistic and sophisticated paradigms. The former ones only use the structure in order to constrain the possible edit scripts over a set of simple operations ; for instance, the conservation/loss of a hydrogen bond is not considered in the similarity computation. On the contrary, the latter ones consider more biologically relevant operations and their associated costs ; for instance, a simultaneous deletion of a hydrogen bond together with one or two of its incident bases is considered as a single operation, to which a specific cost is assigned. However, the existence of such operations in the model makes the problem become hard even for very restricted structures.

In this article, we introduce an intermediate paradigm, called Maximum Arc-Preserving Common Subsequence, or MAPCS. MAPCS uses the structure both in order to constrain the possible edit scripts and to estimate the similarity between two RNA structures, but with operations simpler than the ones of sophisticated paradigms (more precisely, MAPCS can be seen as a more realistic extension of the well-known LAPCS problem [10], while being simpler than EDIT). The reader should notice that MAPCS differs from the sequence-structure alignment problem defined by Bafna *et al.* [2] since arcs, that have to be preserved, add constraints on the possible edit scripts. After some preliminaries and definitions, we first describe how the MAPCS problem is related to the LAPCS and the MLG problem, introduced in [9]. We then study the computational complexity of the MAPCS problem. This study is another step towards establishing more precisely the complexity landscape of the RNA structure comparison problem.

2 Preliminaries and Related Works

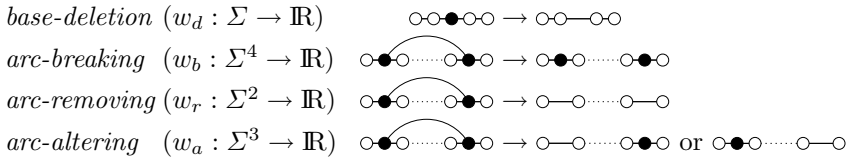
From a combinatorial point of view, one can distinguish two types of modeling allowing for various flexibility and precision in the encoding of RNA structures: (i) a representation that includes both the nucleotide sequence and the hydrogen bonds, called *arc-annotated sequence*, originally introduced by Evans [10],

and (ii) representations using graphs, which do not necessarily take into account the precise label of the nucleotides that compose the sequence, such as 2-interval graphs [19] and linear graphs [9]. We will be more interested here in arc-annotated sequences and linear graphs (intensively studied in the recent past [1, 5–7, 10, 13, 15]).

2.1 Arc-annotated sequences : Problems Edit and Lapcs

Given a finite alphabet Σ, an arc-annotated sequence is defined by a pair (S_1, P_1), where S_1 is a string on Σ^* and P_1 is a set of arcs connecting pairs of characters of S_1. In reference to RNA structures, we will refer to the characters as *bases*. The pair (S_1, P_1) is called an *RNA arc-annotated sequence* if $S_1 \in \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{U}\}^*$, and each arc of P_1 connects either bases A and U, or bases C and G. Any base with no arc incident to it is said to be *free*. Usually, five complexity levels reflecting the structure of the arcs are considered [10]: (1) Plain – there is no arc, (2) Chain – no base is incident to more than one arc and no two arcs are crossing or nest, (3) Nested (Nest) – no base is incident to more than one arc and no two arcs are crossing, (4) Crossing (Cros) – no base is incident to more than one arc and (5) Unlimited (Unlim) – no restriction.

Those five levels respect an obvious inclusion relation denoted by the $\subset$ operator: Plain $\subset$ Chain $\subset$ Nested $\subset$ Crossing $\subset$ Unlimited. Notice that the absence of arcs makes Plain a very low informative level and, since Plain $\subset$ Chain, it is of little interest in the context of RNA structure comparison and will not be considered in this paper. In order to compare arc-annotated sequences, we consider the set of operations (and their associated costs) initially introduced in [17]. This set is composed of four substitution operations which induce renaming of bases in the arc-annotated sequence: *base-match* ($w_m : \Sigma^2 \to \mathbb{R}$), *base-mismatch* ($w_m : \Sigma^2 \to \mathbb{R}$), *arc-match* ($w_{am} : \Sigma^4 \to \mathbb{R}$), *arc-mismatch* ($w_{am} : \Sigma^4 \to \mathbb{R}$). Moreover, it also contains four deletion operations which induce deletion of bases and/or arcs, which we list together with their associated cost:



The *edit distance* between two arc-annotated sequences (S_1, P_1) and (S_2, P_2) is defined as the minimum cost of any edit script from (S_1, P_1) to (S_2, P_2). The problem consisting in finding this distance is called Edit. To any edit script from (S_1, P_1) to (S_2, P_2) corresponds an *alignment* of the bases of S_1 and S_2 such that (i) any base which is inserted or deleted in a sequence is aligned with a *gap* (indicated by $-$) and (ii) any two bases (one per sequence) which are (mis)matched are aligned together. As illustrated in Table 1, Lin et al. proved in [17] that finding the edit distance between an arc-annotated sequence of Crossing type

and one of PLAIN type (denoted as EDIT(CROS, PLAIN)) is *MAX-SNP hard*. Thus, any harder problem (in terms of restriction levels) is also *MAX-SNP hard*. Moreover, they gave a polynomial-time dynamic programming algorithm for the problem EDIT(NEST, PLAIN). Blin et al. [6] showed that EDIT(NEST, NEST) is **NP**-complete.

The LAPCS problem was introduced by Evans in [10], and can be defined as follows: given two arc-annotated sequences (S_1, P_1) and (S_2, P_2), find the alignment of S_1 and S_2 which maximizes the number of matched positions and that satisfies the following conditions: for any arc (i, j) in P_1 (resp. in P_2), if bases i and j are both matched to bases of S_2 (resp. S_1) – say p and q – then (p, q) is an arc in P_2 (resp. P_1). In other words, an arc cannot be away from the alignment if none of its incident bases is away too. The computational complexity of the LAPCS problem has been studied in [10, 15], and the main results are summarized in Table 1. Of importance here is the result of Blin et al. [7], who proved that the LAPCS problem can actually be seen as a very specific case of the EDIT problem. More precisely, LAPCS can be seen as a particular case of EDIT where the cost system for edit operations is the following: $w_r = 2w_d = 2w_a$, and all substitution operations and arc-breakings are prohibited with an arbitrary high cost. The main idea is to penalize deletion operations proportionally to the number of bases that are deleted.

2.2 Linear graphs and the MLG Problem

As mentioned previously, one possible way of representing RNA structures is by means of linear graphs [9]. A linear graph of order n is a vertex-labeled graph where each vertex is labeled by a distinct integer from $\{1, 2, \ldots n\}$ (the order of the vertices is induced by the labels) and is of degree at least one. Any edge between two vertices i and j, with $i < j$, may be defined as the pair (i, j). Linear graphs thus represent RNA structures in which only the hydrogen bonds are considered – the identity of the bases are ignored. Two edges of a linear graph are called *independent* if they do not share a vertex. Similarly to arc-annotated sequences, the four levels of arc structures CHAIN, NESTED, CROSSING and UNLIMITED may be used in this model.

In order to compare linear graphs, we define the notion of *occurrence* of one linear graph in another as follows. Given two linear graphs G_1 and G_2, G_1 is said to occur in G_2 (or G_1 is called a *subgraph* of G_2) if one can obtain G_1 from G_2 (regardless of the vertex labels) by a sequence of edge and vertex deletions. More formally, the deletion of vertex i consists in (1) the deletion of all the edges incident to vertex i, (2) the deletion of vertex i and of any vertex of degree zero and (3) the relabeling of all remaining vertices preserving the original order. Provided with those notations, the RNA structure comparison problem using linear graphs – noted MLG – is defined as follows: given two linear graphs G_1 and G_2, find a maximum size – in terms of edges – common linear subgraph. Hence, this problem comes down to finding a common substructure that has the largest number of arcs between two given structures. We note that there exists some variants of the problem [12, 19].

$A \times B$	Chain	Nested		Crossing			Unlimited			
	Chain	Chain	Nest	Chain	Nest	Cros	Chain	Nest	Cros	Unlim
Edit	$O(nm)$ [10]	$O(nm^3)$ [15]	**NP**C [6]	MAX-SNP hard [17]						
Lapcs	$O(nm)$ [10]	$O(nm^3)$ [15]	**NP**C [15, 10]							
Mlg	$O(nm)$ [14]	$O(n^2m)$ [18]	$O(n^2m^2)$ [18]	$O(n^4 \log^3 n)$ [16]	**NP**C [8, 19]		$O(n^4 \log^3 n)$ [16]		**NP**C [8, 19]	

Table 1. RNA structure comparison: computational complexity of the Lapcs, Edit and Mlg problems considering input structures resp. of A and B types. For both the Edit and the Lapcs problems, n and m denote resp. the number of bases of the sequences of A and B types. For the Mlg problem, n and m denote the number of vertices of each linear graph, $n \geq m$.

As illustrated in Table 1, seeking for a maximal common substructure is easier when the maximality criterion relies only on the number of common arcs (Mlg), rather than on common bases (Lapcs, Edit). More precisely, one may note that when at least one of the input structures is of Chain or Nested type, Mlg is always polynomial time solvable.

3 Maximum Arc-Preserving Common Subsequence

In order to fill the gap which lies between simplistic and sophisticated paradigms like respectively, Lapcs and Edit, we introduce here a new paradigm,that we name Maximum Arc-Preserving Common Subsequence. The purpose of Mapcs is to overcome both the lack of expressiveness of Lapcs and the intrinsic complexity of Edit due to its sophisticated operations. Moreover, as illustrated in Table 1, according to the results of Mlg, restricting the complexity of the similarity criteria may be a way of going further ; indeed, we then may be able to solve, in polynomial time, more instances. The Mapcs is defined formally as follows: given two arc-annotated sequences (S_1, P_1) and (S_2, P_2) and two functions $f_b : \Sigma \to \mathbb{N}^*$ and $f_a : \Sigma^2 \to \mathbb{N}^*$, find a common arc-annotated subsequence (T, Q) that maximizes the following score function: $\sum_{c \in T} f_b(c) + \sum_{(c_1,c_2) \in Q} f_a(c_1, c_2)$. In other words, the Mapcs problem aims at finding a common subsequence similarly to the Lapcs problem, except that the score takes into account both the number of bases and arcs of the common subsequence. We will first focus on two possible extensions of the Mapcs problem, where either f_a or f_b always returns 0 ; for both problems, we state their computational complexity, depending on the form of the input structures (Chain, Nest, Cros or Unlim). Then, we fully investigate the computational complexity of the Mapcs problem itself.

3.1 Extending Mapcs to the case where f_a or f_b always returns zero

Before going into details concerning the computational complexity of the Mapcs problem, we would like to point out two closely related problems, that can be

seen as extensions of MAPCS, if we allow function f_a or f_b to be ignored by always returning zero. In particular, we will see that those two problems are in fact closely related to, respectively, the LAPCS and the MLG problems.

If $f_a : \Sigma^2 \to 0$ and $f_b : \Sigma \to \mathbb{N}^*$, then the corresponding problem is equivalent to the LAPCS problem, whose complexity has been extensively studied, and is summarized in Table 1. If $f_a : \Sigma^2 \to \mathbb{N}^*$ and $f_b : \Sigma \to 0$, then the corresponding problem, that we will call MAPCS*, is closely related to the MLG problem, where the vertices of the linear graphs are now labeled with a letter from the alphabet $\Sigma = \{\mathsf{A}, \mathsf{U}, \mathsf{G}, \mathsf{C}\}$ and where edges only exist between two vertices labeled A and U (resp. C and G). The computational complexity of the MAPCS* problem, depending of the types of the input sequences, is summarized in the following propositions (some proofs are omitted due to space constraints).

Proposition 1. *The* MAPCS*(CHAIN,CHAIN) *problem can be solved in* $O(nm)$ *time, where* n *and* m *are the number of bases of each sequence.*

Proposition 2. *The* MAPCS*(NEST,NEST) *(resp.* MAPCS*(NEST,CHAIN)*) problem can be solved in* $O(n^2m^2)$ *(resp.* $O(nm^2)$*) time, where* n *is the number of bases of the* NEST *sequence and* m *is the number of bases of the other sequence.*

Proof. Note that we can pre-process both sequences so that they do not contain free bases. Clearly, this pre-process does not change the result (sine f_b always returns 0) neither the arc structure (i.e., CHAIN or NESTED), and can be carried out in linear time. Let us first focus on the MAPCS*(NEST,NEST) problem. The proof is directly derived from the work of Lozano and Valiente [18], in which a dynamic programming algorithm was given in order to obtain a maximum common embedded subtree of two trees. Briefly stated, since the two input sequences of our problem are of NESTED type, there is a natural representation of such sequences as trees: each vertex of the tree represents an arc, and an edge joins a father f to its son s if the arc represented by s is directly nested in the arc represented by f. The dynamic programming algorithm from [18] computes the maximum common subtree of two trees, and thus, if adapted to the MAPCS* problem in order for the score function to take function f_a into account, would output a common subsequence having the maximum score. The only thing missing in the algorithm from [18] is that it could match any two vertices of the tree (i.e., in our case, any two arcs). This means that, for instance, an arc A–U could be matched to an arc G–C. However, this can be easily fixed in the algorithm by adding some conditions, in the dynamic programming recursive formula, that will ensure that only similar arcs can be matched. It is possible to show that, using this algorithm to solve MAPCS*(NEST,CHAIN), the size of the dynamic programming table becomes $O(nm^2)$ and thus the time complexity follows. □

Proposition 3. MAPCS*(UNLIM,NEST) *can be solved in* $O(n^4 \log^3 n)$*, where* n *is the maximal number of bases in an input arc-annotated sequence.*

Proof. The proof relies on the same argument as in proof of Proposition 2, using a result from the MLG problem [16]. The problem is to extract from two linear

graphs of UNLIMITED type a NESTED structure having the maximum number of arcs. In our context, since one of the two input sequences is of NESTED type, we can ensure that the result will be NESTED. Moreover, in both problems, one wishes to maximize the number of arcs of the common structure. Thus, the algorithm from [16] could be applied to the MAPCS*(UNLIM,NEST) problem, except that linear graphs are unlabeled graphs, which means that any arc can be matched to any other arc in MLG. However, as for Proposition 2 above, this problem can be easily fixed: indeed, the algorithm from [16] starts by constructing trapezoids that correspond to all possible arc matchings, and then finds the maximum set of trapezoids that are either pairwise included, or totally disjoint, in order to end up with a NESTED structure of maximum size. In our case, it suffices to change the first step of the algorithm by constructing the trapezoids that correspond to "valid" matchings, that is arcs whose bases have the same labels in the same order. Hence the result. □

Theorem 1. *The* MAPCS*(CROS,CROS) *problem is* **NP***-complete.*

Proof. We consider here the natural decision version of the MAPCS* problem, in the specific case where f_a always returns the same constant. Clearly, the problem is in **NP**. In order to prove that it is **NP**-complete, we propose a polynomial reduction from the MAX-CLIQUE problem, defined as follows: Given a graph G and an integer k, is there a clique of cardinality greater than or equal to k in G ? The idea here is to construct, from any graph $G = (V, E)$, two arc-annotated sequences (S_1, P_1) and (S_2, P_2), in which, informally, (S_1, P_1) will represent G and (S_2, P_2) will represent a clique of cardinality k that we wish to find in G. Then, we will prove that finding a common subsequence of maximum score between (S_1, P_1) and (S_2, P_2) is equivalent to finding a clique of size k in G. Now, we formally describe the construction of the two arc-annotated sequences (S_1, P_1) and (S_2, P_2). We first describe S_1: $S_1 = S_1^1 S_1^2 ... S_1^n$, with $S_1^i = \mathsf{A}(\mathsf{CG})^n\mathsf{U}\ \forall i \in \{1, 2, \ldots, n\}$. Now, P_1 is defined as follows: first, within each S_1^i, there is an arc between bases A and U. Then, for each edge (v_i, v_j) in G, we connect the j-th base C (resp. G) of S_1^i to the i-th base G (resp. C) of S_1^j. Let us now describe the construction of (S_2, P_2). We start with S_2: $S_2 = X_1\ \mathsf{A}\ Y_1\ \mathsf{U}\ X_2\ \mathsf{A}\ Y_2\ \mathsf{U} \ldots X_k\ \mathsf{A}\ Y_k\ \mathsf{U}\ X_{k+1}$, where (i) $X_i = (\mathsf{AU})^{n-k}$, and (ii) $Y_i = T_1\ T_2 \ldots T_k$, with $T_j = \mathsf{CG}$ for all $1 \leq j \leq k$. Thus, S_2 can be defined as $S_2 = ((\mathsf{AU})^{n-k}\mathsf{A}(\mathsf{CG})^k\mathsf{U})^k(\mathsf{AU})^{n-k}$. The arcs of P_2 are as follows: each base A is connected by an arc to the first base U on its right. Moreover, for any $1 \leq i < j \leq k$, there is an arc between base C (resp. base G) in T_j of Y_i and base G (resp. base C) in T_i of Y_j. 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 CROSSING type. An illustration of such a construction is given in Figure 1, where $n = 4$ and $k = 3$.

The proof relies on the following equivalence: there exists a clique of cardinality greater than or equal to k in G iff there exists an arc-preserving common subsequence (T, Q) of (S_1, P_1) and (S_2, P_2) whose score is greater than or equal to $(n + k(k-1))f_a$. It is omitted here due to space constraints. □

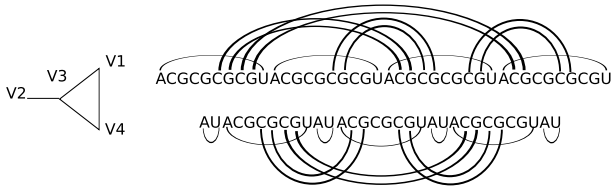


Fig. 1. Illustration of the construction where $n = 4$ and $k = 3$.

$A \times B$	Chain	Nested		Crossing			Unlimited			
	Chain	Chain	Nest	Chain	Nest	Cros	Chain	Nest	Cros	Unlim
Mapcs*	$O(nm)$ Prop. 1	$O(n^2m)$ Prop. 2	$O(n^2m^2)$ Prop. 2	$O(n^4 \log^3 n)$ Prop. 3	**NP**C Thm 1		$O(n^4 \log^3 n)$ Prop. 3		**NP**C Thm 1	

Table 2. Complexity of Mapcs* (n and m are the lengths of the input sequences with $m \leq n$)

Those results show that, though the problem is different, the Mapcs* and Mlg problems are sufficiently close to admit the same computational complexity in each case (cf. Table 2).

3.2 The Mapcs Problem

We now turn to the Mapcs problem in itself, where neither f_a nor f_b returns zero. We first begin by a property relating Mapcs to Lapcs in a very specific case. This property will help us derive some computational complexity results.

Property 1. Given two arc-annotated sequences (S_1, P_1) and (S_2, P_2) s.t. at least one of them is of Plain type, then Lapcs and Mapcs have the same complexity.

Proof. Since at least one of P_1 and P_2 is the empty set, there will be no common arc in any common subsequence (T, Q) of (S_1, P_1) and (S_2, P_2). Therefore, the score of (T, Q) will only depend on f_b for the Mapcs problem. Hence, the polynomial results for Lapcs can be adapted to Mapcs by changing the dynamic programming formulas to take f_a into account, while the **NP**-completeness results for Lapcs actually are valid for Mapcs, when f_b always returns 1. □

Thanks to Property 1, and since Lapcs(Cros,Plain) is **NP**-complete, so does Mapcs(Cros,Plain). This also implies that Mapcs(X,Y) is **NP**-complete, for any $X \in \{\text{Cros}, \text{Unlim}\}$ and Y s.t. $Y \subseteq X$.

Proposition 4. *The* Mapcs*(*Chain,Chain*) (resp.* Mapcs*(*Nest,Chain*)) problem is solvable in $O(nm)$ (resp. $O(nm^3)$) time, where n (resp. m) is the number of bases of the* Nest *(resp.* Chain*) type sequence.*

Proof. Again, the idea here is to adapt an already existing algorithm, more precisely the dynamic programming algorithm designed by Lin et al. in [15] that was

used to show that both the LAPCS(CHAIN,CHAIN) and the LAPCS(NEST,CHAIN) problems are polynomial-time solvable. Indeed, in [15], Lin et al. have designed a dynamic programming algorithm relying on a score function χ which may be refined to take into account the fact that the considered bases are incident or not to an arc ; in our case, it suffices to adapt χ in order to use either f_b or f_a in the computation of the score. □

Theorem 2. *The* MAPCS*(*NEST,NEST*) problem is* **NP***-complete.*

We consider here the natural decision version of MAPCS. We propose a reduction from the MIS-3P problem which is known to be **NP**-complete [4]. The MIS-3P problem consists in, given a cubic planar bridgeless connected graph $G = (V, E)$ and an integer k, finding an independent set of size k in G. Recall, that a graph $G = (V, E)$ is said to be *cubic planar bridgeless connected* if any vertex of V has degree three (cubic), G can be drawn in the plane in such a way that no two edges of E cross (planar), and there are at least two edge-disjoint paths connecting any pair of vertices of G (bridgeless connected). As in [6], the proof is a two-step procedure: we first compute a 2-page book embedding of the input graph, and next transform each page into an RNA arc-annotated sequence.

A *2-page book embedding* of a graph G is a linear ordering of the vertices of G along a line together with an assignment of the edges of G to the two half-planes delimited by the line – called the *pages* – such that no two edges assigned to the same page cross (they may, however, share a vertex). For convenience, we will refer to the page above (resp. below) the line as the *top-page* (resp. *bottom-page*). A *2-page s-embedding* will denote a 2-page book embedding where, in each page, every vertex has degree at least one.

Theorem 3 ([3]). *There exists a polynomial-time algorithm that computes a 2-page s-embedding of any cubic planar bridgeless connected graph.*

According to the above theorem, we thus first compute in polynomial-time a 2-page s-embedding of our input graph $G = (V, E)$, and we write $V = (v_1, v_2, \ldots, v_n)$ for the vertices of G according to the linear ordering induced by the 2-page s-embedding. The corresponding NESTED type arc-annotated sequences (S_1, P_1) and (S_2, P_2) are defined as follows: $S_1 = X\ S_1^1\ X\ S_1^2 \ldots X\ S_1^n\ X$, $S_2 = X\ S_2^1\ X\ S_2^2 \ldots X\ S_2^n\ X$ where (i) $X = \mathsf{C}^{10n}\mathsf{G}^{10n}$ and there is an arc between the i^{th} and the $20n - (i+1)^{th}$ base of X, $1 \leq i \leq 10n$ (all bases of X are thus paired in a nested way and we call all these arcs the *separating arcs* of the two arc-annotated sequences), and (ii) for each $1 \leq i \leq n$, S_1^i (resp. S_2^i) is a segment AAAUUUAUAUA if vertex v_i has degree 2 in the top-page (resp. bottom-page), and S_1^i (resp. S_2^i) is a segment UAUAUAAAAUU otherwise ; moreover, in any segment AAAUUUAUAUA there is an arc between the 1^{st} (resp. 8^{th}) and the 5^{th} (resp. 9^{th}) base, and in any segment UAUAUAAAAUU there is an arc between the 3^{rd} (resp. 7^{th}) and the 4^{th} (resp. 11^{th}) base ; we call all these arcs the *intra-segment arcs* of the two arc-annotated sequences.

What is left is to add the edges of the input graph G into our construction. For each $(v_i, v_j) \in E$, $i < j$, of the top-page we create, in P_1, an arc a_1 linking

a base U of S_1^i and a base A of S_1^j and an arc a_2, nested in a_1, linking the base A (resp. U) directly to the right (resp. left) of the base U (resp. A) of a_1. We proceed in a similar way for the bottom-page by adding, for each edge in that page, two arcs in P_2. Moreover, we impose that when a vertex v_i has degree 1 in the top-page (resp. bottom-page), the two corresponding arcs in P_1 (resp. P_2) are incident to the two leftmost free bases A and U of the segment S_1^i (resp. S_2^i), and to the four rightmost free bases A and U otherwise. We call all these arcs the *inter-segment arcs* of the arc-annotated sequences. It is easy to check that the above construction results in two NESTED type RNA arc-annotated sequences (S_1, P_1) and (S_2, P_2). An example of such a construction is given in Figure 2. The size of the sequences is clearly polynomial in n. Indeed, both S_1 and S_2 have length $20n^2 + 31n$. To complete the construction, we suppose that f_a (resp. f_b) always returns the same constant. We claim that there exists an independent set of size k in G iff there exists an alignment of (S_1, P_1) and (S_2, P_2) with total score $20n(n+1)f_b + 10n(n+1)f_a + (f_a + 6f_b)k + \max\{6f_b, 5f_b + f_a\}(n-k)$.

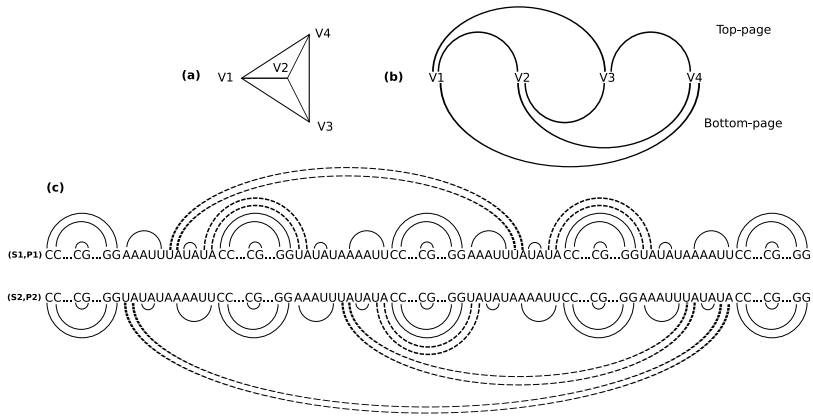


Fig. 2. (a) a cubic planar bridgeless connected graph G of order 4, (b) a 2-page s-embedding of G and, (c) the corresponding NESTED type arc-annotated sequences.

The proof lies on several properties of an optimal alignment of the sequences (S_1, P_1) and (S_2, P_2) (Lemmas 1 to 4, whose proofs are omitted here). We call an alignment of (S_1, P_1) and (S_2, P_2) *canonical* if, for each $1 \leq i \leq n+1$, the i^{th} segment X of (S_1, P_1) is perfectly aligned to the i^{th} segment X of (S_2, P_2).

Lemma 1. *Any optimal alignment of (S_1, P_1) and (S_2, P_2) is canonical.*

Lemma 2. *In any optimal alignment of (S_1, P_1) and (S_2, P_2), no inter-segment arc is conserved.*

We now consider the local alignment of two corresponding segments S_1^i and S_2^i, $1 \leq i \leq n$. Clearly, S_1^i = AAAUUUAUAUA and S_2^i = UAUAUAAAAUU, or

S_1^i = UAUAUAAAAUU and S_2^i = AAAUUUAUAUA. A simple calculation shows that the optimal alignment of S_1^i and S_2^i is of length 6 and preserves only one arc. Such an optimal alignment is obtained by deleting AAAUU in both S_1^i and S_2^i and has score $6f_b + f_a$. We refer to such an optimal alignment as an *optimal local alignment*. Furthermore, any non-optimal alignment of S_1^i and S_2^i results in a score at most $\max\{6f_b, 5f_b + f_a\}$.

Lemma 3. *The total score of any optimal alignment of* (S_1, P_1) *and* (S_2, P_2) *is* $20n(n+1)f_b + 10n(n+1)f_a + (f_a + 6f_b)k + \max\{6f_b, 5f_b + f_a\}(n-k)$*, where* k *is the number of optimal local alignments.*

Lemma 4. *There exists an independent set of size* k *in* G *iff there exists an alignment of* (S_1, P_1) *and* (S_2, P_2) *with total score* $20n(n+1)f_b + 10n(n+1)f_a + (f_a + 6f_b)k + \max\{6f_b, 5f_b + f_a\}(n-k)$.

All the results concerning the MAPCS problem are summarized in Table 3.

$A \times B$	CHAIN	NESTED		CROSSING			UNLIMITED			
	CHAIN	CHAIN	NEST	CHAIN	NEST	CROS	CHAIN	NEST	CROS	UNLIM
MAPCS	$O(nm)$ Prop. 4	$O(nm^3)$ Prop. 4	**NPC** Theorem 2 and Property 1							

Table 3. Complexity of MAPCS (n and m are the lengths of the input sequences with $m \leq n$).

4 Conclusion

In this paper, we have introduced a new model for comparing two RNA structures using arc-annotated sequences – the MAPCS problem – which can be considered as an intermediate problem between LAPCS and EDIT. Indeed, it is less intricate than the EDIT problem, in the sense that some (but not all) of the edit operations have a specific cost. Moreover, it is a natural extension of the LAPCS problem in which a (non zero) score is given to any arc in the common subsequence, in addition to the score already given to its bases in the LAPCS problem. This new model makes RNA motif extraction biologically more relevant than the LAPCS problem, since one can intuitively think of a common RNA subsequence containing many arcs as more "reliable" than one containing as many bases, but less arcs. We have fully studied the computational complexity of the MAPCS problem. We have also found interesting to "locate" this problem compared to other well-known problems for RNA structure comparison, such as EDIT, LAPCS and MLG ; this allows us to show that MAPCS is not a mere extension of LAPCS, but somehow lies in the middle of those three problems. This new paradigm sheds light on a new aspect of the hardness of the RNA structures comparison problem ; namely, the hardness is not necessarily fully correlated to the complexity of the allowed edit operations.

References

1. J. Alber, J. Gramm, J. Guo, and R. Niedermeier. Computing the similarity of two sequences with nested arc annotations. *Th. Comp. Science*, 312(2):337–358, 2004.
2. V. Bafna, S. Muthukrishnan, and R. Ravi. Computing similarity between RNA strings. In *Comb. Patt. Matching (CPM)*, volume 937 of *LNCS*, pages 1–16, 1995.
3. F. Bernhart and P.C. Kainen. The book thickness of a graph. *Journal of Combinatorial Theory, Series B*, 27(3):320–331, 1979.
4. T. Biedl, G. Kant, and M. Kaufmann. On triangulating planar graphs under the four-connectivity constraint. *Algorithmica*, 19:427–446, 1997.
5. G. Blin, G. Fertin, R. Rizzi, and S. Vialette. What makes the arc-preserving subsequence problem hard ? *Trans. on Comp. Systems Biology*, 2:1–36, 2005.
6. G. Blin, G. Fertin, I. Rusu, and C. Sinoquet. Extending the hardness of RNA secondary structure comparison. In *intErnational Symposium on Combinatorics, Algorithms, Probabilistic and Experimental methodologies (ESCAPE)*, LNCS. Springer, 2007. To appear.
7. G. Blin and H. Touzet. How to compare arc-annotated sequences: The alignment hierarchy. In *String Processing and Information Retrieval (SPIRE)*, volume 4209 of *LNCS*, pages 291–303. Springer, 2006.
8. P. Bose, J.F.Buss, and A. Lubiw. Pattern matching for permutations. *Information Processing Letters*, 65(5):277–283, 1998.
9. E. Davydov and S. Batzoglou. A computational model for RNA multiple structural alignment. *Theoretical Computer Science*, 368(3):205–216, 2006.
10. P. Evans. *Algorithms and Complexity for Annotated Sequences Analysis*. PhD thesis, University of Victoria, 1999.
11. P.A. Evans. Finding common RNA pseudoknot structures in polynomial time. In *Comb. Pattern Matching (CPM)*, volume 4009 of *LNCS*, pages 223–232, 2006.
12. D. Goldman, S. Istrail, and C.H. Papadimitriou. Algorithmic aspects of protein structure similarity. In *Found. of Comp. Science (FOCS)*, pages 512–522, 1999.
13. J. Gramm, J. Guo, and R. Niedermeier. Pattern matching for arc-annotated sequences. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 2556 of *LNCS*, pages 182–193. Springer, 2002.
14. D. S. Hirschberg. *The longest common subsequence problem*. PhD thesis, Princeton University, 1975.
15. T. Jiang, G.-H. Lin, B. Ma, and K. Zhang. The longest common subsequence problem for arc-annotated sequences. In *Combinatorial Pattern Matching (CPM)*, volume 1848 of *LNCS*, pages 154–165. Springer, 2000.
16. M. Kubica, R. Rizzi, S. Vialette, and T. Walen. Approximation of RNA multiple structural alignment. In *Combinatorial Pattern Matching (CPM)*, volume 4009 of *LNCS*, pages 211–222. Springer, 2006.
17. G.H. Lin, B. Ma, and K. Zhang. Edit distance between two RNA structures. In *Int. Conf. on Computational Biology (RECOMB)*, pages 211–220. ACM Press, 2001.
18. A. Lozano and G. Valiente. *String Algorithmics*, chapter 7 - On the maximum common embedded subtree problem for ordered trees, pages 155–169. King's College London Publications, 2004.
19. S. Vialette. On the computational complexity of 2-interval pattern matching problems. *Theoretical Computer Science*, 312(2-3):223–249, 2004.