On realizing shapes in the theory of RNA neutral networks

Peter Clote [a] Leszek Gąsieniec [b,3,*] Roman Kolpakov [b,2]
Evangelos Kranakis [c,1] Danny Krizanc [d]

[a]*Department of Biology, Higgins Hall, Boston College, Chestnut Hill, MA 02467*

[b]*Department of Computer Science, The University of Liverpool, Chadwick Building, Peach Street, Liverpool L69 7ZF, UK*

[c]*School of Computer Science, Carleton University, Ottawa, Canada*

[d]*Department of Mathematics and Computer Science, Wesleyan University*

Abstract

It is known [17] that for any two secondary structures S, S' there exists an RNA sequence compatible with both, and that this result does not extend to more than two secondary structures. Indeed, a simple formula for the number of RNA sequences compatible with secondary structures S, S' plays a role in the algorithms of Flamm et al. [6] and of Abfalter et al. [1] to design an RNA *switch*. Here we show that a natural extension of this problem is NP-complete. Unless $P = NP$, there is no polynomial time algorithm, which when given secondary structures $S_1, \ldots, S_k$, for $k \geq 4$, determines the least number of positions, such that after removal of all base pairs incident to these positions there exists an RNA nucleotide sequence compatible with the given secondary structures. We also consider a restricted version of this problem with a "fixed maximum" number of possible stars and show that it has a simple polynomial time solution.

Key words: RNA conformational switch, NP-completeness, neutral network, Nussinov-Jacobson energy model

[*] Send correspondence to Leszek Gąsieniec, `leszek@csc.liv.ac.uk`, tel: +44 151 794 3686, fax: +44 151 794 3715, `http://www.csc.liv.ac.uk/~leszek/`

[1] Research supported in part by the NSERC (Natural Sciences and Engineering Research Council of Canada) and MITACS (Mathematics of Information Technology and Complex Systems) grants.

[2] Research supported in part by the Russian Foundation for Fundamental Research (Grant 02-01-00985) and by the EPSRC research grant GR/R84917/01

[3] Research supported in part by the EPSRC research grant GR/R84917/01

1 Introduction

As determined in [11, 10], the 39 nt. spliced leader RNA (SL RNA) in trypanosomatid protozoa assumes two alternate secondary structure conformations. In particular, [10] showed that *T. brucei* SL RNA can be induced *in vivo* to switch from one conformation to another, using complementary $2'$-O-methyl RNA oligonucleotides. The presence of these two conformational forms of SL RNA is critical for its ability to be spliced onto the $5'$ end of pre-mRNA, thus forming mature mRNA in trypanosomatid protozoa.

An *RNA conformational switch* is an RNA nucleotide sequence having two secondary structures S_0, S, where S_0 denotes the *minimum free energy (mfe) structure* (also here called *native state*), such that S is a distinct metastable secondary structure, separated by an activation energy barrier from the native state S_0, yet whose free energy is close to that of S_0. Increasingly, RNA switches have been found to play an important biological role; see, for instance, [9, 3, 13, 15, 18]. Due to the growing importance of recognizing RNA switches, Voss et al. [21] developed an algorithm called `paRNAss`, which predicts conformational switching for a given RNA sequence. `paRNAss` uses the program `RNAsubopt` from Vienna RNA package [5, 12] to list low energy secondary structures for a given RNA sequence; by clustering these structures, according to different metrics on the space of secondary structures, the program outputs a distance plot. If clusters of low energy secondary structures appear to be distinct and clearly separated from each other, then `paRNAss` predicts a conformational switch for the given RNA sequence.

From the viewpoint of molecular evolution theory, over the past decade P. Schuster and co-workers Stadler, Reidys, Hofacker, Bornberg-Bauer, Flamm, etc. [16, 19, 20, 2] have studied properties of *neutral networks* within the context of protein and especially RNA structures. In the following, we give a brief summary of their approach.

Let n be a fixed, but arbitrary positive integer. A length n RNA nucleotide sequence is considered as a word in the space $C_n = \{A, C, G, U\}^n$. If $a = a_1, \ldots, a_n \in C_n$, then a secondary structure S for a is a collection of ordered pairs (i, j), where $1 \leq i < j \leq n$, which satisfies the following conditions.

(1) If $(i, j) \in S$, then a_i, a_j form a Watson-Crick or GU wobble base pair; i.e., a_i, a_j is among AU,UA,GC,CG,GU,UG.
(2) If $(i, j), (k, \ell) \in S$, then it is not the case that $i < k < j < \ell$; i.e., pseudoknots are disallowed.
(3) If $(i, j), (k, \ell) \in S$, then $i \in \{k, \ell\}$ implies that $i = k$ and $j = \ell$; i.e., there are no base triples.
(4) If $(i, j) \in S$, then $j > i+\theta$, where θ is fixed and usually taken to be 3; i.e., due to molecular rigidity, every loop region must have at least θ unpaired bases.

Let S_n denote the space of all RNA secondary structures of length n. Secondary structures can clearly be identified with balanced parenthesis expressions containing *dots*, where a dot corresponds to an unpaired nucleotide position, and a matching parenthesis which opens at nucleotide position i and closes at nucleotide position j corresponds to base pair (i, j).

Associating each sequence $s \in C_n$ with its *native state* secondary structure S[4] leads to a map $F : C_n \to S_n$ from *genotypes* to *phenotypes*. This map is clearly many-to-one, since *sequence space*[5] C_n has size 4^n, while *shape space* S_n has size at most 3^n.

For any fixed secondary structure S of length n, consider the inverse image $F^{-1}(S)$ consisting of those sequences $s \in S_n$ whose mfe secondary structure is S. Define an undirected graph $G = (V, E)$, where vertex set $V = F^{-1}(S)$, and where edge set E consists of undirected edges $\{s, s'\}$ between vertices s, s' which satisfy the following condition: either s, s' differ by only one nucleotide, which is not involved in a base pair of S, or s, s' differ by at most 2 nucleotides at positions i, j where (i, j) is a base pair of secondary structure S. Intuitively, there is an edge between s, s' when they differ by one *neutral* or *compensatory mutation*. Finally, a *neutral network* is a *connected component* of graph G; i.e., the collection of sequences $s \in C_n$ having a given secondary structure S, all of which are reachable by a sequence of neutral or compensatory mutations. In a series of papers [16, 19, 20, 2, 7], P. Schuster and co-workers investigate mathematical properties of neutral networks.

A particular problem of interest from the perspective of molecular evolution theory is to determine, given different secondary structures S, S', the minimum number $N(S, S')$ of pointwise mutations, such that there exists a sequence $s \in F^{-1}(S)$ which can be transformed into a sequence $s' \in F^{-1}(S')$. Call this the *Covering Problem*. While a rigorous solution of this problem remains elusive, we consider a related problem in this paper.

Define the binary relation $R \subseteq C_n \times S_n$ by

$$R = \{(s, S) : S \text{ is a secondary structure for } s\}.$$

In contrast to the mapping $F : C_n \to S_n$, which associates to each RNA sequence s of length n a unique native state secondary structure S, the relation $R(s, S)$ holds whenever S is a secondary structure which is *compatible* with s.

[4] Depending on the energy model used, there may be more than one minimum free energy (mfe) secondary structure for a given sequence. In such situations, in order to develop the mathematical theory of neutral networks, "native state" can be identified the lexicographically first mfe structure, or alternatively with that structure output by an algorithm such as Zuker's `mfold` [22, 14] or Vienna RNA Package `RNAfold` [12].

[5] Sequence space is also called *configuration* space.

Results in this paper are motivated by the following question. *Given a finite set* $S_1, \ldots, S_k$ *of secondary structures, under what conditions does there exist a single RNA sequence which can realize each of the given structures?*

Given secondary structures $S_1, \ldots, S_k$, we study the problem of determining the minimum number $N(S_1, \ldots, S_k)$ of positions, for which after removal of all base pairs incident to these positions, there exists an RNA sequence $s \in C_n$ which is compatible with each of the structures S_i. Call this the *Min* $*$ *Realizability Problem* (M$*$RP). In this paper, we give an $O(nk)$ algorithm to determine whether $N(S_1, \ldots, S_k) = 0$, and we prove that the Min $*$ Realizability Problem is NP-complete for $k \geq 4$. Our work extends that of Reidys et al. [17] (Intersection Theorem), Flamm et al. [6] (Generalized Intersection Theorem), and Abfalter et al. [1], where results equivalent to our Lemma 8 and Theorem 10 were first proved. Of particular note is the interesting use of the Generalized Intersection Theorem in the latter two papers, in order to sample in a uniform manner from the set of RNA nucleotide sequences which are compatible with two given secondary structures. Such sampling, along with an adaptive walk and a dynamic programming method using *ear decompositions,* allows Abfalter et al. [1] to design RNA conformational switches. In a similar manner, the work of the current paper lays the groundwork for the design of RNA nucleotide sequences compatible with secondary structures admitting pseudoknots, base triples and non-canonical base pairing.

In Section 2, we give the main definitions of shape, binary realizability problem, min $*$ realizability problem and we relate the latter problem with the vertex cover set problem, known to be NP-complete. In Section 3, we consider a bonded version of the min $*$ realizability problem, which we show can be solved in polynomial time.

2 Realizing Shapes

In this section, we define the notions of *shape*, $0, 1$-*labeling*, $*$-labeling and formulate the *Realizability Problem*, the *Binary Realizability Problem,* and the *Min* $*$ *Realizability Problem*. The realizability problem is to construct an RNA nucleotide sequence which is compatible with a given set of secondary structures.

Following is an outline of this section. Lemma 8 and Theorem 10, which solve the realizability problem, were first proved by Reidys et al. [17] (Intersection Theorem), Flamm et al. [6] (Generalized Intersection Theorem), and Abfalter et al. [1], though we were initially unaware of their prior work. The min $*$ realizability problem is to determine the minimum number of nucleotide positions, such that there exists an RNA nucleotide sequence compatible with a given set of secondary structures after edges incident to those positions have been removed. By a series of reductions, we show that the vertex cover set problem can be reduced to the min

$*$ realizability problem; i.e. given an instance of the vertex cover set of valence 3 (i.e. an undirected graph G, whose vertices have degree at most 3), we define an associated set S_1, S_2, S_3, S_4 of secondary structures such that there is a vertex cover of G of size r if and r' many $*$s suffice to subsequently realize S_1, S_2, S_3, S_4. (Here the association of r' and S_1, S_2, S_3, S_4 is given by a polynomial time computable function.) Since the vertex cover set problem, even for valence 3 graphs is NP-complete, it follows that the min $*$ realizability problem is as well. In the sequel, we make these ideas precise and provide details of our results.

Shapes arise from secondary structures when we remove the labels.

Definition 1 *A shape S of size n is a graph with set $V_n = \{v_1, v_2, \ldots, v_n\}$ of nodes and set of independent edges such that for any two edges $\{v_i, v_j\}, \{v_s, v_t\}$, where $i < j$ and $s < t$, it is not the case that $i < s < j < t$.*

The nodes in S are called *bases* and the edges in S are called *basepairs*. Further, we assume also that any shapes of the same size share the same set V_n of nodes and we denote by $E(S)$ the set of basepairs of the shape S.

Definition 2 *The* shape *space, denoted* SHAPE_n*, consists of all shapes of size n.*

Given k shapes $S_1, S_2, \ldots, S_k$ in the shape space SHAPE_n define the shape graph $G(S_1, S_2, \ldots, S_k)$ by combining the k shapes. More formally, define.

Definition 3 *$G(S_1, S_2, \ldots, S_k)$ is the graph on the set V_n of vertices and whose set of edges is the union of the sets of edges $E(S_i),\ i = 1, 2, \ldots, k$.*

For any graph G we denote the number of vertices of G by $|G|$ and the number of edges of G by $\|G\|$. We consider the RNA alphabet $\{A, U, C, G\}$. In this alphabet the letter A is complementary with the letter U, and the letter C is complementary with the letter G.

Definition 4 *A string s=$s_1 \ldots s_n$ over the alphabet $\{A$,U,C,$G\}$ realizes a shape S of size n if for any basepair $\{v_i, v_j\}$ of S the letters s_i and s_j are complementary with one another.*

We are interested in the following problem.

Problem 5 (Realizability Problem) *For any shapes $S_1, S_2, \ldots, S_k$ of the same size construct a single string realizing all these shapes.*

Definition 6 *A binary string $s = s_1 \ldots s_n$ over the alphabet $\{0, 1\}$ realizes a shape S of size n if $s_i \neq s_j$ for any basepair $\{v_i, v_j\}$ of S.*

Note that a string realizing the shapes $S_1, S_2, \ldots, S_k$ over the four letter alphabet $\{A, U, C, G\}$ exists if and only if there exists a binary string realizing the shapes

$S_1, S_2, \ldots, S_k$. So the Realizability Problem is equivalent to

Problem 7 (Binary Realizability Problem) *For any shapes $S_1, S_2, \ldots, S_k$ of the same size construct a single binary string realizing all these shapes.*

It is easy to see that a binary string $s = s_1 \ldots s_n$ realizes the shapes $S_1, S_2, \ldots, S_k$ if and only if for any edge $\{v_i, v_j\}$ of the graph $G(S_1, S_2, \ldots, S_k)$ we have $s_i \neq s_j$. Note also that realization is possible if and only if $G(S_1, S_2, \ldots, S_k)$ has no odd cycles, i.e., it is a bipartite graph. Using the breadth first traversal of $G(S_1, S_2, \ldots, S_k)$, we can check the existence of an odd cycle in $G(S_1, S_2, \ldots, S_k)$. If there is no such a cycle, using the same search, we can construct the string s in the following way: for each new found vertex v_i we define the symbol s_i to be different from the symbol s_j such that v_j is the already found neighbour of v_i. Note that checking of the existence of s as well as constructing of s requires $O(\|G(S_1, S_2, \ldots, S_k\|)$ time. Thus we have

Lemma 8 *Any shapes $S_1, S_2, \ldots, S_k$ of size n can be realized by a single binary string s if and only if the graph $G(S_1, S_2, \ldots, S_k)$ has no odd cycles. Furthermore, one can check the existence of s and, if s exists, construct s in $O(nk)$ time.*

The Intersection Theorem of Reidys et al. [17] states that for any shapes S, S', there exists an RNA sequence which realizes S, S'. The Generalized Intersection Theorem of Flamm et al. [6] states that there exists an RNA sequence which realizes $S_1, S_2, \ldots, S_k$ if and only if $G(S_1, S_2, \ldots, S_k)$ is a bipartite graph, a condition equivalent to the nonexistence of odd cycles. Being previously unaware of the work of [17, 6], we have independently come upon the same conclusion of those authors in Lemma 8 and Theorem 10.

Definition 9 *A $0,1$-labeling of the shape graph $G(S_1, S_2, \ldots, S_k)$ is a function*

$$\ell : \{v_1, v_2, \ldots, v_n\} \to \{0, 1\},$$

on the vertices of the shape graph in such a way that if $\{v_i, v_j\}$ is an edge of $G(S_1, S_2, \ldots, S_k)$ then

$$\ell(v_i) \neq \ell(v_j).$$

Firstly consider the case of two shapes.

Theorem 10 *Any two shapes S_1, S_2 of size n can be realized by a single binary string which can be constructed in $O(n)$ time.*

Proof (Theorem 10). Let $\{v_1, v_2, \ldots, v_n\}$ be the common set of vertices, and let E_1, E_2 be the corresponding sets of edges of S_1 and S_2, respectively. Consider a connected component C of the graph $G(S_1, S_2)$. Since every node of this graph has degree at most two, C must be either a path or a cycle. Note that in the cycle each basepair contained in one of the shapes is followed by a basepair containing in the

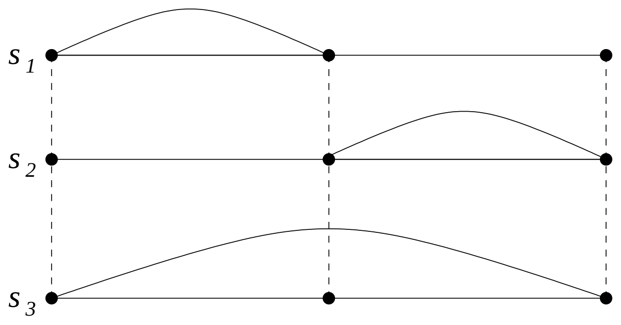


Fig. 1. These three shapes cannot be realized by a single binary string of length three.

other shape. So the cycle has equal numbers of basepairs containing in the shapes S_1 and S_2 respectively. Thus, the graph $G(S_1, S_2)$ cannot have an odd cycle. Hence Theorem 10 follows from Lemma 8. It follows that every connected component is either a path or an even cycle. If $v_{i_1}, v_{i_2}, \ldots, v_{i_s}$ is a connected component of $G(S_1, S_2)$ we define the labeling ℓ in such a way that $\ell(v_{i_r}) \neq \ell(v_{i_{r+1}})$, for $r = 1, 2, \ldots, s-1$. This completes the proof of Theorem 10. □

In general, three or more shapes may not be realizable by a single binary string. An example with three shapes is depicted in Figure 1. So to realize more than two shapes by a single string we need to introduce an extra "don't care" symbol (here we use the symbol $*$) where we allow to match the symbol $*$ with any of 0 and 1 as well as with itself. We can generalize the notion of labeling previously defined to the following more general concept.

Definition 11 *A $*$-labeling of the shape graph $G(S_1, S_2, \ldots, S_k)$ is a labeling of the vertices with 0s, 1s and $*$s, i.e.,*

$$\ell : \{v_1, v_2, \ldots, v_n\} \to \{0, 1, *\},$$

in such a way that if $\{v_i, v_j\}$ is an edge of $G(s_1, s_2, \ldots, s_k)$ such that $\{\ell(v_i), \ell(v_j)\} \subseteq \{0, 1\}$ then

$$\ell(v_i) \neq \ell(v_j).$$

Definition 12 *A binary string $s = s_1 \ldots s_n$ over the alphabet $\{0, 1, *\}$ realizes a shape S of size n if $s_i \neq s_j$ or $s_i = s_j = *$ for any basepair $\{v_i, v_j\}$ of S.*

It is clear that any shape $S \in \text{SHAPE}_n$ is realized by the string $*^n$. So we can state

Problem 13 (Min $*$ Realizability Problem (M$*$RP))
For any shapes $S_1, S_2, \ldots, S_k$ of the same size compute the minimum number of $*$s in a single string over $\{0, 1, *\}$ realizing all these shapes.

Note that M$*$RP can be expressed also as

Problem 14 *For any shapes $S_1, S_2, \ldots, S_k$ of the same size compute the minimum number of vertices that must be removed from the graph $G(S_1, S_2, \ldots, S_k)$ to dis-*

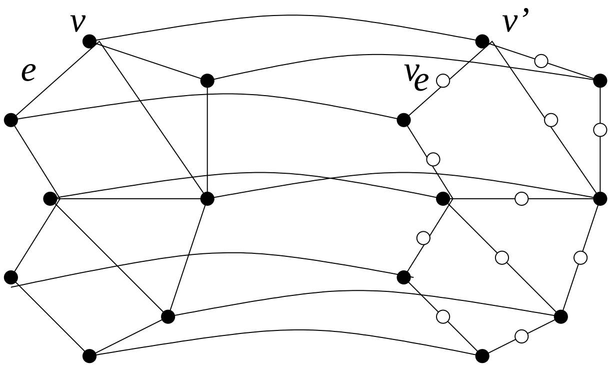


Fig. 2. The graph $\hat{G}$.

connect all odd cycles.

We show that M$*$RP is NP-complete for $k \geq 4$. A subset V' of vertices of a graph G is called a *vertex cover set* if for each edge e of G there is some $v \in V'$ such that $v \in e$. We consider the following problem.

Problem 15 (Vertex Cover Set Problem for Degree D (VCSP[D]))
For any graph G with maximum vertex degree D find the minimum number of vertices in a vertex cover set of G.

It is known (see [8]) that VCSP[3] is NP-complete.

We call a subset V' of vertices of a graph G *feedback vertex set for odd cycles* if V' contains at least one vertex from every **odd** cycle in G.

Problem 16 (Odd Feedback Vertex Set Problem for Degree D (OFVSP[D])) *For any graph G with maximum vertex degree D find the minimum number of vertices in a feedback vertex set for odd cycles of G.*

Theorem 17 *OFVSP[4] is NP-complete.*

Proof (Theorem 17). For any graph $G = (V, E)$ define a new graph $\hat{G} = (\hat{V}, \hat{E})$ as follows (see Figure 2). The set $\hat{V}$ consists of

(1) the set V,
(2) for each $v \in V$ a new vertex v', and
(3) for each edge $e \in E$ a new vertex v_e.

We will denote the set of all new vertices defined in condition 2 by V' and the set of all new vertices defined in condition 3 by V''. The set $\hat{E}$ consists of

(1) the set E,
(2) for each edge $e = \{a, b\} \in E$ two new edges $\{a', v_e\}$ and $\{v_e, b'\}$,
(3) for each $v \in V$ the edge $\{v, v'\}$.

We will denote the set of all edges defined in condition 2 by E'.We prove that the minimum number m of vertices in a vertex cover set for G is equal to the minimum number $\hat{m}$ of vertices in a feedback vertex set for odd cycles for $\hat{G}$. Let U be any vertex cover set for G, and C be any odd cycle in $\hat{G}$. Note that C cannot be entirely contained in the graph $(V' \cup V'', E')$ since all cycles in this graph have even length. So C has to contain at least one edge in the set E. Therefore, C has to contain at least one vertex in U. Thus U is a feedback vertex set for odd cycles for $\hat{G}$. Hence $\hat{m} \leq m$. Now assume that $\hat{m} < m$. Let U' be a feedback vertex set for odd cycles for $\hat{G}$ such that $|U'| = \hat{m}$. If U' contains some vertex v_e of V'' where $e = \{v_1, v_2\}$, then v_e can be replaced by either v_1' or v_2' while preserving U' to be a feedback vertex set for odd cycles for $\hat{G}$. So we can assume that $U' \subseteq V \cup V'$. Let U'' be the subset of V consisting of all vertices v such that at least one of the vertices v, v' is contained in U'. Since $|U''| \leq |U'| < m$, the set U'' cannot be a vertex cover set of G. So there exists an edge $e = \{v_1, v_2\}$ in E such that neither v_1 nor v_2 belongs to U''. Hence, U' does not contain any of vertices v_1, v_2, v_1', v_2'. Thus U' does not contain any of vertices of the odd cycle $v_1 v_2 v_2' v_e v_1' v_1$. The obtained contradiction implies $\hat{m} = m$. Note that the graph $\hat{G}$ can be constructed from G in time linear in the size of G, and the maximal vertex degree of $\hat{G}$ equals to the maximal vertex degree of G plus one. So VCSP[3] is transformed in polynomial time to OFVSP[4]. On the other hand, there exists a simple non-deterministic polynomial time algorithm for solving OFVSP[4]: we guess a subset of vertices in the graph and check in linear time whether after removing this subset the remaining part of the graph contains odd cycles, using the breadth first traversal of this part. $\square$

Let $G = (V, E), G' = (V', E')$ be two graphs. We call the graph G' *homeomorphic extension* of G if G' is obtained from G by replacing some edges (v', v'') of G with chains $ch(v', v'') = (v', u_1)(u_1, u_2) \ldots (u_{k-1}, u_k)(u_k, v'')$ where $u_1, u_2, \ldots, u_k$ are new vertices of degree 2 not yet contained in G. The graph G' is called *odd homeomorphic extension* of G if all chains $ch(v', v'')$ have odd length. For any u_i of $ch(v', v'')$ the nodes v' and v'' are called *the end points* of u_i.

Lemma 18 *If G' is odd homeomorphic extension of G then the minimal feedback vertex sets for odd cycles in G and G' have the same cardinality.*

Proof. Note that there exists one-to-one correspondence between cycles in G and G' such that each odd cycle C in G corresponds in G' to odd cycle containing all vertices of C. So each feedback vertex set for odd cycles in G is a feedback vertex set for odd cycles in G'. Now let A be a minimal feedback vertex set for odd cycles in G'. Note that any vertex in $(V' \setminus V) \cap A$ can be replaced by any end point of this vertex while preserving A to be a feedback vertex set for odd cycles in G'. Thus we can assume that A contains only vertices from G. Therefore, A is also a feedback vertex set for odd cycles in G.

Theorem 19 *Let $k \geq 3$. Then for any graph G of maximum vertex degree k there exist k shapes $S_1, \ldots, S_k$ such that $G(S_1, \ldots, S_k)$ is isomorphic to an odd homeo-*

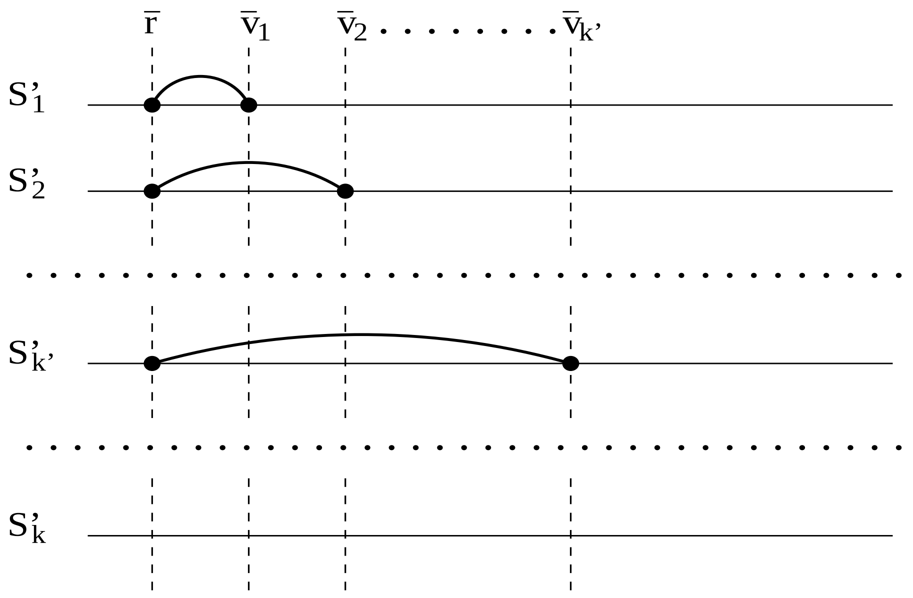


Fig. 3.

morphic extension of G *and* $\|G(S_1,\ldots,S_k)\| = O(\|G\|^2)$.

Proof. Let G be a graph of maximum vertex degree k. Note that each connected component of an odd homeomorphic extension of G is an odd homeomorphic extension of a connected component of G. So without loss of generality we can assume that G is connected. First, we construct shapes $S'_1,\ldots,S'_k$, s.t., $G(S'_1,\ldots,S'_k)$ is isomorphic to an homeomorphic extension of G. For the sake of convenience we extend indices of the shapes to any integer values assuming $S'_i = S'_j$ for any i and j such that $i \equiv j \pmod k$. Where, for any shape S'_i with $i > k$ we will actually mean the shape $S'_{i'}$, s.t., $1 \le i' \le k$ and $i' \equiv i \pmod k$. We call the bases of $G(S'_1,\ldots,S'_k)$ corresponding to vertices of G *original* bases. All other bases of $G(S'_1,\ldots,S'_k)$ are called *additional*. For any vertex v of G we will denote the corresponding to v original base by $\overline{v}$. Consider any rooted DFS spanning tree T of the graph G. We call edges belonging to T *spanning* edges. All other edges of G are called *secondary*. For any subtrees T', T'' of T the subtree is called *proper* subtree of T'' if the root of T' is a child of the root T'' in T. Each spanning edge corresponds to unique basepair in $G(S'_1,\ldots,S'_k)$. These basepairs are constructed inductively from the root to leaves of T in the following way. Let $v_1,\ldots,v_{k'}$, where $k' \le k$, be all children of the root r of T. Then we place the basepairs $\{\overline{r},\overline{v_1}\},\ldots,\{\overline{r},\overline{v_{k'}}\}$ as shown in Figure 3. Assume now that we constructed already basepairs corresponding to edges from some vertex u to all its children in T. Let v' be a child of u, and $v'_1,\ldots,v'_{k'}$, where $k' \le k-1$, be all children of v' in T. Let also $\overline{v''}$ be the next to the right from $\overline{v'}$ in $G(S'_1,\ldots,S'_k)$ original base such that v'' is a child of u (or a vertex which is not a descendant of v' in T if there is no such child), and S'_i be the shape containing the basepair $\{\overline{u},\overline{v'}\}$. Then we place the basepairs $\{\overline{v'},\overline{v'_1}\},\ldots,\{\overline{v'},\overline{v'_{k'}}\}$ in the segment between the bases $\overline{v'}$ and $\overline{v''}$ as shown in Figure 4:

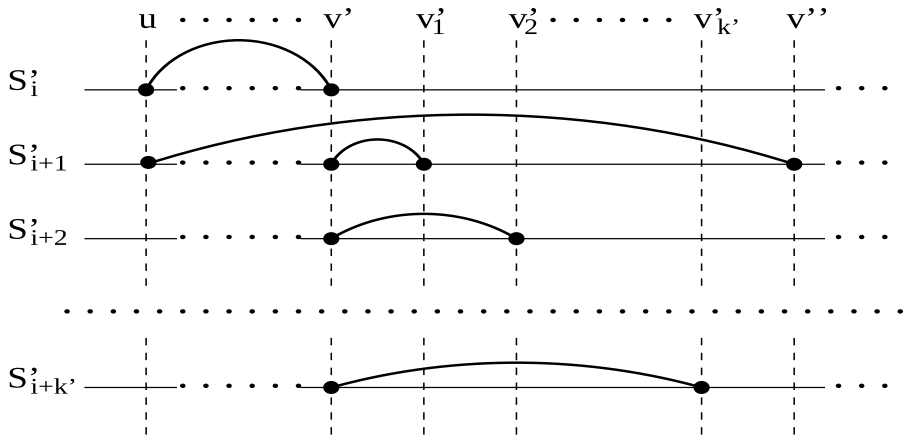


Fig. 4.

Now we have to add to $G(S'_1, \ldots, S'_k)$ chains of basepairs corresponding to secondary edges of G. For these purposes we consider inductively all subtrees of T in the order from trivial subtrees formed by leaves of T to the whole tree T. Let T' be some considered subtree different from a subtree formed by a leave and from the whole tree T. Denote the root of T' by v', the parent of v' in T by u. Let also $\overline{v''}$ be the next to the right from $\overline{v'}$ in $G(S'_1, \ldots, S'_k)$ original base such that $v'' \notin T'$. Since T is a DFS tree, each secondary edge e of G has the following property: one of the ends of e is an ancestor in T for the other end. So we distinguish *ancestor* and *descendant* ends of e. Moreover, if the edge e is incident to a vertex of T' then either both ends of e belong to T' or only descendant end of e belongs to T'. In the first case we say that e *belongs* to T', and in the second case we say that e *originates* in T'. We will produce for T' the following procedure:

(1) for any secondary edge e belonging to T' we construct the chain of basepairs corresponding to e such that this chain is placed completely in the segment between the bases $\overline{v'}$ and $\overline{v''}$;

(2) for any secondary edge e originating in T' we construct a part of the corresponding to e chain of basepairs starting from the base corresponding to the descendant end of e and ending in some additional base called *boundary* base of T' such that this part is placed completely in the segment between the bases $\overline{v'}$ and $\overline{v''}$. Moreover, in this part the basepair incident with the boundary base belongs to the shape containing the basepair $\{\overline{u}, \overline{v'}\}$, and in the segment between the boundary base and the base $\overline{v''}$ there can be only other boundary bases of T' (i.e., all boundary bases of T' are placed in some "buffer" segment containing only these boundary bases (see Figure 5)).

Let $T'_1, T'_2, \ldots, T'_{k'}$ are all proper subtrees of T'. Assume that the required procedure is already produced for these subtrees (if some proper subtree is formed by a leaf of T then we don't produce any procedure for this subtree, but we consider the base corresponding to this leaf as a boundary base in the "buffer" segment for this subtree). So we need to construct for T' all chains corresponding to secondary edges

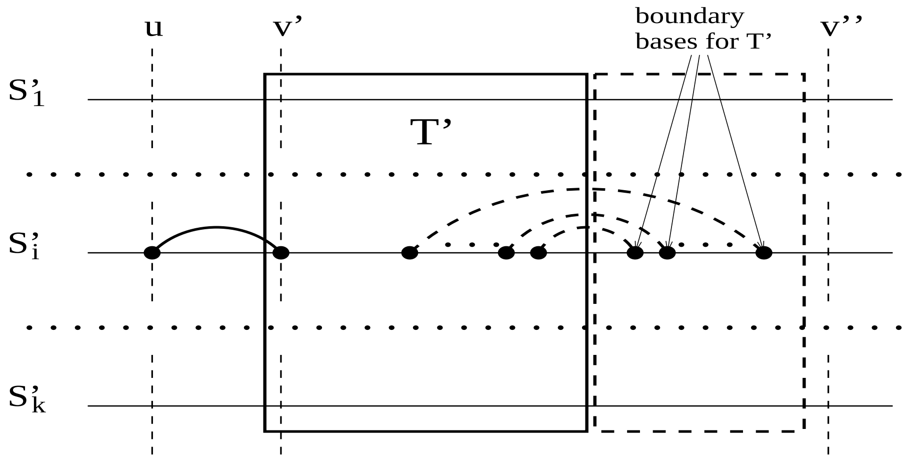
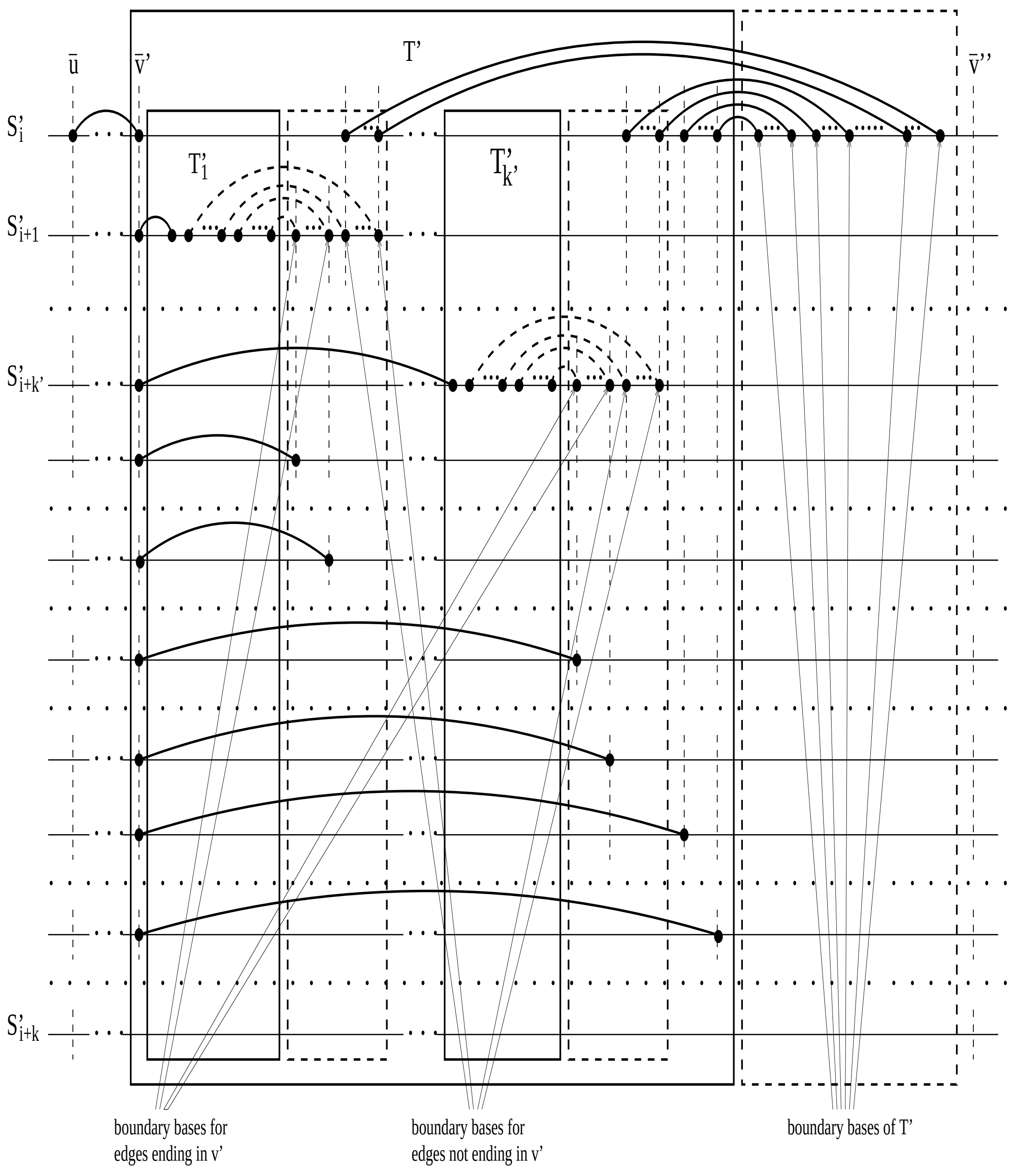


Fig. 5. The segment containing all constructed for T' basepairs except the basepairs incident with boundary bases of T' is outlined by the solid frame, the "buffer" segment for T' is outlined by the dashed frame, and the basepairs incident with boundary bases of T' are depicted by dashed arcs.

belonging to T', but not belonging to any proper subtree of T' and the appropriate parts of all chains corresponding to secondary edges originating in T'. Let e be an edge belonging to T', but not belonging to any proper subtree of T'. Then e originates in some proper subtree T'_j of T', and v' is the ancestor end of e. Since we already produced the required procedure for T'_j, we have constructed a part of the corresponding to e chain starting from the base corresponding to the descendant end of e and ending in the appropriate boundary base of T'_j. Hence, to complete this chain, we have to construct a chain of basepairs linking this boundary base with $\overline{v'}$. Now let e be an edge originating in T'. Then either e originates in some proper subtree T'_j of T' or $\overline{v'}$ is the descendant end of e. In the first case we have to prolong the appropriate part of the chain corresponding to e from the boundary base of T'_j to the boundary base of T'. In the second case we have to construct the appropriate part of the chain corresponding to e from $\overline{v'}$ to the boundary base of T'. Summing up these observations, we conclude that the required procedure for T' consists in the following operations:

(1) for any boundary base of the subtrees $T'_1, \ldots, T'_{k'}$ such that the corresponding edge of G ends in v' we construct a chain of basepairs from this base to $\overline{v'}$;
(2) for any boundary base of the subtrees $T'_1, \ldots, T'_{k'}$ such that the corresponding edge of G doesn't end in v' we construct a chain of basepairs from this base to the appropriate boundary base of T';
(3) for any edge e whose descendant end is v' we construct a chain of basepairs linking $\overline{v'}$ with the corresponding to e boundary base of T'.

The way of constructing all these chains is shown in Figure 6. (In this Figure for each proper subtree we group together for the sake of convenience the boundary bases corresponding to secondary edges ending in v' and the boundary bases corresponding to secondary edges not ending in v' while these bases can be actually



ū
v̄'
T'
v̄''
S'i
T'1
T'k'
S'i+1
S'i+k'
S'i+k
boundary bases for edges ending in v'
boundary bases for edges not ending in v'
boundary bases of T'

Fig. 6.

intermixed among each other. Note that this assumption is not important for our proof.)

Assume now that the required procedure is already created for all proper subtrees $T_1, T_2, \ldots, T_{k'}$ of the tree T. Then we complete the construction of $S'_1, \ldots, S'_k$ as shown in Figure 7.

To obtain the shapes $S_1, \ldots, S_k$ from the shapes $S'_1, \ldots, S'_k$, we have to replace in $G(S'_1, \ldots, S'_k)$ each chain of even number of basepairs corresponding to a secondary edge by a chain of odd number of basepairs. Note that any chain of even number of basepairs has at least two basepairs. Let $\{\alpha_1, \alpha_2\}$, $\{\alpha_2, \alpha_3\}$ be such two

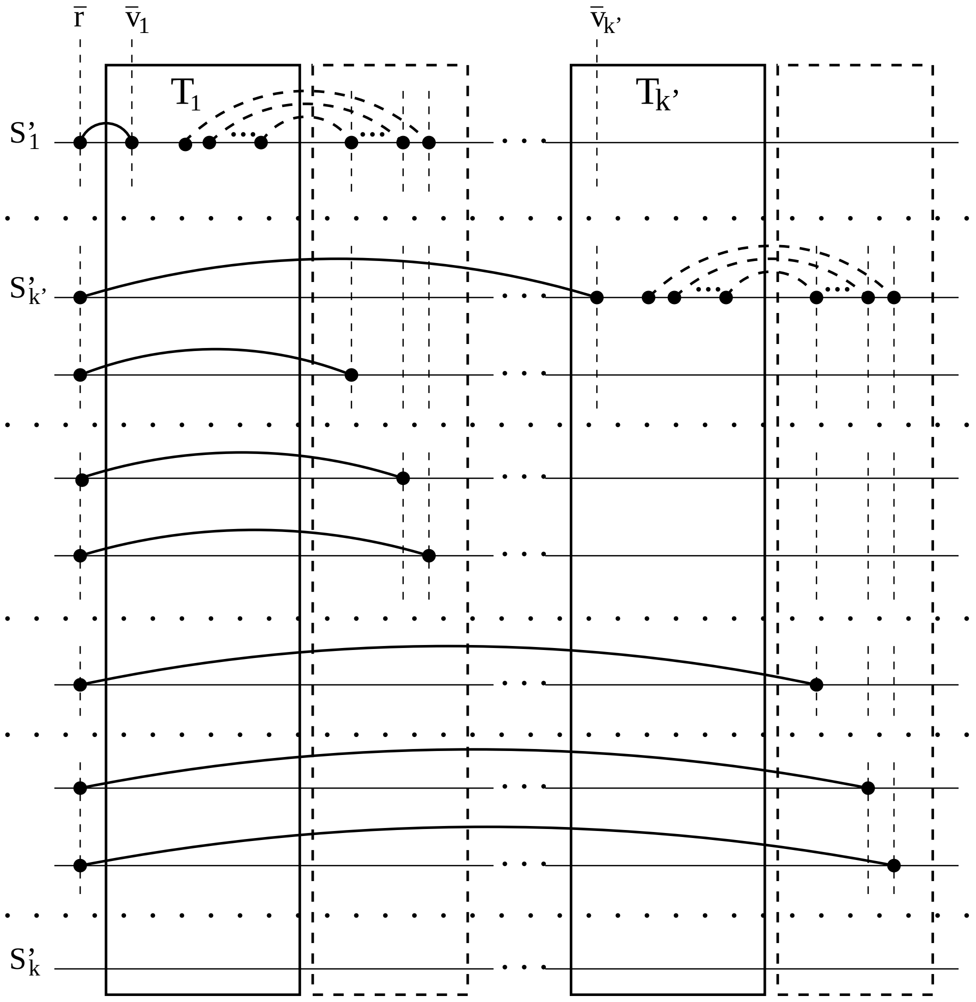


Fig. 7.

basepairs belonging to some shapes S'_i and S'_j respectively. Since $k \geq 3$, there exists a shape S'_l different from S'_i and S'_j. Note also that only the basepairs $\{\alpha_1, \alpha_2\}$ and $\{\alpha_2, \alpha_3\}$ contain the base α_2. So the base can be replaced α_2 by two neighbour bases α'_2 and α''_2, and the basepairs $\{\alpha_1, \alpha_2\}$ and $\{\alpha_2, \alpha_3\}$ can be replaced by the basepairs $\{\alpha_1, \alpha'_2\}$, $\{\alpha'_2, \alpha''_2\}$ and $\{\alpha''_2, \alpha_3\}$ belonging to the shapes S'_i, S'_j and S'_l respectively (see Figure 8). Applying in $G(S'_1, \ldots, S'_k)$ this transformation to all chains of even number of basepairs corresponding to secondary edges, we obtain the required shapes $S_1, \ldots, S_k$.

Note that for any secondary edge e the corresponding to e chain of basepairs in $G(S'_1, \ldots, S'_k)$ has no more than $a + 1$ basepairs where a is the number of subtrees of T in which the edge e originates. So this chain has no more than $|G|$ basepairs. Therefore, the corresponding to e chain of basepairs in $G(S_1, \ldots, S_k)$ has no more than $|G| + 1 = O(\|G\|)$ basepairs. Thus $\|G(S_1, \ldots, S_k)\| = O(\|G\|^2)$.

Since the DFS tree T can be constructed in time linear in $\|G\|$ (see, e.g., [4]) we can also observe that the shapes $S_1, \ldots, S_k$ can be constructed in time polynomial

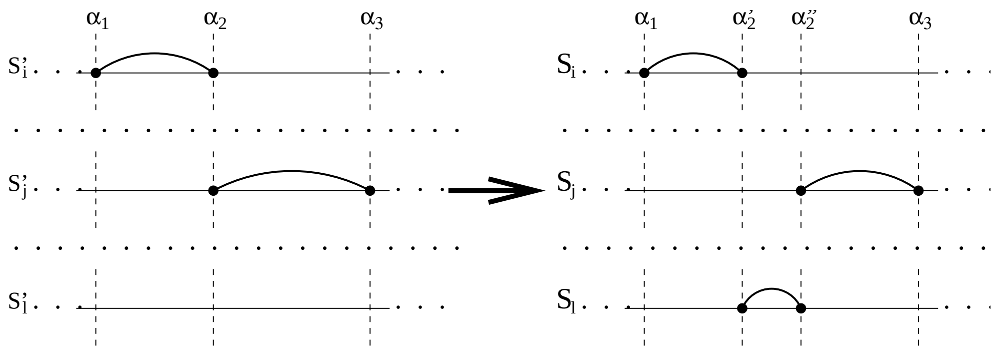


Fig. 8.

in the size of G. Thus, taking into account Lemma 18, we conclude that OFVSP[4] can be transformed in polynomial time into Problem 14 for $k \geq 4$. Therefore, Theorem 17 implies that Problem 14 is NP-hard for $k \leq 4$. On the other hand, we can propose a non-deterministic polynomial time algorithm for solving Problem 14: we guess a subset of bases in $G(S_1, S_2, \ldots, S_k)$ and check in linear time whether after removing this subset the remaining part of $G(S_1, S_2, \ldots, S_k)$ contains odd cycles, using the breadth first traversal of this part. Thus Problem 14 is NP-complete for $k \geq 4$. Since Problem 14 is equivalent to M$*$RP, we obtain

Theorem 20 *M$*$RP is NP-complete for $k \geq 4$.*

3 On Bounded Minimum Realizability

The results of Section 2 show that M$*$RP is NP-complete for $k \geq 4$ but leave open the complexity of this problem for $k = 3$. This raises the question on what is the complexity of a restricted version of this problem whereby the maximum number of possible stars is given in advance. First of all we give a precise definition of the problem.

Problem 21 (Min $*$ Realizability Problem for m Possible $*$s (M$*$RPP[m])) *For any shapes $S_1, S_2, \ldots, S_k$ of the same size compute a string over $\{0, 1, *\}$ which realize all these shapes and contains no more than m $*$s if such string exists.*

This "approximation" variant of the original problem has a simple polynomial time solution. To see this we argue as follows. To solve M$*$RPP[m], we have to decide if there exists a set of m vertices of $G(S_1, S_2, \ldots, S_k)$, such that there is no odd cycle in the set of vertices remaining after removing this set. For any subset of vertices of $G(S_1, S_2, \ldots, S_k)$ we can check the existence of an odd cycle not intersecting with this subset in $O(\|G(S_1, S_2, \ldots, S_k)\|)$ time, using the breadth first traversal of the complementary to this subset part of $G(S_1, S_2, \ldots, S_k)$. So we can make the following conclusion.

Lemma 22 *M$*$RPP[m] can be solved in $O\left(\binom{n}{m}\|G(S_1, S_2, \ldots, S_k)\|\right)$ time.*

In particular, Lemma 22 provides an $O(n\|G(S_1, S_2, \ldots, S_k)\|)$ time algorithm for M$*$RPP[1]. In the sequel we show how to remove the multiplicative factor n above and propose a more effective algorithm for solving this problem. First we define the important concepts of critical vertices and tours.

Definition 23 *A vertex of a graph G is called critical if it is contained in all odd cycles in G.*

Further in a graph we consider also cycles which can contain several times the same vertices and edges.

Definition 24 *Tours are cycles which may contain several times the same vertices and edges.*

First we consider the case of graphs with all critical vertices of degree at most 3. We later indicate all modifications that are necessary to extend this result to arbitrary graphs.

Lemma 25 *All critical vertices of a graph G of vertex degree at most 3 can be found in $O(\|G\|)$ time.*

Proof. First we check in $O(\|G\|)$ time whether G contains odd cycles by breadth first traversal of G. If there are no odd cycles in G then the problem is assumed to be solved. Let G contain odd cycles. Then we find any one odd cycle C in G such that C has no *chords*, i.e., edges which join vertices of C, but do not belong to C. We find previously any arbitrary odd cycle in G. It can be done, using backtracking of traversed edges during the traversal of G.

Let $C' = c'_1 c'_2 \ldots c'_h c'_1$ be the found odd cycle (i.e., h is odd). Then we label the vertices $c'_1, c'_2, \ldots, c'_h$ by two symbols 0 and 1 in such a way that any two consecutive vertices c'_i and c'_{i+1} are labeled by different symbols. After that we check for each of vertices $c'_1, c'_2, \ldots, c'_h$ consecutively if this vertex is joined to another vertex of C' by an edge not belonging to C'. Let us assume that we detect that a vertex c'_i is joined by an edge to a vertex c'_j where $j > i+1$. Then we compare the labels of c'_i and c'_j. If these labels are equal, we reduce the cycle C' to the odd cycle $c'_i c'_{i+1} \ldots c'_j c'_i$ by removing from C' the vertices $c'_{j+1}, c'_{j+2}, \ldots, c'_h, c'_1, c'_2, \ldots, c'_{i-1}$. If these labels are different, we reduce the cycle C' to the odd cycle $c'_1 c'_2 \ldots c'_i c'_j c'_{j+1} \ldots c'_h c'_1$ by removing from C' the vertices $c'_{i+1}, c'_{i+2}, \ldots, c'_{j-1}$. After this reduction, we continue the checking of vertices of the remaining cycle, going in the direction of vertices with greater indices. Each time we detect a new chord, we produce the above described reduction of the cycle. We stop the checking when the index of the next checked vertex is less than the index of the last checked vertex. One can see that after this procedure we obtain the required cycle C without chords.

Let $C = c_1c_2 \ldots c_rc_1$ (r is odd). Denote by G' the graph remaining after removing from G all vertices $c_1, \ldots, c_r$. In $O(\|G'\|)$ time we can also check whether G' contains odds cycles. If G' contains an odd cycle then G has no critical vertices. Assume G' does not contain odd cycles. Then G' is a bipartite graph, so we can label in $O(\|G'\|)$ time all vertices of G' by two numbers 0 and 1 in such a way that for any edge $\{u, v\}$ of G' the vertices u and v are labeled by different symbols. We will denote the label of a vertex v by $l(v)$. Simultaneously we can compute all connected components of G'.

Let $K_1, \ldots, K_s$ be all connected components of G' which are connected to vertices of C by at least two edges. Consider any component K_i. Let K_i is connected to vertices of C by edges $\{v_1, c_{j(1)}\}, \ldots, \{v_p, c_{j(p)}\}$ where $p \geq 2$ and $j(1) < j(2) < \ldots < j(p)$ (note that, since the vertices $c_{j(1)}, \ldots, c_{j(p)}$ have vertex degree at most 3, all these vertices are different, while the vertices $v_1, \ldots, v_p$ can coincide). In what follows we consider the ordered pairs of vertices $(c_{j(1)}, c_{j(2)}), (c_{j(2)}, c_{j(3)}), \ldots, (c_{j(p-1)}, c_{j(p)}), (c_{j(p)}, c_{j(1)})$ which we call *neighbour pair* of K_i. For a neighbour pair $(c_{j(q)}, c_{j(q')})$ the part $c_{j(q)}c_{j(q)+1} \ldots c_{j(q')}$ ($c_{j(q)}c_{j(q)+1} \ldots c_rc_1c_2 \ldots c_{j(q')}$ in the case of $q = p$) of the cycle C is called *the segment* of this neighbour pair. For each neighbour pair $(c_{j(q)}, c_{j(q')})$ we compute the value

$$l + l(v_q) + l(v_{q'}) \tag{1}$$

where l is the length of the segment of $(c_{j(q)}, c_{j(q')})$, i.e., $l = j(q') - j(q)$, if $q = 1, \ldots, p-1$, and $l = r + j(q') - j(q)$, if $q = p$. The neighbour pair is called *odd* if value (1) is odd, otherwise it is called *even*. Since the vertices $v_q, v_{q'}$ are in K_i, there exist paths in K_i between v_q and $v_{q'}$, and for each such a path there exists the cycle consisting of this path, the segment of $(c_{j(q)}, c_{j(q')})$, and the edges $\{v_q, c_{j(q)}\}$ and $\{v_{q'}, c_{j(q')}\}$. Any such cycle is called *associated* with the neighbour pair $(c_{j(q)}, c_{j(q')})$. Note that in any path in K_i any two adjacent vertices have different labels, so the length of any in K_i between v_q and $v_{q'}$ is odd if and only if $l(v_q) + l(v_{q'}) = 1$. Therefore, the length of any cycle associated with a neighbour pair is odd (even) if and only if the neighbour pair is odd (even). Note also that we can check for all neighbour pairs of the components $K_1, \ldots, K_s$ whether the neighbour pairs are odd or even during only one walking along the cycle C. Since all vertices of the cycle C have vertex degree at most 3, the total time required for the checking is bounded by $O(r)$. Thus for any K_i we can compute the number of odd neighbour pairs of K_i. We prove the following fact.

Proposition 26 *For any K_i the number of odd neighbour pairs of K_i is odd.*

Proof. Let $C_1, \ldots, C_p$ be some cycles associated respectively with all neighbour pairs of K_i, and l_t be the length of C_t, $t = 1, \ldots, p$. Then $\sum_{t=1}^{p} l_t = r + 2p + \sum_{t=1}^{p} l'_t$ where l'_t is the length of the part of C_t containing in K_i. Note that all these parts form a tour of length $\sum_{t=1}^{p} l'_t$ in K_i. Since in this tour any two adjacent vertices have different labels, the length $\sum_{t=1}^{p} l'_t$ of this tour has to be even. Hence, since r is odd,

$\sum_{t=1}^{p} l_t$ is also odd. Therefore, among the cycles $C_1, \dots, C_p$ we have odd number of odd cycles. Since each of these odd cycles corresponds to an odd neighbour pair of K_i, we conclude that K_i has odd number of odd neighbour pairs. $\square$

It follows from Proposition 26 that each of the components $K_1, \dots, K_s$ can have either just one or at least three odd neighbour pairs. Note that for any odd neighbour pair all critical vertices of G have to be contained in the segment of this neighbour pair since they have to be contained in both the cycle C and the odd cycle associated with the neighbour pair. Therefore, if some component K_i have at least three odd neighbour pairs then all critical vertices of G have to be contained in the intersection of the segments of all these neighbour pairs. One can easily see that any three different segments of neighbour pairs of the same component have the empty intersection, so in this case G have no critical vertices.

Thus we have to analyse only the case when each of the components $K_1, \dots, K_s$ has just one odd neighbour pair. Denote by L_i the segment of the only odd neighbour pair of the component K_i, $i = 1, \dots, s$, and by R the intersection of the segments $L_1, \dots, L_s$. In this case all critical vertices of G are contained in R. On the other hand, we prove that any vertex of R is critical vertex of G, i.e., any cycle in G not containing at least one vertex of R is even. We consider tours in G which can contain several times only vertices and edges of the cycle C. We call such tours *quasitours*. Let v be any vertex of R. We show that any quasitour T in G not containing this vertex is even. If T is not intersected with the cycle C, i.e., T is a cycle contained completely in G', then T is even, since G' has on odd cycles. Let T is intersected with C. Then T is divided into parts consisting of consecutive edges which either belong or not belong to C. The parts consisting of edge not belonging to C are called *external* parts. Since C has no chords, each external part goes through vertices of one of the components $K_1, \dots, K_s$. A quasitour (cycle) is called *primitive* if it has only one external part. First consider a primitive cycle $\hat{C}$ not containing v. The only external part of $\hat{C}$ goes through vertices of some component K_i. Then we can note that the other part of $\hat{C}$ is a part of C consisting of some number of consecutive segments of neighbour pairs of K_i. We call these neighbour pairs *internal pairs* of $\hat{C}$. The only odd neighbour pair of K_i can not be an internal pair of $\hat{C}$, since the vertex v is contained in the segment of this neighbour pair. So all internal pairs of $\hat{C}$ are even. We prove that $\hat{C}$ is even by induction on the number of internal pairs of $\hat{C}$. If $\hat{C}$ has only one internal pair then $\hat{C}$ is a cycle associated with this pair. Since this pair is an even neighbour pair, $\hat{C}$ has to be even. Assume now that any primitive cycle with no more than d internal pairs is even, and $\hat{C}$ has $d+1$ consecutive internal pairs $(u_1, u_2), (u_2, u_3), \dots, (u_{d+1}, u_{d+2})$. Let the vertex u_j is connected to K_i by an edge $\{u_j, u'_j\}$, $j = 1, 2, \dots, d+1$. Then the external part of $\hat{C}$ consists of the edges $\{u_1, u'_1\}, \{u_{d+2}, u'_{d+2}\}$ and some path P in K_i between the vertices u'_1 and u'_{d+2}.

Consider the shortest path $\hat{P}$ in K_i connecting the vertex u'_{d+1} to vertices of P. We have two cycles which are formed by the path $\hat{P}$ with the edge $\{u_{d+1}, u'_{d+1}\}$

and two different parts of $\hat{C}$ between the vertex u_{d+1} and the end of $\hat{P}$ in P. Note that each of these cycles is a primitive cycle which doesn't contain v and has no more than d internal pairs. So by induction hypothesis these cycles are even. From this observation we can easily conclude that the cycle $\hat{C}$ is even. We can also note that any primitive quasitour can be reduced to a primitive cycle on the same set of vertices by removing an even number of edges. Therefore, any primitive quasitour in G not containing v is even. Now we prove that T is even by induction on the number of external parts in T. If T has only one external part, i.e., T is primitive, then T is even as shown above. Assume that any quasitour with no more than d external parts is even, and T has $d+1$ external parts. Consider any one external part of T. Let this external part in T be between vertices v' and v'' and go through vertices of a component K_i. Recall that all vertices of R are contained in the segment of one neighbour pair of K_i. So all these vertices, including v, are contained in one of parts of C between the vertices v' and v''. Thus the other parts of C don't contain v. Consider two quasitours which are formed by this part of C with two different parts of T between the vertices v' and v''. One of these quasitours is primitive, and the other quasitour has d external parts. Moreover, both these quasitours don't contain v. Therefore, by induction hypothesis these quasitours are even. Hence we conclude that T is also even. Thus, any cycle not containing v is even, i.e., v is a critical vertex of G. So R is a set of all critical vertices of G.

In order to compute R, we choose the direction $c_1 \to c_2 \to \ldots \to c_r \to c_1$ in the cycle C. According to this direction, for any segment in C we define the bound vertices to be *the start* or *the end* of this segment. Let $L = \{L_1, \ldots, L_s\}$. First we replace in L each segment L_i containing both the vertices c_1 and c_r by two segments, one of which is the part of L_i ending in c_r, and the other is the part of L_i starting in c_1. After that we mark all vertices in C which are the starts or the ends of segments from L. Using this marking, we go along the cycle C from c_1 to c_r and count consecutively for any vertex c_j in C the number of segments from L containing c_j (if c_j is marked as the start of a segment then for c_j we increase the counter by one, and if c_j is marked as the end of a segment then for the next vertex c_{j+1} we decrease the counter by one). All vertices which are contained in s segments from L are placed in R. Note that the computing of R requires $O(r+s) = O(|G|)$ time. Thus the total time required for our algorithm is bounded by $O(\|G\|)$. $\square$

This completes the proof of Lemma 25. $\square$

The proposed algorithm can be modified for finding all critical vertices in arbitrary graphs (i.e., graphs with arbitrary vertex degree). In order to describe these modifications, we note that the case of arbitrary graphs has two essential differences from the case of graphs of vertex degree at most 3.

(1) The first difference is that any vertex c_j in C can have several adjacent vertices $v_j^{(1)}, \ldots, v_j^{(\tau)}$ in any component K_i. If some vertices $v_j^{(\alpha)}$ and $v_j^{(\beta)}$ have differ-

ent labels then there exists an odd cycle consisting of the edges $\{v_j^{(\alpha)}, c_j\}$, $\{v_j^{(\beta)}, c_j\}$ and some path in K_i between $v_j^{(\alpha)}$ and $v_j^{(\beta)}$. This cycle intersects with C only in c_j. So in this case c_j is an only potential critical vertex of G, and we can check whether c_j is a critical vertex in time $O(\|G\|)$. Thus, if for some vertex c in C and some component K_i we have two adjacent to c vertices in K_i labeled by different numbers, then we can find all critical vertices of G in time $O(\|G\|)$. Otherwise, for each neighbour pair $(c_{j(q)}, c_{j(q')})$ we compute the value (1), taking as the vertices v_q and $v_{q'}$ any vertices in the corresponding component which are adjacent to $c_{j(q)}$ and $c_{j(q')}$ respectively, and then we continue the procedure described in the proof of Lemma 25.

(2) The second difference of the case of arbitrary graphs is that any vertex in C can be the start or the end of several segments from L. So in this case, during the computation of the number of segments from L containing a vertex in C, we have to increase or to decrease the counter by the appropriate numbers of segments. Note that the described modifications do not increase the time bound for our algorithm.

So we obtain the following extesion of Lemma 25.

Lemma 27 *All critical vertices of an arbitrary graph G can be found in $O(\|G\|)$ time.*

Lemma 27 now implies the main theorem.

Theorem 28 *M$*$RPP[1] can be solved in $O(\|G(S_1, S_2, \ldots, S_k)\|)$ time.*

4 Conclusion and Open Problems

In this paper we have studied the problem of determining the minimum number of positions, for which after removal of all base pairs incident to these positions, there exists an RNA sequence which is compatible with each of k given secondary structures. There are several remaining interesting open problems which are worth investigating.

The first problem concerns the status of the min $*$ realizability problem for $k = 3$ secondary structures.

The second problem is related to the design of a more efficient algorithm to approximate the minimum number of stars required for the realizability of a shape graph $G(S_1, S_2, \ldots, S_k)$. The proof of Lemma 27 described above shows how to reduce the multiplicative factor $\binom{n}{m}$ when $m = 1$ and is quite complex. The complexity of the simple algorithm described in Lemma 22 above has a multiplicative factor n^m, where m is an upper bound on the number of *s, but it is not known whether a polynomial time (in m and n) approximation scheme for the min $*$ realizability problem exists.

The third problem is related to the design of an algorithm to decide whether or not a given arbitrary graph on n nodes and of vertex degree at most k is the shape graph of a string of length n using k shapes, i.e., of the form $G(S_1, S_2, \ldots, S_k)$. A similar question can be posed if the number of permissible $*$s is also given.

References

[1] I.C. Abfalter, C. Flamm, and P.F. Stadler. Design of multi-stable nucleic acid sequences. In *Prooceedings of the German Conference on Bioinformatics*, 2003. `http://www.tbi.univie.ac.at/papers/Abstracts/03-018.pdf`

[2] E. Bornberg-Bauer. Random structures and evolution of biopolymers: A computational case study on RNA secondary structures. *Pharmaceutica Acta Helvetiae*, 71:79–85, 1996.

[3] L.R. Comolli, I. Smirnov, L. Xu, E.H. Blackburn, and T.L. James. A molecular switch underlies a human telomerase disease. *Proc. Natl. Acad. Sci. USA*, 99(26):16998–17003, 2002.

[4] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001. Second edition.

[5] I. Hofacker et al. Vienna RNA Package. `http://www.tbi.univie.ac.at/~ivo/RNA/`

[6] C. Flamm, I.L. Hofacker, S. Mauer-Stroh, P.F. Stadler, and M. Zehl. Design of multistable RNA molecules. *RNA*, 7:254–265, 2001.

[7] W. Fontana, P. F. Stadler, P. Tarazona, E. Weinberger, and P. Schuster. RNA folding and combinatory landscapes. *Physical Review E*, 47(3):2083–2099, 1993.

[8] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman & Company, 1979.

[9] K. Gowda and C. Zwieb. Determinants of a protein-induced rna switch in the large domain of signal recognition particle identified by systematic-site directed mutagenesis. *Nucleic Acids Res.*, 25(14):2835–2840, 1997.

[10] K. A. Harris, D. M. Crothers, and E Ullu. In vivo structural analysis of spliced leader RNAs in Trypanosoma brucei and Leptomonas collosoma: a flexible structure that is independent of cap4 methylations. *RNA*, 1(4):351–362, 1995.

[11] K.A. Harris and D.M. Crothers. The Leptomonas collosoma spliced leader RNA can switch between two alternate structural forms. *Biochemistry*, 32(20):5301–5311, 1993.

[12] I. L. Hofacker, W. Fontana, P. F. Stadler, S. Bonhoeffer, M. Tacker, and P. Schuster. Fast folding and comparison of rna secondary structures. *Monatsh. Chem.*, 125:167–188, 1994.

[13] M. Mandal, B. Boese, J.E. Barrick, W.C. Winkler, and R.R. Breaker. Riboswitches control fundamental biochemical pathways in Bacillus subtilis and other bacteria. *Cell*, 113(5):577–586, 2003.

[14] D.H. Matthews, J. Sabina, M. Zuker, and D.H. Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *J. Mol. Biol.*, 288:911–940, 1999.

[15] A. Nahvi, J.E. Barrick, and R.R. Breaker. Coenzyme B12 riboswitches are widespread genetic control elements in prokaryotes. *Nucleic Acids Res.*, 32(1):143–150, 2004.

[16] C. Reidys, P. F. Stadler, and P. Schuster. Generic properites of combinatory maps: Neutral networks of RNA secondary structures. *Bulletin of Biology*, 59(2):339–397, March 1997.

[17] C. Reidys, P.F. Stadler, and P. Schuster. Generic properties of combinatory maps: neutral networks of RNA secondary structures. *Bull Math Biol.*, 59(2):339–397, 1997.

[18] E.A. Schultes and D.P. Bartel. One sequence, two ribozymes: Implications for the

emergence of new ribozyme folds. *Science*, 289(5478):448–452, 2000.

[19] P. Schuster. Molecular insights into evolution of phenotypes. In J.P. Crutchfield and P. Schuster, editors, *Evolutionary Dynamics – Exploring the Interplay of Accident, Selection, Neutrality, and Function*. Oxford University Press, New York, 2000.

[20] P. Schuster, W. Fontana, P. F. Stadler, and I. L. Hofacker. From sequences to shapes and back: A case study in RNA secondary structures. *Royal Soc London B*, 255:279–284, 1994.

[21] B Voss, C Meyer, and R. Giegerich. Evaluating the predictability of conformational switching in RNA. *Bioinformatics*, 20(10):1573–1582, 2004.

[22] M. Zuker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Res.*, 9:133–148, 1981.