Multiple 3D RNA Structure Superposition Using Neighbor Joining

David Hoksza and Daniel Svozil

Abstract—Recent advances in RNA research and the steady growth of available RNA structures call for bioinformatics methods for handling and analyzing RNA structural data. Recently, we introduced SETTER—a fast and accurate method for RNA pairwise structure alignment. In this paper, we describe MultiSETTER, SETTER extension for multiple RNA structure alignment. MultiSETTER combines SETTER's decomposition of RNA structures into non-overlapping structural subunits with the multiple sequence alignment algorithm ClustalW adapted for the structure alignment. The accuracy of MultiSETTER was assessed by the automatic classification of RNA structures and its comparison to SCOR annotations. In addition, MultiSETTER classification was also compared to multiple sequence alignment-based and secondary structure alignment-based classifications provided by LocARNA and RNADistance tools, respectively. MultiSETTER precompiled Windows libraries, as well as the C++ source code, are freely available from http://siret.cz/multisetter.

Index Terms—Structural bioinformatics, RNA tertiary structure, structural superposition, structural similarity

1 Introduction

Biological research in recent twenty years uncovered an appreciable role of RNA in many important biological processes [1], [2]. The discovery of various roles of RNA, such as its importance in gene expression regulation, greatly expanded our understanding of genome-related processes and resulted in the significant development of RNA research in the last decade. As a consequence, the number of known 3D RNA structures is steadily growing. With the increase of the number of RNA structures, the demand for effective searching in RNA databases also increases. Recently, we introduced SETTER [3], the RNA structure pairwise alignment algorithm and its web server implementation [4]. While several multiple RNA sequence alignment algorithms exist [5], [6], [7], the choice of methods for multiple 3D structure alignment is rather limited. Typically, 3D structure alignment is based either on the RNA secondary structure [8], [9] or on the projection of structural features into a sequence followed by the sequence alignment [10]. However, we are not aware of any algorithm for a direct multiple 3D RNA structure alignment. In this paper, we present MultiSETTER, a method for multiple 3D RNA structure alignment, its experimental evaluation and its publicly available software implementation. MultiSETTER is the extension of SETTER and can be used, e.g., for visual identification of common structural features or for an automatic classification of RNA structures.

- *D. Hoksza is with the Department of Software Engineering, Charles University, Prague, Czech Republic. E-mail: hoksza@ksi.mff.cuni.cz.*
- *D. Svozil is with the Laboratory of Informatics and Chemistry, Institute of Chemical Technology, Prague, Czech Republic. E-mail: svozild@vscht.cz.*

Manuscript received 13 Dec. 2013; revised 16 June 2014; accepted 23 July 2014. Date of publication 25 Aug. 2014; date of current version 1 June 2015. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TCBB.2014.2351810

2 Pairwise Structural Alignment Using SETTER

Similarly to protein structure, RNA structure can be described at four levels of detail. Primary structure is represented by the linear sequence of nucleotides. Unlike DNA, RNA comes single stranded and is free to form the multitude of intramolecular hydrogen bonds. RNA secondary structure consists of complex base-pairing patterns composed of double helices interconnected by various types of loops [11]. Tertiary structure is formed by the overall arrangement and packing of secondary structure elements [12]. If an RNA molecule contains several chains with tertiary structure, their mutual arrangement forms quaternary RNA structure.

SETTER uses base-pairing information to divide the RNA structure into basic alignment elements called generalized secondary structure units (GSSUs) [3]. A GSSU can be understood as a simplified version of a secondary structure motif. Each GSSU consists of three parts—a stem, a neck and a loop. RNA structure is formed by non-overlapping GSSUs which are identified along RNA sequence (Fig. 1).

Definition 1. *Let $S = \{nt_i\}_{i=1}^{n}$ be the nucleotide sequence of an RNA structure R and let $\mathcal{WC}$ be an ordered set of pairs from S where $(i, j) \in \mathcal{WC}$ denotes a base pair participating in the Watson-Crick base pairing. By a GSSU $\mathcal{G}$ we understand a pair of substrings of S $\{nt_i\}_{i=i_1}^{i_2}$ and $\{nt_j\}_{j=j_1}^{j_2}$ ($i_1 \leq i_2 < j_1 \leq j_2, i_2 = j_1 - 1$) of `maximum lengths` such that each nucleotide $nt_x \in \mathcal{G}$:*

- *$i_1 \leq x \leq i_2 : nt_x \notin \mathcal{WC}$ or nt_x is paired with nt_y where $j_1 \leq y \leq j_2$*
- *$j_1 \leq x \leq j_2 : nt_x \notin \mathcal{WC}$ or nt_x is paired with nt_y where $i_1 \leq y \leq i_2$*

In case of ambiguity, maximum length is assigned to the substring occurring earlier in the sequence. Let i_{max} and j_{min} be the highest/lowest indexes of the Watson-Crick paired bases

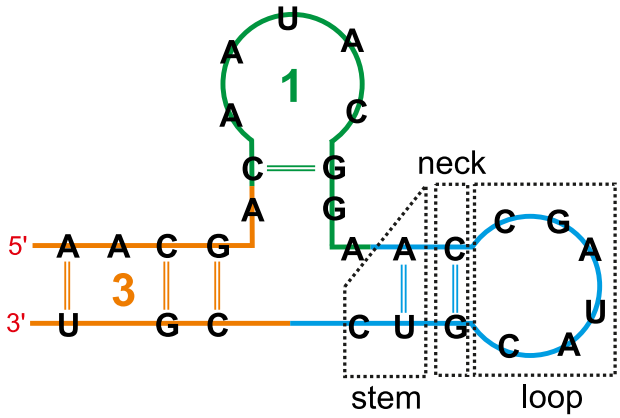


Fig. 1. Three GSSUs extracted from an RNA structure. The sequence starts at the 5' end. The borders between individual GSSUs are indicated by the dashed lines and the numbers show the order of the GSSU generation.

in $\mathcal{G}$. We define a ***loop*** *as $\mathcal{L} = \{nt_i\}_{i=i_{max}+1}^{j_{min}-1} \subset \mathcal{G}$, a* ***stem*** *as $\mathcal{G} \setminus \mathcal{L}$ and a* ***neck*** *as a pair $\{nt_{i_{max}}, nt_{j_{min}}\}$. Finally, by a GSSU substructure we understand the substructure of R consisting of nucleotides of the given GSSU.*

To compare a pair of RNA structures R_1 and R_2, each GSSU structure from R_1 is aligned with each GSSU structure from R_2. To align a pair of GSSU structures, three key residues from each GSSU are superimposed by the Kabsch algorithm [13]. The key residues are formed from two pairs of neck residues and one pair of loop residues such that the best superposition is achieved. The quality of the superposition is assessed by the S-score that represents the similarity of two GSSU structures.

$$
\begin{aligned}
NN_\zeta(x,\mathcal{G}) &= \begin{cases} \min_{1\le i\le|\mathcal{G}|}\{d(x,\mathcal{G}_i)\}\times\zeta & \text{if } t(x)=t(\mathcal{G}_{i_{min}}) \\ \min_{1\le i\le|\mathcal{G}|}\{d(x,\mathcal{G}_i)\} & \text{otherwise,}\end{cases}\\
\gamma(\mathcal{G}^{\mathcal{A}},\mathcal{G}^{\mathcal{B}}) &= \frac{1}{2}\left(\sum_{i=1}^{|\mathcal{G}^{\mathcal{A}}|}\begin{cases}1 & \text{if } NN_1(\mathcal{G}^{\mathcal{A}}{}_i,\mathcal{G}^{\mathcal{B}})\le\epsilon\\ 0 & \text{otherwise}\end{cases}\right.\\
&\quad\left.+\sum_{i=1}^{|\mathcal{G}^{\mathcal{B}}|}\begin{cases}1 & \text{if } NN_1(\mathcal{G}^{\mathcal{A}},\mathcal{G}^{\mathcal{B}}{}_i)\le\epsilon\\ 0 & \text{otherwise,}\end{cases}\right)\\
\delta(\mathcal{G}^{\mathcal{A}},\mathcal{G}^{\mathcal{B}}) &= \min_{t\in T}\left\{\frac{1}{2}\left(\sum_{i=1}^{|\mathcal{G}^{\mathcal{A}}|}NN_\zeta(\mathcal{G}^{\mathcal{A}}{}_i,\tau(\mathcal{G}^{\mathcal{B}},t))\right.\right.\\
&\quad\left.\left.+\sum_{i=1}^{|\mathcal{G}^{\mathcal{B}}|}NN_\zeta(\mathcal{G}^{\mathcal{A}},\tau(\mathcal{G}^{\mathcal{B}}{}_i,t))\right)\right\},\\
S(\mathcal{G}^{\mathcal{A}},\mathcal{G}^{\mathcal{B}}) &= \frac{\frac{\delta(\mathcal{G}^{\mathcal{A}},\mathcal{G}^{\mathcal{B}})}{\min\{|\mathcal{G}^{\mathcal{A}}|,|\mathcal{G}^{\mathcal{B}}|\}}\times\left(1+\frac{||\mathcal{G}^{\mathcal{A}}|-|\mathcal{G}^{\mathcal{B}}||}{\min\{|\mathcal{G}^{\mathcal{A}}|,|\mathcal{G}^{\mathcal{B}}|\}}\right)}{\gamma(\mathcal{G}^{\mathcal{A}},\tau(\mathcal{G}^{\mathcal{B}},t_{opt}))},
\end{aligned}
\tag{1}
$$

where $\mathcal{G}^{\mathcal{A}}$ and $\mathcal{G}^{\mathcal{B}}$ are two compared GSSU structures, $\mathcal{G}_i$ stands for the position of the i-th nucleotide (in the sequence order) of $\mathcal{G}$, and $|\mathcal{G}|$ is the number of nucleotides in $\mathcal{G}$. $NN(x,\mathcal{G})$ is the Euclidean distance from the nucleotide x to its nearest neighbor in $\mathcal{G}$. If x and its nearest neighbor share the same nucleotide type (the function $t(x)$ in the formula), the distance is modified by the factor ζ which takes values from the interval $0 < \zeta \le 1$. The lower the ζ, the more nucleotides with matching types are rewarded. δ computes the raw distance—T is the set of transpositions resulting from the candidate triplet alignments and $\tau(\mathcal{G},t)$ transposes GSSU $\mathcal{G}$ using the transposition t. The S-distance is then normalized by the γ function counting the number of nearest neighbors within the distance ϵ after the optimal transposition t_{opt}.

If at least one of the RNA structures consist of more than one GSSU, the pair of GSSU structures G_1^i, G_2^j with the lowest S-score drives the overall superposition of the two RNA structures. To superimpose structures R_1 and R_2, a translation vector and a rotation matrix defining the superposition of G_1^i, G_2^j is used. The quality of the structure alignment is evaluated by aggregating S-distances of all superposed GSSUs into the $\overline{S}$ score. For further details regarding the SETTER algorithm, including its comparison with other 3D RNA structure pairwise alignment methods, we refer to the original publication [3].

3 Multiple Structural Alignment Using MultiSETTER

MultiSETTER is the extension of SETTER for multiple structure alignment. MultiSETTER is based on principles used in ClustalW [14] algorithm for the multiple sequence alignment of proteins. ClustalW works in the following steps:

- Perform all possible alignments between each pair of sequences and generate a distance matrix.
- From the distance matrix, construct a dendrogram called 'guide tree' using the neighbor-joining method [15].
- Construct the multiple sequence alignment by aligning sequences in the order defined by the guide tree.

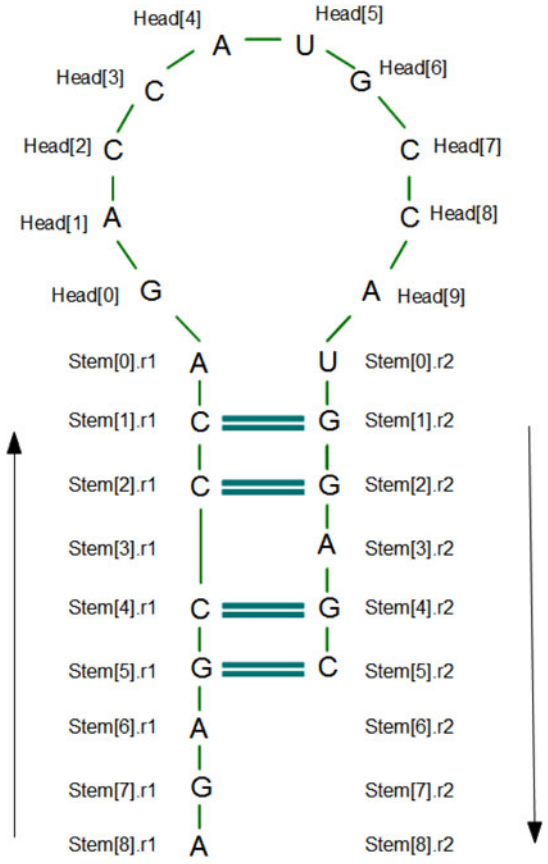
In MultiSETTER, we transformed the sequence-specific parts of ClustalW into their structure counterparts. MultiSETTER works as follows:

- Each pair of RNA structures is aligned by SETTER, and the distance matrix is constructed from the $\overline{S}$ distances.
- The guide tree is calculated from the distance matrix.
- Two most closely related structures are first aligned and the so-called *average RNA structure* is constructed by merging (averaging) positions of individual atom pairs. The alignment then progressively continues, averaging more and more structures, until the root of the guided tree is reached.

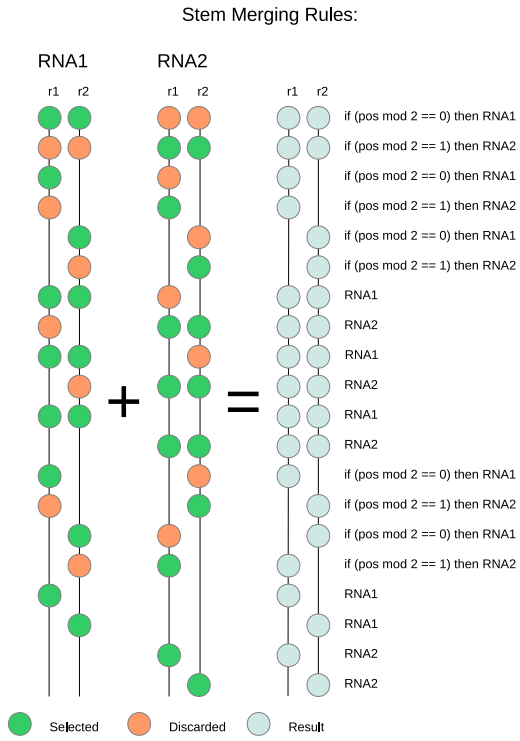
The average RNA structure is constructed by decomposing individual structures into GSSUs, merging corresponding GSSUs into average GSSUs and joining average GSSUs into the final average RNA structure.

3.1 RNA Merging

Before two RNA structures are merged, they are superposed by SETTER that provides the list of GSSU structure pairs G formed during the alignment, the translation vector and the rotation matrix. The structure with the lower sum of $\overline{S}$-distances to the rest of structures is designated as the *lead structure* L, the second structure is referred to as the *other structure* O. The GSSU pairs from G are then merged



(a) The example of the GSSU consisting of a stem and a loop. Residues *Stem[0].r1* and *Stem[0].r2* form a neck. However, the neck is considered to be the part of the loop in the GSSU merging algorithm.



(b) *Alternating* schema of merging two stems. Pos is the placeholder of the position of a pair in a stem.

Fig. 2. GSSU merging.

one by one (see Section 3.2) each into a new GSSU structure $GSSU_M$. The set of merged GSSU structures forms the merged RNA structures.

Because outliers may negatively influence the merging process, they are identified and discarded. This is done by calculating *meanDistance*, the average distance between all GSSU structures. If the distance between two GSSU structures deviates significantly from the *meanDistance*, the merging step for that pair of GSSUs is skipped and the GSSU structure from the lead structure is used as $GSSU_M$. Similarly, the merged GSSU structure is not accepted if its distance to the $GSSU_L$ is greater then the distance (modified by the parameter *param2*) between two original GSSU structures. The parameters *param1* and *param2* modify the influence of outliers in the alignment.

3.2 GSSU Structure mMerging

In general terms, the goal of GSSU structure merging is to construct a new GSSU structure from two input GSSU structures. The new structure should be a structural average of the input structures. The algorithm merges loops and stems separately, with the neck being considered as the part of the stem (see Fig. 2a). In each step, a new merged residue is constructed and placed between the two input residues.

$GSSU_M \leftarrow GSSU_L$
if S($GSSU_L$, $GSSU_O$) < meanDistance * param1 **then**
  $GSSU_M \leftarrow$ merge($GSSU_L$, $GSSU_O$);
  if S($GSSU_M$, $GSSU_L$) + S($GSSU_M$, $GSSU_O$) < S($GSSU_L$, $GSSU_O$) * param2 **then**
    $GSSU_M \leftarrow GSSU_L$
  end if
end if

3.2.1 Stem Merging

Stem merging follows the Algorithm 1. As an input, the algorithm requires the information on which side ($r1$ or $r2$—see Fig. 2a) of which GSSU structure to merge. The algorithm consecutively merges individual stem residues. In every step, one residue is selected as a template and positioned using a translation vector (see next paragraph). Two approaches for the selection of the template residue (function *SelectResidue*, line 16) are implemented. In the first

approach, referred to as *Leading*, $GSSU_L$ residues are used as template residues. In the second approach, called *Alternating*, the modulo 2 function is used to identify individual template residues (see Fig. 2b).

Algorithm 1. Stem merging *(merges r_j stem of $GSSU_L$ and $GSSU_O$ and outputs $GSSU_M$)*

```
translationVector ← (0,0,0);
translationDir ← 1;
stem1 ← GSSU_L.stem; stem2 ← GSSU_O.stem;
if GSSU_L.stem.Length() < GSSU_O.stem.Length() then
  stem1 ← GSSU_O.stem; stem2 ← GSSU_L.stem;
end if
if stem2.Length = 0 then
  GSSU_M.stem.side[r_j] ← stem1.side[r_j];
  return GSSU_M;
end if
step ← stem1.Length() / stem2.Length();
for i = stem1.Length() downto 0 do
  residue1 ← stem1[i].r_j; residue2 ← stem2[step * i].r_j;
    /* Select a residue based on the merging rules. */
  pos ← i;
   mergedResidue ← SelectResidue(residue1, residue2, pos);
  if resiude1 ! = NULL AND residue2 ! = NULL then
      translationVector ← CalculateTranslation(residue1, residue2, mergedResidue);
    translationSource ← mergedResidue.Template;
  else
      /* One of the residues not present ⇒ translation
         vector from the previous iteration used. */
    if mergedResidue.Template = translationSource then
      translationDir ← 1;
    else
      translationDir ← −1;
    end if
    TranslateAtoms(mergedResidue.atoms,
     translationDir * translationVector);
    mergedGSSU.atoms.append(mergedResidue.
     atoms);
    mergedGSSU.stem[i].side[r_j] ← mergedResidue;
  end if
end for
```

To place the atoms of merged residues correctly, a translation vector is calculated by the *CalculateTranslation* function. This function is called only if residues are present in both stems. The resulting vector translates the template residue to the midpoint between the input residues. If the template residue comes from the same side as the residues from the previous step, the same translation vector is used. Otherwise, the translation vector is multiplied by -1 which reverts its direction.

3.2.2 Loop Merging

The loop merging is identical with the stem merging. The only difference is that the *SelectResidue* function is not needed because for each loop's position, both residues exist.

4 MutliSETTER Application

MultiSETTER is available as a command-line application. It provides the access both to the pairwise (SETTER) and multiple (MultiSETTER) structure superposition. Specifically, the tool consists of four modules:

- *Multiple structural superposition module (MSSM).* The input to MSSM is the set of RNA structures in the PDB format together with their secondary structure annotations calculated and stored by 3DNA [16]. The MSSM module constructs the average structure, creates the Jmol script for the visualization of the superposed structures and serializes the average structure for later use (e.g., in the BMSSM module (see below)).
- *Pairwise structure superposition module (PSSM).* The PSSM module implements the SETTER algorithm for the RNA structure pairwise alignment. The input to PSSM is the pair of structures in the PDB format and their corresponding secondary structures, and its output is the Jmol script storing the structure superposition.
- *Batch multiple structural similarity module (BMSSM).* BMSSM enables batch comparison between RNA structures and the previously built average structure. For each input structure, BMSSM produces its distance to the average structure. BMSSM is intended to be used for classification purposes.
- *Batch pairwise structural similarity module (BPSSM).* BPSSM is a supporting module for all-against-all pairwise distance computations. The same functionality can be achieved by the multiple use of the PSSM module.

4.1 Parallelization

MultiSETTER can use multiple CPU cores for the structure superposition. Currently, the following parts of the algorithm are parallelized:

- Distance matrix computation. The construction of the all-against-all distance matrix used in the neighbor joining algorithm requires many pairwise superpositions. Because these are mutually independent, they can be scheduled on separate computational units.
- Merging of RNA structures. As described above, RNA structures are merged GSSU by GSSU. Because merging of one GSSU pair has no influence on merging another GSSU pair, the available computational resources can be utilized in parallel.

4.2 Availability

C++ source code and precompiled Windows binaries of MultiSETTER can be freely downloaded from http://siret.cz/multisetter. Various supporting functions in MultiSETTER are provided by the Boost libraries [17] and the

parallelization is provided by the Intel Thread Building Block Library (TBB) [18]. Because the source code does not contain any platform-specific parts, MultiSETTER can be compiled at any operating system supporting Boost and TBB libraries.

5 Experimental Evaluation

To assess the accuracy of MultiSETTER, RNA structures from the SCOR (Structural classification of RNA) database [19], [20] were automatically classified. SCOR is a human curated database of RNA structures hierarchically organized based on their function and tertiary interactions. At the lowest level of the SCOR hierarchy, functionally similar structures form the so called 'families'. From the SCOR database, we extracted 96 structures classified into 14 families of various sizes and intrinsic diversities. Each family contains at least four structures. SCOR family annotations, PDB codes of structures and their chain IDs are given in the following list.

1) *tRNA (Phe).* 1EHZ-A, 1EVV-A, 1TN2-A, 1TRA-A, 4TNA-A, 4TRA-A, 6TNA-A
2) *tRNA (Gln).* 1C0A-B, 1EFW-C, 1IL2-C, 1ASY-R, 1ASZ-R
3) *tRNA (Asp).* 1EUY-B, 1EXD-B, 1GTR-B, 1GTS-B, 1O0C-B, 1QRS-B, 1QTQ-B
4) *Synthetic.* 1J4Y-A, 1KKA-A, 1LUU-A, 1LUX-A
5) *Zymomonas mobilis.* 1Q2R-E, 1Q2R-F, 1Q2S-E, 1Q2S-F
6) *tRNA (Lys).* 1BZ2-A, 1BZ3-A, 1BZT-A, 1BZU-A, 1FEQ-A, 1FL8-A
7) *5S rRNA in 50S subunit.* 1JJ2-9, 1K73-B, 1K8A-B, 1K9M-B, 1KC8-B, 1KD1-B, 1KQS-9, 1M1K-B, 1M90-B, 1N8R-B, 1NJI-B, 1Q7Y-B, 1Q81-B, 1Q82-B, 1Q86-B, 1QVF-9, 1QVG-9, 1S72-9
8) *P5 stem loop.* 1C0O-A, 1F9L-A, 1GUC-A
9) *Hammerhead ribozyme.* 1NYI-A, 1Q29-A, 1HMH-A, 1MME-A, 299D-A, 359D-A, 379D-A, 488D-A
10) *MS2 phage coat protein binding stem-loop.* 1AQ3-A, 1D0T-A, 1D0U-A, 1ZDH-A, 1ZDI-A, 1ZDJ-A, 1ZDK-A
11) *HIV-1 TAR RNA.* 1ANR-A, 1ARJ-A, 1QD3-A, 397D-A, 1AKX-A, 1LVJ-A, 1UTS-B, 1UUD-B, 1UUI-A
12) *HIV-1 Rev response element Rev binding site.* 1CSL-A, 1DUQ-A, 1EBQ-A, 1EBR-A, 1EBS-A, 1ETF-A, 1I9F-A
13) *HIV-1 psi RNA stem loop SL1.* 1M5L-A, 1N8X-A, 1JTJ-A, 1JU1-A, 1F6U-A, 1OSW-A
14) *RNA double strand, bound to protein.* 1A34-A, 2BBV-A, 1RC7-A, 1DI2-A, 1N35-A.

MultiSETTER accuracy was assigned using the leave-one-out classification of all 96 structures according to the following protocol:

- Pick one of the 96 structures and set it as the query structure RNA_Q.
- Remaining 95 structure form a database that contains 14 structural classes.
- Compute the average structure for each of the 14 classes.
- Compute the distances between RNA_Q and each of the 14 average structures.
- Assign RNA_Q to the family with the closest average structure.

TABLE 1
Comparison of MultiSETTER and SETTER

Position	MultiSETTER		SETTER	
	Alternating (%)	Leading (%)	Best (%)	Mean (%)
1	80.21	78.13	86.60	70.10
2	4.17	6.25	2.06	4.12
3	4.17	3.125	5.15	2.06
4	1.04	2.08	0.00	6.19
5	1.04	1.04	1.03	6.19
6	1.04	2.08	0.00	2.06
7	1.04	0.00	2.06	0.00
8	0.00	1.04	0.00	2.06
9	0.00	0.00	2.06	1.03
10	2.08	2.08	0.00	1.03
11	2.08	2.08	0.00	1.03
12	1.04	1.04	0.00	1.03
13	0.00	1.04	1.03	3.09
14	2.08	0.00	0.00	0.00

For each structure S from a family F, the 14 families are sorted according to their distances to the S. Line i shows how often the family F ends at position i. Alternating *and* Leading *setups (see Section 3.2) were used in MultiSETTER, and* Best *and* Mean *setups (see Section 5) in SETTER.*

The multiple structure alignment-based classification obtained by MultiSETTER was compared with the SETTER pairwise alignment based classification. To classify a query structure with SETTER, its distance to the remaining 95 structures was computed. The class of the query structure was recognized by the following approaches:

- *Best*—The database of 95 structures was sorted in the decreasing order of the distance between individual structures and the query structure. The query structure was assigned to the class of the closest database hit.
- *Mean*—For each family, the average distance between the query structure and the family members was computed. The query structure was assigned to the class with the lowest average distance.

In all experiments, SETTER was run with its default parameters [4]. MultiSETTER parameters *param1* and *param2* were set to 0.9 and 2.0, respectively. All computations were performed on a machine with an Intel Core i7-2620M processor with two cores and four threads, 4 Gigabytes of RAM and hard drive with 7,200 rpm; running 64 bit version of Microsoft Windows 7.

5.1 Accuracy Evaluation

The accuracy of MultiSETTER and SETTER for the classification of RNA structures is summarized in Table 1. Each row in Table 1 contains the percentage of 96 queries for which the class of the query structure ended at the given position. For example, position 2 means that first structure from the same structural class as the query is found at the second position within the list of structures sorted by their distances to the query. The structure at the position 1, though closer to the query, is annotated with wrong classification.

Table 1 shows that most accurate results are obtained by SETTER pairwise structural alignment with the *Best* setup (i.e., the functional classification is based on the most similar

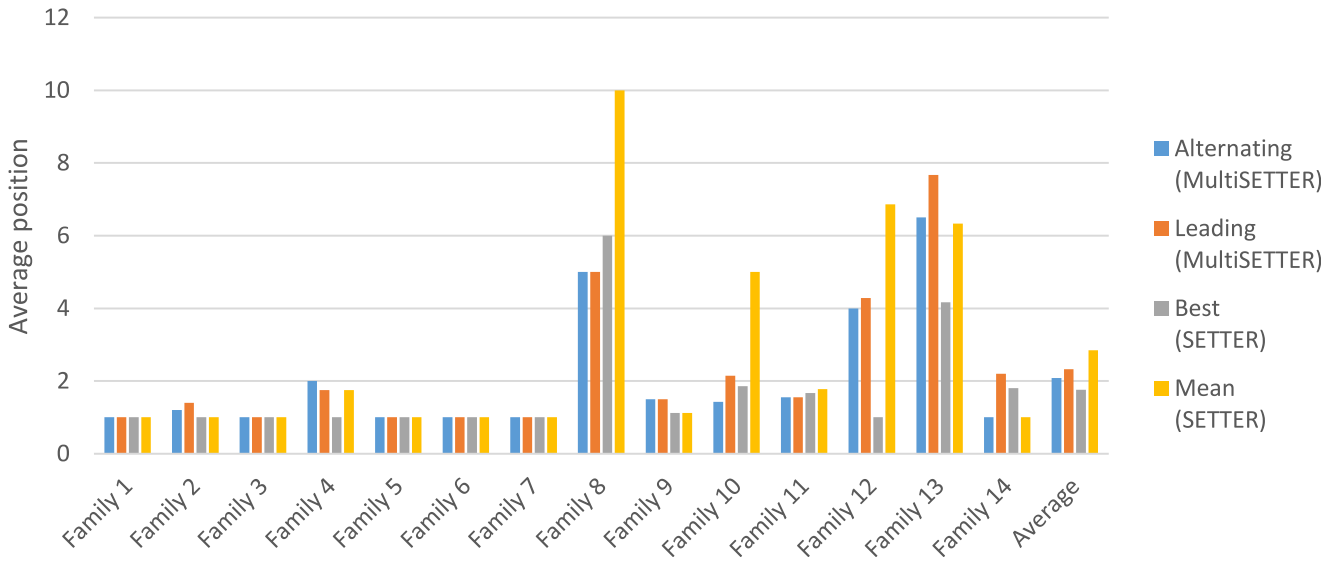


Fig. 3. Average positions of correct classifications of all SETTER and MultiSETTER setups. A lower value means better classification performance.

structure). Compared to the *Mean* setup, the *Best* setup yields better results because it does not rely on the average structure which can be influenced by outliers. Though MultiSETTER also uses the average structure, its classification is much more accurate than that of SETTER with the *Mean* setup. Thus we conclude that the classification by the multiple structure alignment is more precise than the classification based on averaging over individual pairwise alignments.

These conclusions are consistent with the visual inspection of the alignments. Fig. 3 shows average positions of correct classifications decomposed into individual families. For example, if a family contains two structures only, one structure ends at the second and another one at the fourth position, the average position for this family is three. Fig. 3 demonstrates that MultiSETTER typically outperforms SETTER-Mean and, in some cases, also SETTER-Best approaches.

In addition to the quantitative assessment, the quality of the classification can also be verified visually by inspecting the structures sharing a common fold. Figs. 4a, 4b, 4c, and 4d show multiple alignments of four families differing by the level of structural homology. Family 1 (tRNA(Phe)) in Fig. 4a contains homologous structures, while family 12 (HIV-1 Rev response element Rev binding site, Fig. 4c) is more diverse, i.e., it contains more outliers. That is the reason why family 12 is accurately classified using the SETTER pairwise alignment with the *Best* setup (see Fig. 3). Fig. 4b demonstrates how MultiSETTER deals with one outlying structure (see the following section for more discussion on outliers). In addition, MultiSETTER can also align very large structures, such as the following 23S ribosomal subunits (Fig. 4d): 1JJ2-0, 1K8A-A, 1K9M-A, 1M1K-A, 1NJP-0, 1NKW-0, 1S72-0.

5.2 Outliers Influence

In this section, the robustness of MultiSETTER with respect to outliers is validated. We will show that the enrichment of the SCOR family by substantially different RNA structures does not significantly influence the average RNA structure. This will be demonstrated on the family 11 (HIV1-TAR-RNA) which contains nine structures (Fig. 5a) that share a common topology but differ in their local structural characteristics. Gradually, one randomly chosen structure from the family 11 was replaced by one randomly chosen structure from a randomly chosen family. Up to 6 structures were substituted (Figs. 5b, 5c, 5d, 5e, 5f, and 5g), which comprises 66 percent of the family 11. First structure from the family 11 was replaced by the 1EUY-B structure (Fig. 5b) from the family 3 (tRNA(Asp)). 1EUY-B structure is noticeably different from family 11 structures with which it is aligned based on the common U-shape part with the long tails unaligned. Also the alignment of other outlying structures is focused around the U-shaped part (Figs. 5c, 5d, 5e, 5f, and 5g). However, as a number of outliers increases, the quality of the superposition decreases, though only weakly. Thus, we can conclude that a reasonable alignment can be achieved even for a rather heterogeneous set containing as much as 66 percent of outliers (Fig. 5g).

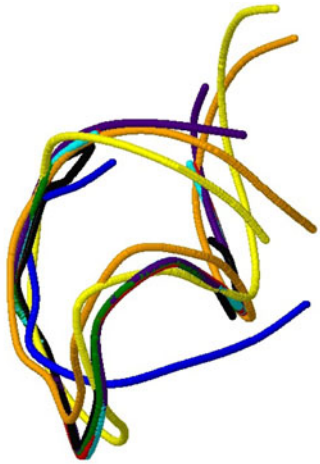
5.3 Comparison with Multiple Sequence and Secondary Structure Alignments

MultiSETTER, being a multiple structure alignment, enables visual examination of multiple structures and identification of their common structural characteristics. However, for the classification purposes also multiple sequence alignment (MSA) or secondary structure alignment (SSA) can be used. In this section, the accuracy of functional classification by MultiSETTER is compared to the classification performed by MSA and SSA methods.

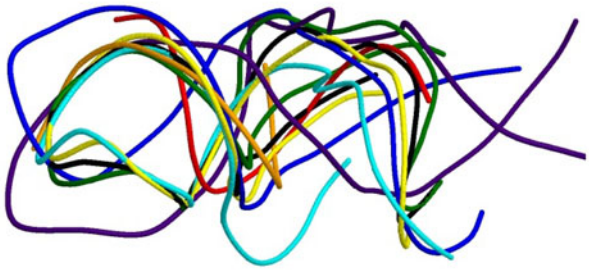
From the plethora of tools available [21], LocARNA [22], the part of the Freiburg RNA Tools [23], was chosen for the MSA. Similarly to MultiSETTER, to classify a sequence S MSAs with and without S were built for each of the 14 families (see Section 5). MSAs were scored by the commonly used sum of pairs (SP) metric. The SP score was used with insertion, deletion and mismatch penalties set to 1. Moreover, the SP score was normalized by the number of sequences in the MSA which corrects for the tendency to classify each sequence into a smaller family.

For the classification using the secondary structure alignment, the RNADistance tool from the ViennaRNA package

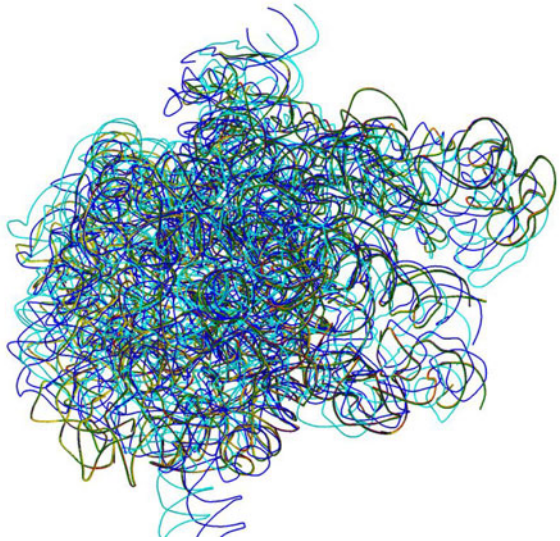
(a) 3D alignment of the family 1 (tRNA(Phe)).



(b) 3D alignment of the family 10 (MS2 phage coat protein binding stem-loop).



(c) 3D alignment of the family 12 (HIV-1 Rev response element Rev binding site).



(d) 3D alignment of the large structures of 23S ribosomal subunits.

Fig. 4. Examples of multiple structure superpositions using MultiSETTER. The average structure is shown in black.

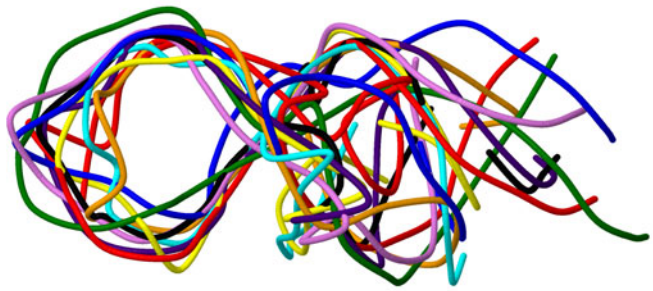
[24] was chosen. RNADistance is based on a tree distance for pairwise RNA secondary structure comparison. Since it supports pairwise distance computations only, the same approach as for the SETTER classification was applied. Each structure was compared with all remaining structures, and the results were sorted according to the decreasing similarity. The families were then sorted based on their highest scoring members.

The results of the classification experiments are summarized in Table 2. If the best scoring family is used for the classification, MultiSETTER outperforms both MSA and SSA. However, MSA is more stable as the correct family ends up only rarely worse that at the fifth position. On the other hand, in some cases, in MultiSETTER the family of the structure to be classified is the least similar one. Same trend, though not that pronounced, can also be observed for the SSA classification. This is not surprising because the alignment by MultiSETTER is based on the decomposition of the structure to secondary structure elements.

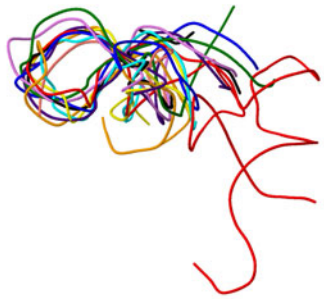
To demonstrate these observations, the specific examples of the erroneous MSA, SSA and MultiSETTER classifications are described in this section. MSA misclassified the 1ZDJ:R structure from the family 10 (MS2 phage coat protein binding stem-loop) to the family 14 (RNA double strand, bound to protein). Figs. 6a and 6b show the alignments of the families 10 and 14 with the 1ZDJ:R sequence. Adding 1ZDJ:R to the family 14 adds less to the SP score than adding the sequence to the family 10. The structure alignments of the families 10 and 14 are shown in Figs. 6c and 6d. Though the 1ZDJ:R structure is an outlier in both families, MultiSETTER assigns it correctly to the family 10 as it is more similar to family 10 average structure than to family 13 average structure.

On the other hand, the 1LUU:A structure was misclassified by MultiSETTER, while MSA yielded its correct classification. The 1LUU:A structure belongs (Fig. 7a) to the family 4 (Synthetic RNA) which has all sequences almost identical. Thus, MSA classifies this structure correctly. However, MultiSETTER assigns the 1LUU:A structure to the family 6 (tRNA(Lys)) which is, indeed, structurally more fitting to (Fig. 7b).

Also SSA classification shows errors which can be avoided when the tertiary structure superposition by



(a) 3D superposition of the family 11.



(b) 3D superposition of the family 11 with 1 structure replaced by the outlier.

(c) 3D superposition of the family 11 with 2 structures replaced by the outliers.

(d) 3D superposition of the family 11 with 3 structures replaced by the outliers.

(e) 3D superposition of the family 11 with 4 structures replaced by the outliers.

(f) 3D superposition of the family 11 with 5 structures replaced by the outliers.

(g) 3D superposition of the family 11 with 6 structures replaced by the outliers.

Fig. 5. The influence of the outliers on the structure alignment. The average RNA structure is shown in black, the outliers are shown in red. The family 11 consists of nine HIV-1 TAR RNA structures.

MultiSETTER is employed. For example, all structures from the family 14 (RNA double strand, bound to protein) are misclassified by RNADistance. In this case, both MSA and MultiSETTER yielded better results; MultiSETTER classified correctly all family 14 structures.

To conclude, while MultiSETTER performs better than MSA classification, these approaches should be considered complementary. We expect higher accuracy by combining sequence and structure alignments of RNA molecules for which both sequence and structure are available. Though MultiSETTER yields results comparable to SSA classification, it offers the advantage of the visual inspection of the alignment in 3D space. This enables qualitatively different analysis and may lead to the discovery of structural and functional relationships not captured by the secondary structure motifs.

TABLE 2
Comparison of the Tertiary Structure Alignment (MultiSETTER), Secondary Structure Alignment (RNADistance) and Multiple Sequence Alignment (LocARNA)

	MultiSETTER	RNADistance	LocARNA
1	80.21	78.13	68.68
2	4.17	6.25	6.02
3	4.17	3.13	10.84
4	1.04	0.08	6.02
5	1.04	1.04	4.82
6	1.04	2.08	1.21
7	1.04	0.0	1.21
8	0.0	1.04	1.21
9	0.0	0.0	0.0
10	2.08	2.08	0.0
11	2.08	2.08	0.0
12	1.04	1.04	0.0
13	0.0	1.04	0.0
14	2.08	0.0	0.0

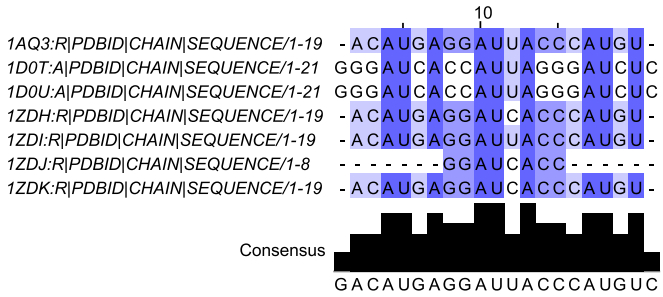
For each structure S from a family F, the 14 families are sorted according to their distances to the S. Line i shows how often the family F ends at position i.

5.4 Runtime Evaluation

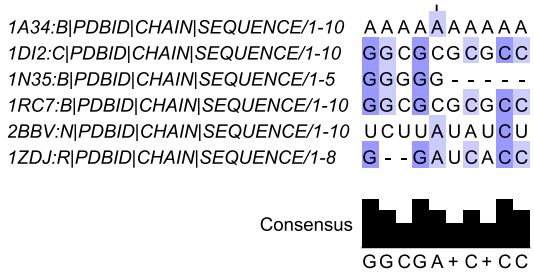
In this section, time requirements of the MultiSETTER and SETTER classification are compared. Reported times do not include neither PDB parsing, nor GSSU generation times because GSSU generation is performed during the initialization phase. In MultiSETTER classification, the average structure of each family is constructed first. This is a one-time task which is performed at the very beginning of the calculation. Computation times for average structures constructions are presented in Table 3. The required time is proportional to the family size; it ranges from less than one second for small families up to 177 seconds for the family 7 (18 structures of the 5S rRNA in 50S subunit). The use of two multithreaded cores leads to the average speedup of about 75 percent. Table 4 shows runtimes of the classification. Because of the precomputation of family average structures, MultiSETTER is more than four times faster than SETTER. This speedup is even more pronounced if the parallel version of MultiSETTER is used.

6 Future Plans and Conclusion

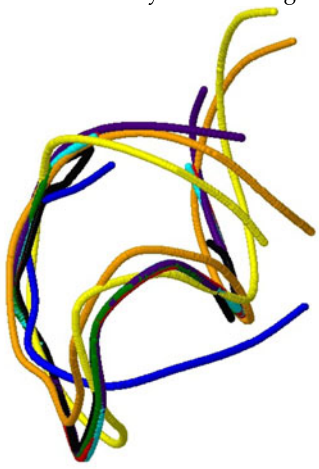
In this paper, we described MultiSETTER algorithm for the multiple RNA structure alignment. MultiSETTER is the extension of the pairwise RNA structure alignment method SETTER [3], [4]. MultiSETTER is based on principles used in ClustalW [14]. The accuracy of MultiSETTER was assessed by the classification using structural data and annotations from the SCOR database [19], [20]. If SETTER classification is based on the average RNA structures of individual families, MultiSETTER yields better results than SETTER. Because the RNA average structures can be precomputed, MultiSETTER classification is several fold faster than



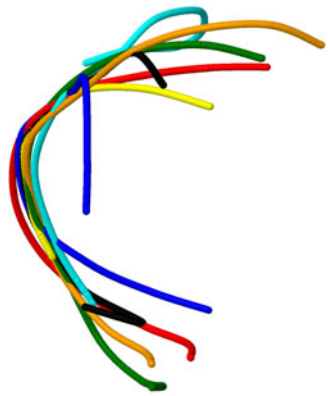
(a) MSA of the family 10 including 1ZDJ:R.



(b) MSA of the family 14 with 1ZDJ:R.

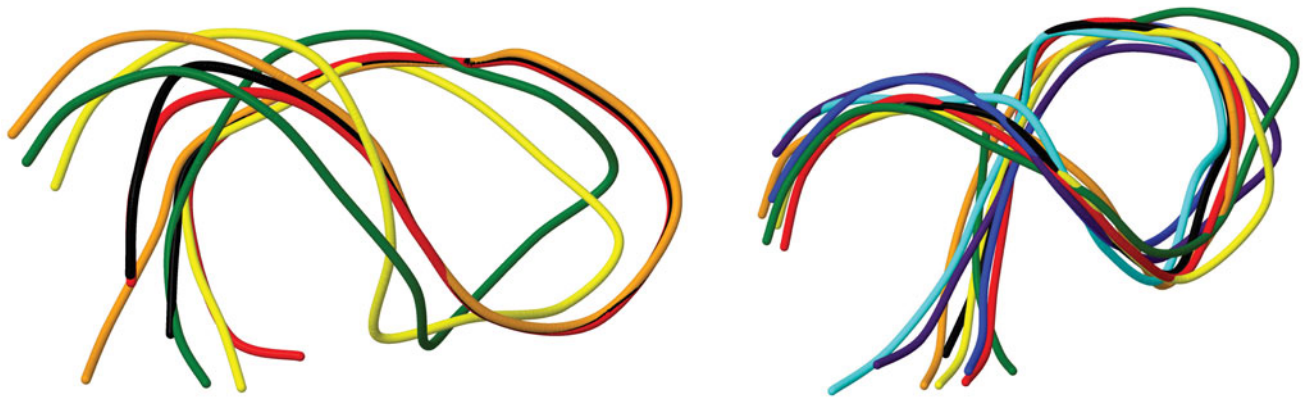


(c) 3D alignment of the family 10 (average structure in black, 1ZDJ:R in blue).



(d) 3D alignment of the family 14 (average structure in black, 1ZDJ:R in blue).

Fig. 6. An example of the wrong sequence-based and correct structure-based classification of the 1ZDJ:R structure. The family 10 contains MS2 phage coat protein binding stem-loop, the family 14 is double stranded RNA bound to protein.



(a) 3D alignment of the family 4 (Synthetic RNA). (b) 3D alignment of the family 6 (tRNA(Lys)).

Fig. 7. An example of the wrong structure-based classification of the 1LUU:A structure. The 1LUU:A structure is shown in green.

SETTER classification. In addition, it was demonstrated that the multiple structure alignment by MultiSETTER is reasonably robust with respect to the outlying structures. The comparison with multiple sequence alignment shows that though MutiSETTER yields generally better results, these two approaches should be considered complementary. We expect higher accuracy by combining sequence and structure alignments of such RNA molecules, for which both the sequence and structure are available. In the near future, we will apply this approach for the development of the automated system for the RNA structure classification. MultiSETTER yields results comparable to the secondary structure-based alignment algorithm RNADistance. However, MultiSETTER offers the advantage of the visual inspection of the alignment in 3D space which may reveal structural and functional relationships not captured by the secondary structure motifs. MultiSETTER Windows binaries, as well as C++ source code, are freely available from http://siret.cz/multisetter.

TABLE 3
Times in (s) Needed to Create the Average Structures of Individual Families

Family	1	2	3	4	5	6	7
1 thread	7.9	3.4	8.0	0.4	0.8	0.8	114.2
4 threads	4.9	2.0	4.7	0.3	0.4	0.5	57.7
Family	8	9	10	11	12	13	14
1 thread	0.4	177.0	1.1	11.6	15.4	2.6	1.6
4 threads	0.2	108.2	0.7	5.9	9.9	1.5	1.0

Single-threaded and four-threaded runtimes are reported.

TABLE 4
Times in (s) Needed to Classify all Structures in the Dataset (the Middle Column) and an Average Time Needed to Classify One Structure (the Right Column)

Method	Time (s)	Avg. time (s)
SETTER	1,669	16.7
MultiSETTER (1 thread)	359	3.6
MultiSETTER (4 threads)	160	1.6

ACKNOWLEDGMENTS

This research was supported by the Czech Science Foundation (GAČR) [project Nr. P202/11/0968].

REFERENCES

[1] S. R. Eddy, "Noncoding RNA genes and the modern RNA world," *Nature Rev. Genet.*, vol. 2, no. 12, pp. 919–929, Dec. 2001.
[2] D. P. Bartel. (2004). MicroRNAs: Genomics, biogenesis, mechanism, and function. *Cell* [Online]. *116*, pp. 281–297. Available: http://wormbase.org/db/misc/paper?name=WBPaper00006373
[3] D. Hoksza and D. Svozil, "Efficient RNA pairwise structure comparison by setter method," *Bioinformatics*, vol. 28, no. 14, pp. 1858–1864, 2012.
[4] P. Cech, D. Svozil, and D. Hoksza, "Setter: Web server for rna structure comparison," *Nucleic Acids Res.*, vol. 40, no. Web-Server-Issue, pp. 42–48, 2012.
[5] H. Kiryu, Y. Tabei, T. Kin, and K. Asai, "Murlet: A practical multiple alignment tool for structural rna sequences," *Bioinformatics*, vol. 23, no. 13, pp. 1588–1598, 2007.
[6] S. Moretti, A. Wilm, D. G. Higgins, I. Xenarios, and C. Notredame, "R-coffee: A web server for accurately aligning noncoding rna sequences," *Nucleic Acids Res.*, vol. 36, no. Web-Server-Issue, pp. 10–13, 2008.
[7] K. Katoh, K. Misawa, K.-I. Kuma, and T. Miyata, "MAFFT: A novel method for rapid multiple sequence alignment based on fast Fourier transform," *Nucleic Acids Res.*, vol. 30, no. 14, pp. 3059–3066, Jul. 2002.
[8] E. Torarinsson, J. H. Havgaard, and J. Gorodkin, "Multiple structural alignment and clustering of RNA sequences," *Bioinformatics*, vol. 23, no. 8, pp. 926–932, Apr. 2007.
[9] Y. Tabei, H. Kiryu, T. Kin, and K. Asai, "A fast structural multiple alignment method for long RNA sequences," *BMC Bioinformatics*, vol. 9, no. 1, pp. 33+, Jan. 2008.
[10] Y.-F. Chang, Y.-L. Huang, and C. L. Lu, "Sarsa: A web tool for structural alignment of rna using a structural alphabet," *Nucleic Acids Res.*, vol. 36, no. Web-Server-Issue, pp. 19–24, 2008.
[11] D. K. Hendrix, S. E. Brenner, and S. R. Holbrook, "RNA structural motifs: Building blocks of a modular biomolecule." *Q. Rev. Biophys.*, vol. 38, no. 3, pp. 221–243, Aug. 2005.
[12] S. R. Holbrook, "Structural principles from large RNAs." *Annu. Rev. Biophys.*, vol. 37, no. 1, pp. 445–464, 2008.
[13] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A*, vol. 32, no. 5, pp. 922–923, Sep. 1976.
[14] L. Ma, G. Blackshields, B. Np, R. Chenna, M. Pa, H. Mcwilliam, F. Valentin, W. Im, A. Wilm, R. Lopez, T. Jd, G. Tj, and H. Dg. (2007, Sep.). ClustalW and ClustalX version 2.0. *Bioinformatics (Oxford, England)* [Online]. *23*, pp. 2947–2948 Available: http://hubmed.org/display.cgi?uids=17846036
[15] N. Saitou and M. Nei, "The neighbor-joining method: A new method for reconstructing phylogenetic trees," *Mol. Biol. Evol.*, vol. 4, no. 4, pp. 406–425, 1987.

[16] X.-J. Lu and W. K. Olson, "3DNA: A versatile, integrated software system for the analysis, rebuilding and visualization of three-dimensional nucleic-acid structures," *Nature Protocols*, vol. 3, no. 7, pp. 1213–1227, Jul. 2008.
[17] *BOOST C++ Libraries* [Online]. Available: http://boost.org, 2014.
[18] J. Reinders, *Intel Threading Building Blocks*, 1st ed. Sebastopol, CA, USA: O'Reilly & Assoc., 2007.
[19] P. S. Klosterman, M. Tamura, S. R. Holbrook, and S. E. Brenner, "SCOR: A Structural Classification of RNA database," *Nucleic Acids Res.*, vol. 30, no. 1, pp. 392–394, Jan. 2002.
[20] M. Tamura, D. K. Hendrix, P. S. Klosterman, N. R. Schimmelman, S. E. Brenner, and S. R. Holbrook, "SCOR: Structural classification of RNA, version 2.0." *Nucleic Acids Res.*, vol. 32, no. Database issue, Jan. 2004.
[21] S. Washietl, "Sequence and structure analysis of noncoding RNAs," *Methods Mol. Biol.*, vol. 609, pp. 285–306, 2010.
[22] S. Will, T. Joshi, I. L. Hofacker, P. F. Stadler, and R. Backofen, "LocARNA-P: Accurate boundary prediction and improved detection of structural RNAs," *RNA*, vol. 18, no. 5, pp. 900–914, May 2012.
[23] C. Smith, S. Heyne, A. S. Richter, S. Will, and R. Backofen. (2010). Freiburg rna tools: A web server integrating intarna, exparna and locarna. *Nucleic Acids Res.* [Online]. *38(Web-Server-Issue)*, pp. 373–377. Available: http://dblp.uni-trier.de/db/journals/nar/nar38.html#SmithHRWB10
[24] R. Lorenz, S. H. F. Bernhart, C. H. zu Siederdissen, H. Tafer, C. Flamm, P. F. Stadler, and I. L. Hofacker. (2011). Viennarna package 2.0," *Algorithms Mol. Biol.* [Online]. *6*, p. 26. Available: http://dblp.uni-trier.de/db/journals/almob/almob6.html#LorenzBSTFSH11

David Hoksza received the PhD degree from the Department of Software Engineering, Charles University in Prague, Prague, Czech Republic, in 2010. Since 2011, he has been an Associate professor of software engineering in the Department of Software Engineering at the Charles University in Prague. His current research interests include structural bioinformatics, chemoinformatics, data engineering and similarity searching.

Daniel Svozil received the PhD degree from the Department of Analytical Chemistry, Charles University in Prague, Prague, Czech Republic, in 1997. Since 2011, he has been an Associate professor of physical chemistry in the Laboratory of Informatics and Chemistry at the Institute of Chemical Technology, Prague. His current research interests include structural bioinformatics of nucleic acids, application of data mining in bioinformatics and chemoinformatics, and chemical space mining and exploration.