IndelsRNAmute: Predicting deleterious multiple point substitutions and indels mutations

Alexander Churkin, Yann Ponty, Danny Barash

IndelsRNAmute: Predicting deleterious multiple point substitutions and indels mutations.*

Alexander Churkin[1], Yann Ponty[2], and Danny Barash[3]

[1] Department of Software Engineering, Sami Shamoon College of Engineering, Beersheba, Israel
alexach3@sce.ac.il

[2] Laboratoire d'Informatique de l'École Polytechique (LIX CNRS UMR 7161), Ecole Polytechnique, Palaiseau, France
yann.ponty@lix.polytechnique.fr

[3] Department of Computer Science, Ben-Gurion University, Beersheba, Israel
dbarash@cs.bgu.ac.il

Abstract. RNA deleterious point mutation prediction was previously addressed with programs such as RNAmute and MultiRNAmute. The purpose of these programs is to predict a global conformational rearrangement of the secondary structure of a functional RNA molecule, thereby disrupting its function. RNAmute was designed to deal with only single point mutations in a brute force manner, while in MultiRNAmute an efficient approach to deal with multiple point mutations was developed. The approach used in MultiRNAmute is based on the stabilization of the suboptimal RNA folding prediction solutions and/or destabilization of the optimal folding prediction solution of the wild type RNA molecule. The MultiRNAmute algorithm is significantly more efficient than the brute force approach in RNAmute, but in the case of long sequences and large m-point mutation sets the MultiRNAmute becomes exponential in examining all possible stabilizing and destabilizing mutations. Moreover, an inherent limitation in both programs is their ability to predict only substitution mutations, as these programs were not designed to work with deletion or insertion mutations. To address this limitation we herein develop a very fast algorithm, based on suboptimal folding solutions, to predict a predefined number of multiple point deleterious mutations as specified by the user. Depending on the user's choice, each such set of mutations may contain combinations of deletions, insertions and substitution mutations. Additionally, we prove the hardness of predicting the most deleterious set of point mutations in structural RNAs.

Keywords: RNA mutations prediction · RNA indels prediction · Suboptimal RNA structure.

* This research was supported by a Joint Research Projects grant from the Israeli Ministry of Science & Technology (MOST) and the French Centre National de la Recherche Scientifique (CNRS)

1 Background

The RNA molecule can be examined at several structural levels. That secondary structure of an RNA is a representation of the pattern, given an initial RNA sequence, of complementary base-pairings that are formed between the nucleic-acids. Represented as a string of four letters, the sequence is a single strand that consists of the nucleotides A, C, G, and U, which are generally assumed to pair to form a secondary structure with minimum free energy. As such, the secondary structure of RNA is experimentally accessible based on minimum free energy calculations, thus making its computational prediction a challenging but practical problem: it can be directly tested in the laboratory with minimal experimental effort relative to, for example, RNA tertiary structure. In addition, in many cases there is a known correspondence between the secondary structure of an RNA and the molecule's ultimate function.

In examining RNA viruses, they are known to possess unique secondary structures. The secondary structure of an RNA virus such as the Hepatitis C Virus (HCV) is mostly elongated due to the large number of base pairings that are formed, thereby lowering its free energy considerably and making the virus much more thermodynamically stable than a random RNA sequence. The typical stem-loop structure motif of an RNA virus, which consists of a long stem (a chain of consecutive base pairs) that ends in an external unpaired loop, has been experimentally observed to play a significant role in both virus replication and translation initiation. For example, in HCV, disruptive mutations were found to cause a structural change that directly led to either an alteration in virus replication [9,13] or to a dramatic reduction in translation initiation [10].

Deleterious mutation prediction in RNAs is a sub-problem of the RNA folding prediction problem, which is fundamental in RNA bioinformatics. Thus, all tools for deleterious mutations analysis utilize methods developed for the RNA folding problem. The most common methods for RNA folding prediction in general are energy minimization methods that use dynamic programming, for example the `mfold` server [14], `RNAstructure` [7] and the `ViennaRNA` package and server [4,5,6]. For the sub-problem considered in this work, the first publicly available methods for the analysis of deleterious mutations in RNAs were the `RNAmute` java tool [1] and a web server `RDMAS` [8]. Both of these methods utilize the Vienna RNA package for RNA folding prediction and are able to analyze only single point mutations in RNA sequences. To deal with multiple point deleterious mutations, the `MultiRNAmute` program [2] was developed, which uses an efficient method to find multiple point mutations using suboptimal folding solutions of an RNA sequence. The approach used in `MultiRNAmute` is based on examining a limited number of mutations, which stabilize some distant suboptimal secondary structure or/and destabilize the optimal secondary structure of the RNA sequence under consideration. Other approaches, among which the most well-known is `RNAmutants` [11], were also developed.

A major limitation of the above described methods is that the methods are able to predict only substitution mutations, but not insertions or deletions. The suggested approach to extend the MultiRNAmute to predict deletions and in-

sertions was briefly introduced in [3]. In addition, although the algorithm used in MultiRNAmute program is considerably more efficient than any brute-force algorithm, it still may become exponential for sizable inputs such as sequences longer than 100 nts and large multiple point mutations sets. Herein out motivation is to develop a method that predicts some predefined number (user's input) of deleterious mutations of different types and stops, without searching all "good" mutations as in MultiRNAmute.

The paper is organized as follow. We first prove the NP-hardness of predicting the most deleterious set of mutations in structural RNAs. We show that, even for a very simple energy model, the associated optimization problem is NP-complete. We then describe a fast algorithm, based on the approach used in MultiRNAmute, for the prediction of a predefined number of deleterious multiple point insertion, deletion and substitution mutations. Our new method is named IndelsRNAmute, and is freely available at:

https://www.cs.bgu.ac.il/~dbarash/Churkin/SCE/IndelsRNAmute/

2 Problem definition and NP-hardness

An RNA w is a **nucleotide sequence** of length n over an **alphabet** $\Sigma = \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{U}\}$. A **secondary structure** is a set of base-pairs $S = \{(a_i, b_i)\}_i \subset [1, n]^2$ such that $a_i < b_i$, and each position is involved in at most one base pair. We consider a simple, base pair based, **energy model** where the energy of a sequence/structure pair (w, S) is given by

$$E_{w,S} := -|\{(x, y) \in S \mid \{w_x, w_y\} \in \mathcal{B}\}| \text{ with } \mathcal{B} := \{\{\mathsf{A}, \mathsf{U}\}, \{\mathsf{C}, \mathsf{G}\}, \{\mathsf{G}, \mathsf{U}\}\}.$$

Non-canonical base pairs do not contribute to the energy in the model.

For a given RNA w, a **mutation** is a pair $\mu = (i, b)$, expressing the choice of a new, mutated, nucleotide $b \in \Sigma - \{w_i\}$ for the position i. An **edit script** $\mathcal{M} = \{\mu_1, \ldots, \mu_m\}$ consists in a set of mutations, each acting on a different position. Denote by ($w^{\mathcal{M}}$ (resp. $S^{\mathcal{M}}$) the **application** of an edit script $\mathcal{M}$ onto a sequence w (resp. structure S). Note that, when edit operations are limited to single-points mutations, one has $S^{\mathcal{M}} = S$.

MaxDelMuts Problem
Input: Wild-type sequence w_{T} of length n, Functional secondary structure MFE, Competing secondary structures $\overline{\mathcal{S}} = \{\overline{S}_1, \ldots, \overline{S}_k\}$, #Mutations m
Output: Maximally deleterious set of mutations

$$\mathcal{M}^{\star} = \underset{\mathcal{M}=\{\mu_1,\ldots,\mu_m\}}{\operatorname{argmax}} \left(E_{w_{\text{T}}^{\mathcal{M}},\text{MFE}^{\mathcal{M}}} - E_{w_{\text{T}},\text{MFE}}\right) + \sum_{i=1}^{k} \left(E_{w_{\text{T}},\overline{S}_i} - E_{w_{\text{T}}^{\mathcal{M}},\overline{S}_i^{\mathcal{M}}}\right)$$

Note that the result of the argmax is not affected by constant terms, so the objective can be equivalently defined as

$$\mathcal{M}^\star = \underset{\mathcal{M}=\{\mu_1,\ldots,\mu_m\}}{\operatorname{argmax}} E_{w_{\mathrm{T}}{}^{\mathcal{M}},\mathrm{MFE}^{\mathcal{M}}} - \sum_{i=1}^{k} E_{w_{\mathrm{T}}{}^{\mathcal{M}},\overline{S}_i^{\mathcal{M}}} \tag{1}$$

Theorem 1. *MaxDelMuts* $\in$ NP

Proof. Clearly, the number of ways to choose locations for m mutations within a sequence w is given by $\binom{n}{m} \in \Theta(2^n)$, while there exists exactly 3^m ways to assign a nucleotide content of those positions, thus the number of sets of mutations is bounded by an exponential function in n. Moreover, evaluating the objective function only requires the free-energy computation for $k+1$ pairs of secondary structures/mutants, which can be performed in $\Theta(n \times k)$ time.

Theorem 2. *MaxDelMuts is* NP*-hard*

Proof. Consider the following problem, proven NP-hard by Yannakakis [12].

MaxCoCycle Problem
Input: Cubic graph $G = (V, E)$
Output: Maximum cardinality co-cycle:

$$V^\star = \underset{V' \subset V}{\operatorname{argmax}} \left|\{(x,y) \in E \mid (x \in V') \oplus (y \in V')\}\right|$$

We show that MaxCoCycle can be reduced to MaxDelMuts.

Indeed, consider an instance $G = (V, E)$ for MaxCoCycle, assuming without loss of generality that $V = [1, n]$. We build an instance of MaxDelMuts, consisting of a sequence $w_{\mathrm{T}} = \mathsf{A}^n$, a number of mutations $m = n$, a functional empty structure $\mathrm{MFE} = \varnothing$, and a set $\overline{\mathcal{S}}$ of competing secondary structures, obtained by partitioning E into $\mathcal{O}(|E|)$ competing secondary structures[4]. Using the simplified expression (1), the objective function becomes

$$\mathcal{M}^\star = \underset{\mathcal{M}}{\operatorname{argmax}} \sum_{i=1}^{k} -E_{w_{\mathrm{T}}{}^{\mathcal{M}},\overline{S}_i^{\mathcal{M}}} = \underset{\mathcal{M}}{\operatorname{argmax}} \sum_{i=1}^{k} \left|\left\{(x,y) \in \overline{S}_i \mid \{w_{\mathrm{T}}{}_x^{\mathcal{M}}, w_{\mathrm{T}}{}_y^{\mathcal{M}}\} \in \mathcal{B}\right\}\right|.$$

Since $\overline{\mathcal{S}}$ represents a partition of E, then the expression of $E^\star$ further simplifies as

$$\mathcal{M}^\star = \underset{\mathcal{M}}{\operatorname{argmax}} \left|\left\{(x,y) \in E \mid \{w_{\mathrm{T}}{}_x^{\mathcal{M}}, w_{\mathrm{T}}{}_y^{\mathcal{M}}\} \in \mathcal{B}\right\}\right|.$$

Let us turn to the properties of $\mathcal{M}$ and $w_{\mathrm{T}}{}^{\mathcal{M}}$. Clearly, since $n = m$, all positions of w_{T} have to be mutated exactly once. Thus, after application of $\mathcal{M}$,

[4] Note that, if crossing pairs (*aka* pseudoknots) are allowed, the Vizing theorem implies that $\overline{\mathcal{S}}$ can be reduced in polynomial time to 3 structures, although this observation bears no consequence on the hardness of the problem.

there is no longer any occurrence of A in ${w_{\mathrm{T}}}^{\mathcal{M}}$. It follows that any base pairs contributing to the objective functions is either $\{\mathsf{G},\mathsf{C}\}$ or $\{\mathsf{G},\mathsf{U}\}$, *i.e.* a valid base pair must present exactly one occurrence of G, thus $(\{{w_{\mathrm{T}}}_x^{\mathcal{M}}, {w_{\mathrm{T}}}_y^{\mathcal{M}}\} \in \mathcal{B})$ is equivalent to $(({w_{\mathrm{T}}}_x^{\mathcal{M}} = \mathsf{G}) \oplus ({w_{\mathrm{T}}}_y^{\mathcal{M}} = \mathsf{G}))$. Denoting as $\mathcal{G}(\mathcal{M}) := \{x \in [1,n] \mid {w_{\mathrm{T}}}_x^{\mathcal{M}} = \mathsf{G}\}$ the set of occurrences of G in ${w_{\mathrm{T}}}^{\mathcal{M}}$, one has

$$\mathcal{M}^\star = \underset{\mathcal{M}}{\operatorname{argmax}} \left|\{(x,y) \in E \mid ((x \in \mathcal{G}(\mathcal{M})) \oplus (y \in \mathcal{G}(\mathcal{M}))\}\right|.$$

In other words, the objective value achieved by $\mathcal{M}$ for MaxDelMuts coincides with the objective value of $\mathcal{G}(\mathcal{M})$ for MaxCoCycle.

This suggests a proof by contradiction for the optimality of $\mathcal{G}(\mathcal{M}^\star) \subseteq V$ as a solution for MaxCoCycle. Denote by α (resp. β) the objective value of $V^\star$ (resp. $\mathcal{G}(\mathcal{M}^\star)$) for MaxCoCycle, and assume that $\alpha > \beta$. Then consider the edit script $\mathcal{M}'$, which sets all positions of $V^\star$ to G, and all other positions to C. $\mathcal{M}'$ provably achieves an objective value of $\alpha > \beta$ for MaxDelMuts. This contradicts the optimality of $\mathcal{M}^\star$, and one concludes that $\alpha = \beta$, *i.e.* $\mathcal{G}(\mathcal{M}^\star)$ represents a (co-)optimal solution of MaxCoCycle. Thus any polynomial algorithm for solving MaxDelMuts, coupled with a linear time computation of $\mathcal{G}(\mathcal{M}^\star)$, would provide an exact polynomial algorithm for the NP-hard MaxCoCycle. Therefore, MaxDelMuts is NP-hard.

3 Methods

Similar to the `MultiRNAmute` method, the `IndelRNAmute` method uses suboptimal secondary structures as a starting point. The motivation behind this decision is to start with some distant (from optimal structure) suboptimal structures and to convert such suboptimal structures to an optimal one by introducing wise mutations, which stabilize the stems of the suboptimal structure and destabilize stems of the optimal one.

The mutation analysis algorithm consists of several steps. First, given an input sequence with several input parameters, the Minimum Free-Energy (MFE) and a set of suboptimal secondary structures are calculated using the `RNAfold` and `RNAsubopt` programs from the Vienna RNA package [4,5], followed by a filtering step to reduce the number of suboptimal structures. Next, for each optimal and suboptimal structure, stems are identified and used for selecting *good* single-point mutations of several types: insertions, deletions and substitutions, depending on the user's choice.

Finally, these single-point mutations are combined together to form deleterious multiple-point mutations for the output. A summary of the algorithm is shown in Algorithm 1. We detail each step of the method in the following sections.

3.1 Input parameters

The parameters of the method include:

Algorithm 1: IndelsRNAmute

Input: RNA sequence w, number of sets of mutations N, number of mutations in the set M, distance $D1$, distance $D2$, e range E, types of supported mutations: INS, DEL, SUB
Output: N sets of deleterious M-points mutations

MFE $\leftarrow$ MFE structure of S ▷ From `RNAfold`
$\mathcal{S} \leftarrow$ Sub-optimal structures of S ▷ From `RNAsubopt` with threshold E
foreach $S \in \mathcal{S}$ **do**
  $D \leftarrow$ base-pair distance (MFE, S)
  if $D \leq D1$ **then** $\mathcal{S} \leftarrow \mathcal{S} \setminus S$
Sort $\mathcal{S}$ by distance from MFE in decreasing order
foreach $(S_1, S_2) \in \mathcal{S}^2, S_1 \neq S_2$ **do**
  $D \leftarrow$ base-pair distance (S_1, S_2)
  if $D \leq D2$ **then** $\mathcal{S} \leftarrow \mathcal{S} \setminus S_2$
foreach $S \in \mathcal{S}$ **do**
  if $SUB = true$ **then**
    $S.sub \leftarrow$ substitutions stabilizing S or/and destabilizing MFE
  if $INS = true$ **then**
    $S.ins \leftarrow$ insertions stabilizing S or/and destabilizing MFE
  if $DEL = true$ **then**
    $S.del \leftarrow$ deletions stabilizing S or/and destabilizing MFE
$\mathcal{M} \leftarrow \varnothing$
#Iter $\leftarrow N/|\mathcal{S}|$ ▷ Assuming $N > |\mathcal{S}|$
foreach $S \in \mathcal{S}$ **do**
  $i \leftarrow 0$
  while $|\mathcal{M}| < N$ *and* $i < \#Iter$ **do**
    Mut $\leftarrow M$ random mutations from $S.sub$, $S.ins$ and $S.del$
    $w_{\text{Mut}} \leftarrow S$ mutated by Mut
    $S_{\text{Mut}} \leftarrow$ MFE structure of w_{Mut} ▷ From `RNAfold`
    Fix length of S_{Mut} and MFE to be equal ▷ In case of indels
    $D \leftarrow$ base-pair distance (S_{Mut}, MFE)
    if $D \geq D1$ **then** $\mathcal{M} \leftarrow \mathcal{M} \cup$ Mut
return $\mathcal{M}$

– RNA sequence field (S) - the maximum sequence length allowed in our application is 1000 bases;

– dist 1 (D_1) - this distance parameter is used for filtering suboptimal solutions that are close to the optimal solution. The suggested value to use is around 30% of the RNA sequence length;

– dist 2 (D_2) - this distance parameter is used for filtering suboptimal solutions that are close to each other. The suggested value to use is around 30% of the RNA sequence length;

- e range (E) - this energy parameter is used in the *RNAsubopt* program to calculate the suboptimal structures within a range of kcals/mol of the mfe. The suggested value is around 15% of the RNA sequence length;
- #Point mutations (M) - number of allowed point mutations in RNA (one M-point mutation set);
- #Results (N) - number of M-point mutations in the output;
- Type of mutations (SUM, INS, DEL) - the user may choose to allow insertions, deletions and substitutions in the M-point mutation set;
- Open Prev Run - The application saves the results in a file, allowing to open previous runs without running the application again;
- Open Run - The user may save the results and insert them later in the GUI.

3.2 Optimal and suboptimal structures calculation

At the initial step, after starting the calculation by pressing "*Start*" in the GUI, the program calculates the dod-bracket representations of the optimal and suboptimal secondary structures of the provided RNA sequence. The optimal structure is calculated using `RNAfold` and the suboptimal structures are calculated using `RNAsubopt` with parameter e from the GUI. Both routines are availabe in the Vienna RNA package [4,5].

3.3 Filtering suboptimal secondary structures

Running RNAsubopt may lead to a huge number of, largely redundant, suboptimal folding solutions. In order to consider a small and diverse set of suboptimal structures, distant from the optimal structure, we use two filters. The first filter removes all suboptimal structures that are similar to the optimal one using $dist1$ input parameter as a distance threshold. After the first filter the suboptimal structures are sorted by their distance from the optimal structure. The second filter removes suboptimal structures that are close to each other.

Herein for each set of similar structures we proceed with only one representative that is the most distant from the optimal structure and also distant from all representatives of other sets. As an example, Table. 1 shows structures generated for an artificial RNA sequence:

`CCGGAAGAGGGGGACAACCCGGGGAAACUCGGGCUAAUCCCCCAUGU`
`GGACCCGCCCCUUGGGGUGUGUCCAAAGGGCUUUGCCCGCUUCCGG`

The table contains the optimal secondary structure and 6 suboptimal structures that passed the filtering stage with dist1 and dist2 thresholds = 30 and e = 15. The first row in the table corresponds to the optimal structure and the six rows below correspond to the suboptimal structures. The last column in the table shows the base-pair distance of the structure from the optimal structure. If more distant suboptimal structures are required, the e parameter in the GUI should be increased.

Structure	Dot-bracket representation	Distance
opt	(((((((.((((((...((((((....))))))....))))))...(((((.(((((....))))).))))))..((((...))))..))))))).....	0
sub1	.........(((.....)))..((((...(((((.((..(((....(((((.(((((....))))).))))))..)))..)))))))..)))).....	43
sub2	(((((((.(((((((((((((((((.....(((((((..........)))..)))))))))...))))..))))).....)))..)))))))....	39
sub3	(((((((.(((.((.....(((...((..(((.......)))..))...)))))((((.(((((....))))).))))..)).)))..)))))))....	39
sub4	(((((..((.(((.((((((((((....))))))......((......))....((((.(((((....))))).))))..)))).)))))))))....	36
sub5	.((((((.(((.((...((((((....))))))...((.(((((...((.....))...))))).))((((...))))..)).)))..))))))).	35
sub6	........((((((...((((((....))))))....))))))..((((....((((.(((((....))))).))))).....))))....	34

Table 1. Optimal and suboptimal structures of the artificial RNA sequence after filtering.

3.4 Collecting candidates for deleterious mutations

For each suboptimal structure that survived the filtering, we find mutations (insertions, deletions and substitutions depending on the user's choice) that may potentially convert the optimal secondary structure to a suboptimal one. To perform this task, we first calculate the start and end positions of all stems in the optimal and all suboptimal structures. For instance, the secondary structure `((((..(((....))))....))))` has two stems, with start/end at positions (1, 21)/(4, 18) and (7, 16)/(9, 14) respectively.

Next, we collect the mutations that stabilize the stems of the suboptimal structure and destabilize the stems of the optimal structure. The program searches for "good" places (indices) in the sequence for potential deleterious mutations. The "good" places for mutations are between stems of the suboptimal structure and in the middle of the stems of the optimal structure. This is true for substitutions, insertion and deletions. In the case of insertions that destibilize the optimal structure, it is possible to insert an exponential (in the size of M-mutation set) number of combinations of insertions in each index of the stem. To solve this problem we allow to insert only one mutation somewhere in the middle of each stem of the optimal structure. This is sufficient for the destabilization of the stem.

Example 1. Deleterious deletions:
`GAGUGUCGACUCCGCC` - RNA wildtype sequence
`((((....))))....` - Optimal structure
`..(.(.((....))))` - Suboptimal structure
In the example above the good indices for deletions are 4 and 6. Deletions U4 and U6 stabilize (elongate) the stems of the suboptimal structure, while mutation U4 also destabilizes (shortens) the single stem of the optimal structure. By introducing two point mutation U4-U6 into the wild type RNA sequence we obtain the following result:
`GAGGCGACUCCGCC` - RNA wildtype sequence
`(((....)))....` - Optimal structure
`..((((....))))` - Suboptimal structure
We can clearly see from the example that mutation U4-U6, consisting of two

deletions, converts the suboptimal structure to become more stable than the optimal one.

Example 2. Deleterious insertions and substitutions:
GAGGGUCGCCUCCGCGC - RNA wildtype sequence
((((....)))).... - Optimal structure
..(.(.((....))).) - Suboptimal structure
In this example, one of the good indices for substitution is 4 and one of the good indices for insertions is 15 (insertion between two stems from the narrow side). Substitution G4C connects two stems of the suboptimal structure and shortens one stem of the optimal structure. Insertion 15A connects two stems of the suboptimal structure. Finally, by introducing the two mutations G4C-15A into the wildtype RNA sequence we obtain the following result:
GAGCGUCGCCUCCGACGC - RNA wildtype sequence
(((......)))..... - Optimal structure
..((((((....)))))) - Suboptimal structure

3.5 Calculation of M-point mutation sets

At this stage, the program combines deletions, insertions and substitutions up to N sets of M mutations. The algorithm is implemented in a recursive way that searches all possible combinations of all types of mutations found in previous stage, but stops after reaching N mutations or all possible combinations of M-sets (if N is very large). For sequences longer than 150 bases, and values of M greater than 3, the number of all possible M-sets may be very large, much larger than N provided by the user.

Practically, it is sufficient to find a small amount of deleterious mutations (no more than 100) for laboratory experiments. In order to obtain a diversity of mutation types in the output, the algorithm combines single-point mutations randomly by choosing the calculation path through mutation types in a random way. To add to the diversity in the output, the algorithm uses all available diverse suboptimal structures for mutation analysis. For example, if the N provided by the user is 100 and the filtering stage produces 5 suboptimal structures, the algorithm will limit itself to 20 random deleterious M-point mutation sets for each suboptimal structure. The deleterious nature of each M-point mutation is validated, by checking that the mutation structure is distant enough from the structure of the wild type RNA sequence.

4 Results

A typical output of IndelsRNAmute is shown in Fig. 1. The results parameters for the sequence discussed in Section 3.3 are $dist1 = 30$, $dist2 = 30$, $e = 15$, $N = 100$, $M = 4$ and all three types of mutations. The output lists up to N M-point mutations, sorted by distance of their structure from the wildtype structure. The most deleterious mutations are listed first. Each row in the table

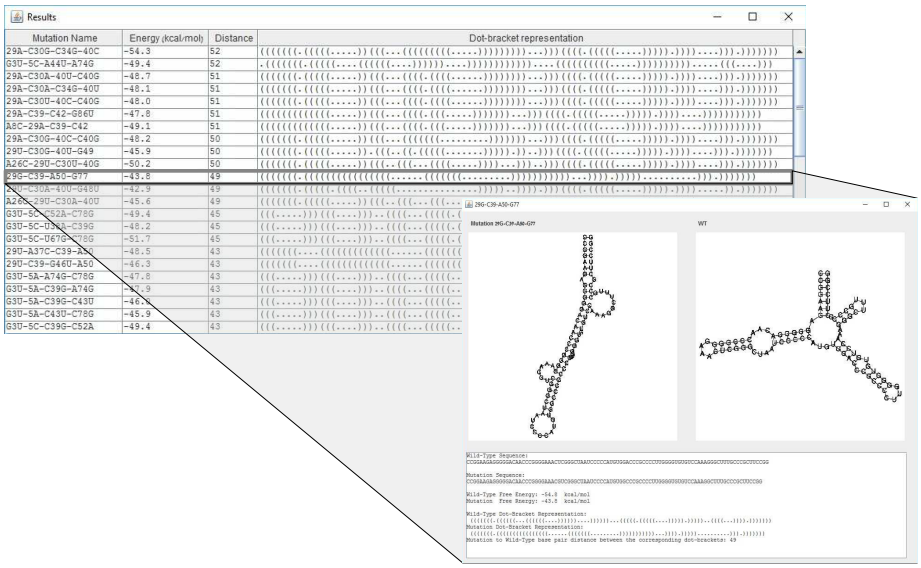


Fig. 1. Typical output of `IndelsRNAmute` and detailed structural analysis of mutation $29G - C39 - A50 - G77$.

includes the name of mutation, free energy, distance from wildtype RNA and the dot-bracket representation of its structure.

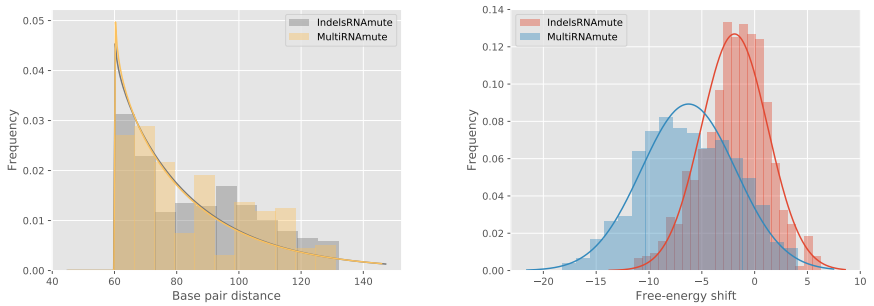
4.1 Interactive features

A user may further investigate a given mutation, by pressing on some row in the table to see more information about a specific mutation. For example, selecting the mutation $29G - C39 - A50 - G77$ will open the screen shown in Fig. 1. The structure of this mutation was obtained from the second suboptimal structure in Table. 1 by one insertion and three deletions. The insertion 29G and deletions C39 and G77 destabilize stems in the optimal structure, while deletion A50 both stabilizes the suboptimal structure by connecting two stems and destabilizes the stem of the optimal structure.

4.2 Analysis of mutations sets for random sequences

The importance of considering indels in the prediction of deleterious sets of mutations is illustrated by Fig. 2. In this analysis, 100 sets of $M = 5$ deleterious mutations were predicted for 10 random sequences of length 200nts, respectively in the presence and absence of support for indels.

As can be seen in Fig. 2.a, mutations in presence/absence of indels are equally deleterious, and induce distributions whose exponential fitted curves are virtually indistinguishable. However, as shown by Fig. 2.b, mutations sets including indels



a) BP distance to the WT MFE structure b) Energy distance to the WT MFE

Fig. 2. Distributions of base-pair distance (a) and energy distance to the WT MFE (b) of mutations sets ($M = 5$) produced for random sequences of length 200nts.

retain comparable free-energy as the wild type, while substitutions appear to induce a drastic decrease of the free-energy.

We interpret those results as indicative of the fact that mutations sets including indels, predicted by `IndelsRNAmute`, are much more geared towards the identification of deleterious sets of mutations, rather than a mere optimization of the thermodynamic stability of alternative structures. This interpretation suggests more realistic sets of mutations being produced by `IndelsRNAmute`. Indeed, due to kinetics effects, alternative structures associated with extreme shifts in MFE may not be reachable within folding landscapes in time comparable to adverse processes such as RNA degradation.

5 Conclusion

We present a method called `IndelsRNAmute` that extends the our `MultiRNAmute` method to predict insertion and deletion mutations in addition to substitutions. The additional advantage of the new method is its efficiency to find a predifined number of deleterious mutations. The running time of `MultiRNAmute` depends on the number of possible deleterious mutations, which may be very large and depend exponentially on number of mutations in the multiple-point mutation set, while the running time of IndelsRNAmute depends on N and only depends linearly on M. For example, for the same input, MultiRNAmute may run more than an hour predicting only substitutions, while `IndelsRNAmute` predicts 100 good mutations in a few seconds, and depending on the user's choice may include insertions, deletions and substitutions. All our mutation prediction methods were shown practical in predicting deleterious mutations in the P5abc subdomain of the *Tetrahymena thermophila* group 1 intron ribozyme, and in the 5BSL3.2 sequence of a subgenomic HCV replicon.

In future work we plan to implement k-medoids clustering, using medoids (centroids) as set representatives instead of our current filtering. However, the

main concern with such a clustering strategy is that finding the optimum k is time consuming and the user will have to provide k as an additional parameter in the GUI and some "good" suboptimal structures may be missed. In all distance calculations in our application we use linear base-pair distance for efficiency, but the method can be easily adopted to work with any other distance, like Hamming distance or tree edit distance.

References

1. Churkin, A., Barash, D.: RNAmute: RNA secondary structure mutation analysis tool. BMC bioinformatics **7**(1), 221 (2006)
2. Churkin, A., Barash, D.: An efficient method for the prediction of deleterious multiple-point mutations in the secondary structure of RNAs using suboptimal folding solutions. BMC bioinformatics **9**(1), 222 (2008)
3. Churkin, A., Barash, D.: A biologically meaningful extension of the efficient method for deleterious mutations prediction in RNAs: Insertions and deletions in addition to substitution mutations. In: International Symposium on Bioinformatics Research and Applications. pp. 174–178. Springer (2018)
4. Hofacker, I.L.: Vienna RNA secondary structure server. Nucleic acids research **31**(13), 3429–3431 (2003)
5. Hofacker, I.L., Fontana, W., Stadler, P.F., Bonhoeffer, L.S., Tacker, M., Schuster, P.: Fast folding and comparison of RNA secondary structures. Monatshefte für Chemie/Chemical Monthly **125**(2), 167–188 (1994)
6. Lorenz, R., Bernhart, S.H., Zu Siederdissen, C.H., Tafer, H., Flamm, C., Stadler, P.F., Hofacker, I.L.: ViennaRNA package 2.0. Algorithms for molecular biology **6**(1), 26 (2011)
7. Reuter, J.S., Mathews, D.H.: RNAstructure: software for RNA secondary structure prediction and analysis. BMC bioinformatics **11**(1), 129 (2010)
8. Shu, W., Bo, X., Liu, R., Zhao, D., Zheng, Z., Wang, S.: RDMAS: a web server for RNA deleterious mutation analysis. BMC bioinformatics **7**(1), 404 (2006)
9. Smith, D.B., Simmonds, P.: Characteristics of nucleotide substitution in the hepatitis C virus genome: constraints on sequence change in coding regions at both ends of the genome. Journal of molecular evolution **45**(3), 238–246 (1997)
10. Tang, S., Collier, A.J., Elliott, R.M.: Alterations to both the primary and predicted secondary structure of stem-loop iiic of the hepatitis C virus 1b 5' untranslated region (5' UTR) lead to mutants severely defective in translation which cannot be complemented intrans by the wild-type 5' UTR sequence. Journal of virology **73**(3), 2359–2364 (1999)
11. Waldispühl, J., Devadas, S., Berger, B., Clote, P.: RNAmutants: a web server to explore the mutational landscape of RNA secondary structures. Nucleic acids research **37**(suppl_2), W281–W286 (2009)
12. Yannakakis, M.: Node-and edge-deletion NP-complete problems. In: Proceedings of the Tenth Annual ACM Symposium on Theory of Computing. pp. 253–264. STOC '78, ACM, New York, NY, USA (1978). https://doi.org/10.1145/800133.804355, `http://doi.acm.org/10.1145/800133.804355`
13. You, S., Stump, D.D., Branch, A.D., Rice, C.M.: A cis-acting replication element in the sequence encoding the NS5B RNA-dependent RNA polymerase is required for hepatitis C virus RNA replication. Journal of virology **78**(3), 1352–1366 (2004)
14. Zuker, M.: Mfold web server for nucleic acid folding and hybridization prediction. Nucleic acids research **31**(13), 3406–3415 (2003)