A Relation between Natural Selection and Computational Complexity

Leonhard Sidl, Maximilian Faissner, Manuel Uhlir, Cristian A Velandia-Huerto, Maria Waldl, Hua-Ting Yao, Ivo L Hofacker, Peter F Stadler

A Relation between Natural Selection and Computational Complexity

Leonhard Sidl[1,2,‡], Maximilian Faissner[1,‡], Manuel Uhlir[1,‡], Cristian A. Velandia-Huerto[1,3], Maria Waldl[4], Hua-Ting Yao[1], Ivo L. Hofacker[1,2], Peter F. Stadler[4,5,1,6]

[1] Department of Theoretical Chemistry, University of Vienna, Währinger Straße 17, A-1090, Wien, Austria
[2] Research Group Bioinformatics and Computational Biology, University of Vienna, Währinger Straße 29, A-1090, Wien, Austria
[3] Center for Anatomy and Cell Biology, Medical University of Vienna, Schwarzspanier Straße 17, 1090, Wien, Austria
[4] Bioinformatics Group, Department of Computer Science and Interdisciplinary Center for Bioinformatics, University Leipzig, D-04107 Leipzig, Germany
[5] Max Planck Institute for Mathematics in the Sciences, D-04103 Leipzig, Germany
[6] Santa Fe Institute, Santa Fe, NM 87501, USA

E-mail: `studla@bioinf.uni-leipzig.de`

Abstract. Biological systems are widely regarded as performing computations. It is much less clear, however, what exactly is computed and how biological computation fits within the framework of standard computer science. Here we investigate a possible connection, focussing on the notion that biological systems may utilize fixed algorithms that imply subsets of solvable problem instances. In simple simulation experiments we show that differences in solvable instance sets may have an impact on evolution.

Keywords: Computational Complexity; Biological Computation; RNA Folding; Developmental Program; Genotype-Phenotype Map;

Submitted to: *J. Phys.: Complexity*

‡ Shared first authors

1. Introduction

Biological systems often are perceived to perform computations [1] that solve complex problems at low computational cost as a consequence of evolutionary adaptation [2, 3]. In particular, the central "information metabolism" of a cell, i.e., DNA replication, transcription, and translation of RNA to proteins is commonly described as a computation. Beyond these string-like transformations of encoded information, and of course, information processing in neural systems, it is usually far from obvious, however, *what* exactly a biological system is computing. Gene regulation has been described in terms of *regulatory circuits* akin to the hardware of a computing machine since the seminal work of Jacob and Monod [4, 5, 6]. Nevertheless, very little seems to be known about the *computational problem(s)* that this hardware is used to solve. An attempt to formalize regulatory mechanisms in terms of "I/O-maps" similar to logic gates was made in [7]. In the area of developmental evolution, the notion of a *developmental program* arose as a key concept [8, 9, 10]. Despite the general agreement that "life computes", the exact nature of natural computation thus is far from being well understood [11]. It seems fair to say that biological computation has been studied predominantly from an information-theoretic and/or dynamical systems perspective [12, 13, 3].

Connections to theoretical computer science so far primarily concern the computational power of natural systems, such as chemical reaction networks [14], DNA computing [15, 16] or the rewriting system arising from chemical modifications of histones [17]. Typically this line of work considers the biological systems as paradigm for an abstract model of computation. The main aim is to formally establish Turing universality or to prove limits on the computational power. Such general statements, however, do not answer the question how "difficult" the computational problems are that a biological system is solving.

Here we take a different point of view and explore whether there is a meaningful connection between a biological system's capabilities to solve a particular task and the concept used in theoretical computer science to describe and quantify the "complexity" of computational problem or of an algorithm used to solve it. In a nutshell, a computational problem is an infinite *set* of individual instances. An answer or solutions is defined for each of them, each together with an answer or solution. Problems are solved by algorithms, i.e., finite sequences of instructions, that produce the appropriate answer from (a representation of) a given instance. The problem in this setting is given *a priori* and is the object that is studied. As a example, consider the graph 3-coloring problem: The set of instances comprises all graphs G, and the solution is "true" if the vertices of G can be colored with three colors such that two adjacent vertices receive a different color. The complexity of the problem is then defined by the most efficient algorithm that can be devised to solve *all* instances of the problem. It is quantified as the asymptotic scaling of the algorithm's resource consumption for the most difficult, i.e., expensive instances, as a function of the size of the instance (here the number of vertices and edges of the graph G). The graph 3-coloring belongs to the class of NP-complete problems [18]. The exponential time hypothesis, which is commonly believed to be true, postulates that the best possible algorithm for this class of problems has exponential running

time [19]. Since problem complexity in this sense refers to the worst case, i.e., the most difficult instance, problems become easier when the set of instances is judiciously restricted. For example, 3-coloring can be computed in linear time, when only triangle-free planar graphs are considered [20].

Levinthal's paradox on protein folding observes that natural proteins fold very fast into their native conformations even though the conformation space of peptide chains is by far too large to be searched exhaustively [21]. This has led to the idea of folding funnels and more general energy landscapes that make it easy for real proteins to find the ground state [22, 23]. However, several computational models of protein folding result in NP-complete problems [24, 25], see also [26]. Moreover, there are many intrinsically disordered proteins (IDPs) that cannot attain a single stable three-dimensional structure [27]. It appears, therefore, that nature does not rely on a solution of protein folding problem for all instances. Instead, the problem is restricted, *by natural selection*, to a subset of aminoacid sequences (instances) that have suitable folding properties.

This simple observation leads to the hypothesis that biological evolution can – at least in some cases – avoid difficult instances. The scope of computational problems in biology, therefore is itself delineated by evolution. Given a developmental program, for example, evolution will lead to the avoidance of those genotypes (instances) that are "too difficult" in the sense that the machinery of development cannot unfold the DNA sequences correctly and in a timely fashion into the encoded phenotype. Such genotypes are unviable and hence will be selected against. This can be seen as excluding the corresponding instance from the specification of the computational problem.

In this contribution, we start to explore to what extent a connection between natural selection on problem complexity can be drawn. In Section 2, we cast the notions put forward in the introduction into the more formal framework computational complexity theory. Using the metaphor of developmental programs as an example, we then illustrate a possible connection between computation and selection. For this purpose, we build on seminal work on the properties of genotype-phenotype relationships from the 1990s [28, 29, 30, 31] and RNA folding as a computationally convenient toy model.

2. Problem Complexity

A *problem* $\mathbb{P}$ in the sense of computational complexity theory is an infinite set of concrete realizations, called *instances*, together with a (possibly empty) set of solutions $S(J)$ for every instance $J \in \mathbb{P}$. A particular instance J thus can be seen as the input for $\mathbb{P}$. The size of instance, $|J|$, measures the amount of information necessary to specify it. The idea is that computing $S(J)$ from J will require more effort for larger instances.

An *algorithm* $\mathscr{A}$ that solves $\mathbb{P}$ takes any J as input and computes $S(J)$ using a certain amount computational resources, which we denote by $\mathrm{cost}(\mathscr{A}, J)$. We denote by $\mathfrak{A}[\mathbb{P}]$ the set of algorithms that solve $\mathbb{P}$, i.e., computes $S(J)$ for all $J \in \mathbb{P}$. Algorithms, in turn, are specified with respect to some model of computation, such as the Turing machines conceptually underlying our digital computers. A plethora of different models of computation have been

studied, some of which, such as DNA computing [15, 16], are derived from biological processes. Many of these are computationally universal, in the sense that they can simulate Turing machines and thus compute everything that can be computed by a Turing machine [32, 33]. For the purpose of this contribution, the concrete choice of an underlying model of computation is not relevant, although the specification and comparison of algorithms requires that we use a fixed model.

The task most commonly encountered is to determine the complexity of a given problem $\mathbb{P}$. There are two ways of approaching this question: The most direct approach is to explicitly describe a specific algorithm $\mathscr{A}$, prove that it correctly solves $\mathbb{P}$, and to analyse $\mathscr{A}$ to determine the computational effort $\mathrm{cost}(\mathscr{A},J)$ required run $\mathscr{A}$ for any instance J of size $|J|=n$. Alternatively, one may proceed by showing that a problem $\mathbb{P}$ is at least as complex (or at most as complex) as another problem $\mathbb{P}^*$ for which complexity results are known. This route is taken in particular to establish that a problem is NP-complete [18].

In order to abstract from concrete computational devices, programming languages, compilers, and the concrete encoding of instance J, one usually is interested in the scaling of an upper bound of the worst case computational cost as a function of the size $|J|$ of the input, i.e., a function $c_{\mathscr{A}}(n)$ that satisfies

$$c_{\mathscr{A}}(n) \geq k \max_{J:|J|=n} \mathrm{cost}(\mathscr{A},J) \tag{1}$$

for all $n \geq n_0$, where $k>0$ and $n_0>0$ are arbitrary constants. One says that $\mathscr{A}$ requires $\mathscr{O}(c(n))$ effort, or simply that $\mathscr{A}$ is in complexity class $\mathscr{O}(c(n))$. The complexity of a problem is then given by the minimal worst-case complexity of any algorithm that solves $\mathbb{P}$, i.e.,

$$c_{\mathscr{P}}(n) = \inf_{\mathscr{A}\in\mathfrak{A}[\mathbb{P}]} c_{\mathscr{A}}(n) \tag{2}$$

Restriction of a problem makes a problem easier in general, since $\mathbb{P}' \subseteq \mathbb{P}$ implies $c_{\mathscr{P}'}(n) \leq c_{\mathscr{P}}(n)$. As mentioned in the introduction, it is an active field of research to determine subclasses of difficult problems that admit efficient solutions, in particular in the area of combinatorial optimization problems on graphs.

Information processing in biological systems can be expected, at least in many cases, to be result of a given algorithm $\mathscr{A}$ that is "hardcoded" e.g. in a gene-regulatory network [34]. The natural question then is, which instances are solvable by the given algorithm $\mathscr{A}$. This questions has been addressed at least for some (classes of) algorithms in the past. Probably the best-understood case is the *canonical greedy algorithm* $\mathscr{G}$, which operates on constrained linear optimization problems. More precisely, the set $\mathbb{P}$ instances of the form $J=(X,w,\mathfrak{F})$ where X is a finite set, $\mathfrak{F}\subseteq 2^X$ is a set of "allowed" subsets, and $w: X\to\mathbb{R}^+$ is a weight function. The greedy algorithm $\mathscr{G}$ sorts the set X in decreasing order w.r.t. w and initialized an empty solution set B. Then it tests, for all $x\in X$ in this order, whether $B\cup\{x\}\in\mathfrak{F}$. If so, x is added to B, otherwise x is discarded. $\mathscr{G}$ thus attempts to find a subset $B\in\mathfrak{F}$ that maximizes the score $\sum_{x\in B} w(x)$. Naturally, one asks under what conditions on $J=(X,w,\mathfrak{F})$ we can be sure that $\mathscr{G}$ correctly solves the problem of maximizing w. A famous result states that this is case for an arbitrary choice of weights if and only if the set systems $(X,\mathfrak{F})$ is a certain

generalization of a matroid [35]. On the other hand, if X is the vertex set of a graph $G = (X, E)$ and $\mathfrak{F}$ are the independent sets of the graph G (that its, the subsets of G that contain no pair of adjacent vertices), then the correct solutions are that of the weighted independent vertex set problem. This optimization problem is well-known to be NP-hard [36] and in general is not solved correctly by $\mathscr{G}$. Moreover, restricting the instance set e.g. to P_5-free graphs, it is possible to find polynomial-time algorithms [37], which are, however, entirely unrelated to $\mathscr{G}$.

3. RNA folding as a computational model of evolution

Our discussion so far suggests asking whether algorithms that correctly solve different sets of instances can have an impact on evolution. We use here the setting of developmental programs, i.e., algorithms $\mathscr{A}$ that take a genotype J as an input and produce a phenotype as an output. For simplicity, we assume that only correctly constructed phenotypes $S(J)$ are viable. It might be more natural to ask whether an algorithm $\mathscr{A}$ computes a sufficiently close approximation to $S(J)$ instead of just distinguishing between correct and incorrect answers. Many hard optimization problems with corresponding NP-complete decision problems also do not admit efficient accurate approximations [38]. At least for an initial study into the interplay of problem complexity, algorithm capabilities, and their evolutionary consequences, we therefore restrict ourselves to the simplest case and ignore details such as the quality of an approximation. In more realistic models, we expect of course that fitness will depend on some measure of approximation quality instead of just being a yes/no answer.

Development is itself a complex problem, and there is at present no complete model of even the simplest developmental process that would link a genotype directly to a phenotype. Structure formation of biopolymers, however, has been used extensively as a convenient, computationally tractable toy model of genotype-phenotype (GP) maps. This is in particular the case for RNA secondary structure, for which the GP maps $J \mapsto S(J)$ for fixed $n = |J|$ have been studied extensively in the 1990s [28, 29]. We therefore (re)use the paradigm here as well.

For every RNA sequence J of length $|J|$ over the alphabet $\{A, C, G, U\}$ of nucleotides we define the solution $S(J)$ as the minimum free energy secondary structure, i.e., the contact map, as predicted by `RNAfold` [39]. The evolutionary dynamics of an evolving RNA population is essentially determined by the underlying GP map.

More precisely, one observes qualitatively the same evolutionary dynamics dominated by diffusion on neutral networks and rapid adaptation on the transitions between neutral networks, independent of particular choice of a fitness function that evaluates phenotypes only [40]: The GP map of RNA secondary structures exhibits a high degree of neutrality, admitting extensive, essentially connected, neutral networks of sequences the fold into a common structure. On the other hand, the neutral networks of any two abundant structures almost touch somewhere in sequence space, i.e., for any two structures ϕ_1 and ϕ_2 there are very similar sequences I_1 and I_2 with $S(I_1) = \phi_1$ and $S(I_2) = \phi_2$. As a consequence populations show a diffusive motion on the neutral network, while off-network mutants explore previously unseen structures at a constant rate [41, 42]. This gives rise to evolutionary trajectories in

Table 1: Summary of characteristics of folding algorithm characteristics and shape prediction from 100 000 uniformly and randomly sampled sequences of length 100nts. Different algorithms access vastly different number of distinct RNA structures, quantified here as coarse-grained structure. The columns show, in the order, the number of different predicted shapes, the number of different predicted shapes covered by `RNAfold` prediction, and the number of correctly predicted shape of each input sequence.

	Algorithm characteristics			**Coarse-grained structure prediction**		
Folding algorithm	Co-transcription	Full length	Energy model	# diff. predicted shapes	# diff. shapes in `RNAfold` prediction	# correct shape predictions
RNAfold		✓	✓	15 288	15 288	100 000
Look behind fold	✓			6	3	7
Basic co-fold	✓			693	679	333
Best helix co-fold	✓		✓	10 987	8 169	14 921
Folding rule		✓	✓	6 172	5 238	11 507
Beam search		✓	✓	14 557	9 941	31 180

phenotype space that show punctuated equilibria, i.e., long periods of stagnation with rapid intermittent innovations [41, 31].

In this setting we can explore the effect of restricting folding to a subset of sequences that produce the correct structure $S(I)$ with simpler heuristic folding algorithms $\mathscr{A}(I)$ that works correctly only on a (small) subset $\mathbb{P}_{\mathscr{A}}$ of $\mathbb{P}$. To this end we introduce several simple heuristics that emphasize different aspects of RNA folding. We then ask how the evolution is influenced by the different scopes of an algorithm $\mathscr{A}$ and by costs associated with the choice of $\mathscr{A}$.

In the next section we describe several variants of simple RNA folding algorithms $\mathscr{A}$. Each is designed to accept an arbitrary sequence as input. We *define* the minimum free energy structures computed by the `ViennaRNA` [39] as the ground truth, i.e., as the correct solution $S(I)$ for each input sequence. The problem solved by an alternative folding algorithm is therefore

$$\mathbb{P}_{\mathscr{A}} \coloneqq \{I | \mathscr{A}(I) = S(I)\} \tag{3}$$

That is, the subset of RNA sequences for which $\mathscr{A}$ produces the same structure as `ViennaRNA`. Sequences I with $\mathscr{A}(I) \neq S(I)$ are treated as failure and consider such an instance (sequence) as not viable. We then proceed to investigate to what extent this affects the structure of the genotype-phenotype map and eventually evolutionary processes

4. Alternative RNA Folding Algorithms

All algorithms described in this section use the energy model, if applicable, for evaluating secondary structures and enforce the constraint that hairpin loops contain at least three unpaired bases. All energy computations were performed using the `ViennaRNA` package [39] with the Turner2004 parameters [43] to ensure consistency. We investigate five simple folding algorithms that can belong to two broad classes. All of them are inspired by the dynamical process of RNA secondary structure formation rather than exact solution by means of dynamic programming [44], which in contrast is quite far removed from the physics of the folding process. Tab. 1 summarizes the characteristics of each folding algorithm.

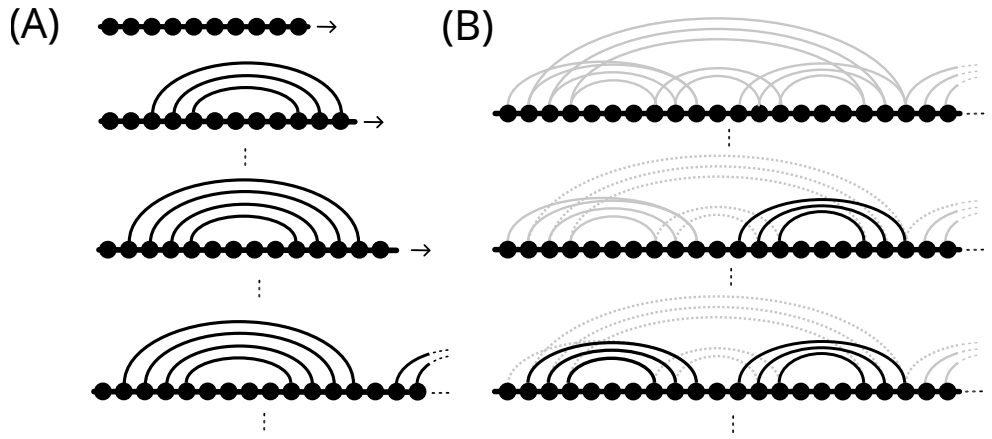


Figure 1: Greedy-like approached to RNA folding either simulate co-transcriptional folding (A) or operate on the complete sequence (B). In the first case, structures are formed while the sequence grows from the 5' to the 3' end and subsequently kept in place. *Look Behind Folding* (A) closes a helix as soon as a see comprising three base pairs become available. The RNA *Folding Rule* (B) stepwisely inserts the most stable helix that does not conflict with previously inserted ones.

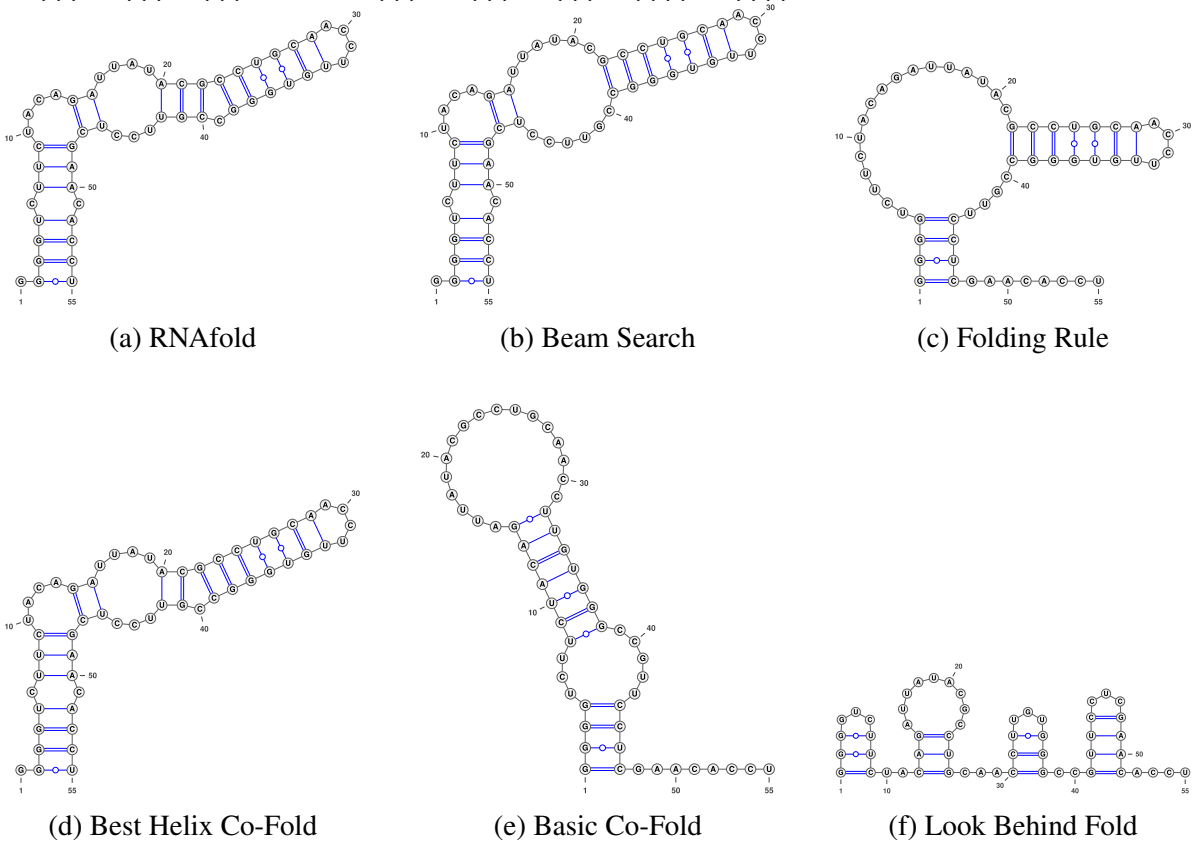
The first group of algorithms is intended to approximate co-transcriptional folding. Base pairs are formed already as the RNA sequence grows from the 5'-end towards the 3'-end, Fig. 1A. The folding process is achieved in two stages. In the first stage, the model tries to pair newly transcribed base(s) to form seed helices, which are then served as reference in the next stage to complete the folding. We consider three variants of this approach:

- *Look behind folding* introduces one transcribed base at each step and pairs with the closest complementary base if a size three seed helix can be formed. Look behind folding thus is a greedy-like approach that tends to form short helices as soon as possible. Once seed helices are identified, the algorithm attempts to extend the helix outwards by adding additional base pairs on each side. The sequence is also zipped inwards by identifying base pair-stacks in between the seeds.
- *Basic co-transcriptional folding* moves along the sequence from 5' to 3' and, for each position, searches for a complementary base by again moving over all bases transcribed at this point. The first base pair found by this procedure which can be extended to a helix of length at least six is chosen and the detected helix is formed. Such a locally stable helix is then frozen in placed and cannot be replaced by an energetically more beneficial structure later on. Following this, the folding is completed by forming shorter helices (at least four base pairs long) using the same procedure as above. In this second stage, special care is taken to avoid the formation of pseudoknots.
- *Best helix co-transcriptional folding* is a variation of the previous algorithm. However, this algorithm reveals three bases in a single step. In addition, instead of forming the first detected helix that fulfills the length requirement, all viable helices are first enumerated. The helix which has the most beneficial effect on the energy of the complete structure

```
GGGGUCUUCUACAGAUUAUACGCCUGCAACCUUGUGGGCCGUUCCUCGAACACCU
.((((.(((....((....(((((((((....)))))))..)))...)))))).))))   RNAfold

.((((.(((....((......(((((((....)))))))).....)))))).))))   Beam search
((((................(((((((....)))))))....))))........   Folding rule

.((((.(((....((....(((((((((....)))))))..))...)))))).))))   Best helix co-fold
((((...(((((((.................)))))))).....))))........   Basic co-fold
(((...)))..(((.........)))...(((...)))..((((...))))....   Look behind fold
```



(a) RNAfold (b) Beam Search (c) Folding Rule

(d) Best Helix Co-Fold (e) Basic Co-Fold (f) Look Behind Fold

Figure 2: Predictions of the different algorithms for a randomly selected input sequence I. **Top.** Structures are displayed in dot-bracket (Vienna) notations: each base pair is shown as a matching pair of parentheses, unpaired nucleotides are shown as dots. In this case, none of the simplified algorithms correctly computes the solution $S(I)$ defined by `RNAfold`. **Below.** Conventional secondary structure drawings.

is then chosen and formed. The energy is calculated using the `ViennaRNA` package with default parameters. In contrast to *Basic co-transcriptional fold*, *Best helix co-fold* uses `RNAfold` in the second phase. Here, the helices inserted in the first pass are used as hard constraints and thus persist in final structure [45]. The main difference to the previous algorithm is that we reveal several new bases at the same time and then finds the most stable helix using the new bases. This avoids traps formed by long helices that are globally unfavorable.

The second groups of model folds the entire input sequence and aims to efficiently identify a near-optimal secondary structure by heuristically exploring a subset of potential

conformations, Fig. 1B. Starting from an open chain as the initial structure state, the folding progresses by moving to the next conformation that provides the greatest improvement in the free energy, based on simple rules such as adding a base pair or a helix. At each step, the algorithm selects the most energetically favorable addition without allowing the introduction of crossing base pairs (pseudoknots). Here, we consider two variants:

- The *RNA folding rule* was proposed in [46] as a greedy approximation to kinetic folding. At each step, the algorithm adds the next most favorable helix to the structure. Once a helix is added, the algorithm does not reconsider previous choices or allow for backtracking, and the process terminates when no further helix can be added without increasing the total folding energy.
- *Beam search folding* is similar to *RNA folding rule* but only one base pair is added at each step. In addition, it employs a beam search strategy to track suboptimal intermediate states. At each step, the k-best candidate structures are stored among all possible candidates generated from the previously stored k-best states. If $k = 1$, the method reduces to a greedy algorithm based on individual base pairs, similar to the RNA folding rule. Since a greedy approach is unlikely to yield globally optimal structures, retaining k suboptimal intermediates significantly increases the probability of identifying the most energetically favorable structure. However, increasing k improves solution quality at the cost of higher computational complexity, so selecting k involves a trade-off between accuracy and running time.

A closer inspection of the structures produced by the different algorithms quickly reveals a few general trends indicating that they focus on different parts of shape space. Fig. 2 shows the predicted structures of a randomly chosen instance I. The baseline structure given by `RNAfold` combines both long and short range interactions with a plethora of complex features such as bulges and interior loops. Contrasting that, the simpler alternative algorithms tend to lack such complex structures and are marked by longer and locally more stable helices. This is not surprising, since all except *Beam Search* utilize the paradigm of greedy algorithms and may easily get trapped in local minima when finding new helices. A particular example here is that *Folding rule* forms the helix at $(1,47)$ instead of $(2,55)$ because of the greedy choice and results a different structure. Despite the similarities of *Basic co-transcriptional folding* and *Best helix co-transcriptional fold*, the predicted structures at times differ drastically, and in the example in Fig. 2.

5. Simulation results

Genotype-Phenotype Maps. In order to get a first impression of the effect to the algorithm we considers the "structure density surface", which measure the effect of mutation randomly placed in the sequence (Hamming distance) on the differences in the predicted structures, quantified as the number of base pairs by which the predicted secondary structures differ (base pair distance). Fig. 3 shows that there are substantial quantitative differences, even though the overall shape of the distributions remain qualitatively the same: With very few

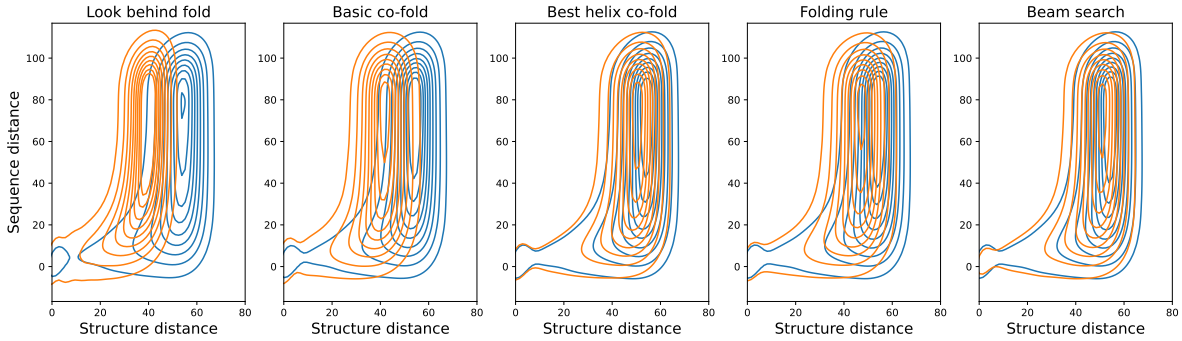


Figure 3: Structure density surfaces of different folding algorithms (orange) compared with the one of `RNAfold` (blue), obtained from 1000 reference sequences with 10 mutants for each Hamming distance class. Structural distance is quantified as the symmetric difference of the sets of base pairs.

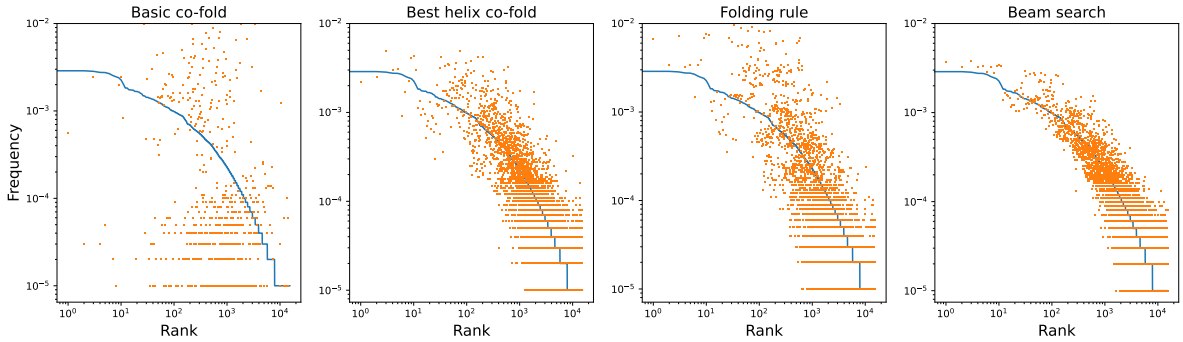


Figure 4: Frequency distribution of the shapes as a function of the frequency rank of shapes obtained from `ViennaRNA` computations. While folding algorithms that use the full folding model as a component to "fill in" structures qualitative follow the blue `ViennaRNA` curve, albeit with appreciable scatter, the shapes distributions are drastically different for basic co-fold and best helix co-fold.

mutations, a large fraction of the structures is retained perfectly. At the same time, a very small distance in sequence may already drastically change the structure. Moreover, already at moderate sequence distance, i.e., Hamming distances of about 20% of the sequence length, there is little resemblance between the structures of the original and the mutated sequences.

For further analysis we used a coarse grained definition of the structures since the fraction of secondary structures for which the simple heuristic algorithms exactly reproduces the `ViennaRNA` prediction very rapidly declined with sequence length. We therefore consider the shape of the RNA as defined in [47]. We chose this form of course graining because it is provided by `ViennaRNA` package [48]. As an example, the shape of *Best Helix co-transcriptional folding*'s prediction in Fig. 2 is the same as the one of `RNAfold`. The space of coarse grained structures produced by the different algorithms differs quite drastically, Tab. 1. For instance, *look behind folding* produces only six distinct shapes, some of which were not encountered at all in the `ViennaRNA` baseline.

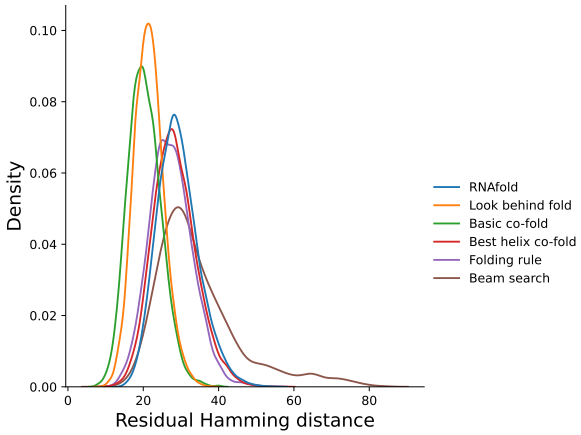


Figure 5: Neutral paths as a function of the algorithm $\mathscr{A}$. Shown is the distribution of the residual Hamming distances of a walk for a given pair of initial and target sequence. The distribution is obtained for 100 random target for each of 50 random initial sequences.

A more detailed view of the frequency distribution of shapes can be found in in Fig. 4. We observe that distributions are quite similar to `RNAfold` for algorithms that use `RNAfold` to complete structures subject to a set of base pairs computed by another method. In contrast, co-transcriptional folding rules and the simple *Folding Rule* show much more scatter. It is also worth noting that despite these differences, we observe similar distributions of structure frequencies with a power-law tail. Individual shapes, however, may differ drastically in their frequency, and thus affect their evolutionary accessibility [29, 31, 49].

Neutral Networks. As a more direct measure of the evolutionary impact of $\mathscr{A}$ we considered the length distribution of neutral paths towards a target, see also [29]. Here, we compute these paths as follows: Given a start sequence I_0 and a target sequence I^*, the unpaired base and base pair in the structure $S(I_0)$ are determined and of each unpaired base or base pair the mutation(s) distinguishing I_0 and I^* are listed. A walk in sequence space is generated from a random permutation of this list by accepting an exchange of a base or pair of bases if this preserves the initial secondary structure $S(I_0)$. Otherwise, the step is rejected and appended to the list. Mutation towards the target is enforced after certain equal distance (to target) mutations are made. The walk terminates with a sequence I' if no further structure-preserving mutation from the list can be found. To extent to which the walk approached the target I^* is quantified as the Hamming distance between I' and I^*. We then record the distribution of the final distance to the target. The longer the resulting path, the closer one can approach an arbitrary target sequence and hence the more extensive is the neutral network for the initial sequence. Since the target sequence I^* will in general not be compatible with the query structure $S(I_0)$, in general a sizable residual distance remains.

This walk measure used in Fig. 5 can be seen as compromise between the definition of neutral walks in [29] (which did not prescribe a target sequence I^* but aimed maximizing the distance from the initial sequence I^*) and the notion of covering radii in [29], where instead of single target sequence I^* a minimum is taken over sequences I^* folding into the same target

structure $S^* = S(I^*)$.

Simulation of Evolutionary Trajectories. Taken together, we observe that different algorithms result in significant quantitative differences in key features of the accessible GP-map for RNA, affecting neutrality, the extent of neutral networks, and the relative frequencies of correctly predicted RNA shapes. In order to see the impact of the folding algorithms on an evolutionary process more directly we simulated a population of RNAs that replicates with rates proportional to a fitness function

$$f(I) = \frac{1}{0.01 + d(I, \text{tRNA})/n}, \tag{4}$$

that quantifies the structural distance to a prescribed target structure. Here the target is tRNA-Val(TAC) from *Escherichia coli K-12* with a length of $n = 76$. Such simulations were used in [31] to investigate the nature of structural innovations and to understand the interplay of diffusive behaviour on neutral networks in relation of adaptation to structural innovations with superior fitness. Here we are primarily interested in the differences of time that alternative folding rules need to reach the target. The structure distance d is again computed as the symmetric difference of the sets of base pairs. Simulations were stopped when a fixed fraction of the population conformed to the target structure. As a stopping criterion for the simulation we use that 50% of the population conforms to the target structure.

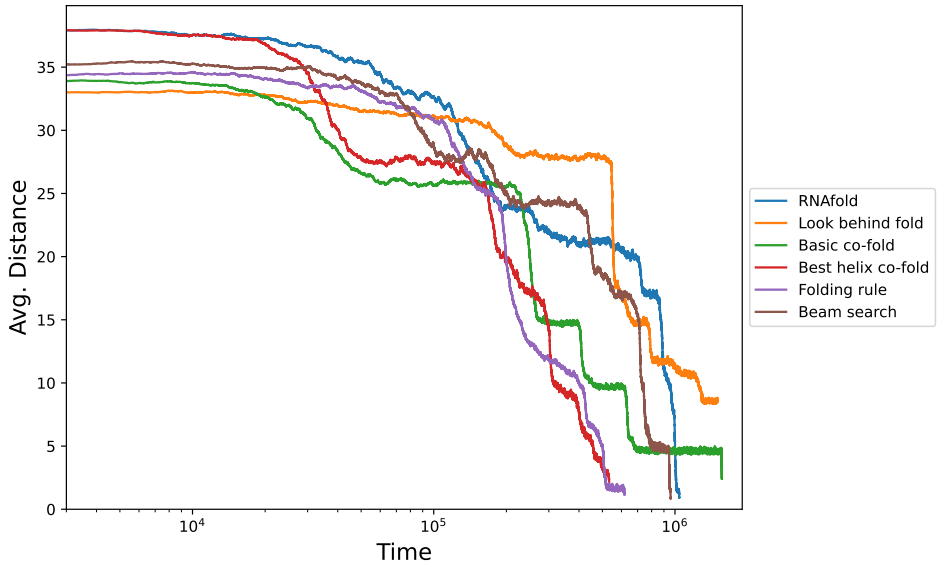


Figure 6: Evolution towards a target tRNA structure. Initial populations of 1000 sequences, with replication accuracy $p = 0.999$ per nucleotide. The simulation stops once more than half of the population folds into the predefined tRNA target, and for *Look behind fold*, the simulation was stopped at 1.5×10^6 time units. Curves represent the average structural distance of the population to the target structure.

Fig. 6 shows the population averages of the distance to the target structure for different algorithms. We observe that the trajectories differ in length, although all of them show a

step-wise behavior. The latter indicates that intermediate structures are found that persist in the population until a superior structure is accessed and selected. Surprisingly, several of the simpler algorithms find the tRNA target much faster than `RNAfold`, suggesting that a more capable algorithm may not always translate to an evolutionary advantage. In detail, we identified algorithms that take less than $0.6\times$ the time of `RNAfold`: *Best Helix co-transcriptional folding* and *Folding rule*. Similar times to `RNAfold` were achieved by *Beam search* ($0.9\times$) and *Basic co-fold* ($1.5\times$), while slow adaptation folding was found in *Look behind fold*, which required an additional time-dependent stopping rule (simulation time reached 1.5×10^6 units) due to innovation events never conforming to the target structure for at least 50% of the population.

6. Discussion

Taken together, our simple simulation results show that the choice of the folding algorithm has an impact on quantitative properties of the genotype-phenotype map. We may conclude therefore, that the scope of $\mathscr{A}$ in systems such as development can be expected to have a substantial impact on the topology of the genotype-phenotype map. This in turn translates into possibly drastic effects on evolutionary trajectories [29, 31, 50].

Many aspects remain to explore. In particular, it will be interesting to investigate models in which the algorithm $\mathscr{A}$ is subject to variation and selection. Does better, in the sense of more accurate, or more general computation, already provide a selective advantage? If so, what exactly are the trade-offs between generality of the problem sets and the cost of execution? After all, we expect that in general an algorithm $\mathscr{A}'$ that solves a proper superset of $\mathbb{P}[\mathscr{A}]$ will also require more resources.

Irrespective of the resource consumption of the computation itself, our simulation results suggest that the most powerful algorithm does not guarantee the most efficient adaptation. In the case of development, which we use as an example here, early failure may not incur dramatic fitness cost, while a restricted space of possible results may lead to faster exploration on neutral networks. In this context we expect that development, where constraints are largely internal, and signal processing where problem instances are posed by an external environment, may behave very differently. The trade-offs involved certainly deserve a much more detailed investigation in future work.

From a biological point of view, we will eventually have to return to the question how one can determine the pertinent algorithms in detail, and how their resource consumption and input dependence could be quantified. It would appear that simplified computational models, including but not limited to the RNA folding models used here, can provide valuable insights, even though they probably fall short of being particularly realistic models of computation in biological systems.

Data Availability

The implementation of folding approaches and simulations can be found at `https://github.com/ViennaRNA/VIECPLX`.

Acknowledgements

Research in the Stadler lab is supported by the German Federal Ministry of Education and Research BMBF through DAAD project 57616814 (SECAI, School of Embedded Composite AI). PFS gratefully acknowledges discussions with Gülce Kardeş and David Wolpert in spring 2023 and feedback received on some of these ideas during the CSH Workshop *Trade-offs between thermodynamic cost, intelligence and fitness in living organisms*. HTY is funded by Austrian Science Fund (FWF), grant no. I 4520. MF, MU and MW are funded by the Novo Nordisk Foundation (grant NNF21OC0066551 "MATOMIC"). CAVH was funded by FWF grant number SFB F80-01. LS was funded by the FWF project number I 6440-N.

References

[1] Melanie Mitchell. Ubiquity symposium: Biological computation. *Ubiquity*, 2011:3, 2011.

[2] Christopher P Kempes, David Wolpert, Zachary Cohen, and Juan Pérez-Mercader. The thermodynamic efficiency of computations made in cells across the range of life. *Philos Trans A Math Phys Eng Sci*, 375(2109):20160343, 2017.

[3] David H. Wolpert, Chris Kempes, Peter F. Stadler, and Joshua A. Grochow, editors. *The Energetics of Computing in Life and Machines*. SFI Press, Santa Fe, NM, 2019.

[4] François Jacob and Jacques Monod. Genetic regulatory mechanisms in the synthesis of proteins. *J. Mol. Biol.*, 3(3):318–356, 1961.

[5] Douglas H Erwin and Eric H Davidson. The evolution of hierarchical gene regulatory networks. *Nat Rev Genet*, 10(2):141–148, 2009.

[6] Harold D. Kim, Tal Shay, Erin K. O'Shea, and Aviv Regev. Transcriptional regulatory circuits: Predicting numbers from alphabets. *Science*, 325:429–432, 2009.

[7] David C. Krakauer, Lydia Müller, Sonja J. Prohaska, and Peter F. Stadler. Design specifications for cellular regulation. *Th. Biosci.*, 135:231–240, 2016.

[8] G Oster and P Alberch. Evolution and bifurcation of developmental programs. *Evolution*, 36(3):444–459, 1982.

[9] Detlev Arendt, Jacob M. Musser, Clare V. H. Baker, Aviv Bergman, Connie Cepko, Douglas H. Erwin, Mihaela Pavlicev, Gerhard Schlosser, Stefanie Widder, Manfred D. Laubichler, and Günter P. Wagner. The origin and evolution of cell types. *Nat Rev Genet*, 17:744–757, 2016.

[10] Somya Mani and Tsvi Tlusty. A topological look into the evolution of developmental programs. *Biophys. J.*, 120(19):4193–4201, 2021.

[11] Dominique Chu, Mikhail Prokopenko, and J. Christian J. Ray. Computation by natural systems. *Interface Focus*, 8(6):20180058, 2018.

[12] Keyan Zahedi and Nihat Ay. Quantifying morphological computation. *Entropy*, 15:1887–1915, 2013.

[13] Neil Dalchau, Gregory Szép, Rosa Hernansaiz-Ballesteros, Chris P. Barnes, Luca Cardelli, Andrew Phillips, and Attila Csikász-Nagy. Computing with biological switches and clocks. *Natural Computing*, 17(4):761–779, 2018.

[14] François Fages, Guillaume Le Guludec, Olivier Bournez, and Amaury Pouly. Strong Turing completeness of continuous chemical reaction networks and compilation of mixed analog-digital programs. In J. Feret

and H. Koeppl, editors, *Computational Methods in Systems Biology. CMSB 2017*, volume 10545 of *Lect. Notes Comp. Sci.*, pages 108–127, Cham, 2017. Springer.

[15] David Soloveichik, Georg Seelig, and Erik Winfree. DNA as a universal substrate for chemical kinetics. *Proc. Natl. Acad. Sci. USA*, 107:5393–5398, 2010.

[16] Lulu Qian, David Soloveichik, and Erik Winfree. Efficient Turing-universal computation with DNA polymer. In Yasubumi Sakakibara and Yongli Mi, editors, *DNA Computing and Molecular Programming*, volume 6518 of *Lect. Notes Comp. Sci.*, pages 123–140, Berlin, Heidelberg, 2011. Springer.

[17] B. Bryant. Chromatin computation. *PLoS ONE*, 7:e35703, 2012.

[18] Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, editors, *Complexity of Computer Computations*, pages 85–103. Plenum, New York, 1972.

[19] Russell Impagliazzo and Ramamohan Paturi. The complexity of k-SAT. In *Proc. 14th IEEE Conf. on Computational Complexity*, pages 237–240, 1999.

[20] Zdeněk Dvořák, Ken-Ichi Kawarabayashi, and Robin Thomas. Three-coloring triangle-free planar graphs in linear time. *ACM Trans. Algorithms*, 7(4):41, 2011.

[21] Cyrus Levinthal. Are there pathways for protein folding? *J. Chim. Phys.*, 65:44–45, 1968.

[22] R Zwanzig, A Szabo, , and B. Bagchi. Levinthal's paradox. *Proc. Natl. Acad. Sci. USA*, 89:20–22, 1992.

[23] Martin Karplus. The Levinthal paradox: yesterday and today. *Folding and Design*, 2:S69–S75, 1997.

[24] R Unger and J Moult. Finding the lowest free energy conformation of a protein is an np-hard problem: proof and implications. *Bull Math Biol*, 55:1183–1198, 1993.

[25] B Berger and T. Leighton. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. *J Comput Biol*, 5:27–40, 1998.

[26] Christophe Guyeux, Nathalie M-L Côté, Jacques M Bahi, and Wojciech Bienia. Is protein folding problem really a NP-complete one? First investigations. *J Bioinform Comput Biol*, 12:1350017, 2014.

[27] Rakesh Trivedi and Hampapathalu Adimurthy Nagarajaram. Intrinsically disordered proteins: An overview. *Int J Mol Sci.*, 23(22):14050, 2022.

[28] Walter Fontana, Peter F Stadler, Erich G Bornberg-Bauer, Thomas Griesmacher, Ivo L. Hofacker, , Thomas Griesmacher, Manfred Ivo L Tacker, Pedro Tarazona, Edward D Weinberger, and Peter Schuster. RNA folding landscapes and combinatory landscapes. *Phys. Rev. E*, 47:2083–2099, 1993.

[29] Peter Schuster, Walter Fontana, Peter F Stadler, and Ivo L Hofacker. From sequences to shapes and back: A case study in RNA secondary structures. *Proc. Roy. Soc. Lond. B*, 255:279–284, 1994.

[30] Walter Fontana and Peter Schuster. Shaping space. The possible and the attainable in RNA genotype-phenotype mapping. *J. Theor. Biol.*, 194:491–515, 1998.

[31] Walter Fontana and Peter Schuster. Continuity in evolution. On the nature of transitions. *Science*, 280:1451–1455, 1998.

[32] Udi Boker and Nachum Dershowitz. How to compare the power of computational models. In S. Barry Cooper, Benedikt Löwe, and Leen Torenvliet, editors, *New Computational Paradigms*, volume 3526 of *Lect. Notes Comp. Sci.*, pages 54–64, Berlin, Heidelberg, 2005. Springer.

[33] Naresh R. Shanbhag, Subhasish Mitra, Gustavode de Veciana, Michael Orshansky, Radu Marculescu, Jaijeet Roychowdhury, Douglas Jones, and Jan M. Rabaey. The search for alternative computational paradigms. *IEEE Design and Test*, 25:334–343, 2008.

[34] Eric H. Davidson. *The Regulatory Genome: Gene Regulatory Networks In Development And Evolution*. Academic Press, Burlington, MA, 2006.

[35] Paul Helman, Bernard M. E. Moret, and Henry D. Shapiro. An exact characterization of greedy structures. *SIAM J. Discrete Math.*, 6, 1993.

[36] M. R. Garey and D. S. Johnson. "Strong" NP-completeness results: Motivation, examples, and implications. *J. ACM*, 25(3):499–508, 1978.

[37] Daniel Lokshantov, Martin Vatshelle, and Yngve Villanger. Independent set in P_5-free graphs in polynomial time. In Chandra Chekuri, editor, *SODA '14: Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 570–581, Philadelphia, PA, 2014. SIAM.

[38] David Zuckerman. On unapproximable versions of NP-complete problems. *SIAM J. Computing*, 25:1293–

1304, 1996.

[39] Ronny Lorenz, Stephan H Bernhart, Christian Höner zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F. Stadler, and Ivo L. Hofacker. ViennaRNA Package 2.0. *Alg. Mol. Biol.*, 6:26, 2011.

[40] Peter F. Stadler. Fitness landscapes arising from the sequence-structure maps of biopolymers. *J. Mol. Struct. (THEOCHEM)*, 463:7–19, 1999.

[41] Martijn A. Huynen, Peter F. Stadler, and Walter Fontana. Smoothness within ruggedness. The role of neutrality in adaptation. *Proc. Natl. Acad. Sci. USA*, 93(1):397–401, 1996.

[42] Martijn A. Huynen. Exploring phenotype space through neutral eolution. *J. Mol. Evol.*, 43(3):165–169, 1996.

[43] David H Mathews, Matthew D Disney, Jessica L Childs, Susan J Schroeder, Michael Zuker, and Douglas H Turner. Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proc. Natl. Acad. Sci. USA*, 101(19):7287–7292, 2004.

[44] Michael Zuker and Patrick Stiegler. Optimal computer folding of larger RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, 9(1):133–148, 1981.

[45] Ronny Lorenz, Ivo L. Hofacker, and Peter F. Stadler. RNA folding with hard and soft constraints. *Alg. Mol. Biol.*, 11:8, 2016.

[46] Hugo M Martinez. An rna folding rule. *Nucleic Acids Res*, 12:323–334, 1984.

[47] B. A. Shapiro. An algorithm for comparing multiple RNA secondary structures. *Computer Applications in the Biosciences*, 4(3):387–393, 1988.

[48] Ivo L Hofacker, Walter Fontana, Peter F Stadler, L Sebastian Bonhoeffer, Manfred Tacker, and Peter Schuster. Fast folding and comparison of RNA secondary structures. *Monatsh. Chem.*, 125:167–188, 1994.

[49] Kamaludin Dingle, Fatme Ghaddar, Petr Šulc, and Ard A Louis. Phenotype bias determines how natural RNA structures occupy the morphospace of all possible shapes. *Mol. Biol. Evol.*, 39(1):msab280, 2021.

[50] Jacobo Aguirre, Javier M. Buldú, Michael Stich, and Susanna C. Manrubia. Topological structure of the space of phenotypes: The case of RNA neutral networks. *PLoS ONE*, 6(10):e26324, 2011.