Published online 18 November 2022

Nucleic Acids Research, 2023, Vol. 51, No. 2 ***e7***
https://doi.org/10.1093/nar/gkac1029

LazySampling and LinearSampling: fast stochastic sampling of RNA secondary structure with applications to SARS-CoV-2

He Zhang [2,1], Sizhen Li[1], Liang Zhang[1], David H. Mathews [3,4,5] and Liang Huang[1,*]

[1]School of Electrical Engineering & Computer Science, Oregon State University, Corvallis, OR, USA, [2]Baidu Research, Sunnyvale, CA, USA, [3]Department of Biochemistry & Biophysics, University of Rochester Medical Center, Rochester, NY 14642, USA, [4]Center for RNA Biology, University of Rochester Medical Center, Rochester, NY 14642, USA and [5]Department of Biostatistics & Computational Biology, University of Rochester Medical Center, Rochester, NY 14642, USA

Received May 27, 2022; Revised September 22, 2022; Editorial Decision October 17, 2022; Accepted October 21, 2022

ABSTRACT

Many RNAs fold into multiple structures at equilibrium, and there is a need to sample these structures according to their probabilities in the ensemble. The conventional sampling algorithm suffers from two limitations: (i) the sampling phase is slow due to many repeated calculations; and (ii) the end-to-end runtime scales cubically with the sequence length. These issues make it difficult to be applied to long RNAs, such as the full genomes of severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). To address these problems, we devise a new sampling algorithm, LazySampling, which eliminates redundant work via on-demand caching. Based on LazySampling, we further derive LinearSampling, an end-to-end linear time sampling algorithm. Benchmarking on nine diverse RNA families, the sampled structures from LinearSampling correlate better with the well-established secondary structures than Vienna RNAsubopt and RNAplfold. More importantly, LinearSampling is orders of magnitude faster than standard tools, being 428× faster (72 s versus 8.6 h) than RNAsubopt on the full genome of SARS-CoV-2 (29 903 nt). The resulting sample landscape correlates well with the experimentally guided secondary structure models, and is closer to the alternative conformations revealed by experimentally driven analysis. Finally, LinearSampling finds 23 regions of 15 nt with high accessibilities in the SARS-CoV-2 genome, which are potential targets for COVID-19 diagnostics and therapeutics.

INTRODUCTION

RNAs are involved in many cellular processes, including expressing genes, guiding RNA modification (1), catalyzing reactions (2) and regulating diseases (3). Knowing the spatial structure of RNAs is one of the keys to better understand these biological processes and further harness RNAs for diagnostics and therapeutics, but determining the 3D structures is hard and expensive (4). Alternatively, RNA secondary structure is helpful for revealing the structure–function relationship, and can be used for inferring the 3D structure (5,6). Therefore, being able to rapidly and accurately predict RNA secondary structures is desired.

Commonly, the minimum free energy (MFE) structure is predicted (8,9), but these methods do not capture the fact that many RNAs fold into multiple structures at equilibrium (10,11). To address this issue, Ding and Lawrence (12) pioneered *stochastic sampling*, which samples secondary structures according to their probabilities in the ensemble (see Figure 1). These samples are not only informative for describing or probing the ensemble, but are also useful for downstream applications. First, we can model the multiple conformations based on the sampled structures, and further cluster them into representative structures (7,13). For example, several studies used sampled structures, together with SHAPE mapping data, to model and cluster the conformations of the frameshifting region in severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) (14,15). Also, we can use a set of samples to estimate the end-to-end distance of an RNA (16). In addition, we often want to predict the probability that a region is completely unpaired, known as the *accessibility* of that region, which plays an important role in small interfering RNA (siRNA) drug design (17–19). Accessibility can be easily estimated from the fraction of sampled structures for which the region is completely unpaired (see Figure 1).

*To whom correspondence should be addressed. liang.huang.sh@gmail.com

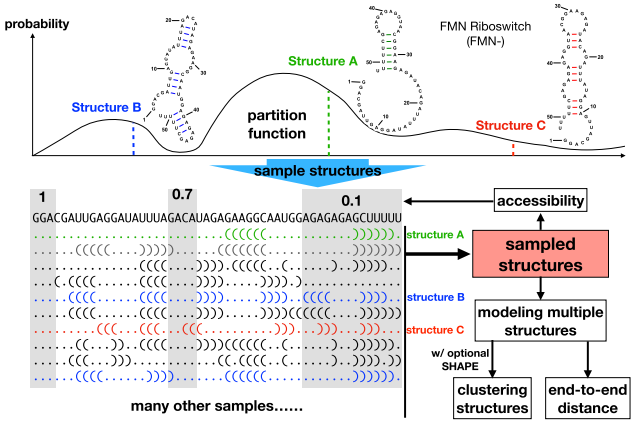

Figure 1. Overview and applications of stochastic sampling of RNA secondary structure. The FMN riboswitch is used as an example, and their centroid structures in the absence of the ligand are shown (7).

However, widely used as it is, the standard sampling algorithm suffers from two main limitations. First, the sampling phase is slow due to many repeated and redundant calculations, wasting a substantial amount of time especially for large sample sizes. Note that our notion of 'redundant' is unrelated to the one in 'non-redundant sampling' (20), which is a variant to output a collection of unique structures, while standard sampling can sample the same structure more than once. Second, its end-to-end runtime scales cubically with sequence length as it has to calculate the partition function using the standard McCaskill algorithm (21) *before* sampling. Therefore, both the sampling and partition function phases scale *superlinearly* with the sequence length. This makes it difficult to be applied to long sequences, such as the full-genome of SARS-CoV-2.

To alleviate these issues, we present three innovations, each built upon the previous one, that together linearize both the sampling and partition function phases to achieve end-to-end linear runtime. The first idea, *Full-Saving Sampling*, eliminates all redundant calculations in the sampling phase by saving all computations during the partition-function phase. The second idea, *LazySampling*, is an even smarter version that only saves computations that are needed during the sampling phase on the fly, greatly reducing the amount of runtime and memory used for saving. These two ideas are based on the observation that node visits are highly unbalanced: most nodes are never visited while a small fraction of 'important' nodes are visited repeatedly. Finally, we further improve LazySampling by replacing its $O(n^3)$-time partition function calculation by our recently proposed $O(n)$-time approximate algorithm, LinearPartition (22). This combination gives rise to *LinearSampling*, an end-to-end linear time sampling algorithm that is orders of magnitude faster than the standard one (see Figure 6 for speedup details).

More importantly, in order to fight current and future pandemics, it is of great value to study the SARS-CoV-2 genome structure and find the regions with high accessibilities, which can be potentially used for coronavirus disease (COVID) diagnostics and therapeutics. However, the conventional tools, including the traditional sampling tools, and other methods to estimate accessibility, e.g. constrained partition function (forcing each region of interest to be fully unpaired, and computing the fraction of the resulting constrained partition function over the global one) and direct computation (23,24), all run in at least $O(n^3)$ time and are too slow on such long sequences with the consideration of global, long-range base pairs. In contrast, LinearSampling can easily scale up to the whole SARS-CoV-2 genomes without local window constraints, and sample 10 000 structures in only 72 s, achieving 428× speedup (72 s versus 8.6 h) compared with RNAsubopt, a widely used sampling tool in the ViennaRNA package (25). We confirm that LinearSampling-derived accessibilities correlate well with the experimentally guided structures (14), resulting in 23 regions of 15 nt with high accessibilities, which are potential targets of diagnostics and drug design.

In addition, the lazy saving strategy can also be applied to other stochastic sampling tasks, such as the Gibbs centroid sampling (26) and structural alignment sampling (27). Also, LinearSampling can load partition function (hypergraph nodes) dumped from LinearTurboFold (28), a tool for simultaneous folding and alignment of RNA homologs, which integrates homologous information into the sampled structures to identify conserved and accessible regions.

MATERIALS AND METHODS

To facilitate our discussion on sampling, we first formulate the search space of RNA folding using the framework of (directed) hypergraphs (29,30) which have been used for both the closely related problem of context-free parsing (31) and RNA folding itself (30,32). This formulation makes it possible to present the standard sampling algorithm as top-down stochastic backtracing that is mirror-symmetric to the bottom-up partition function computation. Then we present our two novel sampling algorithms which eliminate redundant computations by saving and on-demand caching, respectively. Finally, we present our LinearSampling algorithm which is the first sampling algorithm to run in end-to-end linear time.

Hypergraph framework

For RNA sequence $\mathbf{x} = x_1 \dots x_n$, we formalize its folding space as a **hypergraph** $H = \langle V, E \rangle$. Each **node** v is a subsequence (i.e. span) $\mathbf{x}_{i,j} = x_i \dots x_j$, and each **hyperedge** e is a pair $\langle node(e), subs(e) \rangle$ representing a decomposition of $node(e)$ into a list of children nodes $subs(e)$, e.g. $\langle \mathbf{x}_{i,j}, [\mathbf{x}_{i,k}, \mathbf{x}_{k+1,j}] \rangle$ divides one span into two smaller ones. For each node v, we define $\text{INEDGES}(v) \triangleq \{e \mid node(e) = v\}$ to be its set of **incoming hyperedges**, i.e. all decompositions of v.

Hyperedges with one and two child(ren) nodes are called *unary* and *binary* hyperedges, respectively. In order to recursively assemble substructures to form the global structure, each hyperedge $e = \langle v, subs \rangle$ is associated with a **combination function** $f(e) : \mathbb{S}^{|subs(e)|} \mapsto \mathbb{S}$ that assembles substructures from $subs$ into a structure for v (here $\mathbb{S}$ is the set of dot–bracket strings that represent RNA secondary structure). Each hyperedge e is associated with an (extra) **energy term** $w(e) \in \mathbb{R}$.

We take the classical Nussinov algorithm (8) as a concrete example, which scores secondary structures by counting the number of base pairs. The nodes include singleton spans $\mathbf{x}_{i,i}$ that are terminals, and decomposable spans $\mathbf{x}_{i,j}$ $(1 \le i < j \le n)$ that can be decomposed in three ways depending on base x_j:

- x_j is unpaired, which corresponds to a unary hyperedge $\langle \mathbf{x}_{i,j}, [\mathbf{x}_{i,j-1}] \rangle$ with the combination function $f_1(a) =$ "$a\,\cdot$" and energy term 0:

$$\overbrace{\boxed{\mathbf{x}_{i,j-1}}\ \cdot}^{\mathbf{x}_{i,j}} \qquad i \quad j$$

- x_j pairs with x_i (if allowed), denoted by a unary hyperedge $\langle \mathbf{x}_{i,j}, [\mathbf{x}_{i+1,j-1}] \rangle$ with combination function $f_1'(a) =$ "$(\,a\,)$" and energy term -1:

$$\overbrace{(\ \boxed{\mathbf{x}_{i+1,j-1}}\ )}^{\mathbf{x}_{i,j}} \qquad i \quad j$$

- x_j pairs with some x_k $(i < k < j)$, dividing $\mathbf{x}_{i,j}$ into two smaller spans $\mathbf{x}_{i,k-1}$ and $\mathbf{x}_{k+1,j-1}$:

$$\overbrace{\boxed{\mathbf{x}_{i,k-1}}\ (\ \boxed{\mathbf{x}_{k+1,j-1}}\ )}^{\mathbf{x}_{i,j}} \qquad i \quad k \quad j$$

  These bifurcations correspond to a set of binary hyperedges with combination function $f_2(a, b) =$ "$a\,(\,b\,)$" and energy term -1:

$$\textsc{Binary}(\mathbf{x}_{i,j}) = \bigcup_{\substack{i<k<j \\ (x_k, x_j)\ \text{pair}}} \{\langle \mathbf{x}_{i,j}, [\mathbf{x}_{i,k-1},\ \mathbf{x}_{k+1,j-1}] \rangle\}$$

  See Figure 2 for an illustration.

This framework can easily extend to other folding algorithms such as Zuker (9) and LinearFold (33), where nodes are '*labeled spans*' such as $C_{i,j}$ for substructures over $\mathbf{x}_{i,j}$ with (x_i, x_j) paired, $M_{i,j}$ for multiloops over $\mathbf{x}_{i,j}$, etc.

Classical sampling algorithm under a hypergraph framework

All sampling algorithms consist of the (bottom-up) partition function calculation phase and the (top-down) stochastic backtracing phase which is mirror-symmetric to the former.

The partition function phase. In this bottom-up phase, we first calculate the local partition function $Z(v)$ of each node v (see Supplementary Figure S1), summing up the contributions from each incoming hyperedge e (line 7), i.e. $Z(v) = \sum_{e \in \textsc{InEdges}(v)} Z(e)$. This part takes $O(E) = O(n^3)$ time as each hyperedge is traversed once and $O(V) = O(n^2)$ space as we need to store $Z(v)$ for each node v. Note that the hyperedges are by default not saved, and will be recalculated on demand during the sampling phase. If we want to save all hyperedges instead, we need $O(n^3)$ space; the time complexity remains $O(n^3)$, but in practice the overhead for saving (line 8) is quite costly and it may run out of memory (see Figure 4).

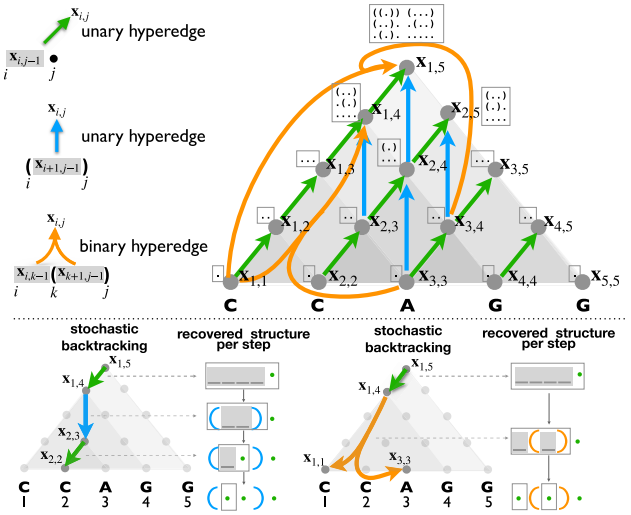

Figure 2. Top: the hypergraph framework and the bottom-up partition function calculation. All nodes and hyperedges are shown. **Bottom**: two sampling instances given the partition function above. The sampling is just top-down stochastic backtracing, i.e. at each visited node, we randomly choose one of its incoming hyperedges, traverse that hyperedge down and recursively visit its children nodes.

Classical 'Non-Saving' Sampling. In the sampling phase, the Non-Saving Sampling algorithm (see Supplementary Figure S2) recursively and stochastically backtraces from the goal node, in the exact opposite direction to the bottom-up partition function phase (see Figures 2 and 3). When visiting a node v, it tries to sample a hyperedge e from v's incoming hyperedges $\textsc{InEdges}(v)$ according to the probability $Z(e)/Z(v)$. This is done by first generating a random number p between 0 and $Z(v)$, and then gradually recovering each incoming hyperedge e, accumulating its $Z(e)$ to a running sum s, until s exceeds p, at which point that current hyperedge e is chosen as sampled. Note that this algorithm in general does *not* need to recover *all* incoming hyperedges of v (see 'partial recovery of hyperedges' in Figure 3), though in the worst case it would. It then backtraces recursively to the corresponding subnode(s) of hyperedge e. See Supplementary Sec. 1.1 for the detailed complexity analysis of Full-Saving Sampling.

Our presentation resembles the original Ding and Lawrence (12) algorithm, but is simpler and cleaner thanks to the hypergraph framework and the mirror symmetry between the bottom-up and top-down phases. In contrast, Ding and Lawrence's recurrences for the two phases are different in nature (see figure 1 of their paper), which results in unnecessarily complicated implementations [see Vienna RNAsubopt for an example; RNAstructure's (34) sampling is much closer to our non-saving version, except for being non-recursive]. Ponty (35) also exploits this symmetry both in theory and in simulations, but his analysis is for the special case under the simplified Nussinov model. Our work is on the full Turner model (36), being the first to formulate

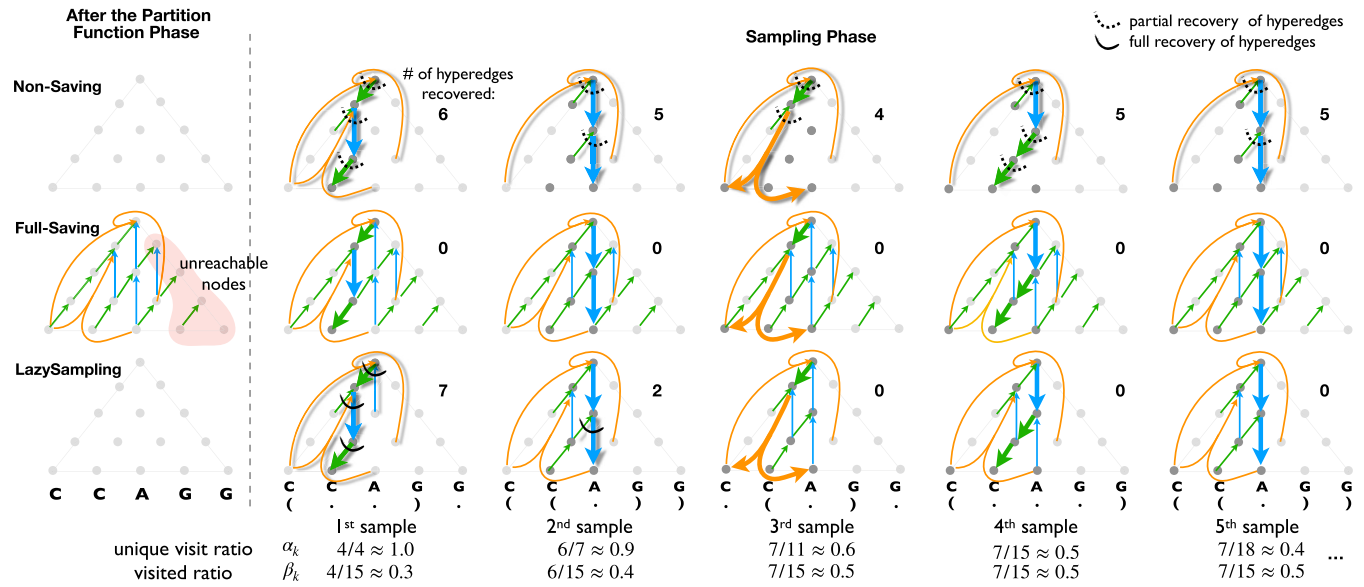

Figure 3. Illustrations of the three sampling algorithms: classical Non-Saving Sampling, our Full-Saving Sampling and our LazySampling. For each sample, the newly recovered hyperedges are shown with shadow, and the bold hyperedges are the chosen derivation in stochastic backtracing (pointing downwards). Non-Saving Sampling needs to repeatedly recover hyperedges in each sample (most of the node visits are non-unique; see α_k), whereas Full-Saving Sampling does not need to recover any hyperedge thanks to storing all of them during the bottom-up phase, but rather oversaves (only less than half of the nodes are visited during the first five samples; see β_k; in fact, six nodes $\{\mathbf{x}_{2,5}, \mathbf{x}_{3,4}, \mathbf{x}_{3,5}, \mathbf{x}_{4,5}, \mathbf{x}_{5,5}, \mathbf{x}_{4,4}\}$ are *never* visited even when $k \to \infty$ as they are not reachable from the goal node $\mathbf{x}_{1,5}$). LazySampling is a smart hybrid between the two, using on-demand caching, thus saving only a small fraction of hyperedges.

general sampling (Nussinov, Zuker, LinearFold, etc.) under a unified framework.

Idea 1: Full-Saving Sampling

One observation is that Non-Saving Sampling wastes time on recovering hyperedges during the sampling phase. First, due to the symmetry, all hyperedges recovered in the sampling phase have already been traversed during the inside phase. To make things worse, a great number of hyperedges are recovered multiple times across different samples because, whenever a node is (re-)visited, its hyperedges need to be re-recovered. This situation worsens with the sample size k. More formally, we define the 'unique visit ratio' among k samples,

$$\alpha_k \triangleq \frac{\#\text{ of unique nodes visited in } k \text{ samples}}{\#\text{of all node visits in } k \text{ samples}} \tag{1}$$

which we will see in Figure 5A to be extremely small, quickly approaching 0% as k increases, meaning most node visits are repeated ones. Actually, all hyperedges could have been saved during the inside phase, thus alleviating the need to recover hyperedges during sampling. Therefore, we devise a new algorithm, Full-Saving Sampling (see Supplementary Figures S2 and S3 for pseudocode). For each node v, for each of its incoming hyperedges e, we save $Z(e)$, which is e's contribution to the local partition function $Z(v)$, once and for all. Then the sampling phase is easier, only requiring sampling a hyperedge e according to its relative contribution (or 'weight') to v, i.e. $Z(e)/Z(v)$ (line 2 in Supplementary Figure S3). See Supplementary Sec. 1.2 for the detailed complexity analysis of generating each sample.

Idea 2: LazySampling = Lazy-Saving Sampling

Though Full-Saving Sampling avoids all re-calculations, it costs too much more space [$O(n^3)$ versus $O(n^2)$] and significantly more time in practice for saving the whole hypergraph. Actually, the vast majority of nodes are *never* visited during the sampling phase even for a large sample size. To quantify this, we define the 'visited ratio' to be:

$$\beta_k \triangleq \frac{\#\text{ of unique nodes visited in } k \text{ samples}}{\#\text{ of all nodes in hypergraph}} \tag{2}$$

Our experiments in Figure 5B show that only <0.5% of all nodes in the hypergraph are ever visited for 20 000 samples of a 3048 nt sequence (URS0000D5C703_9606) using Vienna RNAsubopt, so most of the saving is indeed wasted. In other words, node visits are greatly unbalanced, i.e. a small portion of nodes are repeatedly visited, while most nodes are barely visited. Based on this observation, we devise an even smarter algorithm, LazySampling, which is a hybrid between Non-Saving and Full-Saving Samplings (see Supplementary Figure S4). By 'lazy' we mean only recovering and saving a node v's hyperedges when needed, i.e. the first time v is visited during the sampling phase. In this way, each hyperedge is recovered at most once, and most are not recovered at all. This version balances between space and time, and is the fastest among the three versions in most settings in practice.

The complexity analysis of LazySampling is also a hybrid between the Non- and Full-Saving versions, combined together using the α_k and β_k ratios. We note that the sampling time of LazySampling consists of two parts: (i) the hyperedge recovery (and saving) work and (ii) the sampling

Table 1. Time complexities of four sampling algorithms (n is the sequence length, k is sample size, α_k and β_k are the 'unique visit' (Eq. 1) and 'visited' (Eq. 2) node ratios, respectively, and b is beam size)

Sampling algorithm	Partition function		Sample time (worst-case)	Sample time (best-case)	Sample space (worst-case)
	Time	Space			
Classical Non-Saving Sampling	$O(n^3)$	$O(n^2)$	$O(kn^2)$	$O(kn\log n)$	$O(n)$
Idea 1: Full-Saving Sampling	$O(n^3)$	$O(n^3)$	$O(kn\log n)$	$O(kn)$	$O(n)$
Idea 2: LazySampling	$O(n^3)$	$O(n^2)$	$O(\alpha_k kn^2 + kn\log n)$	$O(\alpha_k kn\log n + kn)$	$O(\beta_k n^3)$
Idea 3: LinearSampling	$O(nb^2)$	$O(nb)$	$O(\alpha_k knb + kn\log b)$	$O(\alpha_k kn\log b + kn)$	$O(\beta_k nb^2)$

The runtime of LazySampling is a hybrid between those of the Non- and Full-Saving ones, i.e. $T^{\mathrm{S}}_{\mathrm{lazy}} = \alpha_k T^{\mathrm{S}}_{\mathrm{non}}(\textit{hyperedge recovery}) + T^{\mathrm{S}}_{\mathrm{full}}(\textit{backtracing})$ (Eq. 3). LinearSampling extends LazySampling by replacing the exact partition function with LinearPartition, and achieves end-to-end linear runtime. Note that the sampled derivations are mostly balanced as the depth of derivation scales O(log n) in practice (see Supplementary Figure S6), therefore the complexity of the average case is close to that of the best case.

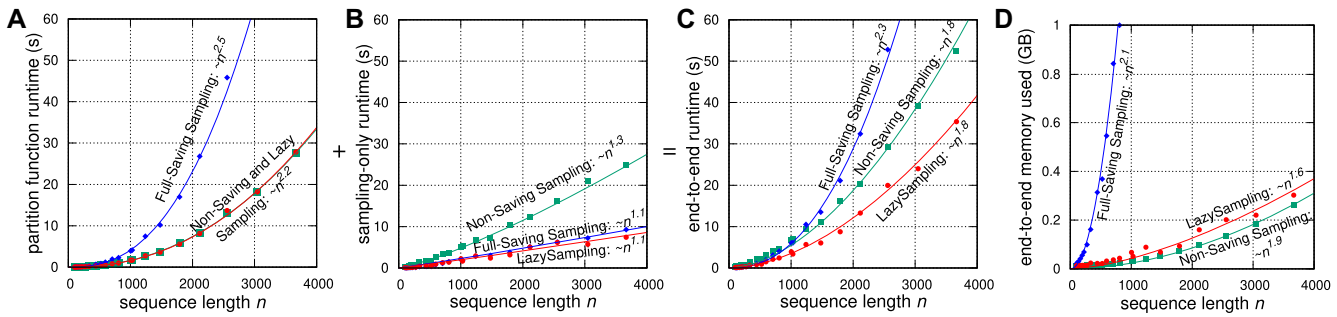

Figure 4. Runtime and memory comparisons against sequence length on the RNAcentral dataset (cut at 4000 nt) under exact partition function. The sample size is 50 000. (**A–C**) Partition function, sampling-only and end-to-end runtime. (**D**) Memory usage.

(backtracing) work after relevant hyperedges are recovered. Part (i) resembles Non-Saving Sampling, but with a ratio of α_k, because only a tiny fraction of node visits are unique, and hyperedge recovery is only performed at the first visit to each node. Part (ii) is identical to Full-Saving Sampling (in both cases, all needed hyperedges are already saved). Therefore, we have the following relationships among the time complexities for the sampling phase of these three versions:

$$T^{\mathrm{S}}_{\mathrm{lazy}}(n,k) = \underbrace{\alpha_k T^{\mathrm{S}}_{\mathrm{non}}(n,k)}_{\textit{hyperedge recovery}} + \underbrace{T^{\mathrm{S}}_{\mathrm{full}}(n,k)}_{\textit{backtracing}} \quad (3)$$

This holds for both the worst- and best-case scenarios in Table 1. The space complexity is easier: LazySampling saves only a fraction (β_k) of all nodes in the hypergraph, thus $O(\beta_k n^3)$. See Table 1 for summary.

Idea 3: LinearSampling = LazySampling + LinearPartition

LazySampling is the most efficient among all three sampling algorithms above, but the end-to-end runtime still scales $O(n^3)$ due to the partition function computation. To address this problem, we use our recently proposed linear-time approximate algorithm, LinearPartition (22), to replace the conventional cubic-time algorithm. It can be used with any of the three sampling algorithms and, in particular, we name the combination, '*LinearPartition + LazySampling*', the **LinearSampling** algorithm as it is the fastest among all combinations and achieves end-to-end linearity.

Supplementary Figure S5 describes a simplified pseudocode using the Nussinov–Jacobson energy model. Inspired by LinearPartition, we employ a beam search to prune out nodes with the small partition function (line 11) during the inside phase. So at each position j, only the top b promising nodes 'survive' [i.e. $O(nb)$ nodes survive in total]. Here the beam size b is a user-specified hyperparameter, and the default $b = 100$ is found to be more accurate for structure prediction than an exact search (22). See Supplementary Sec. 1.3 for the detailed complexity analysis of LinearSampling.

RESULTS

Efficiency and scalability

We benchmark the runtime and memory usage on 36 sequences sampled from RNAcentral (37), ranging from 100 to 27 985 nt, which we refer to as the RNAcentral dataset; see Supplementary Table S2 for statistics. We use a Linux machine (CentOS 7.7.1908) with 2.30 GHz Intel Xeon E5-2695 v3 CPU and 755 GB memory, and gcc 4.8.5.

Comparing Non-Saving, Full-Saving and LazySampling. Figure 4 compares the three sampling strategies under the exact partition function calculation. Non-Saving Sampling and LazySampling have identical partition function runtime, which are both faster than Full-Saving Sampling, benefiting from saving no hyperedges. However, for the sampling phase, Non-Saving has to repeatedly recover hyperedges, resulting in a runtime increase in sampling-only time. LazySampling avoids this cost and reduces the runtime to less than half of Non-Saving's, leading

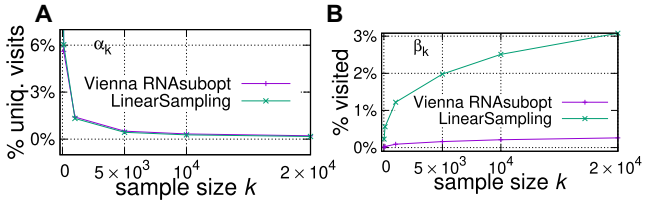

Figure 5. In practice, most node visits are repeated, and only a small portion of all nodes are visited. (**A**) Unique visit ratio α_k. (**B**) visited ratio β_k. Note that β_k is higher in LinearSampling than in RNAsubopt, mainly because the denominator scales quadratically with sequence length n in RNAsubopt, while it scales linearly in LinearSampling; see Table 1 for the analysis of partition function space complexity (which is related to the number of all nodes in the hypergraph) of different systems. Here $n = 3048$ nt. Supplementary Figure S10 demonstrates the trend with sequence length, and Supplementary Figure S11 shows that most of the visits are concentrated on a few nodes.

to a similar performance as Full-Saving at sampling-only time. Regarding end-to-end runtime, which combines the partition function and the sampling phases, LazySampling is the fastest and Full-Saving is the slowest. For memory usage, Full-Saving uses much more memory, while the other two are close. Supplementary Figure S9 shows the comparisons against sample size k, which confirms that LazySampling is the best, especially when k is large.

We also illustrate why LazySampling is a better strategy. Figure 5A shows the unique visit ratio, α_k, is $<5\%$ when $k > 1000$ for both RNAsubopt and LinearSampling, confirming that LazySampling is able to avoid a large number of recalculations during the sampling phase. On the other hand, the visited ratio β_k (Figure 5B) is always smaller than 0.5% and 3% for RNAsubopt and LinearSampling, respectively, and grows slower and slower as the sample size increases, suggesting that saving all hyperedges (i.e. Full-Saving) is not ideal. Supplementary Figure S10 further confirms that both α_k and β_k do not increase with sequence length; therefore, this analysis applies to both short and long sequences.

Comparing LinearSampling with Vienna RNAsubopt global and local modes. We compare the efficiency and scalability between LinearSampling and Vienna RNAsubopt (version 2.4.14) in Figure 6. RNAsubopt (`-p` mode) implemented the conventional sampling algorithm based on the exact partition function, and it enables local sampling (`--maxBPspan` mode) by limiting the sequence distance between paired nucleotides. Figure 6A confirms that LinearSampling scales end-to-end linearly against sequence length n, and is much faster than RNAsubopt, which has an empirical runtime of $O(n^{2.8})$. For 50 000 samples, LinearSampling is 208× faster (2 m versus 7 h) than RNAsubopt on the longest sequence of the RNAcentral dataset (URS0001BDA28C_9606, 27 985 nt), and 251× faster on the SARS-CoV-2 sequence (29 903 nt). Regarding sampling-only runtime, LinearSampling is >3× faster. Figure 6C confirms that the memory usage of LinearSampling is also linear, but RNAsubopt requires $O(n^2)$ memory. LinearSampling uses 1 GB for SARS-CoV-2, while RNAsubopt needs 20 GB.

It is surprising that RNAsubopt local mode has an empirical complexity of $O(n^{3.3})$ for end-to-end runtime, and is even slower than its global mode. For a 9211 nt sequence (URS00009BB84A_10090), RNAsubopt local mode, using a span of 70, takes 83.8 min, and its global mode takes 18.7 min, while LinearSampling only takes 34.9 s. Memory-wise, RNAsubopt local mode uses as much as its global mode.

We also observe that RNAsubopt turns to overflow on long sequences, making it less reliable. For example, RNAsubopt overflows on four sequences in the RNAcentral dataset, shown in Figure 6 with open triangles whose end-to-end and sampling-only runtime drop unreasonably. For one of them (URS00007C400D_9606, 19 071 nt), an overflow happens in the segment [5775, 12619], leading to an unpaired region longer than 6000 nt in all sampled structures. Another one (URS00009C28A8_9606, 22 158 nt) triggers an error during the sampling phase:

```
ERROR: backtrack failed in qm1
```

resulting in an abnormal exit, with only a few structures generated. Though a self-adapted partition function scale (named `pf_scale`) is used in RNAsubopt, overflow is still unavoidable for some long sequences. In contrast, following LinearPartition, LinearSampling uses log-space for the partition function, which does not have overflow issues.

LinearSampling benefits from both LinearPartition and LazySampling. Figure 7A investigates the runtime of Non-Saving Sampling and LazySampling under the linear partition function, and compares them with LinearPartition with sequence length up to ∼30 000 nt. We observe that the partition function phase is fast and no longer the bottleneck, while the classical Non-Saving Sampling is 5–6× slower than LinearPartition, and takes the majority of end-to-end runtime. LazySampling enjoys ∼3.5× speedup from the Non-Saving Sampling and substantially reduces the sampling runtime to roughly the same as LinearPartition's runtime.

We further compare LazySampling with Non-Saving Sampling and RNAsubopt under the exact partition function. RNAsubopt uses the non-saving strategy, but adopts an optimization called the 'Boustrophedon' method (35), which makes it slightly faster than our Non-Saving Sampling. However, it is clear that LazySampling's speedup is more significant.

Correlation with well-established structures across diverse families

We use the ArchiveII dataset (34,38) to investigate the correlation between sampled structures and the ground truth structures. This dataset contains diverse families of RNA sequences and their well-established secondary structures, and each family has a reliable source. We follow the preprocessing steps of a previous study (22), and obtain a subset of 2859 sequences distributed in nine families (see Supplementary Table S1).

We investigate the sampled structure's correlation with the ground truth using 'ensemble defect' (39), the expected number of incorrectly predicted nucleotides over the ensem-

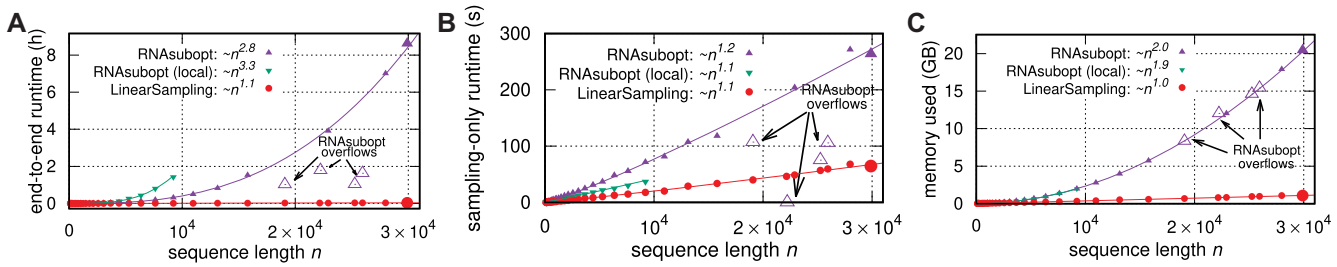

Figure 6. Runtime and memory usage between RNAsubopt, its local mode (span = 70) and LinearSampling on the RNAcentral dataset and SARS-CoV-2 (sample size 50 000). (**A**) End-to-end runtime. (**B**) Sampling-only runtime. (**C**) Memory usage. The large filled triangles and circles around $n = 30\,000$ represent SARS-CoV-2 sequences, and the open triangles are the sequences that RNAsubopt overflows. Note that the sampling-only time of LinearSampling is already the majority of its end-to-end time on SARS-CoV-2 (64.6 out of 123.4 s). See Supplementary Figure S7 for the comparison within 10 000 nt.

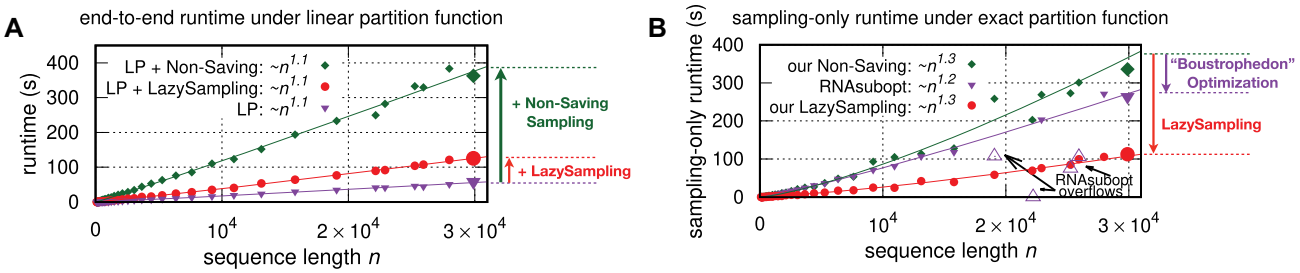

Figure 7. LazySampling can reduce sampling phase runtime with both linear (**A**) and exact (**B**) partition function. (**A**) Linear partition function time (using LinearPartition), Non-Saving Sampling and LazySampling against sequence length. Here LP = LinearPartition. (**B**) The sampling time of RNAsubopt, Non-Saving Sampling and LazySampling under the exact partition function. The runtime of RNAsubopt from Non-Saving Sampling is reduced by 'Boustrophedon' optimization. See Supplementary Figure S8 for the comparison within 10 000 nt, and Supplementary Figure S15 for more runtime analysis regarding the 'Boustrophedon' method.

ble. It is defined as:

$$\Phi(S, \mathbf{y}^*) = \frac{1}{|S|} \sum_{\mathbf{y} \in S} d(\mathbf{y}, \mathbf{y}^*)$$

$$= |\mathbf{y}^*| - 2 \sum_{(i,j) \in \text{pairs}(\mathbf{y}^*)} p_{i,j}(S) - \sum_{j \in \text{unpaired}(\mathbf{y}^*)} q_j(S)$$

where $\mathbf{y}^*$ is the ground truth structure, and $d(\mathbf{y}, \mathbf{y}^*)$ is the distance between $\mathbf{y}$ and $\mathbf{y}^*$, defined as the number of incorrectly predicted nucleotides in $\mathbf{y}$. $p_{i,j}(S)$ is the probability of nucleotide i pairing with nucleotide j in sample S, which can be easily calculated as the number of (i, j) pairs divided by sample size. Similarly, $q_j(S)$ is the probability of j being unpaired in the sample S, i.e. $q_j(S) = 1 - \sum p_{i,j}(S)$. For this database of non-coding RNAs with well-defined structures, we expect that a lower ensemble defect indicates better structure prediction accuracy.

Figure 8A shows the ensemble defect differences between LinearSampling and RNAsubopt on each family and overall. Note that the lower ensemble defect suggests better correlation with the ground truth structures. For short families, the differences between LinearSampling and RNAsubopt are either 0 or close to 0, indicating that the sampling qualities of the two systems are similar on these families. However, on the longer families (i.e. 16S and 23S rRNAs), LinearSampling has lower ensemble defects, showing that it performs better on longer sequences. The only family for which LinearSampling performs worse is tmRNA. We also present the results of RNAsubopt local mode, with base pair length limitations of 70 and 150. RNAsubopt local mode does not have a default span size; we choose 70 following the default setting in RNAplfold (40), and 150 since it is the largest default limit in the local folding literature and software. It is obvious that the local sampling has a much higher (worse) ensemble defect on 23S rRNA, which is probably caused by not predicting the known base pairs beyond the maximum span limit.

An important application of the sampling algorithm is to calculate a region's accessibility. Therefore, we follow Ding and Lawrence (12), calculating accessibilities of window size 4 from structures generated by LinearSampling and RNAsubopt, as well as directly from RNAplfold, which implemented a dynamic programming algorithm for calculating accessibility in cubic time (24), and compare them based on the ground truth structures. We denote the measurement of *accessibility defect* as D, which evaluates the averaged incorrect predictions of the accessibility to the ground truth given a window size. For sampling-based methods, $D(S, \mathbf{y}^*)$ is generated from the samples S and is defined as:

$$D(S, \mathbf{y}^*) = \frac{1}{|\mathbf{y}^*| - 3} \sum_{i=1}^{|\mathbf{y}^*|-3} \left| \text{acc}(\{\mathbf{y}^*\}, i) - \text{acc}(S, i) \right|$$

where $\text{acc}(S, i)$ is the accessibility of region $[i, i + 3]$:

$$\text{acc}(S, i) = \frac{1}{|S|} \sum_{\mathbf{y} \in S} \mathbb{1}[\mathbf{y}_{i,i+3} = \text{“}\cdot\cdot\cdot\cdot\text{”}]$$

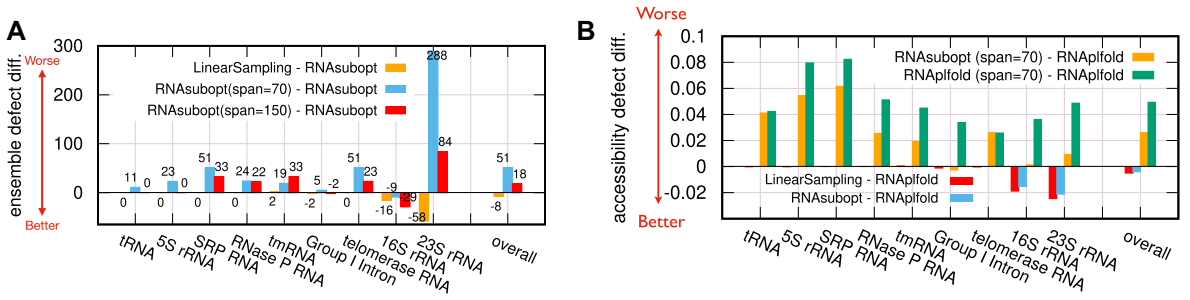

Figure 8. LinearSampling is better correlated with the ground truth structure. (**A**) The ensemble defect difference of each family and overall (averaged by families) of the ArchiveII dataset, comparing LinearSampling, RNAsubopt and its local modes (span = 70 and 150); RNAsubopt is set to be the baseline. (**B**) The accessibility defect difference among LinearSampling, RNAsubopt, RNAplfold and their local modes, using RNAplfold as the baseline. The sample size is 50 000. The families are ordered by their average sequence lengths, from the shortest to the longest; see Supplementary Table S1 for length and identity information of each family.

Figure 8B compares the accessibility defect of LinearSampling with RNAsubopt, RNAplfold and their local modes on the ArchiveII dataset. We observe that LinearSampling outperforms (or is as good as) all others on seven out of nine families, and is the best overall. Notably, both RNAsubopt's and RNAplfold's local modes are worse than their global modes, with only one exception on Group I Intron, indicating that the local modes are less accurate.

It is worth noting that the lower ensemble and accessibility defects of LinearSampling are inherited from LinearPartition, which was shown to be better correlated with the ground truth structures (22) by pruning out structures with low probabilities. To confirm this, we investigate the root-mean-square deviation (RMSD) between the base pairing probability matrices $p(S)$, which is derived from the sample set S, and p', which is generated by Vienna RNAfold or LinearPartition. The results are shown in Supplementary Figure S12, suggesting that structures generated by LinearSampling strictly match with the ensemble distribution of LinearPartition. Since LinearPartition deviates from the exact partition function of RNAfold with a small margin, the folding landscape structures of LinearSampling have different sampling probabilities compared with RNAsubopt. In most cases, structures with lower folding free energies may have higher probabilities in LinearSampling than in RNAsubopt, while low-probability structures in RNAsubopt may have 0 or close to 0 probabilities in LinearSampling.

Applications to SARS-CoV-2

The COVID-19 pandemic swept the world in 2020–2022, and is likely to threaten global health for a long time. Therefore, it is valuable to study the structure of the SARS-CoV-2 genome, which helps us better understand biological processes such as virus replication and translation. Now with the significant speedup, LinearSampling is able to quickly sample structures for the whole genome of SARS-CoV-2, providing an efficient tool for SARS-CoV-2 structural studies. In this section, we run LinearSampling, as well as the conventional tools, on NC_0405512.2, the reference sequence of SARS-CoV-2 (42), investigate the multiple conformations and accessible regions of great interest and show that LinearSampling's structures correlate better with the experimental models. Then we report the LinearSampling's findings on new targets with high accessibility, which can be potentially used for COVID-19 diagnostics and drug design.

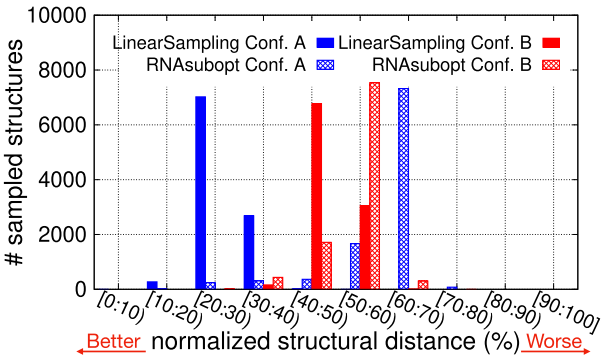

Figure 9. The correlation between sampled structures and DRACO-inferred structures of the SARS-CoV-2 3′-UTR. LinearSampling and RNAsubopt are represented in solid bars and striped bars, respectively; the two DRACO-inferred conformations are in blue (conformation A) and red (conformation B), respectively. The normalized structural distance of sampled structures against DRACO-inferred conformations are grouped into 10 bins, where the distance of 0 means a perfect match of the whole structure, and the distance of 1 means that all positions in the region are incorrectly predicted. The numbers of sampled structure in each bin are illustrated. The majority of samples from LinearSampling are distributed in bins with smaller structural distance compared with RNAsubopt for both conformations, suggesting that LinearSampling has a better ability to model alternative conformations. Here the sample size is 10 000.

Alternative conformations and base pairs. Experimental studies on SARS-CoV-2 have inferred that multiple conformations exist. For example, DRACO analysis (43) suggests two alternative structures in the 3′-untranslated region (UTR) of SARS-CoV-2 based on the clustering of DMS-MaPseq data. To see if LinearSampling's structures are closer to this experimental analysis, we calculate the normalized structural distance (i.e. the number of incorrectly predicted nucleotides normalized by sequence length) between different algorithms and the two conformations, where the lower (normalized) structural distance indicates better correlation with the DRACO results. Figure 9 shows the number of sampled structures derived from LinearSam-

pling and RNAsubopt against normalized structural distance, grouped into 10 bins from the lowest (best) to the highest (worst) of the distances. It is clear that the distribution of distance between LinearSampling and conformation A of the DRACO analysis (solid blue bars) shifts to the direction of lower structural distance compared with RNAsubopt's (striped blue bars). Specifically, for conformation A, there are 7023 among 10 000 of LinearSampling's structures in the bin of [20, 30), suggesting that 70% of LinearSampling's structures are similar to conformation A. In contrast, the majority of RNAsubopt's sample are in the bins that have structural distance >50%, suggesting that the structures of RNAsubopt are similar to neither conformation A nor conformation B. Therefore, LinearSampling's sample correlates better with the DRACO analysis.

Besides the DRACO analysis, COMRADES analysis provides experimental data for short- and long-range base-pairing interactions (44), which can be used to verify the quality of the folding landscape samples regarding the base pairs in different conformations of SARS-CoV-2 genomes. We observed that LinearSampling's oracle structure, compared with that of RNAsubopt, has more base pairs matched with COMRADES analysis with varying sample sizes (Supplementary Figure S13). Regarding the full sample landscapes, LinearSampling has a higher ratio of matched base pairs to predicted ones, and its total number of unmatched base pairs is substantially smaller than RNAsubopt against different sample sizes (Supplementary Figure S14A and B), indicating that LinearSampling has a higher hit rate on the COMRADES-inferred base pairs. The COMRADES data also provides the chimeric reads of each interaction region, where a larger number of chimeric reads indicates high credibility of interaction, and we confirm that LinearSampling has higher average chimeric reads per predicted base pairs for all tested sample sizes (Supplementary Figure S14C). On the other hand, RNAsubopt has higher coverage of COMRADES base pairs, but costs much longer time (Supplementary Figure S14D). The difference in sample landscape between LinearSampling and RNAsubopt is illustrated in a Venn diagram (Supplementary Figure S14E). LinearSampling tends to sample less diverse base pairs, which leads to the benefit of obtaining a higher ratio of matched base pairs and the cost of missing some base pairs shown in the COMRADES analysis; while RNAsubopt tends to sample more diverse base pairs, which has higher coverage but also results in more base pairs that are not supported by experimental data.

Accessibility. We investigate if the accessibilities predicted by LinearSampling match better with the experimentally guided structures, especially on the regions of interest, e.g. the 5′-UTR which has conserved structures and plays a critical role in viral genome replication (46). A number of studies report structural probing data for SARS-CoV-2 genomes (14,15,41,43,47,48), including the selective 2′-hydroxyl acylation analyzed by primer extension (SHAPE) and dimethyl sulfate (DMS) data. These well-accepted experimental techniques for RNA secondary structure probing have been shown to be able to improve the accuracy of structure prediction (49–51). Therefore, we utilize two SHAPE-directed structures (14,41) to evaluate different systems, i.e. LinearSampling, RNAsubopt and RNAplfold. Figure 10 compares the accessibilities derived from the three systems with the experimentally guided structures based on SHAPE reactivities (14). Following the previous study (12), the accessibilities of window size 4 are visualized in the top sub-figure, and LinearSampling clearly correlates better with the SHAPE-directed structure. For example, RNAsubopt and RNAplfold (span = 150) overestimate the accessibilities around the double-stranded region [174, 181], and RNAplfold (both span = 150 and span = 800) overestimates in the region [276, 284]; in contrast, LinearSampling's predictions are close to 0. Also, LinearSampling correctly captures the accessible region around 50, with a high predicted accessibility of nearly 1. We further extend the window size to cover a wider range (from 1 to 11), and visualize the computational results versus the SHAPE-directed structure model in the second row (LinearSampling), the third row (RNAsubopt) and the bottom two rows (RNAplfold). For instance, the black circle at position 52 and window size 5 (indicated by a green arrow) represents a highly accessible region [52, 56] predicted by LinearSampling, which is surrounded by a box, indicating that the prediction is supported by the wetlab experiment. In RNAsubopt and RNAplfold predictions, the same positions are in purple or orange, indicating that they have lower accessibility and are less correlated with the SHAPE reactivities. In addition, LinearSampling predicts the region around position 279, indicated by blue arrows, as an inaccessible region, which is supported by the SHAPE-directed structures. The main differences between the three systems are highlighted in gray shades. In general, LinearSampling's result correlates better with the experimentally guided models.

To further quantify the difference, we calculate the accessibility defects of three important and well-studied regions in SARS-CoV-2, i.e. the 5′-UTR, the frameshifting element (FSE) and the 3′-UTR, shown in Table 2. LinearSampling has lower (better) accessibility defects on all these three regions, except for the FSE region using the reference structure from Manfredonia *et al*., on which LinearSampling is the second best (behind RNAplfold, span = 150) and has only 0.0025 difference from the best one.

New findings on accessible targets. Next, we aim to computationally obtain accessible regions as potential targets for diagnostics and drug discovery. A previous study (45) locates unstructured regions of SARS-CoV-2 by scanning the reference sequence with windows of 120 nt, sliding by 40 nt and then calculating base-pairing probabilities using CONTRAfold (52) for these fragments. In total, 75 accessible regions with ≥15 nt are claimed, where each base has the average unpaired probability of at least 0.6. However, this method has two flaws: (i) it is not correct to evaluate accessibility based on unpaired probabilities due to their mutual dependency; and (ii) it neglects long-range base pairs and has to approximate the accessibilities based on local structures, which is less accurate (see Figure 8). Instead, we measure the accessibilities based on samples generated by LinearSampling, setting the window size to be 15 following Rangan *et al*. (45). We only show the fragments whose

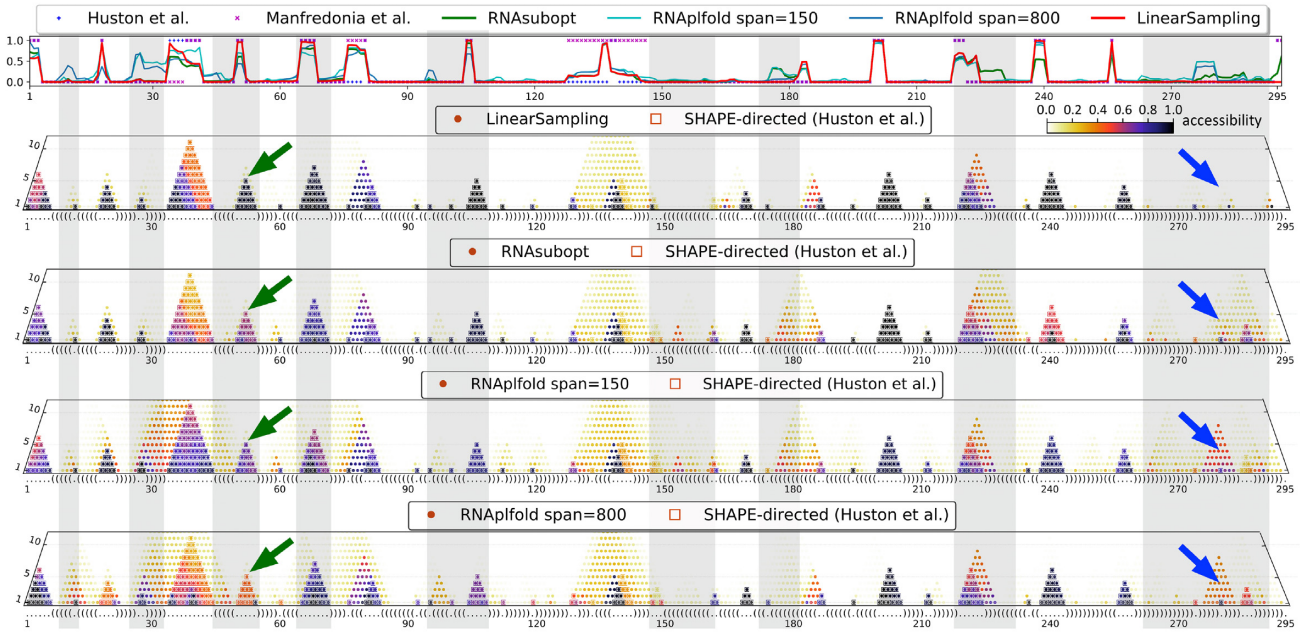

Figure 10. The accessibilities derived from LinearSampling correlate better with the unpaired region in the SHAPE-directed structure of the 5′-UTR of SARS-CoV-2 (14,41). Note that the full sequence was used for the accessibility calculation, but only the 5′-UTR is shown. **First row**: accessibilities of window size 4 derived from SHAPE-directed structures (blue plus symbol and purple cross symbol; agreed on >90% positions), RNAsubopt (green line), RNAplfold local mode (blue lines) and LinearSampling (red line). Following the previous study (12), the accessibility of position i stands for the region $[i, i + 3]$. **Second row**: accessibilities predicted by LinearSampling with window sizes from 1 to 11. Each prediction is presented with a filled circle, where the color correlates with its accessibility value. The SHAPE-directed structure (14) is shown in dot–bracket format along the x-axis, and its accessible regions are annotated in boxes. **Third, fourth and fifth rows**: accessibilities predicted by RNAsubopt and RNAplfold (with a base pairing limit of 150 and 800), respectively. Note that the first row is a special case (window size 4) of the bottom four rows. The main differences are highlighted in gray shading, where LinearSampling is the most accurate in all cases. For example, the green arrows point to window size 5 at position 52 (representing the region [52, 56]), showing the case of a 5 nt accessible region in which LinearSampling predicts the highest accessibility, correlating better with the experimentally guided models. In addition, the blue arrows, pointing to the region around position 279, illustrate an example where an inaccessible region is accurately predicted by LinearSampling but not by other tools. We chose span = 150 and span = 800 for RNAplfold, the former being the largest default limit of local folding in most literature and software, and the latter following a previous study (19).

Table 2. Accessibility defect comparisons between *in silico* predictions (LinearSampling, RNAsubopt and RNAplfold) and experimentally guided models [Huston *et al.* (14) and Manfredonia *et al.* (41)] on UTRs and FSE regions

	5′-UTR	FSE	3′-UTR	Reference
LinearSampling	**0.0824**	**0.4021**	**0.2151**	Huston *et al.* (14)
RNAsubopt	0.1061	0.4461	0.2573	
RNAplfold (span = 150)	0.1304	0.4094	0.2773	
RNAplfold (span = 800)	0.1145	0.4320	0.2528	
LinearSampling	**0.1007**	0.3578	**0.1338**	Manfredonia *et al.* (41)
RNAsubopt	0.1113	0.3647	0.1879	
RNAplfold (span =150)	0.1408	**0.3553**	0.2154	
RNAplfold (span = 800)	0.1201	0.3595	0.1793	

The 5′-UTR, FSE and 3′-UTR are [1, 450], [13470, 13545] and [29550, 29870], respectively (45). The sample size is 10 000.

accessibilities are >0.5, i.e. they are more likely to be open than closed. We list all 23 regions found by LinearSampling in Supplementary Table S3. Some of the regions are overlapped, resulting in a total of nine separate accessible regions, which are illustrated in Figure 11. Of the nine regions, two are in ORF1ab, one in ORF3a, one in the M gene, three in the N gene and two in the S (spike) gene whose proteins can recognize and bind with its receptor (53).

The constant emergence of mutations may lead to changes in new virus variants. Therefore, it is better to target both conserved and accessible regions. LinearSampling is able to integrate conservation information for sampled structures by loading the partition function dumped from LinearTurboFold (28), which simultaneously folds and aligns homologous sequences and can identify conserved regions. This feature of LinearSampling allows mutation-insensitive diagnostics and drug design. The homology-assisted results have been previously reported (28).

DISCUSSION

We focus on simplifying and accelerating the stochastic sampling algorithm for a given RNA sequence. Algorithmically, we present a hypergraph framework under which the classical sampling algorithm can be greatly simplified. We further elaborate this sampling framework to formal-

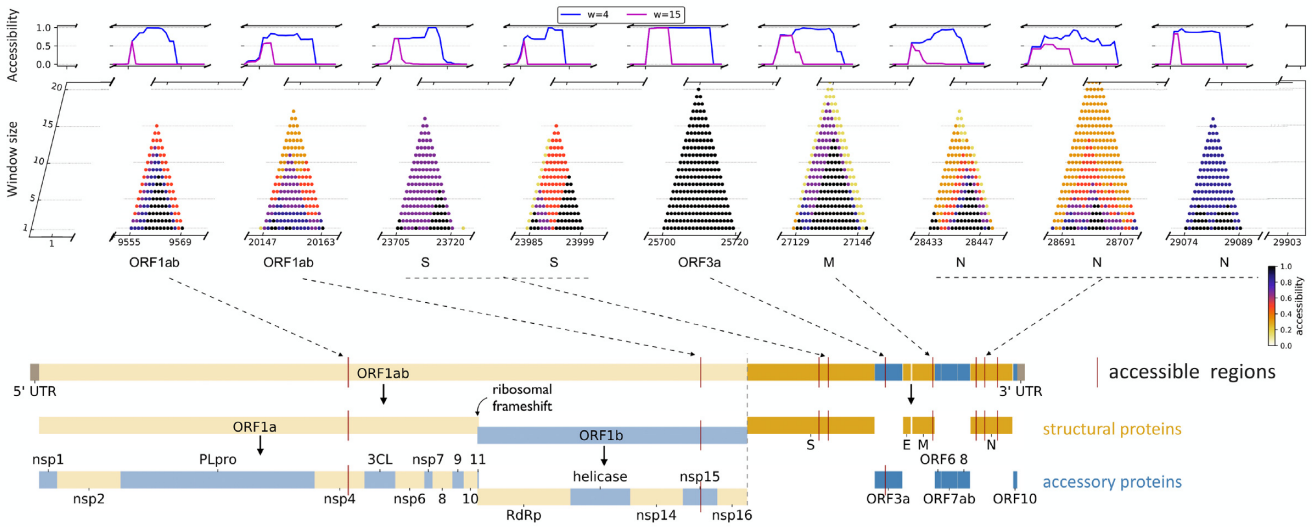

Figure 11. LinearSampling predicts nine separate accessible regions in the SARS-CoV-2 full genome. **Top**: the predicted accessibilities of window size 4 (the blue curve) and 15 (the purple curve) within the nine regions. **Middle**: accessibilities with window sizes from 1 to 20. The region [25700, 25720] in ORF3a is highly accessible, with an accessibility of >0.9. **Bottom**: the relative position of each accessible region in the full-genome of SARS-CoV-2. Most of the regions are in the last third of the genome, and three out of nine are in the N protein. Note that ORF1ab can be further divided.

ize the classical Non-Saving Sampling, and devise two new sampling algorithms, the Full-Saving Sampling that saves all hyperedges *a priori* and avoids re-computing in sampling phase, and LazySampling which eliminates redundant work and avoids unnecessary hyperedges saving via on-demand caching. We show that the LazySampling algorithm, i.e. exact partition function followed by a Lazy-Saving sampling, is the fastest among the three sampling strategies. Then we present LinearSampling, which combines LazySampling and LinearPartition.

LinearSampling is the first sampling algorithm to run in linear time without imposing constraints on the base pair distance, and is orders of magnitude faster than Vienna RNAsubopt. We conclude that

- LinearSampling runs linearly both in end-to-end and sampling-only time, and can scale up to long RNA sequence without any overflow issue;
- its sampled structures correlate better with the ground truth structures on a diverse database, and with the experimentally guided structure of SARS-CoV-2;
- it can be applied to SARS-CoV-2 to discover regions with high accessibilities, which are potential targets for diagnostics and drug design.

DATA AVAILABILITY

Our web server is available at http://linearfold.org/sampling. Our code and dataset used for benchmark are available at https://github.com/LinearFold/LinearSampling, and are uploaded to Zenodo (DOI:10.5281/zenodo.7221764). The data is also available in the RNAcentral Dataset at https://rnacentral.org/ with the accession numbers: URS00000CECFC_6239, URS00000AFBE0_579112, URS000063D36F_9606, URS000011EF6F_6239, URS0000A76ACA_3702, URS0000D54043_2711, URS00006550DA_10090, URS00003E727B_6239, URS0000505673_559292, URS00005126C9_447, URS0001980E62_6239, URS00003B07A3_559292, URS00007A2AA8_1417852, URS0000A96D67_9606, URS0000A77336_9606, URS0001BE2F5C_9606, URS00004A9E12_77133, URS0001B81E2E_3702, URS00008BC195_9606, URS00009B6D67_10090, URS0000D5C703_9606, URS0001BD0E62_9606, URS0000E6F737_132585, URS0000A8277C_9940, URS0000B60349_3981, URS00009BB93F_10090, URS00009BB84A_10090, URS0000D330D9_9601, URS0000D58D93_9606, URS00009AD4F6_9606, URS00007C400D_9606, URS00009C28A8_9606, URS00009B94BB_10090, URS0000ECC0E3_191816, URS0000D5CF18_9606, URS0001BDA28C_9606.

SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online.

FUNDING

This work is supported in part by National Institutes of Health [R35 GM145283 to D.H.M.] and National Science Foundation Grants [IIS-1817231 and IIS-2009071 to L.H.].
Conflict of interest statement. None declared.

REFERENCES

1. Eddy,S.R. (2001) Non-coding RNA genes and the modern RNA world. *Nat. Rev. Genet.*, **2**, 919–929.
2. Doudna,J.A. and Cech,T.R. (2002) The chemical repertoire of natural ribozymes. *Nature*, **418**, 222–228.
3. Kung,J.T.Y., Colognori,D. and Lee,J.T. (2013) Long noncoding RNAs: past, present, and future. *Genetics*, **193**, 651–669.
4. Miao,Z., Adamiak,R.W., Antczak,M., Batey,R.T., Becka,A.J., Biesiada,M., Boniecki,M.J., Bujnicki,J.M., Chen,S.-J., Cheng,C.Y.

et al. (2017) RNA-Puzzles Round III: 3D RNA structure prediction of five riboswitches and one ribozyme. *RNA*, **23**, 655–672.
5. Flores,S.C. and Altman,R.B. (2010) Turning limited experimental information into 3D models of RNA. *RNA*, **16**, 1769–1778.
6. Seetin,M.G. and Mathews,D.H. (2011) Automated RNA tertiary structure prediction from secondary structure and low-resolution restraints. *J. Comput. Chem.*, **32**, 2232–2244.
7. Spasic,A., Assmann,S.M., Bevilacqua,P.C. and Mathews,D.H. (2018) Modeling RNA secondary structure folding ensembles using SHAPE mapping data. *Nucleic Acids Res.*, **46**, 314–323.
8. Nussinov,R. and Jacobson,A.B. (1980) Fast algorithm for predicting the secondary structure of single-stranded RNA. *Proc. Natl Acad. Sci. USA*, **77**, 6309–6313.
9. Zuker,M. and Stiegler,P. (1981) Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Res.*, **9**, 133–148.
10. Cordero,P. and Das,R. (2015) Rich RNA structure landscapes revealed by mutate-and-map analysis. *PLoS Comput. Biol.*, **11**, e1004473.
11. Serganov,A. and Nudler,E. (2013) A decade of riboswitches. *Cell*, **152**, 17–24.
12. Ding,Y. and Lawrence,C.E. (2003) A statistical sampling algorithm for RNA secondary structure prediction. *Nucleic Acids Res.*, **31**, 7280–7301.
13. Ding,Y., Chan,C.Y. and Lawrence,C.E. (2005) RNA secondary structure prediction by centroids in a Boltzmann weighted ensemble. *RNA*, **11**, 1157–1166.
14. Huston,N.C., Wan,H., Strine,M.S., Tavares,R.d.C.A., Wilen,C.B. and Pyle,A.M. (2021) Comprehensive in vivo secondary structure of the SARS-CoV-2 genome reveals novel regulatory motifs and mechanisms. *Mol. Cell*, **81**, 584–598.
15. Lan,T.C., Allan,M.F., Malsick,L.E., Woo,J.Z., Zhu,C., Zhang,F., Khandwala,S., Nyeo,S.S., Sun,Y., Guo,J.U. *et al.* (2022) Secondary structural ensembles of the SARS-CoV-2 RNA genome in infected cells. *Nat. Commun.*, **13**, 1128.
16. Lai,W.-J.C., Kayedkhordeh,M., Cornell,E.V., Farah,E., Bellaousov,S., Rietmeijer,R., Salsi,E., Mathews,D.H. and Ermolenko,D.N. (2018) mRNAs and lncRNAs intrinsically form secondary structures with short end-to-end distances. *Nat. Commun.*, **9**, 4328.
17. Bohula,E.A., Salisbury,A.J., Sohail,M., Playford,M.P., Riedemann,J., Southern,E.M. and Macaulay,V.M. (2003) The efficacy of small interfering RNAs targeted to the type 1 insulin-like growth factor receptor (IGF1R) is influenced by secondary structure in the IGF1R transcript. *J. Biol. Chem.*, **278**, 15991–15997.
18. Tafer,H., Ameres,S.L., Obernosterer,G., Gebeshuber,C.A., Schroeder,R., Martinez,J. and Hofacker,I.L. (2008) The impact of target site accessibility on the design of effective siRNAs. *Nat. Biotechnol.*, **26**, 578–583.
19. Lu,Z.J. and Mathews,D.H. (2008) Efficient siRNA selection using hybridization thermodynamics. *Nucleic Acids Res.*, **36**, 640–647.
20. Michalik,J. (2019) In: *Non-redundant sampling in RNA bioinformatics*. PhD thesis, Université Paris-Saclay.
21. McCaskill,J.S. (1990) The equilibrium partition function and base pair probabilities for RNA secondary structure. *Biopolymers*, **29**, 1105–1119.
22. Zhang,H., Zhang,L., Mathews,D.H. and Huang,L. (2020) LinearPartition: linear-time approximation of RNA folding partition function and base-pairing probabilities. *Bioinformatics*, **36**, i258–i267.
23. Mückstein,U., Tafer,H., Hackermüller,J., Bernhart,S.H., Stadler,P.F. and Hofacker,I.L. (2006) Thermodynamics of RNA–RNA binding. *Bioinformatics*, **22**, 1177–1182.
24. Bernhart,S.H., Mückstein,U. and Hofacker,I.L. (2011) RNA accessibility in cubic time. *Algorithm. Mol. Biol.*, **6**, 3.
25. Lorenz,R., Bernhart,S.H., Höner zu Siederdissen,C., Tafer,H., Flamm,C., Stadler,P.F. and Hofacker,I.L. (2011) ViennaRNA Package 2.0. *Algorithm. Mol. Biol.*, **6**, 1.
26. Thompson,W.A., Newberg,L.A., Conlan,S., McCue,L.A. and Lawrence,C.E. (2007) The Gibbs centroid sampler. *Nucleic Acids Res.*, **35**, W232–W237.
27. Harmanci,A.O., Sharma,G. and Mathews,D.H. (2009) Stochastic sampling of the RNA structural alignment space. *Nucleic Acids Res.*, **37**, 4063–4075.
28. Li,S., Zhang,H., Zhang,L., Liu,K., Liu,B., Mathews,D.H. and Huang,L. (2021) LinearTurboFold: linear-time global prediction of conserved structures for RNA homologs with applications to SARS-CoV-2. *Proc. Natl Acad. Sci. USA*, **118**, e2116269118.
29. Gallo,G., Longo,G. and Pallottino,S. (1993) Directed hypergraphs and applications. *Discrete Appl. Math.*, **42**, 177–201.
30. Finkelstein,A. and Roytberg,M. (1993) Computation of biopolymers: a general approach to different problems. *BioSystems*, **30**, 1–19.
31. Huang,L. and Chiang,D. (2005) Better k-best parsing. In: *Proceedings of the Ninth International Workshop on Parsing Technology*. pp. 53–64.
32. Ponty,Y. and Saule,C. (2011) A combinatorial framework for designing (pseudoknotted) RNA algorithms. In: *Int'l Workshop on Algorithms in Bioinformatics*. pp. 250–269.
33. Huang,L., Zhang,H., Deng,D., Zhao,K., Liu,K., Hendrix,D. and Mathews,D. (2019) LinearFold: linear-time approximate RNA folding by 5′-to-3′ dynamic programming and beam search. *Bioinformatics*, **35**, i295–i304.
34. Mathews,D.H., Sabina,J., Zuker,M. and Turner,D.H. (1999) Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *J. Mol. Biol.*, **288**, 911–940.
35. Ponty,Y. (2008) Efficient sampling of RNA secondary structures from the Boltzmann ensemble of low-energy. *J. Math. Biol.*, **56**, 107–127.
36. Mathews,D.H., Disney,M.D., Childs,J.L., Schroeder,S.J., Zuker,M. and Turner,D.H. (2004) Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proc. Natl Acad. Sci. USA*, **101**, 7287–7292.
37. Williams,K.P. and Lau,B.Y. (2017) RNAcentral: a comprehensive database of non-coding RNA sequences. *Nucleic Acids Res.*, **45**, D128–D134.
38. Sloma,M. and Mathews,D. (2016) Exact calculation of loop formation probability identifies folding motifs in RNA secondary structures. *RNA*, **22**, 1808–1818.
39. Zadeh,J., Wolfe,B. and Pierce,N. (2010) Nucleic acid sequence design via efficient ensemble defect optimization. *J. Comput. Chem.*, **32**, 439–452.
40. Bernhart,S.H., Hofacker,I.L. and Stadler,P.F. (2006) Local RNA base pairing probabilities in large sequences. *Bioinformatics*, **22**, 614–615.
41. Manfredonia,I., Nithin,C., Ponce-Salvatierra,A., Ghosh,P., Wirecki,T.K., Marinus,T., Ogando,N.S., Snijder,E.J., van Hemert,M.J., Bujnicki,J.M. *et al.* (2020) Genome-wide mapping of SARS-CoV-2 RNA structures identifies therapeutically-relevant elements. *Nucleic Acids Res.*, **48**, 12436–12452.
42. Wu,F., Zhao,S., Yu,B., Chen,Y.-M., Wang,W., Song,Z.-G., Hu,Y., Tao,Z.-W., Tian,J.-H., Pei,Y.-Y. *et al.* (2020) A new coronavirus associated with human respiratory disease in China. *Nature*, **579**, 265–269.
43. Morandi,E., Manfredonia,I., Simon,L.M., Anselmi,F., van Hemert,M.J., Oliviero,S. and Incarnato,D. (2021) Genome-scale deconvolution of RNA structure ensembles. *Nat. Methods*, **18**, 249–252.
44. Ziv,O., Price,J., Shalamova,L., Kamenova,T., Goodfellow,I., Weber,F. and Miska,E.A. (2020) The short- and long-range RNA–RNA interactome of SARS-CoV-2. *Mol. Cell*, **80**, 1067–1077.
45. Rangan,R., Zheludev,I.N., Hagey,R.J., Pham,E.A., Wayment-Steele,H.K., Glenn,J.S. and Das,R. (2020) RNA genome conservation and secondary structure in SARS-CoV-2 and SARS-related viruses: a first look. *RNA*, **26**, 937–959.
46. Madhugiri,R., Fricke,M., Marz,M. and Ziebuhr,J. (2016) Coronavirus cis-acting RNA elements. *Adv. Virus Res.***96**, 127–163.
47. Sun,L., Li,P., Ju,X., Rao,J., Huang,W., Ren,L., Zhang,S., Xiong,T., Xu,K., Zhou,X. *et al.* (2021) In vivo structural characterization of the SARS-CoV-2 RNA genome identifies host proteins vulnerable to repurposed drugs. *Cell*, **184**, 1865–1883.
48. Iserman,C., Roden,C.A., Boerneke,M.A., Sealfon,R.S., McLaughlin,G.A., Jungreis,I., Fritch,E.J., Hou,Y.J., Ekena,J., Weidmann,C.A. *et al.* (2020) Genomic RNA elements drive phase separation of the SARS-CoV-2 nucleocapsid. *Mol. Cell*, **80**, 1078–1091.
49. Wilkinson,K.A., Merino,E.J. and Weeks,K.M. (2006) Selective 2'-hydroxyl acylation analyzed by primer extension (SHAPE): quantitative RNA structure analysis at single nucleotide resolution. *Nat. Protoc.*, **1**, 1610–1616.

50. Deigan,K.E., Li,T.W., Mathews,D.H. and Weeks,K.M. (2009) Accurate SHAPE-directed RNA structure determination. *Proc. Natl Acad. Sci. USA*, **106**, 97–102.
51. Cordero,P., Kladwang,W., VanLang,C.C. and Das,R. (2012) Quantitative dimethyl sulfate mapping for automated RNA secondary structure inference. *Biochemistry*, **51**, 7037–7039.
52. Do,C., Woods,D. and Batzoglou,S. (2006) CONTRAfold: RNA secondary structure prediction without physics-based models. *Bioinformatics*, **22**, e90–e98.
53. Huang,Y., Yang,C., Xu,X.-f., Xu,W. and Liu,S.-w. (2020) Structural and functional properties of SARS-CoV-2 spike protein: potential antivirus drug development for COVID-19. *Acta Pharmacol. Sin.*, **41**, 1141–1149.