# Analysis of RNA Sequence Structure Maps by Exhaustive Enumeration

# II. Structures of Neutral Networks and Shape Space Covering

By

Walter Grüner[a], Robert Giegerich[b], Dirk Strothmann[b], Christian Reidys[a], Jacqueline Weber[a], Ivo L. Hofacker[c], Peter F. Stadler[c,d], and Peter Schuster[a,c,d,*]

[a]Institut für Molekulare Biotechnologie
Beutenbergstraße 11, D-07708 Jena, Germany

[b]Technische Fakultät, Univ. Bielefeld
Postfach 100131, D-33501 Bielefeld, Germany

[c]Institut für Theoretische Chemie, Universität Wien
Währingerstraße 17, A-1090 Wien, Austria

[d]Santa Fe Institute
1399 Hyde Park Rd., Santa Fe, NM 87501, USA

Monatshefte für Chemie, in press

*Mailing Address:
Institut für Molekulare Biotechnologie
Beutenbergstraße 11, PF 100 813, D-07708 Jena, Germany
Phone: **49 (3641) 65 6444 Fax: **49 (3641) 65 6446
E-Mail: pks@imb-jena.de

## Abstract

The relations between RNA sequences and secondary structures are investigated by exhaustive folding of all **GC** and **AU** sequences with chain lengths up to 30. The technique of *tries* is used for economic data storage and fast retrieval of information. The computed structural data are evaluated through exhaustive enumeration and used as an exact reference for testing analytical results derived from mathematical models and sampling based on statistical methods. Several new concepts of RNA sequence to secondary structure mappings are investigated, among them the structure of *neutral networks* (being sets of RNA sequences folding into the same structure), *percolation* of sequence space by neutral networks, and the principle of *shape space covering*. The data of exhaustive enumeration are compared to the analytical results of a *random graph model* that reveals the generic properties of sequence to structure mappings based on some base pairing logic. The differences between the numerical and the analytical results are interpreted in terms of specific biophysical properties of RNA molecules.


## 1. Introduction

Folding complete sets of RNA sequences with given chain length $n$ into secondary structures provides previously unknown challenges for theoretical biochemists and computer scientists. This approach appears to be feasible only for small chain lengths ($n \leq 30$) and for sequences from two-base alphabets (**GC** or **AU**). In a previous paper [1] we presented a novel approach towards data accuumulation, retrieval and analysis derived from up to as many as $10^9$ different sequences. Here we continue by presenting data obtained from exhaustive enumeration. In section 2 we shall investigate internal structures of neutral networks, in particular the sequence of components which is tantamount to the size distributions of connected components. The method of *tries* is applied to the problem of retrieval of the desired information from properly ordered sets of folded structures. In addition the relative locations of the components of neutral networks in sequence space will be related to biochemical features of the underlying structures. In section 3 we present strong evidence for *shape space covering*: almost all common structures can be found within a fairly small ball around any random sequence. The results are particularly valuable for a comparison of real data with the predictions of the random graph approach.

## 2. The Sequence of Components

### 2.1. Algorithmic Considerations

In modeling the sequence-structure map of RNA secondary structures on the computer, we distinguish three levels of abstraction:

(1) The problem level: sequences, structures, neutral networks and related notions introduced in the previous sections.

(2) The algorithmic level: lists, graphs and other data structures representing the concepts of the problem level for algorithmic analysis.

(3) The storage level: machine words and UNIX files for storing the data structures of the algorithmic level.

The correspondence between the problem level and the algorithmic level is the content of the formal (mathematical) representation of the underlying biophysics. The correspondence between algorithmic and storage level is established by a pair of inverse functions that write to and read from background storage. The enormous volume of the data in this investigation mandates a careful choice of algorithms and data structures.

By definition, a neutral network $\mathcal{N}$ is a graph with node set $\{x \mid f(x) = s\}$ and an edge between a prir of nodes $x$ and $y$ if they differ by a single character in an unpaired position, or by a single switch of complementary characters in paired positions. The edges of $\mathcal{N}$ cannot be represented explicitly, since the space requirements would be excessive. A simple routine `neighbour`$(x, y)$ can decide whether $(x, y)$ is an edge according to the above criterion, while $\mathcal{N}$ is just represented by its node set as a list of sequences. Correspondingly, $\mathcal{N}$ is stored as a `UNIX` file of consecutive machine words. Furthermore, the exhaustive generation of the sequences and folding them into structures does not produce immediate neighborhood information. To find the neighbours of $x$ in $\mathcal{N}(s)$ takes $\mathcal{O}(\,|\mathcal{N}(s)|\,)$ steps, and so the computational effort for the analysis of connected components of $\mathcal{N}(s)$

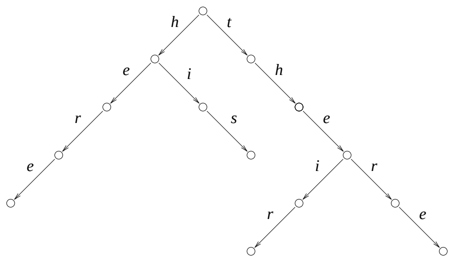
is quadratic in $|\mathcal{N}(s)|$. Some networks contain millions of nodes and hence this approach is infeasible.

We now turn to the problem of finding the neighbours of $x \in \mathcal{N}(s)$, which is the basic primitive for determining connected components in a graph. This is a well-studied problem in computer science. However, textbook algorithms usually assume an explicit representation of edges, such that an edge $(x, y)$ effectively determines the neighbour $y$ of $x$.

### *2.1.1. Tries*

With $\mathcal{N}$ being a list of nodes, our problem is rather a string processing problem and we can make use of standard string processing techniques [2, 3].

**Definition.** A *trie*[1] is a rooted tree with edges labeled by characters, such that all outgoing edges of a node carry different labels.
A *Btrie* of depth $n$ is a trie over a binary alphabet where all paths to a leaf have the same length $n$. Edge labels are optional by the convention that a left branch indicates 0, a right branch indicates 1.



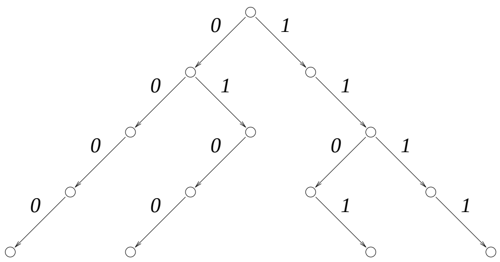




**Figure 1:** Two examples of tries. A trie for { here,his,their,there } (l.h.s) and a trie for { 0000, 0100, 1101, 1111 } (r.h.s.).

[1] "Trie" is an artificial word between "tree" and "retrieval". It is usually pronounced to rhyme with "pie".

A trie represents a set of strings by its paths read from the root towards the leafs. Figure 1 shows two tries. A Btrie of depth $n$ can represent up to $2^n$ sequences of length $n$. Yet independent of its size, it is decidable in constant time ($n$ steps at most) whether some sequence is stored in a trie. Therefore tries are ubiquitous data structures in text processing. Putting together these ideas, a neutral network $\mathcal{N}(s)$ is now represented in the algorithmic model as a pair $(s, t)$, where $t$ is the Btrie representing $\{\texttt{reduce}(s, x) \mid x \in \mathcal{N}(s)\}$.

A Btrie representing $m$ sequences of length $n$ can be constructed in $\mathcal{O}(m\,n)$ steps. Although this construction is optimal, it is too expensive to be carried out each time the data are read from the disk. Moreover, tries provide a kind of prefix sharing for the strings they represent. For example, the trie in Figure 1 (r.h.s.) represents 4 strings of length 4, but has only 13 nodes. Thus further savings in time and space arise if we compactly represent networks as tries on background storage.

The function `writeBtrie` recursively traverses a Btrie in preorder and emits

0 for a node with two subtrees,
10 for a node with a left subtree only,
11 for a node with a right subtree only.

Leafs produce no output, making use of the fact that all sequences in the Btrie are of a known (reduced) length $n'$ (being identical with the depth of the trie), as indicated by the corresponding structure. This establishes a linear encoding of a Btrie $t$ in a prefix code, and we can easily implement the inverse function `readBtrie` such that

$$\texttt{readBtrie}(n', \texttt{writeBtrie}(t)) = t\,.$$

For example, let t be the Btrie of Figure 1 (r.h.s.). We have

`writeBtrie`$(t) =$ “001011101111100” and `readBtrie`(4, “001011101111100”) $= t$.

Table 1 summarizes our algorithmic and storage model of sequence structure maps. With these techniques, the total space required for storing all neutral networks is reduced from 4.3 GB to 849 MB, an overall reduction of 80 %.

**Table 1.** Summary of Data Representations.

| Problem level | Algorithmic level | Storage level |
|---|---|---|
| sequence $x$<br>structure $s$ | $x \in \{0,1\}^{30}$<br>$s \in \text{Language}(G)$ | machine word $\geq 32$ bits<br>ASCII string |
| combinatory map $f$ | `fold()` Algorithm | `C`-Code |
| neutral network $\mathcal{N}$ | $(s,t), t$ Btrie representing $\{\texttt{reduce}(s,x) \mid x \in \mathcal{N}\}$ | Unix file containing $(s, \texttt{writeBtrie}(t))$ |
| edge $(x,y)$ in $\mathcal{N}$ | implicit if<br>$d_H(\texttt{reduce}(s,x), \texttt{reduce}(s,y)) = 1$ | — |
| Sequence of components<br>$C_1, C_2, \cdots, C_r$ | $(s, t_1, t_2, \cdots, t_u, t_0)$ with<br>$t_i$ representing $C_i$ as Btrie<br>$t_0$ Btrie of pseudo component $C_0$ | Unix file containing<br>$(s, \texttt{writeBtrie}(t_1),$<br>$\vdots$<br>$\texttt{writeBtrie}(t_n),$<br>$\texttt{writeBtrie}(t_0))$ |

*2.1.2. Connected component analysis*

Given two sequences $x$ and $y$ in $\mathcal{N}(s)$, it is of special interest whether there exists a neutral path from $x$ to $y$. Equivalently we may ask whether $x$ and $y$ belong to the same connected component of $\mathcal{N}(s)$. Computing a complete decomposition of $\mathcal{N}(s)$ into a sequence of components, stored as a list of Btries, makes available an entire solution for the whole network and further on a model describing size and quantity of components in neutral networks.

A simple decomposition algorithm starts with the extraction of an initial sequence $x$ from $\mathcal{N}(s)$. Finding and removing all neighbors of $x$ from $\mathcal{N}(s)$ is the the essential part of the computation. This step is detailed below. The neighbors of $x$ are temporarily stored in a list $\mathcal{L}$ while the initial sequence $x$ can be saved in a Btrie $C$ representing the arising component. Consecutively we take the first element $l_1$ from $\mathcal{L}$, compute and remove its neighbors from the decreasing $\mathcal{N}(s)$ and append them to $\mathcal{L}$. Delete $l_1$ from $\mathcal{L}$, then repeat the procedure.

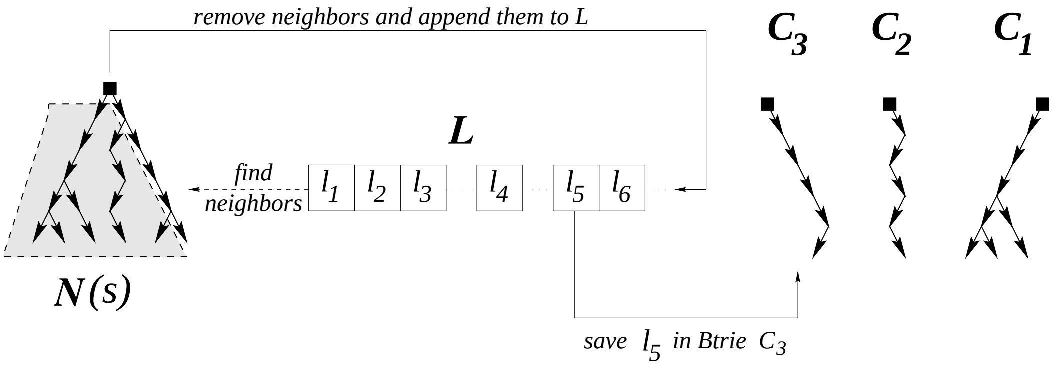



**Figure 2:** A snapshot from component analysis. Components $C_1$ and $C_2$ are completed, while $C_3$ is under construction. $C_1, \ldots C_3$, $\mathcal{L}$ and $\mathcal{N}(s)$ are disjoint and their union constitutes the original neutral network of $s$. Once the components are stored on disk they are deleted from $\mathcal{N}(s)$.

Finding $\mathcal{L}$ empty stops the procedure with the resulting connected component stored in the Btrie $C$. A complete decomposition of $\mathcal{N}(s)$ is achieved by restarting the process with new initial sequences and extracting the corresponding components until $\mathcal{N}(s)$ is empty. We get a sequence $C_1, C_2, \ldots, C_k$ of components as a list of Btries that are stored in the usual way. Space optimization without extra work is attained by storing all components consisting of only one sequence in one pseudo-component $C_0$. Finding and removing all sequences neighboring to a sequence x is done by the function `findrem` specified by the equations (1)-(7) below. If $\texttt{findrem}(d, x, t) = (t', [x_1, \ldots, x_r])$ then $d_H(x, x_i) = d$, and $t'$ is the Btrie obtained from t by removing $x_1, \ldots, x_r$.

A sequence $x$ has at most $n$ neighbors, and the effort of finding and removing them is $\mathcal{O}(n^2)$. For $|\mathcal{N}(s)| = m$, the complete algorithm is $\mathcal{O}(m \cdot n^2)$ in the worst case. For dense networks with $m \approx 2^n$, this approaches $\mathcal{O}(m \cdot (log\ m)^2)$, making connectivity analysis feasible.

(1) $\texttt{findrem}(1, [\,], \texttt{Leaf}) = (\texttt{Empty}, [[\,]])$; neighbor found and removed.

(2) $\texttt{findrem}(n, w, \texttt{Empty}) = (\texttt{Empty}, [\,])$; no neighbors in empty Btrie.

(3) $\texttt{findrem}(0, aw, \texttt{Node}(l, r))$

$$= \big(\texttt{red}(\texttt{Node}(l', r'))\ \texttt{map(0,xs')} +\!\!+ \texttt{map}(1, ys)\,\big), \text{ where}$$

$$(l', xs) = \texttt{findrem}(a, w, l) \text{ and}$$

$$(r', ys) = \texttt{findrem}(1 - a, w, r).$$

(4) $\texttt{findrem}(1, 0w, \texttt{Node}(l, r)) = (\texttt{red}(\texttt{Node}(l', r)), \texttt{map}(0, xs))$, where

$$(l', xs) = \texttt{findrem}(1, w, l).$$

(5) $\texttt{findrem}(1, 1w, \texttt{Node}(l, r)) = (\,\texttt{red}(\texttt{Node}(l, r')), \texttt{map}(1, ys)\,)$, where

$$(r', ys) = \texttt{findrem}(1, w, r).$$

(6) $\texttt{red}(\texttt{Node}(l, r)) = \begin{cases} \texttt{Empty} & \text{if } \ l = r = \texttt{Empty} \\ \texttt{Node}(l, r) & \text{otherwise.} \end{cases}$

(7) $\texttt{map}(a, [w_1, \ldots, w_n]) = [aw_1, \ldots, aw_n]$ for $n \geq 0$.

**Figure 3:** Specification of the function `findrem`.

### *2.1.3. A remark on programming methodology*

Implementing connectivity analysis on the computer required close cooperation between biologists and computer scientists. Such interdisciplinary work has its own challenges. In the early state, it is essential that the mutual understanding is validated and exercised by a quick implementation of algorithmic ideas and an exploration of alternatives. In our case `Miranda`[2], a purely functional language was chosen to prototype the algorithmic model. Due to the high level of abstraction, a functional program for the above analysis is very short (only 2 pages) and clearly reflects the design decisions. Only after our ideas had been tested and their benefits on space and time requirements had been demonstrated, the implementation in `C` was developed.

[2] `Miranda` is a trademark of Research Software Ltd.

2.2. A Sample Session: GC12

The raw data, represented by an exhaustive list of all sequences contained in different files according to their structure are generated using the command `ShapeSpace -l 12 -g`. Here `-l` is an option for the chain length and `-g` means generation of the raw data. This command creates a hierarchy of directories containing one file for each structure. The entries are then sorted and the index file `GC_12.ind` is created in which each line contains the rank of a structure, the size of its preimage, the dot-bracket representation of the structure and the location of the actual data in the directory tree.

The computation of the sequence of components is performed using the command `ShapeSpace -l 12 -d`. It requires that the index file has been created first. The data are written into a hierarchy of directories such that each directory contains no more than 1000 entries. For instance, the data belonging to the structure of rank 22714 in **GC30** will appear in the directory `GC_30/22/` with filename 714. For long chains this presents a formidable task. The computational bottle necks are the following:

(1) CPU resources are crucial for the folding and for the sequence of components decomposition.
(2) Memory is the limiting resource while the raw data are generated.
(3) Access to disk space becomes a problem in the case of longer sequences as partial results have to be moved to and from scratch disks.

The 4096 sequence of the sequence space **GC12** fold into 31 different secondary structures. The most frequent structure is the open structure '............' which is obtained from about 18% of all sequences. The neutral network of the open structure is exceptional in the case of short chains because it is connected. For longer chains it decomposes into (at least) two patches, one of which is centered around poly-**G** and the other one centered poly-**C**.

The most frequent structures have connected preimages consisting of only one (single) component. The neutral networks of the less frequent common structures,

**Table 2.** Decomposition of the Sequence Space **GC12**.

| Rank | Size | Sequence | $\lambda_u$ | $\lambda_p$ | Sequence of Components | | |
|---|---|---|---|---|---|---|---|
| 1 | 767 | ............ | 0.502 | 0.000 | 767 | | |
| 2 | 352 | (((...)))... | 0.766 | 0.771 | 352 | | |
| 3 | 349 | ...(((...))) | 0.762 | 0.755 | 349 | | |
| 4 | 258 | ..(((....))) | 0.760 | 0.514 | 258 | | |
| 5 | 249 | .((((...)))) | 0.982 | 0.972 | 249 | | |
| 6 | 247 | ((((...)))). | 0.978 | 0.964 | 247 | | |
| 7 | 235 | (((....))).. | 0.752 | 0.468 | 235 | | |
| 8 | 227 | ((((....)))) | 0.970 | 0.879 | 227 | | |
| 9 | 179 | .(((...))).. | 0.572 | 0.719 | 92 | 87 | |
| 10 | 175 | ..(((...))). | 0.570 | 0.697 | 89 | 86 | |
| 11 | 162 | .(((.....))) | 0.570 | 0.601 | 88 | 74 | |
| 12 | 161 | (((.....))). | 0.559 | 0.605 | 89 | 72 | |
| 13 | 141 | .(((....))). | 0.563 | 0.548 | 74 | 67 | |
| 14 | 131 | (((......))) | 0.537 | 0.453 | 72 | 59 | |
| 15 | 84 | .....((...)) | 0.429 | 0.381 | 63 | 21 | |
| 16 | 70 | ((...))..... | 0.393 | 0.286 | 70 | | |
| 17 | 43 | ....((...)). | 0.320 | 0.326 | 32 | 11 | |
| 18 | 41 | ..((...))... | 0.378 | 0.244 | 21 | 20 | |
| 19 | 40 | .((...)).... | 0.319 | 0.300 | 29 | 11 | |
| 20 | 39 | ...((...)).. | 0.353 | 0.308 | 21 | 18 | |
| 21 | 31 | ..((......)) | 0.468 | 0.000 | 31 | | |
| 22 | 20 | ...((.....)) | 0.338 | 0.000 | 20 | | |
| 23 | 15 | .((.......)) | 0.367 | 0.000 | 15 | | |
| 24 | 14 | .((......)). | 0.304 | 0.000 | 9 | 5 | |
| 25 | 14 | ((.....))... | 0.286 | 0.000 | 14 | | |
| 26 | 14 | ((......)).. | 0.339 | 0.000 | 11 | 3 | |
| 27 | 13 | ..((.....)). | 0.250 | 0.000 | 7 | 4 | 2 |
| 28 | 11 | .((.....)).. | 0.318 | 0.000 | 8 | 3 | |
| 29 | 11 | ((.......)). | 0.318 | 0.000 | 11 | | |
| 30 | 2 | .((.(...).)) | 0.167 | 0.000 | 2 | | |
| 31 | 1 | ((.(...).)). | 0.000 | 0.000 | 1 | | |

Common structures have at least $\lceil 4096/31 \rceil = 133$ realizations in this example, therefore $r_c = 13$.

however, decompose into two components. The reason for this will be discussed in the following section.

### 2.3. Connectivity in GC30

An analysis of the sequence of components of **GC30** can be used to test the random graph theory discussed in section 2.5.4. It predicts that a neutral network corresponding to a structure with $\lambda_u > 1/2$ and $\lambda_p > 1/2$ should form a single connected component, *at least in the limit of long sequences.* Since a chain length of $n = 30$ is far away from this limit we cannot expect to find a perfect picture. The data show two types of deviations from the predicted behavior:

(1) Many neutral networks consist of one (or a few) large components and a few isolated points (or very small components). The overwhelming fraction of the sequences belongs to the larger components. We can attribute the existence of the isolated points and tiny components to finite size effects.

(2) The splitting of neutral networks in a small number of large components of comparable size, however, is not consistent with the prediction of the random graph model. In order to explain this behavior we need to consider structural features in some more detail. A few typical examples can be found in table 3.

**Table 3.** Selected Sequences of Components in **GC30**

| Rank | Structure | $\lambda_u$ | $\lambda_p$ | Sequence of Components[†] |
|---:|---|---|---|---|
| 1 | ........(((((((((....))))))))) | 0.860 | 0.895 | 1568485 |
| 5 | ..........((((((((....)))))))) | 0.614 | 0.747 | 1328606 |
| 6 | ((((((((....)))))))).......... | 0.611 | 0.742 | 1314205, [2] |
| 7 | .......((((((((.......)))))))) | 0.666 | 0.748 | 637048, 603435 |
| 10 | ((((((((......)))))))))........ | 0.652 | 0.751 | 622112, 583934 |
| 50 | .......((((((((......)))))))). | 0.570 | 0.749 | 77540, 167420, 162767, 157777 |
| 1974 | ......((..(((((((...))))))).)) | 0.562 | 0.576 | 118307 |
| 1975 | .((((((.......)))))).(((...))) | 0.367 | 0.459 | 33824, 31163, 30751, 22388, [173] |
| 1976 | ((((..((((....)))).))))....... | 0.420 | 0.312 | 117782, [514] |
| 1983 | .......((.((((((.....)))))).)) | 0.360 | 0.420 | 53691, 34137, 17123, 12379, 225, 215, [245] |
| 1984 | ...((((((.....)))))).......... | 0.323 | 0.499 | $\lvert f^{-1} \rvert = 117971$ in 455 components* |
| 3030 | ..((((((.........))))))....... | 0.305 | 0.336 | $\lvert f^{-1} \rvert = 88811$ in 804 components* |
| 4723 | ..((((((.........))))))....... | 0.286 | 0.373 | $\lvert f^{-1} \rvert = 58580$ in 649 components* |
| 13135 | ((((......)))).((((......)))). | 0.214 | 0.246 | $\lvert f^{-1} \rvert = 13737$ in 503 components* |

* A graph of the distribution of component sizes can be found in [4].

† Very small components are not shown in detail here. A number in square brackets gives the total number of sequences in them.
Data are taken from reference [5].

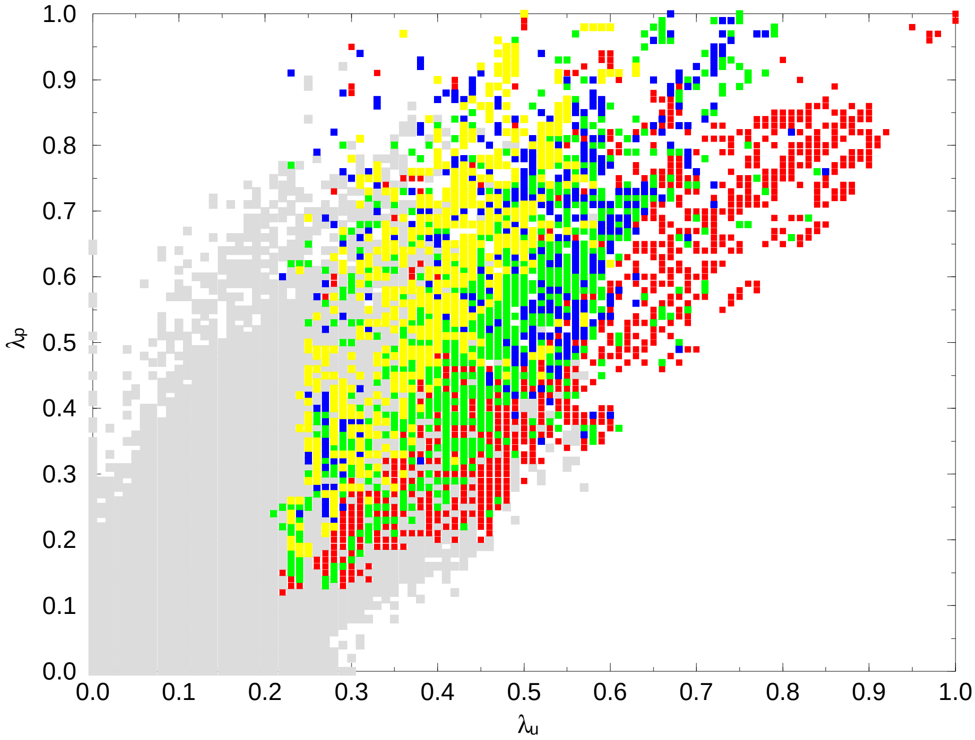



**Figure 4:** Types of neutral networks in **GC30**. The color code indicates the "type" of neutral networks, depending on the values of $\lambda_u$ and $\lambda_p$. grey: networks consisting of more than 4 large components, and very small networks, corresponding to less than 5000 sequences. A component is considered as large in this plot if its size is at least 10% of the size of the largest component. Red: one (single) component, apart from (possibly) a few tiny components. green: essentially two components, blue: three components, yellow: four components. If more than one data point falls within a square then the color that appears most often is shown.

Structures with $\lambda_u$ and $\lambda_p$ above the threshold values show one, two or four (almost equally sized) components when they have none, one or two structural elements that allow to form additional base pairs (figure 5). These elements are, for example, stacking regions with two dangling ends, hairpin loops with five or more members or sufficiently large bulges, internal loops or multi-loops. Structures with these elements are very likely to form the additional base pair whenever complementary bases are in the appropriate positions. This is more likely to be avoided (and the structure in question is more likely to be formed as minimum free energy structure from the corresponding compatible sequence) when there is an excess of

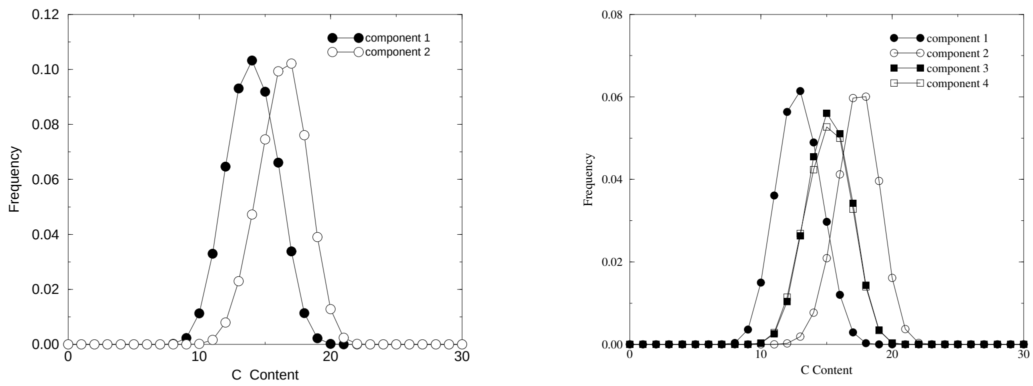



**Figure 5:** Large neutral networks can decompose into a small number of components with almost equal sizes which differ by their **G**/**C** ratio. A neutral network with two componenets as exhibited by the structure with rank 7 (see table 3) is shown on the l.h.s. On the r.h.s. we present a neutral network with four components as found with the structure of rank 50.

**G** or **C**, $\delta\#\mathbf{G}$ or $\delta\#\mathbf{C}$ respectively, in the base composition of the sequence (the **G**/**C** ratio being significantly different from one). One structural element that allows to form an additional base pair thus implies a bias towards more **G** or more **C** in the average composition of the sequences just as it is found with the two components in a two-component neutral network (figure 5, l.h.s.). Inspection of table 3 shows that precisely the same types of structural elements are responsible for the appearence of two equally sized componenets also in the case of these very short sequences.

Two independent structural elements are superimposed in the relative **G**/**C** content and thus we have

$$\delta\#\mathbf{G}\,\&\,\delta\#\mathbf{G}\ ,\ \delta\#\mathbf{G}\,\&\,\delta\#\mathbf{C}\ ,\ \delta\#\mathbf{C}\,\&\,\delta\#\mathbf{G}\ \text{and}\ \delta\#\mathbf{C}\,\&\,\delta\#\mathbf{C}\ .$$

The first and the last combination show a net **G** or **C** bias whereas compensation brings the other two cases back into the middle of sequence space. Exactly this distribution of the four components is found by analyzing the data of the reference sample **GC30**. Three components commonly represent a special case of four components where the two central components happen to be connected.

For structures with $\lambda_u$ and $\lambda_p$ way below the critical values we find many components and a characteristically decreasing size distribution of these components, see figure 4, as predicted by random graph theory. A more detailed analysis of the relationships between the parameters $\lambda_p$ and $\lambda_u$ and the particular features of the corresponding secondary structure graphs will be published elsewhere.

## 3. Shape Space Covering

Extensive computational studies on RNA sequence structure maps [6, 7, 8] provide strong evidence for the existence of a neighborhood (represented as a high-dimensional ball in sequence space) with a radius much smaller than the chain length around every random sequence such that this neighborhood contains sequences whose structures include almost all common shapes. Full enumeration of a sequence space allows us of course to check this conjecture, and provided it is true, to estimate the "covering radius".

To this end we have exhaustively generated all sequences within a ball of radius $R$ in sequence space. The *covering fraction* $\varphi(R, r)$ is defined as the number of all structures with rank not exceeding $r$ encountered within this ball divided by $r$, the rank of the rarest structure under consideration.

Since the shape space covering conjecture claims that all common structures can be found within a small radius $R^*$, the covering fraction $\varphi(R, r_c)$ is of particular interest. The data shown in the l.h.s. of figure 6 are obtained by averaging $\varphi(R, r_c)$ over 200 balls with randomly chosen midpoints.

The $\varphi(R, r)$ plots can be fitted very well by an error function

$$\varphi(R, r) = \frac{1}{2}\mathrm{erf}\ (a(R - R^*))$$

The parameters $R^*$ and $a$ correspond to the threshold value and the slope at the inflection point, which measures the sharpness of the transition. The choice of the

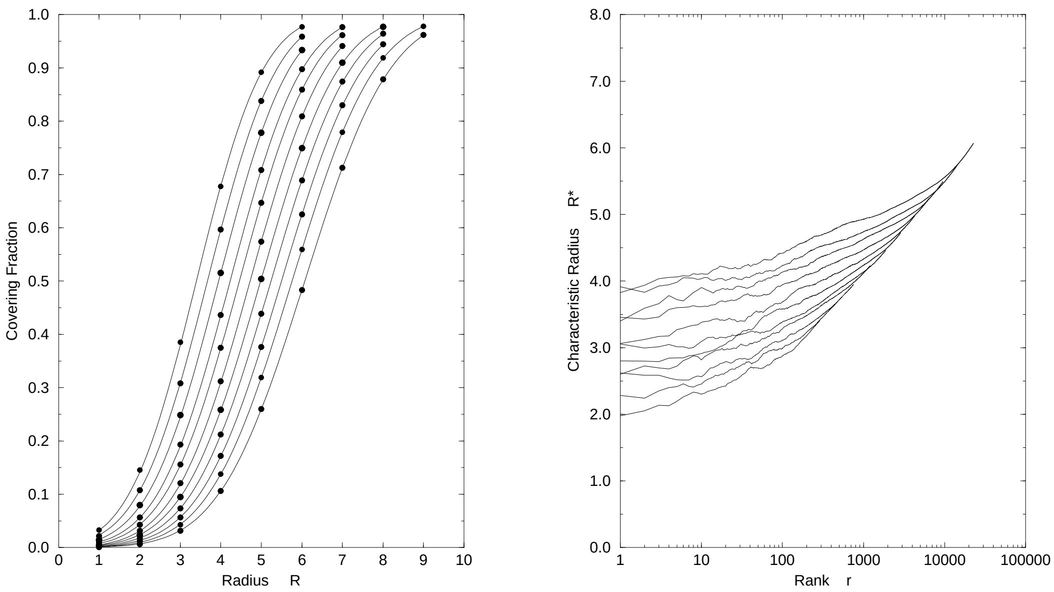



**Figure 6:** Shape space covering of **GC** sequences. The covering fraction $\varphi(R, r_c)$ for sequences with different chain length from $n = 20$ (left most curve) to $n = 30$ (right most curve) is shown on the r.h.s. On the l.h.s. we present the characteristic radius $R^*$ as a function of the maximum rank up to which the structures are considered. The plot shows results for chain lengths from $n = 20$ (lowest curve) through $n = 30$ (top most curve).

error-function in figure 6 is arbitrary. In order to check the robustness of the data with respect to the choice of the fitting curve we have used

$$\tilde{\varphi}(R) = \frac{\exp(\, a(R - R^*)\,)}{1 + \exp(\, a(R - R^*)\,)}$$

as an alternative. The data in table 4 show a very good agreement of the $R^*$ values obtained from $\varphi$ and $\tilde{\varphi}$, respectively.

The characteristic radius $R^*$ will depend on the maximum rank $r$ up to which structures are considered. It turns out, however, that only the most frequent structures have a drastically reduced value of the characteristic radius $R^*$, see the r.h.s. of figure 6. Since the shape space covering conjecture claims that almost all of the $r_c$ common structures can be found within a fairly small radius we refer to the characteristic value belonging to $r_c$ as the *covering radius* $\hat{R}$.

**Table 4.** Shape Space Covering Radii for **GC** Sequences.

| $n$ | $r_c$ | $\hat{R}\,[\varphi]$ | $\hat{R}\,[\tilde{\varphi}]$ | $\varphi'(\hat{R})$ | $\tilde{\varphi}'(\hat{R})$ |
|---|---|---|---|---|---|
| 20 | 287 | 3.397 | 3.471 | 0.307 | 0.350 |
| 21 | 462 | 3.672 | 3.680 | 0.297 | 0.318 |
| 22 | 753 | 3.948 | 3.948 | 0.291 | 0.299 |
| 23 | 1203 | 4.228 | 4.226 | 0.285 | 0.288 |
| 24 | 1867 | 4.459 | 4.464 | 0.278 | 0.283 |
| 25 | 2870 | 4.724 | 4.724 | 0.273 | 0.278 |
| 26 | 4303 | 4.984 | 4.985 | 0.264 | 0.269 |
| 27 | 6373 | 5.237 | 5.238 | 0.258 | 0.262 |
| 28 | 9580 | 5.497 | 5.497 | 0.253 | 0.255 |
| 29 | 14642 | 5.759 | 5.760 | 0.248 | 0.251 |
| 30 | 22719 | 6.069 | 6.069 | 0.240 | 0.245 |

Table 4 presents the covering radii $\hat{R}$ for different chain lengths. A linear regression yields

$$\begin{array}{lclcl}(0.2627 \pm 0.0016)n & - & (1.84 \pm 0.04) & \rho = 0.9998 & \text{from } \varphi \\ (0.2591 \pm 0.0020)n & - & (1.74 \pm 0.05) & \rho = 0.9997 & \text{from } \tilde{\varphi}.\end{array}$$

The slopes of both $\varphi$ and $\tilde{\varphi}$ vary only slowly with $n$. The data are consistent with a logarithmic model

$$\begin{array}{lclcl}(-0.587 \pm 0.015)\log(n) & + & (0.252 \pm 0.021) & \text{from } \varphi \\ (-0.782 \pm 0.056)\log(n) & + & (0.538 \pm 0.079) & \text{from } \tilde{\varphi}.\end{array}$$

Thus the transition becomes sharper with increasing chain length, at least if one considers the scaled quantity $R/n$. In other words, there is a constant $\vartheta^*$ such that, in the limit of large chain lengths, we have complete covering if the scaled radius $R/n$ exceeds $\vartheta^*$, while only a negligible fraction of the structures is covered by balls of size $R/n < \vartheta^*$. The constant $\vartheta^*$ depends of course on the alphabet in question. For **GC** the data shown above are consistent with $\vartheta^*_{\mathbf{GC}} = 0.261 \pm 0.002$.

Alternatively, the covering radius can be estimated by measuring the minimum distance that is necessary to find a given structure from a chosen starting sequence, and averaging over the starting sequences and target structures weighted by their preimage sizes. An upper bound for this mean covering radius can be obtained from computer experiment even for larger chain lengths [7, 9].

A lower bound for the covering radius can be estimated as the mean number of base pairs separating an arbitrary sequences from one that is compatible with some given structure: for a typical structure $s$ with $b$ base pairs $(1-p)b$ point mutations are needed on average to go from an arbitrary sequence $x$ to a compatible sequence $x' \in \mathbf{C}[s]$, where $p = \frac{\beta}{\alpha^2}$ is the probability that two randomly bases can form a base pair.

**Table 5.** Lower and upper bounds for shape space covering radius.

| $n$ | **AUGC** | | **GC** | | **AU** | |
|---|---|---|---|---|---|---|
| 30 | | 5.2 | | 6.3 | | 4.3 |
| 50 | 6.5 | 9.2 | 8.5 | 10.7 | 6.0 | 7.0 |
| 70 | 10.0 | 13.7 | 12.5 | 15.6 | 9.3 | 11.5 |
| 100 | 15.2 | 20.5 | 18.6 | 22.9 | 14.6 | 17.3 |

Data taken from [9]

The covering radii obtained from the exhaustive enumerations (Table 4) and the upper bounds [9] (Table 5) are in excellent agreement. Figure 7 shows the chain length dependence of the covering radii obtained by the different methods. The slight difference between the computed slopes for the upper bound, $0.238 \pm 0.004$, as compared to 0.261 from the enumeration results may be due to finite size effects.

**Table 6.** Alphabet dependence of $\vartheta^*$.

| Estimate | **AUGC** | **GC** | **AU** |
|---|---|---|---|
| Lower Bound | 0.17 | 0.20 | 0.17 |
| Upper Bound | $0.220 \pm 0.004$ | $0.238 \pm 0.004$ | $0.19 \pm 0.01$ |
| Enumeration | $*$ | $0.261 \pm 0.002$ | |

$^*$ Exhaustive computations are not feasible for four-letter alphabets

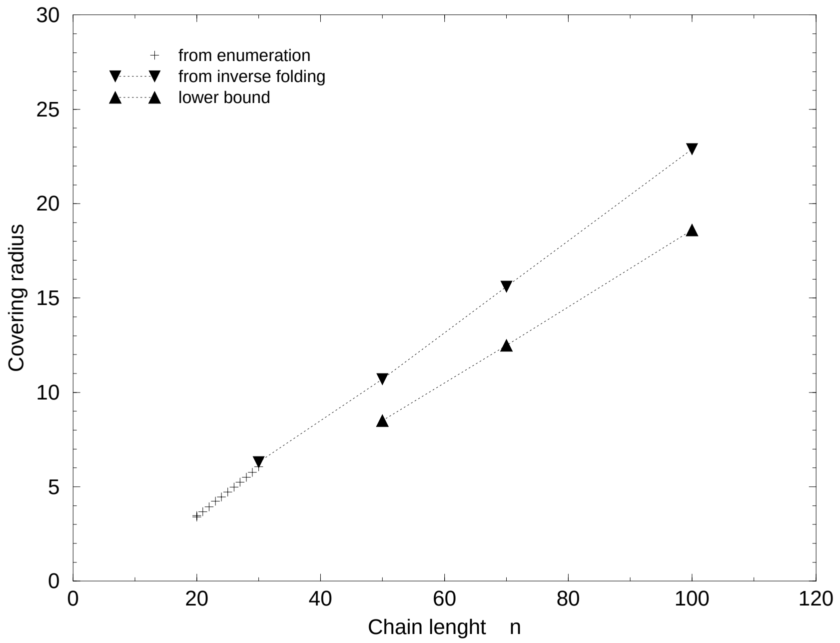



**Figure 7:** Shape space covering radii estimated by different methods for **GC** sequence spaces. ∘ represents the data from table 4, triangles refer to data taken from table 5.

## 4. Conclusions

Folding of all RNA sequences into secondary structures and retrieval of information by exhaustive enumeratioon is limited to short chain lengths ($n \leq 30$) and two-letter alphabets (**GC** and **AU**) since the numbers of sequences to be handled in these cases do not appreciably exceed $10^9$. Even then retrieval of information on more elaborate properties like the sequence of components in neutral networks is so demanding with respect to computer time and data storage requirements that special methodology is required. The technique of *tries* is applied successfully and several issues concerning comparisons of data from folding with the results of random graph theory could be readily addressed. Examples of such issues are the verification of the shape space covering conjecture, the numbers of large components in neutral networks and their distributions in sequence space. Random graph theory predicts a single *giant* component. In reality, one, two, three or four large components (of almost equal size) were found frequently. The deviations from theory were explained by structural features that are inaccessible to the random graph approach. Another important issue is the verification of the shape space covering conjecture. It could be shown that shape space covering does indeed occur with short sequences in the two-letter alphabets. Moreover, the shape space covering radius obtained by exhaustive enumeration agrees very well with a previously derived upper bound from structure statistics.

Exhaustive enumeration is shown to be a highly useful tool for the creation of exact reference samples that can be used to verify results obtained from model assumptions and to prove conjectures derived from statistical approaches. It has to be stressed, however, that it is limited to short chain lengths and to two letter alphabets. Otherwise, the numbers of sequences certainly are prohibitive for exhaustive folding because of two reasons: high demands on computer time **and** data storage capacities. On the other hand, the numbers of (minimum free energy) RNA secondary structures is certainly not larger in case they are derived from four letter alphabets. We can expect therefore statistical methods to be directly applicable to the derivation of results for **GCAU** sequences analogous to those of exhaustive enumeration for **GC** or **AU** sequences. These approach again will be confined to short sequences. Investigations of longer RNA molecules will require entirely different techniques. Investigations along these lines are under way.

## Acknowledgements

The work carried out in Wien has been supported financially by the Austrian *Fonds zur Förderung der wissenschaftlichen Forschung* (Projects No.9942-PHY and No.10578-MAT. Support of research in Jena by the Commission of the European Communities is gratefully acknowledged (CEC Contract Study PSS*0846). Our research was also continuously supported by the Santa Fe Institute.

## References


[1] W. Grüner, R. Giegerich, D. Strothmann, C. Reidys, J. Weber, I.L. Hofacker, P.F. Stadler, and P. Schuster. Analysis of RNA sequence structure maps by exhaustive enumeration. I. Neutral networks. *Mh.Chem.*, 127:***–***, 1996.

[2] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, Reading, MA, 1982.

[3] D.E. Knuth. *The Art of Computer Programming, Vol.3: Sorting and Searching*. Addison-Wesley, Reading, MA, 1973.

[4] C. Reidys, P.F. Stadler, and P. Schuster. Generic properties of combinatory maps - Neutral networks of RNA secondary structures. *Bull. Math. Biol.*, 1995. Submitted. SFI-Preprint Series No. 95-07-058.

[5] W. Grüner. *Evolutionary Optimization on RNA Folding Landscapes*. PhD thesis, Universität Wien, 1994.

[6] W. Fontana, D A.M. Konings, P.F. Stadler, and P. Schuster. Statistics of RNA secondary structures. *Biopolymers*, 33:1389–1404, 1993.

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

[8] P. Schuster. How to search for RNA structures. Theoretical concepts in evolutionary biotechnology. *Journal of Biotechnology*, 41:239–257, 1995.

[9] I.L. Hofacker. *A Statistical Characterisation of the Sequence to Structure Mapping in RNA*. PhD thesis, Universität Wien, 1994.

# Table of Contents