The impact of forbidden substrings on the connectivity of the Hamming graph of RNA sequences

Maher Mallem,
M2 Computer Science student at ENS Paris-Saclay,
supervised by Alain Denise and Yann Ponty at LRI and LIX, France

March 20th - August 7th 2019

Contents

1 Introduction

General context. RNA sequences conceal a lot of structural properties that have yet to be discovered. One of the main problems around RNA sequences is RNA folding. RNAs are three dimensional objects that fold in particular ways. Unlike DNA structures which are two sequences that connect each other, RNA structures are a unique sequence that makes nucleotide pairs with itself and fold in the 3D space as result. The regular A-U and G-C connections are gathered in what is called a secondary structure, while the less standard and less energetic connections that rule the 3D shape of the RNA are called pseudoknots. In the folding problem, one must predict from the RNA sequence, as a "text file", their secondary structure - or their most probable one. Our problem of interest is the RNA inverse folding problem: given a secondary structure, find RNA sequences that will - or will be very likely - to fold this way.

RNA design. First introduced in [HFS+93], RNA design has been studied extensively over the past decades. The simplest form of this problem has only recently been proven NP-complete in [BRS17]. [Pon14] presents the main methods that have been developed over the years, while [CRR+17] lists the currently available software solving the problem and compares their performance. The method we are interested in is the grammar-based one described in [ZPV+13] - and later improved in [LGDP15]. There are two main steps. First a few candidates are determined with help from a grammar based on the secondary structure. Then these candidates are improved through local search (based on the Hamming distance for example).

With forbidden motifs. On top of a secondary structure and a set of mandatory motifs, the user can specify a set $\mathcal{F}$ of forbidden motifs, which is quite uncommon in the field and could definitely be useful biologically speaking. Certain motifs correspond to biological functions. As an example $\mathcal{F}$ could be used to discard the types of RNA that are already known and make it easier to find new RNA sequences. However, since including such a set of forbidden motifs is uncommon, little to no work has been done about the impact of such forbidden motifs on the local search. One of the critical properties that need to be kept is the search space connectivity. Without connectivity there is no guarantee to eventually find an optimal solution. A naive connectivity check on the whole search space would take time $\mathcal{O}(|\Sigma|^n)$, which is too much when the desired length of the RNA sequences is in the hundreds or the thousands. In this work, we investigate the matter. We first establish a few sufficient conditions on $\mathcal{F}$ to guarantee connectivity. Then we investigate the properties of a few related graph structures - De Bruijn graphs and Aho-Corasick automata - in order to develop partial connectivity tests that have a lower time complexity.

2 The problem

2.1 Preliminary definitions

Let $n \in \mathbb{N}, n \geq 2$.
Let Σ be an alphabet, $|\Sigma| \geq 2$. (We are especially interested in the case $\Sigma = \{A, U, G, C\}$).
Let $\mathcal{F}$ be the set of forbidden motifs.
Let $\mathcal{L}_{\mathcal{F},n}$ be the set of words in Σ^n that do not contain any motif in $\mathcal{F}$.
Let $H(w, w')$ be the Hamming distance between two words w, w' in Σ^n.

Let $m(\mathcal{F}) \stackrel{\text{def}}{=} max_{f \in \mathcal{F}} |f|$.
We assume that $n \geq m(\mathcal{F})$. (Otherwise some forbidden motifs would never be problematic).

2.2 The statement

General problem

> Input: $n \geq 2$, $\mathcal{F}$ a set of forbidden motifs, $\delta : \mathcal{L}_{\mathcal{F},n} \to \mathcal{L}_{\mathcal{F},n}$ a neighborhood function on $\mathcal{L}_{\mathcal{F},n}$
>
> Question: Is the graph $G = (\mathcal{L}_{\mathcal{F},n}, \delta)$ *strongly connected*?

In this work, we fix the neighborhood function as the k-Hamming neighborhood δ_k for some $k \in [1..n]$.

With the k-Hamming neighborhood

Definition 1. *Given $k \in \mathbb{N}^*$, we define δ_k the k-Hamming neighborhood as follows:*

$$\forall w \in \mathcal{L}_{\mathcal{F},n}, \delta_k(w) = \{w' \in \mathcal{L}_{\mathcal{F}} \mid H(w, w') \leq k\}.$$

With $k = n$, any $w \in \mathcal{L}_{\mathcal{F},n}$ can be changed into any other $w' \in \mathcal{L}_{\mathcal{F},n}$ in one step. Hence $G = (\mathcal{L}_{\mathcal{F},n}, \delta_n)$ is always strongly connected. Thus with the k-Hamming neighborhood a variant of the general problem can be considered:

> Input: $n \geq 2$, $\mathcal{F}$ a set of forbidden motifs
>
> Question: the minimal $k \in \mathbb{N}^*$ such that the graph $G_{\mathcal{F},n,k} \stackrel{\text{def}}{=} (\mathcal{L}_{\mathcal{F},n}, \delta_k)$ is strongly connected.

Remark 2. *Since δ_k is symmetric for any $1 \leq k \leq n$, $G_{\mathcal{F},n,k}$ is connected iff it is strongly connected. Thus we will use "connected" and "strongly connected" interchangeably when considering k-Hamming neighborhoods.*

3 First results

3.1 With one motif

Consider the case where $\mathcal{F}$ contains a single motif: $\mathcal{F} = \{f\}$.
Then $k = 1$ is sufficient to guarantee strong connectivity.

Result 3. *$\forall f \in \Sigma^+$, $G_{\{f\},n,1}$ is strongly connected.*

Proof. Let w and w' be two words in $\mathcal{L}_{\{f\}}$ (of length n).
As $f \neq \epsilon$, f can be written letter by letter as follows: $f = f_1 \dots f_{|f|}$.
Since $|\Sigma| \geq 2$, let $a \in \Sigma$ such that $a \neq f_1$.
We show that there is a path from w to a^n.
To do so, from left to right we replace each letter in w by a (or keep it the same if already an a).
Formally, from $w = w_1 \dots w_n$ we define the sequence $(u_i)_{0 \leq i \leq n}$ of intermediate words:

$$\forall 0 \leq i \leq n, u_i = a^i w_{i+1} \dots w_n.$$

Then:

- $\forall 0 \leq i \leq n-1, H(u_i, u_{i+1}) \leq 1$,
- for every i in $[1..n]$ we must prove that u_i is in $\mathcal{L}_{\{f\}}$. By contradiction, suppose that f appears in a u_i. Let j be the position in u_i of the leftmost letter of this occurrence of f.
  - if $j \leq i$: then the leftmost letter of f would be a, which is not by definition of a.
  - if $j > i$: then this occurrence of f would be a factor of w, which it cannot be since $w \in \mathcal{L}_{\{f\}}$.

  Contradiction. Hence every u_i is in $\mathcal{L}_{\{f\}}$.

This proves that there is a path from w to a^n with δ_1 as the neighborhood function.
The same can be done to obtain a path from w' to a^n.
Finally, since δ_1 is symmetric this gives a path from w to w' and vice-versa. □

In addition to show that $G = (\mathcal{L}_{\{f\}}, \delta_1)$ is strongly connected, this proof gives us $2n$ as an upper bound to the diameter of G.

Remark 4. *We could have tried to prove Result 1 by induction on $H(w, w')$ instead. But it is unclear to what extent such an induction would be feasible. Consider the following example with $\Sigma = \{A, U\}$:*

$$\mathcal{F} = \{AUA\}, u = AAAA, v = AUUA.$$

There is no way to replace one non-extremal A of u with U without getting an occurrence of AUA. Hence there is no path from u to v in G with decreasing Hamming distance, even though

u and v are connected according to Result 1. The same idea gives counter-examples of arbitrary Hamming distance:

$$\forall i \in \mathbb{N}^*, i \geq 2, \textit{with: } \mathcal{F} = \{AUA\}, u_i = A^{i+2}, v_i = AU^iA,$$

$$\textit{then: } H(u_i, v_i) = i.$$

These examples heavily rely on the fact that $|\Sigma| = 2$. There might be a way to get around this issue when $|\Sigma| \geq 3$ and find paths with non-increasing Hamming distance, but this would have to be looked at.

3.2 With two or more motifs

The idea from the proof of Result 1 could be used again to treat the cases when there is an available letter to do the same trick.

Result 5. *Let F be the set of forbidden motifs.*

- *If there exists $a \in \Sigma$ such that: $\forall f \in \mathcal{F}, f[1] \neq a$, then $G = (\mathcal{L}_{\mathcal{F},n}, \delta_1)$ is strongly connected.*
- *Same result if there exists $a \in \Sigma$ such that: $\forall f \in \mathcal{F}, f[|f|] \neq a$.*

This tells us that we need at least $|\Sigma|$ forbidden motifs to obtain a disconnected graph with δ_1. Indeed if there are less than $|\Sigma|$ motifs, then we know that at least one letter is not the first letter of any forbidden motif.

Corollary 6. *If $|\mathcal{F}| < |\Sigma|$, then $G_{\mathcal{F},n,1}$ is strongly connected.*

An example with $|\Sigma|$ words that gives a disconnected graph with δ_1 is the following:

$$\textit{with} : \Sigma = \{a_1, a_2, \ldots, a_k\}, \textit{let} : \mathcal{F} = \{a_1a_2, a_2a_1, a_3, \ldots, a_k\}.$$

Then the only two allowed words are a_1^n and a_2^n, and there is no way to go from one word to the other.

With Σ or more forbidden motifs it could be hard to predict the connectivity of $G_{\mathcal{F},n,k}$. There could still be alternative solutions like the following one:

Proposition 7. *If P_1 and P_2 are in $\mathcal{L}_{\mathcal{F},|P_1|}$ and disconnected in $G_{\mathcal{F},|P_1|,k}$ (for some $k \in [1..|P_1|]$), then: $\forall (S_1, S_2)$ couple of words of the same length: if P_1S_1 and P_2S_2 are in $\mathcal{L}_{\mathcal{F},|P_1|}$, then P_1S_1 and P_2S_2 are disconnected in $G_{\mathcal{F},|P_1S_1|,k}$.*

Proof. See *Appendices*. □

In other words, if we find two words P_1, P_2 that are disconnected in $G_{\mathcal{F},i,k}$ for some $i \in \mathbb{N}^*$, then any couple of words P_1S_1, P_2S_2 in $G_{\mathcal{F},j,k}$ (for some $j > i$) that have them as their prefixes are disconnected as well. Hence non-connectivity could be established for smaller words and then later extended to words of length n.
The main issue with using the previous proposition is to find prefixes that can be extended with long enough suffixes, which is hard to predict in the forbidden motifs framework. More generally, it would be interesting to retrieve partial connectivity information without having to look at the entire graph $G_{\mathcal{F},n,k}$. This is where De Bruijn graphs come into play.

4 De Bruijn graphs

In this section, we use variants of the original De Bruijn graph first introduced by [DB46] and [Goo46], and use them to establish connectivity results on $G_{\mathcal{F},n,k}$.

4.1 Definition and basic properties

Definition 8. *Given $\mathcal{F}$, we define $\mathcal{DB}_{\mathcal{F}}$ the De Bruijn graph of $\mathcal{F}$ the following way:*

- *Vertices: $\mathcal{L}_{\mathcal{F},m(\mathcal{F})}$ the allowed substrings of length $m(\mathcal{F})$.*
- *Edges: if $u \in \Sigma^{m(\mathcal{F})-1}$, if $a, b \in \Sigma$, then: there is an edge from au to ub iff au and ub are both in $\mathcal{L}_{\mathcal{F},m(\mathcal{F})}$.*

$\mathcal{DB}_{\mathcal{F}}$ can be constructed in time $\theta((\sum_{f\in\mathcal{F}}|f|) + (m(\mathcal{F}) + |\Sigma|) * |\Sigma|^{m(\mathcal{F})})$:

- construct the Aho-Corasick automaton $\mathcal{AC}_{\mathcal{F}}$ of $\mathcal{F}$ (time $\theta(\sum_{f\in\mathcal{F}}|f|)$),
- enumerate all words of length $m(\mathcal{F})$ and check which words are valid ($|\Sigma|^{m(\mathcal{F})}$ words, time $\theta(m(\mathcal{F}))$ for each word with $\mathcal{AC}_{\mathcal{F}}$),
- list the neighbors for each valid word (time $\mathcal{O}(|\Sigma|)$ from the landing state in $\mathcal{AC}_{\mathcal{F}}$).

By definition: $|V(\mathcal{DB}_{\mathcal{F}})| = \mathcal{O}(\Sigma^{m(\mathcal{F})})$, which is often much smaller than $|V(G_{\mathcal{F},n,k})| = \mathcal{O}(\Sigma^n)$. And yet all the elements in $G_{\mathcal{F},n,k}$ - i.e. the allowed words of length n - are encoded in $\mathcal{DB}_{\mathcal{F}}$ as paths.

Proposition 9. *Let $w = w_1 \dots w_n$ be a word of length n. Then:*

$w \in \mathcal{L}_{\mathcal{F},n}$ *iff* $w_1 \dots w_{m(F)} \to w_2 \dots w_{m(F)+1} \to \cdots \to w_{n-m(F)+1} \dots w_n$ *is a valid path in $\mathcal{DB}_{\mathcal{F}}$.*

This makes it easy to find how a prefix can be extended: start from the endpoint of the corresponding path in $\mathcal{DB}_{\mathcal{F}}$ and see how this path can be prolonged.

Two variants of $\mathcal{DB}_{\mathcal{F}}$ will be considered.

Definition 10. *We define:*

- *$\mathcal{DB}_{\mathcal{F},n}$ the graph obtained by removing all the connected components in $\mathcal{DB}_{\mathcal{F}}$ that do not encode any word of length n.*
- *$\mathcal{DB}_{\mathcal{F},\infty}$ the graph obtained by removing all the acyclic connected components in $\mathcal{DB}_{\mathcal{F}}$.*

The idea behind $\mathcal{DB}_{\mathcal{F},n}$ is to only keep the meaningful part in $\mathcal{DB}_{\mathcal{F}}$ that generates the allowed words of length n. By *Proposition* 9 this is the same as removing all the connected components that have no path of length $\geq n - m(\mathcal{F})$.

Remark 11. *For any $n \geq m(\mathcal{F})$, $\mathcal{DB}_{\mathcal{F}}$ and $\mathcal{DB}_{\mathcal{F},n}$ have exactly the same paths of length $n - m(\mathcal{F})$.*

Constructing $\mathcal{DB}_{\mathcal{F},n}$ from $\mathcal{DB}_{\mathcal{F}}$ requires a BFS with exactly n steps, which takes time $\mathcal{O}(n * |\Sigma|^{m(\mathcal{F})})$.

The idea behind $\mathcal{DB}_{\mathcal{F},\infty}$ is to only keep the asymptotically interesting part of $\mathcal{DB}_{\mathcal{F}}$.

Remark 12. *There are arbitrarily long allowed words with $\mathcal{F}$ iff there is a cycle in $\mathcal{DB}_{\mathcal{F}}$, iff $\mathcal{DB}_{\mathcal{F},\infty}$ is not empty.*

Constructing $\mathcal{DB}_{\mathcal{F},\infty}$ from $\mathcal{DB}_{\mathcal{F}}$ only requires a standard BFS. Since the number of edges in $\mathcal{DB}_{\mathcal{F}}$ is linear in the number of nodes, this BFS takes time $\mathcal{O}(|\Sigma|^{m(\mathcal{F})})$.

There is a clear connection between these two variants. With n growing more and more, all the acyclic components in $\mathcal{DB}_{\mathcal{F},n}$ would eventually be removed, until we get $\mathcal{DB}_{\mathcal{F},\infty}$.

Result 13. *There exists $N \in \mathbb{N}^*$ such that: $\forall n \geq N, \mathcal{DB}_{\mathcal{F},n} = \mathcal{DB}_{\mathcal{F},\infty}$.*
Furthermore: $N \leq \Sigma^{m(\mathcal{F})-1} + m(\mathcal{F}) - 1$.

Proof. See *Appendices* □

Remark 14. *The bound for N is tight. The limit cases of paths of length $\Sigma^{m(\mathcal{F})-1} + m(\mathcal{F}) - 2$ are obtained with the De Bruijn sequences of order $m(\mathcal{F}) - 1$.*

4.2 Properties on paths

Result 15. *Let $u_1 \to \cdots \to u_i$ be a path in $\mathcal{DB}_{\mathcal{F}}$ ($i \geq 3$).*
If u_i is a neighbor of u_1, then $(u_2, \cdots, u_{i-1})$ is a cycle in $\mathcal{DB}_{\mathcal{F}}$.

Proof. Since u_i is a a neighbor of u_1 in $\mathcal{DB}_{\mathcal{F}}$, we can write: $u_1 = av$ and $u_i = vb$ for some $v \in \Sigma^{m(\mathcal{F})-1}$ and $a, b \in \Sigma$.
But then we know as well that: $u_2 = vc$ and $u_{i-1} = dv$ for some $a, b \in \Sigma$.
Hence u_2 is a neighbor of u_{i-1} and $(u_2, \cdots, u_{i-1})$ forms a cycle in $\mathcal{DB}_{\mathcal{F}}$. □

In other words: if there is a shortcut to a path in $\mathcal{DB}_{\mathcal{F}}$, then the intermediate elements form a cycle. This result will not be reused in the following connectivity methods, but it is still an interesting property in itself specific to De Bruijn graphs. It especially applies to the acyclic components of $\mathcal{DB}_{\mathcal{F}}$.
Despite its apparent simplicity, this next lemma will be key to some of the upcoming connectivity results.

Lemma 16. *If we follow the same sequence of letters $a_1, a_2, \cdots, a_j$ from two distinct sequences u, v in $\mathcal{DB}_{\mathcal{F}}$, if $j \geq m(\mathcal{F})$, then the two subsequent paths have merged at some index $i \leq m(\mathcal{F})$.*

Proof. After $m(\mathcal{F})$ steps the resulting word is $a_1 \dots a_{m(\mathcal{F})}$ in both paths, so the paths merged either at index $m(\mathcal{F})$ or at a smaller index. □

4.3 Applications to connectivity

In this subsection, we link properties of $\mathcal{DB}_{\mathcal{F}}$ to the connectivity of $G_{\mathcal{F},n,k}$.

In this proposition, we use a property of $\mathcal{DB}_{\mathcal{F}}$ to guarantee a natural way to extend a connectivity result on smaller values of n.

Proposition 17. *If $\mathcal{DB}_{\mathcal{F},n}$ has no dead ends, then:*
$G_{\mathcal{F},n,k}$ is connected $\Rightarrow \forall i \in \mathbb{N}^*, G_{\mathcal{F},n+i,k+i}$ is connected.

Proof. With no dead ends, any allowed word of length n is the prefix of an infinite allowed word. Thus any path $u_1 \to \cdots$ in $G_{\mathcal{F},n,k}$ can be converted into a path $u_1 s_1 \to \cdots$ in $G_{\mathcal{F},n+i,k+i}$ with the s_j some suffixes of length i. Indeed if $H(u_j, u_{j+1}) \leq k$ then necessarily $H(u_j s_j, u_{j+1} s_{j+1}) \leq k + i$. □

The issue we had in the previous section about whether a prefix can be extended is treated by a dead end check on $\mathcal{DB}_{\mathcal{F}}$ which takes time $\mathcal{O}(|\Sigma|^{m(\mathcal{F})})$ - ensure that no adjacency list is empty.
This proposition is by no means perfect. The requirement on $\mathcal{DB}_{\mathcal{F}}$ is stronger than we actually need. We only consider paths in $G_{\mathcal{F},n+i,k+i}$ that are extensions from prefixes in $G_{\mathcal{F},n,k}$ while there are certainly more subtle paths that connect words in $G_{\mathcal{F},n+i,k+i}$. The implication is also not satisfactory since it induces an significant increase in the nummber of edges - it is multiplied by a factor $|\Sigma|^i$. Nevertheless it is a starting point, as well as a demonstration of how de Bruijn graphs could help to solve our main problem.

The following connectivity result takes another direction. We infer a sufficient condition of non connectivity from seemingly insignificant *Lemma* 16.

Lemma 18. *Let $u, v \in \mathcal{L}_{\mathcal{F},m(\mathcal{F})}$ be two elements in different connected components of $\mathcal{DB}_{\mathcal{F}}$.*
Then: $\forall l, s \in \mathbb{N}^*, \forall s_1, s_2 \in \Sigma^s$,
$(s \geq l * m(\mathcal{F})$ and $us_1, vs_2 \in \mathcal{L}_{\mathcal{F},m(\mathcal{F})+s}) \Rightarrow H(us_1, vs_2) \geq l + H(u, v)$.

Proof. If the first $m(\mathcal{F})$ letters of s_1 and s_2 are the same, then starting from u and v in $\mathcal{DB}_{\mathcal{F}}$, by lemma *Lemma* 16, we would reach the same element after $m(\mathcal{F})$ steps and u and v would be connected. Contradiction.
Hence at least one the first $m(\mathcal{F})$ letters must be different between s_1 and s_2.
By repeating this l times we get the result. □

Theorem 19. *$\forall n \geq (k+1) * m(\mathcal{F}), \mathcal{DB}_{\mathcal{F},n}$ is disconnected $\Rightarrow G_{\mathcal{F},n,k}$ is disconnected.*

Proof. Direct implication of *Lemma* 18. □

This theorem is also valid with $\mathcal{DB}_{\mathcal{F},\infty}$ (we only potentially remove some of the components in $\mathcal{DB}_{\mathcal{F},n}$ while leaving unchanged the remaining ones). The condition on n and $m(\mathcal{F})$ is likely to be met in practice since RNA sequences are usually much longer than the desired forbidden motifs and the lower values of k are preferred for better time complexity in the local search phase.
We then have two partial non-connectivity tests of $G_{\mathcal{F},n,k}$ by testing the connectivity of either $\mathcal{DB}_{\mathcal{F},n}$ or $\mathcal{DB}_{\mathcal{F},\infty}$ (time $\mathcal{O}(|\Sigma|^{m(\mathcal{F})})$). Taking into account the construction time of the graphs, the time complexity is:

- with $\mathcal{DB}_{\mathcal{F},n}$: $\mathcal{O}(|\Sigma| * (\sum_{f \in \mathcal{F}} |f|) + (n + |\Sigma|) * |\Sigma|^{m(\mathcal{F})})$
- with $\mathcal{DB}_{\mathcal{F},\infty}$: $\mathcal{O}(|\Sigma| * (\sum_{f \in \mathcal{F}} |f|) + (m(\mathcal{F}) + |\Sigma|) * |\Sigma|^{m(\mathcal{F})})$

Choosing between the two is a matter of compromise between efficiency and time complexity. Using $\mathcal{DB}_{\mathcal{F},n}$ is expected to perform slightly better at the price of a worse time complexity.

Even though the exponent is lower, the time complexity is still exponential in the length of the motifs. In the next section, we will adapt these results and reduce significantly the time complexity with help from the Aho-Corasick automaton.

5 The Aho-Corasick automaton

In this section, we use variants of the Aho-Corasick automaton originally presented in [AC75] and establish their relation to $\mathcal{DB}_{\mathcal{F}}$ and $G_{\mathcal{F},n,k}$.

5.1 Definition

Definition 20. *We define the Aho-Corasick automaton $\mathcal{AC}_{\mathcal{F}}$ as follows:*

- $Q = \{u \text{ strict prefix of some } f \in \mathcal{F}\}$
- $I = \{\epsilon\}$
- $T = Q$
- $\Delta = \Delta_f \uplus \Delta_b$, *with:*
  - Δ_f *the forward edges:* $\{(u, a, ua), a \in \Sigma \text{ and } u, ua \in Q\}$ *(i.e. the trie of the elements in $\mathcal{F}$)*
  - Δ_b *the backward edges:* $\{(u, a, v) \mid ua \notin Q \text{ and } v \text{ is the longest suffix of } ua \text{ which is an element of } Q\}$

With this definition of $\mathcal{AC}_{\mathcal{F}}$: a word w is accepted iff no $f \in \mathcal{F}$ is a substring of w. This is actually the complement of the usual Aho-Corasick automaton since it is widely used for pattern matching.
Since [AC75] it has been known that $\mathcal{AC}_{\mathcal{F}}$ can be built in time $\mathcal{O}(|\Sigma| * (\sum_{f\in\mathcal{F}} |f|))$. It has also been known that the number of edges is at most $|\Sigma|$ times the number of states, which makes graph searches particularly efficient when using adjacency lists as the implementation.
It is worth noting that $\mathcal{AC}_{\mathcal{F}}$ is not necessarily minimal. We will still stick to this automaton for its time complexity insurances. Also by definition the Aho-Corasick automaton has the following property, which will be handy in the next subsection.

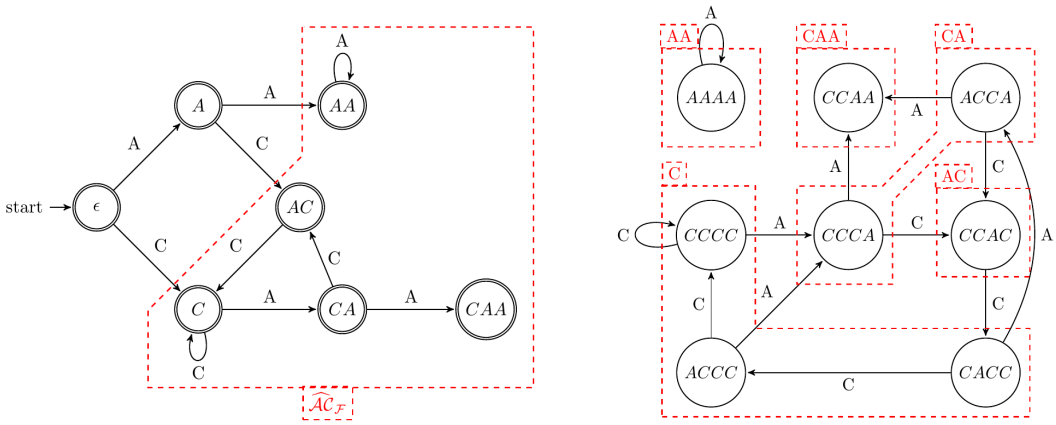
Proposition 21. *When reading a word w through $\mathcal{AC}_{\mathcal{F}}$, the label of the landing state is the longest suffix of w that is a prefix of some $f \in \mathcal{F}$.*

5.2 Relation to the De Bruijn graph

By ignoring the labels on the edges, $\mathcal{AC}_{\mathcal{F}}$ can be seen as a graph and its properties can be compared to the ones of $\mathcal{DB}_{\mathcal{F}}$. Only a specific part of $\mathcal{AC}_{\mathcal{F}}$ will be of interest.

Definition 22. *We define the graph $\widehat{\mathcal{AC}}_{\mathcal{F}}$ from $\mathcal{AC}_{\mathcal{F}}$ by removing the states that can only be reached in less than $m(\mathcal{F})$ steps.*

This can be done with a first BFS for exactly $m(\mathcal{F})$ steps followed by a full BFS from the nodes reached at the end of the first BFS. This takes time:
$\mathcal{O}(|\Sigma| * m(\mathcal{F}) * (\sum_{f\in\mathcal{F}} |f|) + |\Sigma| * (\sum_{f\in\mathcal{F}} |f|)) = \mathcal{O}(|\Sigma| * m(\mathcal{F}) * (\sum_{f\in\mathcal{F}} |f|))$.



(a) $\mathcal{AC}_{\mathcal{F}}$ (b) $\mathcal{DB}_{\mathcal{F}}$ and the equivalence classes

Figure 1: Case $\mathcal{F} = \{ACA, AAC, CAAA\}$ ($\Sigma = \{A, C\}$)

By doing this we only keep the part of $\mathcal{AC}_{\mathcal{F}}$ about the allowed words of length at least $m(\mathcal{F})$. Our motivation is to establish a strong connection between $\widehat{\mathcal{AC}}_{\mathcal{F}}$ and $\mathcal{DB}_{\mathcal{F}}$. First we introduce the following notion.

Definition 23. *If $u \in V(\mathcal{DB}_{\mathcal{F}})$, $s(u) \stackrel{\text{def}}{=}$ the longest suffix s' of u such that $s' \in V(\widehat{\mathcal{AC}}_{\mathcal{F}})$.*

Remark 24. *If $u \in V(\mathcal{DB}_{\mathcal{F}})$, since u is of length $m(\mathcal{F}) \geq m(\mathcal{F})$, its path in $\mathcal{AC}_{\mathcal{F}}$ lands on a state s' in $V(\widehat{\mathcal{AC}}_{\mathcal{F}})$, which is also a suffix of u by Proposition 21. Hence $s(u)$ (i.e. the longest suffix s'' of u such that $s'' \in V(\widehat{\mathcal{AC}}_{\mathcal{F}})$) exists.*

The idea is to group together the nodes of $\mathcal{DB}_{\mathcal{F}}$ according to the following equivalence relation:

$$\forall u, v \in V(\mathcal{DB}_{\mathcal{F}}), u \equiv v \Leftrightarrow s(u) = s(v)$$

It turns out that by merging the nodes in $\mathcal{DB}_{\mathcal{F}}$ of the same equivalence class - i.e. which have the same longest suffix that is a prefix of some $f \in \mathcal{F}$ - and by merging the edges that link the same equivalence classes, we exactly get $\widehat{\mathcal{AC}}_{\mathcal{F}}$! See *Figure* 1 and *Figure* 2 for some examples.
Formally, we build $S(\mathcal{DB}_{\mathcal{F}})$ a graph obtained by merging the nodes of $\mathcal{DB}_{\mathcal{F}}$, then we show that it is actually $\widehat{\mathcal{AC}}_{\mathcal{F}}$.

Definition 25. *We define $S(\mathcal{DB}_{\mathcal{F}})$:*

- $V \stackrel{\text{def}}{=} \{p \in V(\widehat{\mathcal{AC}}_{\mathcal{F}}) \mid \exists u \in V(\mathcal{DB}_{\mathcal{F}}), s(u) = p\}$
- $E \stackrel{\text{def}}{=} \{(p, p') \in V(\widehat{\mathcal{AC}}_{\mathcal{F}})^2 \mid \forall u \in V(\mathcal{DB}_{\mathcal{F}})$ *s.t.* $s(u) = p, \exists u' \in V(\mathcal{DB}_{\mathcal{F}})$ *s.t.* $s(u') = p'$ *and* $(u, u') \in E(\mathcal{DB}_{\mathcal{F}})\}$.

Proposition 26. $S(\mathcal{DB}_{\mathcal{F}}) = \widehat{\mathcal{AC}}_{\mathcal{F}}$

Remark 27. *Notice that the definition of E in $S(\mathcal{DB}_{\mathcal{F}})$ is stronger than what was needed for establishing Proposition 26. We require that* ***all*** *elements u of $V(\mathcal{DB}_{\mathcal{F}})$ in the same equivalence class have an edge corresponding to the merged one in $\widehat{\mathcal{AC}}_{\mathcal{F}}$, not that only at least one of them has an edge. This further shows how strong the connection is between both objects. This key detail will allow us to reuse the connectivity results we got from $\mathcal{DB}_{\mathcal{F}}$, as it will be shown in the following subsection.*

5.3 Speeding up the connectivity tests

In this subsection, we conceive faster variants of the connectivity results in subsection 4.3.

Valid words of length $\geq m(\mathcal{F})$ are encoded as paths in $\widehat{\mathcal{AC}}_{\mathcal{F}}$ in the same fashion as in $\mathcal{DB}_{\mathcal{F}}$. Thus, as we did with $\mathcal{DB}_{\mathcal{F}}$ in subsection 4.1, we define the following variants of $\widehat{\mathcal{AC}}_{\mathcal{F}}$.

Definition 28. *We define:*

- $\widehat{\mathcal{AC}}_{\mathcal{F},n}$ *the graph obtained by removing all the connected components in* $\widehat{\mathcal{AC}}_{\mathcal{F}}$ *that do not encode any word of length* n*.*
- $\widehat{\mathcal{AC}}_{\mathcal{F},\infty}$ *the graph obtained by removing all the acyclic connected components in* $\widehat{\mathcal{AC}}_{\mathcal{F}}$*.*

$\widehat{\mathcal{AC}}_{\mathcal{F},n}$ and $\widehat{\mathcal{AC}}_{\mathcal{F},\infty}$ can be built in the same way we proceeded for the variants of $\mathcal{DB}_{\mathcal{F}}$, taking respectively time $\mathcal{O}(|\Sigma| * n * (\sum_{f\in\mathcal{F}}|f|))$ and $\mathcal{O}(|\Sigma| * (\sum_{f\in\mathcal{F}}|f|))$.

Because of *Remark* 27, we know that no adjacency list is empty in $\widehat{\mathcal{AC}}_{\mathcal{F},n}$ iff no adjacency list is empty in $\mathcal{DB}_{\mathcal{F},n}$. Hence there is no dead end in $\widehat{\mathcal{AC}}_{\mathcal{F},n}$ iff there is no dead end in $\mathcal{DB}_{\mathcal{F},n}$. We have then the following variant of *Proposition* 17.

Corollary 29. *If* $\widehat{\mathcal{AC}}_{\mathcal{F},n}$ *has no dead ends, then:*
$G_{\mathcal{F},n,k}$ *is connected* $\Rightarrow \forall i \in \mathbb{N}^*, G_{\mathcal{F},n+i,k+i}$ *is connected.*

Again this corollary is far from being satisfactory for the reasons stated in subsection 4.3 about *Proposition* 17. This still a significant speedup - $\mathcal{O}(n * (\sum_{f\in\mathcal{F}}|f|))$ instead of $\mathcal{O}((\sum_{f\in\mathcal{F}}|f|) + n * |\Sigma|^{m(\mathcal{F})})$ - for the same implication.
There is no denying that *Corollary* 29 could have been directly proven without knowing anything about $\mathcal{DB}_{\mathcal{F},n}$. This is still an example about how properties on $\mathcal{DB}_{\mathcal{F}}$ can be transferred to $\widehat{\mathcal{AC}}_{\mathcal{F}}$ with *Proposition* 26.

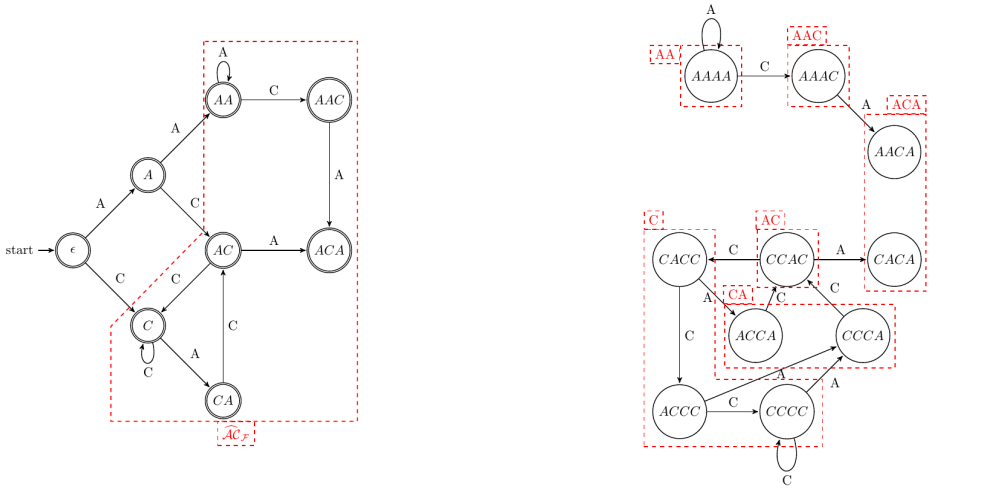
With *Proposition* 26 we can reuse the argument that proved *Lemma* 18 and *Theorem* 19.

Corollary 30. *Let* $u, v \in \mathcal{L}_{\mathcal{F},m(\mathcal{F})}$ *be two elements in different connected components of* $\widehat{\mathcal{AC}}_{\mathcal{F}}$*.*
Then: $\forall l, s \in \mathbb{N}^*, \forall s_1, s_2 \in \Sigma^s$,
$(s \geq l * m(\mathcal{F})$ *and* $us_1, vs_2 \in \mathcal{L}_{\mathcal{F},m(\mathcal{F})+s}) \Rightarrow H(us_1, vs_2) \geq l + H(u, v)$.

Proof. Proposition 26 indicates that every element in $\widehat{\mathcal{AC}}_{\mathcal{F}}$ has at least one corresponding element in $\mathcal{DB}_{\mathcal{F}}$. Let $\overline{u}, \overline{v} \in V(\mathcal{DB}_{\mathcal{F}})$ be such elements for u and v.
By contradiction: if we could follow the same sequence of letters $a_1, a_2, \cdots, a_{m(\mathcal{F})}$ from u and v in $\widehat{\mathcal{AC}}_{\mathcal{F}}$, then by *Remark* 27 we can follow this sequence of letters from $\overline{u}$ and $\overline{v}$ in $\mathcal{DB}_{\mathcal{F}}$. By *Lemma* 16 both paths merge at some point and both $\overline{u}$ and $\overline{v}$ are connected to $a_1 \dots a_{m(\mathcal{F})}$. However by *Remark* 27 there are also paths from u, v to $s(a_1 \dots a_{m(\mathcal{F})})$, which would mean that u and v are connected in $\widehat{\mathcal{AC}}_{\mathcal{F}}$. Contradiction.
Hence at least one the first $m(\mathcal{F})$ letters must be different between s_1 and s_2.
By repeating this l times we get the result. □

Corollary 31. $\forall n \geq (k+1) * m(\mathcal{F}), \widehat{\mathcal{AC}}_{\mathcal{F},n}$ *is disconnected* $\Rightarrow G_{\mathcal{F},n,k}$ *is disconnected.*

Proof. Direct implication of *Corollary* 30. □



(a) $\mathcal{AC}_{\mathcal{F}}$

(b) $\mathcal{DB}_{\mathcal{F}}$ and the equivalence classes

Figure 2: Case $\mathcal{F} = \{AACC, ACAA, ACAC, CAA\}$

This theorem is also valid with $\widehat{\mathcal{AC}}_{\mathcal{F},\infty}$ (we only potentially remove some of the components in $\widehat{\mathcal{AC}}_{\mathcal{F},n}$ while leaving unchanged the remaining ones). Again the condition on n and $m(\mathcal{F})$ is likely to be met in practice for the reasons we presented in subsection 4.3.

Remark 32. *Corollary 31 could have been proven with the following result:*

$$\widehat{\mathcal{AC}}_{\mathcal{F},n} \text{ is disconnected} \implies \mathcal{DB}_{\mathcal{F},n} \text{ is disconnected.}$$

which is directly inferred by Proposition 26. It is worth noting that the reverse is not necessarily true, as the merged nodes from $\mathcal{DB}_{\mathcal{F},n}$ could come from different connected components. See Figure 2 for an example.

We then have an additional partial non-connectivity test of $G_{\mathcal{F},n,k}$ by testing the connectivity of $\widehat{\mathcal{AC}}_{\mathcal{F},n}$ in time $\mathcal{O}(|\Sigma| * n * (\sum_{f \in \mathcal{F}} |f|))$.
By *Remark* 32 these faster non-connectivity tests are expected to work less often than their counterparts with $\mathcal{DB}_{\mathcal{F}}$. This will be quantified in the following section.

6 Experimental results

The four variants of partial connectivity tests that were described in subsections 4.3 and 5.3 have been implemented in Python. the code is available here:
`https://github.com/mohoc/M2-misc/tree/master/code`
Graphs were implemented as lists of lists, which is what was expected to work best on graphs with a linear number of edges like $\mathcal{DB}_{\mathcal{F}}$ and $\widehat{\mathcal{AC}}_{\mathcal{F}}$. Two libraries were particularly useful in this implementation:

- *pyahocorasick* for building the Aho-Corasick automaton efficiently:
  `https://pypi.org/project/pyahocorasick/`
- *tryalgo* for efficient implementations of the connectivity algorithms:
  `https://tryalgo.org/`

Since the correctness of the tests were proven with *Theorem* 19 and *Corollary* 31, we only need to look at recall - i.e. the proportion of cases, among those where $G_{\mathcal{F},n,1}$ is disconnected, where the tests detect it. The sets $\mathcal{F}$ were chosen uniformly at random. Several sets of parameters were tested, with thousands of examples every time. The results are given in *Figure* 3.
As expected:

- $\mathcal{DB}_{\mathcal{F},n}$ always performs better than $\mathcal{DB}_{\mathcal{F},\infty}$
- $\mathcal{DB}_{\mathcal{F},n}$ always performs better than $\widehat{\mathcal{AC}}_{\mathcal{F},n}$
- $\widehat{\mathcal{AC}}_{\mathcal{F},n}$ always performs better than $\widehat{\mathcal{AC}}_{\mathcal{F},\infty}$

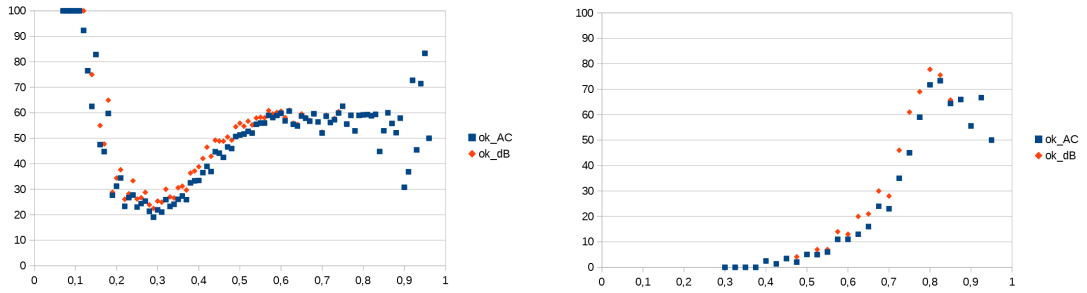
Whatever the variant of the test and the set of parameters, recall is between 25% and 55%, which is a significant proportion of the non-connectivity cases. The difference between $\mathcal{DB}_{\mathcal{F},n}$ and $\widehat{\mathcal{AC}}_{\mathcal{F},n}$ is always minimal - below 5%. The same holds between $\mathcal{DB}_{\mathcal{F},\infty}$ and $\widehat{\mathcal{AC}}_{\mathcal{F},\infty}$. Sometimes the difference between an "n" variant and an "∞" variant is below 5%, sometimes it is much more pronounced. This might depend on the size of the alphabet, as in general the gap is greater with $|\Sigma| \geq 3$ than with $|\Sigma| = 2$.

Remark 33. *Due to the time complexity constraints, only the 1-Hamming neighborhood has been tested. The main reason is the condition $n \geq (k+1) * m(\mathcal{F})$ that require the tests. With $m(\mathcal{F}) \geq 4$ and $k \geq 2$, n must be at least equal to 12, which generates graphs with millions of nodes with a 4-letter alphabet and few forbidden motifs. Not only that, but the number of edges grows exponentially with k. It would then take too much time with our Python implementation to treat the thousands of cases that are needed for accurate enough recall values.*

In *Figure* 4 recall results of $\mathcal{DB}_{\mathcal{F},n}$ and $\widehat{\mathcal{AC}}_{\mathcal{F},n}$ were represented in terms of the proportion of motifs that are forbidden. The results are quite different depending on the size of the alphabet. With $|\Sigma| = 2$ recall starts strong, drops down around 0.3 then stabilizes at about 60%; with $|\Sigma| = 4$, recall is negligible until 0.5 then gradually increases. These behaviors can be explained in part by *Figure* 5, as the low recall values correlate with significant

$\lvert\Sigma\rvert$	$m(\mathcal{F})$	n	sample	$\#G_{\mathcal{F},n,1}$ disconnected	$\mathcal{DB}_{\mathcal{F},n}$	$\mathcal{DB}_{\mathcal{F},\infty}$	$\widehat{\mathcal{AC}}_{\mathcal{F},n}$	$\widehat{\mathcal{AC}}_{\mathcal{F},\infty}$
2	5	10	100000	36630	49.5	45.0	47.1	43.2
2	5	11	100000	35893	48.2	46.3	46.2	44.2
3	5	10	10000	4395	53.9	34.8	49.2	32.3
4	3	6	25000	9447	37.6	29.4	34.3	27.5
4	3	7	10000	3728	37.9	33.1	35.7	30.7
4	4	8	4000	1904	54.3	31.2	50.1	27.7

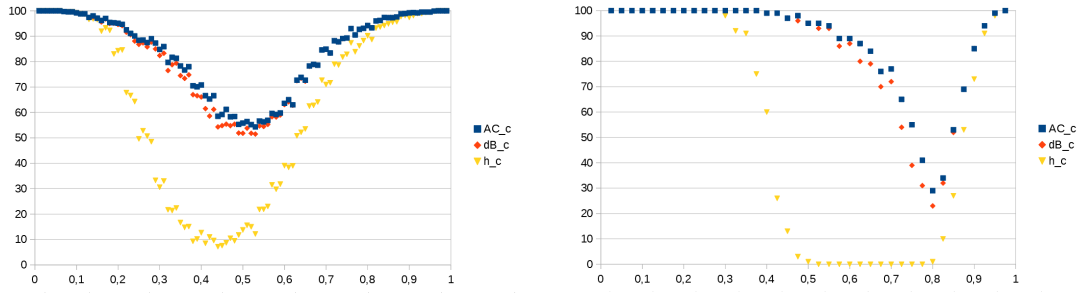
Figure 3: Recall of the non-connectivity tests on several sets of parameters



(a) $|\Sigma| = 2$, $m(\mathcal{F}) = 5$, $n = 10$ (b) $|\Sigma| = 4$, $m(\mathcal{F}) = 4$, $n = 8$

Figure 4: Recall with $\mathcal{DB}_{\mathcal{F},n}$ and $\widehat{\mathcal{AC}}_{\mathcal{F},n}$ in terms of the proportion of motifs that are forbidden. $\mathcal{DB}_{\mathcal{F},n}$ in blue, $\widehat{\mathcal{AC}}_{\mathcal{F},n}$ in orange.

average connectivity difference between $G_{\mathcal{F},n,1}$ and the rest. *Figure* 4 notably confirms the low performance difference between $\mathcal{DB}_{\mathcal{F},n}$ and $\widehat{\mathcal{AC}}_{\mathcal{F},n}$. This places $\widehat{\mathcal{AC}}_{\mathcal{F},n}$ as the most interesting variant in most cases.
Last but not least, *Figure* 5 exposes the limitations of our connectivity methods, i.e. predicting the connectivity of $G_{\mathcal{F},n,k}$ from the connectivity of a variant of either $\mathcal{DB}_{\mathcal{F}}$ or $\widehat{\mathcal{AC}}_{\mathcal{F}}$. $G_{\mathcal{F},n,k}$ is disconnected much more often than any of our graphs when far from the extremities. Treating this middle section would probably require a completely new framework.



(a) $|\Sigma| = 2$, $m(\mathcal{F}) = 5$, $n = 10$ (b) $|\Sigma| = 4$, $m(\mathcal{F}) = 4$, $n = 8$

Figure 5: Average connectivity of $G_{\mathcal{F},n,1}$, $\mathcal{DB}_{\mathcal{F},n}$ and $\widehat{\mathcal{AC}}_{\mathcal{F},n}$ in terms of the proportion of motifs that are forbidden. $\mathcal{DB}_{\mathcal{F},n}$ in blue, $\widehat{\mathcal{AC}}_{\mathcal{F},n}$ in orange, $G_{\mathcal{F},n,1}$ in yellow.

7 Conclusion

We were able to give several sufficient conditions of connectivity in terms of properties of the set $\mathcal{F}$ of forbidden motifs. It has notably been proven that the search space is always connected when dealing with less than $|\Sigma|$ forbidden motifs, and we gave a counterexample with exactly $|\Sigma|$ motifs. We defined several variants of the De Bruijn graph and we used them to conceive partial non-connectivity tests (exponential time with a lower exponent). We also derived several graphs from the Aho-Corasick automaton with the same goal. We proved a strong connection between the De Bruijn variants and the Aho-Corasick ones, which induced additional partial non-connectivity tests that were slightly less effective but much faster (linear time in $\sum_{f\in\mathcal{F}} |f|$).
The partial non-connectivity tests were proven correct. They were implemented in Python and tested on a uniformly random selection of sets $\mathcal{F}$ on several values of the parameters Σ, n and $m(\mathcal{F})$. A significant proportion of the disconnected Hamming graph cases were treated by the tests every time. Finally the condition required for the non-connectivity tests to be usable - $n \geq (k+1) * m(\mathcal{F})$ with k the degree of the Hamming neighborhood - is likely to be met in practice.
The general question remains unsolved. There is a need for sufficient conditions of connectivity in the case of non-extremal proportions of forbidden motifs, i.e. where the search space is very likely to be disconnected.

Bibliography

[AC75] Alfred V. Aho and Margaret J. Corasick. Efficient string matching: An aid to bibliographic search. *Commun. ACM*, 18(6):333–340, June 1975.

[BRS17] Édouard Bonnet, Pawel Rzazewski, and Florian Sikora. Designing rna secondary structures is hard. In *RECOMB*, 2017.

[CRR+17] Alexander Churkin, Matan Drory Retwitzer, Vladimir Reinharz, Yann Ponty, Jérôme Waldispühl, and Danny Barash. Design of RNAs: comparing programs for inverse RNA folding. *Briefings in Bioinformatics*, 19(2):350–358, 01 2017.

[DB46] N. G. De Bruijn. A combinatorial problem. *Proc. Koninklijke Nederlandse Academie van Wetenschappen*, 49:758–764, 1946.

[Goo46] I. J. Good. Normal Recurring Decimals. *Journal of the London Mathematical Society*, s1-21(3):167–169, 07 1946.

[HFS+93] Ivo L. Hofacker, Walter Fontana, Peter F. Stadler, L. Sebastian Bonhoeffer, Manfred Tacker, and Peter Schuster. Fast Folding and Comparison of RNA Secondary Structures. Working Papers 93-07-044, Santa Fe Institute, July 1993.

[HTF+17] Stefan Hammer, Birgit Tschiatschek, Christoph Flamm, Ivo L Hofacker, and Sven Findeiß. RNAblueprint: flexible multiple target nucleic acid sequence design. *Bioinformatics*, 33(18):2850–2858, 04 2017.

[LGDP15] Vincent Le Gallic, Alain Denise, and Yann Ponty. Résultats algorithmiques pour le design d'ARN avec contraintes de séquence. In *SeqBio 2015*, pages 26–31, Orsay, France, November 2015.

[MPB03] Filippo Mignosi Antonio Restivoy Marinella Sciortino Marie-Pierre Béal, Maxime Crochemore. Computing forbidden words of regular languages. *Fundamenta Informaticae*, 56:121–135, 2003.

[MRS01] F. Mignosi, A. Restivo, and M. Sciortino. Forbidden factors and fragment assembly. *RAIRO - Theoretical Informatics and Applications*, 35(6):565–577, 2001.

[MRS02] F. Mignosi, A. Restivo, and M. Sciortino. Words and forbidden factors. *Theoretical Computer Science*, 273(1):99 – 117, 2002. WORDS.

[Pon14] Yann Ponty. Bio-algorithmique des ARN : petite promenade aux interfaces. In Eric Sopena, editor, *1024 - Bulletin de la société informatique de France*, volume 4, pages 23–53. SIF, October 2014.

[ZPV+13] Yu Zhou, Yann Ponty, Stéphane Vialette, Jérôme Waldispühl, Yi Zhang, and Alain Denise. Flexible RNA design under structure and sequence constraints using formal languages. In *ACM-BCB - ACM Conference on Bioinformatics, Computational Biology and Biomedical Informatics - 2013*, Bethesda, Washigton DC, United States, September 2013.

Appendices

Proof of Proposition 7

Proof. Let P_1 and P_2 be two words in $\mathcal{L}_{\mathcal{F},|P_1|}$ that are disconnected in $G_{\mathcal{F},|P_1|,k}$.
By contradiction: if there exist S_1, S_2 such that P_1S_1 and P_2S_2 are connected in $G_{\mathcal{F},|P_1S_1|,k}$, then there is a path $(P_1S_1 = u_0) \to u_1 \to \cdots \to u_i \to (u_{i+1} = P_2S_2)$ in $G_{\mathcal{F},|P_1S_1|,k}$.
We know then that: $\forall 0 \leq j \leq i, H(u_j, u_{j+1}) \leq k$.
For $0 \leq j \leq i+1$ let P'_j be the prefix of length $|P_1|$ in u_j.
Since we take the prefixes, we still have: $\forall 0 \leq j \leq i, H(P'_j, P'_{j+1}) \leq k$.
Hence $(P_1 = P'_0) \to P'_1 \to \cdots \to P'_i \to (P'_{i+1} = P_2)$ is a valid path in $G_{\mathcal{F},|P_1|,k}$.
Hence P_1 and P_2 would be connected in $G_{\mathcal{F},|P_1|,k}$.
Contradiction. □

Proof of Result 13

Proof. :

- The more n grows, the more acyclic connected components are removed until no one remains. Thus we eventually get $\mathcal{DB}_{\mathcal{F},\infty}$.
- An acyclic connected component is removed when n exceeds the max length of a path in the component. Thus a bound on the max length of a path in an acyclic component of $\mathcal{DB}_{\mathcal{F}}$ would give us a bound on N.

  If we have a path $u_1 \to \cdots$ in an acyclic connected component, then all the u_i must be distinct. If the length is $\geq \Sigma^{m(\mathcal{F})-1} - 1$, consider the prefixes of length $m(\mathcal{F}) - 1$ of the $\Sigma^{m(\mathcal{F})-1}$ first elements of the path.

  – If all the prefixes are distinct:

    then all possible words of length $m(\mathcal{F}) - 1$ appear as a prefix. Thus the suffix of $u_{\Sigma^{m(\mathcal{F})-1}}$ matches with the prefix of one the $\Sigma^{m(\mathcal{F})-1}$ first elements. This extra edge creates a cycle. Contradiction.

  – If one prefix appears more than once:

    then we have u_i and u_j with the same prefix of length $m(\mathcal{F}) - 1$, with $1 \leq i < j \leq \Sigma^{m(\mathcal{F})-1}$. By the definition of a De Bruijn graph, the suffix of length $m(\mathcal{F}) - 1$ of u_{j-1} matches with the prefix of u_j, and thus the prefix of u_i. We have an edge $u_{j-1} \to u_i$ with $i \leq j - 1$, which means that we have a cycle. Contradiction.

□

Proof of Proposition 26

Proof. Recall that we want to prove: $S(\mathcal{DB}_{\mathcal{F}}) = \widehat{\mathcal{AC}}_{\mathcal{F}}$

- V:
  (⊆) by definition of $S(\mathcal{DB}_{\mathcal{F}})$.
  (⊇) If $p \in V(\widehat{\mathcal{AC}}_{\mathcal{F}})$:

  Then there is w allowed word of length at least $m(\mathcal{F})$ that lands on p after going through $\mathcal{AC}_{\mathcal{F}}$.

  Let u be the suffix of w of length $m(\mathcal{F})$.

  Then u is an allowed word of length $m(\mathcal{F})$: $u \in V(\mathcal{DB}_{\mathcal{F}})$.

Plus by *Proposition* 21 : $s(u) = p$.

Hence: $p \in V(S(\mathcal{DB}_\mathcal{F}))$.

- E:

  ($\subseteq$) If $(p, p') \in E(S(\mathcal{DB}_\mathcal{F}))$: Since $p \in V(S(\mathcal{DB}_\mathcal{F}))$, there is $u \in V(\mathcal{DB}_\mathcal{F})$ such that $s(u) = p$.

  Then, by definition of $S(\mathcal{DB}_\mathcal{F})$: there is $u' \in V(\mathcal{DB}_\mathcal{F})$ such that $s(u') = p'$ and $(u, u') \in E(\mathcal{DB}_\mathcal{F})$.

  If we call $a_{u'}$ the last letter of u', then the word $ua_{u'}$ is allowed, and when reading it through $\mathcal{AC}_\mathcal{F}$, we go from state p - after reading u - to state p' when reading the last letter (u' is the suffix of length $m(\mathcal{F})$ of $ua_{u'}$ and $s(u') = p'$, hence we land on state p').

  Hence the transition between p and p' exists in $\mathcal{AC}_\mathcal{F}$, and we know from the beginning of the proof that both $V(S(\mathcal{DB}_\mathcal{F}))$ and $V(\widehat{\mathcal{AC}}_\mathcal{F})$ are equal.

  Hence $(p, p') \in E(\widehat{\mathcal{AC}}_\mathcal{F})$.

  ($\supseteq$) If $(p, p') \in E(\widehat{\mathcal{AC}}_\mathcal{F})$:

  Let $a_{p,p'}$ be the letter which labels this edge.

  Let u be any node in $V(\mathcal{DB}_\mathcal{F})$ such that $s(u) = p$ (since $p \in V(S(\mathcal{DB}_\mathcal{F}))$).

  Then when reading the word $ua_{p,p'}$, we go from state p to state p' when reading the last letter.

  Hence if we write $u = bv, b \in \Sigma$:

  * $va_{p,p'} \in V(\mathcal{DB}_\mathcal{F})$
  * $s(va_{p,p'}) = p'$
  * $(u, va_{p,p'}) \in E(\mathcal{DB}_\mathcal{F})$

  Hence $(p, p') \in S(\mathcal{DB}_\mathcal{F})$.

□