Algorithmics results for RNA design with sequence constraints

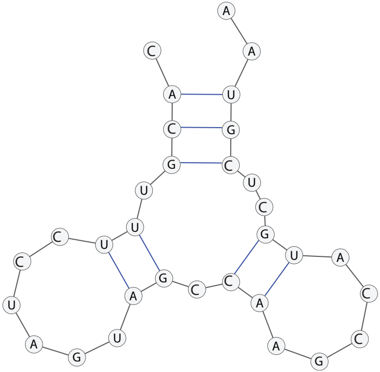
Vincent LE GALLIC
Alain DENISE
Yann PONTY

LRI

25/11/2015

RNA = sequence of
A, U, G, C

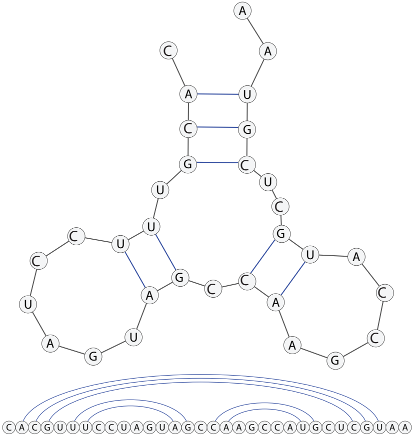
secondary structure =
hydrogen bonds
A – U, G – C
(Watson-Crick)
G – U (Wobble)



RNA structure

RNA = sequence of
A, U, G, C

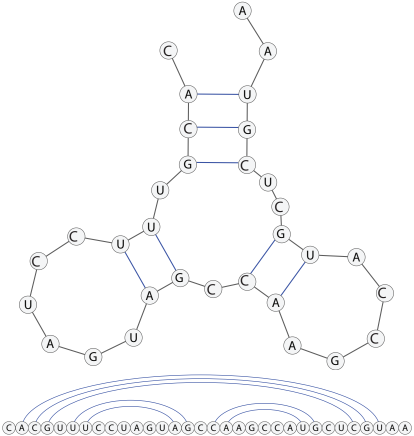
secondary structure =
hydrogen bonds
A – U, G – C
(Watson-Crick)
G – U (Wobble)



RNA structure

RNA = sequence of A, U, G, C

secondary structure = hydrogen bonds
A – U, G – C (Watson-Crick)
G – U (Wobble)



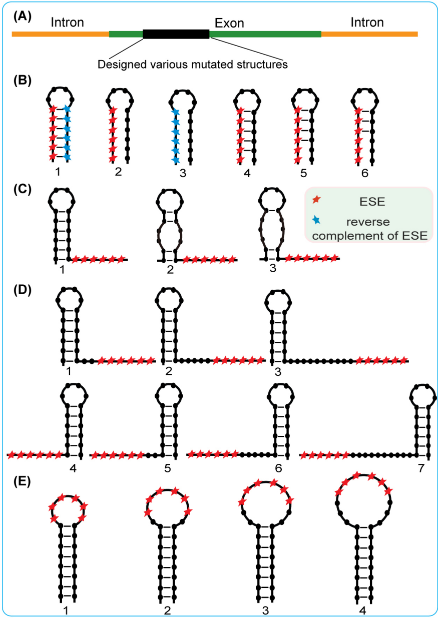
Energy model (Turner) : sequence w, folded in structure S $\rightarrow E(w, S)$

Hypothesis : w folds into S^* minimizing $E(w, S)$

Biological motivation : Exon Splicing Enhancer

In mRNA : binding site, target for splicing

Q: Influence of structural context?

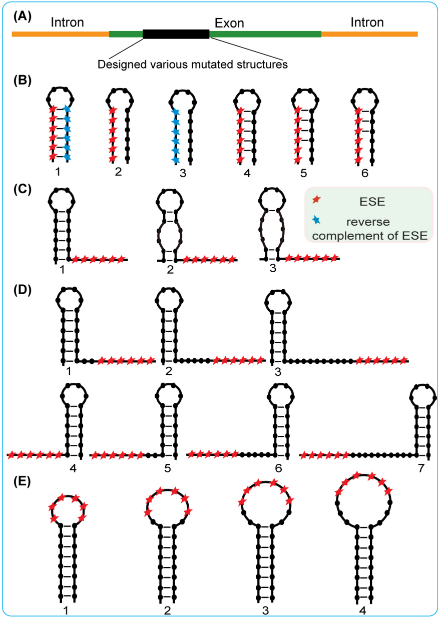


In mRNA : binding site, target for splicing

Q: Influence of structural context?

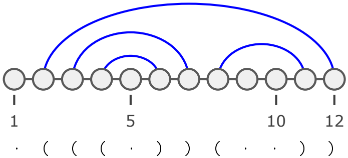
given structure S
$\text{ESE}_i = \text{AGAACU}$
required at specific position

$\text{ESE}_{j \neq i}$
forbidden



Formalisation : structure

one structure = a well-parenthesized word on $\{(,), \bullet, \}$



Formalisation : folding and design

$E(w, S)$ = free energy of w folded into S

Problem : Folding

Data: Sequence w
Result: $\mathrm{MFE}(w)$, set of structures minimizing $E(w, S)$

Problem : Inverse folding or Negative design

Data: Secondary structure S
Result: Sequence w such that $\mathrm{MFE}(s) = \{S\}$ if such a sequence exists, $\varnothing$ otherwise.

Unknown complexty (NP-hard?).

Formalisation : constraints

- Need to include constraints :
  - **Forbidden patterns.** Example : "ESE_j mustn't appear"
  - **Mandatory patterns.** Example : "CAAU must appear at least once"
  - **Positional constraints.** Example : "At position 27 only A or G allowed" or "positions 3-9 ESE_i"
- Other objective : finding sequences which folding is "near" to S

Formalisation : design with constraints

Problem : **Design with constraints**

Data:

- structure S ($|S| = n$)
- set of forbidden patterns $\mathcal{F}$
- set of mandatory patterns $\mathcal{M}$
- positional constraints $PC = \{(i, C_i)/i \in [\![1, n]\!], C_i \subseteq \Sigma\}$

Result: Sequence w such that

- w is compatible with S
- $\forall f \in \mathcal{F}, f \not\preceq s$
- $\forall m \in \mathcal{M}, m \preceq s$
- $\forall i \in [\![1, n]\!], w_i \in C_i$

or $\varnothing$ if no sequence fulfill these constraints

($u \preceq v$ means u factor of v)

Previous algorithms : classification

Stochastic search :

- `RNAInverse`

. . . + divide and conquer :

- `RNA-SSD`
- `INFO-RNA`
- `NUPack`

Random generation :

- `RNAensign`
- `IncaRNAtion`

Genetic algorithms :

- `FRNAKenstein`
- `RNAExInv`

Exact algorithms :

- `RNAiFold`
- `CO4`

Previous algorithms : classification

Stochastic search :

- `RNAInverse`

... + divide and conquer :

- `RNA-SSD`
- `INFO-RNA`
- `NUPack`

Random generation :

- `RNAensign`
- `IncaRNAtion`

Genetic algorithms :

- `FRNAKenstein`
- `RNAExInv`

Exact algorithms :

- `RNAiFold`
- `CO4`

Only one handles forbidden patterns : `NUPack`

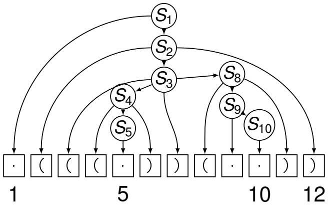
Zhou *et al.* approach

Zhou *et al.* (2013) introduce `CFGRNAD` :

- Define $\mathcal{L}$ = sequences compatible with S and the constraints
- Random generation in $\mathcal{L}$ (dynamic programming)
  - counting
  - generating
- Fold each candidate to check if it folds into S

Goal : optimize the counting and generating steps of this method

Zhou *et al.* : grammar



$S_1 \to \bullet S_2$

$S_2 \to (S_3)$

$S_3 \to (S_4)S_8$

$S_4 \to (S_5)$

$S_5 \to \bullet$

$S_8 \to (S_9)$

$S_9 \to \bullet S_{10}$

$S_{10} \to \bullet$

Zhou *et al.* : actual grammar

Full development of $\mathcal{G}_S$:
$\bullet \longrightarrow$ base $\mathrm{A}, \mathrm{U}, \mathrm{G}, \mathrm{C}$
$(\ldots) \longrightarrow$ pair $\{\mathrm{A}, \mathrm{U}\}, \{\mathrm{G}, \mathrm{C}\}, \{\mathrm{G}, \mathrm{U}\}$

$$
\begin{array}{lcl}
S_1 & \rightarrow & aS_2 \mid uS_2 \mid gS_2 \mid cS_2 \\
S_2 & \rightarrow & aS_3u \mid uS_3a \mid gS_3c \mid cS_3g \mid gS_3u \mid uS_3g \\
 & \ldots & \\
S_5 & \rightarrow & a \mid u \mid g \mid c \\
 & \ldots &
\end{array}
$$

Recognized language = all sequences compatible with S

Zhou *et al.* : actual grammar

Full development of $\mathcal{G}_S$:

$\bullet \longrightarrow$ base A, U, G, C

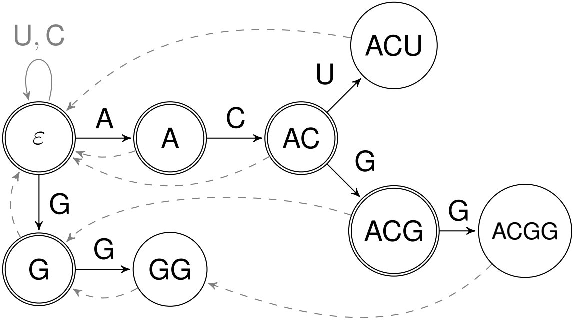
$(\ldots) \longrightarrow$ pair $\{$A, U$\}$, $\{$G, C$\}$, $\{$G, U$\}$

$$\begin{array}{lcl} S_1 & \rightarrow & aS_2 \mid uS_2 \mid gS_2 \mid cS_2 \\ S_2 & \rightarrow & aS_3u \mid \cancel{uS_3a} \mid gS_3c \mid cS_3g \mid gS_3u \mid \cancel{uS_3g} \\ & \ldots & \\ S_5 & \rightarrow & a \mid u \mid g \mid c \\ & \ldots & \end{array}$$

Recognized language = all sequences compatible with S

with $C_2 = \{\mathrm{A}, \mathrm{G}, \mathrm{C}\}$

Forbidden words : Aho-Corasick automaton



Automaton $\mathcal{A}_{\mathcal{F}}$ for $\mathcal{F} = \{\mathrm{ACU}, \mathrm{ACGG}, \mathrm{GG}\}$
pictured with *failure transitions*

Mandatory patterns

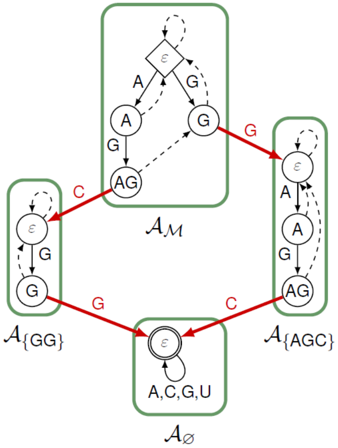


Figure 1: Automaton $\mathcal{A}_{\mathcal{M}}$ for $\mathcal{M} = \{\text{AGC}, \text{GG}\}$

Automaton forbidden + mandatory

$$\mathcal{A}_{\mathcal{F},\mathcal{M}} = \mathcal{A}_{\mathcal{F}} \times \mathcal{A}_{\mathcal{M}}$$

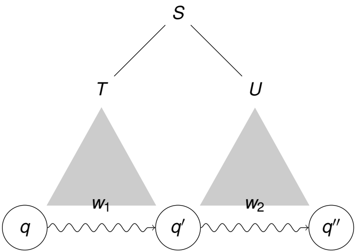
$$|Q_{\mathcal{F},\mathcal{M}}| \in O\left(2^{|\mathcal{M}|}\left(\sum_{f\in\mathcal{F}}|f| + \sum_{m\in\mathcal{M}}|m|\right)\right)$$

- structural + positional constraints $\leftrightarrow \mathcal{G}_S$
- forbidden and mandatory patterns $\leftrightarrow \mathcal{A}_{\mathcal{F},\mathcal{M}}$

Complexity

context-free $\cap$ regular $=$ context-free

$$S \to TU$$
$$\downarrow$$
$$S_{q,q''} \to T_{q,q'} U_{q',q''}$$

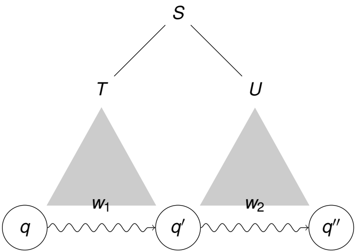


$$|\mathcal{G}_{\mathcal{G}\times\mathcal{A}}| = |\mathcal{G}| \cdot |\mathcal{A}|^3$$

Complexity

context-free $\cap$ regular $=$ context-free

$$S \rightarrow TU$$
$$\downarrow$$
$$S_{q,q''} \rightarrow T_{q,q'} U_{q',q''}$$



$$|\mathcal{G}_{\mathcal{G}\times\mathcal{A}}| = |\mathcal{G}| \cdot |\mathcal{A}|^3 = n \cdot |Q_{\mathcal{F},\mathcal{M}}|^3$$

Complexity of generation with mandatory patterns

Complexity in $O(2^{|\mathcal{M}|})$

Awaited: it's impossible to do "better"

Complexity of generation with mandatory patterns

Complexity in $O(2^{|\mathcal{M}|})$

Awaited: it’s impossible to do "better"

Superstring Problem

Data: Set $\{u_1, \ldots, u_k\}$ of words on Σ and $\ell \in \mathbb{N}$
Result : "True" if it exists u such that $\forall i \in [\![1, k]\!], u_i \preceq u$, et $|u| \leqslant \ell$, "False" otherwise

This problem is NP-complete (cf Maier *et al.* (1977))

Superstring NP-hard $\Rightarrow$ **Design** NP-hard

- $S = \bullet\bullet\cdots\bullet$ of length ℓ
- $\mathcal{F} = \varnothing$
- $\mathcal{M} = \{u_1, \ldots, u_k\}$

Then **Superstring** $\prec$ **pre-Design**

$\rightsquigarrow$ no hope **Design with constraints** $\in$ Poly($|\mathcal{M}|$)

Pattern once and only once

Weakness of Zhou *et al.* approach:
How to enforce a pattern u_{mf} *only* at position $i_{u_{mf}}$ and nowhere else?

- use $\mathcal{G}_S$?
  $\mathcal{A}_{\mathcal{F},\mathcal{M}}$ forbids $u_{mf} \rightsquigarrow \mathcal{L}_{\mathcal{G}_S} \cap \mathcal{L}_{\mathcal{A}_{\mathcal{F},\mathcal{M}}} = \varnothing$
- use $\mathcal{A}_{\mathcal{F},\mathcal{M}}$?
  $|Q_{\mathcal{F},\mathcal{M}}| \in \Omega(n) \rightsquigarrow$ complexity in $O(n|Q_{\mathcal{F},\mathcal{M}}|^3) = O(n^4)$

Pattern once and only once

Weakness of Zhou *et al.* approach:
How to enforce a pattern u_{mf} *only* at position $i_{u_{mf}}$ and nowhere else?

- use $\mathcal{G}_S$?
  $\mathcal{A}_{\mathcal{F},\mathcal{M}}$ forbids $u_{mf} \rightsquigarrow \mathcal{L}_{\mathcal{G}_S} \cap \mathcal{L}_{\mathcal{A}_{\mathcal{F},\mathcal{M}}} = \varnothing$
- use $\mathcal{A}_{\mathcal{F},\mathcal{M}}$?
  $|Q_{\mathcal{F},\mathcal{M}}| \in \Omega(n) \rightsquigarrow$ complexity in $O(n|Q_{\mathcal{F},\mathcal{M}}|^3) = O(n^4)$

$\Longrightarrow$ method in $O(n|Q_{\mathcal{F},\mathcal{M}}|^3)$ where Q depends on $\mathcal{F}$ and $\mathcal{M}$, but not n

Dynamic programming : idea

$C(i,j,q,q'')$ = number of words w compatible with $S_{[i,j]}$ such that $q \xrightarrow{w} q'' \in \delta^*_{\mathcal{F},\mathcal{M}}$

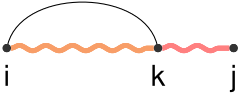
$$\begin{array}{ccccccccccccc} (& (& (& (& (& & \cdots & &) &) &) &) &) \\ & & i & & & & & & & j & & & \\ & q & & & & & & & & & & q'' & \end{array}$$

$C(i,j,q,q'')$ needed for random generation

Dynamic programming : method

Computing $C(i, j, q, q'')$:

- 

  - choose $a \in \Sigma$
  - compute $C(i+1, j, q_{\text{after}}, q'')$ where $q \xrightarrow{a} q_{\text{after}} \in \delta_{\mathcal{F},\mathcal{M}}$
- 

  - choose $a, b \in \Sigma$
  - choose $q' \in Q$
  - compute $C(i+1, k-1, q_{a1}, q')$ where $q \xrightarrow{a} q_{a1} \in \delta_{\mathcal{F},\mathcal{M}}$
  - compute $C(k+1, j, q_{a2}, q'')$ where $q' \xrightarrow{b} q_{a2} \in \delta_{\mathcal{F},\mathcal{M}}$

Dynamic programming : recurrence relations

- $S_{[i,i-1]} = \varepsilon$

$$C(i, i-1, q, q'') = \begin{cases} 1 & \text{if } q = q'' \\ 0 & \text{otherwise} \end{cases}$$

- $S_{[i,j]} = \bullet S_{[i+j,j]}$

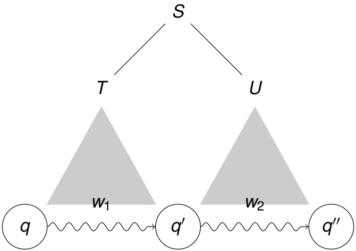
$$C(i, j, q, q'') = \sum_{a \in C_i} \begin{cases} 0 & \text{if } \delta(q, a) \in \mathcal{F} \\ 0 & \text{if } \delta(q, a) = u_{mf} \text{ and } i - (|u_{mf}| - 1) \neq i_{u_{mf}} \\ C(i+1, j, \delta(q, a), q'') & \text{otherwise} \end{cases}$$

- $S_{[i,j]} = (S_{[i+1,k-1]})S_{[k+1,j]}$

$$C(i, j, q, q'') = \sum_{\substack{(a,b) \in C_i \times C_j \\ q' \in Q}} \begin{cases} 0 & \text{if } \delta(q, a) \in \mathcal{F} \text{ ou } \delta(q', b) \in \mathcal{F} \\ 0 & \text{if } \delta(q, a) = u_{mf} \text{ and } i - (|u_{mf}| - 1) \neq i_{u_{mf}} \\ 0 & \text{if } \delta(q', b) = u_{mf} \text{ and } k - (|u_{mf}| - 1) \neq i_{u_{mf}} \\ C(i+1, k-1, \delta(q, a), q') \cdot C(k+1, j, \delta(q', b), q'') & \text{otherwise} \end{cases}$$

Speed enhancement

CFGRNAD tests
every $(q, q'') \in Q^2$
every $q' \in Q$
$\rightsquigarrow O(|Q|^3)$



But:

- accessible/co-accessible ?
- non-terminal $\rightarrow$ fixed-length words

$\rightsquigarrow$ most (q, q', q''), even (q, q''), are *irrelevant*

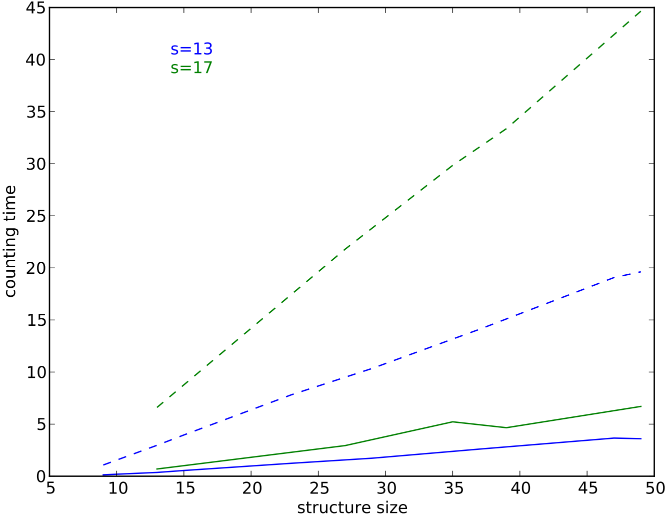
Speed enhancement: method

Optimisation : pre-compute triplets (q, q', q'') such that there is a path $q \rightarrow^* q' \rightarrow^* q''$ in $\mathcal{A}_{\mathcal{F},\mathcal{M}}$.

$q \rightarrow^{n_1} q' \rightarrow^{n_2} q''$:

- $q \rightarrow^{n_1+n_2} q'' \in \delta^*_{\mathcal{F},\mathcal{M}}$
- $q \rightarrow^{n_1} q' \in \delta^*_{\mathcal{F},\mathcal{M}}$ et $q' \rightarrow^{n_2} q'' \in \delta^*_{\mathcal{F},\mathcal{M}}$

Speed enhancement : preliminary results



Better algorithmic complexity ?

Open questions

- $\mathcal{A}_{\mathcal{F},\mathcal{M}}$ optimal *in general*?
  (minimizing is costly, problems with prog. dyn.)
- folding of w into S not guaranteed $\leadsto$ enforce a *bias* in the random generation
- characterize families of automatons with whom complexity in $O(n \cdot |Q|^p), p < 3$

Thank you for your attention