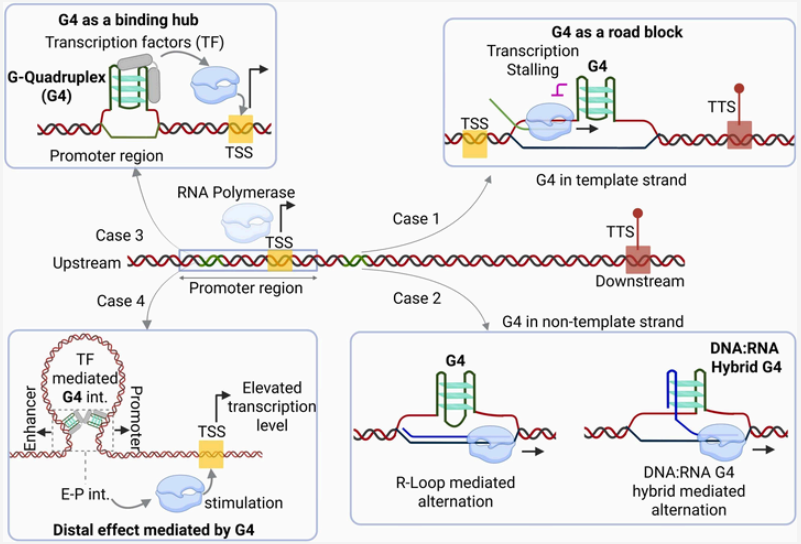
Finding templates for template-free modelling

A quest to predict quadruplexes from sequence

Tomasz Żok
Institute of Computing Science, Poznan University of Technology

Quadruplexes

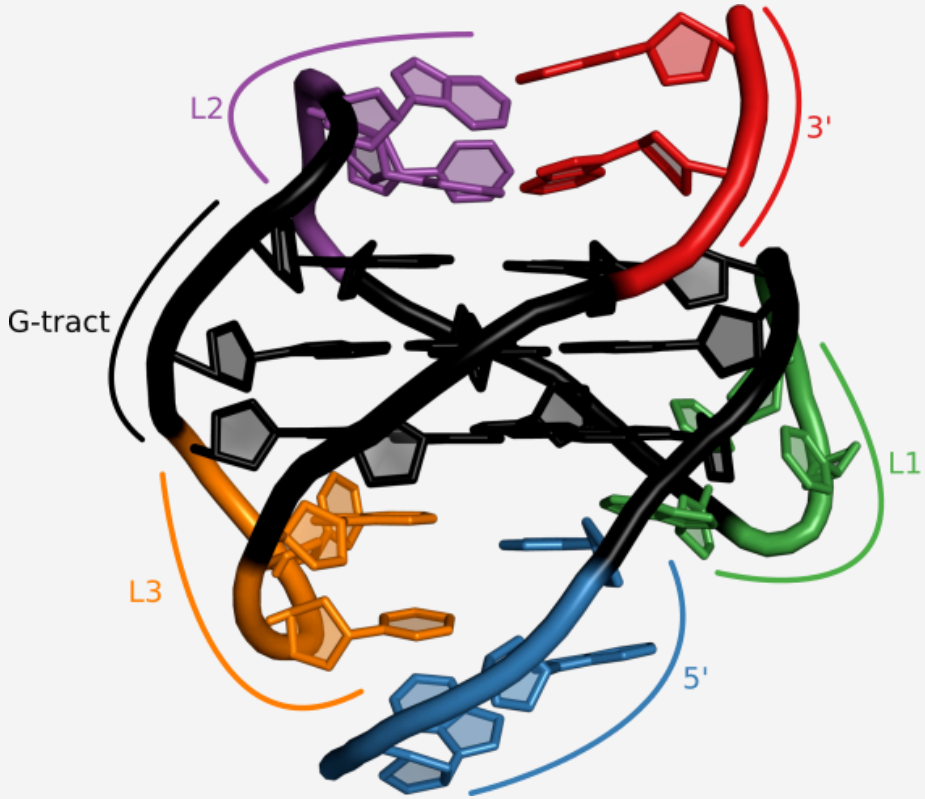
- Quadruplexes form in **G-rich regions** of genomes, including human **telomeres** and **promoters**
- Involved in **regulation of gene expression**, replication, genome stability
- Their presence or absence is linked to **cancers** and **neurodegenerative diseases**
- Potential **therapeutic targets** — and stabilized by small-molecule ligands



Jonchhe, S., Lahiri, S., & Rothenberg, E. (2025). DNA G-quadruplexes: Structural and functional insights. DNA Repair, 156, 103910. https://doi.org/10.1016/j.dnarep.2025.103910

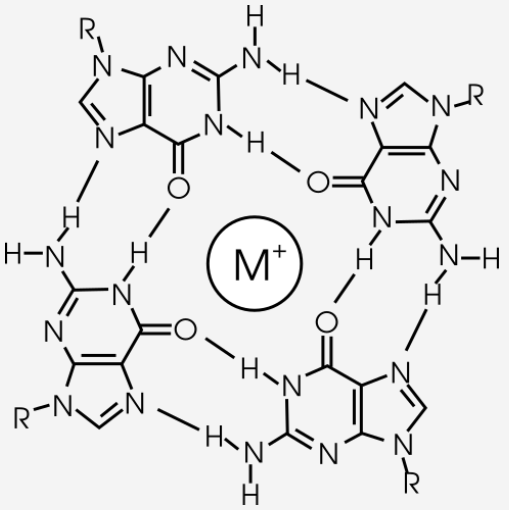
What is a quadruplex?

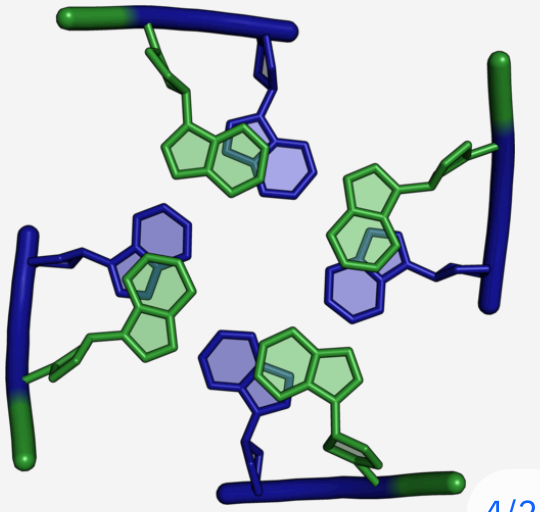
- A **four-stranded** structural motif of DNA, RNA, and analogs
- Built from four **G-tracts** connected by three **loops**
  At least the canonical ones...
- Found across many species, including humans



Tetrads

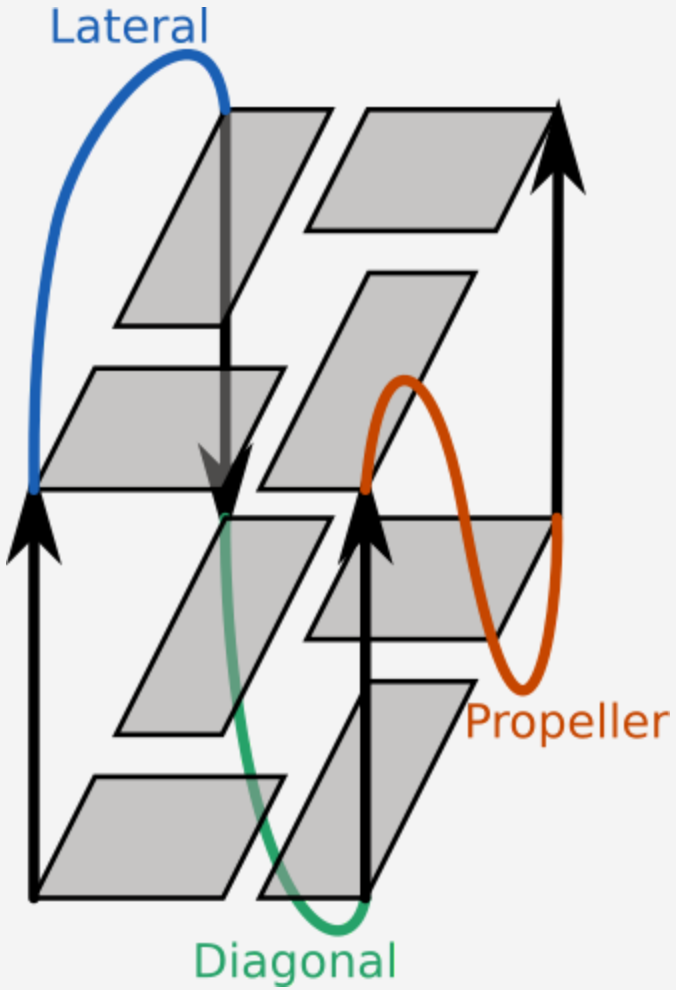
- A **tetrad** = **4 nucleotides** whose bases interact pairwise
  - Usually **guanines** via **cis Watson-Crick / Hoogsteen** edges
  - The tetrad is (approximately) **planar** and often stabilized by a **central ion**
  - Neighbouring tetrads **stack** on top of each other
- A **quadruplex** = **2+ stacked tetrads**



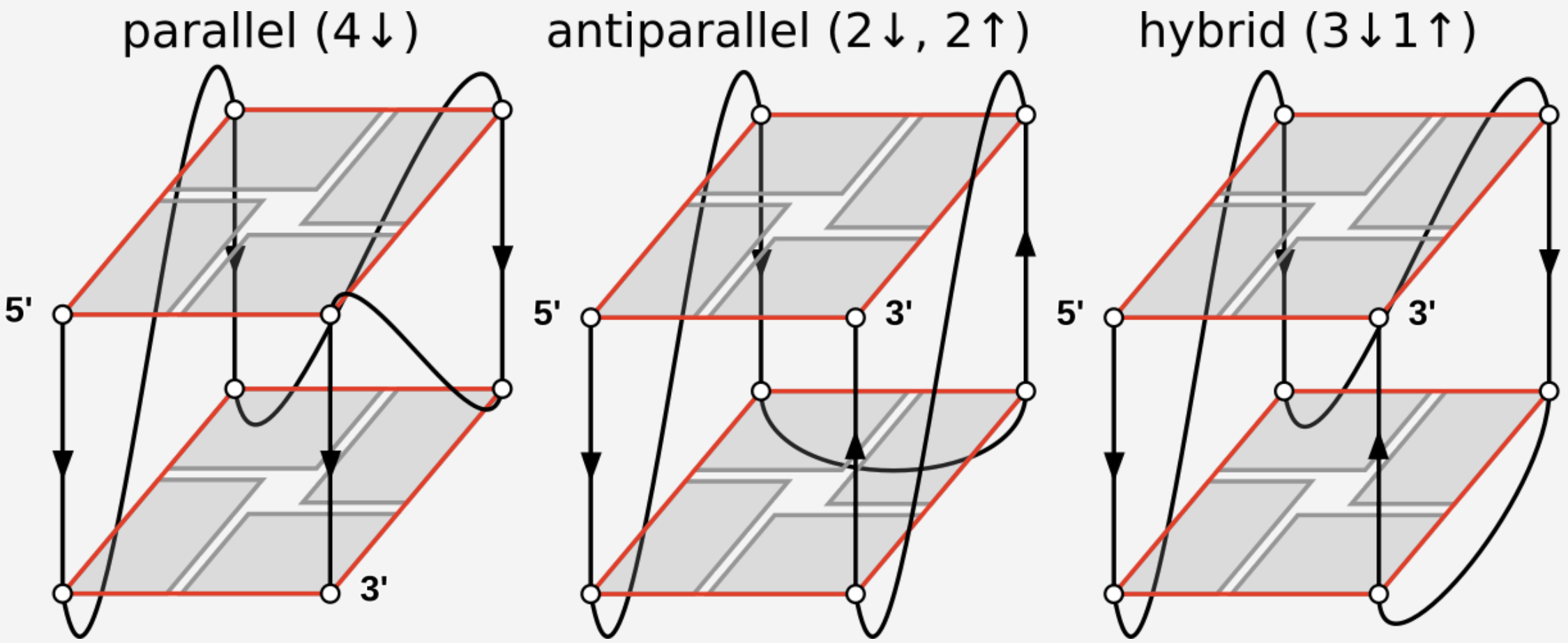


Loops

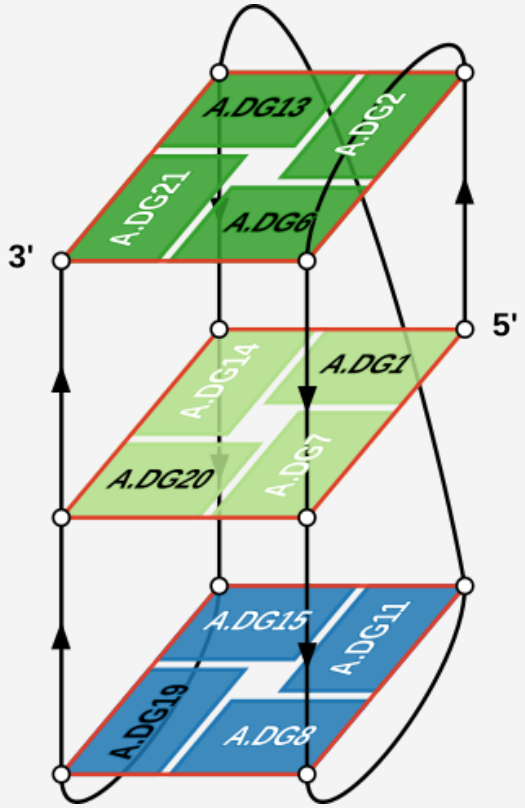
- The linker sequences connecting G-tracts are **loops**
- Three loop types by geometry:
  - **lateral** — connects adjacent strands on one side
  - **diagonal** — crosses the structure
  - **propeller** — connects strands on opposite sides (loop strands run parallel)
- Loop arrangement is one basis for classification

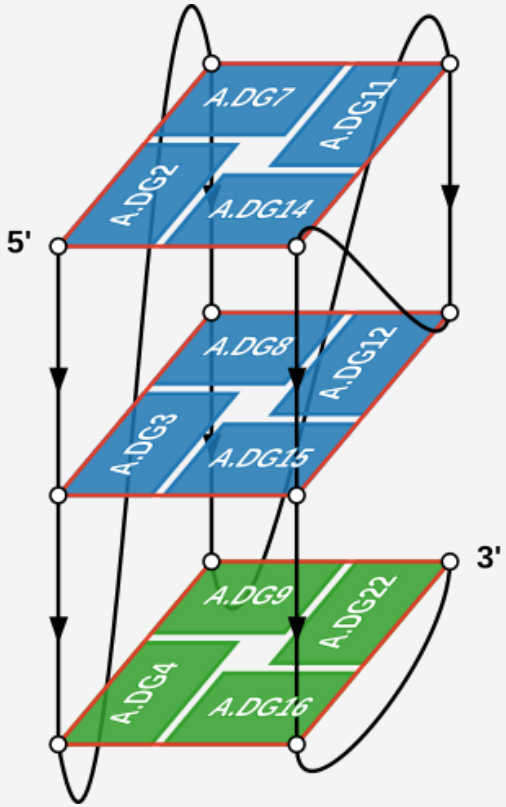


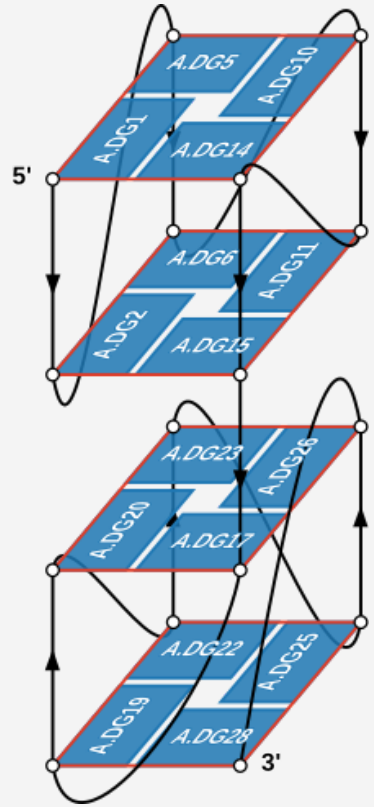
Strand directionality



Cases that break the schemes







V-Loop: First tetrad is *sandwiched* between two others

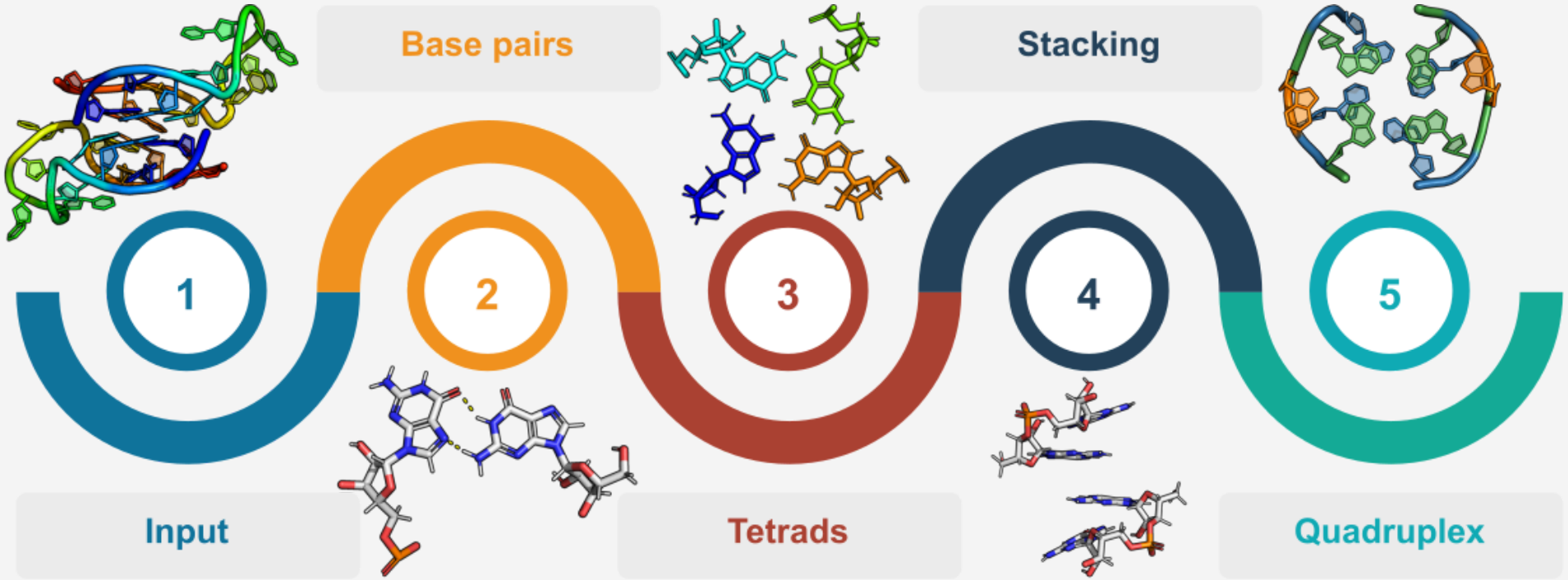
Snapback: a terminal nucleotide is not part of a tract

Two-block: simultaneous right-handed and left-handed components

The challenge: finding quadruplexes in PDB

- The PDB holds **~14k** experimentally solved nucleic acid 3D structures
- **Nothing** in the metadata indicates whether a structure contains a quadruplex

```
ATOM      1  OP3   G A   1      50.193  51.190  50.534  1.00 99.85           O
ATOM      2  P     G A   1      50.626  49.730  50.573  1.00100.19           P
ATOM      3  OP1   G A   1      49.854  48.893  49.562  1.00100.19           O
ATOM      4  OP2   G A   1      52.137  49.542  50.511  1.00 99.21           O
ATOM      5  O5'   G A   1      50.161  49.136  52.023  1.00 99.82           O
ATOM      6  C5'   G A   1      50.216  49.948  53.210  1.00 98.63           C
ATOM      7  C4'   G A   1      50.968  49.231  54.309  1.00 97.84           C
ATOM      8  O4'   G A   1      50.450  47.888  54.472  1.00 97.10           O
ATOM      9  C3'   G A   1      52.454  49.030  54.074  1.00 98.07           C
ATOM     10  O3'   G A   1      53.203  50.177  54.425  1.00 99.39           O
```



The complete suite so far

- **ElTetrado** — CLI: find & classify quadruplexes from 3D coordinates
  https://github.com/RNApolis/eltetrado/
- **DrawTetrado** — CLI: automatic layer-diagram layouts
  https://github.com/RNApolis/drawtetrado/
- **ONQUADRO** — database of all known 3D quadruplex structures
  https://onquadro.cs.put.poznan.pl/
- **WebTetrado** — web app to analyze your own structures
  https://webtetrado.cs.put.poznan.pl/

The inverse problem

- So far we have **3D → analysis**: given coordinates, find quadruplexes
- The natural next question is the **inverse**:

> Given a sequence, can it fold into a quadruplex we already know?

- This motivates a **template-based** approach

- We have a rich library of templates: **ONQUADRO**
- We want to **align** an input sequence against these templates *in quadruplex-native way*
- Useful for **sequence screening**, **annotation**, and as input to **3D modeling** (not necessarily template-based!)
- This is what ***ONQUADRO-aligner*** does

ONQUADRO-aligner: the idea

- **Input**: a nucleic acid sequence
- **Templates**: all unique single-chain G-only quadruplex patterns extracted from the ElTetrado corpus
- For each template, **enumerate** G-combinations, **generate** a candidate, and **score** it
- Report the **best** template and **fit** the input sequence into it

Three scored components per candidate:

1. **Tract distance** — are required tracts present where expected?
2. **Linker distance** — how different sequentially are the loops?
3. How **viable** are the loops according to the literature knowledge?

Quadruplex description model

1. **Nucleic acid type** — DNA / RNA / Other
2. **QRS** — a vector marking which G positions form tetrads e.g. $[0, 1, 1, 0, 0, 1, 1, 0]$
3. **Linker sequences** — the loop strings, e.g. $[\varepsilon, \varepsilon, TT, \varepsilon, TGT, \varepsilon, TT, \varepsilon, \varepsilon]$


```
1 [{"molecule":"DNA","qrs":[1,2,3,3,2,1,1,2,3,3,2,1],"description":["A","","","TTA","","","TTA",""
2 {"molecule":"DNA","qrs":[1,2,2,1,1,2,2,1],"description":["","","TT","","TGT","","TT","",""]},
3 {"molecule":"DNA","qrs":[1,2,3,3,2,1,1,2,3,1,2,3],"description":["TT","","","GTTG","","","TTG","
4 {"molecule":"DNA","qrs":[1,2,2,1,1,2,2,1],"description":["","","TT","","TGT","","TT","",""]},
5 {"molecule":"DNA","qrs":[1,2,2,1,2,1,1,2],"description":["","","TTTT","","CA","","GTTTT","","T"]
6 {"molecule":"DNA","qrs":[1,2,3,1,2,3,1,2,3,1,2,3],"description":["A","","","TTA","","","TTA","",
7 {"molecule":"RNA","qrs":[1,2,1,2,1,2,1,2],"description":["","","A","","UUUU","","A","",""]},
8 {"molecule":"DNA","qrs":[2,1,2,1,2,1,2,1],"description":["","","A","","A","","A","","A"]},
9 {"molecule":"DNA","qrs":[1,2,1,2,1,2,1,2],"description":["","","A","","A","","A","","A"]},
10 {"molecule":"DNA","qrs":[1,1,1,1,2,3,2,3,2,3,2,3],"description":["","GA","GA","GA","GA","","A","
```

Example: 148D

```
GGTTGGTGTGGTTGG
qr..rq...qr..rq
```

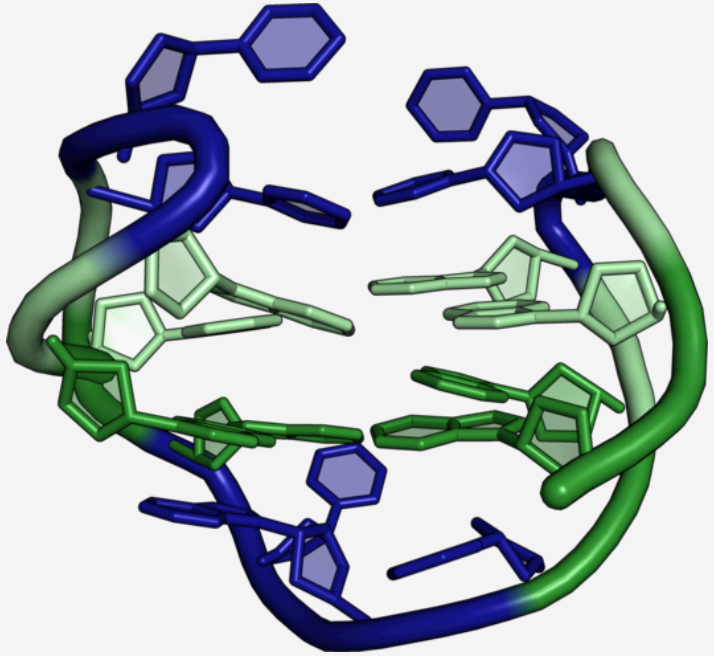
QRS: $[0, 1, 1, 0, 0, 1, 1, 0]$

Linkers: $[\varepsilon, \varepsilon, TT, \varepsilon, TGT, \varepsilon, TT, \varepsilon, \varepsilon]$

QRS length is always $4n_{tetrads}$

Linkers length is always $4n_{tetrads} + 1$

An ε in Linkers vector on positions other than the first and last represents **no bulge**. It is extremely important to preserve as it cannot be *easily* compensated.



Input processing

1. Input sequence: `AGGTGGTTTGGTTGGGA`
2. Indices of Gs: $[2, 3, 5, 6, 10, 11, 14, 15, 16]$
3. 9 Gs $\Rightarrow$ 2 tetrads $\Rightarrow$ need 8 Gs
4. Enumerate $\binom{9}{8} = 9$ combinations:

```
[(2, 3, 5, 6, 10, 11, 14, 15),
 (2, 3, 5, 6, 10, 11, 14, 16),
 (2, 3, 5, 6, 10, 11, 15, 16),
 (2, 3, 5, 6, 10, 14, 15, 16),
 (2, 3, 5, 6, 11, 14, 15, 16),
 (2, 3, 4, 10, 11, 14, 15, 16),
 (2, 3, 5, 10, 11, 14, 15, 16),
 (2, 5, 6, 10, 11, 14, 15, 16),
 (3, 5, 6, 10, 11, 14, 15, 16)]
```

Candidate generation from a template

1. Load template **148D**:
   - QRS: $[0, 1, 1, 0, 0, 1, 1, 0]$
   - Linkers: $[\varepsilon, \varepsilon, TT, \varepsilon, TGT, \varepsilon, TT, \varepsilon, \varepsilon]$
2. Generate candidate from combination 1: `(2,3,5,6,10,11,14,15)`
   - Description:

     ```
     AGGTGGTTTGGTTGGGA
     .qr.rq...qr..rq..
     ```

   - Linkers: $[A, \varepsilon, T, \varepsilon, TTT, \varepsilon, TT, \varepsilon, GA]$
3. Generate candidate from combination 2: `(2,3,5,6,10,11,14,16)`
   - Description:

     ```
     AGGTGGTTTGGTTGGGA
     .qr.rq...qr..r.q.
     ```

   - Linkers: $[A, \varepsilon, T, \varepsilon, TTT, \varepsilon, TT, G, A]$
4. And so on...

Distance metric: tracts + linkers

Tract distance

- For each **interior** segment (not first/last) of the template:
  - if the template segment is ε, add the length of the candidate's segment there
- Counts inserted **bulges** where they do not belong!

Linker distance

- For each segment, compute the **edit distance** between template and candidate linker strings
- Counts **insertions / deletions / mutations** to mutate the template into query sequence

Distance metric: example

- **148D** linkers: $[\varepsilon, \varepsilon, TT, \varepsilon, TGT, \varepsilon, TT, \varepsilon, \varepsilon]$

- Combination 1: $[A, \varepsilon, T, \varepsilon, TTT, \varepsilon, TT, \varepsilon, GA]$
- Tract distance: 0
- Linker distance:
  $(\varepsilon \text{ vs } A) + (TT \text{ vs } T) + (TGT \text{ vs } TTT) + (\varepsilon \text{ vs } GA) = 1 + 1 + 1 + 2 = 5$

- Combination 2: $[A, \varepsilon, T, \varepsilon, TTT, \varepsilon, TT, G, A]$
- Tract distance: $\varepsilon \text{ vs } G = 1$
- Linker distance:
  $(\varepsilon \text{ vs } A) + (TT \text{ vs } T) + (TGT \text{ vs } TTT) + (\varepsilon \text{ vs } G) + (\varepsilon \text{ vs } A) = 1 + 1 + 1 + 1 + 1 = 5$

The brute-force cost

- For each template, enumerate **every** $\binom{m}{k}$ combination of G positions, where m = query G count, $k = 4 \cdot n_{\text{tetrads}}$
- Each combination runs up to $k + 1$ pairwise alignments
- Real example: 35-nt query, 18 Gs, 3 tetrads $\Rightarrow k = 12$, $\binom{18}{12} = 18{,}564$ combinations **per template**, across **201** three-tetrad templates
- ≈ 3.7M combinations total; ~5.15s/template, ~110s on 12 cores

- Cost grows **combinatorially** with the query's G count
- A longer query, or one with more spurious (non-quadruplex) Gs, quickly becomes intractable
- Example: $m = 30, k = 16 \Rightarrow \binom{30}{16} \approx 145{,}000{,}000$ combinations
- We needed a **smarter search** — without changing the result

The dynamic-programming search

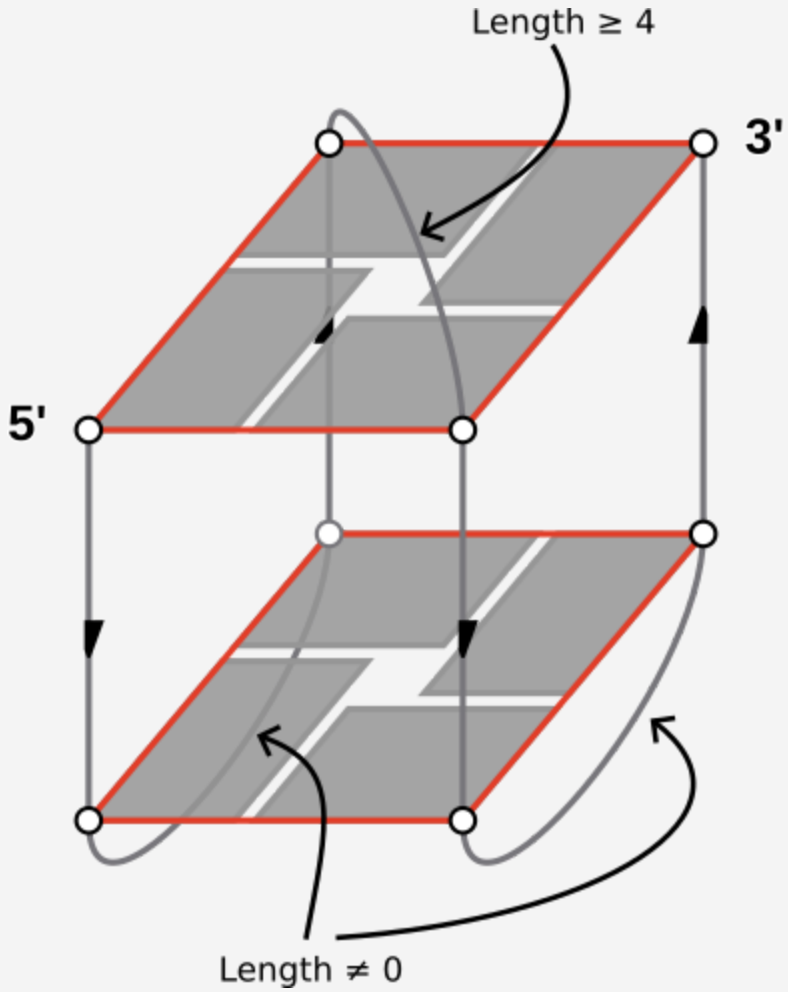
- Fortunately, we can decompose the problem and solve it with **dynamic programming**
  - States: $O(k \cdot m)$
  - Transitions per state: $O(m)$
  - Complexity: $O(k \cdot m^2)$

	-1	1	2	4	5	9	10	13	14	15	
0	0 0										
1		0 1	0 2	0 4	0 5	0 9	0 10	0 13	0 14	0 15	ε
2			**0 1**	**1 3**	**0 4**	**3 8**	**0 9**	**2 12**	**0 13**	**0 14**	**ε**
3				0 2	0 2	0 5	0 6	0 9	0 10	0 11	TT
4					**0 2**	**3 5**	**0 5**	**2 8**	**0 9**	**0 10**	**ε**
5						0 3	0 4	0 6	0 7	0 7	TGT
6							**0 3**	**2 6**	**0 6**	**0 7**	**ε**
7								0 3	0 4	0 5	TT
8									**0 3**	**0 4**	**ε**
9										0 5	ε

`AGGTGGTTTGGTTGGGA`

Viability

- A match with zero tract distance and small linker distance looks almost perfect by our metrics
- But the **loop lengths it implies** may be geometrically impossible:
  - a 0-nt loop (two tetrad Gs directly adjacent)
  - too short diagonal loops
  - some loop-length/topology combinations never observed



Viability: literature

Table 3. Loop lengths of the one-block G4s and the corresponding topologies, mostly when $L_1 = L_3$. The percentage for a given combination of loop lengths is calculated as the number of structures corresponding to the topology divided by all the structures with this combination of loop lengths. For instance, when $L_1 = L_2 = L_3 = 2$, 73% of the structures are parallel and 27% are chair. When there are several L_2 lengths, the percentages are given for each length in the same order. If only one structure exists with the given combination, its PDB ID is provided

L_1	L_2	L_3	Topology	Percentage of structures
0-nt between the strands 1 and 4 (11 structures)			hybrid4	100%
		$L_1 = L_3 = 1$ nt (73 structures)		
1	1 to 15	1	parallel	100%
		$L_1 = L_3 = 2$ nts (59 structures)		
2	1–2	2	parallel	100% – 73%
2	2–3 or 15	2	chair	27% – 100% or 100%
2	4–5	2	basket	100%
		$L_1 = L_3 = 3$ nts (35 structures)		
3	1	3	parallel	100%
3	2	3	hybrid3	100% (6AC7 [77])
3	3	3	hybrid1	50%
			hybrid3	27%
			chair	19%
			basket2	4% (2MBJ)
3	4–5	3	basket	75%
			basket2	25%
		$L_1 = L_3 = 4$ nts (7 structures)		
4	1 or 4	4	basket	14% (2MFT) – 43%
4	4	4	chair	14% (2KM3 [99])
4	10	4	hybrid2	29%
		$L_1 = L_3 = 5$ nts (1 structure)		
5	1	5	parallel	100% (8EDP [100])
		$L_1 \geq 6; L_3 \geq 6$ nts (8 structures)		
7	4	19	chair	100%

Granzhan, A., & Mouawad, L. (2026). Topological rules and anomalies in intramolecular G-quadruplex folding: A comprehensive study. Nucleic Acids Research, 54(9), gkag435. https://doi.org/10.1093/nar/gkag435

Viability: literature

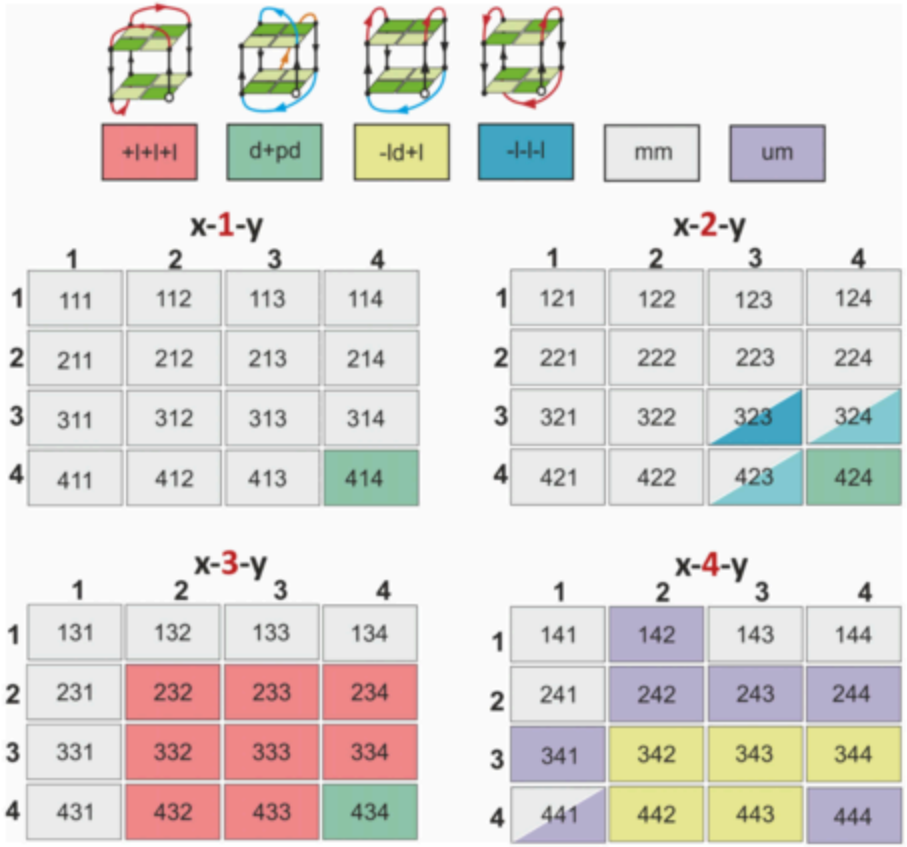


Figure 7. The dominant G4 fold under NMR conditions as a function of the three loop lengths. For unimolecular G4s the assigned loop topology is indicated (if known; otherwise marked as unimolecular, "um"), while multimolecular G4s are grouped into a single category ("mm", gray). Fields corresponding to sequences that produce comparable populations of uni- and multimolecular species are shown split in half into a gray and colored triangle. Systems for which loop topology assignment is tentative—**324** and **423**—are shown in a lighter shade of the appropriate color.

Woś, A., ..., Gudanis-Sobocińska, D., & Andrałojć, W. (2026). How loop lengths shape topological preferences of two-tetrad G-quadruplexes. Nucleic Acids Research, 54(11), gkag590. https://doi.org/10.1093/nar/gkag590

onquadro-aligner

Search a sequence against known single-chain G-quadruplex structures

Sequence

AGGTGGTTTGGTGGGA

Tetrad count

0

List all matching files

Search

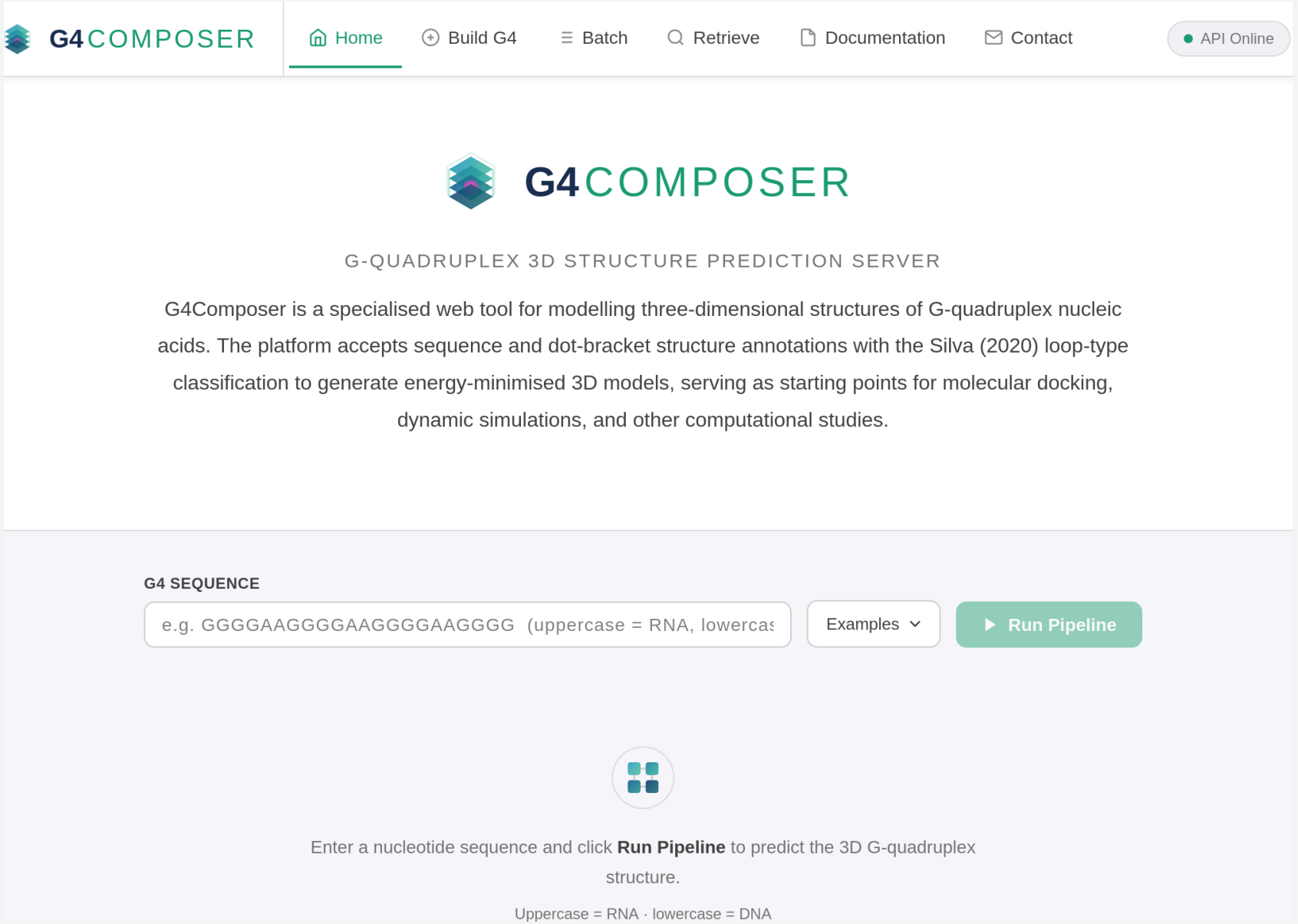
Results (26 matches)

		Perfect	Viability	Tract	Linker	Tetrads	Template	Loops	Topology	QRS
☑	0		🟢 viable	0	6	2	1my9-assembly1	1-3-1	-p-p-p	`.RS.rs...RS.rs..`
☐	1		🟢 viable	0	7	2	2rqj-assembly1	1-3-1	+p+p+p	`.QR.qr...QR.qr..`
☐	2		🟢 viable	0	12	2	8utg-assembly1	1-3-1	-p-p-p	`.QR.qr...QR.qr..`
☐	3		🟢 viable	0	23	2	6e80-assembly1	1-3-1	+p+p+p	`.QR.qr...QR.qr..`
☐	4		🟢 viable	0	24	2	5bjo-assembly1	1-3-1	-p-p-p	`.QR.qr...QR.qr..`
☐	5		⚪ unknown	2	30	2	4wb2-assembly1	1-3-1		`.QR.qr...QR.qr..`
☐	6		🔴 not_viable	0	20	2	6e8t-assembly1	1-3-1	-p-p-l	`.QR.qr...QR.rq..`
☐	7		🔴 not_viable	0	21	2	6e8t-assembly2	1-3-1	-p-p-l	`.QR.qr...QR.rq..`
☐	8		🔴 not_viable	0	22	2	6e8u-assembly1	1-3-1	-p-p-l	`.QR.qr...QR.rq..`

Template-based template-free modelling

- The matches from ONQUADRO-aligner are used to extract folding parameters: **rise**, **twist**, **polarities**
- The folding itself does not use full homologous structures or even their fragments

```
name          6k3y-assembly1
sequence      aggtggtttggtggga
structure     .^^.^^...^^.^^..
chi           ................
sugar         ................
orient        A-;B-
rise          3.2
twist         31.1
path          A1;B1;A4;B4;A3;B3;A2;B2
```


G4COMPOSER
Home
Build G4
Batch
Retrieve
Documentation
Contact
API Online
G4COMPOSER
G-QUADRUPLEX 3D STRUCTURE PREDICTION SERVER
G4Composer is a specialised web tool for modelling three-dimensional structures of G-quadruplex nucleic acids. The platform accepts sequence and dot-bracket structure annotations with the Silva (2020) loop-type classification to generate energy-minimised 3D models, serving as starting points for molecular docking, dynamic simulations, and other computational studies.
G4 SEQUENCE
e.g. GGGGAAGGGGAAGGGGAAGGGG (uppercase = RNA, lowercas
Examples
Run Pipeline
Enter a nucleotide sequence and click Run Pipeline to predict the 3D G-quadruplex structure.
Uppercase = RNA · lowercase = DNA

Acknowledgements

Funding: National Science Centre (NCN), Poland — grant no. 2024/53/B/ST6/02789 and 2023/51/D/ST6/01207

NATIONAL SCIENCE CENTRE
POLAND