

# Bio Project

Stijn Rosaer<sup>1</sup> and Olivier Van Houtte<sup>2</sup>

1- Universiteit Antwerpen - Computer Science  
Prinsstraat 13 2000 Antwerpen - Belgium

**Abstract.** Zoals we weten, is er interesse in het storen van data als DNA. Er bestaan al veel methoden om dit DNA op te bouwen en terug uit te lezen, maar de encodings wijze staat nog niet vast. In dit verslag achterhalen we ofdat er misschien tijd en/of plaats efficiëntere encodings systemen zijn dan gewoon bit voor bit op te slaan en of deze in de praktijk toepasbaar zijn.

## 1 Inleiding

Als we aan DNA denken, worden we al snel herinnerd aan de 4 basen. Adenine (A), guanine (G), cytosine (C) en thymine (T). Met deze basen kunnen we de simpelste vorm van encoding opstellen, namelijk 2 bits per base. We kiezen dan voor elke base een unieke combinatie aan bits.

00 = A, 01 = G, 10 = C, 11 = T

We kunnen dan elke DNA string (met een even aantal bits) omzetten naar DNA.

0111001110 -> GTATC

Het kan echter efficiënter.

## 2 Encoding

In plaats van simpelweg 2 bit aan elke base toe te wijzen, stellen we nu eerst alle mogelijke sets op bestaande uit unieke combinaties van de 4 basen. Er zijn er zo 15.

```

A
G
C
T
A  G
A  C
A  T
G  C
G  T
C  T
A  G  C
A  G  T
A  C  T
G  C  T
A  G  C  T

```

We kunnen aan hier de getallen 0 t.e.m. 14 aan toewijzen. we kunnen deze zelfde getallen ook representeren met 4 bits. Hierbij komen op een probleem. Met 4 bits kan je de 16 getallen representeren, namelijk 0 t.e.m. 15. 1 Getal meer dan de 15 getallen die we met de base-sets bekomen. We zullen dus 1 bit moeten laten vallen. Dat brengt ons op 3 bits die de getallen 0 t.e.m. 7 voorstellen. We kiezen dan 8 sets uit onze 15, en wijzen aan elk van deze sets 1 getal tussen 0 en 7 toe.

0	A		000
1	G		001
2	C		010
3	T		011
4	A	G	100
5	A	C	101
6	A	T	110
7	G	C	111

### 3 Correctheid

De kans op het correct lezen van een character (Denk ik):

$$1 - \left(\frac{1}{2^N}\right)$$

Dat maakt de kans op een volledig correct gelezen string

$$\left(1 - \left(\frac{1}{2^N}\right)\right)^L$$

Met L het aantal characters in de DNA strings en N het aantal strings we gebruiken om te checken.

Onze tests komen zeer goed overeen met deze formule. We testen hier met een input DNA string van lengte  $L = 144$ ;

N	tests (%)	formule (%)	verschil
4	0.0	0.0	0.0
5	1.5	1.0	0.5
6	12.3	10.4	1.9
7	35.6	32.3	3.3
8	58.6	57.0	1.6
9	77.6	75.3	2.3
10	88.3	86.9	1.4
11	93.3	93.2	0.1
12	97.2	96.5	0.7
13	98.6	98.2	0.4
14	99.3	99.1	0.2
15	99.7	99.6	0.1
16	99.8	99.8	0.0
17	99.9	99.9	0.0
18	99.9	99.9	0.0
19	100	100	0.0
...	...	...	...
$\infty$	100	100	0.0

We verklaren de kleine verschillen met afrondings fouten.

## 4 In de praktijk

Het is simpeler om meerdere korte DNA strings te maken en lezen dan 1 lange. Dat is exact wat we met deze vorm van encoding doen. We maken i.p.v. 1 lange DNA string, veel korte, die we dan ook parallel kunnen inlezen indien nodig.

Zoals we weten bestaat DNA uit 2 strings. Zodat we zeker de juiste kant kiezen tijden het omzetten, kunnen we best elke DNA string beginnen en eindigen met een primer. Een kort stukje DNA van een vaste vorm, die het begin/ einde en de oriëntatie van het DNA aanduidt. Dit verlengt onze DNA string een beetje, maar zou een relatief kleine impact hebben op het eindresultaat.

## 5 Conclusie

Er is dus zeker nog meer onderzoek mogelijk naar DNA als data storage. Onze inspiratie voor deze paper was dan ook een artikel uitgebracht vorig jaar. Maar wij vinden dit al zeker een zeer goede encoding.

## References

- [1] Choi, Y., Ryu, T., Lee, A.C. et al. High information capacity DNA-based data storage with augmented encoding characters using degenerate bases. *Sci Rep* 9, 6582 (2019). <https://doi.org/10.1038/s41598-019-43105-w>