Lossless seeds for searching short patterns with high error rate

C. Vroland, M. Salson, and H. Touzet

University of Lille, France

IWOCA, 2014

Approximate String Matching

Searching a string that matches a pattern with errors

- pattern of size m
- text of size n
- maximal number of errors: k

Authorized errors

- substitution
- insertion
- deletion

A new algorithm for searching short patterns with high error rate:

- a new kind of seed: 01^*0 seed

Main existing searching methods

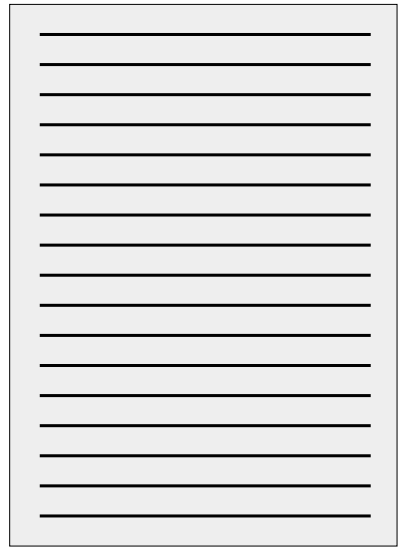
- Neighborhood exploration
- Exact seed filtration
- Hybrid method

Neighborhood Exploration

Pattern

GATTACA

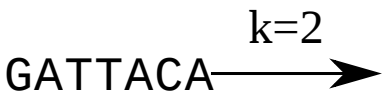
Text

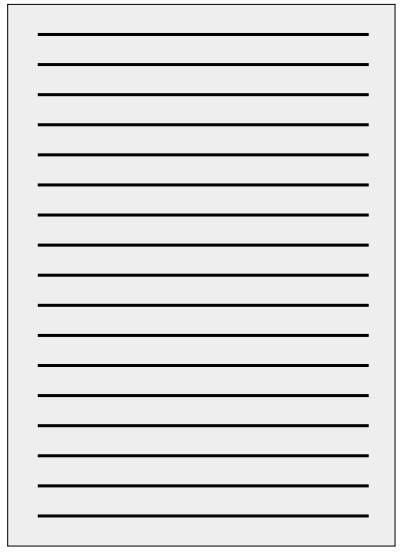


Neighborhood Exploration

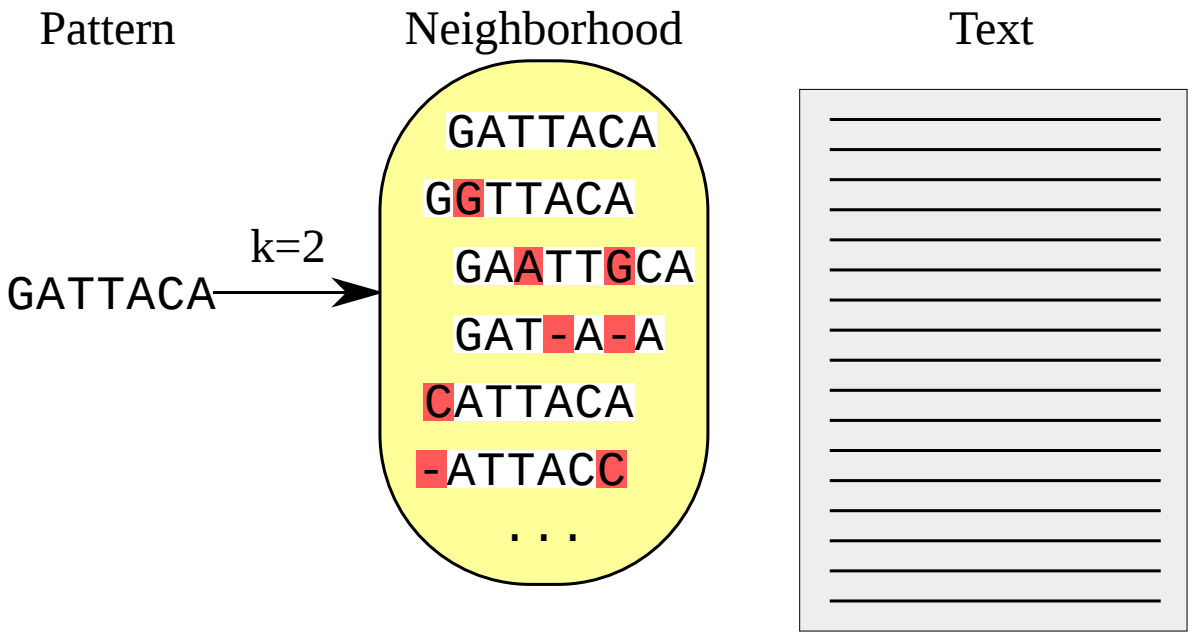
Pattern

Text

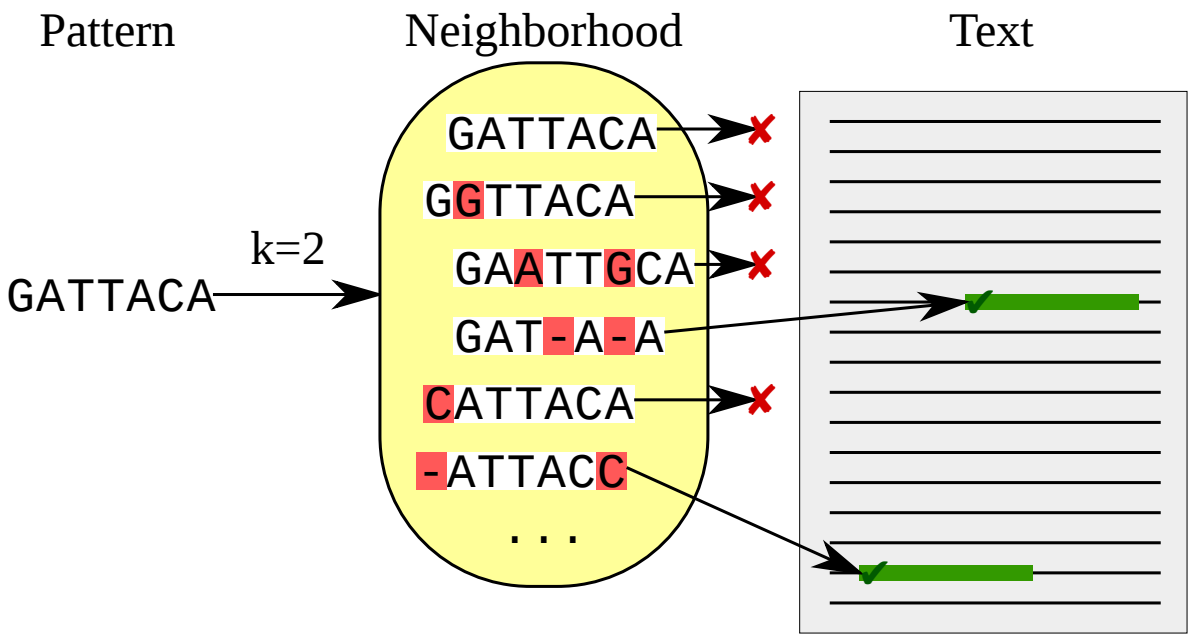




Neighborhood Exploration



Neighborhood Exploration

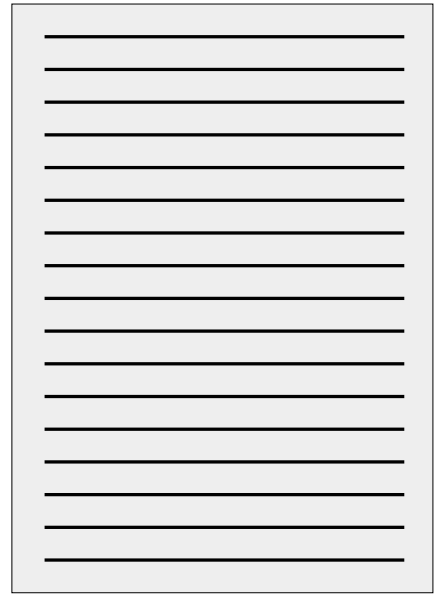


Exact Seed Filtration

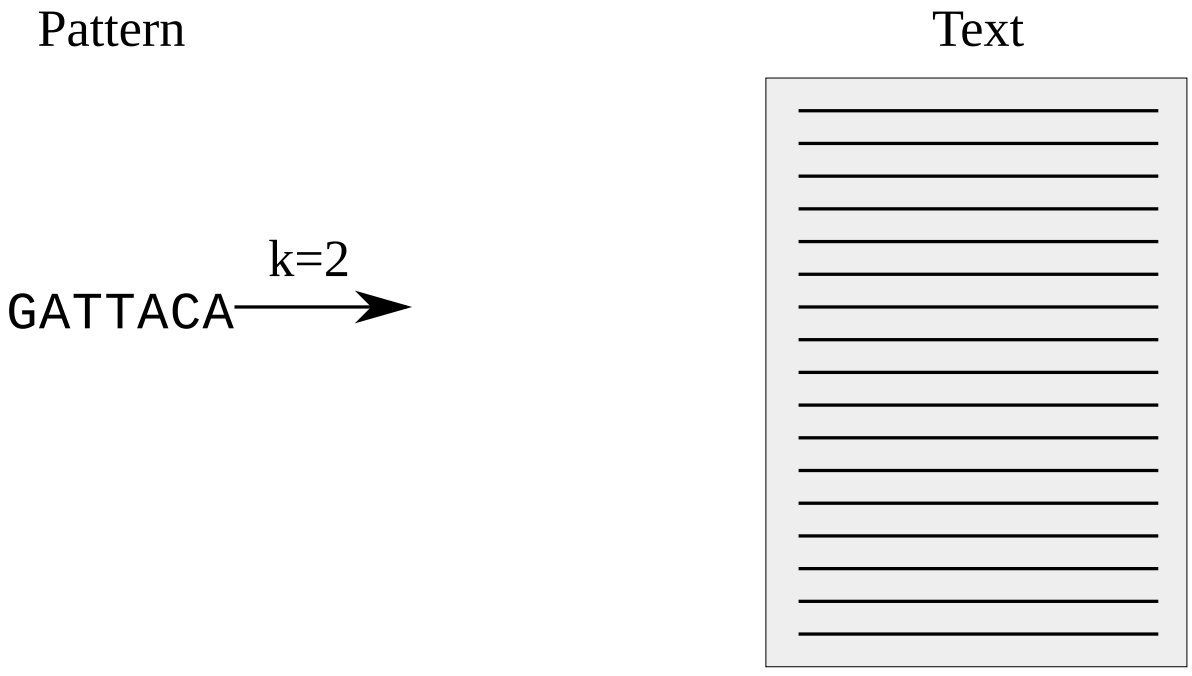
Pattern

GATTACA

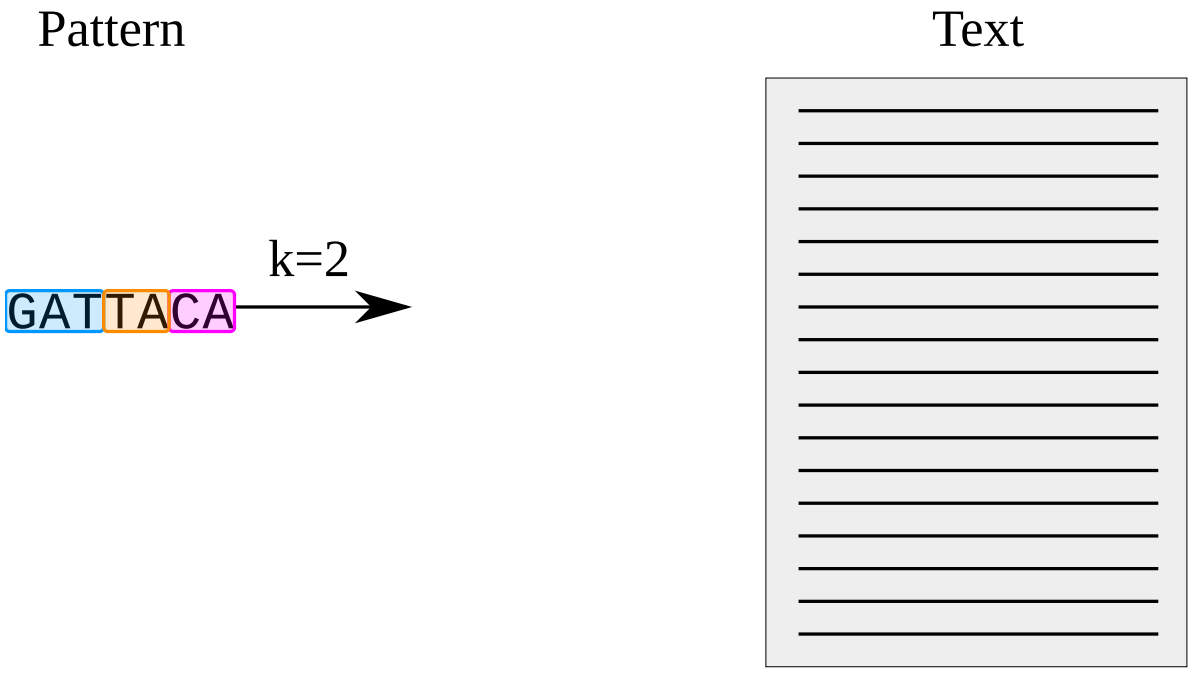
Text



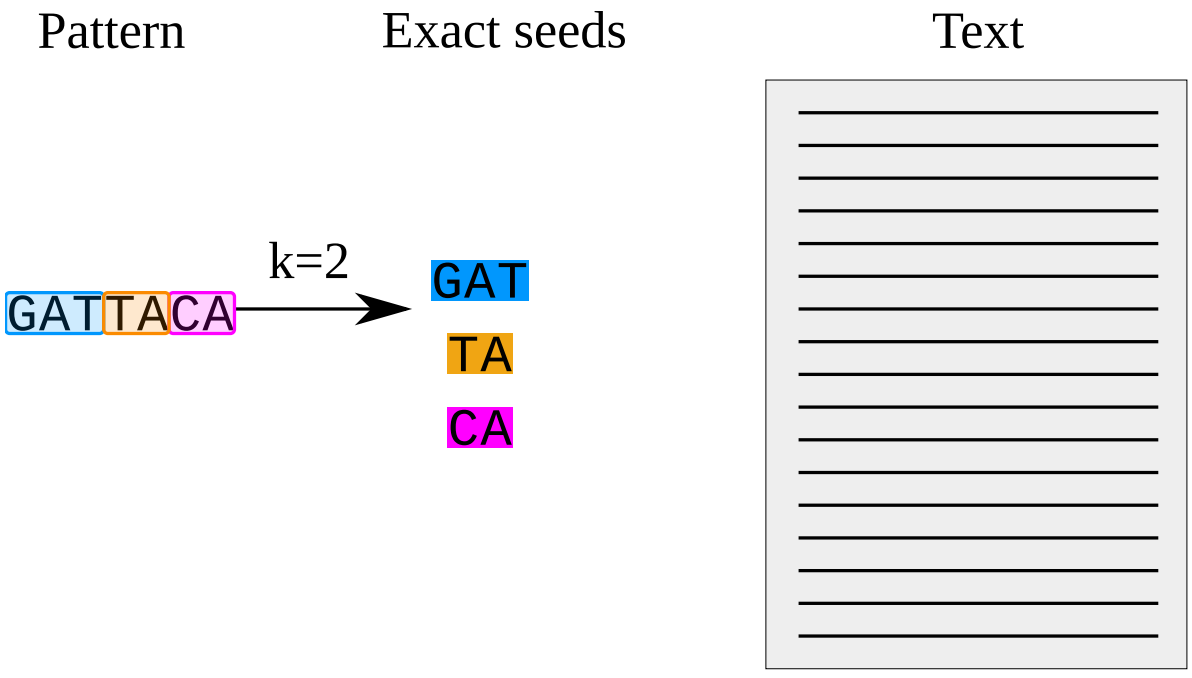
Exact Seed Filtration



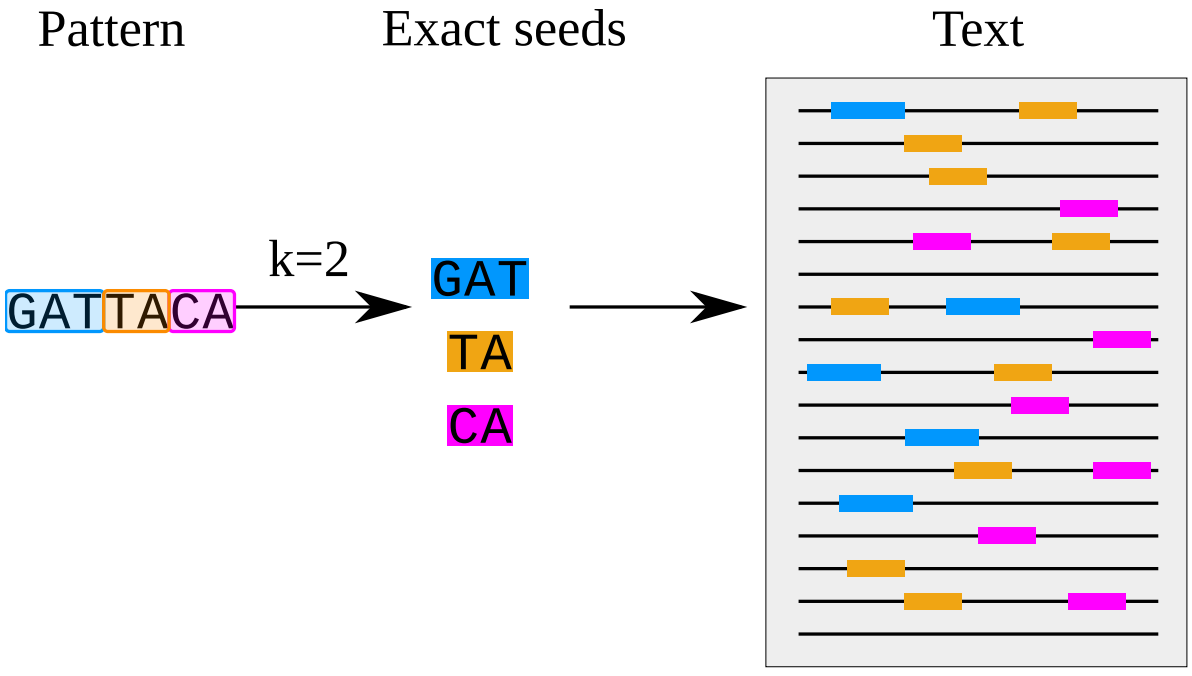
Exact Seed Filtration



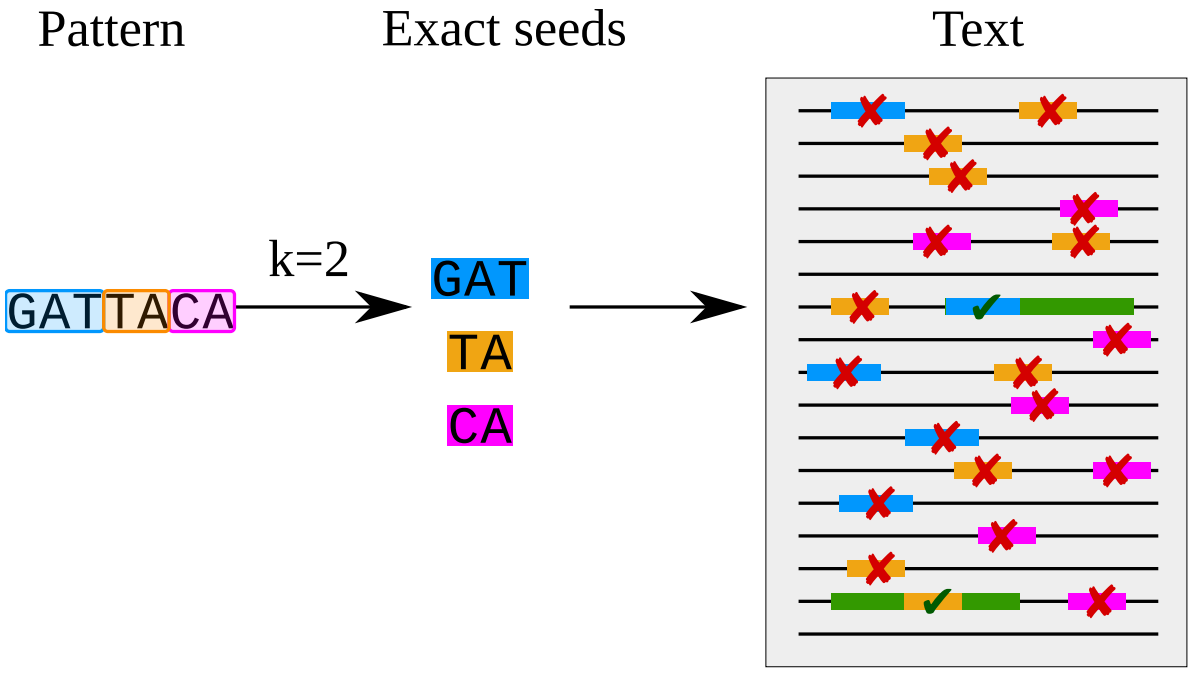
Exact Seed Filtration



Exact Seed Filtration



Exact Seed Filtration

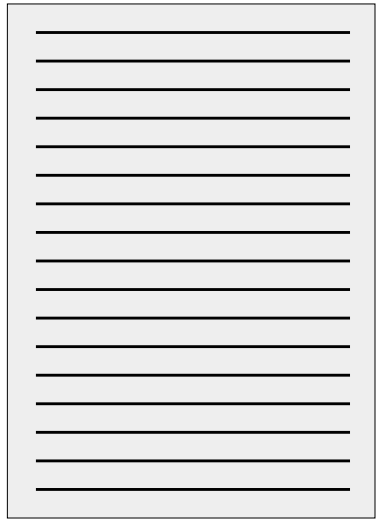


Hybrid Method

Pattern

GATTACA

Text



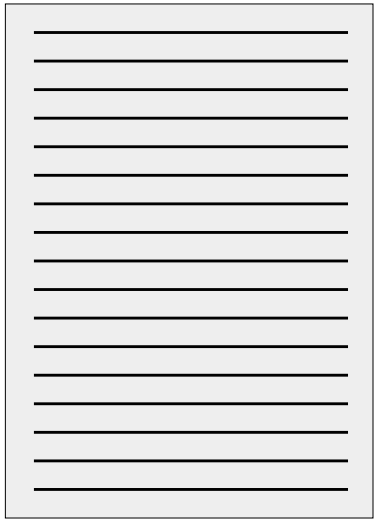
Hybrid Method

Pattern

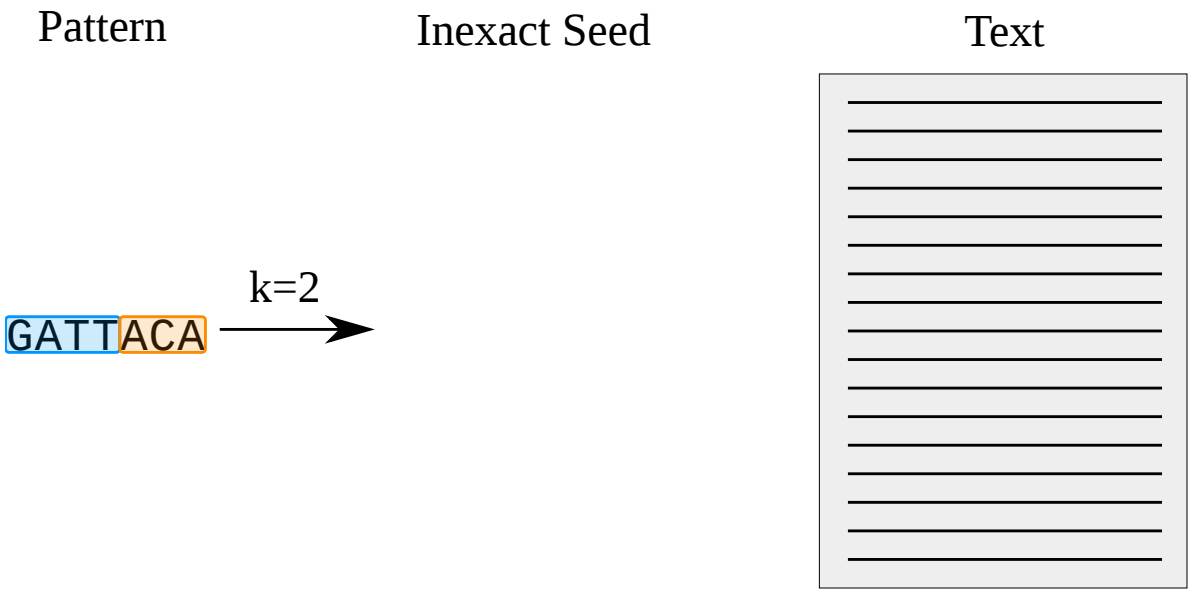
GATTACA

k=2

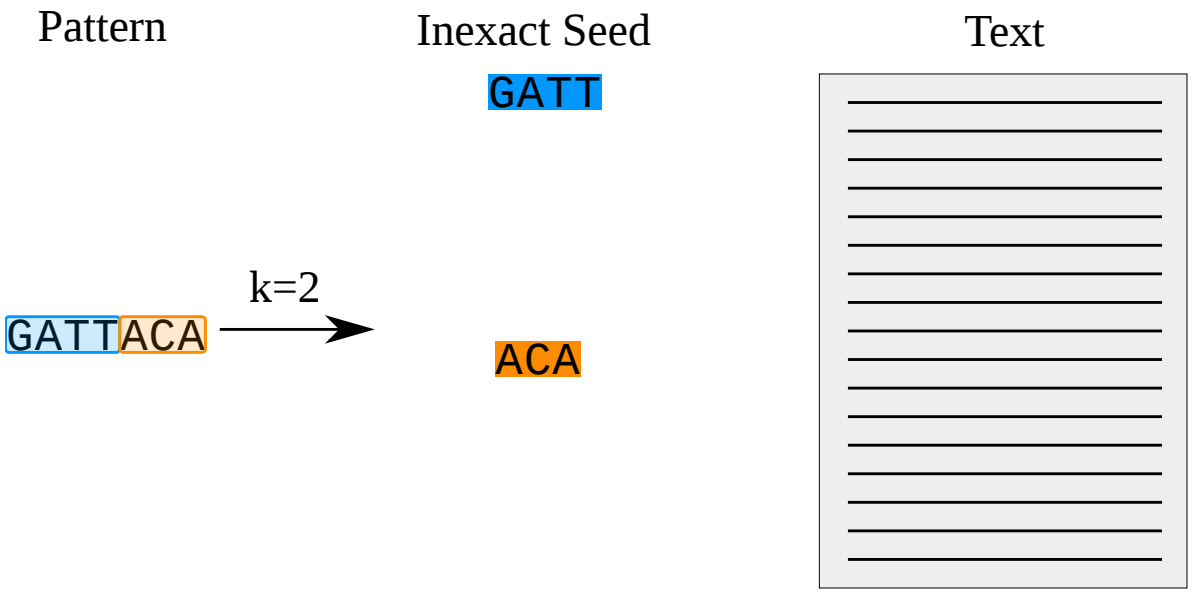
Text



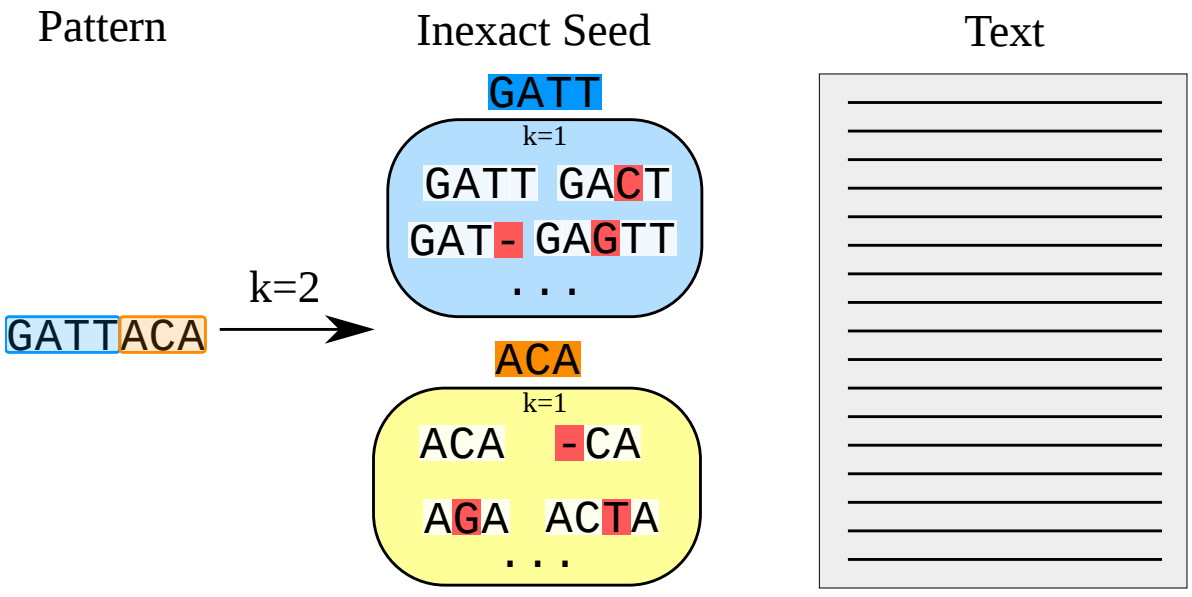
Hybrid Method



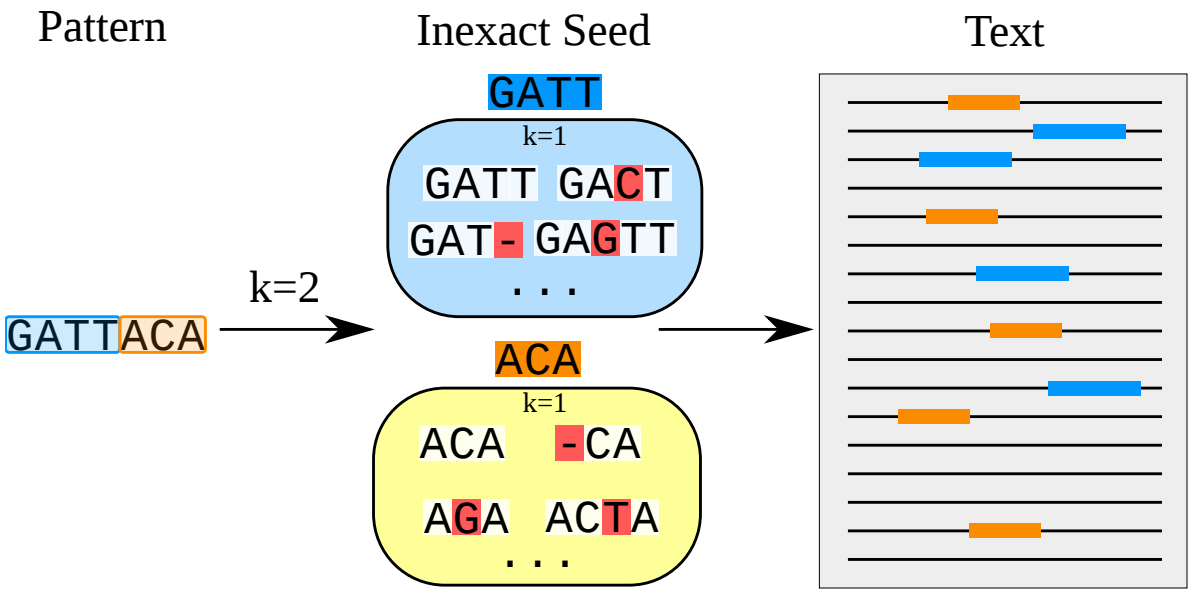
Hybrid Method



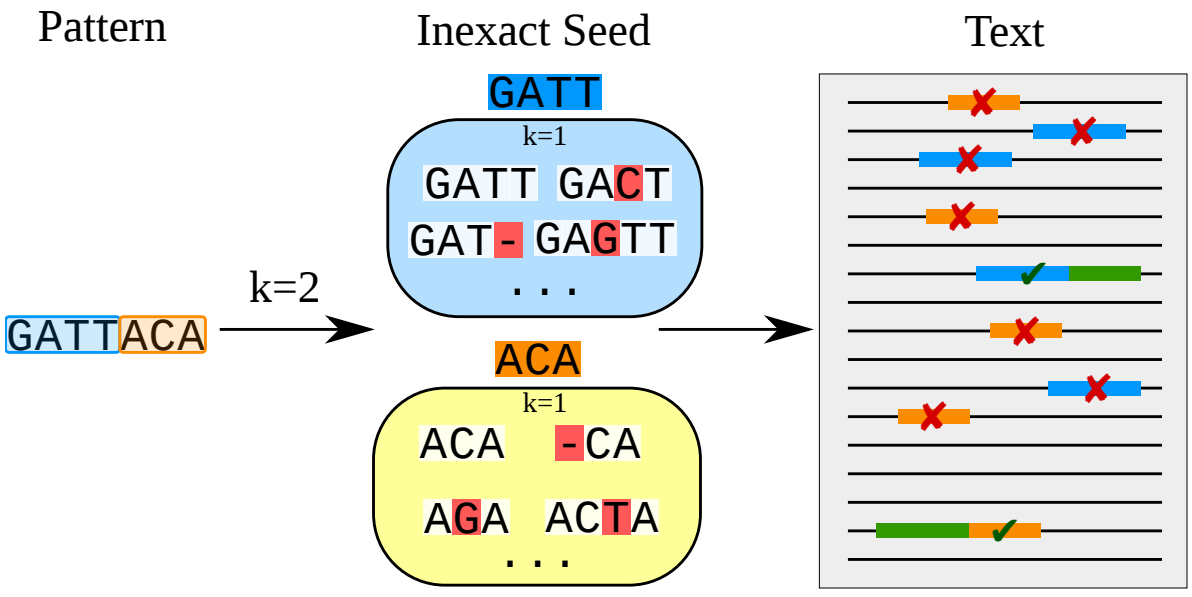
Hybrid Method



Hybrid Method



Hybrid Method



A New Kind of Hybrid Method

A new approximate seed: the 01^*0 seed

Suitable for short patterns and high error rate

$2 \times 4 > 1 \times 5$?

Example: $m = 20$ and $k = 3$

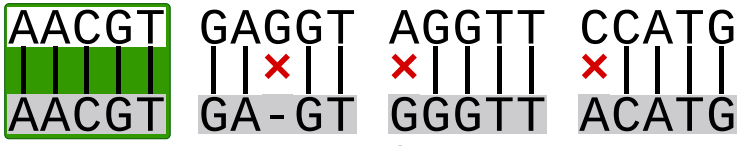
$2 \times 4 > 1 \times 5$?

Example: $m = 20$ and $k = 3$

4 parts of size 5
1 exact part

$2 \times 4 > 1 \times 5$?

Example: $m = 20$ and $k = 3$

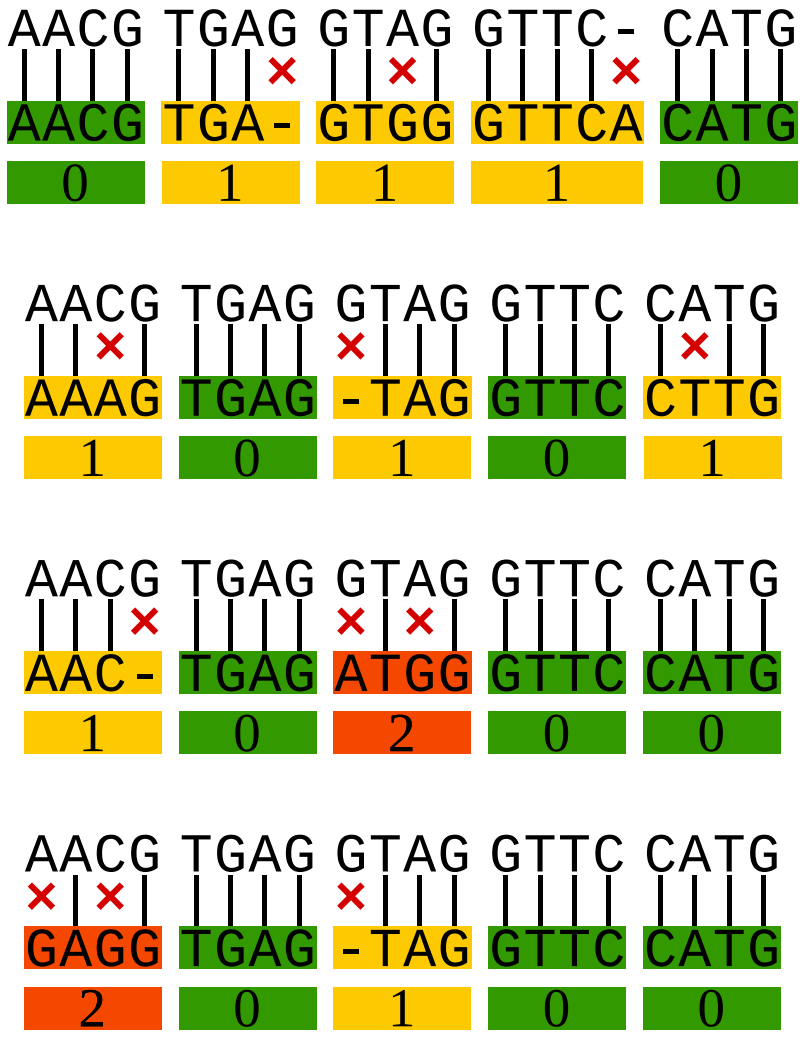


4 parts of size 5
1 exact part

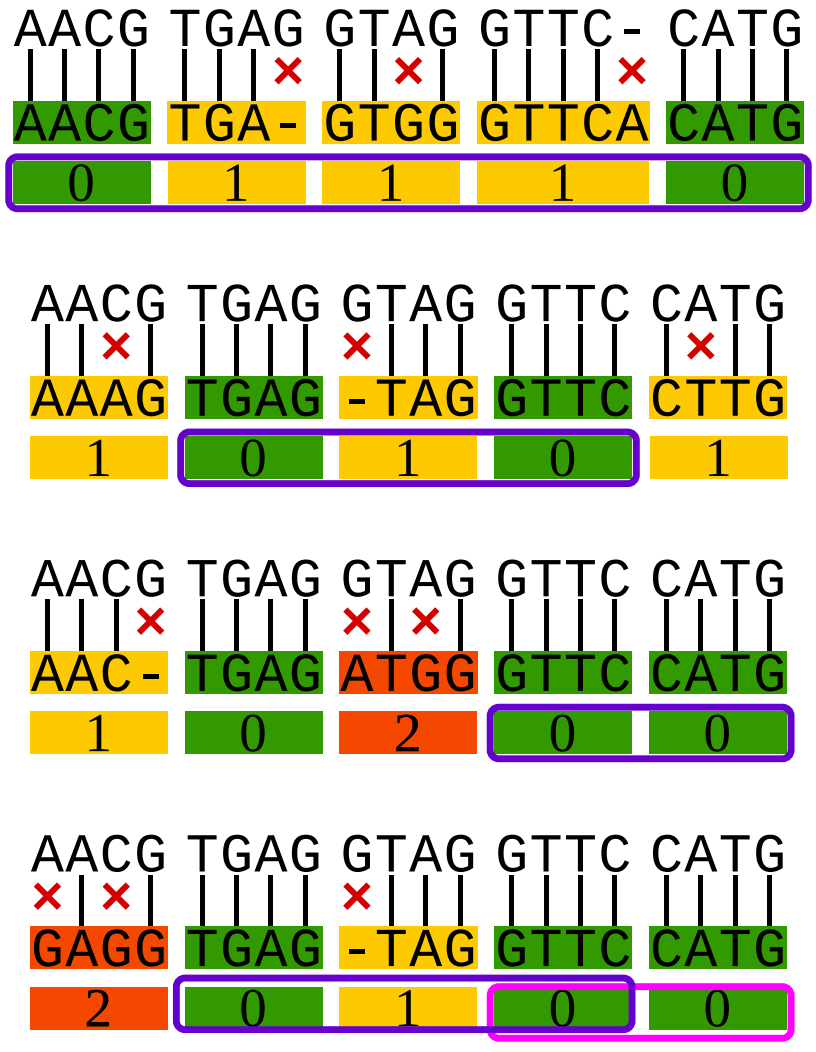
5 parts of size 4
2 exact parts

Distribution of Errors Between Exact Parts

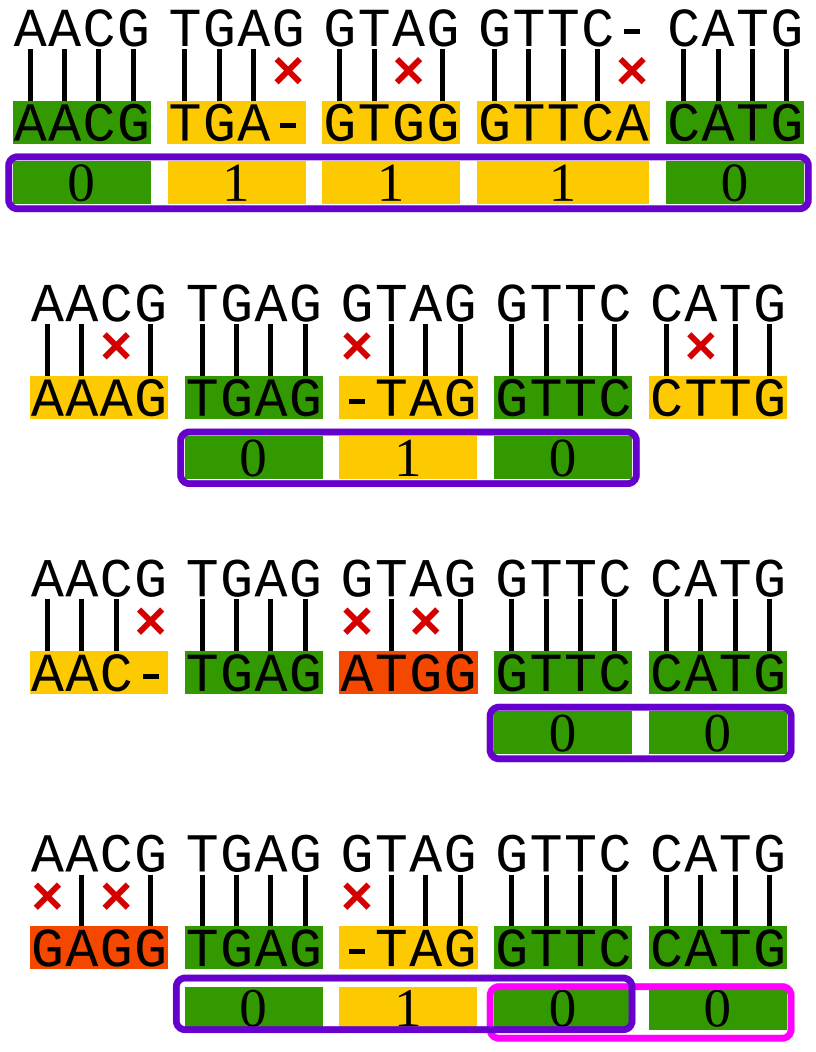
Distribution of Errors Between Exact Parts



Distribution of Errors Between Exact Parts



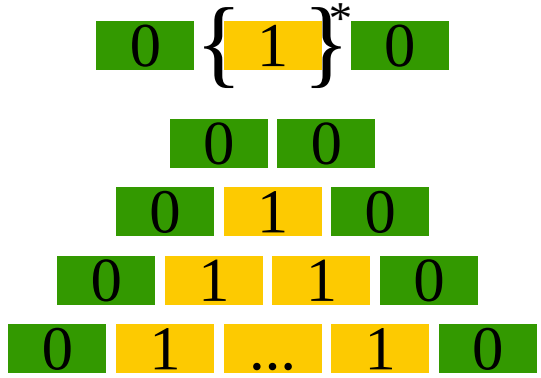
Distribution of Errors Between Exact Parts



the 01*0 Seed

Definition

A 01*0 seed is a pair of exact parts surrounding 0 or more parts with exactly 1 error.

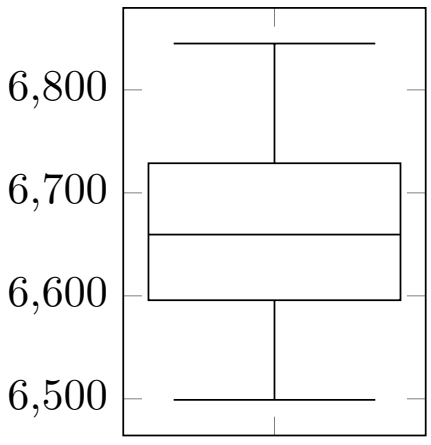


Lemma

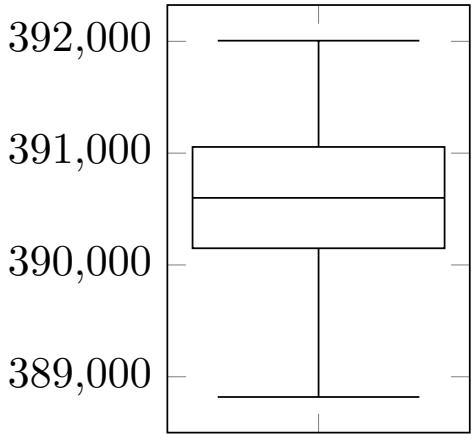
It always exists when the pattern is partitioned in at least $k + 2$ parts.

Good Filtration Efficiency

- larger exact part: 2×4 characters instead of 5 for a classic exact seed filtration
- uniform distribution of errors: 1 error by inner part



01*0 seed



exact seed (5 parts of size 4)

number of candidates per pattern

random DNA strings, 100 patterns, $m = 20$, $k = 3$, $n = 10^8$

Implementation: Bwolo

Indexed text:

- many patterns
- FM-Index

Seeds are generated and searched at the same time in the index

Extension:

1. in the index with a controlled number of errors
2. in the on-line text

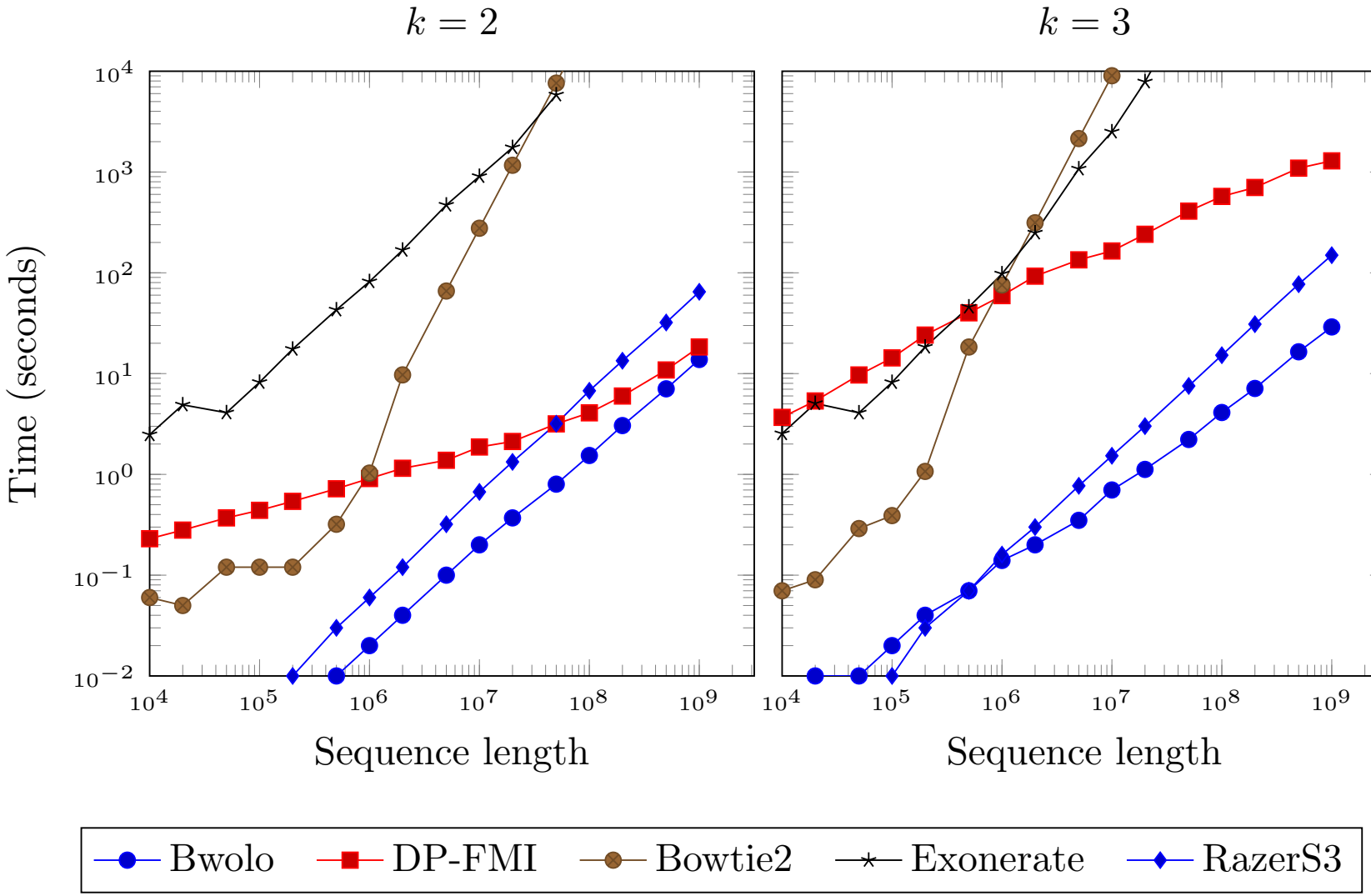
Technically:

- C++
- Seqan library (`www.seqan.de`)

Results – tools

Tool	Index	Method	Detail
Bwolo	FM-Index	Hybrid	01^*0 seeds and extension in index, verification in on-line text
Exonerate	on-line	Neighborhood	Dynamic programming
RazerS3	on-line	Exact Seed	Bit parallel algorithm for the verification step
DP-FMI	FM-Index	Neighborhood	Dynamic programming, in-house implementation
Bowtie2	FM-Index	Exact Seed	Seed filtration in index, verification in on-line text

Bernoulli Sequences



Alphabet: DNA
Text: 10^4 – 10^9 characters

100 patterns of length 20

Simulated Reads on Human Genome

	10,000 Reads		10^7 Reads	
	Time (s)	Memory(Mo)	Time (s)	Memory(Mo)
Bwolo	97	6,522	55,493	9,054
RazerS3	502	6,469	467,413	152,045
Bowtie2	156,164	8,260	-	-

From Schbath *et al.* Journal of Computational Biology (2012).

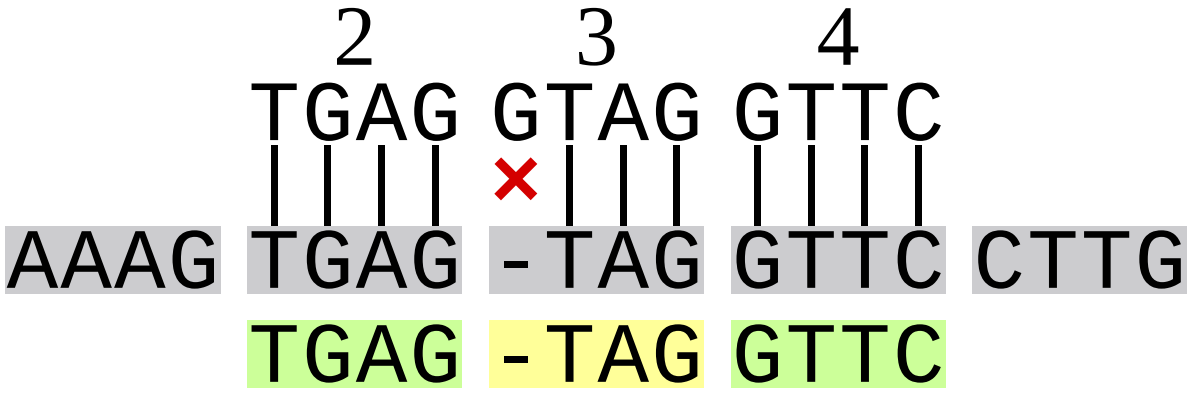
- $n = 2.7 \times 10^9$
- $m = 40$, extracted from the text
- $k = 3$

Conclusion and Perspectives

- New elegant and efficient seeding approach: trade-off between filtration and search time
- Application to approximate pattern matching: `http://bioinfo.lifl.fr/bwolo`
- Average-case time complexity analysis ?
- Many possible evolutions:
  - application in bioinformatics
  - indexed patterns
  - very suitable for bit parallel algorithms
  - homology search, alignment

Merci

Extension of the Seed on One Side

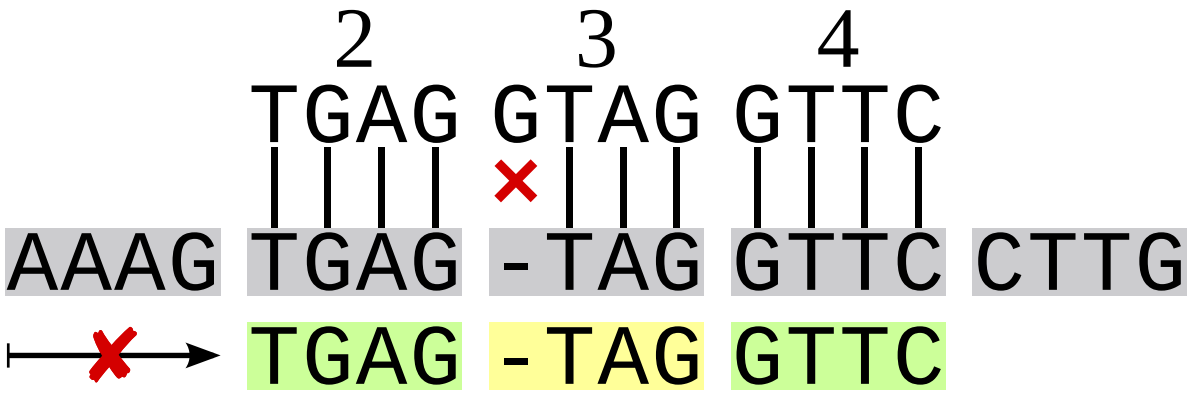


Finding no sub-seed before part 2 means there are errors in previous parts.

Extension on one side (either left or right of the 01^*0 seed) needs only as much error as there are parts on that side.

Extension on one side may be done before verification step.

Extension of the Seed on One Side

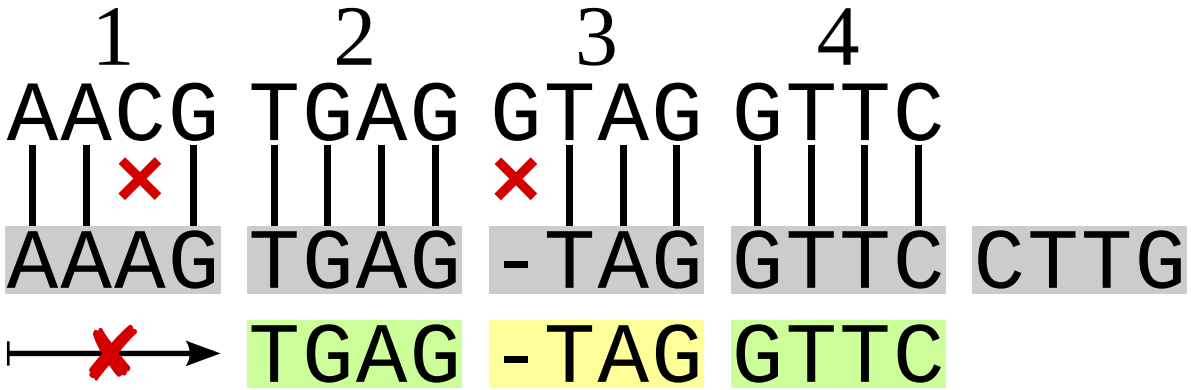


Finding no sub-seed before part 2 means there are errors in previous parts.

Extension on one side (either left or right of the 01*0 seed) needs only as much error as there are parts on that side.

Extension on one side may be done before verification step.

Extension of the Seed on One Side

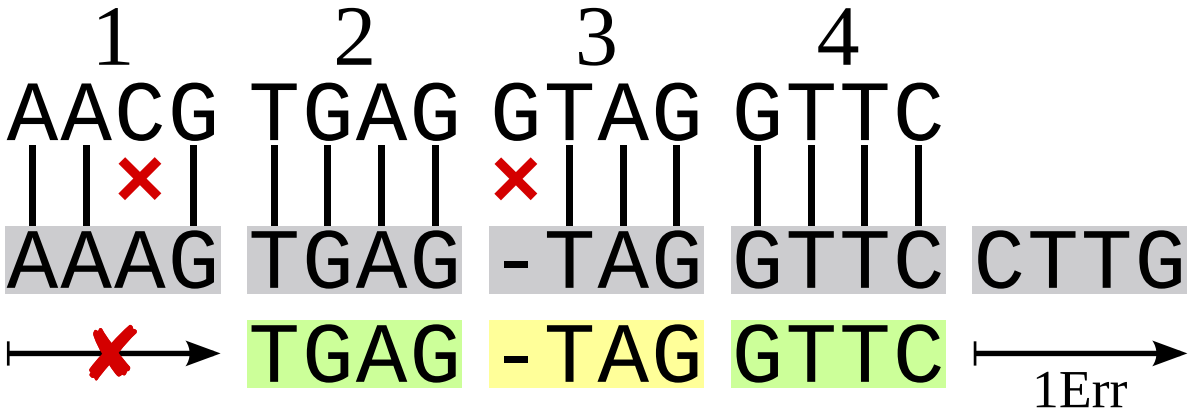


Finding no sub-seed before part 2 means there are errors in previous parts.

Extension on one side (either left or right of the 01^*0 seed) needs only as much error as there are parts on that side.

Extension on one side may be done before verification step.

Extension of the Seed on One Side

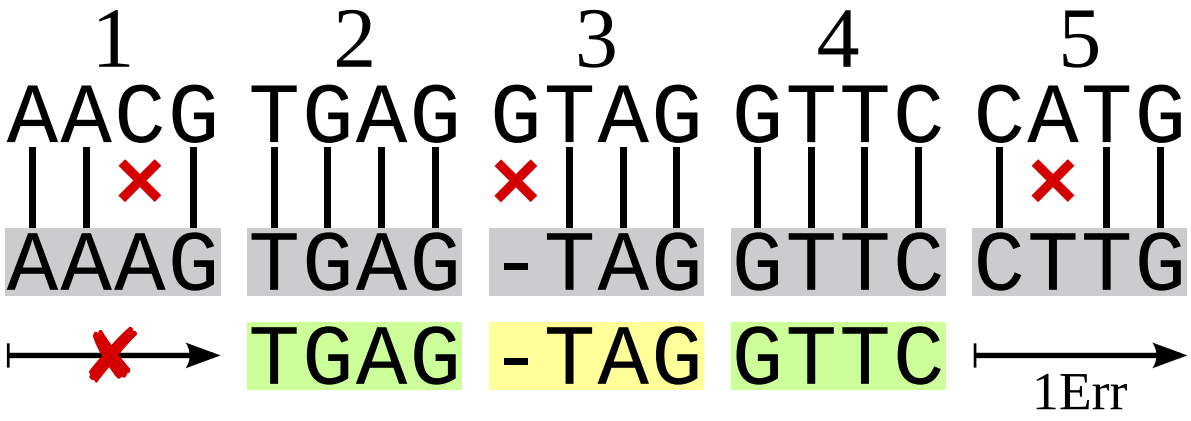


Finding no sub-seed before part 2 means there are errors in previous parts.

Extension on one side (either left or right of the 01^*0 seed) needs only as much error as there are parts on that side.

Extension on one side may be done before verification step.

Extension of the Seed on One Side



Finding no sub-seed before part 2 means there are errors in previous parts.

Extension on one side (either left or right of the 01*0 seed) needs only as much error as there are parts on that side.

Extension on one side may be done before verification step.