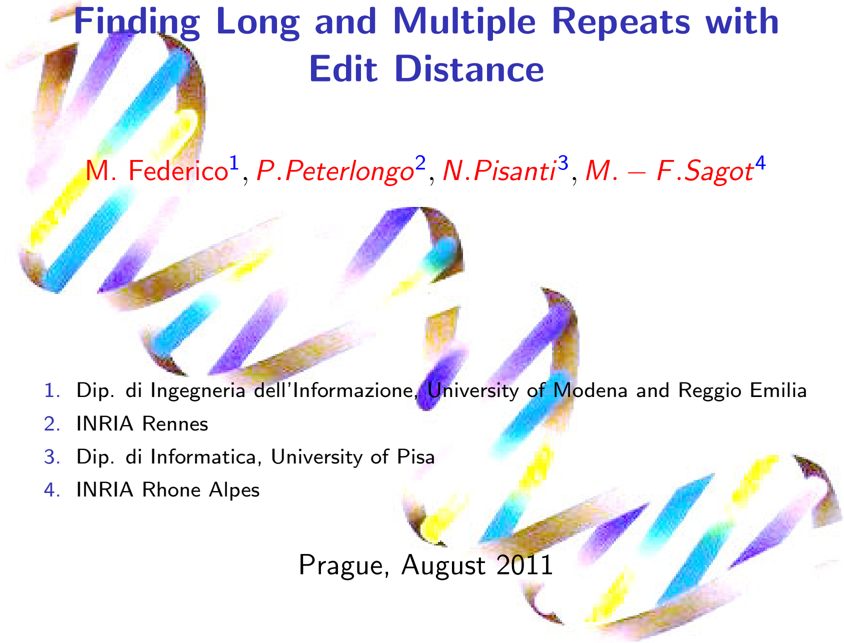
Finding Long and Multiple Repeats with Edit Distance



M. Federico[1], P.Peterlongo[2], N.Pisanti[3], M. − F.Sagot[4]

1. Dip. di Ingegneria dell'Informazione, University of Modena and Reggio Emilia
2. INRIA Rennes
3. Dip. di Informatica, University of Pisa
4. INRIA Rhone Alpes

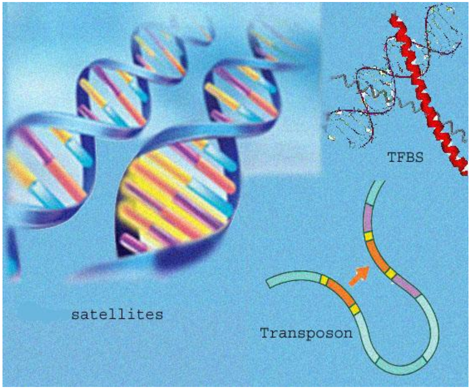
Prague, August 2011

Biological Problem

- Lots of data: seeking information for comparisons and annotation.
- Analysis of biological sequences in order to find regularities such as repeats.

Repeat size changes according to the type of signal:

- TFBS (5 - 25 bases)
- Satellites (>100 bases)
- Transposable elements, LTRs (1.5 - 10 Kbases and hundreds, resp.)
- Homolog genes (from hundreds to thousands bases)



Algorithmical Problem we want to address

Input: one or more genomes/chromosomes/DNA fragments

Output: repeats that are

- long: > 100 bp
- multiple: number of occurrences ≥ 2
- approximate: each pair of occurrences may show *substitutions*, *insertions* or *deletions* in up to 10 to 15% of their length

Motivations

Multiple local alignment is a computationally expensive task

- Dynamic Programming complexity is exponential with the number of input sequences
- Pairwise DP or DP of a sequence against itself (for finding repetitions within a single sequence) are not practical for input size as big as whole genomes
- Heuristics exist, but there is no guarantee of complete results

Filters

Tools that quickly discard fragments of sequences that are guaranteed not to contain any occurrence of a repeat, as they do not fulfill a necessary condition

The necessary condition must be:

- as strong as possible
- fast to check

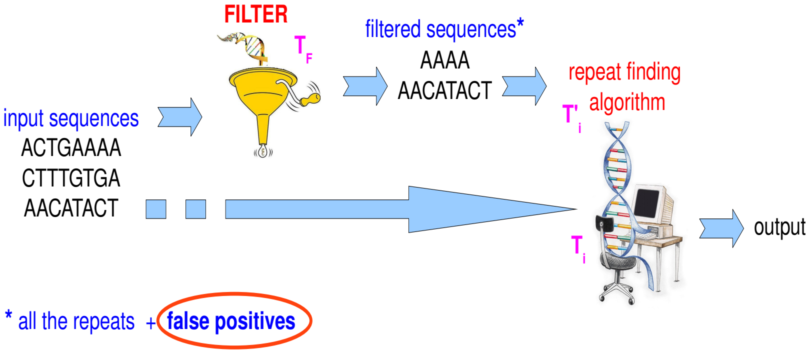
Filters

Tools that quickly discard fragments of sequences that are guaranteed not to contain any occurrence of a repeat, as they do not fulfill a necessary condition

Lossy	vs	**Lossless**
inexact		**exact**

Lossless Filters

lossless filters allow biologists to have exact results in reasonable time



$$T_F + T'_i \ll T_i$$

Lossless Filters for multiple repetitions: state of the art

- Hamming distance NIMBUS [1] [2]
- Edit distance **TUIUIU** [3] [4]

[1] P.Peterlongo, N.Pisanti, F.Boyer, M.-F. Sagot, SPIRE 2005.
[2] P.Peterlongo, N.Pisanti, F.Boyer, A.Pereira do Lago, M.-F. Sagot, Journal of Discrete Algorithms 2008.
[3] P.Peterlongo, G.A.T.Sacomoto, A.Pereira do Lago, N.Pisanti, M.-F.Sagot, BMC Algorithms for Molecular Biology 2009.
[4] M.Federico, N.Pisanti, P.Peterlongo: AICCSA 2010.

Basic idea of filters with edit distance

If two L long sequences are *identical* then they share exactly $L - q + 1$ q-**grams**.

ATTAAAATTT
ATTAAAATTT

Basic idea of filters with edit distance

If two L long sequences are *identical* then they share exactly $L - q + 1$ ***q*-grams**.

ATTAAAATTT
ATTAAAATTT

e.g. L=10 and q=2, the 9 q-grams are AT, TA, AA, AA, AT, TA, AT, TT, TT.

Basic idea of filters with edit distance

If two sequences are *similar* then they must still share **at least a certain number** of *q*-**grams**.

ATTAAAATTT
ATAAATATTT

Basic idea of filters with edit distance

If two sequences are *similar* then they must still share **at least a certain number** of *q*-**grams**.

ATTAAAATTT
ATAAATATTT

If two sequences do not share enough exactly conserved parts, then they cannot be similar

TUIUIU: target

Given:

- $L > 0$: lenght of repeats
- $r \geq 2$: minimum number of repeat occurrences
- $0 \leq d < L$: maximum number of insertions, substitutions, deletions between each pair of repeat occurrences
- S: a set of one or more input sequences

TUIUIU keeps fragments from S that may be part of an (L, r, d)-Erepeat:

a set $\{w_1, \ldots, w_r\}$ of r *pairwise non overlapping* words of length in range $[L - d, L + d]$ such that $d_E(w_i, w_j) \leq d$

$d_E(w_i, w_j) =$ edit distance between w_i and w_j

TUIUIU: necessary condition

- Based on the minimal number of portions of fixed length q shared by the occurrences of a repeat (q-grams)

```
ATTAAAATTT
ATAAATATTT
```

TUIUIU: necessary condition

- Based on the minimal number of portions of fixed length q shared by the occurrences of a repeat (q-grams)

```
ATTAAAATTT
ATAAATATTT
```

- GOOD

  the minimum number of q-grams that occurrences of a repeat must share is

$$\mathbf{p = L - q + 1 - qd}$$

  first introduced by E.Ukkonen in *TCS*, 1992 for different purposes; also used by SWIFT

  + "parallelogram condition" also already used by SWIFT

  + "vertical projection"

TUIUIU: necessary condition

- Based on the minimal number of portions of fixed length q shared by the occurrences of a repeat (q-grams)

```
ATTAAAATTT
ATAAATATTT
```

TUIUIU: necessary condition

- Based on the minimal number of portions of fixed length q shared by the occurrences of a repeat (q-grams)

ATTAAAATTT
ATAAATATTT

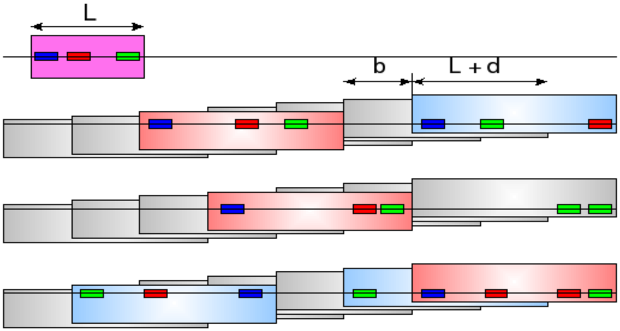
- EXCELLENT

  the at least p q-grams shared by occurrences w and w' of a repeat must be in the **same order** in w and w'

  $+$ "horizontal projection"

TUIUIU

Input sequences are divided into blocks ($L + b + d < 2L$, $b =$ the smallest power of 2 greater than d. Among blocks containing enough shared q-grams, it counts those that are in the same order



$r = 3$ and minimum number of shared q-grams is $p = 3$

What's left after TUIUIU

False Positives (fragments kept by the filter while not being part of any repeat):

What's left after TUIUIU

False Positives (fragments kept by the filter while not being part of any repeat):

- FP_{rect} due to check the condition for window of size L against blocks of size almost $2L$

What's left after TUIUIU

False Positives (fragments kept by the filter while not being part of any repeat):

- FP_{rect} due to check the condition for window of size L against blocks of size almost $2L$
- FP_{cond} due to the fact that the condition the filter checks is only a necessary condition, but not sufficient

What's left after TUIUIU

False Positives (fragments kept by the filter while not being part of any repeat):

- FP_{rect} due to check the condition for window of size L against blocks of size almost $2L$
- FP_{cond} due to the fact that the condition the filter checks is only a necessary condition, but not sufficient
- FP^* due to check the condition between a window and blocks (*star* shape) rather than all windows against all pairwise (*clique* shape)

What's left after TUIUIU

False Positives (fragments kept by the filter while not being part of any repeat):

- FP_{rect} due to check the condition for window of size L against blocks of size almost $2L$
- FP_{cond} due to the fact that the condition the filter checks is only a necessary condition, but not sufficient
- FP^{*} due to check the condition between a window and blocks (*star* shape) rather than all windows against all pairwise (*clique* shape)

FP_{rect} and FP_{cond} are totally removed by an additional step of local alignment between windows and blocks.

What's left after TUIUIU

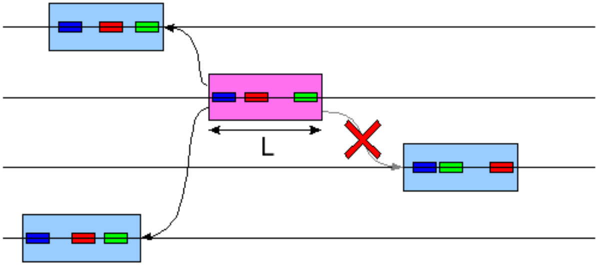
Types of False Positives (fragments of sequences kept by the filter while not being members of a searched repeat):

- FP_{rect} due to check the condition for window of size L against blocks of size almost $2L$
- FP_{cond} due to the fact that the condition the filter checks is only a necessary condition, but not sufficient
- FP^* due to check the condition between a window and blocks (*star* shape) rather than all windows against all pairwise (*clique* shape)

How to reduce/eliminate the FP^*??

TUIUIU: how it filters

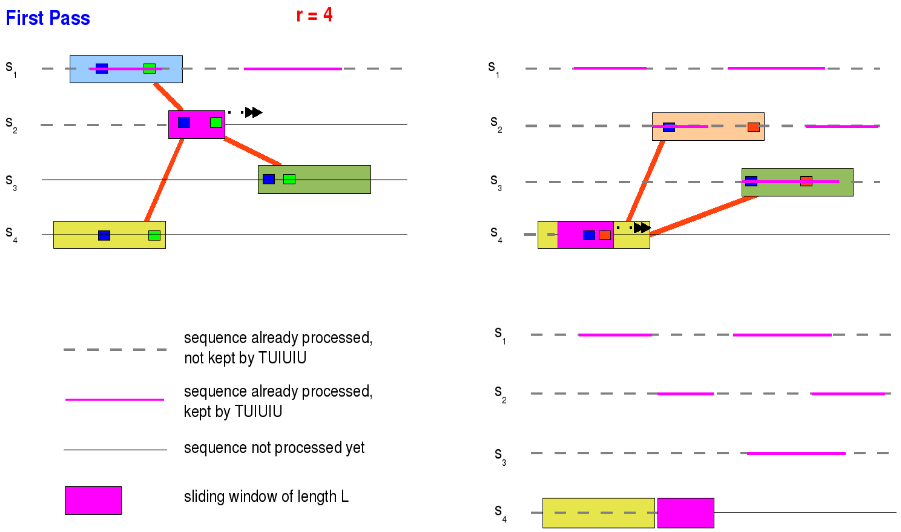
- star approach vs clique approach:
  TUIUIU keeps fragments that satisfy the necessary condition with $r - 1$ other fragments, without checking the condition between these $r - 1$ other fragments



- $p = 3$ (minimum number of shared q-grams in the same order)

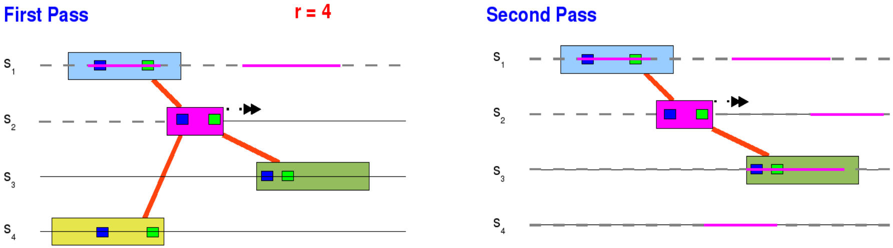
1) Reducing *FP**: Double Pass Strategy

- Motivation:


First Pass
r = 4
s_1
s_2
s_3
s_4
sequence already processed, not kept by TUIUIU
sequence already processed, kept by TUIUIU
sequence not processed yet
sliding window of length L

1) Reducing *FP**: Double Pass Strategy

- Solution:
  1. run TUIUIU once on the input sequences
  2. run TUIUIU once again on the filtered sequences (faster)

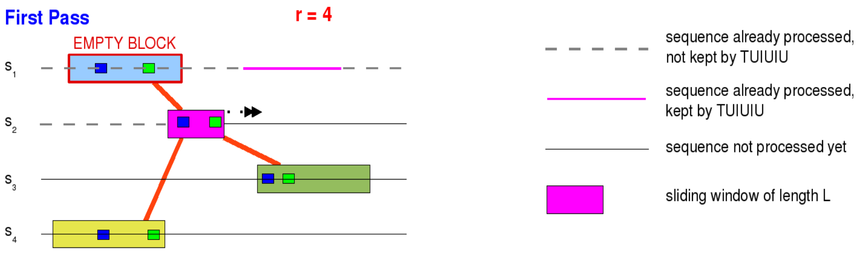


During the second pass:

- only fragments of kept sequences are considered
- only blocks containing kept fragments are tested while checking the necessary condition

2) Reducing FP^*: Empty Block Strategy

- Motivation:

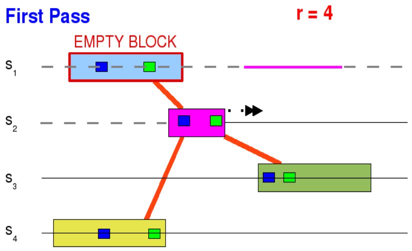

First Pass
r = 4
EMPTY BLOCK
s1
s2
s3
s4
sequence already processed, not kept by TUIUIU
sequence already processed, kept by TUIUIU
sequence not processed yet
sliding window of length L

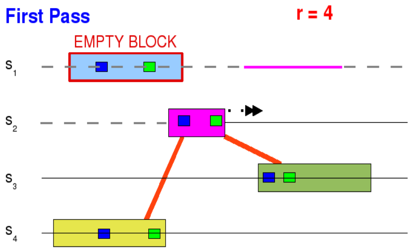
Necessary condition check:

- ALL possible blocks of all sequences are taken into account, including *blocks of already filtered sequences that may not contain any kept fragment*: empty blocks

2) Reducing *FP**: Empty Block Strategy

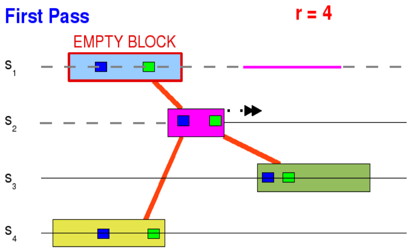
- Solution:
  - double pass with extra time cost (even if negligible)
  - detect empty blocks *on the fly* during the first pass (reduce the search space and speed up the computation)

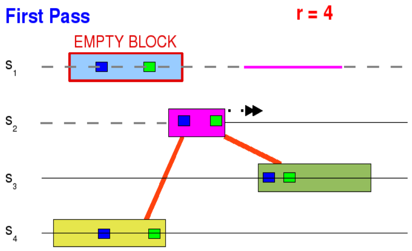




2) Reducing *FP**: Empty Block Strategy

- Solution:
  - double pass with extra time cost (even if negligible)
  - detect empty blocks *on the fly* during the first pass (reduce the search space and speed up the computation)





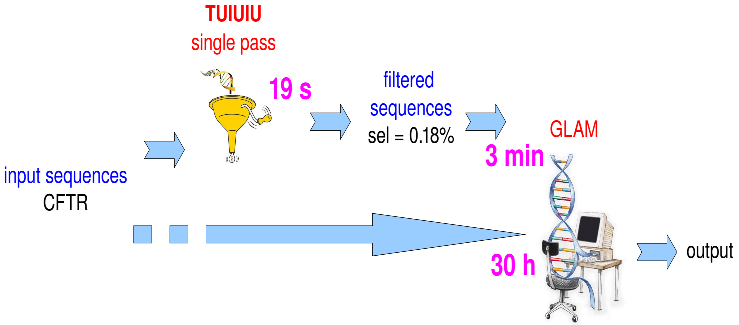
- double pass and empty block can co-exist

Speedup of local alignment algorithms

Pre-processing of input sequences to speed up GLAM2[1]

- 5 orthologous regions cystic fibrosis transmembrane conductance regulator gene (humans)
- Size: 5518041 bases
- Parameters: L=100, r=5, d=7, q=11

- Intel(R) Quad-core Xeon(R) E5405/2GHz
- 10GB RAM

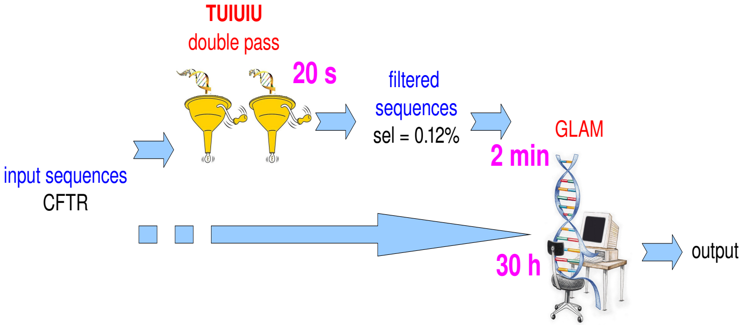


Speedup of local alignment algorithms

Pre-processing of input sequences to speed up GLAM2[1]

- 5 orthologous regions cystic fibrosis transmembrane conductance regulator gene (humans)
- Size: 5518041 bases
- Parameters: L=100, r=5, d=7, q=11

- Intel(R) Quad-core Xeon(R) E5405/2GHz
- 10GB RAM



FILMRED

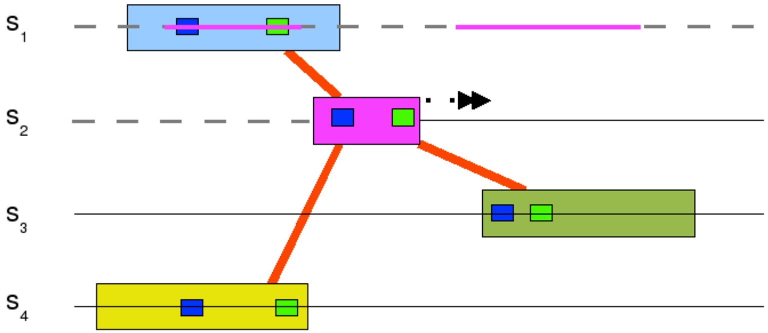
We have a fast and accurate (no false negatives and very few false positives) filter for finding multiple repeats with pairwise limited edit distance

FILMRED is a tool to detect long repeats:

- that uses the filter as a preprocessing step
- that uses the information collected by the filter to speed up the actual alignment step
- with a biologist-friendly visualization of results

Using data from the filter

Not just pre-processing of input sequences to speed up Multiple Local Alignement.



Sketch of a possible pipeline

- Running TUIUIU
- Align each kept window against all its 'friends' blocks (all FP_{rect} and FP_{cond} are removed).
  again EBS here + some more tricks to speed up
- Clique detection among blocks (many FP* are removed).
- Actual alignment of what is left and output.

The output is... a redundant set of redundant cliques!

Clique detection

- Consider a graph with a node per each block containing a kept window, and en edge connecting two blocks if they contain windows that are 'friends'.
- Search for maximal cliques in this graph using Bron-Kerbosch's algorithm using the vertex with largest degree as pivot.
- Keep cliques of size $k \geq r$: they are repetitions occurring k times!
- This step is (surprisingly?) fast.

But indeed we get a redundant set of redundant cliques

Some first tests of our MLA

Dataset is five ortholog sequences of total size $\approx 5.5Mb$.

First test: $L = 100, d = 5, r = 3$

Initially there are $\approx 5.5Mb$ windows
Time filter (four passes): 1021.03s to keep 4659 windows
Time alignment: 4.16s to keep 2306 windows
Time clique check: 0.03s to keep 202 cliques (actually ≈ 20 repetitions)
Time multiple alignment: 0.66 s
TOTAL TIME: 1025.88s (about 17')

A redundant clique

```
—
clique found! 169071, 169070, 95585, 95584, 95583
—
```

Five occurrences that are actually two!

Possible reasons are

- Tandem repeats (unlikely).
- The same window can be contained in two consecutive blocks.
- Parameter d is too large: actual repetition is more conserved and allows shifts.

A redundant amount of cliques

—

```
clique found! 169071, 169070, 95585, 95584, 95583,
clique found! 169071, 169070, 169069, 95584, 95583,
clique found! 169070, 169069, 169068, 95583,
clique found! 169072, 169071, 95586, 95585, 95584,
clique found! 169072, 169071, 169070, 95585, 95584,
clique found! 169073, 169072, 95587, 95586, 95585,
clique found! 169073, 169072, 169071, 95586, 95585,
clique found! 169074, 169073, 95588, 95587, 95586,
clique found! 169074, 169073, 169072, 95587, 95586,
clique found! 169075, 169074, 95589, 95588, 95587,
clique found! 169075, 169074, 169073, 95588, 95587,
...
```

—

Many cliques for actually a single repetition: the length of the actual unique repetitions was underestimated: parameter L was too small.

Redundancy removal

- The two clique redundancies type can co-exist.
- Redundancy occurs when parameters are not accurate: remove redundancy means performing an automatic parameters tuning
- Remove redundancy by blocks merging on the fly, leading to cliques made of enlarged non-overlapping blocks.

Some experiments: timing

[Intel(R) Quad-core Xeon(R) E5405/2GHz with 10 GB of RAM]

Performances of the different phases of FilmRed to find (L, r, d)-*Erepeats* on the CFTR dataset (five ortholog regions of the cystic fibrosis transmembrane conductance regulator gene in chicken, cow, human, mouse and tetra for a total of 5518041 bases), with $L = 100$ and $r = 5$, and $d = 7, 12, 14, 15$.

	Filter		Semiglobal Align		Clique detection		Total
d	time(s)	sel	time(s)	sel	time	#cliques	time(s)
7	64.20	0.05%	56.56	0	-	-	120.76
12	1017.51	0.01%	0.88	0	-	-	1018.39
14	3772.65	0.02%	1.41	0.001%	0.00	1	3774.06
15	7128.19	0.65%	740.01	0.003%	0.01	1	7868.21

Finding LTRs, an interesting application

Performances of the different phases of FILMRED with $r = 3$ on a data set of size 26392324b of the mobilome of three *S. Cerevisiae* genomes.

		Filter		Semiglobal Align		Clique detection		Total
L	d	time(s)	sel	time(s)	sel	time	#cliques	time(s)
200	20	29.44	0.17%	744.48	0.09%	6.30	24	780.22
300	30	31.68	0.16%	1473.65	0.07%	2.13	13	1507.46
5000	500	9.00	0	-	-	-	-	9.00

By using the annotation available for one of the three yeasts, we found that all detected repetitions were either real LTR or are part of a retrotransposon.

Searching LTRs in Sunflower

We compared repeats found by FilmRed in the Sunflower with the ones found by the signature-based repeat finding tool LTR_Finder.
We observed that all the repeats identified by the other tool are found also by FilmRed. The latter, however, returns also further repeats, which are not identified by the former.

		Filter		Semiglobal Align		Clique detection		Total
L	d	time(s)	sel	time(s)	sel	time	#cliques	time(s)
200	20	0.44	3.32%	3.38	1.10%	0.00	3	3.82
300	30	0.46	3.42%	7.36	0.98%	0.00	2	7.82
200	25	0.59	5.66%	4.24	2.57%	0.00	3	4.83
300	45	178.25	41.70%	35.59	3.15%	0.00	2	213.84

Future work

- A filter (also) based on (maximal) longer seeds with few errors
- k-mers statistics taking into account also their relative distances (and not just the order.