Extracting Approximate Patterns

Johann Pelfrêne[1,2], Saïd Abdeddaïm[1,3] and Joël Alexandre[1]
johann.pelfrene@univ-rouen.fr

http://johann.jalix.org/research/

[1] ABISS - University of Rouen, France

[2] Exonhit Therapeutics, Paris, France

[3] LIFAR - University of Rouen, France

Introduction

- Collaboration bioinformatics lab and bio-company
- Huge amount of data :
  release 75 of embl, may 30th 2003
  16 988 340 sequences
  8 595 205 394 characters
  (human genome: 3 969 969 766)
- Select some sequences corresponding to a query, and align them
- BLAST (human controled), CLUSTAL
- possible to find common patterns ?

Problem

- Given a text t, of length n
- and a quorum q
  - minimal number of occurrence of a pattern
  - q-patterns: patterns occurring at least q times
- Index the "most interesting" patterns such that:
  - just a few patterns
  - loss of few information
  - patterns form a basis for the others
- Focus on patterns with don’t cares (unit length)
- A this time, just extraction of patterns

Related works

- Based on Parida et al. (SODA'00)

  "Pattern discovery on character sets and real-valued data: linear bound on irredundant motifs and an efficient polynomial time algorithm"

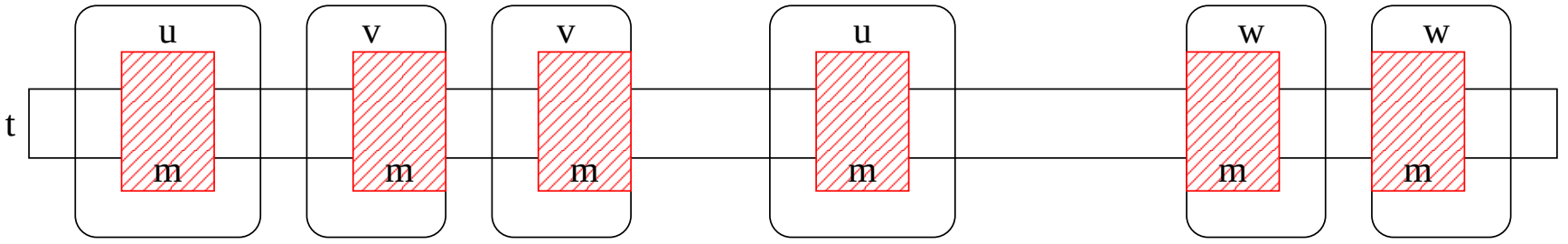
- Apostolico (AI & heuristics methods in bioinformatics, 2002)

  "Pattern discovery and the algorithmics of surprise"

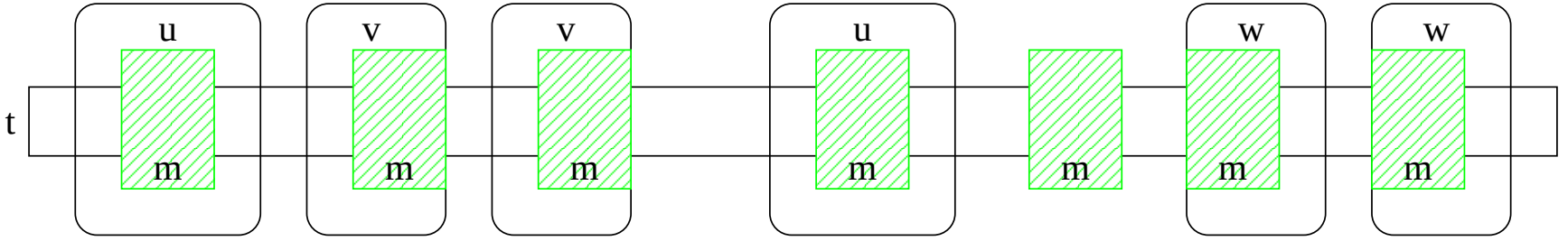
- Pisanti et al. (MFCS'03)

  "A basis of tiling motifs for generating repeated patterns and its complexity for higher quorum"

Primitive patterns



Pattern m is not primitive in S because of patterns u, v and w.



Pattern m is primitive in S because one of its position is not covered.

The algorithm

- Incremental algorithm:
  - Given the primitive q-patterns $\mathcal{P}(t, q)$
  - We compute the primitive $(q+1)$-patterns
- The algorithm consists in three main steps:
  - Degenerate the q-patterns
  - Compute the positions lists
  - Discard the non primitive patterns

Step 1: degenerate patterns

$\textsc{Step1}(t, q, \mathcal{P}(t, q))$

1 **for** $m \in \mathcal{P}(t, q), |pos(m, t)| = q$ **do**
2 **for** $shift \leftarrow 1$ **to** $|t|$ **do**
3 **if** $shift \notin pos(m, t)$ **then**
4 $\textsc{CreatePattern}(m, t, shift)$

Step 1: create patterns

- let $t = AAGTAGATAAATAA$, $q = 2$
- $m = AA\_TA$ is a primitive 2-pattern
- $pos(m, AAGTAGATAAATAA) = \{1, 9\}$
- with $shift = 10$, we obtain $m' = AA\_\_A$

```
AAGTAGATAAATAA
          AA_TA
          AA__A
```

- degenerating 1 pattern is $O(n^2)$
- for all the b q-patterns, $O(bn^2)$

Step 2: compute the positions

- Fisher and Paterson (74)
  "String matching and other products"
- for each letter of the alphabet:
  encode string into integer
  make an integer product (FFT)
- computes the Hamming score between the pattern and each factor of the text
- for 1 pattern, time complexity $O(n \log^2 n \log \log n)$
- step 1 gives bn patterns: $O(bn^2 \log^2 n \log \log n)$

Step 2: example

- pattern $m = AA\_\_A = A_1A_2\_\_A_3$

  A_1 AAGTAGATAAATAA

  A_2 AAGTAGATAAATAA

  A_3 AAGTAGATAAATAA

- encoded as integers:

  11001010111011

  $+$ 11001010111011

  $+$ 11001010111011

  $=$ 11013111222133 1121

  AAGTAGATAAATAA

- $11013111222133 1121 = 11001010111011 \times 10011$

Step 3: Test the primitivity

- Given a text t, a pattern m and q
  decide if m is primitive or not
- $t = AAGTAGATAAATAA$, $m = AA\_\_A$, $q = 2$

```
              AAGTAGATAAATAA 1
   AAGTAGATAAATAA            9
AAGTAGATAAATAA              10
12101111133113110111011 A
00011001100120001000100 T
00110110000100100000000 G
```

Time complexity

- step 1 generate bn patterns in time $O(bn^2)$
- step 2 is $O(bn^2 \log^2 n \log\log n)$
- step 3 is $O(bn^2 \log^2 n \log\log n)$
- in the worst case, $O(bn^2 \log^2 \log\log n)$

Number of primitive patterns

- bounded by C_{n-1}^{q-1} ...
- for $q = 2$, at most $n - 1$ primitive patterns
- for $q = 1$, only t is primitive
  initialization of the algorithm

Complete example

- $t = AAGTAGATAAATAA$, $q = 2$:
  $A\_\_\_\_\_AA\_\_A\{1,2\}$
  $A\_\_TA\_ATAA\{1,5\}$, $A\_\_A\_\_\_A\{2,7\}$
  $A\_A\_A\_A\{5,7\}$, $AG\_\_\_A\_\_AA\{2,5\}$
  $AA\_TA\{1,9\}$
- and $q = 3$:
  $A\_\_\_\_\_AA\{1,2,5\}$
  $A\_\_TA\{1,5,9\}$, $A\_\_A\{2,7,10,11\}$
  $A\_\_\_A\_A\{1,5,7\}$, $A\_\_\_\_A\{2,5,9\}$
  $A\_A\_A\{5,7,9\}$, $AA\_\_A\{1,9,10\}$

Conclusion

- Define the primitive patterns
- Give an algorithm for testing the primitivity
- Give an incremental algorithm in $O(bn^2 \log^2 n \log\log n)$
- Bound b by C_{n-1}^{q-1}

Future works

- On biological examples (ESTs), many (most) unuseful patterns
- like for $t = AAGTAGATAAATAA$, $q = 2$
  $m = A\_\_\_\_\_AA\_\_A$
- retain only $O(n)$ at each step ?