An Efficient Exact Solution to the (l,d) Planted Motif Problem

Maria Clara Isabel Sia*
Julieta Nabos
Proceso Fernandez

November 16, 2015

Introduction

The (l, d) planted motif problem

Find a motif of length l=8 across these DNA sequences.
Each contains the motif with at most d=2 mismatches.

S_1	`atcactcgttctcctctaatgtgtaaagacgtactaccgacctta`
S_2	`acgccgaccggtccgatccttgtatagctcctaacgggcatcagc`
S_3	`tcctgactgcatcgcgatctcggtagtttcctgttcatcattttt`
S_4	`ggccctcagcatcgtgcgtcctgctaacacattcccatgcagctt`
S_5	`tgaaaagaatttacggtaaaggatccacatccaatcgtgtgaaag`

Planted motif: ? ?

Introduction

The (l, d) planted motif problem

Find a motif of length l=8 across these DNA sequences.
Each contains the motif with at most d=2 mismatches.

S_1 atcactcgttctcctctaatgtgtaaagacgtactaccgacctta

S_2 acgccgaccggtccgatccttgtatagctcctaacgggcatcagc

S_3 tcctgactgcatcgcgatctcggtagtttcctgttcatcattttt

S_4 ggccctcagcatcgtgcgtcctgctaacacattcccatgcagctt

S_5 tgaaaagaatttacggtaaaggatccacatccaatcgtgtgaaag

Planted motif: ccatcgtt

Introduction

The (l, d) planted motif problem

- motifs: significant sub-sequences occuring repeatedly in DNA
- DNA motif finding must allow for mismatches due to mutation
- motif search is known as a difficult (NP-complete) problem in computational biology and CS

Introduction

Key concepts

- *l*-mer

  - sequence of length l

  $S_1 = \texttt{atcactcgttctcctctaatgtgtaaagacgtactaccgacctta}$

- Hamming distance d_H

  - number of mismatches between l-mers

  $$x_1 = \texttt{cgatcctt} \qquad d_H(x_1, x_2) = 2$$
  $$x_2 = \texttt{ccatcgtt}$$

Introduction

Key concepts

- *d*-neighborhood

  - ex. the set of all *d*-neighbors of acgt, d=2:

acgt,

ccgt, gcgt, tcgt, aagt, aggt, atgt,
acat, acgt, actt, acga, acgc, acgg,

cagt, cggt, ctgt, ccat, ccct, cctt, ccga, ccgc, ccgg,
gagt, gggt, gtgt, gcat, gcct, gctt, gcga, gcgc, gcgg,
tagt, tggt, ttgt, tcat, tcct, tctt, tcga, tcgc, tcgg,
aaat, aact, aatt, aaga, aagc, aagg, agat, agct, agtt,
agga, aggc, aggg, atat, atct, attt, atga, atgc, atgg,
acaa, acac, acag, acca, accc, accg, acta, actc, actg

Problem statement

EMS-GT

Demonstration

S_1 atcactcgttctcctctaatgtgtaaagacgtactaccgacctta

S_2 acgccgaccggtccgatccttgtatagctcctaacgggcatcagc

S_3 tcctgactgcatcgcgatctcggtagtttcctgttcatcattttt

S_4 ggccctcagcatcgtgcgtcctgctaacacattcccatgcagctt

S_5 tgaaaagaatttacggtaaaggatccacatccaatcgtgtgaaag

EMS-GT

Demonstration

neighborhood of **atcactcg**

S_1 atcactcgttctcctctaatgtgtaaagacgtactaccgacctta

S_2 acgccgaccggtccgatccttgtatagctcctaacgggcatcagc

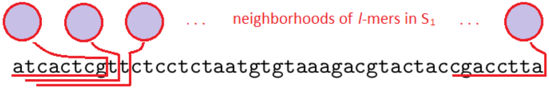
S_3 tcctgactgcatcgcgatctcggtagtttcctgttcatcatttt

S_4 ggccctcagcatcgtgcgtcctgctaacacattcccatgcagctt

S_5 tgaaaagaatttacggtaaaggatccacatccaatcgtgtgaaag

EMS-GT

Demonstration



S_1 atcactcgttctcctctaatgtgtaaagacgtactaccgacctta

S_2 acgccgaccggtccgatccttgtatagctcctaacgggcatcagc

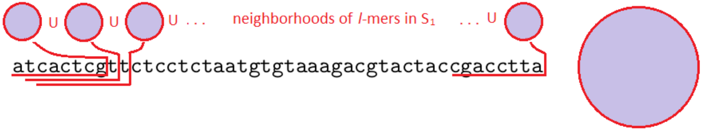
S_3 tcctgactgcatcgcgatctcggtagtttcctgttcatcatttt

S_4 ggccctcagcatcgtgcgtcctgctaacacattcccatgcagctt

S_5 tgaaaagaatttacggtaaaggatccacatccaatcgtgtgaaag

EMS-GT

Demonstration



∪ ∪ ∪ . . . neighborhoods of *l*-mers in S_1 . . . ∪

S_1 atcactcgttctcctctaatgtgtaaagacgtactaccgacctta

S_2 acgccgaccggtccgatccttgtatagctcctaacgggcatcagc

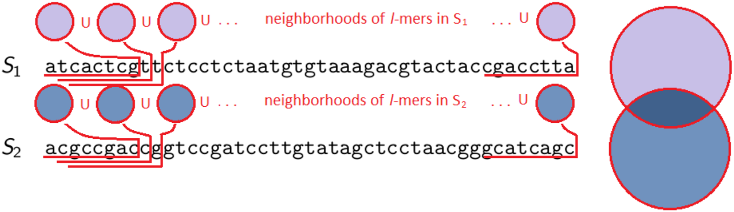
S_3 tcctgactgcatcgcgatctcggtagtttcctgttcatcatttt

S_4 ggccctcagcatcgtgcgtcctgctaacacattcccatgcagctt

S_5 tgaaaagaatttacggtaaaggatccacatccaatcgtgtgaaag

EMS-GT

Demonstration


∪ ∪ ∪ . . . neighborhoods of *l*-mers in S_1 . . . ∪
∪ ∪ ∪ . . . neighborhoods of *l*-mers in S_2 . . . ∪

S_1 atcactcgttctcctctaatgtgtaaagacgtactaccgacctta

S_2 acgccgacggtccgatccttgtatagctcctaacgggcatcagc

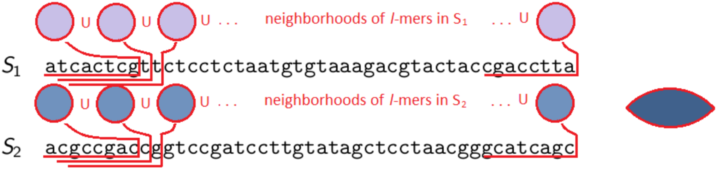
S_3 tcctgactgcatcgcgatctcggtagtttcctgttcatcatttt

S_4 ggccctcagcatcgtgcgtcctgctaacacattcccatgcagctt

S_5 tgaaaagaatttacggtaaaggatccacatccaatcgtgtgaaag

EMS-GT

Demonstration



∪ ∪ ∪ . . . neighborhoods of l-mers in S_1 . . . ∪

S_1 atcactcgttctcctctaatgtgtaaagacgtactaccgacctta

∪ ∪ ∪ . . . neighborhoods of l-mers in S_2 . . . ∪

S_2 acgccgacggtccgatccttgtatagctcctaacgggcatcagc

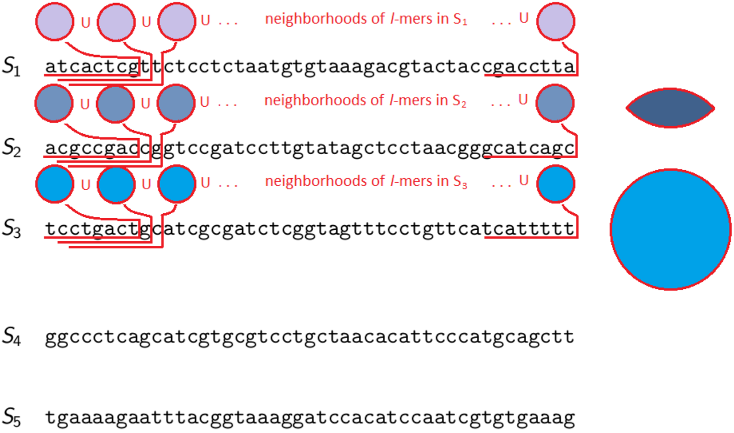
S_3 tcctgactgcatcgcgatctcggtagtttcctgttcatcatttt

S_4 ggccctcagcatcgtgcgtcctgctaacacattcccatgcagctt

S_5 tgaaaagaatttacggtaaaggatccacatccaatcgtgtgaaag

EMS-GT

Demonstration



∪ ∪ ∪ . . . neighborhoods of *l*-mers in S_1 . . . ∪
∪ ∪ ∪ . . . neighborhoods of *l*-mers in S_2 . . . ∪
∪ ∪ ∪ . . . neighborhoods of *l*-mers in S_3 . . . ∪

S_1 atcactcgttctcctctaatgtgtaaagacgtactaccgacctta

S_2 acgccgaccggtccgatccttgtatagctcctaacgggcatcagc

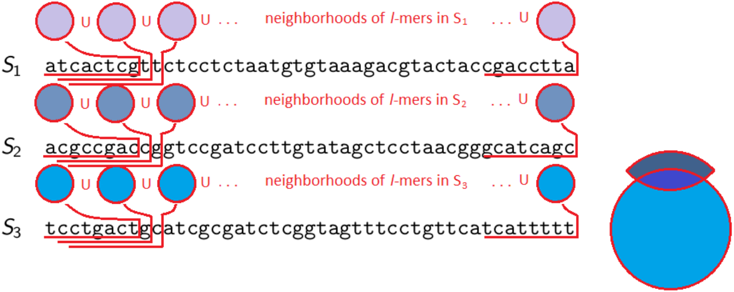
S_3 tcctgactgcatcgcgatctcggtagtttcctgttcatcatttt

S_4 ggccctcagcatcgtgcgtcctgctaacacattcccatgcagctt

S_5 tgaaaagaatttacggtaaggatccacatccaatcgtgtgaaag

EMS-GT

Demonstration



∪ ∪ ∪ . . . neighborhoods of l-mers in S_1 . . . ∪

S_1 atcactcgttctcctctaatgtgtaaagacgtactaccgacctta

∪ ∪ ∪ . . . neighborhoods of l-mers in S_2 . . . ∪

S_2 acgccgacggtccgatccttgtatagctcctaacgggcatcagc

∪ ∪ ∪ . . . neighborhoods of l-mers in S_3 . . . ∪

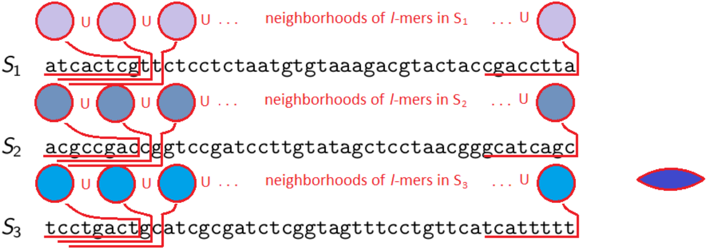
S_3 tcctgactgcatcgcgatctcggtagtttcctgttcatcatttt

S_4 ggccctcagcatcgtgcgtcctgctaacacattcccatgcagctt

S_5 tgaaaagaatttacggtaaggatccacatccaatcgtgtgaaag

EMS-GT

Demonstration



∪ ∪ ∪ . . . neighborhoods of *l*-mers in S_1 . . . ∪

S_1 atcactcgttctcctctaatgtgtaaagacgtactaccgacctta

∪ ∪ ∪ . . . neighborhoods of *l*-mers in S_2 . . . ∪

S_2 acgccgacggtccgatccttgtatagctcctaacgggcatcagc

∪ ∪ ∪ . . . neighborhoods of *l*-mers in S_3 . . . ∪

S_3 tcctgactgcatcgcgatctcggtagtttcctgttcatcatttt

S_4 ggccctcagcatcgtgcgtcctgctaacacattcccatgcagctt

S_5 tgaaaagaatttacggtaaaggatccacatccaatcgtgtgaaag

EMS-GT

Introduction

- exact motif search (EMS) algorithm
  - *exact* - performs an exhaustive search for possible motifs

    - as opposed to *heuristic* methods
- uses a generate-and-test (GT) search approach
  - *generate* - narrows the search to a set of candidate motifs
  - *test* - checks each candidate to see if it is a motif

EMS-GT

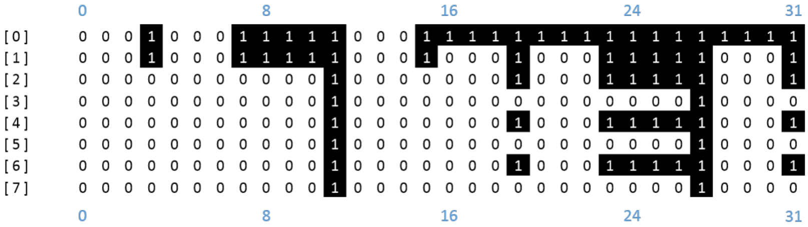
Representing sets

- EMS-GT must operate on sets of l-mers.
- There are 4^l possible l-mers that can be formed with $\{\texttt{a},\texttt{c},\texttt{g},\texttt{t}\}$
- Thus, to represent a set of l-mers, EMS-GT uses 4^l bits,
  - set to 1 if the corresponding l-mer is a member of the set,
  - set to 0 otherwise.
- For efficiency, EMS-GT stores the 4^l bits as $\frac{4^l}{32}$ 32-bit integers.

EMS-GT

Representing sets

- $N(\ \texttt{acgt},\ 1\ )\qquad l{=}4;\ \ 4^l = 256,\ \frac{4^l}{32} = 8$

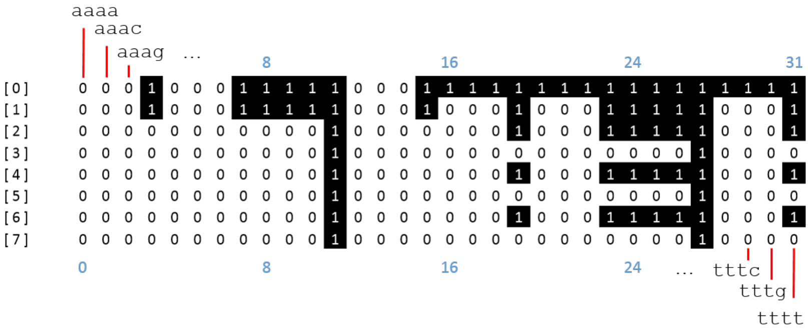


	0								8								16								24							31
[0]	0	0	0	1	0	0	0	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
[1]	0	0	0	1	0	0	0	1	1	1	1	1	0	0	0	1	0	0	0	1	0	0	0	1	1	1	1	1	0	0	0	1
[2]	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	0	0	0	1
[3]	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
[4]	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	0	0	0	1
[5]	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
[6]	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	0	0	0	1
[7]	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	0								8								16								24							31

EMS-GT

Representing sets

- $N($ acgt, 1 $)$ $\quad l$=4; $4^l = 256$, $\frac{4^l}{32} = 8$

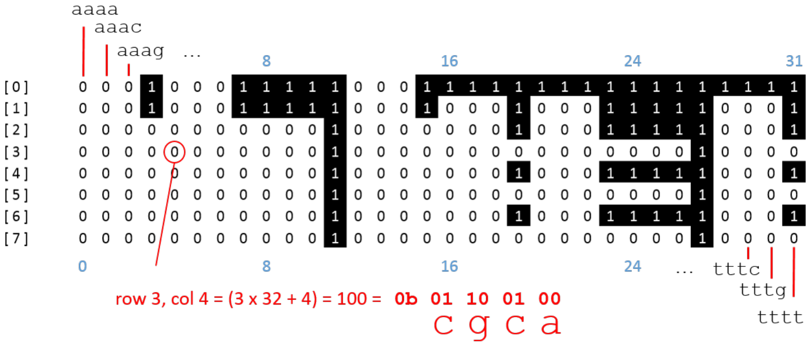


[0]	0	0	0	1	0	0	0	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
[1]	0	0	0	1	0	0	0	1	1	1	1	1	0	0	0	1	0	0	0	1	0	0	0	1	1	1	1	1	0	0	0	1
[2]	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	0	0	0	1
[3]	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
[4]	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	0	0	0	1
[5]	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
[6]	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	0	0	0	1
[7]	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

EMS-GT

Representing sets

- $N($ acgt, 1 $)$ $\quad l{=}4;\;\; 4^l = 256,\; \frac{4^l}{32} = 8$



	0								8								16								24							31
[0]	0	0	0	1	0	0	0	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
[1]	0	0	0	1	0	0	0	1	1	1	1	1	0	0	0	1	0	0	0	1	0	0	0	1	1	1	1	1	0	0	0	1
[2]	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	0	0	0	1
[3]	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
[4]	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	0	0	0	1
[5]	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
[6]	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	0	0	0	1
[7]	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	0								8								16								24							

row 3, col 4 = (3 x 32 + 4) = 100 = **0b 01 10 01 00**

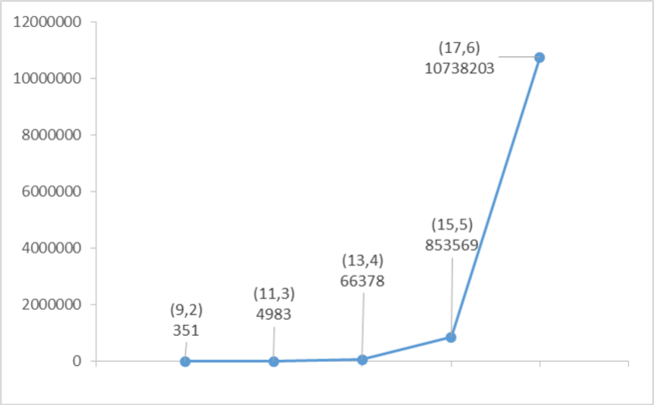
c g c a

EMS-GT

Generating *d*-neighborhoods

Q: *How can we generate the neighborhood of an l-mer x?*

- generate each *d*-neighbor, find its bit flag, and set to 1?

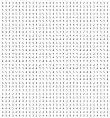


EMS-GT

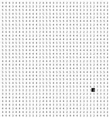
Generating d-neighborhoods

Q: *How can we generate the neighborhood of an l-mer x?*

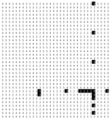
- generate the neighborhood in blocks, using $(k+2)$ patterns
  - use the last k characters of x to determine the patterns
  - use the first $l-k$ characters of x to assign patterns to blocks



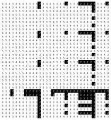
Pattern -1



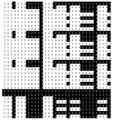
Pattern 0



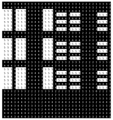
Pattern 1



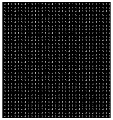
Pattern 2



Pattern 3



Pattern 4



Pattern 5

EMS-GT

Building sets in blocks

EMS-GT

Results