Improving an Exact Solution to the (l,d) Planted Motif Problem

Maria Clara Isabel Sia

October 22, 2015

- motifs are repeated sub-sequences in DNA that have some biological significance
- DNA motif finding searches for motifs over a set of DNA sequences, allowing for mismatches due to mutation
- known as a difficult problem in computational biology and CS (proven NP-complete)

Introduction

The (l,d) planted motif problem

Find a motif of length l=8 across these 5 DNA sequences, each containing 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

Solutions to the (l,d) planted motif problem)

There are two types of methods used by motif search algorithms:

- heuristic methods (ex. probabilistic sampling, projection) perform an iterative local search which is efficient, but not guaranteed to find all motifs
- exact methods (ex. combinatorial search, tree pruning) perform an exhaustive search which will find all possible motifs, at the cost of time/space efficiency

Introduction

l-mers, Hamming distances, and d-neighborhoods

- l-mer
- Hamming distance d_H
- d-neighbor

Introduction

l-mers, Hamming distances, and d-neighborhoods

- l-mer
  - sequence of length l

  $S_1 =$ `atcactcgttctcctctaatgtgtaaagacgtactaccgacctta`

- Hamming distance d_H
- d-neighbor

Introduction

l-mers, Hamming distances, and *d*-neighborhoods

- *l*-mer
- Hamming distance d_H

  - number of mismatches between *l*-mers x_1 and x_2

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

- *d*-neighbor

Introduction

l-mers, Hamming distances, and *d*-neighborhoods

- *l*-mer
- Hamming distance d_H
- *d*-neighbor
  - two *l*-mers x and x' are *d*-neighbors if $d_H(x, x') \leq d$

N(ccatcgtt, 2) $\rightarrow$ *d*-neighborhood of ccatcgtt, d=2

= { ccatcgtt,

acatcgtt,gcatcgtt,tcatcgtt,caatcgtt,cgatcgtt,ctatcgtt,

...all *l*-mers with 1 mismatch

aaatcgtt,agatcgtt,atatcgtt,gaatcgtt,ggatcgtt,gtatcgtt,

taatcgtt,tgatcgtt,ttatcgtt,acctcgtt,acgtcgtt,acttcgtt,

...all *l*-mers with 2 mismatches

}

EMS-GT

Nabos, 2014

- an exact motif search (EMS) algorithm based on the candidate generate-and-test (GT) principle
- solves any (l,d) planted motif problem instance, $l \leq 17$
- operates on a bit-based representation of the search space

EMS-GT

Generate-and-test approach

EMS-GT proceeds in two steps:

1. Generate the set C of candidate motifs: find the common d-neighbors of the first n' sequences $S_1, S_2, ..., S_{n'}$.

$$C = \mathcal{N}(S_1, d) \cap \mathcal{N}(S_2, d) \cap ... \cap \mathcal{N}(S_{n'}, d), \qquad n' \leq n$$

2. Test every candidate $c \in C$: if a d-neighbor of c appears in each of the remaining sequences $S_{n'+1}, S_{n'+2}, ...S_n$, accept c as a motif.

EMS-GT

Generate-and-test approach

$(l,d) = (8,2)$

S_1 `atcactcgttctcctctaatgtgtaaagacgtactaccgacctta`

S_2 `acgccgaccggtccgatccttgtatagctcctaacgggcatcagc`

S_3 `tcctgactgcatcgcgatctcggtagtttcctgttcatcatttt`

S_4 `ggccctcagcatcgtgcgtcctgctaacacattcccatgcagctt`

S_5 `tgaaaagaatttacggtaaaggatccacatccaatcgtgtgaaag`

EMS-GT

Bit-based representation of the search space

- The search space contains all 4^l possible l-mers that can be formed with $\Sigma = \{\texttt{a}, \texttt{c}, \texttt{g}, \texttt{t}\}$.
- To represent sets in this space, EMS-GT assigns each of the 4^l l-mers a bit flag, which is 1 if the l-mer is a member of the set, 0 otherwise.
- For efficiency, EMS-GT stores the 4^l bits as $\frac{4^l}{32}$ 32-bit integers.

EMS-GT

Bit-based representation of the search space

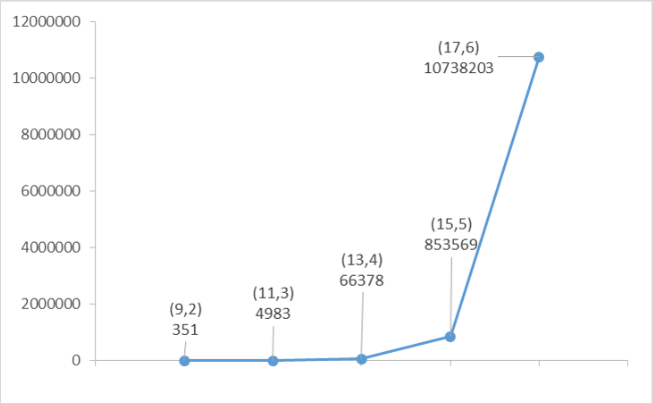
- $N($ acgt, 1 $)$: $4^l = 256$, $\frac{4^l}{32} = 8$

[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 generates a neighborhood bit-array by generating each individual neighbor, then finding and setting its bit flag.

EMS-GT

Bit-based representation of the search space



- l-mer neighborhoods grow very quickly with (l, d), meaning that EMS-GT must spend more time locating and setting bits.

Research objectives

Improving EMS-GT

The main objectives of this research are:

1. To develop a speedup technique for EMS-GT that takes advantage of distance-related patterns in the search space;
2. To evaluate the speedup technique with regard to improvement in runtime; and
3. To evaluate the improved version of the EMS-GT algorithm against state-of-the-art motif search algorithms.

Speedup technique

Key observation

- If a bit-array N_x representing the neighborhood of l-mer x is partitioned into blocks of 4^k bits each,

Speedup technique

Key observation

- If a bit-array N_x representing the neighborhood of l-mer x is partitioned into blocks of 4^k bits each,

Ex. $N($ `acgtacgtacgt`$, 5), k = 5$

block size $= 4^5 = 1024 = 32 \times 32$

Speedup technique

Key observation

- If a bit-array N_x representing the neighborhood of l-mer x is partitioned into blocks of 4^k bits each,

$$\text{Ex. } N(\texttt{ acgtacgtacgt}, 5\ ),\ k = 5$$
$$\text{block size} = 4^5 = 1024 = 32 \times 32$$

- the 4^k l-mers represented in a block will all begin with the same prefix (first $l - k$ characters), and will differ only in the k-suffix (last k characters);

Speedup technique

Key observation

- If a bit-array N_x representing the neighborhood of l-mer x is partitioned into blocks of 4^k bits each,

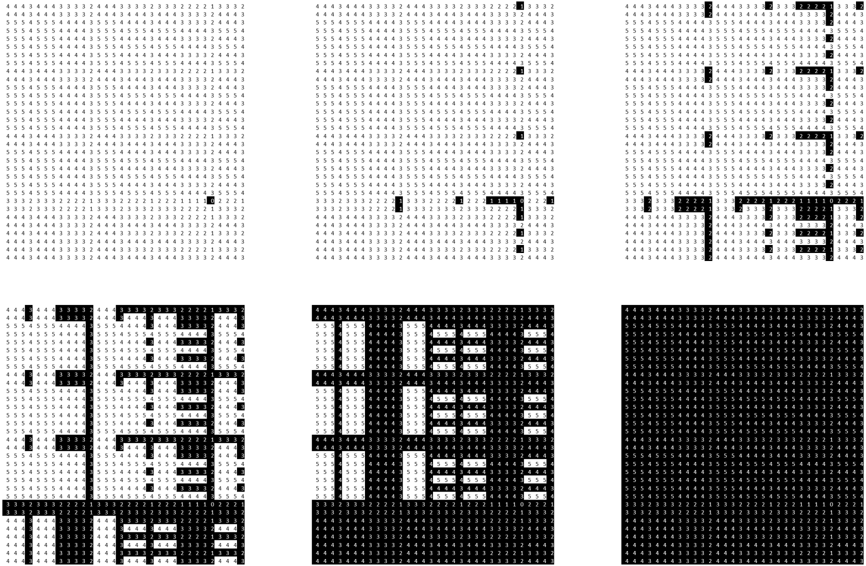
$$\text{Ex. } N(\texttt{ acgtacgtacgt}, 5\text{ }),\ k = 5$$
$$\text{block size} = 4^5 = 1024 = 32 \times 32$$

- the 4^k l-mers represented in a block will all begin with the same prefix (first $l - k$ characters), and will differ only in the k-suffix (last k characters);

- each block conforms to one of $(k + 2)$ patterns.

Speedup technique

Block patterns in the d-neighborhood of acgtacgtacgt, d=5, k=5



Speedup technique

Key observation

- In a d-neighbor of x, the d allowable mismatches from x are distributed between the prefix and the k-suffix.

```
acgtacg tacgt
acgaaaa tccga
```

- If a block's prefix already has p mismatches from x's prefix, then within that block, any neighbor must have a suffix with at most $d - p$ mismatches from x's suffix.

aaaaa →

4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2	2	2	2	1	2	2	2	1	1	1	1	0	2	2	2	1
3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3

← ttttt

aaaaa →

4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2	2	2	2	1	2	2	2	1	1	1	1	0	2	2	2	1
3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3

← ttttt

aaaaa →

4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2	2	2	2	1	2	2	2	1	1	1	1	0	2	2	2	1
3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3

← ttttt

aaaaa →	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2	
	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	
	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	
	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	
	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	
	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	
	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	
	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	
	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2	
	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	
	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	
	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	
	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	
	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	
	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	
	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	
	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2	
	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	
	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	
	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	
	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	
	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	
	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	
	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	
	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2	2	2	2	1	2	2	2	1	1	1	1	0	2	2	2	1	
	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2	
	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2	
	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	
	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2	
	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	
	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2	
	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	← ttttt

aaaaa →

4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2	2	2	2	1	2	2	2	1	1	1	1	0	2	2	2	1
3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3

← ttttt

aaaaa →

4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2	2	2	2	1	2	2	2	1	1	1	1	0	2	2	2	1
3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3

← ttttt

aaaaa →

4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4	5	5	5	4	5	5	5	4	4	4	4	3	5	5	5	4
3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2	2	2	2	1	2	2	2	1	1	1	1	0	2	2	2	1
3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	3	3	3	2	3	3	3	2	2	2	2	1	3	3	3	2
4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3	4	4	4	3	4	4	4	3	3	3	3	2	4	4	4	3

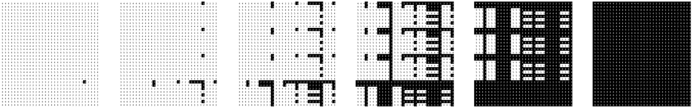
← ttttt

Speedup technique

Generate and apply patterns

To generate N_x for $x = yz$, we perform two steps:

1. From x's suffix z, generate P, the set of block patterns.

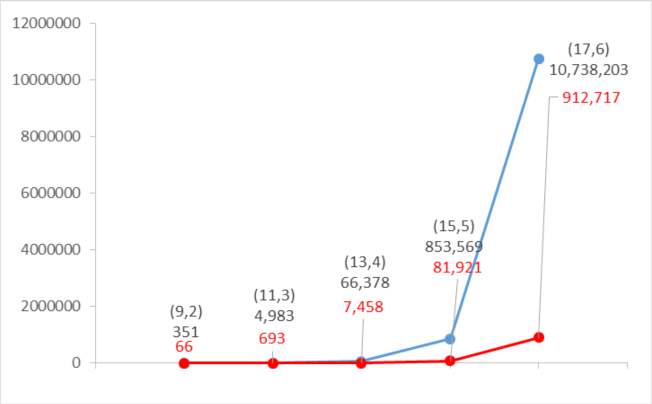


P_0 P_1 P_2 P_3 P_4 P_5

2. From x's prefix y, recursively generate each d-neighbor y', and apply $P_{(d-d_H(y,y'))}$ to the block whose prefix is y'.

Results

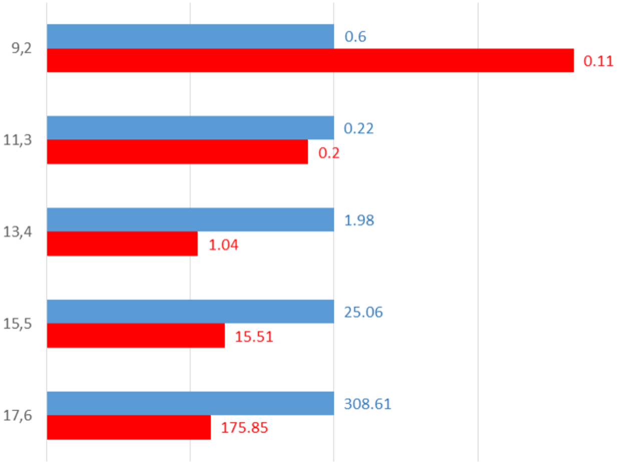
Reduction in recursive neighbor generation



- We still generate d-neighbors recursively, but for a shorter sequence y, of length $l - k$; this graph shows that, for k=5, neighborhood size is reduced by a factor of 10.

Results

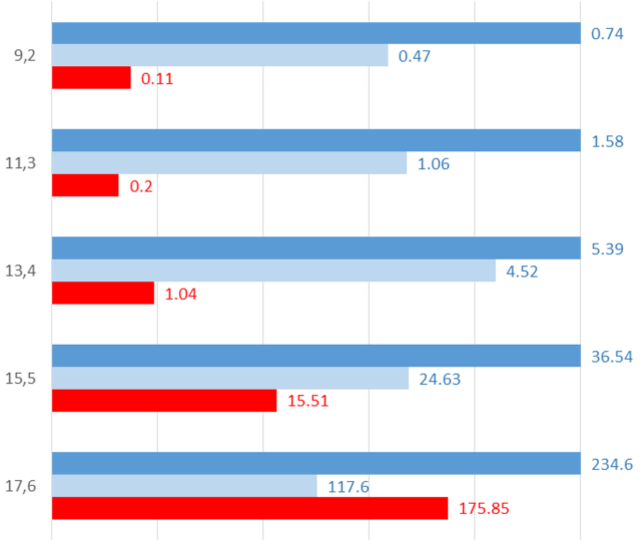
EMS-GT **without** vs. **with** the speedup technique



Runtime in seconds, averaged over 20 synthetic datasets per (l,d)

Results

PMS8 vs. **qPMS9** vs. **EMS-GT with speedup technique**



Runtime in seconds, averaged over 20 synthetic datasets per (l,d)

Conclusions

- speedup technique
- runtime improvement
- comparison with state-of-the-art