Optimal Computation of Avoided Words

Yannis Almirantis[1] Panagiotis Charalampopoulos[2]
Jia Gao[2] Costas S. Iliopoulos[2] Manal Mohamed[2]
Solon P. Pissis[2] Dimitris Polychronopoulos[3]

[1]National Center for Scientific Research Demokritos, Greece

[2]King's College London, UK

[3]Imperial College London, UK

AXA 2016

Venice, Italy, 23 Jun. 2016

Avoided Word

Let $w = w[0]w[1] \mathinner{.\,.} w[m-1]$ and $x = x[0]x[1] \mathinner{.\,.} x[n-1]$ over Σ, $n > m$.

Avoided Word

Let $w = w[0]w[1] \mathinner{.\,.} w[m-1]$ and $x = x[0]x[1] \mathinner{.\,.} x[n-1]$ over Σ, $n > m$.

We denote by:

Avoided Word

Let $w = w[0]w[1] \dots w[m-1]$ and $x = x[0]x[1] \dots x[n-1]$ over Σ, $n > m$.

We denote by:

- $f(w)$ the $\#$ occurrences (observed frequency) of w in x.

Avoided Word

Let $w = w[0]w[1] \mathinner{.\,.} w[m-1]$ and $x = x[0]x[1] \mathinner{.\,.} x[n-1]$ over Σ, $n > m$.

We denote by:

- $f(w)$ the $\#$ occurrences (observed frequency) of w in x.
- $f(w_p)$ the $\#$ occurrences of $w_p = w[0 \mathinner{.\,.} m-2]$ in x.

Avoided Word

Let $w = w[0]w[1] \,.\,.\, w[m-1]$ and $x = x[0]x[1] \,.\,.\, x[n-1]$ over Σ, $n > m$.

We denote by:

- $f(w)$ the # occurrences (observed frequency) of w in x.
- $f(w_p)$ the # occurrences of $w_p = w[0 \,.\,.\, m-2]$ in x.
- $f(w_i)$ the # occurrences of $w_i = w[1 \,.\,.\, m-2]$ in x.

Avoided Word

Let $w = w[0]w[1] \,.\,.\, w[m-1]$ and $x = x[0]x[1] \,.\,.\, x[n-1]$ over Σ, $n > m$.

We denote by:

- $f(w)$ the # occurrences (observed frequency) of w in x.
- $f(w_p)$ the # occurrences of $w_p = w[0 \,.\,.\, m-2]$ in x.
- $f(w_i)$ the # occurrences of $w_i = w[1 \,.\,.\, m-2]$ in x.
- $f(w_s)$ the # occurrences of $w_s = w[1 \,.\,.\, m-1]$ in x.

Avoided Word

Let $w = w[0]w[1] \dots w[m-1]$ and $x = x[0]x[1] \dots x[n-1]$ over Σ, $n > m$.

We denote by:

- $f(w)$ the # occurrences (observed frequency) of w in x.
- $f(w_p)$ the # occurrences of $w_p = w[0 \dots m-2]$ in x.
- $f(w_i)$ the # occurrences of $w_i = w[1 \dots m-2]$ in x.
- $f(w_s)$ the # occurrences of $w_s = w[1 \dots m-1]$ in x.

We define the **expected frequency** of word w in x as:

Avoided Word

Let $w = w[0]w[1] \ldots w[m-1]$ and $x = x[0]x[1] \ldots x[n-1]$ over Σ, $n > m$.

We denote by:

- $f(w)$ the # occurrences (observed frequency) of w in x.
- $f(w_p)$ the # occurrences of $w_p = w[0 \ldots m-2]$ in x.
- $f(w_i)$ the # occurrences of $w_i = w[1 \ldots m-2]$ in x.
- $f(w_s)$ the # occurrences of $w_s = w[1 \ldots m-1]$ in x.

We define the **expected frequency** of word w in x as:

$$E(w) = \frac{f(w_p) \times f(w_s)}{f(w_i)}. \qquad (1)$$

Avoided Word

Let $w = w[0]w[1] \ldots w[m-1]$ and $x = x[0]x[1] \ldots x[n-1]$ over Σ, $n > m$.

We denote by:

- $f(w)$ the # occurrences (observed frequency) of w in x.
- $f(w_p)$ the # occurrences of $w_p = w[0 \ldots m-2]$ in x.
- $f(w_i)$ the # occurrences of $w_i = w[1 \ldots m-2]$ in x.
- $f(w_s)$ the # occurrences of $w_s = w[1 \ldots m-1]$ in x.

We define the **expected frequency** of word w in x as:

$$E(w) = \frac{f(w_p) \times f(w_s)}{f(w_i)}. \tag{1}$$

Why?

Avoided Word

Let $w = w[0]w[1] \mathinner{.\,.} w[m-1]$ and $x = x[0]x[1] \mathinner{.\,.} x[n-1]$ over Σ, $n > m$.

We denote by:

- $f(w)$ the # occurrences (observed frequency) of w in x.
- $f(w_p)$ the # occurrences of $w_p = w[0 \mathinner{.\,.} m-2]$ in x.
- $f(w_i)$ the # occurrences of $w_i = w[1 \mathinner{.\,.} m-2]$ in x.
- $f(w_s)$ the # occurrences of $w_s = w[1 \mathinner{.\,.} m-1]$ in x.

We define the **expected frequency** of word w in x as:

$$E(w) = \frac{f(w_p) \times f(w_s)}{f(w_i)}. \tag{1}$$

Why?

$$\underbrace{\frac{f(w_p)}{f(w_i)}}_{\text{prob. } w_p \text{ precedes } w_i}$$

Avoided Word

Let $w = w[0]w[1] \,..\, w[m-1]$ and $x = x[0]x[1] \,..\, x[n-1]$ over Σ, $n > m$.

We denote by:

- $f(w)$ the # occurrences (observed frequency) of w in x.
- $f(w_p)$ the # occurrences of $w_p = w[0 \,..\, m-2]$ in x.
- $f(w_i)$ the # occurrences of $w_i = w[1 \,..\, m-2]$ in x.
- $f(w_s)$ the # occurrences of $w_s = w[1 \,..\, m-1]$ in x.

We define the **expected frequency** of word w in x as:

$$E(w) = \frac{f(w_p) \times f(w_s)}{f(w_i)}. \tag{1}$$

Why?

$$\underbrace{\frac{f(w_p)}{f(w_i)}}_{\text{prob. } w_p \text{ precedes } w_i} \quad \times \quad \underbrace{\frac{f(w_s)}{f(w_i)}}_{\text{prob. } w_s \text{ succeeds } w_i}$$

Avoided Word

Let $w = w[0]w[1]\,.\,.\,w[m-1]$ and $x = x[0]x[1]\,.\,.\,x[n-1]$ over Σ, $n > m$.

We denote by:

- $f(w)$ the # occurrences (observed frequency) of w in x.
- $f(w_p)$ the # occurrences of $w_p = w[0\,.\,.\,m-2]$ in x.
- $f(w_i)$ the # occurrences of $w_i = w[1\,.\,.\,m-2]$ in x.
- $f(w_s)$ the # occurrences of $w_s = w[1\,.\,.\,m-1]$ in x.

We define the **expected frequency** of word w in x as:

$$E(w) = \frac{f(w_p) \times f(w_s)}{f(w_i)}. \tag{1}$$

Why?

$$\underbrace{\frac{f(w_p)}{f(w_i)}}_{\text{prob. } w_p \text{ precedes } w_i} \times \underbrace{\frac{f(w_s)}{f(w_i)}}_{\text{prob. } w_s \text{ succeeds } w_i} \times f(w_i)$$

Avoided Word

Let $w = w[0]w[1]\dots w[m-1]$ and $x = x[0]x[1]\dots x[n-1]$ over Σ, $n > m$.

We denote by:

- $f(w)$ the # occurrences (observed frequency) of w in x.
- $f(w_p)$ the # occurrences of $w_p = w[0\dots m-2]$ in x.
- $f(w_i)$ the # occurrences of $w_i = w[1\dots m-2]$ in x.
- $f(w_s)$ the # occurrences of $w_s = w[1\dots m-1]$ in x.

We define the **expected frequency** of word w in x as:

$$E(w) = \frac{f(w_p) \times f(w_s)}{f(w_i)}. \tag{1}$$

Why?

$$\underbrace{\frac{f(w_p)}{f(w_i)}}_{\text{prob. } w_p \text{ precedes } w_i} \times \underbrace{\frac{f(w_s)}{f(w_i)}}_{\text{prob. } w_s \text{ succeeds } w_i} \times f(w_i) = E(w). \tag{2}$$

The Problem

The Problem

We define the **standard deviation** (χ^2 test) of w in x as:

The Problem

We define the **standard deviation** (χ^2 test) of w in x as:

$$\mathit{std}(w) = \frac{f(w) - E(w)}{\max\{\sqrt{E(w)}, 1\}}. \tag{3}$$

The Problem

We define the **standard deviation** (χ^2 test) of w in x as:

$$\mathit{std}(w) = \frac{f(w) - E(w)}{\max\{\sqrt{E(w)}, 1\}}. \tag{3}$$

Given $\rho < 0$, a word w is called ρ-**avoided** in x if $\mathit{std}(w) \leq \rho$.

The Problem

We define the **standard deviation** (χ^2 test) of w in x as:

$$\mathit{std}(w) = \frac{f(w) - E(w)}{\max\{\sqrt{E(w)}, 1\}}. \tag{3}$$

Given $\rho < 0$, a word w is called ρ-**avoided** in x if $\mathit{std}(w) \leq \rho$.

AvoidedWordsComputation **Input:** A word x of length n, an integer $k > 2$, and a $\rho < 0$ **Output:** All ρ-avoided words of length k in x

The Problem

We define the **standard deviation** (χ^2 test) of w in x as:

$$\textit{std}(w) = \frac{f(w) - E(w)}{\max\{\sqrt{E(w)}, 1\}}. \tag{3}$$

Given $\rho < 0$, a word w is called ρ-**avoided** in x if $\textit{std}(w) \leq \rho$.

AvoidedWordsComputation **Input:** A word x of length n, an integer $k > 2$, and a $\rho < 0$ **Output:** All ρ-avoided words of length k in x

This problem was first considered by Brendel, Beckmann, and Trifonov in 1986.

The Problem

We define the **standard deviation** (χ^2 test) of w in x as:

$$\textit{std}(w) = \frac{f(w) - E(w)}{\max\{\sqrt{E(w)}, 1\}}. \tag{3}$$

Given $\rho < 0$, a word w is called ρ-**avoided** in x if $\textit{std}(w) \leq \rho$.

AvoidedWordsComputation **Input:** A word x of length n, an integer $k > 2$, and a $\rho < 0$ **Output:** All ρ-avoided words of length k in x

This problem was first considered by Brendel, Beckmann, and Trifonov in 1986.

No non-trivial solution was provided [Brendel et al, 1986].

Useful Properties

Useful Properties

Definition

An absent word w of x is *minimal* if and only if all its proper factors occur in x.

Useful Properties

Definition

An absent word w of x is *minimal* if and only if all its proper factors occur in x.

Lemma (Absent)

Any absent ρ-avoided word w in x is a minimal absent word of x.

Useful Properties

Definition

An absent word w of x is *minimal* if and only if all its proper factors occur in x.

Lemma (Absent)

Any absent ρ-avoided word w in x is a minimal absent word of x.

Proof.

Useful Properties

Definition

An absent word w of x is *minimal* if and only if all its proper factors occur in x.

Lemma (Absent)

Any absent ρ-avoided word w in x is a minimal absent word of x.

Proof.

$$\textit{std}(w) = \frac{f(w) - E(w)}{\max\{\sqrt{E(w)}, 1\}} \leq \rho < 0.$$

Useful Properties

Definition

An absent word w of x is *minimal* if and only if all its proper factors occur in x.

Lemma (Absent)

Any absent ρ-avoided word w in x is a minimal absent word of x.

Proof.

$$\textit{std}(w) = \frac{f(w) - E(w)}{\max\{\sqrt{E(w)}, 1\}} \leq \rho < 0.$$

$$f(w) - E(w) < 0$$

Useful Properties

Definition

An absent word w of x is *minimal* if and only if all its proper factors occur in x.

Lemma (Absent)

Any absent ρ-avoided word w in x is a minimal absent word of x.

Proof.

$$\textit{std}(w) = \frac{f(w) - E(w)}{\max\{\sqrt{E(w)}, 1\}} \leq \rho < 0.$$

$$f(w) - E(w) < 0 \rightarrow E(w) > 0$$

Useful Properties

Definition

An absent word w of x is *minimal* if and only if all its proper factors occur in x.

Lemma (Absent)

Any absent ρ-avoided word w in x is a minimal absent word of x.

Proof.

$$\textit{std}(w) = \frac{f(w) - E(w)}{\max\{\sqrt{E(w)}, 1\}} \leq \rho < 0.$$

$$f(w) - E(w) < 0 \rightarrow E(w) > 0$$

$$\frac{f(w_p) \times f(w_s)}{f(w_i)} > 0$$

Useful Properties

Definition

An absent word w of x is *minimal* if and only if all its proper factors occur in x.

Lemma (Absent)

Any absent ρ-avoided word w in x is a minimal absent word of x.

Proof.

$$\mathit{std}(w) = \frac{f(w) - E(w)}{\max\{\sqrt{E(w)}, 1\}} \leq \rho < 0.$$

$$f(w) - E(w) < 0 \rightarrow E(w) > 0$$

$$\frac{f(w_p) \times f(w_s)}{f(w_i)} > 0 \rightarrow f(w_p) > 0, f(w_s) > 0.$$

$\square$

Useful Properties

Useful Properties

Lemma (Occurring)

Let w be a word occurring in x and $\mathcal{T}(x)$ be the suffix tree of x. Then, if w_p is a path-label of an implicit node of $\mathcal{T}(x)$, $std(w) \geq 0$.

Useful Properties

Lemma (Occurring)

Let w be a word occurring in x and $\mathcal{T}(x)$ be the suffix tree of x. Then, if w_p is a path-label of an implicit node of $\mathcal{T}(x)$, $std(w) \geq 0$.

Proof.

Useful Properties

Lemma (Occurring)

Let w be a word occurring in x and $\mathcal{T}(x)$ be the suffix tree of x. Then, if w_p is a path-label of an implicit node of $\mathcal{T}(x)$, $std(w) \geq 0$.

Proof.

For any occurring w we have $f(w_i) \geq f(w_s)$.

Useful Properties

Lemma (Occurring)

Let w be a word occurring in x and $\mathcal{T}(x)$ be the suffix tree of x. Then, if w_p is a path-label of an implicit node of $\mathcal{T}(x)$, $std(w) \geq 0$.

Proof.

For any occurring w we have $f(w_i) \geq f(w_s)$.
This implies $f(w_p) \geq \frac{f(w_s)}{f(w_i)} \times f(w_p)$

Lemma (Occurring)

Let w be a word occurring in x and $\mathcal{T}(x)$ be the suffix tree of x. Then, if w_p is a path-label of an implicit node of $\mathcal{T}(x)$, $std(w) \geq 0$.

Proof.

For any occurring w we have $f(w_i) \geq f(w_s)$.
This implies $f(w_p) \geq \frac{f(w_s)}{f(w_i)} \times f(w_p) = E(w)$.

Useful Properties

Lemma (Occurring)

Let w be a word occurring in x and $\mathcal{T}(x)$ be the suffix tree of x. Then, if w_p is a path-label of an implicit node of $\mathcal{T}(x)$, $std(w) \geq 0$.

Proof.

For any occurring w we have $f(w_i) \geq f(w_s)$.
This implies $f(w_p) \geq \frac{f(w_s)}{f(w_i)} \times f(w_p) = E(w)$.
If w_p is a path-label of an implicit node then $f(w_p) = f(w)$.

Useful Properties

Lemma (Occurring)

Let w be a word occurring in x and $\mathcal{T}(x)$ be the suffix tree of x. Then, if w_p is a path-label of an implicit node of $\mathcal{T}(x)$, $std(w) \geq 0$.

Proof.

For any occurring w we have $f(w_i) \geq f(w_s)$.
This implies $f(w_p) \geq \frac{f(w_s)}{f(w_i)} \times f(w_p) = E(w)$.
If w_p is a path-label of an implicit node then $f(w_p) = f(w)$.
Hence $\frac{f(w)-E(w)}{\max\{\sqrt{E(w)},1\}} =$

Useful Properties

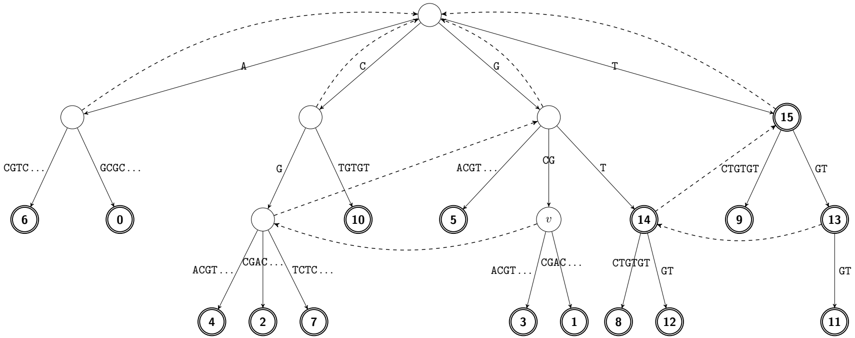
Lemma (Occurring)

Let w be a word occurring in x and $\mathcal{T}(x)$ be the suffix tree of x. Then, if w_p is a path-label of an implicit node of $\mathcal{T}(x)$, $std(w) \geq 0$.

Proof.

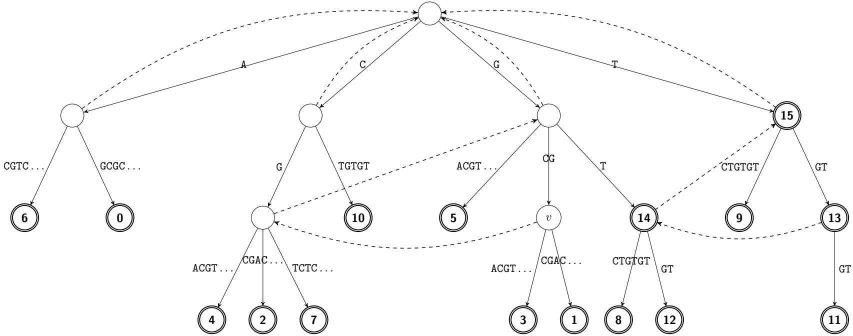
For any occurring w we have $f(w_i) \geq f(w_s)$.
This implies $f(w_p) \geq \frac{f(w_s)}{f(w_i)} \times f(w_p) = E(w)$.
If w_p is a path-label of an implicit node then $f(w_p) = f(w)$.
Hence $\frac{f(w)-E(w)}{\max\{\sqrt{E(w)},1\}} = \frac{f(w_p)-E(w)}{\max\{\sqrt{E(w)},1\}} \geq 0$. □

Example



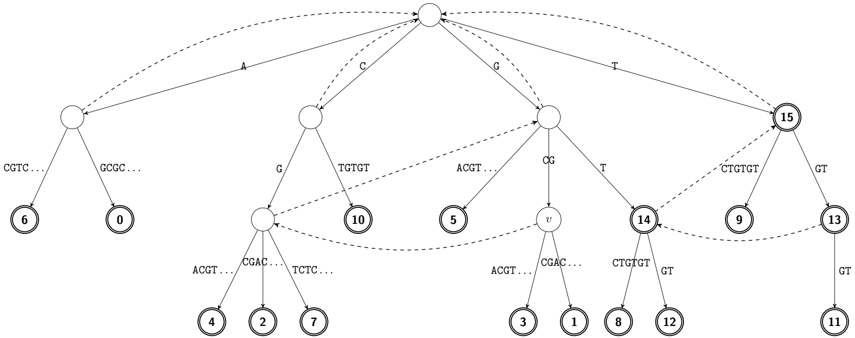
A
C
G
T
15
CGTC...
GCGC...
G
TGTGT
ACGT...
CG
T
CTGTGT
GT
6
0
10
5
v
14
9
13
ACGT...
CGAC...
TCTC...
ACGT...
CGAC...
CTGTGT
GT
GT
4
2
7
3
1
8
12
11

Example



Let $k = 3$ and $\rho = -0.4$.

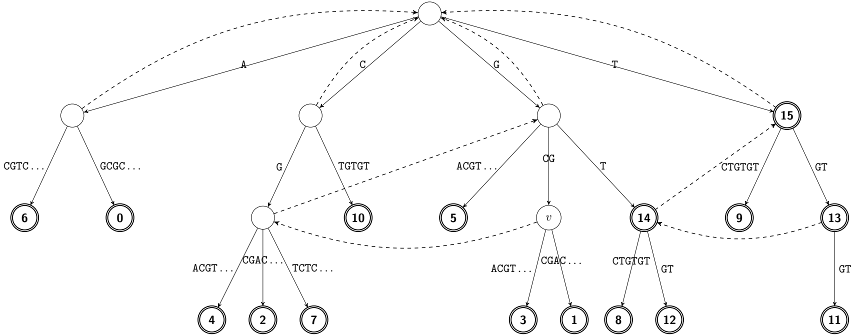
Example



Let $k = 3$ and $\rho = -0.4$.

- word $w_1 =$ CGT is an **occurring** ρ-avoided word:

Example

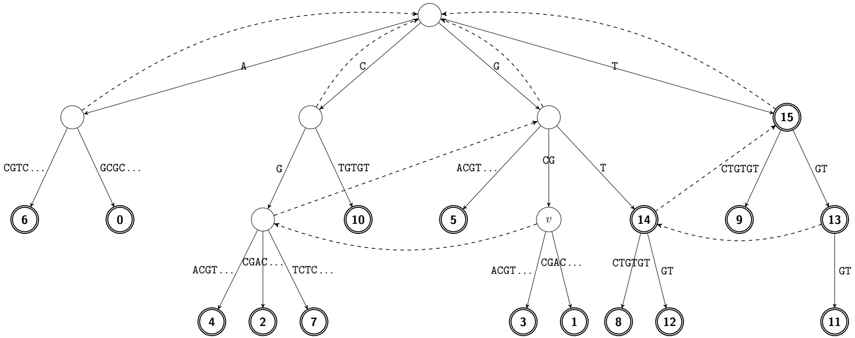


Let $k = 3$ and $\rho = -0.4$.

- word $w_1 = \texttt{CGT}$ is an **occurring** ρ-avoided word:

$$E(w_1) = 3 \times 3/6 = 1.5,\ \mathit{std}(w_1) = (1 - 1.5)/\sqrt{1.5} = -0.408248.$$

Example



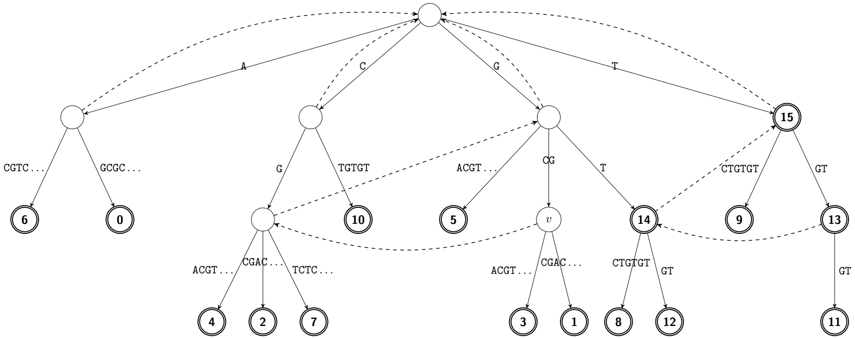
Let $k = 3$ and $\rho = -0.4$.

- word $w_1 =$ CGT is an **occurring** ρ-avoided word:

$$E(w_1) = 3 \times 3/6 = 1.5, \; \mathit{std}(w_1) = (1 - 1.5)/\sqrt{1.5} = -0.408248.$$

- word $w_2 =$ AGT is an **absent** ρ-avoided word:

Example



Let $k = 3$ and $\rho = -0.4$.

- word $w_1 =$ CGT is an **occurring** ρ-avoided word:

$$E(w_1) = 3 \times 3/6 = 1.5,\ \mathit{std}(w_1) = (1 - 1.5)/\sqrt{1.5} = -0.408248.$$

- word $w_2 =$ AGT is an **absent** ρ-avoided word:

$$E(w_2) = 1 \times 3/6 = 0.5,\ \mathit{std}(w_2) = (0 - 0.5)/1 = -0.5.$$

The Algorithm

$\textsc{AvoidedWords}(x,\ k,\ \rho)$

$\mathcal{T}(x) \leftarrow \textsc{SuffixTree}(x)$
for each node $v \in \mathcal{T}(x)$ **do**
    $\mathcal{D}(v) \leftarrow$ word-depth of v
    $\mathcal{C}(v) \leftarrow$ number of terminal nodes in the subtree rooted at v
$\textsc{AbsentAvoidedWords}(x, k, \rho)$
$\textsc{OccurringAvoidedWords}(x, k, \rho)$

The Algorithm

$\textsc{AvoidedWords}(x,\ k,\ \rho)$

$\mathcal{T}(x) \leftarrow \textsc{SuffixTree}(x)$
for each node $v \in \mathcal{T}(x)$ **do**
    $\mathcal{D}(v) \leftarrow$ word-depth of v
    $\mathcal{C}(v) \leftarrow$ number of terminal nodes in the subtree rooted at v
$\textsc{AbsentAvoidedWords}(x, k, \rho)$
$\textsc{OccurringAvoidedWords}(x, k, \rho)$

$\textsc{AbsentAvoidedWords}(x, k, \rho)$

The Algorithm

$\textsc{AvoidedWords}(x,\ k,\ \rho)$

$\mathcal{T}(x) \leftarrow \textsc{SuffixTree}(x)$
for each node $v \in \mathcal{T}(x)$ **do**
  $\mathcal{D}(v) \leftarrow$ word-depth of v
  $\mathcal{C}(v) \leftarrow$ number of terminal nodes in the subtree rooted at v
$\textsc{AbsentAvoidedWords}(x, k, \rho)$
$\textsc{OccurringAvoidedWords}(x, k, \rho)$

$\textsc{AbsentAvoidedWords}(x, k, \rho)$

- Compute all minimal absent words of length k (Lemma 1).

The Algorithm

$\textsc{AvoidedWords}(x,\ k,\ \rho)$

$\mathcal{T}(x) \leftarrow \textsc{SuffixTree}(x)$
for each node $v \in \mathcal{T}(x)$ **do**
    $\mathcal{D}(v) \leftarrow$ word-depth of v
    $\mathcal{C}(v) \leftarrow$ number of terminal nodes in the subtree rooted at v
$\textsc{AbsentAvoidedWords}(x, k, \rho)$
$\textsc{OccurringAvoidedWords}(x, k, \rho)$

$\textsc{AbsentAvoidedWords}(x, k, \rho)$

- Compute all minimal absent words of length k (Lemma 1).
- Find all nodes corresponding to w_p, w_i, and w_s.

The Algorithm

$\textsc{AvoidedWords}(x,\ k,\ \rho)$

$\mathcal{T}(x) \leftarrow \textsc{SuffixTree}(x)$
for each node $v \in \mathcal{T}(x)$ **do**
    $\mathcal{D}(v) \leftarrow$ word-depth of v
    $\mathcal{C}(v) \leftarrow$ number of terminal nodes in the subtree rooted at v
$\textsc{AbsentAvoidedWords}(x, k, \rho)$
$\textsc{OccurringAvoidedWords}(x, k, \rho)$

$\textsc{AbsentAvoidedWords}(x, k, \rho)$

- Compute all minimal absent words of length k (Lemma 1).
- Find all nodes corresponding to w_p, w_i, and w_s.
- Compute *std*(w) using their $\mathcal{C}(v)$ count.

The Algorithm

$\textsc{AvoidedWords}(x,\ k,\ \rho)$

$\mathcal{T}(x) \leftarrow \textsc{SuffixTree}(x)$
for each node $v \in \mathcal{T}(x)$ **do**
  $\mathcal{D}(v) \leftarrow$ word-depth of v
  $\mathcal{C}(v) \leftarrow$ number of terminal nodes in the subtree rooted at v
$\textsc{AbsentAvoidedWords}(x, k, \rho)$
$\textsc{OccurringAvoidedWords}(x, k, \rho)$

$\textsc{AbsentAvoidedWords}(x, k, \rho)$

- Compute all minimal absent words of length k (Lemma 1).
- Find all nodes corresponding to w_p, w_i, and w_s.
- Compute *std*(w) using their $\mathcal{C}(v)$ count.
- Note that $f(w) = 0$!

The Algorithm

$\textsc{AvoidedWords}(x,\ k,\ \rho)$

$\mathcal{T}(x) \leftarrow \textsc{SuffixTree}(x)$
for each node $v \in \mathcal{T}(x)$ **do**
    $\mathcal{D}(v) \leftarrow$ word-depth of v
    $\mathcal{C}(v) \leftarrow$ number of terminal nodes in the subtree rooted at v
$\textsc{AbsentAvoidedWords}(x, k, \rho)$
$\textsc{OccurringAvoidedWords}(x, k, \rho)$

$\textsc{AbsentAvoidedWords}(x, k, \rho)$

- Compute all minimal absent words of length k (Lemma 1).
- Find all nodes corresponding to w_p, w_i, and w_s.
- Compute *std*(w) using their $\mathcal{C}(v)$ count.
- Note that $f(w) = 0$!

$\mathcal{O}(\sigma n)$ time, $\sigma = |\Sigma|$, to compute the words [Crochemore et al, 1998];

The Algorithm

$\textsc{AvoidedWords}(x,\ k,\ \rho)$

$\mathcal{T}(x) \leftarrow \textsc{SuffixTree}(x)$
for each node $v \in \mathcal{T}(x)$ **do**
  $\mathcal{D}(v) \leftarrow$ word-depth of v
  $\mathcal{C}(v) \leftarrow$ number of terminal nodes in the subtree rooted at v
$\textsc{AbsentAvoidedWords}(x, k, \rho)$
$\textsc{OccurringAvoidedWords}(x, k, \rho)$

$\textsc{AbsentAvoidedWords}(x, k, \rho)$

- Compute all minimal absent words of length k (Lemma 1).
- Find all nodes corresponding to w_p, w_i, and w_s.
- Compute *std*(w) using their $\mathcal{C}(v)$ count.
- Note that $f(w) = 0$!

$\mathcal{O}(\sigma n)$ time, $\sigma = |\Sigma|$, to compute the words [Crochemore et al, 1998]; and $\mathcal{O}(1)$ work per word using Weighted Ancestors queries [Gawrychowski et al, 2014].

The Algorithm

$\textsc{AvoidedWords}(x,\ k,\ \rho)$

$\mathcal{T}(x) \leftarrow \textsc{SuffixTree}(x)$
for each node $v \in \mathcal{T}(x)$ **do**
    $\mathcal{D}(v) \leftarrow$ word-depth of v
    $\mathcal{C}(v) \leftarrow$ number of terminal nodes in the subtree rooted at v
$\textsc{AbsentAvoidedWords}(x, k, \rho)$
$\textsc{OccurringAvoidedWords}(x, k, \rho)$

The Algorithm

$\textsc{AvoidedWords}(x,\ k,\ \rho)$

$\mathcal{T}(x) \leftarrow \textsc{SuffixTree}(x)$
for each node $v \in \mathcal{T}(x)$ **do**
    $\mathcal{D}(v) \leftarrow$ word-depth of v
    $\mathcal{C}(v) \leftarrow$ number of terminal nodes in the subtree rooted at v
$\textsc{AbsentAvoidedWords}(x, k, \rho)$
$\textsc{OccurringAvoidedWords}(x, k, \rho)$

$\textsc{OccurringAvoidedWords}(x, k, \rho)$

The Algorithm

$\textsc{AvoidedWords}(x,\ k,\ \rho)$

$\mathcal{T}(x) \leftarrow \textsc{SuffixTree}(x)$
for each node $v \in \mathcal{T}(x)$ **do**
    $\mathcal{D}(v) \leftarrow$ word-depth of v
    $\mathcal{C}(v) \leftarrow$ number of terminal nodes in the subtree rooted at v
$\textsc{AbsentAvoidedWords}(x, k, \rho)$
$\textsc{OccurringAvoidedWords}(x, k, \rho)$

$\textsc{OccurringAvoidedWords}(x, k, \rho)$

- For occurring ones w_p is a path-label of an **explicit** node (Lemma 2).

The Algorithm

$\textsc{AvoidedWords}(x,\ k,\ \rho)$

```
T(x) ← SuffixTree(x)
for each node v ∈ T(x) do
        D(v) ← word-depth of v
        C(v) ← number of terminal nodes in the subtree rooted at v
AbsentAvoidedWords(x, k, ρ)
OccurringAvoidedWords(x, k, ρ)
```

$\textsc{OccurringAvoidedWords}(x, k, \rho)$

- For occurring ones w_p is a path-label of an **explicit** node (Lemma 2).
- Find all nodes v corresponding to w_p of length $\mathcal{D}(v) = k - 1$.

The Algorithm

$\textsc{AvoidedWords}(x,\ k,\ \rho)$

$\mathcal{T}(x) \leftarrow \textsc{SuffixTree}(x)$
for each node $v \in \mathcal{T}(x)$ **do**
    $\mathcal{D}(v) \leftarrow$ word-depth of v
    $\mathcal{C}(v) \leftarrow$ number of terminal nodes in the subtree rooted at v
$\textsc{AbsentAvoidedWords}(x, k, \rho)$
$\textsc{OccurringAvoidedWords}(x, k, \rho)$

$\textsc{OccurringAvoidedWords}(x, k, \rho)$

- For occurring ones w_p is a path-label of an **explicit** node (Lemma 2).
- Find all nodes v corresponding to w_p of length $\mathcal{D}(v) = k - 1$.
- Find all nodes corresponding to w_i using the suffix link of v.

The Algorithm

```
AvoidedWords(x, k, ρ)
  T(x) ← SuffixTree(x)
  for each node v ∈ T(x) do
        D(v) ← word-depth of v
        C(v) ← number of terminal nodes in the subtree rooted at v
  AbsentAvoidedWords(x, k, ρ)
  OccurringAvoidedWords(x, k, ρ)
```

OccurringAvoidedWords(x, k, ρ)

- For occurring ones w_p is a path-label of an **explicit** node (Lemma 2).
- Find all nodes v corresponding to w_p of length $\mathcal{D}(v) = k - 1$.
- Find all nodes corresponding to w_i using the suffix link of v.
- Find all nodes corresponding to w using the children of v.

The Algorithm

$\textsc{AvoidedWords}(x,\ k,\ \rho)$

$\mathcal{T}(x) \leftarrow \textsc{SuffixTree}(x)$
for each node $v \in \mathcal{T}(x)$ **do**
    $\mathcal{D}(v) \leftarrow$ word-depth of v
    $\mathcal{C}(v) \leftarrow$ number of terminal nodes in the subtree rooted at v
$\textsc{AbsentAvoidedWords}(x, k, \rho)$
$\textsc{OccurringAvoidedWords}(x, k, \rho)$

$\textsc{OccurringAvoidedWords}(x, k, \rho)$

- For occurring ones w_p is a path-label of an **explicit** node (Lemma 2).
- Find all nodes v corresponding to w_p of length $\mathcal{D}(v) = k - 1$.
- Find all nodes corresponding to w_i using the suffix link of v.
- Find all nodes corresponding to w using the children of v.
- Find all nodes corresponding to w_s using the suffix link of v.

The Algorithm

$\textsc{AvoidedWords}(x,\ k,\ \rho)$

```
T(x) ← SuffixTree(x)
for each node v ∈ T(x) do
    D(v) ← word-depth of v
    C(v) ← number of terminal nodes in the subtree rooted at v
AbsentAvoidedWords(x, k, ρ)
OccurringAvoidedWords(x, k, ρ)
```

$\textsc{OccurringAvoidedWords}(x, k, \rho)$

- For occurring ones w_p is a path-label of an **explicit** node (Lemma 2).
- Find all nodes v corresponding to w_p of length $\mathcal{D}(v) = k - 1$.
- Find all nodes corresponding to w_i using the suffix link of v.
- Find all nodes corresponding to w using the children of v.
- Find all nodes corresponding to w_s using the suffix link of v.
- Compute *std*(w) using their $\mathcal{C}(v)$ count.

The Algorithm

$\textsc{AvoidedWords}(x,\ k,\ \rho)$

1. $\mathcal{T}(x) \leftarrow \textsc{SuffixTree}(x)$
2. **for** each node $v \in \mathcal{T}(x)$ **do**
3. $\quad\mathcal{D}(v) \leftarrow$ word-depth of v
4. $\quad\mathcal{C}(v) \leftarrow$ number of terminal nodes in the subtree rooted at v
5. $\textsc{AbsentAvoidedWords}(x, k, \rho)$
6. $\textsc{OccurringAvoidedWords}(x, k, \rho)$

$\textsc{OccurringAvoidedWords}(x, k, \rho)$

- For occurring ones w_p is a path-label of an **explicit** node (Lemma 2).
- Find all nodes v corresponding to w_p of length $\mathcal{D}(v) = k - 1$.
- Find all nodes corresponding to w_i using the suffix link of v.
- Find all nodes corresponding to w using the children of v.
- Find all nodes corresponding to w_s using the suffix link of v.
- Compute *std*(w) using their $\mathcal{C}(v)$ count.

$\mathcal{O}(n)$ time to visit all nodes;

The Algorithm

$\textsc{AvoidedWords}(x,\ k,\ \rho)$

$\mathcal{T}(x) \leftarrow \textsc{SuffixTree}(x)$
for each node $v \in \mathcal{T}(x)$ **do**
    $\mathcal{D}(v) \leftarrow$ word-depth of v
    $\mathcal{C}(v) \leftarrow$ number of terminal nodes in the subtree rooted at v
$\textsc{AbsentAvoidedWords}(x, k, \rho)$
$\textsc{OccurringAvoidedWords}(x, k, \rho)$

$\textsc{OccurringAvoidedWords}(x, k, \rho)$

- For occurring ones w_p is a path-label of an **explicit** node (Lemma 2).
- Find all nodes v corresponding to w_p of length $\mathcal{D}(v) = k - 1$.
- Find all nodes corresponding to w_i using the suffix link of v.
- Find all nodes corresponding to w using the children of v.
- Find all nodes corresponding to w_s using the suffix link of v.
- Compute *std*(w) using their $\mathcal{C}(v)$ count.

$\mathcal{O}(n)$ time to visit all nodes; and $\mathcal{O}(1)$ work per word.

The Algorithm

$\textsc{AvoidedWords}(x,\ k,\ \rho)$

$\mathcal{T}(x) \leftarrow \textsc{SuffixTree}(x)$
for each node $v \in \mathcal{T}(x)$ **do**
    $\mathcal{D}(v) \leftarrow$ word-depth of v
    $\mathcal{C}(v) \leftarrow$ number of terminal nodes in the subtree rooted at v
$\textsc{AbsentAvoidedWords}(x, k, \rho)$
$\textsc{OccurringAvoidedWords}(x, k, \rho)$

The Algorithm

$\textsc{AvoidedWords}(x,\ k,\ \rho)$

$\mathcal{T}(x) \leftarrow \textsc{SuffixTree}(x)$
for each node $v \in \mathcal{T}(x)$ **do**
    $\mathcal{D}(v) \leftarrow$ word-depth of v
    $\mathcal{C}(v) \leftarrow$ number of terminal nodes in the subtree rooted at v
$\textsc{AbsentAvoidedWords}(x, k, \rho)$
$\textsc{OccurringAvoidedWords}(x, k, \rho)$

Theorem

Alg. $\textsc{AvoidedWords}$ *solves problem* $\textsc{AvoidedWordsComputation}$ *in time and space* $\mathcal{O}(n)$ *for constant-sized alphabets.*
For integer alphabets, the algorithm solves the problem in time $\mathcal{O}(\sigma n)$*.*

Generalisation

ALLAVOIDEDWORDSCOMPUTATION
Input: A word x of length n and a $\rho < 0$
Output: All ρ-avoided words in x

Generalisation

ALLAVOIDEDWORDSCOMPUTATION
Input: A word x of length n and a $\rho < 0$
Output: All ρ-avoided words in x

Theorem

Given a word x of length n over an integer alphabet of size σ and a real number $\rho < 0$, all ρ-avoided words in x can be computed in time $\mathcal{O}(\sigma n)$. This is time-optimal if $2 \leq \sigma \leq n$.

Generalisation

AllAvoidedWordsComputation
Input: A word x of length n and a $\rho < 0$
Output: All ρ-avoided words in x

Theorem

Given a word x of length n over an integer alphabet of size σ and a real number $\rho < 0$, all ρ-avoided words in x can be computed in time $\mathcal{O}(\sigma n)$. This is time-optimal if $2 \leq \sigma \leq n$.

Proof.

Generalisation

AllAvoidedWordsComputation
Input: A word x of length n and a $\rho < 0$
Output: All ρ-avoided words in x

Theorem

Given a word x of length n over an integer alphabet of size σ and a real number $\rho < 0$, all ρ-avoided words in x can be computed in time $\mathcal{O}(\sigma n)$. This is time-optimal if $2 \leq \sigma \leq n$.

Proof.

For **occurring** words this is trivial in $\mathcal{O}(n)$: linear # of explicit nodes.

Generalisation

ALLAVOIDEDWORDSCOMPUTATION
Input: A word x of length n and a $\rho < 0$
Output: All ρ-avoided words in x

Theorem

Given a word x of length n over an integer alphabet of size σ and a real number $\rho < 0$, all ρ-avoided words in x can be computed in time $\mathcal{O}(\sigma n)$. This is time-optimal if $2 \leq \sigma \leq n$.

Proof.

For **occurring** words this is trivial in $\mathcal{O}(n)$: linear # of explicit nodes.
The $\mathcal{O}(\sigma n)$ bound on the number of minimal **absent** words is tight.

Generalisation

ALLAVOIDEDWORDSCOMPUTATION
Input: A word x of length n and a $\rho < 0$
Output: All ρ-avoided words in x

Theorem

Given a word x of length n over an integer alphabet of size σ and a real number $\rho < 0$, all ρ-avoided words in x can be computed in time $\mathcal{O}(\sigma n)$. This is time-optimal if $2 \leq \sigma \leq n$.

Proof.

For **occurring** words this is trivial in $\mathcal{O}(n)$: linear # of explicit nodes.
The $\mathcal{O}(\sigma n)$ bound on the number of minimal **absent** words is tight.
Consider $\rho \geq -\frac{1}{n}$.

Generalisation

ALLAVOIDEDWORDSCOMPUTATION
Input: A word x of length n and a $\rho < 0$
Output: All ρ-avoided words in x

Theorem

Given a word x of length n over an integer alphabet of size σ and a real number $\rho < 0$, all ρ-avoided words in x can be computed in time $\mathcal{O}(\sigma n)$. This is time-optimal if $2 \leq \sigma \leq n$.

Proof.

For **occurring** words this is trivial in $\mathcal{O}(n)$: linear # of explicit nodes.
The $\mathcal{O}(\sigma n)$ bound on the number of minimal **absent** words is tight.
Consider $\rho \geq -\frac{1}{n}$.
For every (minimal absent word) w we have $E(w) \geq \frac{1}{n}$.

Generalisation

ALLAVOIDEDWORDSCOMPUTATION
Input: A word x of length n and a $\rho < 0$
Output: All ρ-avoided words in x

Theorem

Given a word x of length n over an integer alphabet of size σ and a real number $\rho < 0$, all ρ-avoided words in x can be computed in time $\mathcal{O}(\sigma n)$. This is time-optimal if $2 \le \sigma \le n$.

Proof.

For **occurring** words this is trivial in $\mathcal{O}(n)$: linear # of explicit nodes.
The $\mathcal{O}(\sigma n)$ bound on the number of minimal **absent** words is tight.
Consider $\rho \ge -\frac{1}{n}$.
For every (minimal absent word) w we have $E(w) \ge \frac{1}{n}$.
Since $f(w) = 0$ we have *std*$(w) = \frac{-E(w)}{\max\{\sqrt{E(w)},1\}}$

Generalisation

AllAvoidedWordsComputation
Input: A word x of length n and a $\rho < 0$
Output: All ρ-avoided words in x

Theorem

Given a word x of length n over an integer alphabet of size σ and a real number $\rho < 0$, all ρ-avoided words in x can be computed in time $\mathcal{O}(\sigma n)$. This is time-optimal if $2 \leq \sigma \leq n$.

Proof.

For **occurring** words this is trivial in $\mathcal{O}(n)$: linear # of explicit nodes.
The $\mathcal{O}(\sigma n)$ bound on the number of minimal **absent** words is tight.
Consider $\rho \geq -\frac{1}{n}$.
For every (minimal absent word) w we have $E(w) \geq \frac{1}{n}$.
Since $f(w) = 0$ we have $std(w) = \frac{-E(w)}{\max\{\sqrt{E(w)},1\}} \leq -\frac{1}{n} \leq \rho$. $\square$

Conclusion and Open Problem

Conclusion and Open Problem

Time-optimal algorithm for constant-sized and integer alphabets.

Conclusion and Open Problem

Time-optimal algorithm for constant-sized and integer alphabets.

Can the problem be solved in $\mathcal{O}(n + \textit{output})$ for integer alphabets?

Conclusion and Open Problem

Time-optimal algorithm for constant-sized and integer alphabets.

Can the problem be solved in $\mathcal{O}(n + \textit{output})$ for integer alphabets?

Free open-source implementation: `http://github.com/solonas13/aw`

Conclusion and Open Problem

Time-optimal algorithm for constant-sized and integer alphabets.

Can the problem be solved in $\mathcal{O}(n + \textit{output})$ for integer alphabets?

Free open-source implementation: `http://github.com/solonas13/aw`

Paper is on arXiv (to appear in WABI 2016):
Y. Almirantis, P. Charalampopoulos, J. Gao, C. S. Iliopoulos, M, Mohamed,
S. P. Pissis, and D. Polychronopoulos: Optimal Computation of Avoided Words.

Conclusion and Open Problem

Time-optimal algorithm for constant-sized and integer alphabets.

Can the problem be solved in $\mathcal{O}(n + \textit{output})$ for integer alphabets?

Free open-source implementation: `http://github.com/solonas13/aw`

Paper is on arXiv (to appear in WABI 2016):
Y. Almirantis, P. Charalampopoulos, J. Gao, C. S. Iliopoulos, M, Mohamed, S. P. Pissis, and D. Polychronopoulos: Optimal Computation of Avoided Words.

Thanks!