Discovering Hidden Repetitions

Florin Manea[a]
Joint work with Paweł Gawrychowski[b], Robert Mercaş[c], Dirk Nowotka[a]

[a]Christian-Albrechts-Universität zu Kiel
[b]Max-Planck-Institute für Informatik Saarbrücken
[c]Otto-von-Guericke-Universität Magdeburg

Toronto, April 2013

Pseudo-repetitions

A word w is

- *repetition*: $w = t^n$, for some proper prefix t (called root)
  primitive word: not a repetition.
- *f-repetition*: $w \in t\{t, f(t)\}^*$, for some proper prefix t (called root)
  f-primitive word: not an f-repetition.

Pseudo-repetitions

A word w is

- *repetition*: $w = t^n$, for some proper prefix t (called root)
  primitive word: not a repetition.
- *f-repetition*: $w \in t\{t, f(t)\}^*$, for some proper prefix t (called root)
  f-primitive word: not an f-repetition.

Example

ACGTAC

- *primitive* from the classical point of view

Pseudo-repetitions

A word w is

- *repetition*: $w = t^n$, for some proper prefix t (called root)
  primitive word: not a repetition.
- *f-repetition*: $w \in t\{t, f(t)\}^*$, for some proper prefix t (called root)
  f-primitive word: not an f-repetition.

Example

$$ACGTAC$$

- *primitive* from the classical point of view
- *f-primitive* for morphism f with $f(A) = T$, $f(C) = G$

Pseudo-repetitions

A word w is

- *repetition*: $w = t^n$, for some proper prefix t (called root)
  primitive word: not a repetition.
- *f-repetition*: $w \in t\{t, f(t)\}^*$, for some proper prefix t (called root)
  f-primitive word: not an f-repetition.

Example

ACGTAC

- *primitive* from the classical point of view
- *f-primitive* for morphism f with $f(A) = T$, $f(C) = G$
- *f-power* for antimorphism f with $f(A) = T$, $f(C) = G$:

$$ACGTAC = AC \cdot f(AC) \cdot AC$$

Why Pseudo-repetitions?

Repetitions: central in combinatorics on words and applications!

Why Pseudo-repetitions?

Repetitions: central in combinatorics on words and applications!

[Czeizler, Kari, Seki. On a special class of primitive words. TCS, 2010.]
Originated from computational biology:
– Watson-Crick complement: an antimorphic involution
– a single-stranded DNA and its complement encode the same information.

Why Pseudo-repetitions?

Repetitions: central in combinatorics on words and applications!

[Czeizler, Kari, Seki. On a special class of primitive words. TCS, 2010.]
Originated from computational biology:
– Watson-Crick complement: an antimorphic involution
– a single-stranded DNA and its complement encode the same information.

Generally: strings with intrinsic (yet, hidden) repetitive structure.

Why Pseudo-repetitions?

Repetitions: central in combinatorics on words and applications!

[Czeizler, Kari, Seki. On a special class of primitive words. TCS, 2010.]
Originated from computational biology:
– Watson-Crick complement: an antimorphic involution
– a single-stranded DNA and its complement encode the same information.

Generally: strings with intrinsic (yet, hidden) repetitive structure.
Such structures appear also in music: ternary song form.

Why Pseudo-repetitions?

Repetitions: central in combinatorics on words and applications!

[Czeizler, Kari, Seki. On a special class of primitive words. TCS, 2010.]
Originated from computational biology:
– Watson-Crick complement: an antimorphic involution
– a single-stranded DNA and its complement encode the same information.

Generally: strings with intrinsic (yet, hidden) repetitive structure.
Such structures appear also in music: ternary song form.

[Kari, Seki. An improved bound for an extension of Fine and Wilf theorem, and its optimality. Fundam. Informat. 2010.]
[Chiniforooshan, Kari, Xu. Pseudopower avoidance. Fundam. Informat., 2012.]
[Blondin Massé, Gaboury, Hallé. Pseudoperiodic words. DLT 2012]
[M., Müller, Nowotka. The avoidability of cubes under permutations. DLT 2012.]
[M., Mercas, Nowotka. F & W theorem and pseudo-repetitions. MFCS 2012.]
[Gawrychowski, M., Mercas, Nowotka, Tiseanu. Finding Pseudo-Repetitions. STACS 2013.]
[Gawrychowski, M., Nowotka. Discovering Hidden Repetitions. CiE 2013.]

Finding Pseudo-repetitions

Problem

Given $w \in V^*$ *and* f*, decide whether this word is an* f*-repetition.*

Finding Pseudo-repetitions

Problem

Given $w \in V^*$ *and* f, *decide whether this word is an* f*-repetition.*

Problem

Given $w \in V^+$, *decide whether there exists an* $f : V^* \to V^*$ *and a prefix* t *of* w *such that* $w \in t\{t, f(t)\}^+$.

Finding Pseudo-repetitions

Problem

Given $w \in V^*$ and f, decide whether this word is an f-repetition.

Problem

Given $w \in V^+$, decide whether there exists an $f : V^* \to V^*$ and a prefix t of w such that $w \in t\{t, f(t)\}^+$.

Problem

Given a word $w \in V^*$ and f,
(1) Enumerate all (i, j, ℓ), $1 \leq i, j, \ell \leq |w|$, such that there exists t with $w[i..j] \in \{t, f(t)\}^\ell$.
(2) Given k, enumerate all (i, j), $1 \leq i, j \leq |w|$, so there exists t with $w[i..j] \in \{t, f(t)\}^k$.

Basic tools

Computational model: RAM with logarithmic word size.

A word u, with $|u| = n$, over $|V| \in \mathcal{O}(n^c)$.

Build in linear time:
– suffix array data structure for u;
– data structures allowing us to answer in $\mathcal{O}(1)$ queries:
"How long is the longest common prefix of $u[i..n]$ and $u[j..n]$?", denoted $LCPref_u(i, j)$.

Basic tools

Computational model: RAM with logarithmic word size.

A word u, with $|u| = n$, over $|V| \in \mathcal{O}(n^c)$.

Build in linear time:
– suffix array data structure for u;
– data structures allowing us to answer in $\mathcal{O}(1)$ queries:
"How long is the longest common prefix of $u[i..n]$ and $u[j..n]$?", denoted $LCPref_u(i, j)$.

In our case:

- w is the input word,
- f a fixed anti-/morphism,
- $u = wf(w)$, $|u| \in \mathcal{O}(|w|)$.

Basic tools

Computational model: RAM with logarithmic word size.

A word u, with $|u| = n$, over $|V| \in \mathcal{O}(n^c)$.

Build in linear time:
– suffix array data structure for u;
– data structures allowing us to answer in $\mathcal{O}(1)$ queries:
"How long is the longest common prefix of $u[i..n]$ and $u[j..n]$?", denoted $LCPref_u(i, j)$.

In our case:

- w is the input word,
- f a fixed anti-/morphism,
- $u = wf(w)$, $|u| \in \mathcal{O}(|w|)$.
- Constant time: does $w[i..j]$ / $f(w[i..j])$ occur at position s in w?

Basic tool: Fine and Wilf Theorem

[Fine, Wilf: *Uniqueness theorem for periodic functions* (1965).]

Theorem

If $\alpha \in u\{u, v\}^*$ *and* $\beta \in v\{u, v\}^*$ *have a common prefix of length at least* $|u| + |v| - \gcd(|u|, |v|)$, *then* u *and* v *are powers of a common word.*

Basic tools

Basic structure of pseudo-repetitions (used for $y = f(x)$).

Lemma (Uniqueness-1)

x, y words over V; x, y not powers of the same word, $w \in \{x, y\}^*$.
There exists a unique decomposition of w in factors x, y. □

Basic tools

Basic structure of pseudo-repetitions (used for $y = f(x)$).

Lemma (Uniqueness-1)

x, y words over V; x, y not powers of the same word, $w \in \{x, y\}^*$. There exists a unique decomposition of w in factors x, y. □

Lemma (Uniqueness-2)

f non-erasing anti-/morphism, x, y, z words over V, $f(x) = f(z) = y$, $\{x, y\}^* x \{x, y\}^* \cap \{z, y\}^* z \{z, y\}^* \neq \emptyset$. Then $x = z$. □

Basic tools

How to find the unique decomposition?
(Take y to be the longest of x and $f(x)$.)

Lemma (Shifts)

$x, y \in V^+$, $w \in \{x, y\}^* \setminus \{x\}^*$, $|x| \leq |y|$, x, y not powers of some word.
$M = \max\{p \mid x^p$ is a prefix of $w\}$ and $N = \max\{p \mid x^p$ is a prefix of $y\}$.

We have:

- $M \geq N$.

Basic tools

How to find the unique decomposition?
(Take y to be the longest of x and $f(x)$.)

Lemma (Shifts)

$x, y \in V^+$, $w \in \{x, y\}^* \setminus \{x\}^*$, $|x| \leq |y|$, x, y *not powers of some word.*
$M = \max\{p \mid x^p$ *is a prefix of* $w\}$ *and* $N = \max\{p \mid x^p$ *is a prefix of* $y\}$.

We have:

- $M \geq N$.
- *If* $M = N$ *then* $w \in y\{x, y\}^*$ *holds.*

Basic tools

How to find the unique decomposition?
(Take y to be the longest of x and $f(x)$.)

Lemma (Shifts)

$x, y \in V^+$, $w \in \{x, y\}^* \setminus \{x\}^*$, $|x| \le |y|$, x, y not powers of some word.
$M = \max\{p \mid x^p$ is a prefix of $w\}$ and $N = \max\{p \mid x^p$ is a prefix of $y\}$.

We have:

- $M \ge N$.
- *If $M = N$ then $w \in y\{x, y\}^*$ holds.*
- *If $M > N$ then exactly one of the following holds:*
  – $w \in x^{M-N} y\{x, y\}^* \setminus x^{M-N-1} yxV^*$,
  – $w \in x^{M-N-1} y\{x, y\}^+ \setminus x^{M-N} yV^*$ *and* $N > 0$.

□

Deciding whether w is an f-repetition

1. Test whether there exists x such that $w = x^k$, with $k \geq 2$.

Deciding whether w is an f-repetition

1. Test whether there exists x such that $w = x^k$, with $k \geq 2$.
2. For all $t = w[1..i]$, $|f(t)| \geq 1$, t, $f(t)$ not powers of some $x \in V^*$ do 3&4.
3. Let x be the shortest of t and $f(t)$, and y the longest. Apply Shifts Lemma!

Deciding whether w is an f-repetition

1. Test whether there exists x such that $w = x^k$, with $k \geq 2$.
2. For all $t = w[1..i]$, $|f(t)| \geq 1$, t, $f(t)$ not powers of some $x \in V^*$ do 3&4.
3. Let x be the shortest of t and $f(t)$, and y the longest. Apply Shifts Lemma!
4. We construct a maximal prefix $w[i+1..s-1] \in \{x, y\}^*$ of $w[i+1..n]$:
   – Initially, $s = i + 1$.
   – Let $M = \max\{p \mid x^p \text{ prefix of } w[s..n]\}$, $N = \max\{p \mid x^p \text{ prefix of } y\}$;
   – If $w[s..n] = x^M$, we are done!
   – If $x^{M-N}y$ occurs at position s, shift $s+ = (M - N)|x| + |y|$, iterate;
   – If $M > N$ and $x^{M-N-1}yx$ occurs at s, shift $s+ = (M - N - 1)|x| + |y|$, iterate;

Deciding whether w is an f-repetition

1. Test whether there exists x such that $w = x^k$, with $k \geq 2$.
2. For all $t = w[1..i]$, $|f(t)| \geq 1$, t, $f(t)$ not powers of some $x \in V^*$ do 3&4.
3. Let x be the shortest of t and $f(t)$, and y the longest. Apply Shifts Lemma!
4. We construct a maximal prefix $w[i+1..s-1] \in \{x, y\}^*$ of $w[i+1..n]$:
   – Initially, $s = i + 1$.
   – Let $M = \max\{p \mid x^p$ prefix of $w[s..n]\}$, $N = \max\{p \mid x^p$ prefix of $y\}$;
   – If $w[s..n] = x^M$, we are done!
   – If $x^{M-N}y$ occurs at position s, shift $s+ = (M-N)|x| + |y|$, iterate;
   – If $M > N$ and $x^{M-N-1}yx$ occurs at s, shift $s+ = (M-N-1)|x| + |y|$, iterate;

Time complexity:
– f general $\mathcal{O}(\sum_{1 \leq i \leq n} \lfloor \frac{n}{i} \rfloor) \subseteq \mathcal{O}(n \log n)$.

Deciding whether w is an f-repetition

1. Test whether there exists x such that $w = x^k$, with $k \geq 2$.
2. For all $t = w[1..i]$, $|f(t)| \geq 1$, t, $f(t)$ not powers of some $x \in V^*$ do 3&4.
3. Let x be the shortest of t and $f(t)$, and y the longest. Apply Shifts Lemma!
4. We construct a maximal prefix $w[i+1..s-1] \in \{x, y\}^*$ of $w[i+1..n]$:
   – Initially, $s = i + 1$.
   – Let $M = \max\{p \mid x^p \text{ prefix of } w[s..n]\}$, $N = \max\{p \mid x^p \text{ prefix of } y\}$;
   – If $w[s..n] = x^M$, we are done!
   – If $x^{M-N}y$ occurs at position s, shift $s+ = (M-N)|x| + |y|$, iterate;
   – If $M > N$ and $x^{M-N-1}yx$ occurs at s, shift $s+ = (M-N-1)|x| + |y|$, iterate;

Time complexity:
– f general $\mathcal{O}(\sum_{1 \leq i \leq n} \lfloor \frac{n}{i} \rfloor) \subseteq \mathcal{O}(n \log n)$.
– f uniform: $\mathcal{O}(\sum_{i|n} \lfloor \frac{n}{i} \rfloor) \subseteq \mathcal{O}(n \log \log n)$.

Optimal time for f uniform

- In the algorithm: $y = f(t)$ and $x = t$.
  Each shift: $|t^k f(t)|$. But k can be 0...

Optimal time for f uniform

- In the algorithm: $y = f(t)$ and $x = t$.
  Each shift: $|t^k f(t)|$. But k can be 0...
- Idea: shift with a word from $\{t, f(t)\}^\alpha$, for some fixed α depending on n but not on t.

Optimal time for f uniform

- In the algorithm: $y = f(t)$ and $x = t$.
  Each shift: $|t^k f(t)|$. But k can be 0...
- Idea: shift with a word from $\{t, f(t)\}^\alpha$, for some fixed α depending on n but not on t.
- Consequence: for each t we do $\frac{n}{\alpha|t|}$ steps...
- ... the overall complexity $\mathcal{O}(\frac{n \log \log n}{\alpha})$.

Optimal time for f uniform

- In the algorithm: $y = f(t)$ and $x = t$.
  Each shift: $|t^k f(t)|$. But k can be 0...
- Idea: shift with a word from $\{t, f(t)\}^\alpha$, for some fixed α depending on n but not on t.
- Consequence: for each t we do $\frac{n}{\alpha|t|}$ steps...
- ... the overall complexity $\mathcal{O}(\frac{n \log\log n}{\alpha})$.
- Linear time: $\alpha = \lceil \log\log n \rceil$.

Optimal time for f uniform

- In the algorithm: $y = f(t)$ and $x = t$.
  Each shift: $|t^k f(t)|$. But k can be 0...
- Idea: shift with a word from $\{t, f(t)\}^\alpha$, for some fixed α depending on n but not on t.
- Consequence: for each t we do $\frac{n}{\alpha|t|}$ steps...
- ... the overall complexity $\mathcal{O}(\frac{n \log\log n}{\alpha})$.
- Linear time: $\alpha = \lceil \log\log n \rceil$.
- Doable: preprocessing + careful organisation of data ...

Summary

Theorem (STACS 2013)

Given $w \in V^*$ and $f : V^* \to V^*$ be a constant size anti-/morphism. One can decide whether $w \in t\{t, f(t)\}^+$ in $\mathcal{O}(n \log n)$ time. If f is uniform we only need $\mathcal{O}(n)$ time. □

Summary

Theorem (STACS 2013)

Given $w \in V^*$ and $f : V^* \to V^*$ be a constant size anti-/morphism. One can decide whether $w \in t\{t, f(t)\}^+$ in $\mathcal{O}(n \log n)$ time. If f is uniform we only need $\mathcal{O}(n)$ time. □

Theorem (STACS 2013)

Given $w \in V^*$ and $f : V^* \to V^*$ be a constant size anti-/morphism, we decide whether $w \in \{t, f(t)\}\{t, f(t)\}^+$ in $\mathcal{O}(n^{1+\frac{1}{\log \log n}} \log n)$ time. If f is non-erasing we solve the problem in $\mathcal{O}(n \log n)$ time, while when f is uniform we only need $\mathcal{O}(n)$ time. □

The second problem

Given $w \in V^+$, decide whether there exists an anti-/morphism $f : V^* \to V^*$ and a prefix t of w such that $w \in t\{t, f(t)\}^+$.

Theorem (CiE 2013)

Given a word w and a vector T of $|V|$ numbers, we decide whether there exists an anti-/morphism f of length type T such that $w \in t\{t, f(t)\}^+$ in $\mathcal{O}(n(\log n)^2)$ time. If T defines uniform anti-/morphisms: $\mathcal{O}(n)$ time.

The second problem

Given $w \in V^+$, decide whether there exists an anti-/morphism $f : V^* \to V^*$ and a prefix t of w such that $w \in t\{t, f(t)\}^+$.

Theorem (CiE 2013)

Given a word w and a vector T of $|V|$ numbers, we decide whether there exists an anti-/morphism f of length type T such that $w \in t\{t, f(t)\}^+$ in $\mathcal{O}(n(\log n)^2)$ time. If T defines uniform anti-/morphisms: $\mathcal{O}(n)$ time.

Theorem (CiE 2013)

For a word $w \in V^+$, deciding the existence of $f : V^* \to V^*$ and a prefix t of w such that $w \in t\{t, f(t)\}^+$ with $|t| \geq 2$ (respectively, $w \in t\{t, f(t)\}\{t, f(t)\}^+$) takes linear time (respectively, is NP-complete) in the general case, is NP-complete for f non-erasing, and takes $\mathcal{O}(n^2)$ time for f uniform.

Repetitive factors

Given a word $w \in V^*$ and f,
(1) Enumerate all (i, j, ℓ), $1 \leq i, j, \ell \leq |w|$, such that there exists t with $w[i..j] \in \{t, f(t)\}^\ell$.
(2) Given ℓ, enumerate all (i, j), $1 \leq i, j \leq |w|$, so there exists t with $w[i..j] \in \{t, f(t)\}^k$.

Repetitive factors

Given a word $w \in V^*$ and f,
(1) Enumerate all (i, j, ℓ), $1 \leq i, j, \ell \leq |w|$, such that there exists t with $w[i..j] \in \{t, f(t)\}^\ell$.
(2) Given ℓ, enumerate all (i, j), $1 \leq i, j \leq |w|$, so there exists t with $w[i..j] \in \{t, f(t)\}^k$.

General approach:

Construct data structures enabling us to answer in constant time queries $rep(i, j, \ell)$:
"Is there $t \in V^*$ such that $w[i..j] \in \{t, f(t)\}^\ell$?",
for all $1 \leq i \leq j \leq |w|$ and $1 \leq \ell \leq |w|$.

Repetitive factors

Given a word $w \in V^*$ and f,
(1) Enumerate all (i, j, ℓ), $1 \leq i, j, \ell \leq |w|$, such that there exists t with $w[i..j] \in \{t, f(t)\}^{\ell}$.
(2) Given ℓ, enumerate all (i, j), $1 \leq i, j \leq |w|$, so there exists t with $w[i..j] \in \{t, f(t)\}^{k}$.

General approach:

Construct data structures enabling us to answer in constant time queries $rep(i, j, \ell)$:
"Is there $t \in V^*$ such that $w[i..j] \in \{t, f(t)\}^{\ell}$?",
for all $1 \leq i \leq j \leq |w|$ and $1 \leq \ell \leq |w|$.

Second question: we answer queries $rep(i, j, \ell)$ for a fixed ℓ, given as input together with w.

Results (STACS 2013)

Building the data structures (answer queries for all ℓ, resp. for given ℓ)

- f general: $\mathcal{O}(n^{3.5})$, resp. $\mathcal{O}(n^2\ell)$.
- f non-erasing: $\mathcal{O}(n^3)$, resp. $\mathcal{O}(n^2)$.
- f literal: $\mathcal{O}(n^2)$, resp. $\mathcal{O}(n^2)$.

Tools: combinatorics on words (the Uniqueness Lemmas) + number theoretic algorithms + data structures.

Results (STACS 2013)

Building the data structures (answer queries for all ℓ, resp. for given ℓ)

- f general: $\mathcal{O}(n^{3.5})$, resp. $\mathcal{O}(n^2\ell)$.
- f non-erasing: $\mathcal{O}(n^3)$, resp. $\mathcal{O}(n^2)$.
- f literal: $\mathcal{O}(n^2)$, resp. $\mathcal{O}(n^2)$.

Tools: combinatorics on words (the Uniqueness Lemmas) + number theoretic algorithms + data structures.

Finding the set of all ℓ-repetitive factors (for all ℓ, resp. for a given ℓ):

- f general: $\mathcal{O}(n^{3.5})$, resp. $\mathcal{O}(n^2\ell)$.
- f non-erasing: $\underline{\Theta(n^3)}$, resp. $\underline{\Theta(n^2)}$.
- f literal: $\underline{\Theta(n^2 \log n)}$, resp. $\underline{\Theta(n^2)}$.

Highlighted bounds: no other algorithm performs better in the worst case.

THANK YOU!