Whitehead's classical algorithm and a modern version in polynomial time

Enric Ventura

Departament de Matemàtica Aplicada III
Universitat Politècnica de Catalunya

&

CRM-Montreal

McGill seminar, Montreal

Sep. 30th, 2010.

Outline

Outline

Definitions and notation

- $A = \{a_1, \ldots, a_k\}$ is a finite alphabet (n letters).
- $A^{\pm 1} = A \cup A^{-1} = \{a_1, a_1^{-1}, \ldots, a_k, a_k^{-1}\}$.
- Usually, $A = \{a, b, c\}$.
- $(A^{\pm 1})^*$ the free monoid on $A^{\pm 1}$ (words on $A^{\pm 1}$).
- $F_A = (A^{\pm 1})^* / \sim$ is the free group on A (words on $A^{\pm 1}$ modulo reduction).
- Every $w \in A^*$ has a unique reduced form,
- 1 denotes the empty word, and $|\cdot|$ the (shortest) length in F_A:
  $|1| = 0, \quad |aba^{-1}| = |abbb^{-1}a^{-1}| = 3, \quad |uv| \leqslant |u| + |v|$.
- $\|\cdot\|$ denotes the (shortest) length in the conjugacy class (i.e. cyclically):
  $\|abbb^{-1}a^{-1}\| = 1$.
- $Aut(F_A)$ and $End(F_A)$ as usual.

Definitions and notation

- $A = \{a_1, \ldots, a_k\}$ is a finite alphabet (n letters).
- $A^{\pm 1} = A \cup A^{-1} = \{a_1, a_1^{-1}, \ldots, a_k, a_k^{-1}\}$.
- Usually, $A = \{a, b, c\}$.
- $(A^{\pm 1})^*$ the free monoid on $A^{\pm 1}$ (words on $A^{\pm 1}$).
- $F_A = (A^{\pm 1})^* / \sim$ is the free group on A (words on $A^{\pm 1}$ modulo reduction).
- Every $w \in A^*$ has a unique reduced form,
- 1 denotes the empty word, and $|\cdot|$ the (shortest) length in F_A: $|1| = 0, \quad |aba^{-1}| = |abbb^{-1}a^{-1}| = 3, \quad |uv| \leqslant |u| + |v|$.
- $\|\cdot\|$ denotes the (shortest) length in the conjugacy class (i.e. cyclically): $\|abbb^{-1}a^{-1}\| = 1$.
- $Aut(F_A)$ and $End(F_A)$ as usual.

Definitions and notation

- $A = \{a_1, \ldots, a_k\}$ is a finite alphabet (n letters).
- $A^{\pm 1} = A \cup A^{-1} = \{a_1, a_1^{-1}, \ldots, a_k, a_k^{-1}\}$.
- Usually, $A = \{a, b, c\}$.
- $(A^{\pm 1})^*$ the free monoid on $A^{\pm 1}$ (words on $A^{\pm 1}$).
- $F_A = (A^{\pm 1})^* / \sim$ is the free group on A (words on $A^{\pm 1}$ modulo reduction).
- Every $w \in A^*$ has a unique reduced form,
- 1 denotes the empty word, and $|\cdot|$ the (shortest) length in F_A:
  $|1| = 0, \quad |aba^{-1}| = |abbb^{-1}a^{-1}| = 3, \quad |uv| \leqslant |u| + |v|$.
- $\|\cdot\|$ denotes the (shortest) length in the conjugacy class (i.e. cyclically):
  $\|abbb^{-1}a^{-1}\| = 1$.
- $Aut(F_A)$ and $End(F_A)$ as usual.

Definitions and notation

- $A = \{a_1, \ldots, a_k\}$ is a finite alphabet (n letters).
- $A^{\pm 1} = A \cup A^{-1} = \{a_1, a_1^{-1}, \ldots, a_k, a_k^{-1}\}$.
- Usually, $A = \{a, b, c\}$.
- $(A^{\pm 1})^*$ the free monoid on $A^{\pm 1}$ (words on $A^{\pm 1}$).
- $F_A = (A^{\pm 1})^* / \sim$ is the free group on A (words on $A^{\pm 1}$ modulo reduction).
- Every $w \in A^*$ has a unique reduced form,
- 1 denotes the empty word, and $|\cdot|$ the (shortest) length in F_A:
  $|1| = 0, \quad |aba^{-1}| = |abbb^{-1}a^{-1}| = 3, \quad |uv| \leqslant |u| + |v|$.
- $\|\cdot\|$ denotes the (shortest) length in the conjugacy class (i.e. cyclically):
  $\|abbb^{-1}a^{-1}\| = 1$.
- $Aut(F_A)$ and $End(F_A)$ as usual.

Definitions and notation

- $A = \{a_1, \ldots, a_k\}$ is a finite alphabet (n letters).
- $A^{\pm 1} = A \cup A^{-1} = \{a_1, a_1^{-1}, \ldots, a_k, a_k^{-1}\}$.
- Usually, $A = \{a, b, c\}$.
- $(A^{\pm 1})^*$ the free monoid on $A^{\pm 1}$ (words on $A^{\pm 1}$).
- $F_A = (A^{\pm 1})^* / \sim$ is the free group on A (words on $A^{\pm 1}$ modulo reduction).
- Every $w \in A^*$ has a unique reduced form,
- 1 denotes the empty word, and $|\cdot|$ the (shortest) length in F_A:
  $|1| = 0, \quad |aba^{-1}| = |abbb^{-1}a^{-1}| = 3, \quad |uv| \leqslant |u| + |v|$.
- $\|\cdot\|$ denotes the (shortest) length in the conjugacy class (i.e. cyclically):
  $\|abbb^{-1}a^{-1}\| = 1$.
- $Aut(F_A)$ and $End(F_A)$ as usual.

Definitions and notation

- $A = \{a_1, \ldots, a_k\}$ is a finite alphabet (n letters).
- $A^{\pm 1} = A \cup A^{-1} = \{a_1, a_1^{-1}, \ldots, a_k, a_k^{-1}\}$.
- Usually, $A = \{a, b, c\}$.
- $(A^{\pm 1})^*$ the free monoid on $A^{\pm 1}$ (words on $A^{\pm 1}$).
- $F_A = (A^{\pm 1})^* / \sim$ is the free group on A (words on $A^{\pm 1}$ modulo reduction).
- Every $w \in A^*$ has a unique reduced form,
- 1 denotes the empty word, and $|\cdot|$ the (shortest) length in F_A: $|1| = 0, \quad |aba^{-1}| = |abbb^{-1}a^{-1}| = 3, \quad |uv| \leqslant |u| + |v|$.
- $\|\cdot\|$ denotes the (shortest) length in the conjugacy class (i.e. cyclically): $\|abbb^{-1}a^{-1}\| = 1$.
- $Aut(F_A)$ and $End(F_A)$ as usual.

Definitions and notation

- $A = \{a_1, \ldots, a_k\}$ is a finite alphabet (n letters).
- $A^{\pm 1} = A \cup A^{-1} = \{a_1, a_1^{-1}, \ldots, a_k, a_k^{-1}\}$.
- Usually, $A = \{a, b, c\}$.
- $(A^{\pm 1})^*$ the free monoid on $A^{\pm 1}$ (words on $A^{\pm 1}$).
- $F_A = (A^{\pm 1})^* / \sim$ is the free group on A (words on $A^{\pm 1}$ modulo reduction).
- Every $w \in A^*$ has a unique reduced form,
- 1 denotes the empty word, and $|\cdot|$ the (shortest) length in F_A:
  $|1| = 0, \quad |aba^{-1}| = |abbb^{-1}a^{-1}| = 3, \quad |uv| \leqslant |u| + |v|.$
- $\|\cdot\|$ denotes the (shortest) length in the conjugacy class (i.e. cyclically):
  $\|abbb^{-1}a^{-1}\| = 1.$
- $Aut(F_A)$ and $End(F_A)$ as usual.

Definitions and notation

- $A = \{a_1, \ldots, a_k\}$ is a finite alphabet (n letters).
- $A^{\pm 1} = A \cup A^{-1} = \{a_1, a_1^{-1}, \ldots, a_k, a_k^{-1}\}$.
- Usually, $A = \{a, b, c\}$.
- $(A^{\pm 1})^*$ the free monoid on $A^{\pm 1}$ (words on $A^{\pm 1}$).
- $F_A = (A^{\pm 1})^* / \sim$ is the free group on A (words on $A^{\pm 1}$ modulo reduction).
- Every $w \in A^*$ has a unique reduced form,
- 1 denotes the empty word, and $|\cdot|$ the (shortest) length in F_A:
  $|1| = 0, \quad |aba^{-1}| = |abbb^{-1}a^{-1}| = 3, \quad |uv| \leqslant |u| + |v|$.
- $\|\cdot\|$ denotes the (shortest) length in the conjugacy class (i.e. cyclically):
  $\|abbb^{-1}a^{-1}\| = 1$.
- $Aut(F_A)$ and $End(F_A)$ as usual.

Definitions and notation

- $A = \{a_1, \ldots, a_k\}$ is a finite alphabet (n letters).
- $A^{\pm 1} = A \cup A^{-1} = \{a_1, a_1^{-1}, \ldots, a_k, a_k^{-1}\}$.
- Usually, $A = \{a, b, c\}$.
- $(A^{\pm 1})^*$ the free monoid on $A^{\pm 1}$ (words on $A^{\pm 1}$).
- $F_A = (A^{\pm 1})^* / \sim$ is the free group on A (words on $A^{\pm 1}$ modulo reduction).
- Every $w \in A^*$ has a unique reduced form,
- 1 denotes the empty word, and $|\cdot|$ the (shortest) length in F_A:
  $|1| = 0, \quad |aba^{-1}| = |abbb^{-1}a^{-1}| = 3, \quad |uv| \leqslant |u| + |v|$.
- $\|\cdot\|$ denotes the (shortest) length in the conjugacy class (i.e. cyclically):
  $\|abbb^{-1}a^{-1}\| = 1$.
- $Aut(F_A)$ and $End(F_A)$ as usual.

Whitehead problem

Whitehead Problem

For a group G, find an algorithm s.t. given $u, v \in G$ decides whether there exists $\varphi \in Aut(G)$ such that $\varphi(u) = v$.

Theorem (Whitehead, 30's)

Whitehead problem is solvable in F_A.

"Proof":
First part: reduce $\|u\|$ and $\|v\|$ as much as possible by applying autos:

$$u \to u_1 \to u_2 \to \cdots \to u',$$

$$v \to v_1 \to v_2 \to \cdots \to v'.$$

Second part: analyze who is image of who by some auto, in the (finite!) sphere of given radius n, $S_n = \{w \in F_k \mid \|w\| = n\}$. $\square$

Whitehead problem

Whitehead Problem

For a group G, find an algorithm s.t. given $u, v \in G$ decides whether there exists $\varphi \in Aut(G)$ such that $\varphi(u) = v$.

Theorem (Whitehead, 30's)

Whitehead problem is solvable in F_A.

"Proof":
First part: reduce $\|u\|$ and $\|v\|$ as much as possible by applying autos:

$$u \to u_1 \to u_2 \to \cdots \to u',$$

$$v \to v_1 \to v_2 \to \cdots \to v'.$$

Second part: analyze who is image of who by some auto, in the (finite!) sphere of given radius n, $S_n = \{w \in F_k \mid \|w\| = n\}$. $\square$

Whitehead problem

Whitehead Problem

For a group G, find an algorithm s.t. given $u, v \in G$ decides whether there exists $\varphi \in Aut(G)$ such that $\varphi(u) = v$.

Theorem (Whitehead, 30's)

Whitehead problem is solvable in F_A.

"Proof":

First part: reduce $\|u\|$ and $\|v\|$ as much as possible by applying autos:

$$u \to u_1 \to u_2 \to \cdots \to u',$$

$$v \to v_1 \to v_2 \to \cdots \to v'.$$

Second part: analyze who is image of who by some auto, in the (finite!) sphere of given radius n, $S_n = \{w \in F_k \mid \|w\| = n\}$. □

Whitehead problem

Whitehead Problem

For a group G, find an algorithm s.t. given $u, v \in G$ decides whether there exists $\varphi \in Aut(G)$ such that $\varphi(u) = v$.

Theorem (Whitehead, 30's)

Whitehead problem is solvable in F_A.

"Proof":
First part: reduce $\|u\|$ and $\|v\|$ as much as possible by applying autos:

$$u \to u_1 \to u_2 \to \cdots \to u',$$

$$v \to v_1 \to v_2 \to \cdots \to v'.$$

Second part: analyze who is image of who by some auto, in the (finite!) sphere of given radius n, $S_n = \{w \in F_k \mid \|w\| = n\}$. □

Whitehead minimization problem

Let us concentrate in the first part:

Whitehead Minimization Problem (WMP)

Given $u \in F_A$, find $\varphi \in Aut(F_A)$ such that $\|\varphi(u)\|$ is minimal.

Lemma (Whitehead)

Let $u \in F_A$. If $\exists\, \varphi \in Aut(F_A)$ such that $\|\varphi(u)\| < \|u\|$ then $\exists$ a Whitehead automorphism α such that $\|\alpha(u)\| < \|u\|$.

Definition

Whitehead automorphisms are those of the form:

$$\begin{array}{rcll} F_A & \to & F_A & \\ a_i & \mapsto & a_i & \quad \text{(the multiplier)} \\ a_i \neq a_j & \mapsto & a_i^{\epsilon_j} a_j a_i^{\delta_j} & \end{array}$$

where $\epsilon_j = 0, -1$ and $\delta_j = 0, 1$ (there are $\sim k \cdot 4^k$ many, where $k = |A|$).

Whitehead minimization problem

Let us concentrate in the first part:

Whitehead Minimization Problem (WMP)

Given $u \in F_A$, find $\varphi \in Aut(F_A)$ such that $\|\varphi(u)\|$ is minimal.

Lemma (Whitehead)

Let $u \in F_A$. If $\exists\, \varphi \in Aut(F_A)$ such that $\|\varphi(u)\| < \|u\|$ then $\exists$ a Whitehead automorphism α such that $\|\alpha(u)\| < \|u\|$.

Definition

Whitehead automorphisms are those of the form:

$$\begin{array}{rcl} F_A & \to & F_A \\ a_i & \mapsto & a_i \qquad \text{(the multiplier)} \\ a_i \neq a_j & \mapsto & a_i^{\epsilon_j} a_j a_i^{\delta_j} \end{array}$$

where $\epsilon_j = 0, -1$ and $\delta_j = 0, 1$ (there are $\sim k \cdot 4^k$ many, where $k = |A|$).

Whitehead minimization problem

Let us concentrate in the first part:

Whitehead Minimization Problem (WMP)

Given $u \in F_A$, find $\varphi \in Aut(F_A)$ such that $\|\varphi(u)\|$ is minimal.

Lemma (Whitehead)

Let $u \in F_A$. If $\exists\, \varphi \in Aut(F_A)$ such that $\|\varphi(u)\| < \|u\|$ then $\exists$ a Whitehead automorphism α such that $\|\alpha(u)\| < \|u\|$.

Definition

Whitehead automorphisms are those of the form:

$$\begin{array}{rcll} F_A & \to & F_A & \\ a_i & \mapsto & a_i & \text{(the multiplier)} \\ a_i \neq a_j & \mapsto & a_i^{\epsilon_j} a_j a_i^{\delta_j} & \end{array}$$

where $\epsilon_j = 0, -1$ and $\delta_j = 0, 1$ (there are $\sim k \cdot 4^k$ many, where $k = |A|$).

Classical Whitehead's algorithm (first part)

Classical whitehead algorithm is:

- *Keep applying whitehead automorphisms to given u until finding one that decreases its cyclic length.*
- *Repeat until all whiteheads are non-decreasing.*

This is polynomial on $\|u\|$, but exponential on the ambient rank, k.

There are several recent results (theoretical, heuristic, probabilistic) suggesting that Whitehead algorithm is faster in practice.

Classical Whitehead's algorithm (first part)

Classical whitehead algorithm is:

- *Keep applying whitehead automorphisms to given u until finding one that decreases its cyclic length.*
- *Repeat until all whiteheads are non-decreasing.*

This is polynomial on $\|u\|$, but exponential on the ambient rank, k.

There are several recent results (theoretical, heuristic, probabilistic) suggesting that Whitehead algorithm is faster in practice.

Classical Whitehead's algorithm (first part)

Classical whitehead algorithm is:

- *Keep applying whitehead automorphisms to given u until finding one that decreases its cyclic length.*
- *Repeat until all whiteheads are non-decreasing.*

This is polynomial on $\|u\|$, but exponential on the ambient rank, k.

There are several recent results (theoretical, heuristic, probabilistic) suggesting that Whitehead algorithm is faster in practice.

Classical Whitehead's algorithm (first part)

Classical whitehead algorithm is:

- *Keep applying whitehead automorphisms to given u until finding one that decreases its cyclic length.*
- *Repeat until all whiteheads are non-decreasing.*

This is polynomial on $\|u\|$, but exponential on the ambient rank, k.

There are several recent results (theoretical, heuristic, probabilistic) suggesting that Whitehead algorithm is faster in practice.

Outline

Improvement

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2 k^3)$.

main idea: given $u \in F_k$, we find in polynomial time one of the whiteheads that decreases $\|u\|$ the most possible.

Key point: How does a given Whitehead automorphism α affect the length of a given word u ?

Three ingredients:

1) Codify u as its Whitehead's graph (classic in Group Theory),
2) Codify α as a cut in this graph ($\approx$ classic in Group Theory),
3) Use max-flow min-cut algorithm (classic in Computer Science),
4) ... put together and mix (new!).

Improvement

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2 k^3)$.

main idea: given $u \in F_k$, we find in polynomial time one of the whiteheads that decreases $\|u\|$ the most possible.

Key point: How does a given Whitehead automorphism α affect the length of a given word u ?

Three ingredients:

1) Codify u as its Whitehead's graph (classic in Group Theory),
2) Codify α as a cut in this graph ($\approx$ classic in Group Theory),
3) Use max-flow min-cut algorithm (classic in Computer Science),
4) ... put together and mix (new!).

Improvement

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2 k^3)$.

main idea: given $u \in F_k$, we find in polynomial time one of the whiteheads that decreases $\|u\|$ the most possible.

Key point: How does a given Whitehead automorphism α affect the length of a given word u ?

Three ingredients:

1) Codify u as its Whitehead's graph (classic in Group Theory),
2) Codify α as a cut in this graph ($\approx$ classic in Group Theory),
3) Use max-flow min-cut algorithm (classic in Computer Science),
4) ... put together and mix (new!).

Improvement

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2 k^3)$.

main idea: given $u \in F_k$, we find in polynomial time one of the whiteheads that decreases $\|u\|$ the most possible.

Key point: How does a given Whitehead automorphism α affect the length of a given word u ?

Three ingredients:

1) Codify u as its Whitehead's graph (classic in Group Theory),
2) Codify α as a cut in this graph ($\approx$ classic in Group Theory),
3) Use max-flow min-cut algorithm (classic in Computer Science),
4) ... put together and mix (new!).

Improvement

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2 k^3)$.

main idea: given $u \in F_k$, we find in polynomial time one of the whiteheads that decreases $\|u\|$ the most possible.

Key point: How does a given Whitehead automorphism α affect the length of a given word u ?

Three ingredients:

1) Codify u as its Whitehead's graph (classic in Group Theory),
2) Codify α as a cut in this graph ($\approx$ classic in Group Theory),
3) Use max-flow min-cut algorithm (classic in Computer Science),
4) ... put together and mix (new!).

Improvement

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2 k^3)$.

main idea: given $u \in F_k$, we find in polynomial time one of the whiteheads that decreases $\|u\|$ the most possible.

Key point: How does a given Whitehead automorphism α affect the length of a given word u ?

Three ingredients:

1) Codify u as its Whitehead's graph (classic in Group Theory),
2) Codify α as a cut in this graph ($\approx$ classic in Group Theory),
3) Use max-flow min-cut algorithm (classic in Computer Science),
4) ... put together and mix (new!).

Improvement

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2 k^3)$.

main idea: given $u \in F_k$, we find in polynomial time one of the whiteheads that decreases $\|u\|$ the most possible.

Key point: How does a given Whitehead automorphism α affect the length of a given word u ?

Three ingredients:

1) Codify u as its Whitehead's graph (classic in Group Theory),
2) Codify α as a cut in this graph ($\approx$ classic in Group Theory),
3) Use max-flow min-cut algorithm (classic in Computer Science),
4) ... put together and mix (new!).

Whitehead's graph

First ingredient: Whitehead's graph of a word.

Definition

Given $u \in F_k$ (cyclically reduced), its (unoriented) Whitehead graph, denoted $Wh(u)$, is:

- *vertices: $A^{\pm 1}$,*
- *edges: for every pair of (cycl.) consecutive letters $u = \cdots xy \cdots$ put an edge between x and y^{-1}.*

Example

$u = aba^{-1}c^{-1}bbabc^{-1}$,

Whitehead's graph

First ingredient: Whitehead's graph of a word.

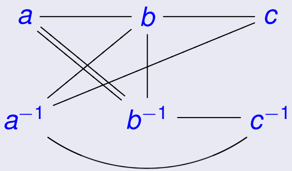
Definition

Given $u \in F_k$ (cyclically reduced), its (unoriented) Whitehead graph, denoted $Wh(u)$, is:

- *vertices: $A^{\pm 1}$,*
- *edges: for every pair of (cycl.) consecutive letters $u = \cdots xy \cdots$ put an edge between x and y^{-1}.*

Example

$u = aba^{-1}c^{-1}bbabc^{-1}$,



Cut in a graph

Second ingredient: Cut in a graph.

Definition

Given a Whitehead's automorphism α, we represent it as the (a, a^{-1})-cut

$$(T = \{a\} \cup \{\text{letters that go multiplied on the right by } a\}, a)$$

of the set $A^{\pm 1}$.

Example

$$\begin{array}{rcl} \alpha\colon \langle a, b, c\rangle = F_3 & \to & F_3 \\ a & \mapsto & ab \\ b & \mapsto & b \\ c & \mapsto & b^{-1}cb \end{array}$$

a b c

a^{-1} b^{-1} c^{-1}

Cut in a graph

Second ingredient: Cut in a graph.

Definition

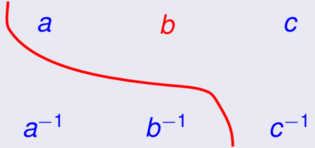
Given a Whitehead's automorphism α, we represent it as the (a, a^{-1})-cut

$$(T = \{a\} \cup \{\textit{letters that go multiplied on the right by } a\}, a)$$

of the set $A^{\pm 1}$.

Example

$$\begin{array}{rcl} \alpha\colon \langle a, b, c\rangle = F_3 & \to & F_3 \\ a & \mapsto & ab \\ b & \mapsto & b \\ c & \mapsto & b^{-1}cb \end{array}$$



Rephrasement of Wh. Lemma

Lemma (Whitehead)

Given a word $u \in F_k$ and a Whitehead automorphism α, think α as a cut in $Wh(u)$, say $\alpha = (T, a)$, and then

$$\|\alpha(u)\| - \|u\| = \mathrm{cap}(T) - \deg(a).$$

Proof: Analyzing combinatorial cases (see Lyndon-Schupp).

Rephrasement of Wh. Lemma

Lemma (Whitehead)

Given a word $u \in F_k$ and a Whitehead automorphism α, think α as a cut in $Wh(u)$, say $\alpha = (T, a)$, and then

$$\|\alpha(u)\| - \|u\| = \mathrm{cap}(T) - \deg(a).$$

Proof: Analyzing combinatorial cases (see Lyndon-Schupp).

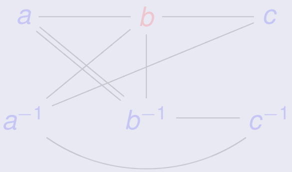
Example

Example

Consider $u = aba^{-1}c^{-1}bbabc^{-1}$ *and*

$$\alpha\colon \begin{array}{rcl} F_3 & \to & F_3 \\ a & \mapsto & ab \\ b & \mapsto & b \\ c & \mapsto & b^{-1}cb \end{array}$$

like before. We have $\alpha(u) = aba^{-1}b^{-1}c^{-1}bbbabc^{-1}b$. *Furthermore,*

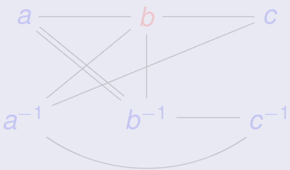


and, in fact,

$$12 - 9 = \|\alpha(u)\| - \|u\| = \mathrm{cap}(T) - \deg(b) = 7 - 4.$$

Example

Consider $u = aba^{-1}c^{-1}bbabc^{-1}$ *and*
$$\begin{array}{rccl} \alpha\colon & F_3 & \to & F_3 \\ & a & \mapsto & ab \\ & b & \mapsto & b \\ & c & \mapsto & b^{-1}cb \end{array}$$
like before. We have $\alpha(u) = aba^{-1}b^{-1}c^{-1}bbbabc^{-1}b$. *Furthermore,*



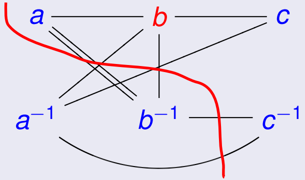
and, in fact,

$$12 - 9 = \|\alpha(u)\| - \|u\| = \mathrm{cap}(T) - \deg(b) = 7 - 4.$$

Example

Example

Consider $u = aba^{-1}c^{-1}bbabc^{-1}$ *and* $\alpha\colon \begin{array}{ccc} F_3 & \to & F_3 \\ a & \mapsto & ab \\ b & \mapsto & b \\ c & \mapsto & b^{-1}cb \end{array}$ *like before. We have* $\alpha(u) = aba^{-1}b^{-1}c^{-1}bbbabc^{-1}b$. *Furthermore,*



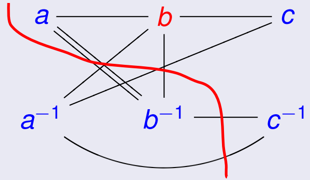
and, in fact,

$$12 - 9 = \|\alpha(u)\| - \|u\| = \mathrm{cap}(T) - \deg(b) = 7 - 4.$$

Example

Example

Consider $u = aba^{-1}c^{-1}bbabc^{-1}$ *and* $\alpha\colon \begin{array}{ccc} F_3 & \to & F_3 \\ a & \mapsto & ab \\ b & \mapsto & b \\ c & \mapsto & b^{-1}cb \end{array}$ *like before. We have* $\alpha(u) = aba^{-1}b^{-1}c^{-1}bbbabc^{-1}b$. *Furthermore,*



and, in fact,

$$12 - 9 = \|\alpha(u)\| - \|u\| = \mathrm{cap}(T) - \deg(b) = 7 - 4.$$

Max-flow min-cut algorithm

Third ingredient: Max-flow min-cut algorithm.

Hence, Whitehead's Minimization Problem reduces to:

- *run over all possible multipliers, say a, (there are $2k$),*
- *find an (a, a^{-1})-cut with minimal possible capacity.*

This can be done by using the classical max-flow min-cut algorithm ...

...which works in polynomial time w.r.t. the number of edges of the graph ($= \|u\|$) and the number of vertices ($= 2k$).

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2 k^3)$.

Max-flow min-cut algorithm

Third ingredient: Max-flow min-cut algorithm.

Hence, Whitehead's Minimization Problem reduces to:

- *run over all possible multipliers, say a, (there are $2k$),*
- *find an (a, a^{-1})-cut with minimal possible capacity.*

This can be done by using the classical max-flow min-cut algorithm ...

...which works in polynomial time w.r.t. the number of edges of the graph ($= \|u\|$) and the number of vertices ($= 2k$).

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2 k^3)$.

Max-flow min-cut algorithm

Third ingredient: Max-flow min-cut algorithm.

Hence, Whitehead's Minimization Problem reduces to:

- *run over all possible multipliers, say a, (there are $2k$),*
- *find an (a, a^{-1})-cut with minimal possible capacity.*

This can be done by using the classical max-flow min-cut algorithm ...

...which works in polynomial time w.r.t. the number of edges of the graph $(= \|u\|)$ and the number of vertices $(= 2k)$.

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2 k^3)$.

Max-flow min-cut algorithm

Third ingredient: Max-flow min-cut algorithm.

Hence, Whitehead's Minimization Problem reduces to:

- *run over all possible multipliers, say a, (there are $2k$),*
- *find an (a, a^{-1})-cut with minimal possible capacity.*

This can be done by using the classical max-flow min-cut algorithm ...

...which works in polynomial time w.r.t. the number of edges of the graph ($= \|u\|$) and the number of vertices ($= 2k$).

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2 k^3)$.

Max-flow min-cut algorithm

Third ingredient: Max-flow min-cut algorithm.

Hence, Whitehead's Minimization Problem reduces to:

- *run over all possible multipliers, say a, (there are* $2k$*),*
- *find an* (a, a^{-1})*-cut with minimal possible capacity.*

This can be done by using the classical max-flow min-cut algorithm ...

...which works in polynomial time w.r.t. the number of edges of the graph ($= \|u\|$) and the number of vertices ($= 2k$).

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k *in time* $O(n^2 k^3)$*.*

Outline

Primitivity

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2 k^3)$.

Observation

u is primitive $\Leftrightarrow$ the orbit of u contains a $\Leftrightarrow$ bottom of the orbit has length 1.

Corollary (Roig, V., Weil, 2007)

Given a word $u \in F_k$, one can check whether u is primitive in F_k in time $O(n^2 k^3)$, where $n = \|u\|$.

Primitivity

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2 k^3)$.

Observation

u is primitive $\Leftrightarrow$ the orbit of u contains a $\Leftrightarrow$ bottom of the orbit has length 1.

Corollary (Roig, V., Weil, 2007)

Given a word $u \in F_k$, one can check whether u is primitive in F_k in time $O(n^2 k^3)$, where $n = \|u\|$.

Primitivity

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2 k^3)$.

Observation

u is primitive $\Leftrightarrow$ the orbit of u contains a $\Leftrightarrow$ bottom of the orbit has length 1.

Corollary (Roig, V., Weil, 2007)

Given a word $u \in F_k$, one can check whether u is primitive in F_k in time $O(n^2 k^3)$, where $n = \|u\|$.

Deciding free-factorness

Observation

A given subgroup $H \leqslant F_k$ with basis $\{h_1, \ldots, h_r\}$ ($r(H) = r \leqslant k$) is a free factor of F_k if and only if bottom of the orbit of $(h_1, \ldots, h_r)$ has length $1 + \overset{r}{\cdots} + 1 = r$.

Corollary (Roig, V., Weil, 2007)

Given a f.g. subgroup $H \leqslant F_k$, one can check whether H is a free factor of F_k in time $O((n^2k^4 + n^3k^2)\log(nk))$, where $n = \|H\|$.

Corollary (Roig, V., Weil, 2007)

Given f.g. subgroups $H \leqslant K \leqslant F_k$, one can check whether H is a free factor of K in polynomial time w.r.t. the given generators of H and K.

Deciding free-factorness

Observation

A given subgroup $H \leqslant F_k$ with basis $\{h_1, \ldots, h_r\}$ ($r(H) = r \leqslant k$) is a free factor of F_k if and only if bottom of the orbit of $(h_1, \ldots, h_r)$ has length $1 + \overset{r}{\cdots} + 1 = r$.

Corollary (Roig, V., Weil, 2007)

Given a f.g. subgroup $H \leqslant F_k$, one can check whether H is a free factor of F_k in time $O((n^2k^4 + n^3k^2)\log(nk))$, where $n = \|H\|$.

Corollary (Roig, V., Weil, 2007)

Given f.g. subgroups $H \leqslant K \leqslant F_k$, one can check whether H is a free factor of K in polynomial time w.r.t. the given generators of H and K.

Deciding free-factorness

Observation

A given subgroup $H \leqslant F_k$ with basis $\{h_1, \ldots, h_r\}$ ($r(H) = r \leqslant k$) is a free factor of F_k if and only if bottom of the orbit of $(h_1, \ldots, h_r)$ has length $1 + \overset{r}{\cdots} + 1 = r$.

Corollary (Roig, V., Weil, 2007)

Given a f.g. subgroup $H \leqslant F_k$, one can check whether H is a free factor of F_k in time $O((n^2k^4 + n^3k^2)\log(nk))$, where $n = \|H\|$.

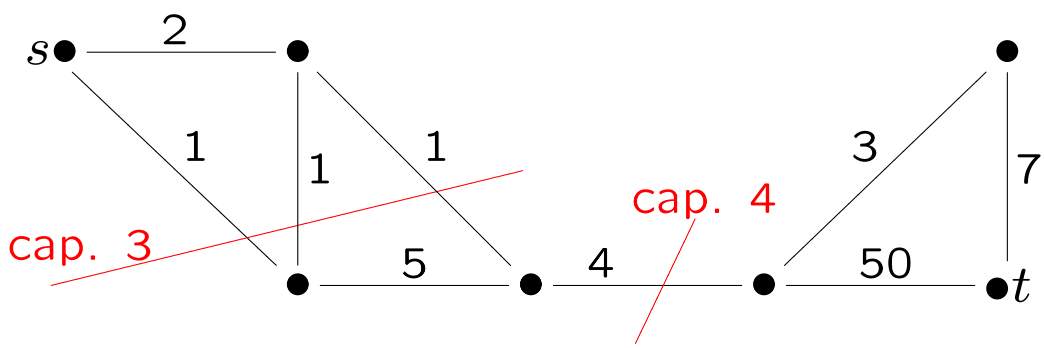
Corollary (Roig, V., Weil, 2007)

Given f.g. subgroups $H \leqslant K \leqslant F_k$, one can check whether H is a free factor of K in polynomial time w.r.t. the given generators of H and K.

Outline

Third ingredient: max-flow min-cut algorithm.

Given a graph X (unoriented and with weights on edges), and two vertices $s, t \in VX$, find the max flow from s to t:



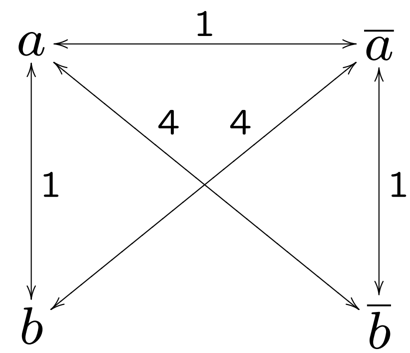
Observation:
maximal $(s \to t)$-flow $\leq$ cap. of any (s, t)-cut.

Theorem:
max. $(s \to t)$-flow $=$ cap. of min. (s, t)-cut,

and it is possible to find both in polynomial time w.r.t. the size of the graph.

Example: Find one of the best Whitehead autos for $u = ba\overline{b}\,\overline{a}\,\overline{b}\,\overline{a}\,\overline{a}baba$.

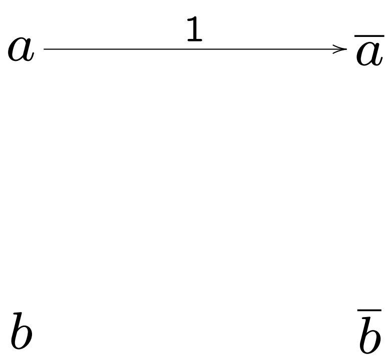
Wh. graph $=$

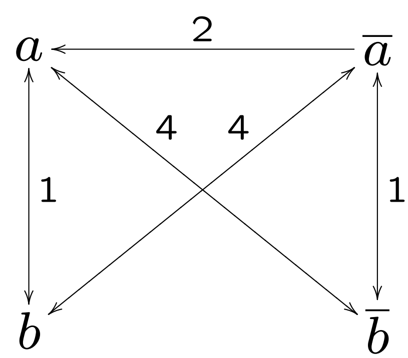


- Choose first multiplier, say a;
- Choose an augmenting path from a to $\overline{a}$:

$$a \xrightarrow{\ 1\ } \overline{a}\ ;$$

- Total flow: residual graph:

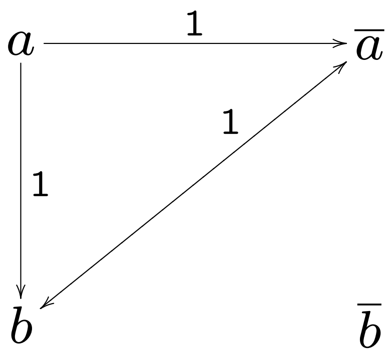


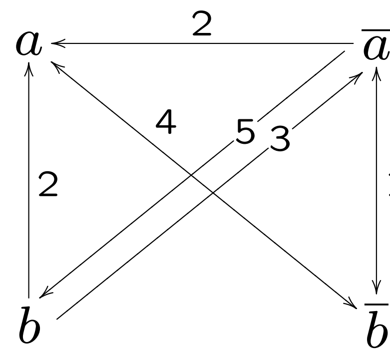


- Choose another augm. path from a to $\overline{a}$:

$$a \xrightarrow{\ 1\ } b \xrightarrow{\ 1\ } \overline{a}\ ;$$

- Total flow: residual graph:

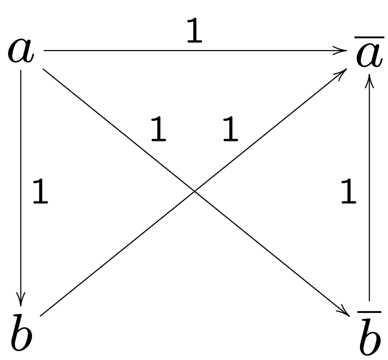


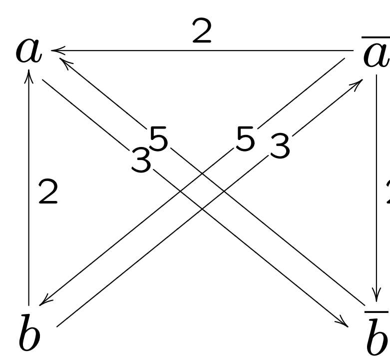


- Choose another augm. path from a to $\overline{a}$:

$$a \xrightarrow{1} \overline{b} \xrightarrow{1} \overline{a}\,;$$

- Total flow: residual graph:

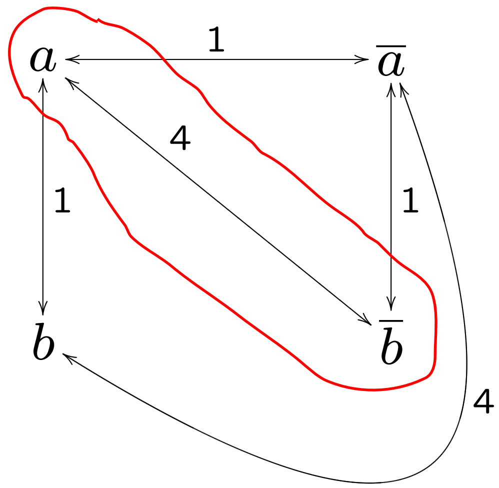




- No paths from a to $\overline{a}$, so STOP.

The total flow carried from a to $\overline{a}$ is 3 and corresponds to the cut

$$Y = \{v \mid \exists \text{ path } a \to v \text{ in res. graph}\}.$$



; $\quad$ cap($\{a, \overline{b}\}$) = 3.

So, the Whitehead auto

$$Y = \{a, \overline{b}\} \quad \equiv \quad \begin{array}{ccc} a & \overset{\alpha}{\mapsto} & a \\ b & \mapsto & \overline{a}b \end{array}$$

satisfies $\|u\alpha\| - \|u\| = 3 - 6 = -3$.

- Repeat for multiplier b (and get less).

$$\begin{array}{rcl} u = ba\overline{b}\,\overline{a}\,\overline{b}\,\overline{a}\,\overline{a}baba & \mapsto & (\not\overline{a}b)a(\overline{b}\not a)\not\overline{a}(\overline{b}\not a)\not\overline{a}\,\overline{a}(\overline{a}b)\not a(\not\overline{a}b)\not a \\ & \sim & ba\overline{b}\,\overline{b}\overline{a}\,\overline{a}bb \end{array}$$

$$\|u\| = 11 \quad , \quad \|u\alpha\| = 8.$$

THANKS