Algorithmic problems in free groups

Pascal Weil

LaBRI, CNRS and Université de Bordeaux

MC2, LIP, January 2007

The free group on A, $F(A)$

- the "simplest" group built from the symbols in set A: start with strings (*words*) on alphabet $A \cup \bar{A}$, and let $a\bar{a} = \bar{a}a = 1$

The free group on A, $F(A)$

- the "simplest" group built from the symbols in set A: start with strings (*words*) on alphabet $A \cup \bar{A}$, and let $a\bar{a} = \bar{a}a = 1$
- *i.e.*, perform reductions: $ua\bar{a}v \longrightarrow uv$, $u\bar{a}av \longrightarrow uv$. For instance $a^3\bar{a}\,\bar{b}ab^2\bar{b}\,\bar{a}a \xrightarrow{*} a^2\bar{b}ab$

The free group on A, $F(A)$

- the "simplest" group built from the symbols in set A: start with strings (*words*) on alphabet $A \cup \bar{A}$, and let $a\bar{a} = \bar{a}a = 1$
- *i.e.*, perform reductions: $ua\bar{a}v \longrightarrow uv$, $u\bar{a}av \longrightarrow uv$. For instance $a^3\bar{a}\,\bar{b}ab^2\bar{b}\,\bar{a}a \xrightarrow{*} a^2\bar{b}ab$
- this rewriting system is confluent, it terminates: it yields a unique representation by means of *reduced* words

The free group on A, $F(A)$

- the "simplest" group built from the symbols in set A: start with strings (*words*) on alphabet $A \cup \bar{A}$, and let $a\bar{a} = \bar{a}a = 1$
- *i.e.*, perform reductions: $ua\bar{a}v \longrightarrow uv$, $u\bar{a}av \longrightarrow uv$. For instance $a^3\bar{a}\,\bar{b}ab^2\bar{b}\,\bar{a}a \xrightarrow{*} a^2\bar{b}ab$
- this rewriting system is confluent, it terminates: it yields a unique representation by means of *reduced* words
- $F(A)$ and $F(B)$ are isomorphic iff A and B have the same cardinality : the notions of *rank*, *basis* make sense for free groups

The free group on A, $F(A)$

- the "simplest" group built from the symbols in set A: start with strings (*words*) on alphabet $A \cup \bar{A}$, and let $a\bar{a} = \bar{a}a = 1$
- *i.e.*, perform reductions: $ua\bar{a}v \longrightarrow uv$, $u\bar{a}av \longrightarrow uv$. For instance $a^3\bar{a}\,\bar{b}ab^2\bar{b}\,\bar{a}a \xrightarrow{*} a^2\bar{b}ab$
- this rewriting system is confluent, it terminates: it yields a unique representation by means of *reduced* words
- $F(A)$ and $F(B)$ are isomorphic iff A and B have the same cardinality : the notions of *rank*, *basis* make sense for free groups
- The subgroups of a free group are free...

The free group on A, $F(A)$

- the "simplest" group built from the symbols in set A: start with strings (*words*) on alphabet $A \cup \bar{A}$, and let $a\bar{a} = \bar{a}a = 1$
- *i.e.*, perform reductions: $ua\bar{a}v \longrightarrow uv$, $u\bar{a}av \longrightarrow uv$. For instance $a^3\bar{a}\,\bar{b}ab^2\bar{b}\,\bar{a}a \xrightarrow{*} a^2\bar{b}ab$
- this rewriting system is confluent, it terminates: it yields a unique representation by means of *reduced* words
- $F(A)$ and $F(B)$ are isomorphic iff A and B have the same cardinality : the notions of *rank*, *basis* make sense for free groups
- The subgroups of a free group are free...
- ... but a subgroup may have a larger rank than the group in which it sits!

The free group on A, $F(A)$

- the "simplest" group built from the symbols in set A: start with strings (*words*) on alphabet $A \cup \bar{A}$, and let $a\bar{a} = \bar{a}a = 1$
- *i.e.*, perform reductions: $ua\bar{a}v \longrightarrow uv$, $u\bar{a}av \longrightarrow uv$. For instance $a^3\bar{a}\,\bar{b}ab^2\bar{b}\,\bar{a}a \xrightarrow{*} a^2\bar{b}ab$
- this rewriting system is confluent, it terminates: it yields a unique representation by means of *reduced* words
- $F(A)$ and $F(B)$ are isomorphic iff A and B have the same cardinality : the notions of *rank*, *basis* make sense for free groups
- The subgroups of a free group are free...
- ... but a subgroup may have a larger rank than the group in which it sits!
- Every subset of $\{a^n b a^{-n} \mid n \geq 0\}$ is a basis of a subgroup of $F(a, b)$.

Algorithmic problems in free groups

- Algorithmic problems on elements (reduced words) and on finitely generated subgroups of free groups: compute the rank, a basis, the index, conjugacy problems, etc.

Algorithmic problems in free groups

- Algorithmic problems on elements (reduced words) and on finitely generated subgroups of free groups: compute the rank, a basis, the index, conjugacy problems, etc.
- The general idea for subgroups: to represent the finitely generated subgroups of a free group by combinatorial objects (certain kinds of *automata*)

Algorithmic problems in free groups

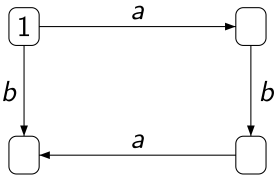
- Algorithmic problems on elements (reduced words) and on finitely generated subgroups of free groups: compute the rank, a basis, the index, conjugacy problems, etc.
- The general idea for subgroups: to represent the finitely generated subgroups of a free group by combinatorial objects (certain kinds of *automata*)
- results in effective (often efficient) computations on subgroups of free groups

Representation of subgroups

- $H = \langle abab^{-1}, aba^{-1}b^{-1}, aba^3b^{-1} \rangle$

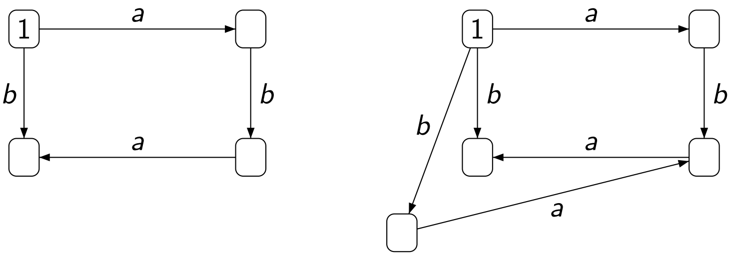
Representation of subgroups

- $H = \langle abab^{-1}, aba^{-1}b^{-1}, aba^3b^{-1} \rangle$



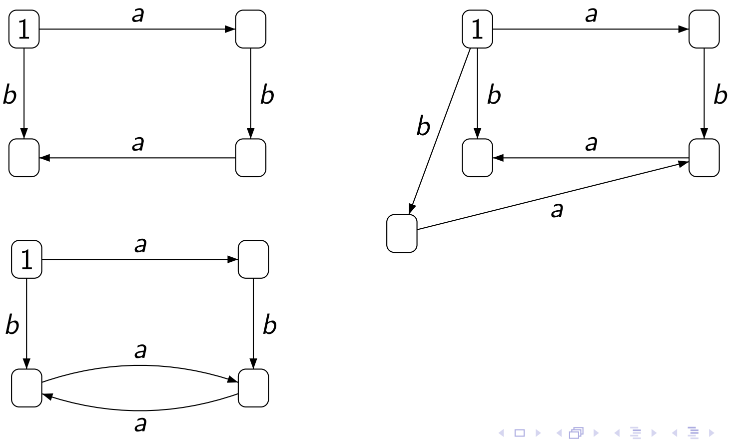
Representation of subgroups

- $H = \langle abab^{-1}, aba^{-1}b^{-1}, aba^3b^{-1} \rangle$



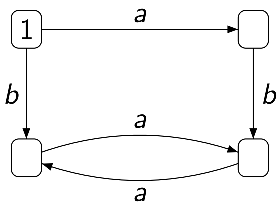
Representation of subgroups

- $H = \langle abab^{-1}, aba^{-1}b^{-1}, aba^3b^{-1} \rangle$



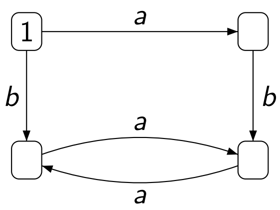
Representation of subgroups

- $H = \langle abab^{-1}, aba^{-1}b^{-1}, aba^3b^{-1} \rangle$
- reduced automaton: finite, connected, deterministic and co-deterministic, with a distinguished state 1; every vertex $v \neq 1$ has valency at least 2



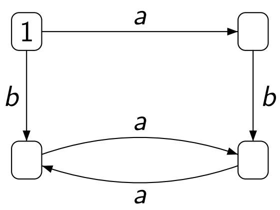
Representation of subgroups

- $H = \langle abab^{-1}, aba^{-1}b^{-1}, aba^3b^{-1} \rangle$
- reduced automaton: finite, connected, deterministic and co-deterministic, with a distinguished state 1; every vertex $v \neq 1$ has valency at least 2
- H = all reduced words that read from 1 to 1



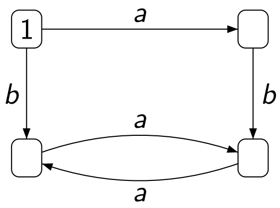
Representation of subgroups

- $H = \langle abab^{-1}, aba^{-1}b^{-1}, aba^3b^{-1} \rangle$
- this automaton is denoted by $\Gamma(H)$, or $\Gamma_A(H)$. It characterizes H, not the generating set

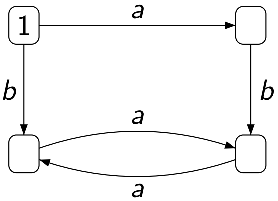


Representation of subgroups

- $H = \langle abab^{-1}, aba^{-1}b^{-1}, aba^3b^{-1}\rangle$
- this automaton is denoted by $\Gamma(H)$, or $\Gamma_A(H)$. It characterizes H, not the generating set
- $\Gamma(H)$ is computable in $O(n \log^* n)$

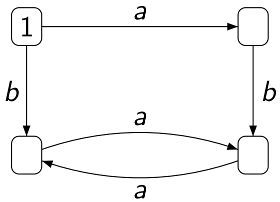


Basic invariants of subgroups



$$H = \langle abab^{-1}, aba^{-1}b^{-1}, aba^3b^{-1} \rangle$$

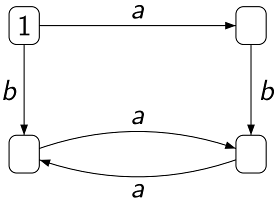
Basic invariants of subgroups



$$H = \langle abab^{-1}, aba^{-1}b^{-1}, aba^3b^{-1} \rangle$$

- aba^3b^{-1} is a product of the other generators

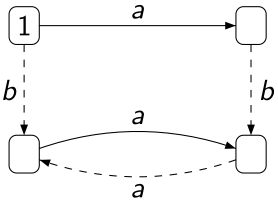
Basic invariants of subgroups



$H = \langle abab^{-1}, aba^{-1}b^{-1} \rangle$

- aba^3b^{-1} is a product of the other generators
- an efficient solution of the (generalized) membership problem

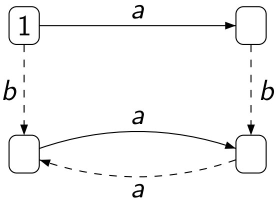
Basic invariants of subgroups



$H = \langle abab^{-1}, aba^{-1}b^{-1} \rangle$

- aba^3b^{-1} is a product of the other generators
- an efficient solution of the (generalized) membership problem
- the rank is the number of "independent" loops, which can be read on $\Gamma(H)$: choose a spanning tree T; then each A-labeled edge not in T yields a basis element

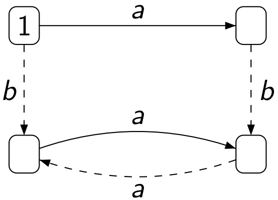
Basic invariants of subgroups



$H = \langle abab^{-1}, aba^{-1}b^{-1} \rangle$

- aba^3b^{-1} is a product of the other generators
- an efficient solution of the (generalized) membership problem
- the rank is the number of “independent” loops, which can be read on $\Gamma(H)$: choose a spanning tree T; then each A-labeled edge not in T yields a basis element
- H has rank 2, and for the dotted spanning tree, we get basis $abab^{-1}, ba^2b^{-1}$

Basic invariants of subgroups

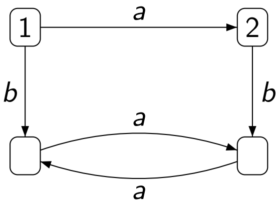


$H = \langle abab^{-1}, aba^{-1}b^{-1} \rangle$

- aba^3b^{-1} is a product of the other generators
- an efficient solution of the (generalized) membership problem
- the rank is the number of "independent" loops, which can be read on $\Gamma(H)$: choose a spanning tree T; then each A-labeled edge not in T yields a basis element
- H has rank 2, and for the dotted spanning tree, we get basis $abab^{-1}, ba^2b^{-1}$
- $\text{rank}(H) = |E(H)| - |V(H)| + 1$.

Finite-index subgroups

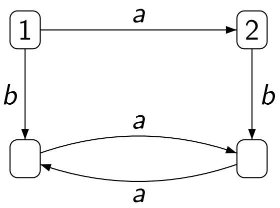
Each vertex is an H-coset:



vertex 2 is Ha, or also $Hbab^{-1}$

Finite-index subgroups

Each vertex is an H-coset:

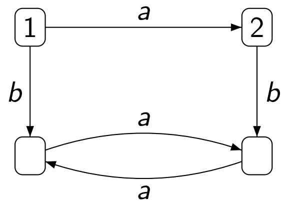


vertex 2 is Ha, or also $Hbab^{-1}$

Recall: $F(A)$ is partitioned by the H-cosets Hg $(g \in F(A))$, the index of H is the number of these cosets.

Finite-index subgroups

Each vertex is an H-coset:



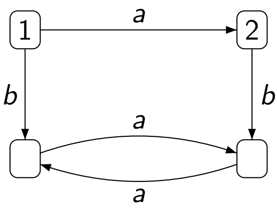
vertex 2 is Ha, or also $Hbab^{-1}$

- In general, not all H-cosets occur as vertices of $\Gamma(H)$. But if $\Gamma(H)$ is a permutation graph, they do, and H has finite index, equal to the number of vertices $|V(H)|$. That's an iff.

Recall: $F(A)$ is partitioned by the H-cosets Hg ($g \in F(A)$), the index of H is the number of these cosets.

Finite-index subgroups

Each vertex is an H-coset:

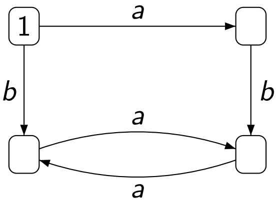


vertex 2 is Ha, or also $Hbab^{-1}$

- In general, not all H-cosets occur as vertices of $\Gamma(H)$. But if $\Gamma(H)$ is a permutation graph, they do, and H has finite index, equal to the number of vertices $|V(H)|$. That's an iff.
- this yields a proof of the Nielsen-Schreier formula for finite index subgroups: if H has finite index $\imath(H)$, then $\mathrm{rank}(H) - 1 = \imath(H)(\mathrm{rank}(F(A)) - 1)$

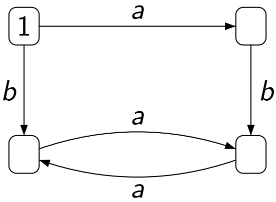
Computing the conjugates of H

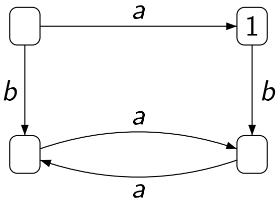
- $H = \langle abab^{-1}, aba^{-1}b^{-1} \rangle$



Computing the conjugates of H

- $H = \langle abab^{-1}, aba^{-1}b^{-1} \rangle$

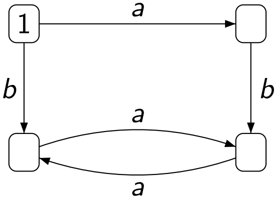


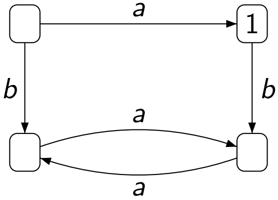


$a^{-1}Ha$

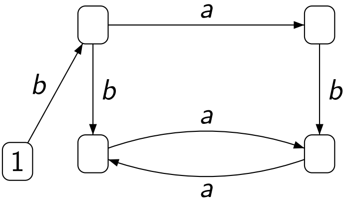
Computing the conjugates of H

- $H = \langle abab^{-1}, aba^{-1}b^{-1} \rangle$





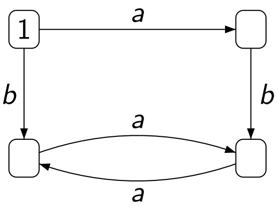
$a^{-1}Ha$

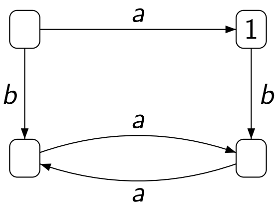


bHb^{-1}

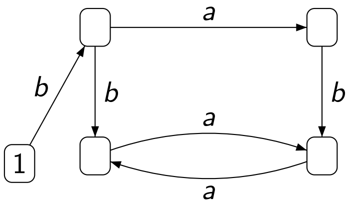
Computing the conjugates of H

- $H = \langle abab^{-1}, aba^{-1}b^{-1} \rangle$





$a^{-1}Ha$



bHb^{-1}

- an algorithm to solve the conjugacy problem

Rank of the intersection of subgroups

- *Howson* (1954): if H and K are finite rank subgroups of $F(A)$, then $H \cap K$ is finitely generated

Rank of the intersection of subgroups

- *Howson* (1954): if H and K are finite rank subgroups of $F(A)$, then $H \cap K$ is finitely generated
- $\langle a^2, abab^{-1}a^{-1}, ab^2ab^{-2}\rangle \cap \langle a, bab^{-1}\rangle = \langle a^2, abab^{-1}a^{-1}\rangle$

Rank of the intersection of subgroups

- *Howson* (1954): if H and K are finite rank subgroups of $F(A)$, then $H \cap K$ is finitely generated
- $\langle a^2, abab^{-1}a^{-1}, ab^2ab^{-2}\rangle \cap \langle a, bab^{-1}\rangle = \langle a^2, abab^{-1}a^{-1}\rangle$
- *Hanna Neumann* (1956)
  $\mathrm{rank}(H \cap K) - 1 \leq 2(\mathrm{rank}(H) - 1)(\mathrm{rank}(K) - 1)$

Rank of the intersection of subgroups

- *Howson* (1954): if H and K are finite rank subgroups of $F(A)$, then $H \cap K$ is finitely generated
- $\langle a^2, abab^{-1}a^{-1}, ab^2ab^{-2}\rangle \cap \langle a, bab^{-1}\rangle = \langle a^2, abab^{-1}a^{-1}\rangle$
- *Hanna Neumann* (1956)
  $\mathrm{rank}(H \cap K) - 1 \leq 2(\mathrm{rank}(H) - 1)(\mathrm{rank}(K) - 1)$
- Let $\widetilde{\mathrm{rk}}(H) = \max(\mathrm{rank}(H) - 1, 0)$, the *reduced rank* of H,
  Hanna Neumann Conjecture (**HNC**)

$$\widetilde{\mathrm{rk}}(H \cap K) \leq \widetilde{\mathrm{rk}}(H)\ \widetilde{\mathrm{rk}}(K)$$

Status of the conjecture

- **HNC**: $\widetilde{\mathrm{rk}}(H \cap K) \leq \widetilde{\mathrm{rk}}(H)\ \widetilde{\mathrm{rk}}(K)$

Status of the conjecture

- **HNC**: $\widetilde{\mathrm{rk}}(H \cap K) \leq \widetilde{\mathrm{rk}}(H)\ \widetilde{\mathrm{rk}}(K)$
- HNC holds if H has finite index (elementary), if H has rank 1 (immediate), or 2 (Tardos, 1992), or 3 (Dicks and Formanek, 2001)

Status of the conjecture

- **HNC**: $\widetilde{\mathrm{rk}}(H \cap K) \leq \widetilde{\mathrm{rk}}(H)\ \widetilde{\mathrm{rk}}(K)$
- HNC holds if H has finite index (elementary), if H has rank 1 (immediate), or 2 (Tardos, 1992), or 3 (Dicks and Formanek, 2001)
- It also holds if H is *positively generated*, i.e. H is generated by a finite set of words using letters from A and not from A^{-1} (Meakin and Weil, Khan, 2002)

Status of the conjecture

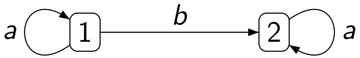
- **HNC**: $\widetilde{\mathrm{rk}}(H \cap K) \leq \widetilde{\mathrm{rk}}(H)\ \widetilde{\mathrm{rk}}(K)$
- HNC holds if H has finite index (elementary), if H has rank 1 (immediate), or 2 (Tardos, 1992), or 3 (Dicks and Formanek, 2001)
- It also holds if H is *positively generated*, i.e. H is generated by a finite set of words using letters from A and not from A^{-1} (Meakin and Weil, Khan, 2002)
- H is positively generated if and only if $\Gamma(H)$ is strongly connected. When is H *potentially connected*? That is, such that $\varphi(H)$ is positively generated for some injective endomorphism φ of $F(A)$.

Status of the conjecture

- **HNC**: $\widetilde{\mathrm{rk}}(H \cap K) \leq \widetilde{\mathrm{rk}}(H)\ \widetilde{\mathrm{rk}}(K)$
- HNC holds if H has finite index (elementary), if H has rank 1 (immediate), or 2 (Tardos, 1992), or 3 (Dicks and Formanek, 2001)
- It also holds if H is *positively generated*, i.e. H is generated by a finite set of words using letters from A and not from A^{-1} (Meakin and Weil, Khan, 2002)
- H is positively generated if and only if $\Gamma(H)$ is strongly connected. When is H *potentially connected*? That is, such that $\varphi(H)$ is positively generated for some injective endomorphism φ of $F(A)$.
- How is this approached by means of automata? compute $\Gamma(H \cap K)$, knowing $\Gamma(H)$ and $\Gamma(K)$

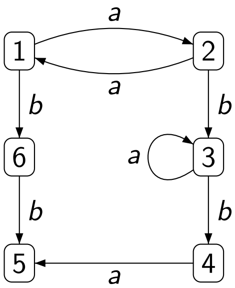
Example

$\Gamma(K)$



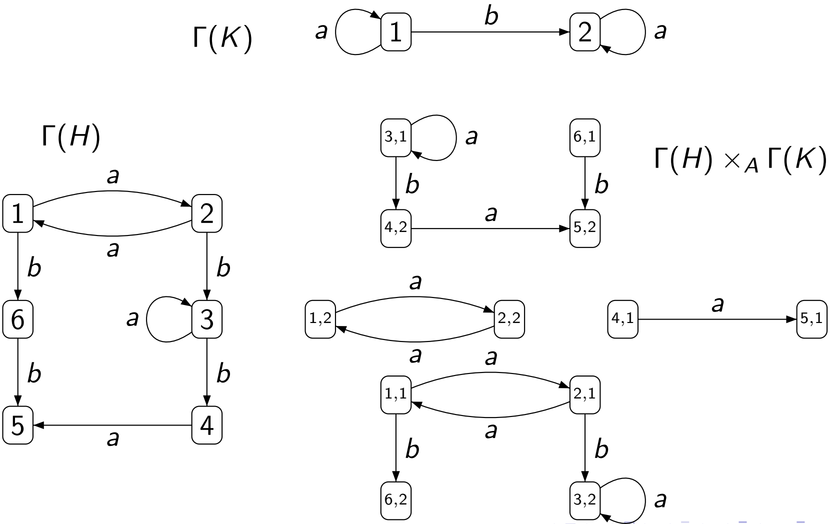
$$K = \langle a, bab^{-1} \rangle$$

$\Gamma(H)$



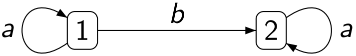
$$H = \langle a^2, abab^{-1}a^{-1}, ab^2ab^{-2} \rangle$$

Example

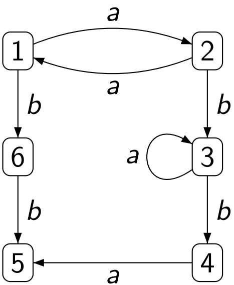

$\Gamma(K)$
a
1
b
2
a
$\Gamma(H)$
1
2
3
4
5
6
a
b
$\Gamma(H) \times_A \Gamma(K)$
3,1
6,1
4,2
5,2
1,2
2,2
4,1
5,1
1,1
2,1
6,2
3,2

Example

$\Gamma(K)$

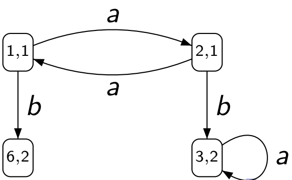


$\Gamma(H)$



$$H \cap K = \langle a^2, abab^{-1}a^{-1} \rangle$$

$\Gamma(H \cap K)$



Translation of HNC in graph-theoretic terms

- Translation of HNC: On each connected component of $\Gamma(H) \times_A \Gamma(K)$,

$$|E| - |V| \leq (|E(H)| - |V(H)|)\ (|E(K)| - |V(K)|)$$

Translation of HNC in graph-theoretic terms

- Translation of HNC: On each connected component of $\Gamma(H) \times_A \Gamma(K)$,

$$|E| - |V| \leq (|E(H)| - |V(H)|)\ (|E(K)| - |V(K)|)$$

- a very simple, graph-theoretic problem. Very simple to state, yet very elusive...

The Whitehead minimization problem

- Problem: To find a minimum length word in the automorphic orbit of a given word u

The Whitehead minimization problem

- Problem: To find a minimum length word in the automorphic orbit of a given word u
- ... or a minimum length element in the automorphic orbit of a conjugacy class of a word (aka a cyclic word $[u]$)

The Whitehead minimization problem

- Problem: To find a minimum length word in the automorphic orbit of a given word u
- ... or a minimum length element in the automorphic orbit of a conjugacy class of a word (aka a cyclic word $[u]$)
- $u = ab\bar{c}ea\bar{d}ca$ (u and $[u]$ have length 8)

The Whitehead minimization problem

- Problem: To find a minimum length word in the automorphic orbit of a given word u
- ... or a minimum length element in the automorphic orbit of a conjugacy class of a word (aka a cyclic word $[u]$)
- $u = ab\bar{c}ea\bar{d}ca$ (u and $[u]$ have length 8)
- Let $\varphi\colon a \mapsto \bar{c}ac$, $b \mapsto \bar{c}bc$, $c \mapsto c$, $d \mapsto dc$, $e \mapsto ec$, then $\varphi(u) = \bar{c}abea\bar{d}ac$ and $\mathrm{cc}(\varphi(u)) = abea\bar{d}a$. So $|\varphi(u)| = 8$ and $|\varphi([u])| = 6$

The Whitehead minimization problem

- Problem: To find a minimum length word in the automorphic orbit of a given word u
- ... or a minimum length element in the automorphic orbit of a conjugacy class of a word (aka a cyclic word $[u]$)
- $u = ab\bar{c}ea\bar{d}ca$ (u and $[u]$ have length 8)
- Let $\varphi\colon a \mapsto \bar{c}ac$, $b \mapsto \bar{c}bc$, $c \mapsto c$, $d \mapsto dc$, $e \mapsto ec$, then $\varphi(u) = \bar{c}abea\bar{d}ac$ and $\mathrm{cc}(\varphi(u)) = abea\bar{d}a$. So $|\varphi(u)| = 8$ and $|\varphi([u])| = 6$
- there is an analogous problem for a finitely generated subgroup H instead of a word u

The Whitehead minimization problem

- Problem: To find a minimum length element in the automorphic orbit of a given word or cyclic word

The Whitehead minimization problem

- Problem: To find a minimum length element in the automorphic orbit of a given word or cyclic word

- Application: decide whether a given element $u \in F$ is primitive

The Whitehead minimization problem

- Problem: To find a minimum length element in the automorphic orbit of a given word or cyclic word

- Application: decide whether a given element $u \in F$ is primitive
- This is the so-called easy part of the equivalence problem (does v belong to the automorphic orbit of u?)

The Whitehead minimization problem

- Problem: To find a minimum length element in the automorphic orbit of a given word or cyclic word
- It is decidable by the *Whitehead method*

- Application: decide whether a given element $u \in F$ is primitive
- This is the so-called easy part of the equivalence problem (does v belong to the automorphic orbit of u?)

The Whitehead minimization problem

- Problem: To find a minimum length element in the automorphic orbit of a given word or cyclic word
- It is decidable by the *Whitehead method*
- Interest in the algorithmic complexity of this problem

- Application: decide whether a given element $u \in F$ is primitive
- This is the so-called easy part of the equivalence problem (does v belong to the automorphic orbit of u?)

The Whitehead minimization problem

- Problem: To find a minimum length element in the automorphic orbit of a given word or cyclic word
- It is decidable by the *Whitehead method*
- Interest in the algorithmic complexity of this problem
- The word case reduces to the cyclic word case at little extra cost
- Application: decide whether a given element $u \in F$ is primitive
- This is the so-called easy part of the equivalence problem (does v belong to the automorphic orbit of u?)

The Whitehead method

- $\mathbb{W}(A)$ the set of non-length preserving Whitehead automorphisms

The Whitehead method

- $\mathbb{W}(A)$ the set of non-length preserving Whitehead automorphisms
- **Theorem** (Whitehead). If there exists $\varphi \in \mathrm{Aut}(F)$ such that $|\varphi([u]) < |[u]|$, then there exists such a $\varphi \in \mathbb{W}(A)$

The Whitehead method

- $\mathbb{W}(A)$ the set of non-length preserving Whitehead automorphisms
- **Theorem** (Whitehead). If there exists $\varphi \in \mathrm{Aut}(F)$ such that $|\varphi([u]) < |[u]|$, then there exists such a $\varphi \in \mathbb{W}(A)$
- **Algorithm**. Try every Whitehead automorphism φ until $|[\varphi(u)]| < |[u]|$. If there is one, replace $[u]$ by $[\varphi(u)]$ and repeat. If there is none, $[u]$ has minimum length.

The Whitehead method

- Let $r = \mathrm{rank}(F) = \mathrm{card}(A)$ and $n = |[u]|$. At most n iterations. Trying one Whitehead automorphism takes time $O(n)$. $\mathrm{card}(\mathbb{W}(A)) = O(r4^r)$. The complexity is in $O(n^2 r4^r)$

- **Algorithm**. Try every Whitehead automorphism φ until $|[\varphi(u)]| < |[u]|$. If there is one, replace $[u]$ by $[\varphi(u)]$ and repeat. If there is none, $[u]$ has minimum length.

The Whitehead method

- Let $r = \text{rank}(F) = \text{card}(A)$ and $n = |[u]|$. At most n iterations. Trying one Whitehead automorphism takes time $O(n)$. $\text{card}(\mathbb{W}(A)) = O(r4^r)$. The complexity is in $O(n^2r4^r)$
- Suggestion that this is can be done faster (Myasnikov *et al.*, Kapovich, Schupp, Shpilrain, etc)

- **Algorithm**. Try every Whitehead automorphism φ until $|[\varphi(u)]| < |[u]|$. If there is one, replace $[u]$ by $[\varphi(u)]$ and repeat. If there is none, $[u]$ has minimum length.

The Whitehead method

- Let $r = \mathrm{rank}(F) = \mathrm{card}(A)$ and $n = |[u]|$. At most n iterations. Trying one Whitehead automorphism takes time $O(n)$. $\mathrm{card}(\mathbb{W}(A)) = O(r4^r)$. The complexity is in $O(n^2 r4^r)$
- Suggestion that this is can be done faster (Myasnikov *et al.*, Kapovich, Schupp, Shpilrain, etc)
- Our result (A. Roig, E. Ventura, P. Weil): an algorithm that is polynomial in n and in r

- **Algorithm**. Try every Whitehead automorphism φ until $|[\varphi(u)]| < |[u]|$. If there is one, replace $[u]$ by $[\varphi(u)]$ and repeat. If there is none, $[u]$ has minimum length.

The Whitehead method

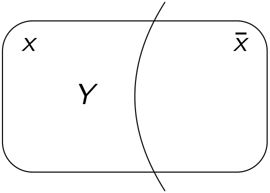
- Let $r = \mathrm{rank}(F) = \mathrm{card}(A)$ and $n = |[u]|$. At most n iterations. Trying one Whitehead automorphism takes time $O(n)$. $\mathrm{card}(\mathbb{W}(A)) = O(r4^r)$. The complexity is in $O(n^2r4^r)$
- Suggestion that this is can be done faster (Myasnikov *et al.*, Kapovich, Schupp, Shpilrain, etc)
- Our result (A. Roig, E. Ventura, P. Weil): an algorithm that is polynomial in n and in r
- In fact, a modification of the algorithm below: do not try every $\varphi \in \mathbb{W}(A)$, but choose an optimal one fast
- **Algorithm**. Try every Whitehead automorphism φ until $|[\varphi(u)]| < |[u]|$. If there is one, replace $[u]$ by $[\varphi(u)]$ and repeat. If there is none, $[u]$ has minimum length.

The Whitehead automorphisms $\mathbb{W}(A)$

- $x \in \tilde{A} = A \sqcup \bar{A}$. $Y \subseteq \tilde{A}$ is an *x-cut* if $x \in Y$ and $\bar{x} \notin Y$

The Whitehead automorphisms $\mathbb{W}(A)$

- $x \in \tilde{A} = A \sqcup \bar{A}$. $Y \subseteq \tilde{A}$ is an *x-cut* if $x \in Y$ and $\bar{x} \notin Y$
- if Y is an x-cut, (x, Y) defines $\varphi \in \mathbb{W}(A)$



$\varphi(x) = x$ and if $a \neq x, \bar{x}$

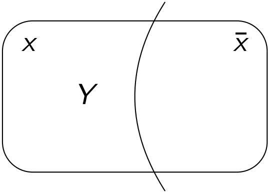
$\varphi(a) = x^{\lambda} a x^{\rho}$ with

$\lambda = -1$ if $\bar{a} \in Y$, 0 otherwise

$\rho = 1$ if $a \in Y$, 0 otherwise

The Whitehead automorphisms $\mathbb{W}(A)$

- $x \in \tilde{A} = A \sqcup \bar{A}$. $Y \subseteq \tilde{A}$ is an *x-cut* if $x \in Y$ and $\bar{x} \notin Y$
- if Y is an x-cut, (x, Y) defines $\varphi \in \mathbb{W}(A)$



$\varphi(x) = x$ and if $a \neq x, \bar{x}$

$\varphi(a) = x^{\lambda} a x^{\rho}$ with

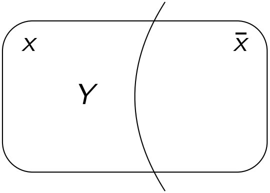
$\lambda = -1$ if $\bar{a} \in Y$, 0 otherwise

$\rho = 1$ if $a \in Y$, 0 otherwise

- Given u, is there (x, Y) such that $|\varphi([u])| - |[u]| < 0$?

The Whitehead automorphisms $\mathbb{W}(A)$

- $x \in \tilde{A} = A \sqcup \bar{A}$. $Y \subseteq \tilde{A}$ is an *x-cut* if $x \in Y$ and $\bar{x} \notin Y$
- if Y is an x-cut, (x, Y) defines $\varphi \in \mathbb{W}(A)$



$\varphi(x) = x$ and if $a \neq x, \bar{x}$

$\varphi(a) = x^{\lambda} a x^{\rho}$ with

$\lambda = -1$ if $\bar{a} \in Y$, 0 otherwise

$\rho = 1$ if $a \in Y$, 0 otherwise

- Given u, is there (x, Y) such that $|\varphi([u])| - |[u]| < 0$?
- Given u, find (x, Y) that minimizes $|\varphi([u])| - |[u]|$

The Whitehead graph of a cyclic word

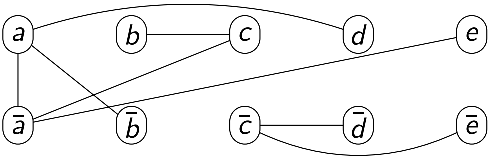
$u = ab\bar{c}ea\bar{d}ca$

The Whitehead graph of a cyclic word

$u = ab\bar{c}ea\bar{d}ca$

A subword xy in the cyclic word yields an edge between x and $\bar{y}$

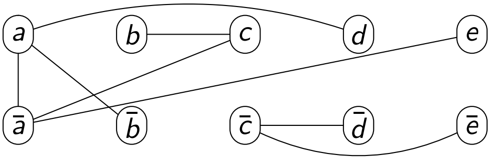
The Whitehead graph of a cyclic word



$u = ab\bar{c}ea\bar{d}ca$

A subword xy in the cyclic word yields an edge between x and $\bar{y}$

The Whitehead graph of a cyclic word

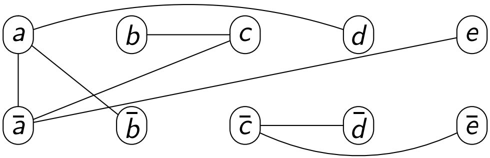


$u = ab\bar{c}ea\bar{d}ca$

A subword xy in the cyclic word yields an edge between x and $\bar{y}$

- $\varphi \in \mathbb{W}(A)$, determined by (x, Y)

The Whitehead graph of a cyclic word



$u = ab\bar{c}ea\bar{d}ca$

A subword xy in the cyclic word yields an edge between x and $\bar{y}$

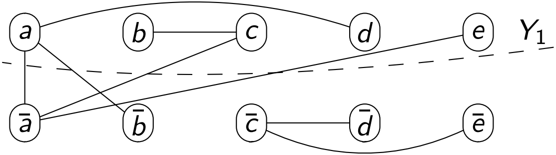
- $\varphi \in \mathbb{W}(A)$, determined by (x, Y)
- Evaluate $|\varphi([u])| - |[u]|$ in graph-theoretic terms, depending on x, Y and the Whitehead graph of $[u]$

A cut formula

- $\mathrm{cap}(Y) =$ number of edges between Y and Y^c

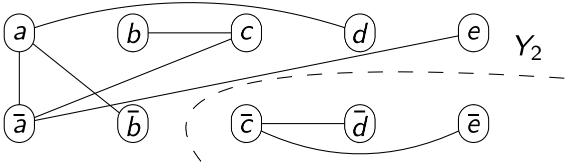
A cut formula

- cap(Y) = number of edges between Y and Y^c
- $Y_1 = \{a, b, c, d, e\}$, $\mathrm{cap}(Y_1) = 4$



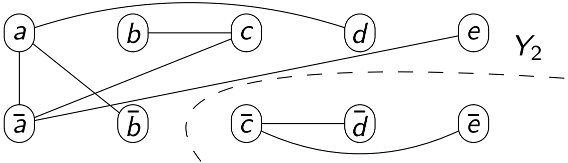
A cut formula

- $\mathrm{cap}(Y) =$ number of edges between Y and Y^c
- $Y_2 = \{a, \bar{a}, \bar{b}, c, d, e\}$, $\mathrm{cap}(Y_2) = 0$



A cut formula

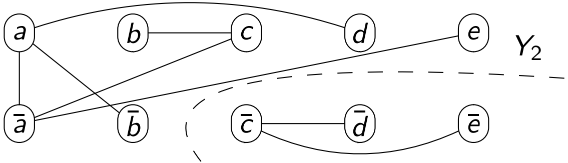
- $\mathrm{cap}(Y) =$ number of edges between Y and Y^c
- $Y_2 = \{a, \bar{a}, \bar{b}, c, d, e\}$, $\mathrm{cap}(Y_2) = 0$



- If $\varphi \in \mathbb{W}(A)$ is determined by (x, Y), then $|\varphi([u])| - |[u]| = \mathrm{cap}(Y) - \deg(x)$

A cut formula

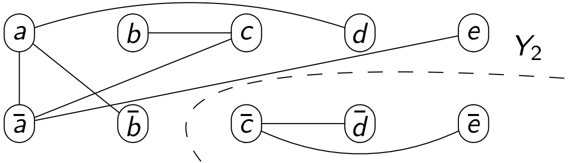
- $\text{cap}(Y) =$ number of edges between Y and Y^c
- $Y_2 = \{a, \bar{a}, \bar{b}, c, d, e\}$, $\text{cap}(Y_2) = 0$



- If $\varphi \in \mathbb{W}(A)$ is determined by (x, Y), then $|\varphi([u])| - |[u]| = \text{cap}(Y) - \deg(x)$
- This is a rewording of a formula in [Lyndon-Schupp]

A cut formula

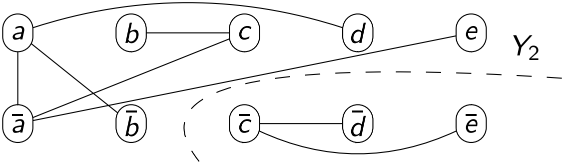
- $\text{cap}(Y) =$ number of edges between Y and Y^c
- $Y_2 = \{a, \bar{a}, \bar{b}, c, d, e\}$, $\text{cap}(Y_2) = 0$



- If $\varphi \in \mathbb{W}(A)$ is determined by (x, Y), then $|\varphi([u])| - |[u]| = \text{cap}(Y) - \deg(x)$
- if $\varphi \longleftrightarrow (d, Y_2)$, then $|\varphi([u])| = |[u]| - 1$

A cut formula

- cap(Y) = number of edges between Y and Y^c
- $Y_2 = \{a, \bar{a}, \bar{b}, c, d, e\}$, $\text{cap}(Y_2) = 0$



- If $\varphi \in \mathbb{W}(A)$ is determined by (x, Y), then $|\varphi([u])| - |[u]| = \text{cap}(Y) - \deg(x)$
- if $\varphi \longleftrightarrow (c, Y_2)$, then $|\varphi([u])| = |[u]| - 2$

The min-cut problem

- Given u, find (x, Y) that minimizes $|\varphi([u])| - |[u]|$

The min-cut problem

- Given u, find (x, Y) that minimizes $|\varphi([u])| - |[u]|$
- Given u, find (x, Y) that minimizes $\text{cap}(Y) - \deg(x)$

The min-cut problem

- Given u, find (x, Y) that minimizes $|\varphi([u])| - |[u]|$
- Given u, find (x, Y) that minimizes $\mathrm{cap}(Y) - \deg(x)$
- Given u, for each $x \in A$, find Y that minimizes $\mathrm{cap}(Y)$

The min-cut problem

- Given u, find (x, Y) that minimizes $|\varphi([u])| - |[u]|$
- Given u, find (x, Y) that minimizes $\mathrm{cap}(Y) - \deg(x)$
- Given u, for each $x \in A$, find Y that minimizes $\mathrm{cap}(Y)$
- This is a standard problem in combinatorial optimization: the *min-cut problem*

The min-cut problem

- Given u, find (x, Y) that minimizes $|\varphi([u])| - |[u]|$
- Given u, find (x, Y) that minimizes $\text{cap}(Y) - \deg(x)$
- Given u, for each $x \in A$, find Y that minimizes $\text{cap}(Y)$
- This is a standard problem in combinatorial optimization: the *min-cut problem*
- There exists an algorithm (Dinic, based on the max-flow min-cut theorem) that solves this problem in $O(nr^2)$ (recall: $n = |[u]|$ and $r = \text{rank}(F)$)

The min-cut problem

- Given u, find (x, Y) that minimizes $|\varphi([u])| - |[u]|$
- Given u, find (x, Y) that minimizes $\mathrm{cap}(Y) - \deg(x)$
- Given u, for each $x \in A$, find Y that minimizes $\mathrm{cap}(Y)$
- This is a standard problem in combinatorial optimization: the *min-cut problem*
- There exists an algorithm (Dinic, based on the max-flow min-cut theorem) that solves this problem in $O(nr^2)$ (recall: $n = |[u]|$ and $r = \mathrm{rank}(F)$)
- The Whitehead minimization problem is thus solved in $O(n^2 r^3)$

The min-cut problem

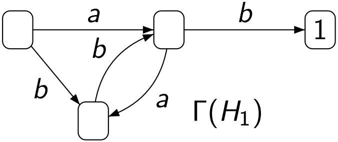
- Given u, find (x, Y) that minimizes $|\varphi([u])| - |[u]|$
- Given u, find (x, Y) that minimizes $\mathrm{cap}(Y) - \deg(x)$
- Given u, for each $x \in A$, find Y that minimizes $\mathrm{cap}(Y)$
- This is a standard problem in combinatorial optimization: the *min-cut problem*
- There exists an algorithm (Dinic, based on the max-flow min-cut theorem) that solves this problem in $O(nr^2)$ (recall: $n = |[u]|$ and $r = \mathrm{rank}(F)$)
- The Whitehead minimization problem is thus solved in $O(n^2r^3)$
- w.l.o.g. $r \leq n$, so there is a solution in $O(n^5)$

The Whitehead problem for subgroups

- A finitely generated subgroup H of F is represented by a finite automaton $\Gamma(H)$

The Whitehead problem for subgroups

- A finitely generated subgroup H of F is represented by a finite automaton $\Gamma(H)$

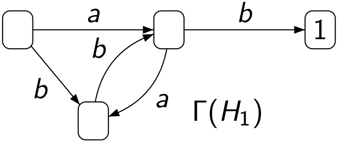


$\Gamma(H_1)$

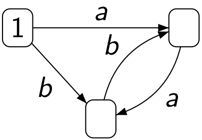
$$H_1 = \langle b^{-1}ab^2, b^{-1}ab^{-1}ab \rangle$$

The Whitehead problem for subgroups

- A finitely generated subgroup H of F is represented by a finite automaton $\Gamma(H)$



$\Gamma(H_1)$

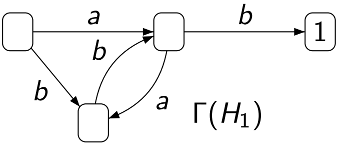


$\Gamma(H_2)$

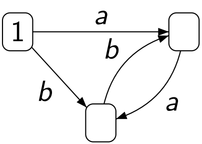
$H_1 = \langle b^{-1}ab^2, b^{-1}ab^{-1}ab \rangle$ $\qquad$ $H_2 = \langle a^2b^{-1}, b^2a^{-1} \rangle$

The Whitehead problem for subgroups

- A finitely generated subgroup H of F is represented by a finite automaton $\Gamma(H)$



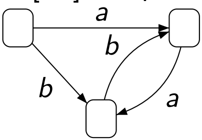
$\Gamma(H_1)$



$\Gamma(H_2)$

$H_1 = \langle b^{-1}ab^2, b^{-1}ab^{-1}ab \rangle$ $\qquad$ $H_2 = \langle a^2b^{-1}, b^2a^{-1} \rangle$

- H_1 and H_2 are conjugates, and the conjugacy class $[H_1] = [H_2]$ is represented by a *cyclically reduced* graph



$\Gamma([H_1]) = \Gamma([H_2])$

The Whitehead problem for subgroups

- A finitely generated subgroup H of F is represented by a finite automaton $\Gamma(H)$
- Say that the *size* of H is the number of vertices.

The Whitehead problem for subgroups

- A finitely generated subgroup H of F is represented by a finite automaton $\Gamma(H)$
- Say that the *size* of H is the number of vertices.
- Problem: To find a minimum size element in the automorphic orbit of a given subgroup H, or of the conjugacy class $[H]$

The Whitehead problem for subgroups

- A finitely generated subgroup H of F is represented by a finite automaton $\Gamma(H)$
- Say that the *size* of H is the number of vertices.
- Problem: To find a minimum size element in the automorphic orbit of a given subgroup H, or of the conjugacy class $[H]$
- A part of the equivalence problem (does $[K]$ belong to the orbit of $[H]$?)

The Whitehead problem for subgroups

- A finitely generated subgroup H of F is represented by a finite automaton $\Gamma(H)$
- Say that the *size* of H is the number of vertices.
- Problem: To find a minimum size element in the automorphic orbit of a given subgroup H, or of the conjugacy class $[H]$
- A part of the equivalence problem (does $[K]$ belong to the orbit of $[H]$?)
- Application: decide whether a given f.g. subgroup $H \leq F$ is a free factor of F

The Whitehead problem for subgroups

- A finitely generated subgroup H of F is represented by a finite automaton $\Gamma(H)$
- Say that the *size* of H is the number of vertices.
- Problem: To find a minimum size element in the automorphic orbit of a given subgroup H, or of the conjugacy class $[H]$
- A part of the equivalence problem (does $[K]$ belong to the orbit of $[H]$?)
- Application: decide whether a given f.g. subgroup $H \leq F$ is a free factor of F
- The cyclic word problem is a special case: if $H = \langle u \rangle$, then H (or $\Gamma([H])$) can be identified with the cyclic word $[u]$.

The Whitehead method still applies

- **Theorem** (Gersten). If there exists $\varphi \in \mathrm{Aut}(F)$ such that $|\varphi([H]) < ||[H]||$, then there exists such a $\varphi \in \mathbb{W}(A)$

The Whitehead method still applies

- **Theorem** (Gersten). If there exists $\varphi \in \mathrm{Aut}(F)$ such that $|\varphi([H]) < |[H]|$, then there exists such a $\varphi \in \mathbb{W}(A)$
- **Algorithm**. Try every Whitehead automorphism φ until $|\varphi([H])| < |[H]|$. If there is one, replace $[H]$ by $[\varphi(H)]$ and repeat. If there is none, $[H]$ has minimum size.

The Whitehead method still applies

- **Theorem** (Gersten). If there exists $\varphi \in \mathrm{Aut}(F)$ such that $|\varphi([H]) < |[H]|$, then there exists such a $\varphi \in \mathbb{W}(A)$
- **Algorithm**. Try every Whitehead automorphism φ until $|\varphi([H])| < |[H]|$. If there is one, replace $[H]$ by $[\varphi(H)]$ and repeat. If there is none, $[H]$ has minimum size.
- We propose again a modification of this algorithm: do not try every $\varphi \in \mathbb{W}(A)$, but choose an optimal one fast

The Whitehead method still applies

- **Theorem** (Gersten). If there exists $\varphi \in \mathrm{Aut}(F)$ such that $|\varphi([H]) < |[H]|$, then there exists such a $\varphi \in \mathbb{W}(A)$
- **Algorithm**. Try every Whitehead automorphism φ until $|\varphi([H])| < |[H]|$. If there is one, replace $[H]$ by $[\varphi(H)]$ and repeat. If there is none, $[H]$ has minimum size.
- We propose again a modification of this algorithm: do not try every $\varphi \in \mathbb{W}(A)$, but choose an optimal one fast
- Given H, find (x, Y) that minimizes $|\varphi([H])| - |[H]|$

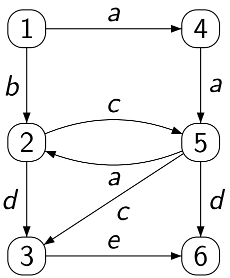
The Whitehead hypergraph of a cyclically reduced graph Γ

- This Whitehead hypergraph is a generalization of the Whitehead graph of a cyclic word. Vertex set $\tilde{A}$

The Whitehead hypergraph of a cyclically reduced graph Γ

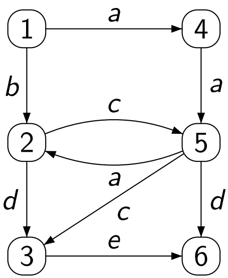
- This Whitehead hypergraph is a generalization of the Whitehead graph of a cyclic word. Vertex set $\tilde{A}$
- A vertex v of Γ yields a hyperedge d_v: the set of letters that label edges into v

The Whitehead hypergraph of a cyclically reduced graph Γ

- This Whitehead hypergraph is a generalization of the Whitehead graph of a cyclic word. Vertex set $\tilde{A}$
- A vertex v of Γ yields a hyperedge d_v: the set of letters that label edges into v
- 

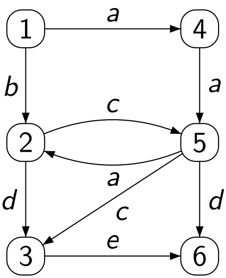
$$d_1 = \{\bar{a}, \bar{b}\}$$
$$d_2 = \{a, b, \bar{c}, \bar{d}\}$$
$$d_3 = \{c, d, \bar{e}\}$$
$$d_4 = \{a, \bar{a}\}$$
$$d_5 = \{a, \bar{a}, c, \bar{c}, d\}$$
$$d_6 = \{d, e\}$$

The cut formula still holds



$$d_1 = \{\bar{a}, \bar{b}\}$$
$$d_2 = \{a, b, \bar{c}, \bar{d}\}$$
$$d_3 = \{c, d, \bar{e}\}$$
$$d_4 = \{a, \bar{a}\}$$
$$d_5 = \{a, \bar{a}, c, \bar{c}, d\}$$
$$d_6 = \{d, e\}$$

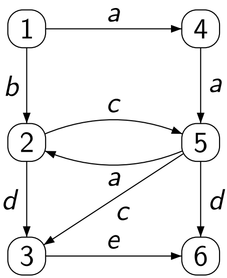
The cut formula still holds

- 

$$d_1 = \{\bar{a}, \bar{b}\}$$
$$d_2 = \{a, b, \bar{c}, \bar{d}\}$$
$$d_3 = \{c, d, \bar{e}\}$$
$$d_4 = \{a, \bar{a}\}$$
$$d_5 = \{a, \bar{a}, c, \bar{c}, d\}$$
$$d_6 = \{d, e\}$$

- $\mathrm{cap}(Y) =$ number of hyperedges that meet Y and Y^c

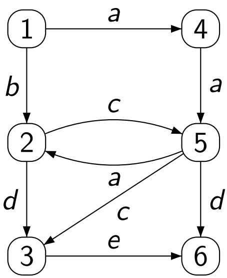
The cut formula still holds

- 

$$d_1 = \{\bar{a}, \bar{b}\}$$
$$d_2 = \{a, b, \bar{c}, \bar{d}\}$$
$$d_3 = \{c, d, \bar{e}\}$$
$$d_4 = \{a, \bar{a}\}$$
$$d_5 = \{a, \bar{a}, c, \bar{c}, d\}$$
$$d_6 = \{d, e\}$$

- $\text{cap}(Y) =$ number of hyperedges that meet Y and Y^c
- If $\varphi \in \mathbb{W}(A)$ is determined by (x, Y), then $|\varphi([H])| - |[H]| = \text{cap}(Y) - \deg(x)$

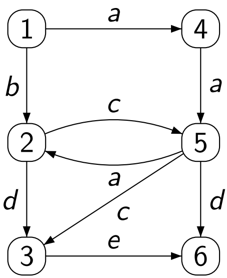
The cut formula still holds

- 

$$d_1 = \{\bar a, \bar b\}$$
$$d_2 = \{a, b, \bar c, \bar d\}$$
$$d_3 = \{c, d, \bar e\}$$
$$d_4 = \{a, \bar a\}$$
$$d_5 = \{a, \bar a, c, \bar c, d\}$$
$$d_6 = \{d, e\}$$

- $\mathrm{cap}(Y) =$ number of hyperedges that meet Y and Y^c
- If $\varphi \in \mathbb{W}(A)$ is determined by (x, Y), then $|\varphi([H])| - |[H]| = \mathrm{cap}(Y) - \deg(x)$
- This is a rewording of a formula of Gersten

The cut formula still holds

- 

$$d_1 = \{\bar{a}, \bar{b}\}$$
$$d_2 = \{a, b, \bar{c}, \bar{d}\}$$
$$d_3 = \{c, d, \bar{e}\}$$
$$d_4 = \{a, \bar{a}\}$$
$$d_5 = \{a, \bar{a}, c, \bar{c}, d\}$$
$$d_6 = \{d, e\}$$

- $\mathrm{cap}(Y) =$ number of hyperedges that meet Y and Y^c
- If $\varphi \in \mathbb{W}(A)$ is determined by (x, Y), then $|\varphi([H])| - |[H]| = \mathrm{cap}(Y) - \deg(x)$
- This is a rewording of a formula of Gersten
- Given H, for each $x \in A$, find Y that minimizes $\mathrm{cap}(Y)$

and an algorithm exists in the literature

- Given H, for each $x \in A$, find Y that minimizes $\mathrm{cap}(Y)$

and an algorithm exists in the literature

- Given H, for each $x \in A$, find Y that minimizes $\mathrm{cap}(Y)$
- This *min-cut problem for hypergraphs* is an instance of a standard problem in combinatorial optimization: the minimization of submodular functions

and an algorithm exists in the literature

- Given H, for each $x \in A$, find Y that minimizes $\mathrm{cap}(Y)$
- This *min-cut problem for hypergraphs* is an instance of a standard problem in combinatorial optimization: the minimization of submodular functions
- There exists an algorithm (Cunningham) that solves this problem in $O(nr^3 \log(nr))$

and an algorithm exists in the literature

- Given H, for each $x \in A$, find Y that minimizes $\mathrm{cap}(Y)$
- This *min-cut problem for hypergraphs* is an instance of a standard problem in combinatorial optimization: the minimization of submodular functions
- There exists an algorithm (Cunningham) that solves this problem in $O(nr^3 \log(nr))$
- The Whitehead minimization problem is thus solved in $O((n^2r^4 + n^3r^2)\log(nr))$

and an algorithm exists in the literature

- Given H, for each $x \in A$, find Y that minimizes $\mathrm{cap}(Y)$
- This *min-cut problem for hypergraphs* is an instance of a standard problem in combinatorial optimization: the minimization of submodular functions
- There exists an algorithm (Cunningham) that solves this problem in $O(nr^3 \log(nr))$
- The Whitehead minimization problem is thus solved in $O((n^2r^4 + n^3r^2)\log(nr))$
- w.l.o.g. $r \leq n$, so there is a solution in $O(n^6 \log n)$

Thank you for your attention!