

A Hybrid Search Algorithm for the Whitehead Minimization Problem

A.D. Myasnikov

*Department of Mathematical Sciences,
Stevens Institute of Technology,
Hoboken, New Jersey*

and

R.M Haralick

*Department of Computer Science,
The Graduate Center of CUNY,
New York*

Abstract

The Whitehead Minimization problem is a problem of finding elements of the minimal length in the automorphic orbit of a given element of a free group. The classical algorithm of Whitehead that solves the problem depends exponentially on the group rank. Moreover, it can be easily shown that exponential blowout occurs when a word of minimal length has been reached and, therefore, is inevitable except for some trivial cases.

In this paper we introduce a deterministic Hybrid search algorithm and its stochastic variation for solving the Whitehead minimization problem. Both algorithms use search heuristics that allow one to find a length-reducing automorphism in polynomial time on most inputs and significantly improve the reduction procedure. The stochastic version of the algorithm employs a probabilistic system that decides in polynomial time whether or not a word is minimal. The stochastic algorithm is very robust. It has never happened that a non-minimal element has been claimed to be minimal.

Key words: Automorphism Problem, Free Groups, Heuristic Search.

Email addresses: amiasnik@stevens.edu (A.D. Myasnikov),
haralick@ptah.gc.cuny.edu (R.M Haralick).

1 Introduction

The Whitehead Minimization problem is a problem of finding elements of the minimal length in the automorphic orbit of a given element of a free group. This problem is of great importance in group theory and topology and continually attracts a great deal of attention from the research community.

Starting from the seminal paper of Whitehead (1936), the Whitehead Minimization problem was studied extensively for more than 70 years (see Lyndon and Schupp (1977); Cohen et al. (1981); Lee (2003); Khan (2004); Kapovich et al. (2004); Miasnikov and Shpilrain (2005); Kaimanovich et al. (2005); Kapovich (2006)) and still the complexity of this problem is unknown.

One of the most important applications of the Whitehead Minimization problem is that its solution is part of the solution to the famous *Automorphism Problem* in free groups introduced by J.H.C.Whitehead in 1936. Methods used to solve the Whitehead Minimization problem can be used to decide whether an element is a part of a generating basis of a free group. The same methods and their generalizations are used in solving equations over free groups (see Razborov (1985)). To practitioners, the Whitehead Minimization problem could be of interest because of its relation to non-commutative variations of the public key cryptographic scheme by Moh (1999).

All known methods of solving the Whitehead Minimization problem have exponential dependence on the rank of a free group. Moreover, the worst case scenario occurs when solving a termination problem (which is to decide whether or not a given element is minimal) for a minimal element. Since the goal of the Whitehead Minimization problem is to find a minimal element, the worst case is inevitable for almost all elements except for elements of a very particular type. This observation leads us to a conclusion that the known deterministic techniques are not suitable for groups of large ranks.

Haralick et al. (2005, 2004); Miasnikov (2004); Miasnikov and Myasnikov (2004), using methods of pattern recognition and exploratory data analysis, show that by introducing proper strategies one can construct a length reduction process which is very efficient on most inputs. Furthermore, in these papers we formulate several conjectures (see 2) regarding the various properties of the problem.

In this paper we present a new algorithm for solving the Whitehead Minimization problem. It is a *hybrid* algorithm in a sense that it employs several stochastic, as well as deterministic, procedures based on the conjectures stated by Haralick et al. (2005).

We combine a stochastic search algorithm and heuristic search procedures

(both described in Section 3.1) with the probabilistic classification system “recognizing” minimal elements (see Section 3.2) to construct a Hybrid Deterministic Whitehead Reduction (HDWR) algorithm solving the Length Reduction Problem in a polynomial number of steps (in terms of group rank) on most input words from a free group. The resulting algorithm is deterministic and still requires an exponential number of steps to prove that a word is minimal.

We present a fast probabilistic algorithm HPWR which is a slight modification of HDWR. Algorithm HPWR is very robust and extremely fast on most input words, including words in free groups of large ranks. Although we do not have a formal proof of the correctness of HPWR, in all the experiments that we have performed it has never happened that the algorithm has produced an incorrect output.

The algorithms HDWR and its probabilistic version HPWR are described in Section 3. We give experimental results evaluating the performance of these two algorithms in Section 4. Comparison with the standard deterministic procedure is also presented in Section 4.

2 The Whitehead minimization problem

Let $X = \{x_1, \dots, x_n\}$ be a finite alphabet; $X^{-1} = \{x^{-1} \mid x \in X\}$ be the set of formal inverses of letters from X and $X^{\pm 1} = X \cup X^{-1}$. A word $w = y_1 \dots y_m$ in the alphabet $X^{\pm 1}$ is called *reduced* if $y_i \neq y_{i+1}^{-1}$ for $i = 1, \dots, m-1$ (here we assume that $(x^{-1})^{-1} = x$). Applying reduction rules $xx^{-1} \rightarrow \varepsilon, x^{-1}x \rightarrow \varepsilon$ (where ε is the empty word), one can reduce each word w in the alphabet $X^{\pm 1}$ to a reduced word \bar{w} . The word \bar{w} is uniquely defined and does not depend on a particular sequence of reductions. Denote by $F = F(X)$ the set of reduced words over $X^{\pm 1}$. The set F forms a group with respect to the multiplication $u \cdot v = \overline{uv}$, called a *free* group with *basis* X . The cardinality $|X|$ is called the *rank* of $F(X)$. We write F_n instead of F to indicate that the rank of F is equal to n .

A bijection $\phi : F \rightarrow F$ is called an *automorphism* of F if $\phi(uv) = \phi(u)\phi(v)$ for every $u, v \in F$. The set $Aut(F)$ of all automorphisms of F forms a group with respect to the composition of maps. Every automorphism $\phi \in Aut(F)$ is completely determined by the images $\phi(x)$ of elements $x \in X$. Sometimes it is more convenient to use non-functional notation $w\phi$ to denote the action of automorphism ϕ on w .

The following two subsets of $Aut(F)$ play an important part in both group theory and topology. An automorphism $t \in Aut(F)$ is called a *Nielsen auto-*

morphism if for some $x \in X$ t fixes all elements $y \in X, y \neq x$ and maps x to one of the elements $x^{-1}, y^{\pm 1}x, xy^{\pm 1}$. Note that automorphisms that map x to x^{-1} , leaving everything else unchanged, cannot cause alterations of the word length. Such automorphisms will be called length invariant automorphisms. By $N(X)$ we denote the set of all Nielsen automorphisms of F except the length-invariant ones.

A non-trivial automorphism $t \in \text{Aut}(F)$ is called a *Whitehead automorphism* if it has one of the following types:

- 1) t permutes the elements of $X^{\pm 1}$;
- 2) t fixes a given element $a \in X^{\pm 1}$ and maps each element $x \in X^{\pm 1}, x \neq a^{\pm 1}$ to one of the elements $x, xa, a^{-1}x$, or $a^{-1}xa$.

It is easy to see that automorphisms of the first type are length-invariant. By $W(X)$ we denote the set of Whitehead's automorphisms of the second type. Obviously, every Nielsen automorphism is also a Whitehead automorphism.

Observe that

$$|N(X)| = 4n(n-1), \quad |W(X)| = 2n4^{(n-1)} - 2n$$

where $n = |X|$ is the rank of F .

It is known (see Lyndon and Schupp (1977)) that every automorphism from $\text{Aut}(F)$ is a product of finitely many Nielsen (hence Whitehead) automorphisms.

The automorphic orbit $\text{Orb}(w)$ of a word $w \in F$ is the set of all automorphic images of w in F :

$$\text{Orb}(w) = \{v \in F \mid \exists \varphi \in \text{Aut}(F) \text{ such that } w\varphi = v\}.$$

A word $w \in F$ is called *minimal* (or *automorphically minimal*) if $|w| \leq |w\varphi|$ for any $\varphi \in \text{Aut}(F)$. By w_{\min} we denote a word of minimal length in $\text{Orb}(w)$. Notice that since there may be several elements of the minimal length in same orbit, w_{\min} is not unique in general.

Problem 2.1 (Minimization Problem (MP)) For a word $u \in F$ find an automorphism $\varphi \in \text{Aut}(F)$ such that $u\varphi = u_{\min}$.

Whitehead (1936) proved the following result which gives a solution to the minimization problem.

Theorem 2.1 (Whitehead) Let $u \in F_n(X)$. If $|u| > |u_{\min}|$, then there exists $t \in W(X)$ such that

$$|u| > |ut|.$$

An automorphism $\phi \in \text{Aut}(F)$ is called a length-reducing automorphism for a given word $u \in F$ if $|u\phi| < |u|$. The theorem above claims that the finite set $W(X)$ contains a length-reducing automorphism for every non-minimal word $u \in F$. This allows one to design a simple search algorithm for (MP).

Let $u \in F$. For each $t \in W(X)$ compute the length of the word ut until $|u| > |ut|$, then put $t_1 = t, u_1 = ut_1$. Otherwise stop and output $u_{min} = u$. This procedure is called the *Whitehead Length Reduction* routine (WLR). Now Whitehead Reduction (WR) algorithm proceeds as follows. Repeat WLR on u , and then on the resulting u_1 , and so on, until at some step k WLR gives an output u_{min} . Then $ut_1 \dots t_{k-1} = u_{min}$, so $\phi = t_1 \dots t_{k-1}$ is the required automorphism.

Notice, that the iteration procedure WR simulates the classical greedy descent method (t_1 is a successful direction from u ; t_2 is a successful direction from u_1 ; etc.) Theorem 2.1 guarantees that the greedy approach will always converge to a global minimum.

Clearly, there could be at most $|u|$ repetitions of WLR on an input $u \in F$

$$|u| > |ut_1| > \dots > |ut_1 \dots t_l| = u_{min}, \quad l \leq |u|.$$

Hence the worst case complexity of the Whitehead's algorithm is bounded from above by

$$cA_n|u|^2,$$

where $A_n = 2n4^{(n-1)} - 2n$ is the number of Whitehead automorphisms in $W(X)$ and the constant c is a stretching factor by which the length of a word increases after a Whitehead automorphism is applied (ignoring the low level implementation details.) One letter can be mapped into a word of length of at most 3, so c is bounded by 3 and does not depend on the rank of a group or the word's length. Since A_n depends exponentially on the rank of a free group, in the worst case scenario the algorithm seems to be impractical for free groups with large ranks. One can try to improve on the number of steps which it takes to find a length-reducing automorphism for a given non-minimal element from F . In this context, the question of interest is the complexity of the following

Problem 2.2 (Length Reduction Problem(LRP)) *For a given non-minimal element $u \in F$ find a length-reducing automorphism.*

We refer to Haralick et al. (2005); Miasnikov and Myasnikov (2004) for a general discussion of this problem. Haralick et al. (2005) offers some empirical evidence that by using smart strategies in selecting Whitehead automorphisms $t \in W(X)$ one can dramatically improve the average complexity of WLR in terms of the rank of a group. Some of the experimental results were formulated as the following conjectures:

Conjecture 2.1 (Haralick et al. (2005)) *Let U_k be the set of all non-minimal elements in F of length k and $NU_k \subset U_k$ the subset of elements which have Nielsen length-reducing automorphisms. Then*

$$\lim_{k \rightarrow \infty} \frac{|NU_k|}{|U_k|} = 1.$$

Conjecture 2.2 (Haralick et al. (2005)) *The feature vectors of weights of the Whitehead Graphs of elements from F are separated into bounded regions in the corresponding space. Each such region can be bounded by a hypersurface and corresponds to a particular Nielsen automorphism in a sense that all elements in the corresponding class can be reduced by that automorphism.*

Arguably the conjectures above are not intuitive and most likely would have been difficult to arrive at without observations obtained using computer experiments. At this point we would like to mention a new development in this area which was not available during the submission of this paper. Kapovich (2006) has recently posted a preprint, giving a mathematical proof of the Conjecture 2.2. To our best knowledge this is the first time non-trivial conjectures were obtained using statistical and exploratory data analysis techniques.

Unfortunately, one can easily see that the worst case behavior of the algorithm WR occurs when a word of minimal length has been reached. Except for some trivial cases (when a minimal word is a generator, for example) all Whitehead automorphisms need to be applied to a minimal word before we can conclude that it is, indeed, minimal. It seems that no algorithm is known to avoid time-consuming computation in this case. We would like to emphasize the importance of this fact by formulating it as a separate problem:

Problem 2.3 (Minimal Word Classification Problem(MWCP)) *For a given $u \in F(X)$, decide whether u is minimal or not.*

We discuss this problem in the previous papers. Haralick et al. (2004) gives a probabilistic solution which is based on regression models. Miasnikov (2004) used the so-called support vector machines to improve the performance in free groups of large ranks. In this paper we introduce a new, significantly more efficient probabilistic system based on the empirical distribution of minimal elements in the corresponding feature space (see Section 3.2).

3 Description of the Hybrid algorithms

3.1 Heuristics for the Length Reduction Problem

We have addressed this problem in the preceding papers. Our first approach was to develop a simple Stochastic Whitehead Reduction (SWR) algorithm to solve LRP. It is implemented as a combination of a greedy descent procedure with genetic search techniques.

Define the search space \mathcal{S} as the set of all finite sequences

$$\mu = \langle t_1, \dots, t_s \rangle$$

of Whitehead automorphisms $t_i \in W(X)$. For such μ and a word $u \in F$ define $u\mu = ut_1 \dots t_s$.

The solution to LRP is any sequence $\mu^* \in \mathcal{S}$ such that

$$|u\mu^*| < |u|.$$

Among all such solutions we prefer the ones that give maximal length reduction of the image. In SWR, we define the criterion function which evaluates a solution μ as

$$\mathcal{F}(\mu) = |u\mu|.$$

The details on the implementation and evaluation of SWR can be found in Miasnikov and Myasnikov (2004).

To our great surprise, this naive stochastic algorithm significantly outperformed the standard algorithm, especially in free groups of large ranks. For example, there were very few runs of WR for words $w \in F_{10}$ with $|w| > 100$ that finished within an hour and there were no such runs for $|w| > 200$. Nevertheless, the stochastic algorithm still was able to find minimal words in a matter of seconds. What seemed to be more important is that the stochastic algorithm did not show exponential dependence on the group's rank.

We strongly believe that if a stochastic algorithm performs very well, then there must be a purely mathematical reason behind this phenomenon which can be uncovered by a proper statistical analysis. Following this philosophy, we performed an analysis of successful solutions produced by SWR. The results helped us to define a number of search heuristics described by Haralick et al. (2005). Below we give a brief description of these heuristics.

First, we observed that among all Whitehead automorphisms in the successful solutions, Nielsen automorphisms statistically had a greater chance to occur. Further experiments showed that more than 99% of non-minimal elements can

be reduced by one of the Nielsen automorphisms. Our first heuristic is based on this observation and simply suggests trying Nielsen automorphisms first in the routine WLR, i.e., in this case we assume that in the fixed listing of automorphisms of $W(X)$, the automorphisms from $N(X)$ come first. We refer to this heuristic as *Nielsen First*. Note, that the Nielsen First heuristic is very general and does not use information about the input word itself. We showed that one can significantly improve the performance of the search procedure by incorporating heuristics that use some knowledge about the input.

Let $u \in F(X)$. The undirected *Labeled Whitehead graph* $W(u) = (V, E(u))$ of the word u is a complete undirected graph, where the set of vertices V is equal to the set $X^{\pm 1}$. Every edge $e = (x, y)$, $x \neq y$ of the Whitehead graph is assigned a weight $\omega_e = n_e/|u|$, where n_e is the number of times subwords xy^{-1} or yx^{-1} occur in u . Note that $\omega_e = 0$ if the subwords corresponding to the edge e do not occur in v . Now, for a given word $u \in F$ define a special vector representation (called a *feature vector*) $f(u) \in \mathbb{R}^{|E(u)|}$ such that

$$f(u) = \langle \omega_{e_1}, \dots, \omega_{e_{|E(u)|}} \rangle .$$

The edges e_i are assumed to be taken in some fixed order. Since the Whitehead graph is complete, the number of edges and, therefore, the size of feature vectors is $3n^2 - n$ for all elements in a free group F_n . The set of all feature vectors is usually called a *feature space* and is denoted by \mathcal{F} .

Experiments show that there is a correlation between the location of the feature vectors in the corresponding space and the length-reducing Nielsen automorphisms.

Let $t \in N(X)$ be a Nielsen automorphism. Define the set

$$O_t = \{w \mid r \in N(X) \text{ and } |wr| < |w| \iff r = t\}$$

as a set of all elements that can be reduced only by t and no other Nielsen automorphism. We also define a set $B_{m,t} \subset O_t$:

$$B_{m,t} = \{w \mid w \in O_t, |w| \leq m\},$$

which is a finite set of elements from O_t with the length of at most m . For a large m we define

$$\lambda_t = \frac{1}{|B_{m,t}|} \sum_{w \in B_{m,t}} f(w)$$

as an estimate of the *mean feature vector* of the elements in O_t .

Now, let

$$d(w, t) = \|f(w) - \lambda_t\|$$

be the distance (in this case, Euclidean distance) between the feature vector of a given word w and the estimate λ_t of the mean feature vector corresponding to the Nielsen automorphism t .

Haralick et al. (2005) experimentally show that in about 99% of the time a randomly generated non-minimal element w can be reduced by a Nielsen automorphism t^* such that

$$d(w, t^*) = \min\{d(w, t) \mid t \in N(X)\}.$$

Now we define the second heuristic, which is called the *Centroid* heuristic. For a given word u compute distances $d(u, t)$ for all $t \in N(X)$ and sort them in the increasing order:

$$d(u, t_1) \leq d(u, t_2) \leq \dots \leq d(u, t_k).$$

Apply automorphisms t_1, \dots, t_k sequentially until a length-reducing Nielsen automorphism (if any) is found.

Now let $e = (x, y^{-1})$, $x, y^{-1} \in V$ be an edge in the Whitehead graph of a word w such that $x \neq y$. By construction of the Whitehead graph, e corresponds to subwords $s_e = (xy)^{\pm 1}$.

There are only two Nielsen automorphisms that reduce lengths of subwords in s_e :

$$\psi_e^x : x \rightarrow xy^{-1}, z \rightarrow z \quad \forall z \neq x$$

and

$$\psi_e^y : y \rightarrow x^{-1}y, z \rightarrow z \quad \forall z \neq y.$$

Denote $\psi_e = \{\psi_e^x, \psi_e^y\}$. We call automorphisms ψ_e the *length reducing with respect* to the edge e .

We can order Nielsen automorphisms $\psi_{e_i} \in N(X)$:

$$\langle \psi_{e_1}, \psi_{e_2}, \dots, \psi_{e_k} \rangle \tag{1}$$

such that the corresponding edges e_1, \dots, e_k are chosen according to the decreasing order of the values of their weights

$$\omega(e_1) \geq \omega(e_2) \geq \dots \geq \omega(e_k).$$

In the third heuristic we apply Nielsen automorphisms in the order given by (1). This heuristic is called the Maximal Edge heuristic. In the paper (Haralick et al. (2005)) we present empirical evidence that most non-minimal elements can be reduced by one of the automorphisms in ψ_{e_1} , given that $\omega(e_1)$ is maximal.

	F_3	F_4	F_5
$ N $	24	48	80
S_1	1	1	1
S_{10}	1	2	3

Centroid

	F_3	F_4	F_5
$ N $	24	48	80
S_1	2	3	4
S_{10}	6	7	6

Maximal Edge

	F_3	F_4	F_5
$ N $	24	48	80
S_1	23	45	70
S_{10}	21	37	50

Nielsen First

Table 1

99th percentile of the number of Nielsen automorphisms computed for different heuristics applied to test sets S_1 and S_{10} in free groups F_3 , F_4 and F_5 .

To get a better understanding of how effective these methods are, we estimate the 99th percentile of the number of Nielsen automorphisms required to reduce a non-minimal word from a given test set (see Table 1).

We can see that the Centroid heuristic is the most effective and is able to predict a length reducing automorphism with very high accuracy. The Maximal Edge heuristic also uses very few automorphisms where the Nielsen First heuristic must apply at least 60% of Nielsen automorphisms in the best case.

The Nielsen First heuristic does not require any additional computations and, therefore, its computational complexity is of order $O(1)$. The Maximal Edge heuristic requires $O(|w|+n^2)$ steps. The Centroid heuristic requires $O(|w|+n^4)$ elementary steps, where n is the rank of a free group. Therefore, one becomes aware of a tradeoff between the length of the input word and the rank of a group. Since the Centroid and Maximal Edge heuristics are more accurate they become more attractive when the length of the input word increases because fewer superfluous automorphisms will be applied to the input word.

In Section 3.3 we show how the stochastic algorithm SWR can be combined with Centroid and Maximal Edge heuristics to improve the solution of the Whitehead Minimization problem.

3.2 Probabilistic System for Classification of Minimal Words

We have already mentioned that the worst case of the standard Whitehead algorithm applied to solve LRP occurs when the word is already minimal. Note that exponential blowout is inevitable in the WR algorithm, unless minimal words are of a very special type. Being able to solve the Minimal Word Classification Problem efficiently is crucial for an efficient solution to the Whitehead Minimization Problem. In Haralick et al. (2004) and Miasnikov (2004) we describe several stochastic classification systems (classifiers) based on pattern recognition techniques such as regression and support vector machines. These classifiers are able to decide whether a given word is minimal in polynomial

time (with respect to group rank) with a very small error of misclassification.

Conclusions in Miasnikov (2004) suggest that one of the classes (minimal or non-minimal) of elements could be located in a compact region in the feature space \mathcal{F} and can be bounded by a hypersurface.

To support this conjecture we perform the following experiment. Assume that the feature vectors of minimal elements follow the multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$ with the mean μ and the covariance Σ . We estimate μ and Σ from a set of randomly generated minimal elements. Experiments show that more than 97% of minimal elements lie inside the hyperellipse, corresponding to the 99.9% confidence interval for μ . Moreover, no non-minimal elements fall inside that region. This is a very strong indication that the feature vectors of minimal elements indeed lie compactly in \mathcal{F} .

Using this result we construct a new probabilistic system WMIN to solve the Minimal Word Classification Problem. We decide that a given word u is minimal if its feature vector $f(u)$ falls inside the corresponding hyperellipsoid and we decide that u is otherwise non-minimal. To be more precise, let μ and Σ be, respectively, the mean and the covariance matrix of feature vectors of minimal elements. Using the so-called Mahalanobis distance we define the decision rule:

$$decide(u) = \begin{cases} \text{minimal,} & \text{if } (x - \mu)^T \Sigma^{-1} (x - \mu) < \rho; \\ \text{non - minimal,} & \text{otherwise} \end{cases}$$

where $(x - \mu)^T$ is the transpose of a column vector $x - \mu$. One way of estimating the threshold ρ was indicated above, where it was taken to correspond to the 99.9% confidence interval of μ , given that feature vectors follow multivariate normal distribution. However, in this case the error of misclassifying minimal elements is unacceptably large (greater than 5%). This indicates that the feature vectors actually are not normally distributed.

A practical way to estimate ρ is to estimate the distribution of distances from feature vectors of minimal elements to their mean. Then we take ρ such that $100(1 - \alpha)$ percent of minimal elements have distances less than ρ for a given α . Note that α corresponds to a confidence level in a non-parametric hypothesis testing.

To compute Mahalanobis distance we need to obtain μ and Σ . One way is to estimate them from a set of randomly generated minimal elements. This process is usually called “training” the classifier. Unfortunately, to generate the sample of minimal elements we require to solve the length reduction problem which, as we have argued, is hard in groups of large ranks. Below we suggest a more efficient training procedure.

Kapovich et al. (2004) show that a random cyclically reduced element in a free group is minimal with asymptotic probability 1. It is also easily shown that any minimal element is already cyclically reduced. Following these two facts, we suggest estimating μ and Σ from a set of randomly generated elements of a whole free group. This can be implemented very efficiently even in groups with large ranks (see Miasnikov and Myasnikov (2004) for more details on generating random elements in a free group).

We would like to mention here the two kinds of errors that may occur when solving the Minimal Word Classification Problem. The first is the error of classifying a non-minimal element as minimal. It is called the *false positive* error. The second is the error of classifying a minimal element as a non-minimal element. It is called the *false negative* error. The rate of the false positive error in all our experiments was zero. This property of the classifier is very important for a successful implementation of the probabilistic version of the hybrid algorithm. We will return to this discussion when we describe HPWR in Section 3.4.

3.3 HDWR

The deterministic hybrid procedure HDWR is given in Figure 1. The algorithm contains two major parts. The first part consists of a number of so-called *fast checks* – linear or polynomial procedures that can solve the length reduction problem on some inputs. In fact, the fast checks used in HDWR are expected to reduce most non-minimal elements in a free group. The problem is that there are non-minimal words which cannot be reduced by fast procedures. Using the fast checks alone, one cannot decide whether an input word is minimal or not. We need to provide a termination condition of the algorithm. This task is solved by the second part of the algorithm which is a version of the standard deterministic algorithm WR. Note that in most cases, the computationally ineffective procedure WR is expected to be executed only on minimal elements.

HDWR is an iterative procedure. On each iteration, the length reduction problem is solved for the word w_c , which is an automorphic image of the minimal length of the input word w found so far. The algorithm terminates when there are no reductions possible and the current word w_c is returned as a minimal word w_{min} .

The first step in the algorithm is the classification procedure WMIN (line 5) which decides whether a current word is minimal or not. Even though reduction procedures used as fast checks do not require significant computational resources, this step helps avoiding superfluous computations by distinguishing minimal elements on the first stage of the algorithm.

DETERMINISTIC HYBRID ALGORITHM

```
1:   SET the current word  $w_c$ ;  $reduced = \mathbf{true}$ ;  
2:   WHILE  $reduced$  BEGIN  
3:      $reduced = \mathbf{false}$ ;  
4:     /* Begin fast checks */  
5:     IF  $w_c$  NOT classified as minimal BEGIN  
6:       IF  $\psi_{e_{max}}$  (Maximal Edge) reduces  $w_c$   
7:          $w_c = reduced\ word$ ;  $reduced = \mathbf{true}$ ;  
8:       ELSE IF Centroid reduces  $w_c$   
9:          $w_c = reduced\ word$ ;  $reduced = \mathbf{true}$ ;  
10:      ELSE IF Stochastic Algorithm reduces  $w_c$   
11:         $w_c = reduced\ word$ ;  $reduced = \mathbf{true}$ ;  
12:      END IF  
13:     /* End fast checks */  
14:     IF NOT  $reduced$  AND  $W(X)$  reduce  $w_c$   
15:        $w_c = reduced\ word$ ;  $reduced = \mathbf{true}$ ;  
16:     END WHILE  
17:   STOP;
```

Fig. 1. Algorithm HDWR.

Fast reduction procedures are based on the search heuristics described in Section 3.1. We use the Maximal Edge heuristic as the first fast check because it requires the least number of steps ($O(n^2 + |w|)$) when compared to other methods. Moreover, n^2 part appears when we construct the Whitehead graph which is required for all heuristics. Note that more than 90% of non-minimal elements are expected to be reduced using one of the two automorphisms corresponding to the maximal weight edge of $WG(w_c)$.

Let $e_{max}(w_c)$ be the maximal edge in $WG(w_c)$ and $\psi_{e_{max}}$ be the set of two length-reducing automorphisms with respect to the edge $e_{max}(w_c)$. On line 6 of the algorithm HDWR we apply automorphisms $\psi_{e_{max}}$ to w_c . The maximal number of steps required to perform the fast check is

$$O(n^2 + |w_c|).$$

These steps include the construction of the feature vector $f_{WG}(w_c)$ and the application of the automorphisms $\psi_{e_{max}}$. Following the observations from Har-

alick et al. (2005) we expect most non-minimal elements to be reduced at line 6. If the word w_c has been reduced by one of the two automorphisms, say $\psi'_{e_{max}}$, we substitute w_c with $\psi'_{e_{max}}(w_c)$ and start a new iteration. If the Maximal Edge check fails, we continue the reduction process utilizing the next step.

As the next fast check we use the Centroid heuristic. Let $\psi_{e_{max}}$ be the set of two automorphisms applied at line 6 and $N(X)$ be the Nielsen. We order automorphisms

$$\langle \varphi_1, \dots, \varphi_k \rangle, \varphi_i \in N_n - \psi_{e_{max}}, \quad (2)$$

such that

$$d(w_c, \varphi_1) \leq d(w_c, \varphi_2) \leq \dots \leq d(w_c, \varphi_k).$$

We apply automorphisms $\varphi_1, \dots, \varphi_k$ in the order given by (2) to the word w_c . If one of the automorphisms has reduced the length of w_c , we stop and start a new iteration with the new, reduced word w_c . The maximal number of steps required to execute this fast check is

$$O(n^4 + n^2|w_c|).$$

By line 10 we already know that the word w_c does not have Nielsen length-reducing automorphisms. The suggested strategy at this point is to try to reduce w_c by executing SWR for a predefined number of generations. If SWR fails to find a length-reducing automorphism, we continue with the algorithm WR. A very conservative bound for the expected maximal number of generations of the stochastic algorithm was given in Miasnikov and Myasnikov (2004). Note that since algorithm SWR performs better than WR only in groups with relatively large ranks (greater than 5), it might happen that performance improves if we omit step 10 when the rank of a free group is small.

The maximal time complexity to find a length-reducing automorphism for w_c using HDWR is still

$$O(2^n|w_c|).$$

However, following the discussion in Section 3.1, we expect the length reduction process to be extremely efficient for most non-minimal words. Unfortunately, as previously stated, the worst case behavior of the algorithm occurs when the current word w_c becomes minimal and, therefore, it is inevitable except for some trivial cases. In the next section we introduce a probabilistic algorithm that addresses this problem.

STOCHASTIC HYBRID ALGORITHM

```
1:   SET the current word  $w_c$ ;  $reduced = \mathbf{true}$ ;  
2:   WHILE  $reduced$  BEGIN  
3:      $reduced = \mathbf{false}$ ;  
4:     IF  $\psi_{e_{max}}$  (Maximal Edge) reduces  $w_c$   
5:        $w_c = reduced\ word$ ;  $reduced = \mathbf{true}$ ;  
6:     ELSE IF Centroid reduces  $w_c$   
7:        $w_c = reduced\ word$ ;  $reduced = \mathbf{true}$ ;  
8:     ELSE IF  $w_c$  classified as minimal  
9:       STOP;  
10:    ELSE IF Stochastic Algorithm reduces  $w_c$   
11:       $w_c = reduced\ word$ ;  $reduced = \mathbf{true}$ ;  
12:    END WHILE  
13:  STOP;
```

Fig. 2. Algorithm HPWR.

3.4 HPWR

Words of both types, minimal and non-minimal, may cause an exponential blowout in the algorithm WR. However, non-minimal words do not seem to be a major problem since we have shown that most of them can be reduced by one of the Nielsen automorphisms.

The bottleneck in solving the length reduction problem occurs in the lack of a fast algorithm to decide whether a word is minimal or not. In fact, the only known deterministic solution is the algorithm WR itself. Recall that the worst case of the algorithm occurs when the input word is already minimal. In this case all of the Whitehead automorphisms have to be applied to the word before the decision that the word is minimal can be made.

In this section we introduce a Hybrid Probabilistic Whitehead Reduction algorithm HPWR for solving Whitehead's Minimization problem. In HPWR the decision on whether or not a word is minimal is made using a probabilistic classification system WMIN. This allows one to avoid the exponential blowout for the cost of a possibility of a very small classification error.

We construct HPWR from HDWR first by removing the last step (line 14)

from the algorithm (see Figure 2). Note the increased role the stochastic algorithm SWRplays. This is the only method in HPWR which is capable of reducing non-minimal elements that do not have Nielsen length-reducing automorphisms.

Secondly, we move the classification step behind the fast reduction procedures. To explain this modification we would like to return to the discussion of the roles played by the two types of classification errors of the classifier WMIN in the view of its new application.

Recall that the two errors are: the false positive error (classifying a non-minimal element as minimal) and the false negative error (classifying a minimal element as a non-minimal element). Now observe that once the classifier WMIN decides that the word w_c is minimal, algorithm HPWR terminates and returns w_c as the result. There is no backtracking or additional checking performed after the decision is made. This means that if a non-minimal word w_c is classified as minimal the algorithm will produce an incorrect result. On the contrary, when a minimal word is misclassified as non-minimal, the cost of such error is the number of extra computational steps performed by the algorithm in order to reduce a non-reducible word. What is important is that the algorithm still produces a correct result.

Let ϵ be the probability of committing the false positive error by WMIN. Now assume that during the reduction process classifier WMIN was called k times to decide whether an element w_c is minimal or not. The probability that the algorithm terminates with a correct answer is $(1 - \epsilon)^k$. This shows that the probability of giving an incorrect answer grows rapidly with the number of times the minimality decision is made.

Note that most of the reductions are expected to be done by fast check procedures. Moving the classification step behind the fast checks allows us to reduce the error of producing an incorrect answer while still maintaining a small computational cost on average.

The arguments above show that the false positive error of the classifier has crucial importance. It is necessary to keep the rate of the false positive error as minimal as possible in order for the algorithm to perform correctly.

It has been noted in Section 3.2 that the error of misclassifying non-minimal elements was zero in all our experiments and, therefore, we expect it to be very small in all instances.

Dataset	Size	Min. length	Avg. length	Max. length	Std. deviation
S_1	10143	3	605.8	1306	359.3
S_{10}	2535	3	1507.9	13381	1527.9
S_P	5645	3	1422.1	143020	5379.0

a) F_3 ;

Dataset	Size	Min. length	Avg. length	Max. length	Std. deviation
S_1	10176	4	629.3	1366	374.9
S_{10}	2498	5	2273.7	34609	2679.1
S_P	5741	4	4785.3	763650	19266.4

b) F_4 ;

Dataset	Size	Min. length	Avg. length	Max. length	Std. deviation
S_1	10165	5	650.6	1388	385.4
S_{10}	2566	7	2791.1	28278	3234.9
S_P	3821	5	2430.5	160794	6491.0

c) F_5 ;

Table 2

Description of the test sets of non-minimal elements in free groups F_3 , F_4 and F_5 .

4 Evaluation

In this section we evaluate the algorithms HDWR and HPWR and compare their performance to the performance of the algorithm WR.

We evaluate these algorithms on the following test sets of randomly generated cyclically reduced non-minimal elements:

- S_1 : contains minimal and non-minimal elements in equal proportions. Non-minimal elements are obtained with one Whitehead automorphism.
- S_P : set of pseudo-randomly generated *primitive elements* in F . Recall, that $w \in F(X)$ is primitive if and only if there exists an automorphism $\alpha \in \text{Aut}(F)$ such that $\alpha(w) \in X^\pm$.
- S_{10} : generated similarly to S_1 , but up to 10 automorphisms are used to generate non-minimal elements.

Some characteristics of the sets in free groups F_3 , F_4 and F_5 are given in Table 2.

Let \mathcal{A} be one of the algorithms WR, HDWR or HPWR. By an elementary step of the algorithm \mathcal{A} , we mean one application of a Whitehead automorphism to a given word. Below we evaluate the performance of \mathcal{A} with respect to the number of elementary steps.

Let $N_s = N_s(\mathcal{A}, S)$ be the average number of elementary steps required by \mathcal{A} to find a minimal element for a given input $w \in S$, where $S \subset F_n$ is a test set.

By $N_{red} = N_{red}(\mathcal{A}, S)$ we denote the average number of elementary length-reducing steps required by \mathcal{A} to reduce a given element $w \in S$ to a minimal one, so N_{red} is the average number of "productive" steps performed by \mathcal{A} . It follows that if t_1, \dots, t_l are all the length reducing automorphisms found by \mathcal{A} when executing its routine on an input $w \in S$ then $|wt_1 \dots t_l| = |w_{min}|$ and the average value of l is equal to N_{red} .

We use values N_s and N_{red} as measures evaluating the performance of the algorithms. In addition we record the CPU time $T(w)$ spent by an algorithm to produce a solution for a particular word w . Since HPWR is a probabilistic algorithm there exists a possibility of producing an incorrect solution. We measure the error of a probabilistic Whitehead reduction algorithm \mathcal{A} by computing the fraction of elements for which \mathcal{A} failed to return a minimal element. Let $Sol_{\mathcal{A}}(w) \in F_n$ be a solution produced by algorithm \mathcal{A} . If result is correct, then $|Sol_{\mathcal{A}}(w)| = |w_{min}|$. The error rate of \mathcal{A} with respect to the test set S

$$E(\mathcal{A}) = \frac{|\{w \in S \mid |Sol_{\mathcal{A}}(w)| > |w_{min}|\}|}{|S|}.$$

In all the experiments we have done with the stochastic algorithm HPWR the error rate was zero, i.e. it has never happened that a non-minimal element has been claimed to be minimal.

First, we experiment with groups of smaller ranks. For elements in free groups F_3 , F_4 and F_5 , algorithm WR can decide in a practically acceptable amount of time on whether an element is minimal or not. This allows us to obtain the true values of lengths of minimal elements for each of the input words and access the error rate of probabilistic algorithms. Results are presented in Tables 3 - 5, where

$$T_{avg} = \frac{1}{|S|} \sum_{w \in S} T(w)$$

and S is the corresponding test set.

From the tables we can see that both algorithms, HDWR and HPWR, significantly outperform WR on the sets of primitive elements with the error of HPWR being small (actually zero). This shows that the fast checks are efficient reduction heuristics. The same picture holds for other sets as well. However, the performance of HDWR deteriorates on sets S_1 and S_{10} , where it is much more difficult to decide whether or not an element is minimal.

\mathcal{A}	N_s		N_{red}		T_{avg}, s	
	mean	std	mean	std	mean	std
WR	360.2	267.5	27.3	18.5	0.11	0.46
HDWR	41.2	29.1	24.2	15.0	0.01	0.05
HPWR	41.1	29.6	24.2	15.0	0.01	0.05

a) F_3 ;

\mathcal{A}	N_s		N_{red}		T_{avg}, s	
	mean	std	mean	std	mean	std
WR	2679.7	2356.5	57.5	37.3	2.03	8.75
HDWR	118.1	114.8	45.5	28.0	0.08	0.31
HPWR	118.3	117.1	45.5	28.0	0.07	0.26

b) F_4 ;

\mathcal{A}	N_s		N_{red}		T_{avg}, s	
	mean	std	mean	std	mean	std
WR	16319.9	20284.53	79.3	52.6	5.52	16.4
HDWR	276.5	539.4	58.9	38.6	0.12	0.29
HPWR	239.2	324.8	58.0	35.6	0.08	0.16

c) F_5 ;

Table 3

Comparison of algorithms WR, HDWR and HPWR on the test sets of primitive elements S_P in free groups F_3 , F_4 and F_5 , where N_s is the average number of elementary steps to find a minimal element, N_{red} the average number of length-reducing steps, T_{avg} is the average time (in seconds) spent on an input.

We have already mentioned that in the case of a minimal element all of the Whitehead automorphisms must be applied to confirm that it is indeed minimal. The sizes of the sets of Whitehead elementary automorphisms in free groups F_3 , F_4 and F_5 are $|\Omega_3| = 90$, $|\Omega_4| = 504$, $|\Omega_5| = 2550$ respectively. From the tables 4 and 5 we can see that the values of N_s in all cases is just a little greater than the size of Ω_n . This indicates that HDWR spends most of its automorphisms and, therefore, time, on elements of minimal length. On the contrary, algorithm HPWR seems to be able to avoid exponential blowout by quickly recognizing minimal elements using the classifier WMIN. Note that N_s computed for HPWR is smaller than $|\Omega_n|$ in all experiments.

To show that algorithm HPWR is applicable to groups of large ranks, we

\mathcal{A}	N_s		N_{red}		T_{avg}, s	
	mean	std	mean	std	mean	std
WR	129.0	33.1	2.14	1.15	0.05	0.03
HDWR	117.5	10.4	1.69	0.81	0.04	0.03
HPWR	53.6	142.4	1.70	0.82	0.011	0.01

a) F_3 ;

\mathcal{A}	N_s		N_{red}		T_{avg}, s	
	mean	std	mean	std	mean	std
WR	734.9	188.3	3.24	1.72	0.30	0.20
HDWR	552.1	61.1	2.42	1.19	0.21	0.12
HPWR	140.7	387.9	2.43	1.29	0.02	0.06

b) F_4 ;

\mathcal{A}	N_s		N_{red}		T_{avg}, s	
	mean	std	mean	std	mean	std
WR	3541.3	908.2	4.28	2.19	1.45	1.05
HDWR	2601.6	341.7	3.29	1.73	0.70	0.41
HPWR	316.8	895.2	3.29	1.73	0.05	0.06

c) F_5 ;

Table 4

Comparison of algorithms WR, HDWR and HPWR on the test sets S_1 in free groups F_3 , F_4 and F_5 , where N_s is the average number of elementary steps to find a minimal element, N_{red} the average number of length-reducing steps, T_{avg} is the average time (in seconds) spent on an input.

perform experiments with primitive elements in free groups F_{10} , F_{15} and F_{20} (see Table 6). We can see that HPWR was able to find solutions quickly with N_s growing very slowly with the rank.

5 Conclusion

The search heuristics described in Haralick et al. (2005) can be successfully applied for solving the Whitehead Reduction problem. Probabilistic algorithm HPWR is very robust and can be used in groups with large ranks whereas any

\mathcal{A}	N_s		N_{red}		T_{avg}, s	
	mean	std	mean	std	mean	std
WR	203.5	86.5	8.45	5.49	0.12	0.13
HDWR	124.5	13.5	6.54	3.94	0.05	0.03
HPWR	64.7	153.9	6.54	3.94	0.02	0.01

a) F_3 ;

\mathcal{A}	N_s		N_{red}		T_{avg}, s	
	mean	std	mean	std	mean	std
WR	1278.6	527.5	17.1	10.7	1.01	0.65
HDWR	569.7	67.5	11.6	7.16	0.36	0.33
HPWR	172.0	416.0	11.6	7.16	0.04	0.03

b) F_4 ;

\mathcal{A}	N_s		N_{red}		T_{avg}, s	
	mean	std	mean	std	mean	std
WR	7650.5	4468.0	27.1	17.8	5.87	8.81
HDWR	2650.5	342.9	17.1	10.8	1.06	0.63
HPWR	360.7	904.5	16.9	10.6	0.08	0.08

c) F_5 ;

Table 5

Comparison of algorithms WR, HDWR and HPWR on the test sets S_{10} in free groups F_3 , F_4 and F_5 , where N_s is the average number of elementary steps to find a minimal element, N_{red} the average number of length-reducing steps, T_{avg} is the average time (in seconds) spent on an input.

other known algorithm fails to produce similar results due to the fact that the worst case is inevitable for most inputs. The computational advantage of HPWR increases when the rank of a free group increases. Indeed, HPWR performs about 11 times faster than WR in F_3 and more than 60 times faster than in F_5 .

	N_s		N_{red}		T_{avg}, s	
	mean	std	mean	std	mean	std
F_{10}	595.2	9195.5	55.5	37.0	0.20	0.58
F_{15}	671.1	883.1	106.1	55.5	1.03	0.59
F_{20}	736.3	874.4	128.4	61.8	2.80	1.41

Table 6

Performance of the algorithm HPWR on sets of primitive elements in free groups F_{10} , F_{15} and F_{20} , where N_s is the average number of elementary steps to find a minimal element, N_{red} the average number of length-reducing steps, T_{avg} is the average time (in seconds) spent on an input.

6 Acknowledgments

The authors thank Alexei G. Miasnikov for his inspiring support of this work. We also would like to thank Mike Newman for his valuable comments and suggestions on this paper.

References

- Cohen, M., Metzler, W., Zimmermann, A., 1981. What does a basis of $f(a, b)$ look like? *Math. Ann.* 257, 435–445.
- Haralick, R. M., Miasnikov, A. D., Myasnikov, A. G., 2004. Pattern recognition approaches to solving combinatorial problems in free groups. *Contemporary Mathematics* 349, 197–213.
- Haralick, R. M., Miasnikov, A. D., Myasnikov, A. G., 2005. Heuristics for the Whitehead Minimization Problem. *J. Experimental Mathematics* 14 (1), 7–14.
- Kaimanovich, V., Kapovich, I., Schupp, P., 2005. The subadditive ergodic theorem and generic stretching factors for free group automorphisms, *Israel J. Math.*, to appear, <http://arxiv.org/abs/math.GR/0504105>.
- Kapovich, I., 2006. Clusters, currents and Whitehead’s algorithm, preprint, <http://lanl.arxiv.org/abs/math.GR/0511478>.
- Kapovich, I., Schupp, P., Shpilrain, V., 2004. Generic properties of Whitehead’s algorithm and isomorphism rigidity of random one-relator groups. *Pacific J. Math.* To appear.
- Khan, B., 2004. The structure of automorphic conjugacy in the free group of rank two. *Computational and experimental group theory, Contemp. Math.* 349, 115–196.
- Lee, D., 2003. Counting words of minimum length in an automorphic orbit, preprint, <http://www.arxiv.org/math.GR/0311410>.

- Lyndon, R., Schupp, P., 1977. Combinatorial Group Theory. Series of Modern Studies in Math. 89. Springer-Verlag.
- Miasnikov, A., Shpilrain, V., 2005. Automorphic orbits in free groups, *Journal of Algebra*, to appear.
- Miasnikov, A. D., 2004. Recognition of Whitehead-minimal elements in free groups of large ranks. *Artificial Intelligence and Symbolic Computation (Lecture notes in Artificial Intelligence)* 3249, 211–221.
- Miasnikov, A. D., Myasnikov, A. G., 2004. Whitehead method and genetic algorithms. *Contemporary Mathematics* 349, 89–114.
- Moh, T. T., 1999. A public key system with signature and master key functions. *Communications in Algebra* 27 (5), 2207–2222.
- Razborov, A., 1985. On systems of equations in a free group. *Math. USSR, Izvestiya* 25 (1), 115–162.
- Whitehead, J. H. C., 1936. On equivalent sets of elements in a free group. *Annals of Mathematic* 37, 782 – 800.