Efficient algorithms for highly compressed data: The Word Problem in Higman's group is in P

Volker Diekert[1], Jürn Laun[1], and Alexander Ushakov[2]

1 **FMI, Universität Stuttgart**
Universitätsstr. 38, 70569 Stuttgart, Germany
{diekert,laun}@fmi.uni-stuttgart.de

2 **Department of Mathematics, Stevens Institute of Technology**
Hoboken, NJ 07030, USA
sasha.ushakov@gmail.com

Abstract

Power circuits are data structures which support efficient algorithms for highly compressed integers. Using this new data structure it has been shown recently by Myasnikov, Ushakov and Won that the Word Problem of the one-relator Baumslag group is is decidable in polynomial time. Before that the best known upper bound was non-elementary. In the present paper we provide new results for power circuits and we give new applications in algorithmic group theory: 1. We define a modified reduction procedure on power circuits which runs in quadratic time thereby improving the known cubic time complexity. 2. We improve the complexity of the Word Problem for the Baumslag group to cubic time thereby providing the first practical algorithm for that problem. 3. The Word Problem of Higman's group is decidable in polynomial time. It is due to the last result that we were forced to advance the theory of power circuits.

Thanks. Part of this work was done when the first two authors visited the Stevens Institute of Technology in September 2010 and March 2011. The support and the hospitality of the Stevens Institute is greatly acknowledged. The work of the third author was partially supported by NSF grant DMS-0914773.

1998 ACM Subject Classification F.2.2 Computations on discrete structures, G.2.2 Graph Algorithms

Keywords and phrases Algorithmic group theory, Data structures, Compression, Word Problem

Digital Object Identifier 10.4230/LIPIcs.STACS.2012.218

1 Introduction

Power circuits have been introduced in [18] as a data structure for integers which supports $+$, $-$, $\leq$, and $(x, y) \mapsto 2^x y$. Thus, by iteration it is possible to represent, by very small circuits, huge values (involving the tower function). Efficient algorithms for power circuits yield efficient algorithms for arithmetic with integers in highly compressed form. This idea of *efficient algorithms for highly compressed data* is the main underlying theme of the present paper. In this sense our paper is simultaneously about compression, data structures and about algorithmic group theory.

In 1910 Max Dehn [5] formulated fundamental algorithmic problems for groups. The most prominent one is the *Word Problem:* "Given a finite presentation of some fixed group G, decide whether an input word w represents the trivial element 1_G in G." It took until the 1950's that Novikov and Boone constructed (independently) finitely presented groups with an undecidable Word Problem [21, 3]. There are also finitely presented groups with a

29th Symposium on Theoretical Aspects of Computer Science (STACS'12).
Editors: Christoph Dürr, Thomas Wilke; pp. 218–229

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

decidable Word Problem but with arbitrarily high complexity [25, Theorem 1.3]. In these examples the difficult instances are extremely sparse and, inherently due to the constructions, these groups never appear in any natural setting.

A finitely presented group has a decidable word problem if and only if there is a recursive upper bound on its Dehn function. Although the Dehn function gives a lot more of information about the group (e.g., if it is linear, then the group is hyperbolic and the Word Problem is linear), the Dehn function is not necessarily a good indicator for the complexity of the Word Problem [16, 23]. However, for "natural examples" the connection between the Dehn function and the complexity of the Word Problem was believed to be rather tight.

Such a natural example was the Baumslag group $G_{(1,2)}$ (sometimes called Baumslag-Gersten group). It is a non-cyclic one-relator group all of whose finite factor groups are cyclic [1]. Being a one-relator group the word problem is decidable. However, the only known general way to solve the Word Problem in one-relator groups is by a so-called Magnus break-down procedure [17, 15] which computes normal forms. It was developed in the 1930s and there has been no progress ever since. Its time-complexity on $G_{(1,2)}$ is non-elementary, since it cannot be bounded by any tower of exponents. Actually, Gersten showed that the Dehn function of $G_{(1,2)}$ is non-elementary [9], see also [24]. Therefore (until recently) $G_{(1,2)}$ was the simplest candidate for a group with a non-polynomial Word Problem in the worst case. But then it turned out that its Word Problem is in P: Using the ability of power circuits to compress huge numbers, Myasnikov, Ushakov and Won showed that the Word Problem of the Baumslag group is solvable in polynomial time [19].

It should be noted that the question of algorithmic hardness of the Word Problem in one-relator groups is still wide open, but some researchers conjecture that it is polynomial (even quadratic, see [2]), based on observations on generic-case complexity [11].

The contributions of the present paper are as follows: In a first part, we give new efficient manipulations of the data structure of *power circuits*. We improves the complexity of the reduction algorithm of power circuits from cubic to quadratic time. With the help of this improved reduction algorithm (and some other ideas) we can, as our second result, reduce the complexity of the Word Problem in $G_{(1,2)}$ significantly from $\mathcal{O}(n^7)$ in [19] down to $\mathcal{O}(n^3)$. This cubic algorithm is the first practical algorithm which works for that problem on all reasonably short instances. The algorithm has been implemented and tested. It is available in the CRAG library [20].

Another new application of power circuits shows that the Word Problem in Higman's group H_4 is decidable in polynomial time. This is our third and main result. Higman's group H_4 is a very interesting group with 4 generators and 4 simple defining relations. Higman [10] constructed H_4 in 1951 as the first example of a finitely presented infinite group where all finite quotient groups are trivial. This leads immediately to an infinite simple group which is finitely generated; and no such group was known before Higman's construction. The group H_4 is constructed by amalgamation (see Section 5 or [27]), which yields decidability of the Word Problem, but the procedure computes normal forms and the length of normal forms can be a tower function in the input length. Thus, Higman's group was another natural, but rather complicated candidate for a finitely presented group with an extremely hard Word Problem. Our paper eliminates H_4 as a candidate: We show that the Word Problem of H_4 is in $\mathcal{O}(n^6)$. Actually, the algorithm for H_4 is more complicated than for the Baumslag group $G_{(1,2)}$.

We obtain this result by new techniques for efficient manipulations of multiple markings in a single power circuit and their ability for huge compression rates. Compression techniques have been applied elsewhere for solving word problems, [12, 13, 26]. But in these papers

the authors use straight-line programs whose compression rates are far too small (at best exponential) to cope with Baumslag or Higman groups.

Due to lack of space in this conference version of the paper, we present a slightly less efficient, yet much less technical version of the reduction procedure. In formal statements we use the "soft-$\mathcal{O}$ notation". Full proofs for the complexity bounds without the poly-logarithmic factors as stated e.g. in the abstract can be found online [7].

Algorithms and problems are classified by their *time complexity* on a random-access machine (RAM).

The *tower function* $\tau : \mathbb{N} \to \mathbb{N}$ is defined as usual: $\tau(0) = 1$ and $\tau(i+1) = 2^{\tau(i)}$ for $i \geq 0$. For instance $\tau(4) = 2^{2^{2^{2^1}}} = 2^{16}$ and $\tau(6)$ written in binary cannot be stored in the memory of any conceivable real-world computer. We use standard notation and facts from group theory as found in the classical text book [15].

2 Power circuits

This section is based on [18], but with improved time complexities. In addition, we provide new material such as our treatment of multiple markings which makes the data structure more versatile. This is used for our results on Higman's group. Let Γ be a set and δ be a mapping $\delta : \Gamma \times \Gamma \to \{-1, 0, +1\}$. This defines a directed graph (Γ, Δ), where Γ is the set of vertices and the set of directed arcs (or edges) is $\Delta = \sigma\delta = \{(P, Q) \in \Gamma \times \Gamma \mid \delta(P, Q) \neq 0\}$ (the *support* of the mapping δ). Note that the sign of $\delta(P, Q)$ is to be read as the edge's label and has nothing to do with its orientation. Throughout we require that (Γ, Δ) is a *dag* (*directed acyclic graph*). In particular, $\delta(P, P) = 0$ for all vertices P.

A *marking* is a mapping $M : \Gamma \to \{-1, 0, +1\}$. We can also think of a marking as a subset of Γ where each element in M has a sign (+ or −). (Thus, we also speak about a *signed subset*.) Each node $P \in \Gamma$ is associated with a marking, which is called its Λ-marking or successor marking Λ_P, consisting of the target nodes of outgoing arcs from P:

$$\Lambda_P : \Gamma \to \{-1, 0, +1\}\,,\ Q \mapsto \delta(P, Q)$$

Thus, the marking Λ_P is the signed subset which corresponds to the targets of outgoing arcs from P. We define the *evaluation* $\varepsilon(P)$ of a node ($\varepsilon(M)$ of a marking resp.) by imposing:

$$\varepsilon(P) = 2^{\varepsilon(\Lambda_P)} \quad \text{for a node } P, \qquad \varepsilon(M) = \sum_{P \in \Gamma} M(P)\varepsilon(P) \quad \text{for a marking } M.$$

Leaves evaluate to 1. The values $\varepsilon(P)$ and $\varepsilon(M)$ can be computed bottom-up in the dag, making $\varepsilon(P)$ and $\varepsilon(M)$ well-defined real numbers. The evaluation of a node P is positive.

▶ **Definition 1.** A *power circuit* is a pair $\Pi = (\Gamma, \delta)$ with $\delta : \Gamma \times \Gamma \to \{-1, 0, +1\}$ such that (Γ, Δ) is a dag as above with the additional property that $\varepsilon(M) \in \mathbb{Z}$ for all markings M.

We will see in Corollary 8 that it is possible to check in quasi-quadratic time whether a dag (Γ, Δ) is a power circuit. (One checks $\varepsilon(\Lambda_P) \geq 0$ for all nodes P.)

▶ **Example 2.** We can represent every integer in the range $[-n, n]$ as the evaluation of some marking in a power circuit with node set $\{P_0, \ldots, P_\ell\}$ such that $\varepsilon(P_i) = 2^i$ for $0 \leq i \leq \ell$ and $\ell = \lfloor \log_2 n \rfloor$. Thus, we can convert the binary notation of an integer n into a power circuit with $\mathcal{O}(\log |n|)$ vertices and $\mathcal{O}((\log |n|) \log\log |n|)$ arcs.

▶ **Example 3.** A power circuit can realize tower functions, since a chain of $n+1$ nodes allows us to represent $\tau(n)$ as the evaluation of the last node.

We denote the empty marking (the constant zero mapping) by 0 and for any marking M there is an obvious definition of $-M$ having $\varepsilon(-M) = -\varepsilon(M)$. The insertion of a new node $\textsc{Clone}(P)$ without incoming arcs and with $\Lambda_{\textsc{Clone}(P)} = \Lambda_P$ is called *cloning of a node* P. The notion is extended to markings, where $\textsc{Clone}(M)$ is obtained by cloning all nodes in $\sigma(M)$ and defining $\textsc{Clone}(M)(\textsc{Clone}(P)) = M(P)$ for $P \in \sigma(M)$ and $\textsc{Clone}(M)(P) = 0$ otherwise. We say that M is a *source*, if no node in $\sigma(M)$ has any incoming arcs. Note that $\textsc{Clone}(M)$ is always a source.

If M and K are markings, then $M + K$ given by $(M + K)(P) = M(P) + K(P)$ is a mapping where -2 and 2 may appear as images. For every P with $M(P) + K(P) = \pm 2$, let $P' = \textsc{Clone}(P)$ and redefine $M + K$ by putting $(M + K)(P) = (M + K)(P') = \pm 1$. In this way we can realize addition (and subtraction) in a power circuit by cloning at most $|\sigma(M) \cap \sigma(K)|$ nodes. Note that any other marking in the power circuit remains unaffected by this operation.

Next, consider markings U and X with $\varepsilon(U) = u$ and $\varepsilon(X) = x$ such that $u2^x \in \mathbb{Z}$ (e.g. due to $x \geq 0$). We obtain a marking V with $\varepsilon(V) = u2^x$ and $|\sigma(V)| = |\sigma(U)|$ as follows. First, let $V = \textsc{Clone}(U)$ and $X' = \textsc{Clone}(X)$. Next, introduce additional arcs from every $P' \in \sigma(V)$ to every $Q' \in \sigma(X')$ with signs given by $\delta(P', Q') = X'(Q')$. Note that the cloning of X avoids double arcs from V to X. The cloning of U is not necessary, if U happens to be a source.

We now introduce the reduction of a power circuit which allows us to compare markings.

▶ **Definition 4.** A *reduced power circuit* consists of

i) a power circuit $\Pi = (\Gamma, \delta)$ in which no two different nodes evaluate to the same number,
ii) an ordered list $[P_1, \ldots, P_n]$ of the nodes Γ such that $\varepsilon(P_i) < \varepsilon(P_{i+1})$ for all $1 \leq i < n$,
iii) a bit vector $[b(1), \ldots, b(n-1)]$ where $b(i) = 1$ if and only if $2\varepsilon(P_i) = \varepsilon(P_{i+1})$.

▶ **Proposition 5** ([18]). There is an $\mathcal{O}(|\Gamma|)$ time algorithm which on input a reduced power circuit and two markings K and M compares $\varepsilon(K)$ and $\varepsilon(M)$. It outputs whether the two values are equal and if not, which one of them is larger. In the latter case it also tells whether $|\varepsilon(K) - \varepsilon(M)|$ is exactly 1 or ≥ 2. (This is essentially an argument about binary sums $\sum_{i \geq 0} a_i \cdot 2^i$ with $a_i \in \{-1, 0, +1\}$.) ◀

Algorithm 1 ExtendReduction

Input: A dag $\Pi = (\Gamma \dot\cup U, \delta)$ with no arcs pointing from Γ to U, such that $(\Gamma, \delta|_{\Gamma\times\Gamma})$ is a reduced power circuit and a list $\mathcal{M}$ of markings of Π.
Output: The output of the procedure is "no", if Π is not a power circuit (because $\varepsilon(P) \notin \mathbb{Z}$ for some node P). Else, the output is a reduced power circuit $\Pi' = (\Gamma', \delta')$ and a list $\mathcal{M}'$ of markings of Π' where:

i) $\Gamma \subseteq \Gamma'$ and $\delta|_{\Gamma\times\Gamma} = \delta'|_{\Gamma\times\Gamma}$
ii) $|\Gamma'| \leq |\Gamma| + 3\,|U|$
iii) For all $Q \in U$ there is a node $Q' \in \Gamma'$ with $\varepsilon(Q) = \varepsilon(Q')$.
iv) For every marking $M \in \mathcal{M}$ there is a corresponding marking $M' \in \mathcal{M}'$ with $\varepsilon(M') = \varepsilon(M)$ and $|\sigma(M')| \leq |\sigma(M)|$.

```
1 compute a topological order [Q_1, ..., Q_|U|] of U, i.e., an ordering of the nodes such
      that for i ≤ j there are no arcs from Q_i to Q_j;
2 for i = 1, ..., |U| do
3     U := U \ {Q_i};
4     let [P_1, P_2, ...] be the ordered list of the current node set Γ;
```

```
5   using binary search, find the minimal j such that
        ε(Q_i) ≤ ε(P_j); /* check ε(Λ_{Q_i}) ≤ ε(Λ_{P_j}) */
6   if ε(Λ_{Q_i}) < 0 then return no fi;
7   if ε(Q_i) < ε(P_j) then /* check ε(Λ_{Q_i}) < ε(Λ_{P_j}) */
8       Γ := Γ ∪ {Q_i};
9       insert {Q_i} into Γ's sorted list of nodes between P_j and P_{j+1};
10      set the bit vector for Q_i according to whether ε(Λ_{Q_i}) + 1 = ε(Λ_{P_j})
11  else /* ε(Q_i) = ε(P_j) */
12      find the maximum n such that b(1) = ... = b(n − 1) = 1;
13      as in example 2, create a new node B with ε(B) = 2^{n+1} and adjust the
            ordered list of Γ and the bit vector accordingly;
14      using the ordered list of Γ and the bit vector, find the maximal number k
            such that there is a chain of nodes P_j = R_0, R_1, ..., R_{k−1} with
            ε(R_ℓ) = ε(P_j) · 2^ℓ (for 0 ≤ ℓ < k);
15      R_k := CLONE(R_{k−1});
16      find the maximal ℓ such that Λ_{R_k}(P_1) = ... = Λ_{R_k}(P_{ℓ−1}) = +1;
17      remove P_1, ..., P_{ℓ−1} from Λ_{R_k} and set Λ_{R_k}(P_ℓ) := +1 instead;
18      insert R_k into Γ's ordered list between P_{j+k−1} and P_{j+k} and set the bit
            vector for R_k according to whether ε(Λ_{R_k}) + 1 = ε(Λ_{P_{j+k}})
19      foreach M ∈ M ∪ {Λ_Q : Q ∈ U} with M(Q_i) =: s ≠ 0 do
20          remove Q_i from M;
21          ℓ := 0; while M(R_ℓ) = s do remove R_ℓ from M od;
22          let M(R_ℓ) := M(R_ℓ) + s
23      od fi od
```

▶ **Theorem 6.** *The procedure* ExtendReduction *given in Algorithm 1 is correct and runs in time* $\widetilde{\mathcal{O}}((|\Gamma| + |U|) \cdot |U| + |\sigma(M_1)| + \ldots + |\sigma(M_m)|)$*, where* $M_1, \ldots, M_m$ *are the markings in* $\mathcal{M}$ *whose support contains nodes in* U*.*

Proof. In the outer loop, the nodes are moved from U to Γ one by one. The topological order ensures that arcs originating from the currently processed node Q_i all end in Γ. The binary search used to determine the right place of U_i inside Γ takes $\log(|\Gamma|+|U|)$ comparisons (remember that $|\Gamma|$ grows in each cycle) and thus $\widetilde{\mathcal{O}}(|\Gamma| + |U|)$ time.

For the insertion of U_i into Γ, we distinguish two cases. The first one (lines 7 to 10) is rather easy: If there is no node in Γ with the same value as U_i, we can just insert U_i and adjust the bit vector using the comparision procedure from 5. No marking involving U_i (this includes Λ markings of other nodes in U) needs to be changed. The second case (lines 11 to 23) is somewhat more complicated. Here we have $\varepsilon(U_i) = \varepsilon(V_j)$. The general idea is to remove U_i and use V_j instead in all markings M having U_i in their support. However, this does not work when $M(U_i) = M(V_j)$ (M would become doubly marked or target of a double arc). In that case, we remove both U_i and V_j from M and replace them by a node R_1 with value $2 \cdot \varepsilon(P_j)$. If again, R_1 becomes doubly marked by M, repeat (lines 19 to 23). This continues until we reach a node unmarked by M (or marked with the opposite sign) or when the sequence of nodes starting at P_j and each node having double the value of its predecessor (we call such a sequence a chain) ends. In order to cope with the latter, we create a new node R_k (which is, of course, completely unmarked) in lines 12 to 18. Doubling a node is done by increasing its Λ-marking by one in the obvious way. An extension of the chain starting at the 1-node might be needed, so we create that first (new node B).

Now, let us look at the time complexity. Topological ordering of U takes linear time in the number of arcs which is bounded by $|U| \cdot (|\Gamma| + |U|)$. For the analysis of the main loop, let us first ignore lines 19 to 23. Apart from the $\log(|\Gamma| + |U|)$ comparisons used in the binary search (each $\mathcal{O}(|\Gamma|$ time), we only need a constant number of comparisons and $\mathcal{O}(|\Gamma| + |U|)$ time e.g. for going through the bit vector. This is $\widetilde{\mathcal{O}}((|\Gamma| + |U|))$ per iteration.

Now to lines 19 to 23: As for the adjustment of markings, note that in every cycle of the inner while loop, we lose one mark or one edge originating in U. This adds up to $\mathcal{O}((|\Gamma| + |U|) \cdot |U| + |\sigma(M_1)| + \ldots + |\sigma(M_m)|)$. Note that markings having their support entirely in Γ never have to be altered during the whole process. ◀

▶ **Corollary 7.** *There is a* $\widetilde{\mathcal{O}}(|\Gamma|^2 + \sum_{M \in \mathcal{M}} |\sigma(M)|)$ *time procedure* Reduce *that reduces a power circuit* $\Pi = (\hat{\Gamma}, \hat{\delta})$ *with markings* $\mathcal{M}$*. The number of nodes at most triples. This is a special case of Theorem 6 where* $\Gamma = \emptyset$ *and* $U = \hat{\Gamma}$*.* ◀

▶ **Corollary 8.** *As a by-product, the procedure* Reduce *tests whether a dag* (Γ, δ) *defines a power circuit (i.e., all markings evaluate to integers).* ◀

By introducing a more sophisticated data structure, one gets rid of the log factors in the time complexity of ExtendReduction and Reduce, see [7]. Note that quadratic time is optimal, since the the number of arcs can be quadratic in the number of nodes.

3 Arithmetic in the semi-direct product $\mathbb{Z}[1/2] \rtimes \mathbb{Z}$

The basic data structure for this paper deals with the semi-direct product $\mathbb{Z}[1/2] \rtimes \mathbb{Z}$. Here $\mathbb{Z}[1/2]$ denotes the ring of rational numbers with denominators in $2^{\mathbb{N}}$ (It is also known as the ring of *dyadic rationals*.) Thus, an element in $\mathbb{Z}[1/2]$ is a rational number r which can be written as $r = u2^x$ with $u, x \in \mathbb{Z}$. We view $\mathbb{Z}[1/2]$ as an abelian group with addition. Multiplication by 2 defines an automorphism of $\mathbb{Z}[1/2]$, and hence the semi-direct product $\mathbb{Z}[1/2] \rtimes \mathbb{Z}$ becomes a (non-commutative) group where elements are pairs $(r, m) \in \mathbb{Z}[1/2] \times \mathbb{Z}$ and with the following explicit formulae for multiplication and inverses:

$$(r, m) \cdot (s, n) = (r + 2^m s, m + n), \qquad (r, m)^{-1} = (-r2^{-m}, -m)$$

The group $\mathbb{Z}[1/2] \rtimes \mathbb{Z}$ is isomorphic to the Baumslag-Solitar group $\mathbf{BS}(1, 2) = \langle a, t \mid tat^{-1} = a^2 \rangle$ via the mapping $a \leftrightarrow (1, 0), t \leftrightarrow (0, 1)$.

A sequence of s group operations may lead to exponentially large or exponentially small values in the first component. Binary representation can cope with these values. We equip $\mathbb{Z}[1/2] \rtimes \mathbb{Z}$ with a partially defined *swap operation*. For $(r, m) \in \mathbb{Z} \times \mathbb{Z} \subseteq \mathbb{Z}[1/2] \rtimes \mathbb{Z}$ we define $\mathrm{swap}(r, m) := (m, r)$. This looks innocent, but note that a sequence of $2^{\mathcal{O}(n)}$ defined operations starting with $(1, 0)$ may yield a pair $(0, \tau(n))$ where τ is the tower function. Indeed $\mathrm{swap}(1, 0) = (0, 1) = (0, \tau(0))$ and

$$\mathrm{swap}((0, \tau(n))(1, 0)(0, -\tau(n))) = \mathrm{swap}(\tau(n + 1), 0) = (0, \tau(n + 1)). \tag{1}$$

We also use triples to denote elements in $\mathbb{Z}[1/2] \rtimes \mathbb{Z}$. A triple $[u, x, k]$ with $u, x, k \in \mathbb{Z}$ and $x \leq 0 \leq k$ denotes the pair $(u2^x, k + x) \in \mathbb{Z}[1/2] \rtimes \mathbb{Z}$. For each element in $\mathbb{Z}[1/2] \rtimes \mathbb{Z}$ there are infinitely many corresponding triples. Using the generators a and t of $\mathbf{BS}(1, 2)$ one can write:

$$\begin{aligned} [u, x, k] = (u2^x, k + x) &= (0, x)(u, k) \in \mathbb{Z}[1/2] \rtimes \mathbb{Z} \\ &= t^x a^u t^k \in \mathbf{BS}(1, 2) \text{ and} \\ [u, x, k] \cdot [v, y, \ell] &= [u2^{-y} + v2^k, x + y, k + \ell] \end{aligned}$$

In what follows we use power circuits with *triple markings* for elements in $\mathbb{Z}[1/2] \rtimes \mathbb{Z}$. For $T = [U, X, K]$, where U, X, K are markings in a power circuit with $\varepsilon(U) = u$, $\varepsilon(X) = x \leq 0$, $\varepsilon(K) = k \geq 0$, we define $\varepsilon(T) \in Z[1/2] \rtimes \mathbb{Z}$ to be the triple $\varepsilon(T) = [u, x, k] = (u2^x, x + k)$.

Let us note that the Word Problem of $(\mathbb{Z}[1/2] \rtimes \mathbb{Z}, \cdot, \mathrm{swap})$ is solvable in polynomial time [7].

4 Solving the Word Problem in the Baumslag group

The Baumslag group[1] $G_{(1,2)}$ is a one-relator group with two generators a and b and the defining relation $a^{a^b} = a^2$. (The notation g^h means conjugation, here $g^h = hgh^{-1}$. Hence $a^{a^b} = bab^{-1}aba^{-1}b^{-1}$.) The group $G_{(1,2)}$ can be written as an HNN extension of $\mathbf{BS}(1,2) \simeq \mathbb{Z}[1/2] \rtimes \mathbb{Z}$ with *stable letter* b; and $\mathbf{BS}(1,2)$ is an HNN extension of $\mathbb{Z} \simeq \langle a \rangle$ with *stable letter* t:

$$\begin{aligned}\langle a, b \mid a^{a^b} = a^2 \rangle &\simeq \langle a, t, b \mid a^t = a^2, a^b = t \rangle \simeq \mathrm{HNN}(\langle a, t \mid a^t = a^2 \rangle, b, \langle a \rangle \simeq \langle t \rangle) \\ &\simeq \mathrm{HNN}\left(\mathrm{HNN}(\langle a \rangle, t, \langle a \rangle \simeq \langle a^2 \rangle), b, \langle a \rangle \simeq \langle t \rangle\right)\end{aligned}$$

Before the work of Myasnikov, Ushakov and Won [19], $G_{(1,2)}$ had been a possible candidate for a one-relator group with an extremely hard (non-elementary) word problem in the worst case by the result of Gersten [9]. (Indeed, the tower function occurs as follows: Let $T(0) = t$ and $T(n+1) = bT(n)aT(n)^{-1}b^{-1}$. Then $T(n) = t^{\tau(n)}$ by a translation of Equation 1.) The purpose of this section is to improve the $\mathcal{O}(n^7)$ time-estimation of [19] to (quasi-)cubic time. Theorem 9 yields the first practical algorithm to solve the Word Problem in $G_{(1,2)}$ for a worst-case scenario[2]. It has been implemented and runs in reasonable time on instances of several thousand letters.

▶ **Theorem 9.** *The Word Problem of the Baumslag group $G_{(1,2)}$ is decidable in time $\widetilde{\mathcal{O}}(n^3)$.*

Proof. We assume that the input is already in compressed form, given by a sequence of letters b, b^{-1} and triple markings $[U, X, K]$, the latter representing elements in $\mathbb{Z}[1/2] \rtimes \mathbb{Z}$, which in turn encode words over $a^{\pm 1}$'s and $t^{\pm 1}$'s. We use the following invariants:

i) U, X, K have pairwise disjoint supports.
ii) U is a source.
iii) All incoming arcs to $X \cup K$ have their origin in U.
iv) Arcs from U to X have the opposite sign of the corresponding node-sign in X.

These are clearly satisfiable in case we start with a sequence of $a^{\pm 1}$'s, $t^{\pm 1}$'s, and $b^{\pm 1}$'s (e.g. create disjoint power circuits, one for each marking, as in Example 2). The formula $[u, x, k] \cdot [v, y, \ell] = [u2^{-y} + v2^k, x + y, k + \ell]$ allows to multiply elements in $\mathbb{Z}[1/2] \rtimes \mathbb{Z}$ without destroying the invariants or increasing the total number of nodes in the power circuits (the invariants make sure that cloning is not necessary). The total number of multiplications is bounded by n. Taking into account that there are at most n^2 arcs, we are within the time bound $\mathcal{O}(n^3)$.

Now we perform leftmost Britton reductions, see [15]. In terms of group generators this means replacing factors ba^sb^{-1} by t^s and $b^{-1}t^sb$ by a^s (always replacing the leftmost occurance first). Thus, if we see a subsequence $b[u, x, k]b^{-1}$, then we must check if $x + k = 0$ and after that if $u2^x \in \mathbb{Z}$. If we see a subsequence $b^{-1}[u, x, k]b$, then we must check $u = 0$. In the positive cases we swap, in the other case we do nothing. Let us give the details: For a test we reduce a copy of the circuit using REDUCE which takes time $\widetilde{\mathcal{O}}(n^2)$. After each test for a Britton reduction, the copy is deleted. There are two possibilities for necessary tests.

1.) $u = 0$. If yes, remove in the original power circuit the source U, this makes $X \cup K$ a source; replace $[u, x, k]$ by $[x + k, 0, 0]$. The invariants are satisfied.

[1] sometimes called Baumslag-Gersten group, not to be confused with the Baumslag-Solitar group $\mathbf{BS}(1,2)$
[2] It is easy to design simple algorithms which perform extremely well on random inputs. But all these algorithms fail on short instances, e.g. in showing $tT(6) = T(6)t$.

2.) $x + k = 0$. If yes, check whether $u2^x \in \mathbb{Z}$. If yes, replace $[u, x, k]$ in the original power circuit by either $[0, u2^x, 0]$ or $[0, 0, u2^x]$ depending on whether $u2^x$ is negative or positive. We get $u2^x$ without increasing the number of nodes, since arcs from U to X have the opposite signs of the node-signs in X. Thus, if E has been the set of arcs before the test, it is switched to $U \times X \setminus E$ after the test. The new marking for $u2^x$ is a source and does not introduce any cycle, because its support is still the support of the source U.

It is easy to see that computing a Britton reduction on an input sequence of size n, we need at most $2n$ tests and at most n of them are successful. Hence we are still within the time bound $\widetilde{\mathcal{O}}(n^3)$. ◀

5 Solving the Word Problem in Higman's group H_4

The Higman group H_q has a finite presentation with generators $a_1, \ldots, a_q$ and defining relations $a_p a_{p-1} a_p^{-1} = a_{p-1}^2$ for all $p \in \mathbb{Z}/q\mathbb{Z}$. It is known [27] that H_q is trivial for $q \leq 3$ and infinite for $q \geq 4$. From now on, we focus on the group H_4 which is the one usually referred to as *the* Higman group. This group was the first example of a finitely generated group where all finite quotient groups are trivial. It was another potential natural candidate for a group with an extremely hard (non-elementary) word problem in the worst case. Indeed, define:

$$w(p, 0) = a_p \ (p \in \mathbb{Z}/4\mathbb{Z}), \qquad w(p-1, i+1) = w(p, i) a_{p-1} w(p, i)^{-1} \ (i \in \mathbb{N}, p \in \mathbb{Z}/4\mathbb{Z})$$

By induction, $w(p, n) = a_p^{\tau(n)} \in H_4$, where $\tau(n)$ is the n-th value of the tower function, but the length of the words $w(p, n)$ is $2^{n+1} - 1$, only. Hence there is a "tower-sized gap" between input length and length of a canonical normal form.

We define the group G_{12} by the generators a_1 and a_2, and defininig relation $a_2 a_1 a_2^{-1} = a_1^2$. In the same way we define G_{23}, G_{34}, and G_{41}. As we saw in Section 3, each of these groups is isomorphic to the Baumslag-Solitar group $\mathbf{BS}(1, 2)$ and to $\mathbb{Z}[1/2] \rtimes \mathbb{Z}$. Furthermore, we define the group G_{123} by the generators a_1, a_2, a_3 and defining relations $a_2 a_1 a_2^{-1} = a_1^2$ and $a_3 a_2 a_3^{-1} = a_2^2$. (Similarly define G_{341}.) We have $G_{123} = G_{12} *_{F_2} G_{23}$ where F_2 is the free subgroup of both G_{12} and G_{23} generated by a_2. Finally, we get H_4 as an amalgamated product

$$H_4 \simeq G_{123} *_{F_{13}} G_{341},$$

where F_{13} is the free subgroup of rank two of G_{123} and G_{341} with basis $\{a_1, a_3\}$, see [27].

This isomorphism yields a direct proof that H_4 is an infinite group, see [27]. In the following we use some well-known facts about amalgamated products, see [15, 27, 6]. In order to solve the Word Problem, we start with an alternating sequence of group elements from G_{123} and G_{341}. The sequence can be shortened, only if one factor happens to be in the subgroup F_{13}. In this case we swap the factor from G_{123} to G_{341} and vice versa. By abuse of language we call this procedure again a *Britton reduction.* (This is perhaps not standard notation, but it conveniently unifies the same phenomenon in amalgamated products and HNN-extensions.) A sequence evaluating to 1 in H_4 can be entirely eliminated using these kinds of Britton reductions.

Elements in the groups $G_{i,i+1}$ ($i \in \mathbb{Z}/4\mathbb{Z}$) are represented by triple markings $T = [U, X, K]$ in some power circuit. In order to remember that we evaluate T in the group $G_{i,i+1}$, we give each T a *type* $(i, i+1)$, which is recorded as a subscript. For $\varepsilon(T) = [u, x, k]$ we obtain:

$$\begin{aligned} \varepsilon(T_{(i,i+1)}) &= a_{i+1}^x a_i^u a_{i+1}^k && \in G_{i,i+1} \\ &= a_i^{u2^x} a_{i+1}^{x+k} && \text{if } u2^x \in \mathbb{Z} \end{aligned}$$

From now on we work with a single power circuit Π together with a sequence $(T_j)_{j\in J}$ of triple markings of various types. This is given as a tuple $\mathcal{T} = (\Gamma, \delta; (T_j)_{j\in J})$. If (Γ, δ) is reduced, then $\mathcal{T}$ is called a *main data structure* .

▶ **Definition 10.** The *weight* $\omega(T)$ of a triple marking $T = [U, X, K]$ is defined as $\omega(T) = |\sigma(U)| + |\sigma(X)| + |\sigma(K)|$. The *weight* $\omega(\mathcal{T})$ of a main data structure $\mathcal{T}$ is defined as $\omega(\mathcal{T}) = \sum_{j\in J} \omega(T_j)$. Its *size* $\|\mathcal{T}\|$ is defined by $\|\mathcal{T}\| = |\Gamma|$.

The following basic operations are defined on a main data structure. Applying a basic operation means replacing the left-hand side of the equation by the right-hand side, thus forgetting any markings of the replaced triple(s). To improve readability, we write them down only for G_{123}, but they are also defined for G_{341}.

- Multiplication: $[u, x, k]_{(1,2)} \cdot [v, y, \ell]_{(1,2)} = [u2^{-y} + v2^k, x + y, k + \ell]_{(1,2)}$
- Swapping from $(1, 2)$ to $(2, 3)$: $[0, x, k]_{(1,2)} = [x + k, 0, 0]_{(2,3)}$
- Swapping from $(2, 3)$ to $(1, 2)$: $[z, 0, 0]_{(2,3)} = \begin{cases} [0, 0, z]_{(1,2)} & \text{for } z \geq 0 \\ [0, z, 0]_{(1,2)} & \text{for } z < 0. \end{cases}$
- Splitting: $[u, x, k]_{(1,2)} = [u2^x, 0, 0]_{(1,2)} \cdot [0, x, k]_{(1,2)}$ for $u2^x \in \mathbb{Z}$
  $[u, x, k]_{(2,3)} = [0, x, k]_{(2,3)} \cdot [u2^{-k}, 0, 0]_{(2,3)}$ for $u2^{-k} \in \mathbb{Z}$
  We allow splitting operations only when immediatelly followed by a multiplication, thus we never increase the number of triple markings inside $\mathcal{T}$.

We keep $\mathcal{T}$ as a main data structure by doing addition and multiplication by powers of 2 using clones (as described in Section 2) and calling ExtendReduction on these after each basic operation.

▶ **Proposition** 11. Let $\mathcal{T} = (\Gamma, \delta; (T_j)_{j\in J})$ be a main data structure of size at most m, weight at most w (and with $|J| + w \leq m$). The following assertions hold.

i) No basic operation increases the weight of $\mathcal{T}$.
ii) Each basic operation increases the size $\|\mathcal{T}\|$ by $\mathcal{O}(w)$.
iii) Each basic operation takes time $\widetilde{\mathcal{O}}(mw)$.
iv) A sequence of s basic operations takes time $\widetilde{\mathcal{O}}(s^2m^2)$ and the size of $\mathcal{T}$ remains bounded by $\mathcal{O}(m + sw)$.

Proof. We can do the necessary tests, because the power circuit is reduced (Proposition 5). Before each operation we replace the involved markings in $(T_j)_{j\in J}$ by clones, which increases the size by $\mathcal{O}(w)$, but does not increase the weight. Note that there is enough time to create the clones with all their outgoing arcs. With the new clones we can perform the operations by using the algorithms described in Section 2. We regain the main data structure by calling ExtendReduction which integrates the modified clones into the reduced representation.

In order to get iv), we observe that the final size of the circuit is bounded by $\mathcal{O}(m + sw)$, so we need at most $s \cdot \widetilde{\mathcal{O}}((m + sw) \cdot w) \subseteq \widetilde{\mathcal{O}}(s^2m^2)$ time for all s operations. ◀

▶ **Theorem 12.** *The Word Problem of H_4 can be solved in time $\widetilde{\mathcal{O}}(n^6)$.*

The traditional input for a Word Problem solving algorithm is a word over the generators a_p and their inverses a_p^{-1}. We solve a slightly more general problem by assuming that the input consists of a power circuit $\Pi = (\Gamma, \delta)$ together with a sequence of s triple markings of various types. Each triple marking $[U, X, K]_{(p,p+1)}$ corresponds to $a_{p+1}^{\varepsilon(X)} a_p^{\varepsilon(U)} a_{p+1}^{\varepsilon(K)} \in H_4$.

Let w be the total weight of $\mathcal{T} = (\Gamma, \delta; (T_j)_{1\leq j\leq s})$. For simplicity we assume $s \leq w$ and that w and sizes of clones are bounded by $\|\mathcal{T}\| = |\Gamma|$. Having $s \leq w \leq n \in \mathcal{O}(w)$, we can think of $n = |\Gamma|$ as our input size. We transform $\mathcal{T}$ into a main data structure by invoking Reduce.

During the procedure $|\Gamma|$ increases, but the number of triple markings remains bounded by s and the weight by w. In order to achieve our main result we show how to solve the word problem with $\mathcal{O}(s^2)$ basic operations on the main data structure $\mathcal{T}$. Assuming this, by Proposition 11, the final size will be bounded by $m \in \mathcal{O}(s^2 w)$; and the time for all basic operations is $\widetilde{\mathcal{O}}(s^4 w^2) \subseteq \widetilde{\mathcal{O}}(n^6)$.

We collect sequences of triple markings of type $(1,2)$ and $(2,3)$ in *intervals* $\mathcal{L}$, which in turn receive type $(1,2,3)$; and we collect triple markings of type $(3,4)$ and $(4,1)$ in intervals of type $(3,4,1)$. Each interval has (as a sequence of triple markings) a semantics $\varepsilon(\mathcal{L})$ which is a group element either in G_{123} or in G_{341} depending on the type of $\mathcal{L}$. Thus, it makes sense to ask whether $\varepsilon(\mathcal{L}) \in F_{13}$. These tests are crucial and dominate the runtime of the algorithm.

Now the sequence $(T_j)_{1 \le j \le s}$ of triple markings appears as a sequence of intervals:

$$(\mathcal{L}_1, \ldots, \mathcal{L}_f; \mathcal{L}_{f+1}, \ldots, \mathcal{L}_t).$$

We introduce a separator ";" dividing the list in two parts. The following invariants are kept up:

i) All $\mathcal{L}_1, \ldots, \mathcal{L}_f$ satisfy $\varepsilon(\mathcal{L}_i) \notin F_{13}$. In particular, these intervals are not empty and they represent non-trivial group elements in $(G_{123} \cup G_{341}) \setminus F_{13}$.
ii) The types of intervals left of the separator are alternating.

In the beginning each interval consists of exactly one triple marking, thus $f = 0$ and $t = s$. The algorithm will stop either with $1 \le f = t$ or with $f = 0$ and $t = 1$.

Now we describe how to move forward: First, assume $f = 0$ (thus $t > 1$). If $\varepsilon(\mathcal{L}_1) \notin F_{13}$, then move the separator to the right, i.e. we obtain $f = 1$. If $\varepsilon(\mathcal{L}_1) \in F_{13}$, then, after swapping $\mathcal{L}_1$, we can join the intervals $\mathcal{L}_1$ and $\mathcal{L}_2$. In this case we still have $f = 0$, but t decreases by 1.

Now assume that $0 < f < t$. If $\mathcal{L}_f$ and $\mathcal{L}_{f+1}$ have the same type, then append $\mathcal{L}_{f+1}$ to $\mathcal{L}_f$, and move the separator to the left of $\mathcal{L}_f$. Thus, f and t decrease by 1.

If $\mathcal{L}_f$ and $\mathcal{L}_{f+1}$ have different types, then we test whether $\varepsilon(\mathcal{L}_{f+1}) \in F_{13}$. If $\varepsilon(\mathcal{L}_{f+1}) \notin F_{13}$, then move the separator to the right, i.e. f increases by 1. If $\varepsilon(\mathcal{L}_{f+1}) \in F_{13}$, then we swap $\mathcal{L}_{f+1}$ and join the intervals $\mathcal{L}_f$ and $\mathcal{L}_{f+1}$ into one new interval. Since we do not know whether the new interval belongs to F_{13}, we put the separator in front of it, decreasing both f and t by 1.

If we terminate with $1 \le f = t$, then $\varepsilon(\mathcal{L}_1) \cdots \varepsilon(\mathcal{L}_t) \in H_4$ is a Britton-reduced sequence in the amalgamated product. It represents a non-trivial group element, since $t \ge 1$.

In the other case we terminate with $f = 0$ and $t = 1$. We will make sure that the subgroup membership test for F_{13} can as a by-product also answer the question whether a sequence $\varepsilon(\mathcal{L})$ represents the trivial group element. If we do so, one more test on $(\mathcal{L}_1)$ yields the answer we need.

The number of membership tests for F_{13} is bounded by $2s$. All that remains, is to prove:

▶ **Lemma 13.** *The test for membership of $\varepsilon(\mathcal{L})$ in the subgroup F_{13} can be realized with $\mathcal{O}(s)$ basic operations in the main data structure $\mathcal{T}$. The test yields either "no" or it says "yes" with the additional information whether or not $\varepsilon(\mathcal{L})$ is the trivial group element. Moreover, in the "yes" case we can also swap the type of $\mathcal{L}$ within the same bound on basic operations.*

Proof. Assume that $\mathcal{L}$ is of type $(1,2,3)$, so it contains only triples of types $(1,2)$ and $(2,3)$. Let s be the length of $\mathcal{L}$. In a first round we create a sequence of triple markings $(T_1, \ldots, T_t)$ with $t \le s$ such that for $1 \le i < t$ the type of T_i is $(1,2)$ if and only if the type of T_{i+1} is

$(2,3)$. We can do so by $s-t$ multiplications from left to right without changing the semantics of $g = \varepsilon(T_1)\cdots\varepsilon(T_t) \in G_{123}$.

Next, we make this sequence Britton-reduced (which also gives us the information whether the sequence represents the identity in the group). Again, we scan from left to right. If we are at $T = T_i$ with value $[u,x,k]$ we have to check whether either $[u,x,k]_{(1,2)} = (0,z) \in \mathbb{Z}[1/2]\rtimes\mathbb{Z}$ or $[u,x,k]_{(2,3)} = (z,0) \in \mathbb{Z}[1/2]\rtimes\mathbb{Z}$ for some integer $z \in \mathbb{Z}$.

For the type $(1,2)$ we have $[u,x,k]_{(1,2)} = (0,z)$ if and only if $u = 0$, which in a reduced circuit means that the support of the marking for u is empty. Hence this test is trivial. If the test is positive, we can replace $[u,x,k]_{(1,2)}$ by $[0,x,k]_{(1,2)}$ and perform a swap to type $(2,3)$. If $t > 1$ we perform multiplications with its neighbors, thereby decreasing the value t.

For the type $(2,3)$ we have $[u,x,k]_{(2,3)} = (z,0)$ if and only if both $k+x = 0$ and $u2^x \in \mathbb{Z}$. These tests are possible in linear time and if successful, we continue as in the precedent case.

The final steps are more subtle. Let $\varepsilon(T_j) = g_j \in G_{12}\cup G_{23}$. Recall that $(g_1,\ldots,g_t)$ is already a Britton-reduced sequence. We have $g_1\cdots g_t \in F_{13}$ if and only if there is a sequence $(h_0,h_1,\ldots,h_t)$ with the following properties:

i) $h_0 = h_t = 1$ and $h_j \in \langle a_2\rangle$ for all $0 \le j \le t$.
ii) $h_{j-1}g_j = g'_j h_j$ with $g'_j \in F_1 \cup F_3$ for all $1 \le j \le t$.

Assume that such a sequence $(h_0,h_1,\ldots,h_t)$ exists. Then we have $g'_j \in \langle a_1\rangle$ if and only if $g_j \in G_{12}$. Moreover, whenever $gh = g'h' \in G_{123}$ with $g,g' \in F_1\cup F_3$ and $h,h' \in \langle a_2\rangle$, then $g = g'$ and $h = h'$. This follows because $g'^{-1}g = h'h^{-1} \in F_{13}\cap\langle a_2\rangle = \{1\}$. Thus, the product $h_{j-1}g_j$ uniquely defines $g'_j \in F_1\cup F_3$ and $h_j \in \langle a_2\rangle$, because $h_0 = 1$ is fixed.

The invariant during a computation from left to right is that $\varepsilon(T_j) = h_{j-1}g_j$. We obtain $\varepsilon(T_j) = g'_j h_j$ by a splitting operation. If no splitting is possible we know that $g \notin F_{13}$ and we can stop. If, however, a splitting is possible, then we have two cases. If j is the last index ($j = t$), then, in addition, we must have $h_j = 1$. We can test this. If the test fails, we stop with $g \notin F_{13}$. If we are not at the last index we perform a swap of the right-hand factor and multiply it with the next triple marking, which has the correct type to do so. As our sequence has been Britton-reduced, the total number of triple markings remains constant. There can be no cancelations at this point. Thus, the test gives us the answer to the subgroup membership problem using $\mathcal{O}(s)$ basic operations. ◀

6 Conclusion and future research

The Word Problem is a fundamental problem in algorithmic group theory. In some sense "almost all" finitely presented groups are hyperbolic [22] and satisfy a "small cancelation" property, so the Word Problem is solvable in linear time! For hyperbolic groups there are also efficient parallel algorithms and the Word Problem is in $\mathbf{NC^2}$ [4]. On the other hand, for many naturally defined groups little is known. Among one-relator groups the Baumslag group $G_{(1,2)}$ was supposed to have the hardest Word Problem [19], but we solved it in cubic time. The method generalizes to the higher Baumslag groups $G_{(m,n)}$ in case that m divides n, but this requires more "power circuit machinery" and has not been worked out in full details yet, see [19]. The situation for $G_{(2,3)}$ is open and related to questions in number theory. Baumslag and Higman groups are built via HNN extensions and amalgamated products. Many algorithmic problems are open for such constructions, for advances about theories of HNN-extensions and amalgamated products we refer to [14].

Another interesting open problem concerns the Word Problem in *Hydra* groups. Doubled hydra groups have Ackermannian Dehn functions [8], but still it is possible that their Word Problem is solvable in polynomial time.

References

1 G. Baumslag. A non-cyclic one-relator group all of whose finite quotients are cyclic. *J. Austr. Math. Soc.*, 10(3-4):497–498, 1969.

2 G. Baumslag, A. G. Myasnikov, and V. Shpilrain. Open problems in combinatorial group theory. Second Edition. In *Combinatorial and geometric group theory*, volume 296 of *Contemporary Mathematics*, pages 1–38. American Mathematical Society, 2002.

3 W. W. Boone. The Word Problem. *Annals of Mathematics*, 70(2):207–265, 1959.

4 J.-Y. Cai. Parallel computation over hyperbolic groups. In *Proc. 24th ACM Symp. on Theory of Computing, STOC 92*, pages 106–115. ACM-press, 1992.

5 M. Dehn. Über unendliche diskontinuierliche Gruppen. *Math. Ann.*, 71(1):116–144, 1911.

6 V. Diekert, A. J. Duncan, and A. G. Myasnikov. Geodesic rewriting systems and pregroups. In O. Bogopolski et al. (eds.), *Combinatorial and Geometric Group Theory*, Trends in Mathematics, pages 55–91. Birkhäuser, 2010.

7 V. Diekert, J. Laun, and A. Ushakov. Efficient algorithms for highly compressed data: The Word Problem in Higman's group is in P. *ArXiv e-prints*, Mar. 2011.

8 W. Dison and T. R. Riley. Hydra groups. *ArXiv e-prints*, abs/1002.1945, Feb. 2010.

9 S. M. Gersten. Isodiametric and isoperimetric inequalities in group extensions. 1991.

10 G. Higman. A finitely generated infinite simple group. *J. LMS*, 26:61–64, 1951.

11 I. Kapovich, A. G. Miasnikov, P. Schupp, and V. Shpilrain. Generic-case complexity, decision problems in group theory and random walks. *J. Algebra*, 264:665–694, 2003.

12 M. Lohrey. Word problems and membership problems on compressed words. *SIAM J. Comput.*, 35(5):1210–1240, 2006.

13 M. Lohrey and S. Schleimer. Efficient computation in groups via compression. In V. Diekert et al. (eds.), *CSR*, *LNCS*, 4649:249–258. Springer, 2007.

14 M. Lohrey and G. Sénizergues. Theories of HNN-extensions and amalgamated products. In M. Bugliesi et al. (eds.), *ICALP*, *LNCS*, 4052:504–515. Springer, 2006.

15 R. Lyndon and P. Schupp. *Combinatorial Group Theory*. Springer, 2001.

16 K. Madlener and F. Otto. Pseudo-natural algorithms for finitely generated presentations of monoids and groups. *J. Symb. Comput.*, 5:339–358, 1988.

17 W. Magnus. Das Identitätsproblem für Gruppen mit einer definierenden Relation. *Math. Ann.*, 106:295–307, 1932.

18 A. G. Myasnikov, A. Ushakov, and D. W. Won. Power circuits, exponential algebra, and time complexity. *ArXiv e-prints*, abs/1006.2570, 2010. To appear in IJAC.

19 A. G. Myasnikov, A. Ushakov, and D. W. Won. The Word Problem in the Baumslag group with a non-elementary Dehn function is polynomial time decidable. *Journal of Algebra*, 345(1):324–342, 2011.

20 A. G. Myasnikov and S. Ushakov. Cryptography And Groups (CRAG). Software Library.

21 P. S. Novikov. On the algorithmic unsolvability of the word problem in group theory. *Trudy Mat. Inst. Steklov*, pages 1–143, 1955. In Russian.

22 A. Y. Ol'shanskii. Almost every group is hyperbolic. *Int. J. Alg. Comp.*, 2:1–17, 1992.

23 F. Otto, D. E. Cohen, and K. Madlener. Separating the intrinsic complexity and the derivational complexity of the word problem for finitely presented groups. *Math. Log. Q.*, 39:143–157, 1993.

24 A. N. Platonov. Isoparametric function of the Baumslag-Gersten group. *Vestnik Moskov. Univ. Ser. I Mat. Mekh.*, 3:12–17, 2004. Russian. Engl. transl. Moscow Univ. Math. Bull. 59 (3) (2004), 12–17.

25 M. V. Sapir, J.-C. Birget, and E. Rips. Isoperimetric and Isodiametric Functions of Groups. *Ann. Math.*, 156:345–466, 2002.

26 S. Schleimer. Polynomial-time word problems. *Comm. Math. Helv.*, 83:741–765, 2008.

27 J.-P. Serre. *Trees*. Springer, 1980.