ON GELMAN'S SUBGROUP COUNTING THEOREM

ERIC FREDEN AND MICHAEL GRADY

ABSTRACT. In a recent paper, E. Gelman gave an exact formula for the number of subgroups of given index for the Baumslag-Solitar groups $BS(p,q)$ when p and q are coprime. We use Gelman's proof as the basis of an algorithm to compute a maximal set of inequivalent permutation representations of $BS(p,q)$ having degree n. The computational complexity for each representation is linear in both space and time as a function of the index. We compare the performance of this algorithm with the powerful Todd-Coxeter procedure, which in general has no polynomial bound on the number of cosets used in the enumeration process.

1. INTRODUCTION

The Baumslag-Solitar groups are a class of two-generator one-relator groups that have played an important role in combinatorial group theory. They are defined by the presentation: $BS(p,q) = \langle t, b \mid tb^p t^{-1} = b^q \rangle$. A familiar example is $BS(1,1)$, the free abelian group on two generators, $\mathbb{Z} \times \mathbb{Z}$. Recent results by Gelman, Button and Dudkin provide a wealth of information on the subgroups of $BS(p,q)$, including the number of subgroups of given index, the number of normal subgroups of given index and canonical subgroup generators [1,3,7]. We will show that these results readily yield efficient algorithms for computing with these groups.

Gelman's subgroup counting theorem is of particular interest in this investigation. It gives an exact formula for the number of subgroups of given index for $BS(p,q)$ when p and q are coprime [7].

Theorem 1 (Gelman). *Let p and q be different from zero such that gcd(p,q) = 1, and let $a_n(p,q)$ denote the number of subgroups of $BS(p,q)$ having index n. Then*

JCMCC **90** (2014), pp. 21-37

$$a_n(p,q) = \sum_{\substack{d|n \\ gcd(d,pq)=1}} d$$

The proof involves an elegant counting argument for the number of permutation representations of $BS(p,q)$ having degree n. We use Gelman's proof as the basis of an algorithm to compute a maximal set of inequivalent permutation representations of $BS(p,q)$ having degree n. The computational complexity for each representation is linear in both space and time as a function of the index. We compare the performance of this algorithm with the powerful Todd-Coxeter procedure, which in general has no polynomial bound on the number of cosets used in the enumeration process.

After describing a fundamental technique for counting the subgroups, we illustrate its application using three different computational approaches: a brute force algorithm, coset enumeration and an algorithm based on Gelman's proof. GAP code for the algorithms given in this paper appears in the Appendix.

2. Permutation Representations

A fundamental technique for counting the subgroups of a finitely generated group involves first counting its transitive permutation representations. Let G be a finitely generated group and H a subgroup of index n in G. Then G permutes the cosets of H by right multiplication. Giving H the label 1 and labeling the remaining cosets 2 through n in any order, this group action yields a transitive permutation representation

$$\phi : G \to S_n$$

where S_n is the symmetric group on n symbols. Conversely, given any transitive permutation representation of G having degree n, there is a subgroup H of index n in G. It is the inverse image of the stabilizer of 1 [9].

Putting

$trans_n(G) = |\{\phi : G \to S_n \mid \phi(G) \text{ is transitive}\}|$

$norm_n(G) = |\{\phi : G \to S_n \mid \phi(G) \text{ is transitive and Order}(\phi(G)) = n\}|$

we have

Proposition 2. *The number of subgroups of index n in G is*

$$\frac{trans_n(G)}{(n-1)!}$$

Proof. The labeling of cosets 2 through n described above was arbitrary. There are $(n-1)!$ such labelings, all representing equivalent group actions. □

Proposition 3. *The number of normal subgroups of index n in G is*

$$\frac{norm_n(G)}{(n-1)!}$$

Proof. If $\phi(G)$ is transitive of degree and order n, then the point stabilizer is the identity and $\phi(G)$ is a regular permutation group. Now H is the inverse image of the stabilizer of 1, so $H = Ker(\phi)$. But $Ker(\phi)$ is the largest normal subgroup of G contained in H (see [2], pages 8-12). □

In the remainder of this paper the symbols 't' and 'b' will denote the generators of $BS(p,q)$, while 'τ' and 'β' will denote their finite images in S_n.

3. Brute force search

A simple example will be used to illustrate the wealth of information that can be extracted from this fundamental technique. Consider the following brute force algorithm for finding subgroups of $BS(2,3)$ having index n: for every ordered pair of elements τ and β in S_n, test if they satisfy the relation $\tau\beta^2\tau^{-1} = \beta^3$ and that the group they generate is transitive on $\{1,2,\cdots,n\}$. Here, $\langle\tau,\beta\rangle$ denotes the subgroup of S_n generated by τ and β, and the psuedocode resembles GAP syntax.

BRUTE-FORCE-PERM-REPS(n)

Input: Positive integer n.

Output: Generators for every permutation representation of degree n and the total number of subgroups of index n in $BS(2,3)$.

count = 0
for every ordered pair $(\tau, \beta) \in S_n \times S_n$
    $\Gamma = \langle \tau, \beta \rangle$
    if $\tau\beta^2\tau^{-1}\beta^{-3} = ()$ and Γ is transitive on $\{1, 2, ...n\}$
        count = *count* + 1
        Print τ, β
Print "The total number of subgroups of index n is: " *count*$/(n-1)!$

Running the corresponding GAP code for the case $n = 5$ produces 144 pairs satisfying the conditions, giving $144/4! = 6$ distinct subgroups of index 5 (see Appendix). Generator pairs for the corresponding inequivalent permutation representations are given below:

$\tau = (1,2)(3,5)$ $\beta = (1,2,3,4,5)$
$\tau = (1,5)(2,4)$ $\beta = (1,2,3,4,5)$
$\tau = (1,3)(4,5)$ $\beta = (1,2,3,4,5)$
$\tau = (1,4)(2,3)$ $\beta = (1,2,3,4,5)$
$\tau = (2,5)(3,4)$ $\beta = (1,2,3,4,5)$
$\tau = (1,2,3,4,5)$ $\beta = ()$

For example, one of the pairs is $\tau = (1,4)(2,3)$, $\beta = (1,2,3,4,5)$, and a Schreier coset graph for this pair is shown in Figure 3.1. The vertices represent the cosets of a subgroup $H < BS(2,3)$ where H is represented by the label 1. The directed edges record the action of the generators of $BS(2,3)$ on the cosets of H. In this case, the solid edges correspond to multiplication by b and the dashed edges show multiplication by t.

A Schreier coset graph is a finite state automata that can compute subgroup membership. Any path that starts and ends on label 1 spells out a word whose preimage lies in H. This can be detected computationally as follows: w lies in H if $\phi(w)$ lies in the stabilizer of 1 for $\langle \tau, \beta \rangle$. For example, the words b^5, tb^2 and t^2 lie in H. From the Schreier graph, it is clear that the labeling of cosets 2 through 5 is arbitrary, and that any permutation of these labels would yield an automata accepting precisely the same elements. To visualize what this subgroup looks like, we start with a partial Cayley graph of $BS(2,3)$, then mark those elements accepted by the automata (Figure 3.2). The vertices indicate group elements. Upward directed edges indicate multiplication by t, downward by t^{-1}. Right directed edges indicate multiplication by b, leftward by b^{-1}. The picture that

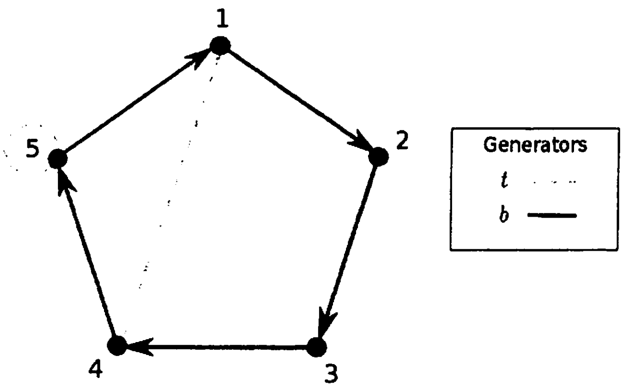

FIGURE 3.1. Schreier graph for the subgroup $\langle b^5, tb^2\rangle$ having index 5 in $BS(2,3)$.

emerges is similar to a sublattice of the integral lattice $\mathbb{Z} \times \mathbb{Z}$. Subgroups of index 5 are similar to sublattices that, so to speak, use every fifth lattice point. This figure shows only a small portion of the main sheet, the plane containing the subgroups $\langle t\rangle$ and $\langle b\rangle$. The full Cayley graph would require gluing an infinite number of planar sheets, as hinted at in Figure 3.3.

The Schreier graph encodes additional information, for example:

1. H cannot be a normal subgroup since the Schreier graph is not a Cayley graph. In fact, $\langle \tau\rangle$ is the point stabilizer of 5. Since τ has order 2, the order of the permutation group $\langle \tau, \beta\rangle$ is 10, by the Orbit-Stabilizer Theorem.

2. H has five conjugates: we can get five inequivalent Schreier graphs by rotating the labels, (and thus moving the point stabilizer). These correspond to the first five generator pairs shown above. Their Cayley subgroup graphs would look like Figure 3.2 with every other row shifted by from one to four places.

The last generator pair listed above, namely $\tau = (1,2,3,4,5)$, $\beta = ()$ obviously generate a permutation group of order 5, the same as the index. The corresponding Schreier graph is the Cayley graph of the cyclic group of order 5. The inverse image of the stabilizer of 1 in $\langle \tau, \beta\rangle$ is the subgroup $H = \langle b, t^5\rangle$. Therefore H is normal in $BS(2,3)$.

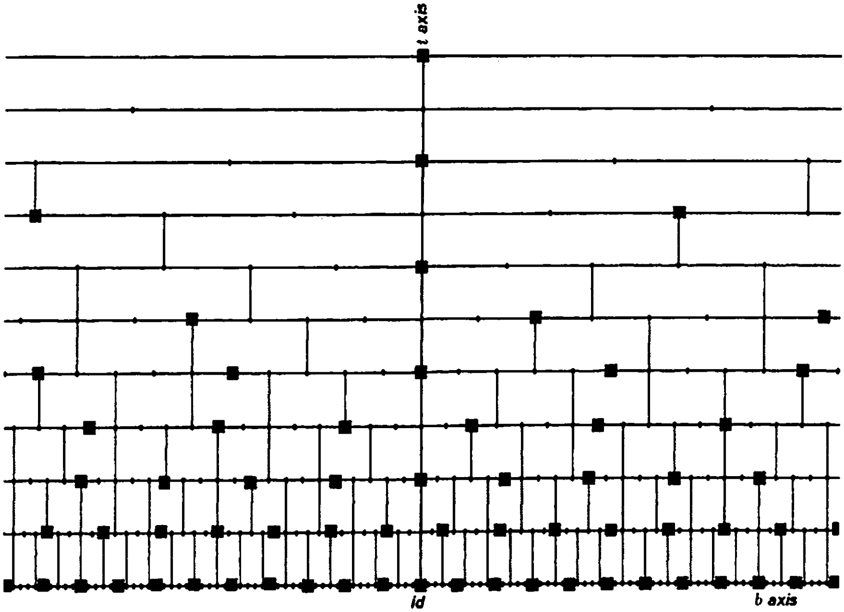

FIGURE 3.2. Partial Cayley graph for a subgroup $\langle b^5, tb^2 \rangle$ having index 5 in $BS(2,3)$.

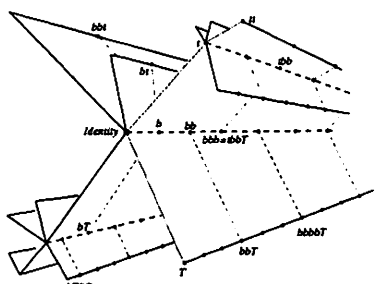

FIGURE 3.3. Partial Cayley 2-complex for $BS(2,3)$.

This naïve approach, while illuminating, clearly does not scale since it has complexity $O(n!^2)$. More effective techniques will be described in the next two sections.

4. Counting Cosets

Todd-Coxeter coset enumeration is one of the most important procedures in computational group theory. If group G is given by a finite presentation and H is given by a set of generators lying in G, the procedure attempts to verify that $|G : H|$ is finite. If the index is not finite, the procedure will run forever. Even when the index is finite and small, a computation can run out of space since there is no polynomial bound on the number of cosets used in the enumeration process. However, if the procedure terminates, it will return the index along with a coset table, which is equivalent to a transitive permutation representation of G on the cosets of H (see [8], page 149).

The Todd-Coxeter procedure is in many instances a powerful and effective approach, but requires a set of subgroup generators. Dudkin has given a set of canonical generators for $BS(p,q)$ [3].

Proposition 4 (Dudkin). *For every divisor d satisfying the Gelman conditions $d|n$ and $gcd(d,pq) = 1$, the subgroups of $BS(p,q)$ with index n are generated as: $\langle b^d, t^{n/d}b^i\rangle$ for $0 \leq i < d$.*

There are several GAP commands that perform coset enumeration. Among the most useful are Index, FactorCosetOperation and CosetTable. The Index(G,H) command returns the index of subgroup H in G. FactorCosetOperation(G,H) returns the action of G on the cosets of H, while CosetTable(G,H) returns a coset table consisting of a generator list for each generator and its inverse. The following GAP code illustrates the use of the first two commands to verify that the subgroup generated by b^5 and tb^2 found above does indeed have index 5 in $BS(2,3)$ and to display a suitable permutation representation. It begins by defining $BS(2,3)$ as the quotient of the free group on two generators by the normal closure of the defining relator.

GAP code for computing index and permutation representations:

```
f := FreeGroup(2);
x := f.1;
y := f.2;
rels := [ x*y^2*x^(-1)*y^(-3) ];
G := f/rels;
t := G.1;
b := G.2;
```

Index n	Cosets Required
18	> 256,000
19	> 512,000
20	> 1 million
21	> 2 million
22	> 4 million
23	> 8 million
24	> 16 million

TABLE 1. Space complexity of coset enumeration for subgroups $\langle b, t^n \rangle$ in $BS(2,3)$.

```
H := Subgroup(G, [b^5, t*b^2]);
Print(Index(G,H));
Print(FactorCosetOperation(G,H));
```

Even simple programs like this will quickly encounter problem cases for the default coset enumeration procedure. The normal subgroups with generators $[b, t^n]$ are especially conspicuous. Table 1 shows how many cosets are required to handle the cases $18 \leq n \leq 24$:

When $n = 17$, the computation requires less than 256,000 cosets and is reasonably fast. The space required then doubles each time n is incremented by 1 until it quickly consumes whatever memory resources are available. It is common to encounter difficulties of this kind when using the Todd-Coxeter procedure over a wide range of indices [6]. By throwing out these problem cases, GAP can find the remaining permutation representions up to index 60 in several minutes on a typical workstation before encountering similar difficulties. The excellent ACE package for GAP [4] can speed up the computation by a factor of 50 or more, but then encounters additional problem cases at higher indices. In contrast, the algorithm described in the next section has no problem cases and can compute all permutation representations up to $n = 100$ in less than 1 second.

5. Linear Time Algorithm

The Todd-Coxeter procedure is both powerful and general. However, as we have seen, it is common to encounter presentations which require excessive memory to process. It is therefore not surprising that more efficient computations can be made in limited domains given a deeper understanding of the peculiarities of the special case. Gelman provided this deeper understanding by showing that the generators τ, β must have the form given in Theorem 5. We use this result as the basis of an algorithm for computing a maximal set of inequivalent permutation representations of $BS(p,q)$ having degree n. The computational complexity for each representation is linear in both space and time as a function of the index. This means that questions such as index, subgroup membership and normality can also be answered in linear space and time.

The original proof of Theorem 1 was based on demonstrating that the generators τ, β for all transitive permutation representations of $BS(p,q)$ possess certain properties, then counting how many elements of S_n have these properties. In Theorem 5, we have stated these results explicitly and refer the reader to the Gelman paper for the proof.

Theorem 5 (Gelman). *Let $d|n$ with $gcd(d,pq) = 1$. Let τ and β generate a transitive permutation representation of $BS(p,q)$ on n symbols, i.e. $\langle \tau, \beta \rangle$ is a transitive subgroup of S_n such that $\tau\beta^p\tau^{-1} = \beta^q$. Then*

1. *β must be regular, consisting of k cycles of length d, where $k = n/d$.*
2. *τ has the effect of cyclically permuting the d-cycles of β.*

Using this result, the following algorithm describes how to compute a maximal set of d inequivalent permutation representations of $BS(p,q)$ having degree n where $d|n$ and $gcd(p,q) = gcd(d,pq) = 1$. All such representations share the same β. The computation for a given τ is done by the inner loop and clearly has linear space and time complexity as a function of the index.

```
GELMAN-PERM-REPS(p, q, n, d)
  Input: Positive integers p, q, n and d such that d|n,
           gcd(p, q) = gcd(d, pq) = 1.
  Output: Generators for d inequivalent permutation
           representations of degree n for BS(p, q).
 1  k = n/d
 2  a = (1, 2, ..., n)
 3  β = a^k
 4  qorbits = orbits of 1...n under the action of ⟨β^q⟩
 5  bq = concatenation of the lists in qorbits
 6  porbits = cyclic permutation of the orbits of 1...n
 7           under the action of ⟨β^p⟩
 8  for i from 0 to d − 1
 9       porbits = cyclic permutation of the elements in
10                the first orbit of porbits
11       bp = concatenation of the lists of porbits
12       for j from 1 to n
13            τ[bq[j]] = bp[j]
14       Print β, τ, d and i
```

Lines 2 and 3 give a permutation consisting of k cycles, each of length d. Lines 4 through 11 find the permutations τ which conjugate β^p to give β^q. Each of these permutation representions corresponds to the action of the generators of $BS(p,q)$ on the cosets of one of the subgroups $\langle b^d, t^k b^i\rangle$ for $0 \leq i \leq d-1$. GAP implementation of this algorithm for $BS(2,3)$ is given in the Appendix.

Theorem 6. *The algorithm GELMAN-PERM-REPS yields a maximal set of d inequivalent permutation representations of $BS(p,q)$ having degree n where $n = kd$ and $gcd(p,q) = gcd(d,pq) = 1$.*

Proof. The algorithm clearly yields d permutation representations, so we must show they are inequivalent. We do this by matching each representation with a corresponding subgroup $H \in \{\langle b^d, t^k b^i\rangle \mid 0 \leq i < d\}$ of $BS(p,q)$. Since β consists of k d-cycles, its order is d. Therefore β^d is in the Stabilizer of 1 of $\langle \tau, \beta\rangle$, so that b^d is in H. Let $\Lambda = [x_1, x_2, ..., x_d]$ denote the first orbit of β, the orbit containing 1. This will also be the first orbit of β^q. Now τ^k moves x_i to an element x_j in Λ by Theorem 5 and the order of τ is a multiple of k. Suppose that τ has order k. Then τ^k is in the Stabilizer of 1, implying $t^k \in H$, and $H = \langle b^d, t^k\rangle$ using Proposition 4. Suppose now that τ has order greater than k. Let τ_i denote the permutation generated in the i^{th} iteration of the algorithm. Suppose $\tau_i^k(1) =$

$\tau_j^k(1)$ with $i \neq j$. Then $\tau_i^{k-1}(1) = \tau_j^{k-1}(1)$, since each τ is a permutation. By induction, $\tau_i(1) = \tau_j(1)$, giving $i = j$, a contradiction. We can use this fact to determine the corresponding subgroup H of $BS(p,q)$. In the Schreier graph for $\langle \tau, \beta \rangle$, the orbits of β can be represented by nested d-sided polygons, where action by τ cyclically permutes the polygons. Let the outer d-gon contain the orbit of 1. This d-cycle shows that β^d is in the Stabilizer of 1, as indicated above. The fact that $\tau_i^k(1) = x_j$ shows that, τ_i^k is in the Stabilizer of Λ, so that a path starting at vertex 1 will end in the outer d-gon after applying τ k times. This means that $\tau^k \beta^i$ is also in the Stabilizer of 1 for some i in the range $[0, d-1]$. This corresponds to the subgroup $H = \langle b^d, t^k b^i \rangle$ of $BS(p,q)$. By the above remarks, all d distinct representations are found by the algorithm. □

By way of illustration, consider the case $n = 10, d = 5$ for $BS(2,3)$. Here, $k = n/d = 2$. So β consists of two 5-cycles. Let $a = (1,2,3,4,5,6,7,8,9,10)$, giving $\beta = a^k = (1,3,5,7,9)(2,4,6,8,10)$, $\beta^2 = (1,5,9,3,7)(2,6,10,4,8)$ and $\beta^3 = (1,7,3,9,5)(2,8,4,10,6)$. We must find the τ which conjugates β^2 to give β^3. For transitivity, it must cyclically permute the blocks of β. Rewrite β^2 by swapping the blocks and follow this with the well known algorithm for finding conjugates where one writes β^3 above β^2 and considers each element in the second row to be the image of the element above it.
Thus $\beta^3 = (1,7,3,9,5)(2,8,4,10,6)$
$\beta^2 = (2,6,10,4,8)(1,5,9,3,7)$
gives $\tau = (1,2)(3,10)(4,9)(5,8)(6,7)$.

The Schreier graph for this representation is shown in Figure 5.1. It corresponds to the normal subgroup $< b^5, t^2 >$, so the quotient of $BS(2,3)$ by $< b^5, t^2 >$ is isomorphic to the dihderal group D_{10}.

The remaining four representations are obtained by cyclically permuting the elements in the first block of β^2. Every shift corresponds to a different subgroup. That the corresponding representations are inequivalent is easy to see from the Schreier graphs. For example, here we shift the first cycle of β^2 three places:
$\beta^3 = (1,7,3,9,5)(2,8,4,10,6)$
$\beta^2 = (4,8,2,6,10)(1,5,9,3,7)$
giving $\tau = (1,4,9,6,7,8,5,10,3,2)$.

The Schreier graph for this representation is shown in Figure 5.2.

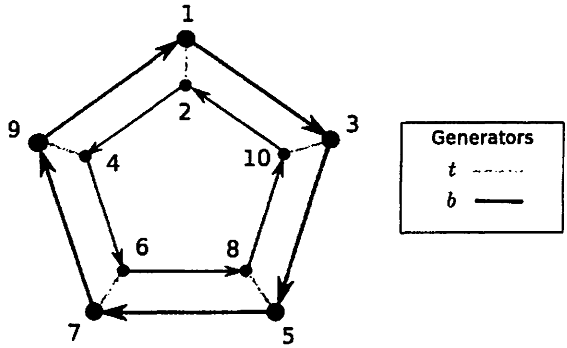

FIGURE 5.1. Schreier graph for the subgroup $\langle b^5, t^2\rangle$ having index 10 in $BS(2,3)$.

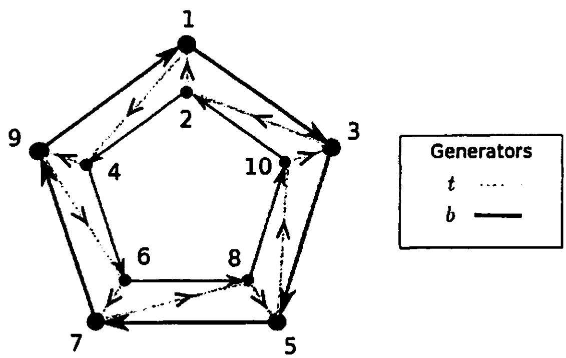

FIGURE 5.2. Schreier graph for the subgroup $\langle b^5, t^2 b\rangle$ having index 10 in $BS(2,3)$.

6. APPLICATIONS

Fast algorithms are useful in the hunt for patterns and for testing conjectures. The following result was hypothesized in this way, then subsequently proved.

Theorem 7. *Let $gcd(p,q) = 1$, $n = kd$, with $gcd(d,pq) = 1$. If H is a finite index subgroup of $BS(p,q)$ with canonical generators $\langle b^d, t^k b^i\rangle$, $i \le 0 < d$, then H is isomorphic to $BS(p^k, q^k)$.*

Before giving the proof, a few remarks on the geometry of Baumslag-Solitar groups is in order. For the general group $BS(p,q)$, we call $\langle b \rangle$ the horocyclic subgroup and the defining relator $tb^pt^{-1}b^{-q}$ is referred to as a "horobrick". The Cayley graph consists of "sheets" each of which is endowed with a coarse euclidean geometry (when $p = q$) or coarse hyperbolic geometry (when $p < q$) glued along $\langle b \rangle$-cosets referred to as "horocycles". (The underlying black grid in Figure 3.2 comprises such a sheet.) This geometry is quasi-isometric with the upper half space model of the hyperbolic plane (and thus satisfies the thin triangles criterion, making each sheet a Gromov hyperbolic space). The quasi-isometry induces a natural orientation to each sheet. Paths with labels $t^n, n > 0$ go "up", labels $b^{\pm n}$ are "horizontal", while paths labelled t^{-n} go "down".

Since the defining relator must be maintained in each sheet, each $\langle b \rangle$-coset of $BS(p,q)$ gives rise to q upper half-sheets and p lower half-sheets (see Figure 3.3). The Cayley 2-complex of a presentation is obtained from the Cayley graph by filling in each basic relator and its conjugates with a topological disk. For $BS(p,q)$ the Cayley 2-complex is homeomorphic to the product of the real line with a simplicial tree. We frequently adopt monoid notation and denote b^{-1} and t^{-1} by B and T , respectively. This is useful for representing strings in computer code as well as for annotating graphics.

Proof. First we show that the given presentation for H satisfies the necessary Baumslag-Solitar relation. Indeed,

$$(t^kb^i)(b^d)^{p^k}(t^kb^i)^{-1} = t^kb^{dp^k}t^{-k} = t^{k-1}(tb^{dp^k}t^{-1})t^{1-k} =$$

$$= t^{k-1}b^{dp^{k-1}q}t^{1-k} = t^{k-2}b^{dp^{k-2}q^2}t^{2-k} = \cdots = b^{dq^k} = (b^d)^{q^k}. \qquad (*)$$

Thus H is isomorphic to a quotient of $BS(p^k, q^k)$. We subsequently show that there are no additional independent relators.

Observe that any (cyclically reduced) relator $\mathcal{R}$ in H forms a relator in $BS(p,q)$ composed of horobricks glued together. The assumption that d is relatively prime to pq implies that horizontal edge paths formed by concatenating b^d don't sychronize with either the tops or bottoms of horizontal rows of the basic horobrick of $BS(p,q)$ unless we use at least p-many such horobricks.

If it were true that $t \in H$, we could create a relator $t(b^d)^pt^{-1}(b^d)^{-q}$. However, it is easy to see that t^k is the smallest power of t that occurs in any word in the generators of H. Thus if we want to wrap around a relator of $BS(p,q)$ using the generators of H, such a relator must be at least k horobricks high in the vertical direction. We have shown that our relation in equation $(*)$ above defines a minimal relator in H (minimal in the sense that there is no proper cyclically reduced relator inside it).

Starting at the identity vertex, tesselate the Cayley 2-complex of $BS(p,q)$ using generators of H as edge paths along with the minimal relator $(*)$ and its conjugates, thus creating a copy of the Cayley graph of $BS(p^k,q^k)$ as a subgraph of $BS(p,q)$. We claim this subgraph is that of H.

Indeed, any cyclically reduced relator $\mathcal{R}$ that is not a consequence of equation $(*)$ will also comprise a relator in $BS(p,q)$, necessarily consisting of a union of elementary horobricks. Now we exploit the fact that Baumslag-Solitar groups are essentially 2-dimensional: the intersection of $\mathcal{R}$ and the Cayley 2-subcomplex corresponding to $BS(p^k,q^k)$ will contain a 2-cell properly contained within a copy of the minimal relator defined by equation $(*)$ (else $\mathcal{R}$ is a consequence of this minimal relator after all). But the boundary of this supposed 2-cell defines a proper relator inside $(*)$ which is impossible. We conclude that $\mathcal{R}$ fails to exist and thus H is isomorphic to $BS(p^k,q^k)$. □

References

[1] J. O. Button, A formula for the normal subgroup growth of Baumslag-Solitar groups, *J. Group Theory* **11** (2008) 879-884.

[2] P. Cameron, *Permutation Groups*, Cambridge University Press, Cambridge, 1999.

[3] F. A. Dudkin, Subgroups of finite index in Baumslag-Solitar groups, *Algebra and Logic* **49** (2010) 221-232.

[4] Gamble, Hulpke, Havas, Ramsey, Advanced Coset Enumerator, Version 5.1, 2012 (http://www.gap-system.org).

[5] The GAP group, GAP – Groups, Algorithms and Programming, Version 4.5.4, 2012 (http://www.gap-system.org).

[6] G. Havas, C. Ramsey, Experiments in Coset Enumeration, *Groups and Computation* **III** (2001) 183-187.

[7] E. Gelman, Subgroup growth of Baumslag-Solitar groups, *J. Group Theory* **8** (2005) 801-806.

[8] Holt, Eick, O'Brien, *Handbook of Computational Group Theory*, CRC Press, 2005.

[9] A. Lubotsky, D. Segal, *Subgroup Growth*, Birkhauser, Basel, 2003.

Appendix

Here we give GAP implementations for the two algorithms described in this paper: BRUTE-FORCE-PERM-REPS from section 3 and GELMAN-PERM-REPS from section 5.

GAP code for BRUTE-FORCE-PERM-REPS:

```
# Usage: BRUTE_FORCE_PERM_REPS(n)
# Returns generators for all permutation representations of BS(2,3)
# having index n, and a count of the total number of subgroups with
# that index.
# Warning: Complexity is O(n!*n!).

BRUTE_FORCE_PERM_REPS := function(n)
local sym,count,t,b;
sym := SymmetricGroup(n);
count := 0;
for t in sym do
for b in sym do
   if (t*b^2*t^(-1)*b^(-3) = ()) and
(IsTransitive(Group(t,b),[1..n])) then
Print("t = ", t, " ", "b = ", b, "\n");
   count := count + 1;
   fi;
od;
od;
Print("Total number of perm reps = ", count, "\n");
Print("Total number of subgroups of index ", n, " is: ",
count/Factorial(n-1), "\n");
return;
end;

# Example:
BRUTE_FORCE_PERM_REPS(5);
```

GAP code for GELMAN-PERM-REPS:

```
# Usage: GELMAN_PERM_REPS(p,q,n,d)
# Input: Positive integers p, q, n and d such that d|n and (p,q) =
# (d,pq)=1.
```

```
# Output: Generators for all d permutation representations of degr
# in BS(p,q).

GELMAN_PERM_REPS := function(p,q,n,d)
local k,lst,ncycle,b,qorbits,porbits,i,bq,bp,kcycle,a,a2,t;

if (n mod d) = 0 then
if Gcd(d,6) = 1 then
k := n/d;
lst := [2..n];
Append(lst,[1]);
ncycle := PermList(lst);
b := ncycle^k;
qorbits := OrbitsPerms([b^q],[1..n]);
bq := Concatenation(qorbits);
porbits := OrbitsPerms([b^p],[1..n]);
# We need to cyclically permute the orbits of bp
kcycle := ();
if k > 1 then
lst := [2..k];
Append(lst,[1]);
kcycle := PermList(lst);
fi;
porbits := Permuted(porbits, kcycle);

for i in [0..d-1] do
# Cyclically permute the elements in the first
block of porbits
a := porbits[1];
a2 := a{[2..d]};
a2[d] := a[1];
     porbits[1] := a2;
bp := Concatenation(porbits);
     t := MappingPermListList(bq,bp);
Print("Perm rep ", i, " : t = ", t, "  b = ", b,
"\n");
od;
fi;
fi;
  return;
end;

# Example:
```

```
GELMAN_PERM_REPS(2,3,10,5);
```

Department of Mathematics, Southern Utah University, Cedar City UT

Department Computer Science & Informations Systems, Southern Utah University, Cedar City UT

E-mail address: freden@suu.edu

E-mail address: gradym@suu.edu