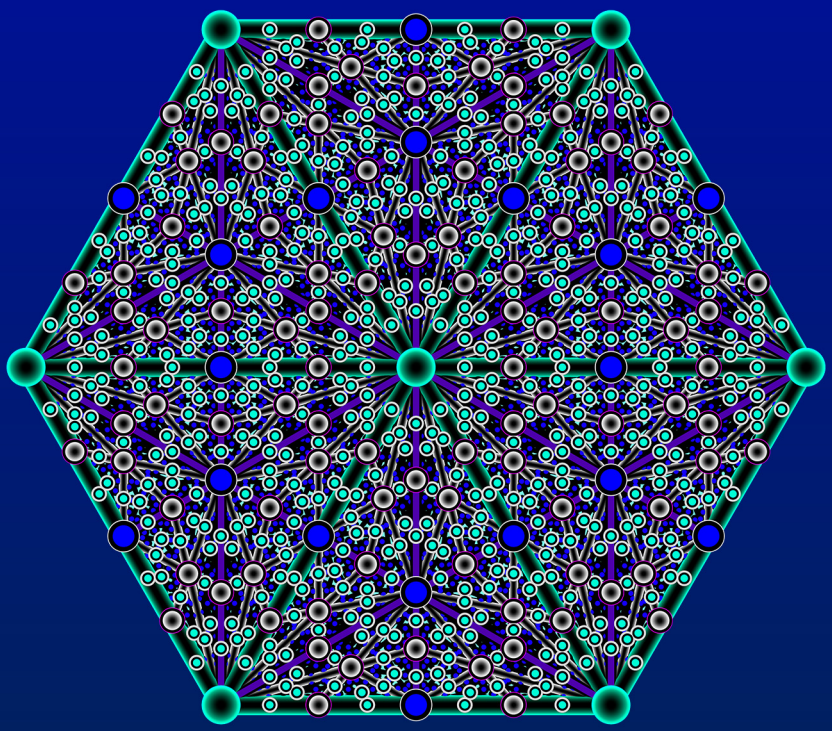
JOURNAL OF EXPERIMENTAL MATHEMATICS



ASSOCIATION FOR MATHEMATICAL RESEARCH

VOLUME 1 ISSUE 1

2025

JOURNAL OF EXPERIMENTAL MATHEMATICS

PUBLISHED BY

Association for Mathematical Research, amathr.org,
Davis, CA; Jenkintown PA.

JOURNAL OF EXPERIMENTAL MATHEMATICS, 1(1):36–50, 2025
jexpmath.org

4-strand Burau is unfaithful modulo 5

Joel Gibson Geordie Williamson * Oded Yacobi †

Received 18 Feb 2024; Revised 22 Apr 2024; Accepted 24 Apr 2024

Abstract:

We introduce a new algorithm for finding kernel elements in the Burau representation. Our algorithm applies reservoir sampling to a statistic on matrices which is closely correlated with Garside length. Using this we exhibit an explicit kernel element in the Burau representation on 4-strands reduced modulo 5.

Key words and phrases:
Braid Group; Burau Representation; Garside Normal Form; Reservoir Sampling

1 Introduction

The Burau representation of the n-strand braid group $\pi_n : B_n \to \mathrm{GL}_{n-1}(\mathbb{Z}[v,v^{-1}])$, plays a prominent role in many applications of braid groups, including to geometry, topology and mathematical physics [Bir74]. A key question underlying many of these developments is determining when π_n is faithful.

It is not difficult to prove that π_3 is faithful (cf. Propostion 3.3), but despite intensive study, there wasn't substantial progress on this question until 1991, when Moody proved that π_n is not faithful for $n \geq 9$ [Moo91]. His work was developed further by Long and Paton to show unfaithfulness for $n \geq 6$ [LP93]. Bigelow extended this also to $n = 5$ [Big99].

The case $n = 4$ remains open, and has attracted considerable attention. One reason for this is that a negative answer provides a non-trivial knot with trivial Jones polynomial, answering a central question in knot theory [Big02, Ito15].

Henceforth we set $\pi = \pi_4$. Cooper and Long approached the faithfulness of π by reducing modulo primes. That is, they considered the composition of π with the natural map $\mathrm{GL}_3(\mathbb{Z}[v,v^{-1}]) \to \mathrm{GL}_3(\mathbb{F}_p[v,v^{-1}])$, where $\mathbb{F}_p$ is the field with p elements [CL97]. Following their work, we denote this representation $\pi \otimes \mathbb{F}_p$. The study of the faithfulness of $\pi \otimes \mathbb{F}_p$ is interesting for several reasons:

*Supported by the Australian Research Council under the grant DP230100654
†Supported by the Australian Research Council under the grant DP230100654

DOI: 10.56994/JXM.001.001.003

1. Examples when $\pi \otimes \mathbb{F}_p$ is unfaithful can give insights into the difficulty of establishing the faithfulness of π.

2. It might be the case that $\pi \otimes \mathbb{F}_p$ is unfaithful modulo all primes, yet π is faithful. Controlling the kernel modulo infinitely many primes might provide a route to establishing faithfulness of π.

One of the main results in [CL97] is that $\pi \otimes \mathbb{F}_2$ is not faithful. In [CL98] the same authors extended their methods to show that $\pi \otimes \mathbb{F}_3$ is also not faithful. They noted their program runs into obstacles at $p = 5$: "This case remains open and has some features which suggest it may be different to the first two primes." Our main result resolves this case.

Theorem 1.1. *The representation $\pi \otimes \mathbb{F}_5$ is not faithful.*

Some remarks on our main theorem:

1. Our approach, which we discuss in the next section, is heavily computational and is quite different from the existing methods. A complete implementation of our algorithm (in Python) is available on github [GWY].

2. We discover several kernel elements modulo 5, the smallest of which is of Garside length 54[1]. We note that a straightforward computation shows that there are 10^{40} elements of this Garside length, thus finding this element by brute force search is not feasible.

3. We have made extensive searches with our algorithm to try to discover a kernel element of $\pi \otimes \mathbb{F}_7$, without success. Our methods are strong enough to demonstrate the known fact that the (integral) Burau representation for $n = 6$ is unfaithful. We were unable to rediscover Bigelow's (integral) kernel elements for $n = 5$, which indicates the limitations of our algorithm.

4. Recently, Datta proved that π is "generically faithful", in the sense that almost all braids are not in the kernel of π as their length tends to infinity [Dat]. Datta also proves that this property extends to $\pi \otimes \mathbb{F}_p$ for $p > 3$. Combined with our work, $\pi \otimes \mathbb{F}_5$ therefore provides an example of a generically faithful representation which is not faithful.

Our original goal was to use Machine Learning techniques to attack this problem. Thus far we have not been able to get anything along these lines to work—we outline our attempts in §6.4. We do believe though that this is an excellent hard test case for Machine Learning methods in pure mathematics. Our implementation [GWY] is intended to make it as straightforward as possible for interested researchers to try Machine Learning techniques on this problem.

2 Recollections

We recall some basic facts about the braid group. A presentation of B_n in terms of the Artin generators is given by

$$B_n = \left\langle \sigma_1, \ldots, \sigma_{n-1} \mid \sigma_i\sigma_j = \sigma_j\sigma_i \text{ if } |i-j| > 1 \text{ and } \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j \text{ if } |i-j| = 1 \right\rangle.$$

[1] see §2 for the definition of Garside length

On these generators, the **(reduced) Burau representation** $\pi_n : B_n \to \mathrm{GL}_{n-1}(\mathbb{Z}[v,v^{-1}])$ is defined as follows:

$$\sigma_1 \mapsto \begin{pmatrix} -v^2 & -v \\ 0 & 1 \end{pmatrix} \oplus I_{n-3}, \quad \sigma_{n-1} \mapsto I_{n-3} \oplus \begin{pmatrix} 1 & 0 \\ -v & -v^2 \end{pmatrix}$$
$$\sigma_i \mapsto I_{i-1} \oplus \begin{pmatrix} 1 & 0 & 0 \\ -v & -v^2 & -v \\ 0 & 0 & 1 \end{pmatrix} \oplus I_{n-i-3}$$

for $1 < i < n-1$. We call $\pi_n(\alpha)$ the **Burau matrix** of $\alpha \in B_n$, and its reduction modulo a prime p, its p-Burau matrix. Note that our presentation of π_n differs slightly from many standard accounts, e.g. the one in [Wik24]. Our presentation is obtained from that one by setting $t = v^2$ and then conjugating the generating matrices by the diagonal matrix with entries $(1, -v^{-1}, v^{-2}, \ldots)$. We prefer our presentation because it arises naturally from the cell representations of the Hecke algebra, but for the purpose of our results the difference is immaterial.[2]

Let S_n denote the symmetric group with simple generators $\Phi = \{s_1, \ldots, s_{n-1}\}$. Let $\ell_C(w)$ denote the Coxeter length of $w \in S_n$, which is the length of a reduced expression for w in terms of the simple generators. For $w \in S_n$ write $\widetilde{w} \in B_n$ for the corresponding positive lift in the braid group. Explicitly, if $w = s_{i_1} s_{i_2} \cdots$ is a reduced expression, then $\widetilde{w} = \sigma_{i_1} \sigma_{i_2} \cdots$. Let $w_0 \in S_n$ denote the longest element, which explicitly is given by

$$w_0 = (s_1 s_2 \cdots s_{n-1})(s_1 s_2 \cdots s_{n-2}) \cdots (s_1 s_2)(s_1).$$

We write $\Delta = \widetilde{w_0}$. Note that

$$\pi_n(\Delta) = (-1)^{n+1} \begin{pmatrix} 0 & & v^n \\ & \iddots & \\ v^n & & 0 \end{pmatrix} \tag{2.1}$$

Let $R(w)$ (respectively $L(w)$) denote the right (respectively left) descent set of $w \in S_n$, which are defined as follows:

$$R(w) = \{s_i \in \Phi \mid \ell_C(ws_i) < \ell_C(w)\}$$
$$L(w) = \{s_i \in \Phi \mid \ell_C(s_i w) < \ell_C(w)\}$$

A braid $\alpha \in B_n$ has a **Garside normal form (GNF)**

$$\alpha = \Delta^d \widetilde{w_1} \cdots \widetilde{w_\ell}, \tag{2.2}$$

where $d \in \mathbb{Z}$, $w_1 \neq w_0$, $w_\ell \neq e$, and for every k, $R(w_k) \supseteq L(w_{k+1})$. This expression is unique, and $\ell_G(\alpha) := \ell$ is the **Garside length** of α. It will be useful below to denote the GNF of α in (2.2) by

[2] In more detail, consider the Jones representation of B_n in the Temperley-Lieb algebra, $J : B_n \to TL_n^\times$. Then π_n is the composition of J with the cell module TL_n^1 consisting of Temperley-Lieb diagrams with n points on the top, $n-2$ points on the bottom and exactly one cup. Our presentation of π_n is given in terms of the basis $f_1, \ldots, f_{n-1}$, where f_i is $(-v)^{i-1}$ times the diagram with the cup connecting points i and $i+1$.

$\alpha = (\Delta^d; w_1, \ldots, w_\ell)$. For more details on the rich theory underlying Garside structures we refer the reader to the comprehensive text [DDG+15].

For $k \leq \ell$ we'll be interested in the associated **Garside prefix** of α: $\Delta^d \widetilde{w_1} \cdots \widetilde{w_k}$. A simple but useful observation: if $u \in S_n$ satisfies $R(w_\ell) \supseteq L(u)$, then the GNF of $\alpha\widetilde{u}$ is obtained by concatenation:

$$\alpha\widetilde{u} = \Delta^d \widetilde{w_1} \cdots \widetilde{w_\ell}\widetilde{u}.$$

We term $\widetilde{u}$ a **Garside suffix** of α.

3 Basic idea

Suppose one can find a statistic on matrices which detects Garside length. In other words, we have a function $\mathsf{p} : \mathrm{GL}_{n-1}(\mathbb{Z}[v, v^{-1}]) \to \mathbb{N}$ such that $\mathsf{p}(\pi_n(\alpha)) = C \cdot \ell_G(\alpha)$ for some constant $C>0$. Then this implies the faithfulness of π_n:

$$\pi_n(\alpha) = \mathrm{Id} \implies \mathsf{p}(\pi_n(\alpha)) = 0 \implies \ell_G(\alpha) = 0 \implies \alpha = \Delta^d$$

for some $d \in \mathbb{Z}$, and $\pi_n(\Delta^d) = \mathrm{Id}$ if and only if $d = 0$. Note that this type of argument is used by Krammer to prove the linearity of the Lawrence-Krammer-Bigelow representation of the braid group [Kra02].

On the other hand, if p is correlated with ℓ_G only for generic braids, then one can try to use p as an ansatz to find kernel elements: braids with surprisingly low p-value might point towards elements in the kernel. We explain both of these situations, beginning with the simple case of the 3-strand Burau representation.

First we define the statistic which we'll be using throughout. Given A, a non-zero $r \times s$ array with entries in Laurent polynomials in v, let $\deg(A)$ be the highest power of v that occurs in an entry of A, and let $\mathrm{val}(A)$ be the lowest power. We set

$$\mathsf{projlen}(A) = \deg(A) - \mathrm{val}(A). \tag{3.1}$$

Note that multiplication of a Burau matrix by Δ does not affect $\mathsf{projlen}$:

$$\mathsf{projlen}(\pi_n(\alpha)) = \mathsf{projlen}(\pi_n(\alpha)\Delta) = \mathsf{projlen}(\Delta\pi_n(\alpha)). \tag{3.2}$$

Proposition 3.3. *For any $\alpha \in B_3$ we have $\mathsf{projlen}(\pi_3(\alpha)) = 2\ell_G(\alpha)$. Hence, the 3-strand Burau representation is faithful.*

Proof. Let $\alpha \in B_3$ have GNF given by (2.2) and let $\pi_3(\alpha) = [c_1, c_2]$, where c_i are the column vectors in the matrix. Then the following trichotomy holds:

1. $\deg(c_1) = \deg(c_2), \mathrm{val}(c_1) = \mathrm{val}(c_2)$, and $\alpha = \Delta^d$ for some $d \in \mathbb{Z}$.
2. $\deg(c_1) > \deg(c_2), \mathrm{val}(c_1) > \mathrm{val}(c_2)$, and $R(\alpha) = \{s_1\}$.
3. $\deg(c_1) < \deg(c_2), \mathrm{val}(c_1) < \mathrm{val}(c_2)$, and $R(\alpha) = \{s_2\}$.

Moreover, in each case $\mathsf{projlen}(\pi_3(\alpha)) = 2\ell_G(\alpha)$.

To see this, we induct on $\ell = \ell_G(\alpha)$. If $\ell = 0$ then we are in the first case of the trichotomy. Note that in this case $0 = \mathsf{projlen}(\pi_3(\alpha)) = 2\ell_G(\alpha)$. Now let $\ell \geq 1$, and suppose that $\alpha' := \Delta^d \widetilde{w_1} \cdots \widetilde{w_{\ell-1}}$ falls into the second case of the trichotomy. Setting $\pi_3(\alpha') = [c_1', c_2']$, we then have that $\deg(c_1') > \deg(c_2'), \mathrm{val}(c_1') > \mathrm{val}(c_2')$, $R(\alpha') = \{s_1\}$, and $\mathsf{projlen}(\pi_3(\alpha')) = 2\ell_G(\alpha')$. The condition $R(\widetilde{w_{\ell-1}}) \supseteq R(\widetilde{w_\ell})$ forces $w_\ell = s_1$ or $w_\ell = s_1 s_2$. If $w_\ell = s_1$ then $[c_1, c_2] = [-v^2 c_1', -v c_1' + c_2']$, and we are in case (2). Otherwise, $w_\ell = s_1 s_2$ and $[c_1, c_2] = [-v c_2', v^3 c_1' - v^2 c_2']$, so we are in case (3). Either way we have the equality $\mathsf{projlen}(\pi_3(\alpha)) = 2\ell_G(\alpha)$. The other cases follow similarly. □

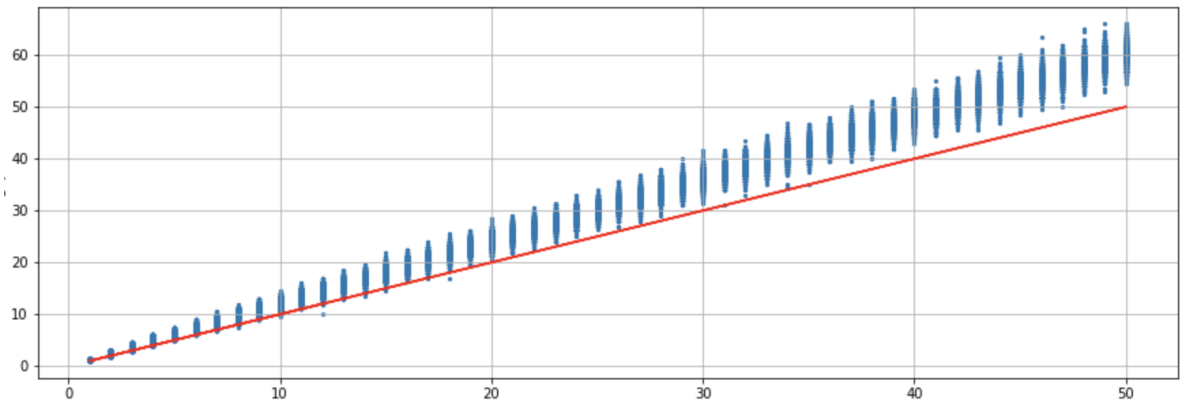
Remark 3.4. *In [CI17] Calvez and Ito also study the relation between the degree and valuation of a Burau matrix and Garside length. They show in particular that for a special class of braids they term "simply-nested", the degree can recover the dual supremum length. Specialised to $n = 3$ this gives another proof of the faithfulness of the 3-strand Burau representation.*

Moving onto the 4-strand braid group, it is easy to see that a straightforward analogue of Proposition 3.3 does not hold. For instance, taking $\alpha = \widetilde{u}$, where $u = s_1 s_2 s_1 s_3$ we have that

$$\pi_4(\alpha) = \begin{pmatrix} 0 & 0 & -v^4 \\ v^3 & v^2 & v^3 \\ 0 & -v & -v^2 \end{pmatrix}$$

We see that $\mathsf{projlen}(\pi_4(\alpha)) = 3$ is not even, and one cannot read off $R(u) = \{s_1, s_3\}$ directly from the columns achieving $\deg(\pi_4(\alpha))$. Nevertheless, we can ask whether generically projlen is a good heuristic for Garside length.

The following table shows the result of sampling 1000 random braids in B_4 of each Garside length, up to 50. Each braid is depicted by a blue dot in the plane, where the x-axis is labelled by the Garside length of the braid, and the y-axis is labelled by $\frac{1}{2}\mathsf{projlen}$. The $y = x$ line is drawn in red.



We see that it is almost always the case that $\frac{1}{2}\mathsf{projlen}$ is larger than the Garside length. Moreover, there appears to be a strong linear correlation between these two statistics for generic braids. However, there are a few exceptions corresponding to blue dots below the red line.

Now consider an element

$$\alpha = \Delta^d \widetilde{w_1} \cdots \widetilde{w_\ell}$$

in the kernel of some Burau representation. Because projlen of a kernel element is zero and projlen is unaffected by multiplication by Δ (3.2) we have

$$0 = \mathsf{projlen}(\alpha) = \mathsf{projlen}(\widetilde{w_1} \cdots \widetilde{w_\ell})$$

It is now reasonable to suspect that the projlens of Garside prefixes of $\widetilde{w_1} \cdots \widetilde{w_\ell}$:

$$\mathsf{projlen}(\widetilde{w_1}), \mathsf{projlen}(\widetilde{w_1}\widetilde{w_2}), \ldots, \mathsf{projlen}(\widetilde{w_1} \cdots \widetilde{w_{\ell-1}}), \mathsf{projlen}(\widetilde{w_1} \cdots \widetilde{w_\ell}) = 0. \tag{3.5}$$

are typically smaller than those of a generic braid of the same length. Thus searching for GNFs with low projlen might point us in the direction of kernel elements.

4 The algorithm

We'll now describe our algorithm for sampling braids with low projlen. We picture the set-up for our search as placing a "bucket" at each point in $\mathbb{Z}_{\geq 0} \times \mathbb{Z}_{\geq 0}$. Initially, these buckets are empty and over time they will be filled with Garside normal forms of 4-strand braids, and their Burau matrices. The bucket $\mathsf{B}_{(\ell,m)}$ at the point (ℓ, m) will only contain braids α such that $\ell_G(\alpha) = \ell$ and $\mathsf{projlen}(\pi(\alpha)) = m$.

The method by which we fill the buckets is called **reservoir sampling** [Vit85]. This is a well-known algorithm which selects a random sample of k elements without repetition from a set of N elements which arrive sequentially, and where the value of N is unknown in advance. Typically, N is enormous. Thus it is not possible to store all elements seen and the sampling is done in one pass. At any point in time, the selection of k elements, i.e. the "reservoir", should be uniformly distributed among all elements seen thus far.

We recall the idea in more detail since it is so fundamental to our approach. Suppose we're sampling elements from a set X and that the elements arrive as a sequence $x_1, x_2, \ldots, x_N$.

Let us first consider the case $k = 1$. Our reservoir then consists of a single element, which at the i-th step we will denote r_i. At the first step we let $r_1 = x_1$. In the second step, we replace r_1 with x_2 with probability $\frac{1}{2}$, otherwise $r_2 = r_1$. In the third step we replace r_2 with x_3 with probability $\frac{1}{3}$, otherwise $r_3 = r_2$. At the i^{th} step we replace r_{i-1} with x_i with probability $1/i$. Note that indeed, for every i, r_i is uniformly distributed among $x_1, \ldots, x_i$.

If $k > 1$, then the reservoir is a k-subset of X, which at the i-th step we will denote $R_i = \{r_1^i, \ldots, r_k^i\}$. First we just fill the reservoir: $r_j^1 = x_j$ for $j \leq k$. At the next step we randomly generate a number $j \in \{1, \ldots, k+1\}$, and if $j \leq k$ replace the j-th element of R_1 by x_{k+1}:

$$r_q^2 := \begin{cases} x_{k+1} \text{ if } q = j, \\ r_q^1 \text{ otherwise.} \end{cases}$$

Otherwise, $R_2 = R_1$. Next, randomly generate a number $j \in \{1, \ldots, k+2\}$, and if $j \leq k$ replace the j-th element of R_2 by x_{k+2}, and so on. At the i-th step, we have that R_i is a uniformly distributed k-subset of $\{x_1, \ldots, x_{i+k-1}\}$.

Going back to our set-up, we will now describe an adaptation of the above ideas to place braids in appropriate buckets. At the ℓ-th step in our algorithm, we will be placing braids in buckets at points (ℓ, m) for various m. We fix ahead of time an upper bound k, which is the maximum number of braids contained in a single bucket. Set $\mathsf{B}_\ell = \bigcup_{m \geq 0} \mathsf{B}_{(\ell,m)}$.

To begin place the identity $e \in B_4$ in $\mathsf{B}_{(0,0)}$. Now let $\ell > 0$, and at the ℓ-th step proceed as follows:

[hbt!] Set $N_m = 0$ for every m Counts how many braids we've tried to place in $\mathsf{B}_{(\ell,m)}$ $\alpha \in \mathsf{B}_{\ell-1}$ $\widetilde{u}$ a Garside suffix of α $m = \mathsf{projlen}(\alpha\widetilde{u})$ $M = |\mathsf{B}_{(\ell,m)}|$ $M < k$ place $\alpha\widetilde{u}$ and its Burau matrix in $\mathsf{B}_{(\ell,m)}$ randomly generate a number $j \in \{1, \ldots, N_m\}$ $j < k$ replace j-th element of $\mathsf{B}_{(\ell,m)}$ by $\alpha\widetilde{u}$ and its Burau matrix discard $\alpha\widetilde{u}$ Set $N_m := N_m + 1$ Note that since we are storing the Burau matrices along the way, it is easy for us to compute $\mathsf{projlen}(\alpha\widetilde{u})$. Also, we can easily modify this algorithm to study the Burau representation modulo primes (we simply remember the p-Burau matrices instead).

A critical and subtle aspect of the above algorithm is the choice of bucket size k. Roughly speaking, the larger the bucket size k, the more extensive the search, so larger k are typically better. In practice, we run the algorithm a few times to get an idea of memory usage, and then increase k to be as large as possible for available memory.

5 Main result

We'll now describe the most interesting output of our algorithm: in the case $p = 5$ we obtained several elements in the kernel of $\pi \otimes \mathbb{F}_5$. We stress that these are the first known kernel elements in Burau representations $\pi \otimes \mathbb{F}_p$ with $p > 3$, and this represents the first explicit progress on this question in 25 years!

The smallest we found, κ, has Garside length 54. We have

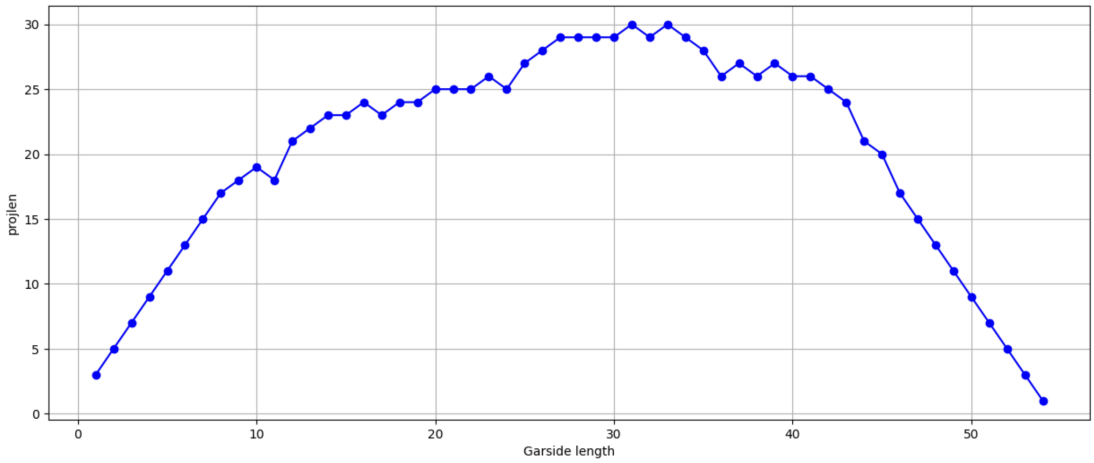
$$\kappa = \Delta^{-27}\alpha$$

where Δ denotes the Garside element and α is the following word in the Artin generators (here i corresponds to σ_i):

$$1,3,1,3,1,3,2,2,1,3,3,2,2,1,3,3,2,2,2,1,3,1,3,2,1,2,1,3,1,3,2,1,2,1,3,1,3,2,2,2,1,3,3,$$
$$2,2,1,3,3,2,2,1,3,1,3,1,2,3,2,1,1,3,2,1,2,1,3,1,3,2,1,2,1,3,1,3,2,2,1,3,2,2,1,3,2,2,2,1,$$
$$3,3,2,2,1,3,3,2,2,1,3,1,2,3,2,1,1,3,2,1,2,1,3,1,3,2,1,2,1,3,1,2,3,2,1,1,3,2,2,1,3,3,2,$$
$$2,1,3,3,2,2,2,1,3,2,2,1,3,1,2,3,2,2,1,3,1,2,3,2,2,1,3,1,2,3,2,1$$

The length of α in the Artin generators is 162.

The following plot depicts the journey of α through the various buckets, as described in (3.5):



The x-axis is labelled by Garside length, the y-axis by projlen[3], and each blue dot represents a bucket which contains a Garside prefix of κ.

The rightmost dot corresponds to the bucket containing α. It's interesting to note that this trajectory follows a generic path until about Garside length 30, where suddenly the reservoir sampling starts finding elements with smaller than expected projlen. Around length 40, the algorithm hits a "point of no return", and makes a beeline for α.

In case the reader is interested, here is the Garside normal form of κ:

$$\kappa = (\Delta^{-27}; s_1s_3, s_1s_3, s_1s_3s_2, s_2s_1s_3, s_3s_2, s_2s_1s_3, s_3s_2, s_2, s_2s_1s_3, s_1s_3s_2s_1, s_2s_1s_3,$$
$$s_1s_3s_2s_1, s_2s_1s_3, s_1s_3s_2, s_2, s_2s_1s_3, s_3s_2, s_2s_1s_3, s_3s_2, s_2s_1s_3, s_1s_3, s_1s_2s_3s_2s_1, s_1s_3s_2s_1,$$
$$s_2s_1s_3, s_1s_3s_2s_1, s_2s_1s_3, s_1s_3s_2, s_2s_1s_3s_2, s_2s_1s_3s_2, s_2, s_2s_1s_3, s_3s_2, s_2s_1s_3, s_3s_2, s_2s_1s_3,$$
$$s_1s_2s_3s_2s_1, s_1s_3s_2s_1, s_2s_1s_3, s_1s_3s_2s_1, s_2s_1s_3, s_1s_2s_3s_2s_1, s_1s_3s_2, s_2s_1s_3, s_3s_2,$$
$$s_2s_1s_3, s_3s_2, s_2, s_2s_1s_3s_2, s_2s_1s_3, s_1s_2s_3s_2, s_2s_1s_3, s_1s_2s_3s_2, s_2s_1s_3, s_1s_2s_3s_2s_1)$$

Since randomness is built into the algorithm, every time we run it we get different outcomes. About 20% of the time we actually find a kernel element when using a bucket size of 15K, and this takes under 10 minutes on a standard personal computer (see [GWY]).

Other runs of our algorithm discovered two other elements in the kernel of Garside lengths 59 and 65 respectively. They are

$$\kappa_1 = \Delta^{-29}\alpha_1 \quad \text{and} \quad \kappa_2 = \Delta^{-33}\alpha_2$$

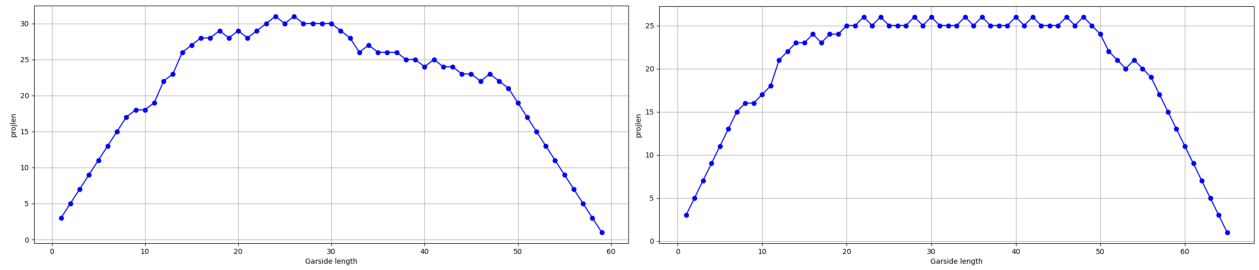
where α_1 and α_2 are given by the following words in the Artin generators, of lengths 174 and 198

[3]It is actually labelled by projlen + 1 but we ignore this technicality.

respectively:

$$\alpha_1 = (1,2,1,3,2,2,1,3,1,3,2,2,2,1,3,1,2,2,1,3,1,2,2,1,3,1,3,1,2,3,2,1,1,2,3,2,2,1,3,1,2,3,2,2,1,3,1,3,2,2,2,1,3,1,2,2,1,3,1,$$
$$2,2,2,1,3,1,2,3,2,2,1,3,1,2,3,2,2,1,3,1,2,3,2,1,1,3,1,3,1,3,2,2,1,3,1,2,2,1,3,1,2,2,2,1,3,2,2,1,3,1,3,2,1,2,1,3,1,3,2,1,2,1,$$
$$3,1,2,3,2,1,1,3,2,2,1,3,3,2,2,1,3,3,2,2,1,3,1,2,3,2,1,1,3,2,1,2,1,3,1,3,2,1,2,1,3,1,3,2,2,1,3,2,2,2,1,3,1)$$
$$\alpha_2 = (1,3,1,3,1,3,2,2,1,3,3,2,2,1,3,3,2,2,1,3,1,2,3,2,1,1,3,2,1,2,1,3,1,3,2,1,2,1,3,1,3,2,2,2,1,3,3,2,2,1,3,3,2,2,1,3,1,2,3,2,$$
$$1,1,3,2,1,2,1,3,1,3,2,1,2,1,3,1,3,2,2,2,1,3,3,2,2,1,3,3,2,2,1,3,1,2,3,2,1,1,3,2,1,2,1,3,1,3,2,1,2,1,3,1,3,2,2,2,1,3,3,2,2,1,3,3,2,2,1,3,1,$$
$$2,3,2,1,1,3,2,1,2,1,3,1,3,2,1,2,1,3,1,3,2,2,2,1,3,3,2,2,1,3,3,2,2,1,3,1,2,3,2,1,1,3,2,1,2,1,3,1,3,2,1,2,1,3,1,3,2,2,1,3,2,2,1,3,2,2,1,3,2)$$

Here are the trajectories of α_1 and α_2 through the buckets:



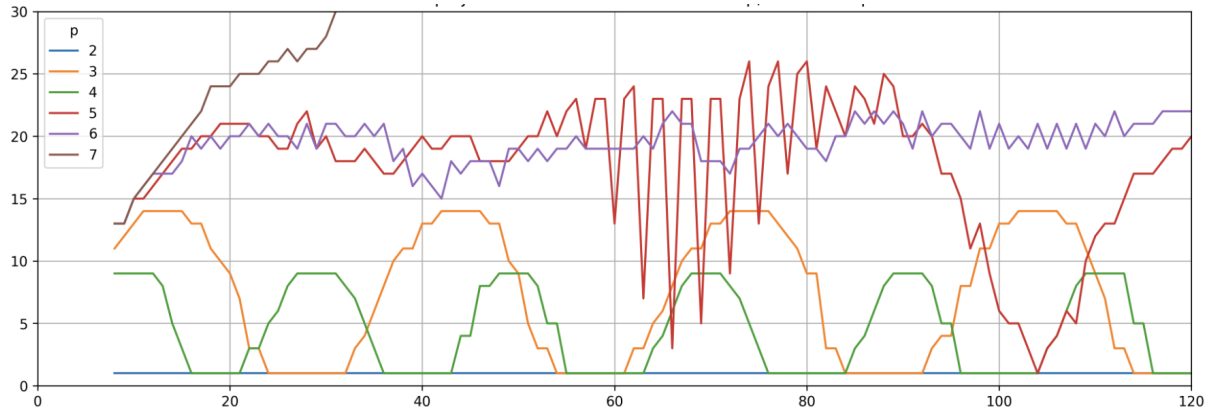
Remark 5.1. *The reader wishing to explore more can experiment with the notebook "p=5 kernel elements" available at [GWY]. We have also implemented a check of our results in Magma in the file "MagmaCheck.m" which is also available at [GWY]. It should be easy to modify the Magma file to suit any computer algebra system.*

6 How good is our algorithm?

We regard the fact that our algorithm is able to discover a kernel element nearly 25 years after the last discovery of a kernel element (for $p = 3$ in [CL98]) as some evidence that our approach is interesting. On the other hand, it is interesting to ask: can our algorithm rediscover known kernel elements?

6.1 Four strand braids

Here we see a plot Garside of length (x-axis) against the minimal projlens found (y-axis), over several runs of our algorithms modulo various integers $m = 2,3,4,5,6,7$:



It immediately finds many kernel elements modulo 2 and 3. It also finds kernel elements modulo 4. Note also the jagged red line: during these runs the algorithm found an element of Garside length around 65 with low projlen, but did not find a kernel element of this length. It did, however find a kernel element of Garside length around 105 (where the red line finally touches the x-axis). Finally, note that although we know that kernel elements exist modulo 6 (indeed, take the commutator of a kernel element modulo $p = 2$ and $p = 3$!), our algorithm didn't find them. (The purple line.)

6.2 Six strand braids

We now discuss the case of the Burau representation on 6 strands, where our algorithm was able to discover a non-trivial kernel element. This is an interesting case, as here a kernel element is known (by the work of Bigelow), and we gain some insight by asking: how "close" we are to finding this kernel element.

It turns out that there is a beautiful and simple idea that allows us to accurately answer this question. For concreteness, let us take $n = 6$ in which case Bigelow's kernel element is of Garside length 16:

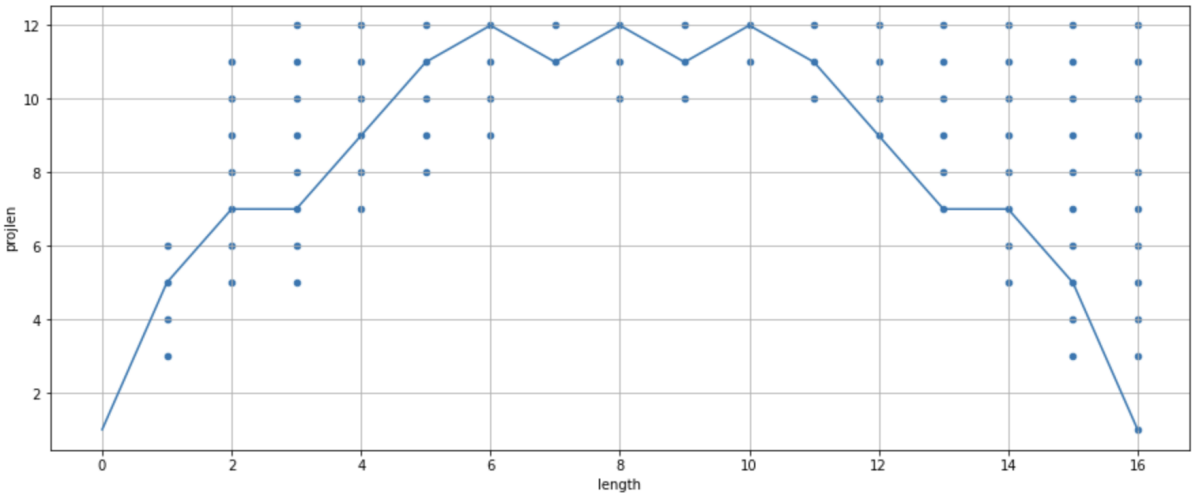
$$\beta = \Delta^d \widetilde{w_1}\widetilde{w_2}\cdots\widetilde{w_{16}}.$$

Now we can modify our code slightly to force it to add each of the Garside divisors

$$\beta_1 = \widetilde{w_1}, \quad \beta_2 = \widetilde{w_1}\widetilde{w_2}, \quad \ldots, \quad \beta_{16} = \widetilde{w_1}\widetilde{w_2}\cdots\widetilde{w_{16}}$$

to their appropriate buckets during the reservoir search. In other words, we peform the reservoir sampling

as usual, however we force it to successfully find β:



Now for each reservoir containing β_i we can compute the probability that a random sample would contain β_i. If we denote this probability by $P(\beta_i)$, then

$$P(\beta) = \prod_{i=1}^{16} P(\beta_i) = \prod_{i=1}^{16} \frac{k}{\max(r(\beta_i), k)}$$

where k denotes the bucket size as above, and $r(\beta_i)$ denotes the total number of elements seen by the bucket containing β_i.

With $k = 500$ this probability is smaller than 10^{-6}. Increasing the bucket size to 10^4 increased this probability to around 0.0005. At this point it is believable that running on a slightly larger computer might find this element. This turned out to be the case: increasing the bucket size to $4 \cdot 10^4$ resulted in the discovery of a kernel element of Garside length 28, after four hours of search [GWY]. (Note that we are as yet unable to rediscover Bigelow's kernel element of length 16, however the above heuristics suggest this should be possible with a further mild increase in bucket size.)

6.3 Five strand braids

A similar analysis for $n = 5$ shows that this case is considerably harder—with a bucket size of 500 the chance of finding Bigelow's kernel element is of the order of 10^{-30}! We have not looked closely at how this probability changes when using considerably bigger buckets, however it seems likely that a new idea is needed.

6.4 Future Directions

We briefly comment on three avenues which we believe warrant further exploration.

6.4.1 Monte Carlo methods

The task of searching through Garside normal forms in order to find kernel elements is a textbook example of tree search. This problem features prominently in computer Chess and Go. In Go, major progress was made in the 1990s and 2000s by employing Monte Carlo evaluation strategies [BH04]: in order to decide the strength of a board position, one randomly finishes the game 5000 times and sees how often one wins. We tried a similar method here:

1. A database D of promising nodes (with score) is initiated with the identity element.
2. At each step, α is sampled from D (randomly with weighting based on the score). We then do reservoir sampling starting from each Garside suffix $\alpha\widetilde{u}$ for N steps. Each suffix is then added to D, with score based on the lowest projlen seen.

(Thus, we regard exploring the tree of all possible Garside normal forms as a one-player game, where the goal is to find kernel elements, and a "move" consists of replacing the current element by a Garside successor.)

We implemented this algorithm and it consistently found kernel elements for $p = 5$. We did several long runs in other settings (e.g. $(n = 4, p = 7)$, $(n = 5, p = 41)$ etc.) without finding kernel elements. In this setting there are several design choices (e.g. how to implement score and select based upon it, how to choose N, ...) which have a big impact on performance. This is worth investigating systematically and could lead to a much better algorithm.

6.4.2 Machine learning methods

We begun this project with the aim of employing machine learning methods. We outline here one unsuccessful attempt to use vanilla supervised learning to improve our sampling. Many other experiments are possible, and we hope that our software can provide a good basis for further experiments.

Consider all elements of Garside length ℓ. Imagine an oracle O which takes as input the matrix of an element of Garside length ℓ and tells us the minimum projlen amongst all Garside successors of Garside length $\ell + k$ for some value of k (e.g. $k = 5$). Then it seems intuitive that using the oracle O to weight reservoir sampling (i.e. elements with low projlen in k further steps are more likely to be kept in a bucket) would lead to better results.

With this motivation in mind, we tried to train a vanilla neural network which takes as input the matrix of α, and attempts to predict the minimal projlen in k steps time. Our models attained reasonable training accuracy ($\sim 80\%$), but seemed to make no difference at all to the long-term performance of the algorithm.[4] In other words, adding the neural network made no discernable difference to the smallest projlen found for large Garside lengths. Thus, our attempt to use neural networks to spot patterns in

[4]Some further details on our algorithm: We encode a matrix M of Laurent polynomials as a list of matrices encoding the coefficients of v^i. Because we only care about our matrices up to scalar multiple, we can assume that the non-zero matrices range from 0 to the projlen: $M_0, M_1, \ldots, M_{\mathsf{projlen}(M)}$. Now, after multiplying M by all Garside successors of length k, it is easy to see that the resulting projlens only depends on the first and last k' matrices in our list, i.e. on $M_0, M_1, \ldots, M_{k'}, M_{\mathsf{projlen}(M)-k'+1}, \ldots M_{\mathsf{projlen}(M)}$ for some small value of k'. We further simplified our dataset by remembering only $M'_0, M'_1, \ldots, M'_{k'}, M'_{\mathsf{projlen}(M)-k'+1}, \ldots M'_{\mathsf{projlen}(M)}$ where M'_i is a zero-one matrix whose entries record which entries in M'_i are non-zero. This list was then serialized and used as input to a vanilla neural network.

Burau matrices was unsuccessful. This suggests that such patterns are either not there, or are difficult to detect.[5]

6.4.3 Commutators

It is striking that all kernel elements found in [CL97, CL98, Big99] are commutators. It is tempting to try to modify our search regime to use this fact as a prior. It is not clear how to do so, but a reasonable proposal could lead to better results. We have not tried to express the elements that we have found as (close relatives of) commutators.

Acknowledgments

The observation that Garside length is closely correlated with projlen (cf. (3.1)) was made in joint work of the third author with Nicolas Libedinsky, and related ideas appeared earlier in work of Calvez and Ito [CI17] (cf. Remark 3.4). We would like to thank Robert Bryant, Giles Gardem, Nicolas Libedinsky and Nolan Wallach for useful discussions, and an anonymous referee for greatly improving our exposition.

References

[BH04] Bruno Bouzy and Bernard Helmstetter. Monte-carlo go developments. *Advances in Computer Games: Many Games, Many Challenges*, pages 159–174, 2004. 47

[Big99] Stephen Bigelow. The Burau representation is not faithful for $n = 5$. *Geom. Topol.*, 3:397–404, 1999. 36, 48

[Big02] Stephen Bigelow. Does the Jones polynomial detect the unknot? volume 11, pages 493–505. 2002. Knots 2000 Korea, Vol. 2 (Yongpyong). 36

[Bir74] Joan S. Birman. *Braids, links, and mapping class groups*, volume No. 82 of *Annals of Mathematics Studies*. Princeton University Press, Princeton, NJ; University of Tokyo Press, Tokyo, 1974. 36

[CI17] Matthieu Calvez and Tetsuya Ito. A Garside-theoretic analysis of the Burau representations. *J. Knot Theory Ramifications*, 26(7), 2017. 40, 48

[CL97] D. Cooper and D. D. Long. A presentation for the image of Burau$(4) \otimes Z_2$. *Invent. Math.*, 127(3):535–570, 1997. 36, 37, 48

[CL98] D. Cooper and D. D. Long. On the Burau representation modulo a small prime. In *The Epstein birthday schrift*, volume 1 of *Geom. Topol. Monogr.*, pages 127–138. Geom. Topol. Publ., Coventry, 1998. 37, 44, 48

[5] One can also imagine that a drop in projlen is a rare event, which is too rare to be picked up by the neural network.

[Dat] Amitesh Datta. A strong characterization of the entries of the Burau matrices of 4-braids: The Burau representation of the braid group b_4 is faithful almost everywhere. 37

[DDG+15] Patrick Dehornoy, François Digne, Eddy Godelle, Daan Krammer, and Jean Michel. *Foundations of Garside theory*, volume 22 of *EMS Tracts in Mathematics*. European Mathematical Society (EMS), Zürich, 2015. 39

[GWY] Joel Gibson, Geordie Williamson, and Oded Yacobi. Software for braids and their quantum representations. https://github.com/geordw/braids-project. Accessed: 2023-10-03. 37, 43, 44, 46

[Ito15] Tetsuya Ito. A kernel of a braid group representation yields a knot with trivial knot polynomials. *Math. Z.*, 280(1-2):347–353, 2015. 36

[Kra02] Daan Krammer. Braid groups are linear. *Ann. of Math. (2)*, 155(1):131–156, 2002. 39

[LP93] D. D. Long and M. Paton. The Burau representation is not faithful for $n \geq 6$. *Topology*, 32(2):439–447, 1993. 36

[Moo91] John Atwell Moody. The Burau representation of the braid group B_n is unfaithful for large n. *Bull. Amer. Math. Soc. (N.S.)*, 25(2):379–384, 1991. 36

[Vit85] Jeffrey Scott Vitter. Random sampling with a reservoir. *ACM Trans. Math. Software*, 11(1):37–57, 1985. 41

[Wik24] Wikipedia contributors. Burau representation, 2024. [Online; accessed 17-April-2024]. 38

AUTHORS

Joel Gibson

Sydney Mathematical Research Institute
University of Sydney
Sydney, Australia

joel@jgibson.id.au
https://www.jgibson.id.au/

Geordie Williamson

School of Mathematics and Statistics
University of Sydney
Australia

g.williamson@sydney.edu.au
https://www.maths.usyd.edu.au/u/geordie/

Oded Yacobi

School of Mathematics and Statistics
University of Sydney
Australia

oded.yacobi@sydney.edu.au
https://www.maths.usyd.edu.au/u/oyacobi/