Braid groups in cryptography, untangling their viability.

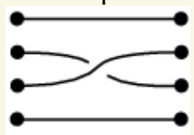
Overview

In general, a cryptographic scheme aims to secure data between two parties who wish to share information amongst each other within an insecure channel. To do this we usually employ a function that is easy to compute its self, but is hard to compute in its inverse without knowing a certain piece of information, referred to as the secret key. The vast majority of modern algorithms (AES, RSA, etc.) use the discrete logarithm problem in the integers (a commutative group) as the basis to formulate their functions.

Non-commutative groups present a different type of hard problem known as the conjugacy search problem. It is analogous to the discrete logarithm problem in the sense that conjugacy and exponentiation share similar properties.

The braid group, which consists of n strands each connecting a pair of dots, presents an interesting candidate for a non-commutative group in which to implement a cryptographic scheme. It has a clear identity element, a well-defined operation, and two well studied normal form presentations, Artin and Garside. We investigate the difficulty in solving the conjugacy search problem using the Artin presentation and super-summit sets. We hope to shed some light on the frequency of the exponential bound when computing super-summit sets and to what degree it cuts down on computation time.

Figure 1: A simple braid in B4



Methodology

We implement the conjugacy search algorithm outlined by Elrifai & Morton using Python.

We modify an existing implementation of the braid group as a data structure, clearly defining its operation and inverse. Each braid becomes stored in it's left canonical band-generator normal form. This ensures the B_n set for the chosen n is represented in a finite amount of words and enables us to use various properties of this representation in modeling our algorithm. We use a generator element, δ and model each braid as a power of δ and a number of canonical band permutations, noted A_i.

$$W = \delta^u A_1 A_2 ... A_k$$

We now seek to find an element of the super-summit-set of both braids we wish to compare with the tightest bound. That is to say in band-generator representation, the difference between the power of delta and the amount of positive band permutations becomes the smallest.

Once these two elements are found we use the theorem outlined by Elrifai and Morton stating the existence of a sequence of braids such that each one is conjugate to the next by a factor in [0,1] from the first summit element to the next. We algorithmically compute all possible chains of conjugates using a list data structure to keep track of this. If we find that the second summit-set element at any point we end our search concluding the elements are conjugate as they are part of the same summit-set. If not we continue until we have exhausted all possible conjugate elements. This step becomes potentially exponentially complex and is the source of our analysis. We repeat our experiment for 1000 randomly generated pairs of braids and graphically analyze the results for the groups of 3, 4 and 5 strands.

Results and further analysis

Figure 2: Time in seconds after 1000 samples

n	Average	Min	Max	Conjugates
3	$1.20 * 10^{-3}$	$8.11 * 10^{-5}$	$1.27 * 10^{-2}$	10.6%
4	$5.81 * 10^{-3}$	$1.80 * 10^{-4}$	$2.76 * 10^{-1}$	5.1%
5	$5.27 * 10^{-2}$	$1.01 * 10^{-4}$	6.3	4.4%

We note the rapid growth in the maximum value for each n. The computation was only feasible up to B_5 in this project, and the algorithm was originally formulated only to be effective up to B_6. Nonetheless the exponential trend is beginning to become evident in these results, however the deviation between minimum and maximum results is surprising. In the future we would like to optimize the algorithm further and run it on larger values of n.

References

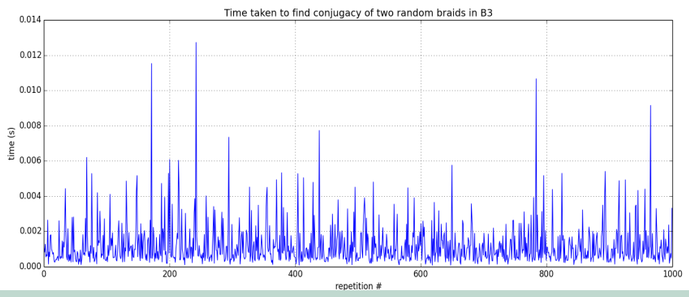
1. Elrifai, E.A., & Morton H.R. (1991). Algorithms for positive braids
2. Cha, J.C., et al. (2001). An Efficient Implementation of Braid Groups, *Advances in Cryptography, ASIACRYPT*

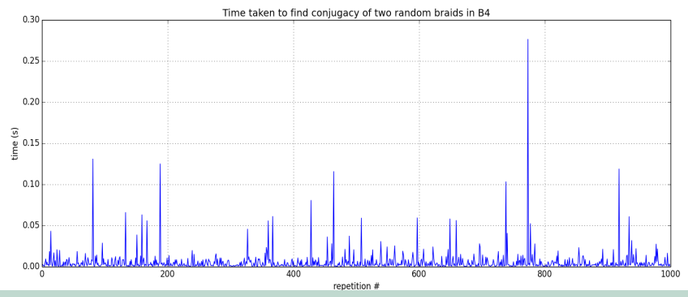
Acknowledgements

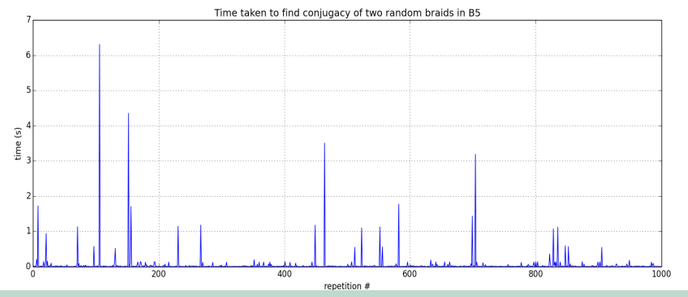
- Hadi Salmasian for his invaluable guidance and wisdom.
- Andrew Geng for creating the code which this project builds on.
- University of Ottawa and UROP for funding this research.

The code used will be published at https://github.com/obenn/braids

Trials







Oliver Benning
obenn009@uottawa.ca
University of Ottawa

uOttawa