Biased Tadpoles: a Fast Algorithm for Centralizers in Large Matrix Groups

Daniel Kunkle
Gene Cooperman

Northeastern University
Boston, USA 02115

What is a Tadpole?

A tadpole provides a way to find a canonical element of a set typically in time asymptotically proportional to the square root of the size of the set. The ability to find a canonical element of a set is useful in several areas of computational group theory. We will return and give a formal definition of tadpoles after a small amount of background.

What is a Group?

A group G is a set with an associative multiplication operation, an identity element e, and an inverse operation $\cdot^{-1}$ such that $gg^{-1} = e$ for all elements $g \in G$.

However, of greater interest for this talk is the representations of a group. Permutations and matrices both have natural identities and inverses:

- If G is a permutation group acting on a set of points Ω, then we have an *action* function: $f\colon \Omega \times G \longrightarrow \Omega$. Further, for fixed $g \in G$, $x \mapsto f(x,g)$ is a permutation of Ω. This action function is usually not written explicitly. Instead, we write $x^g = y$ for $x, y \in \Omega$, $g \in G$.
- If G is a finite matrix group acting on V: vectors over a finite field (for examples all n-dimensional vectors over $GF(2)$, then we have an *action* function: $f\colon V \times G \longrightarrow V$. Further, for fixed $g \in G$ (g a matrix acting on V), $v \mapsto f(v,g)$ is a linear function. This action function is usually not written explicitly. Instead, we write $vg = w$ for $v, w \in V$, $g \in G$.

Permutation and Matrix Representations

Permutation groups have a well developed body of algorithms, often operating in polynomial time. An exception to that is that no polynomial time centralizer algorithm is known. However, there are good heuristics for centralizers for permutations.

Matrix groups are harder to compute with, and fewer polynomial time algorithms are known. Centralizers for matrix groups have few good heuristics outside of the case of GF(2) (the finite field with two elements).

What are Centralizers and Conjugacy Classes?

The *centralizer* in G of an element $g \in G$, $\mathcal{C}_G(g)$, is the set of elements of G that commute with G: $\mathcal{C}_G(g) = \{h \in G \colon hg = gh\}$.

The *conjugate* of $g \in G$ by $h \in G$ is $g^h = h^{-1}gh$. The *conjugacy class* g^G of $g \in G$ is $g^G = \{g^h \colon h \in G\}$.

This provides another representation of a group *in its conjugate action*.

- If G is a finite group, then the group G acts on the conjugacy class g^G for some fixed $g \in G$. The *action* function is: $f \colon g^G \times G \to g^G$.

It is well known that $|g^G||\mathcal{C}_G(g)| = |G|$. A closely related fact is that if w is a word in elements of G such that $g^w = g$, then $w \in \mathcal{C}_G(g)$.

We will be interested in finding $\mathcal{C}_G(g)$ in the case that G is a matrix group. We will do so by finding words w such that $g^w = g$. In the paper, it is shown that the words w (when multiplied out) are random elements of $\mathcal{C}_G(g)$. The expected number of random elements to generate a group is $O(\log|G|)$. So, relatively few such words w will suffice to generate $\mathcal{C}_G(g)$.

What is a Tadpole?

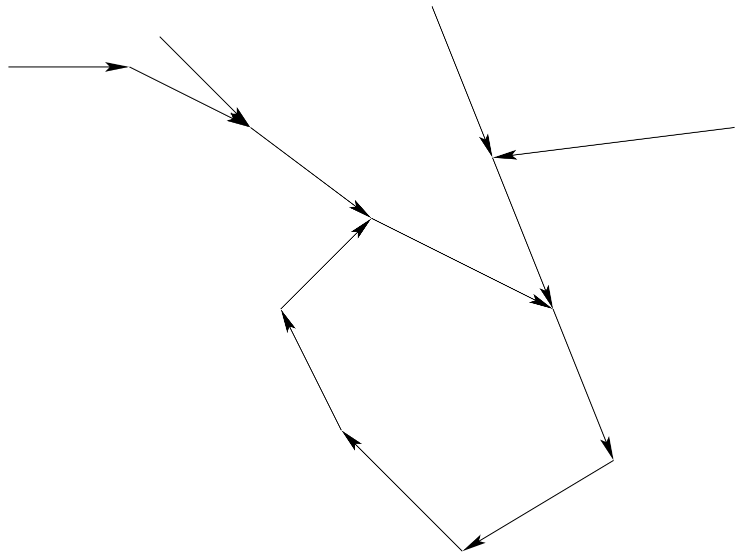
A tadpole walk on a permutation domain and group generators is a construction in which one takes a pseudo-random, but deterministic function f (such as a hash function) that acts on the permutation domain and returns an element of the group. One then views this as a dynamical system on the points of the permutation domain. Given a point x_0, one computes the new point $x_1 = {x_0}^{f(x_0)}$. Similarly, let $x_2 = {x_1}^{f(x_1)}$. As one iterates through $x_0, x_1, x_2, \ldots$, one arrives at an attractor cycle. One can choose a canonical element on that attractor cycle (for example the lexically least element).

First Idea for Centralizer: Consider the conjugacy class g^G as the permutation domain. Start at canonical element x' of some attractor cycle. Take a random element $h \in G$ (this is efficient for matrix groups). Let $x_0 = {x'}^h$. Then follow an iteration over a tadpole. That iteration will lead back to a canonical element of an attractor cycle. (We show this is often the original element x', and we can guarantee this with a small patch.)

This produces a word: $w = hf(x_0)f(x_1)\ldots$ That word w is in the centralizer of g.

Tadpole in Pictures

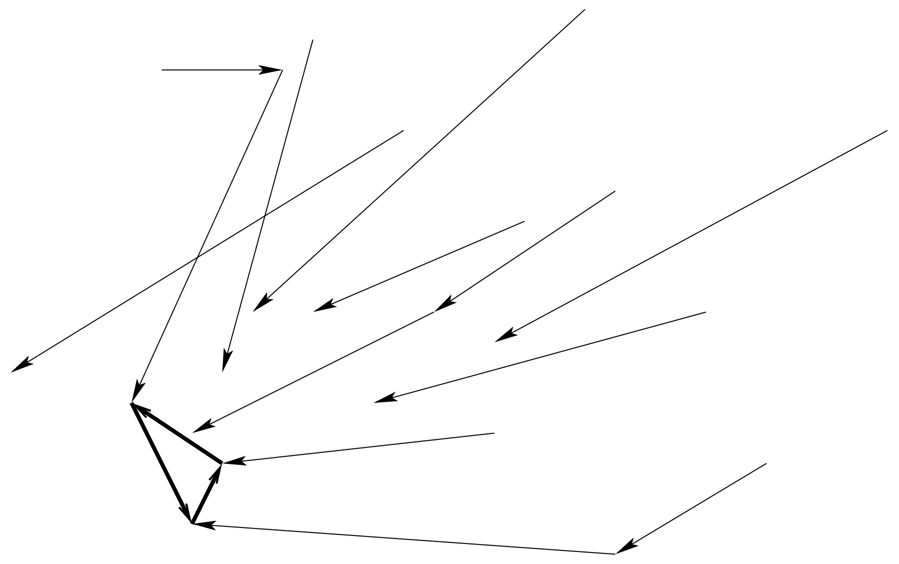
Tadpole:



For n the number of states, it was shown that the expected number of cycles is $(1/2)\log_2 n$. See: G. Cooperman and M. Tselman. Using tadpoles to reduce memory and communication requirements for exhaustive, breadth-first search using distributed computers. In *Proc. of ACM Symposium on Parallel Architectures and Algorithms* (SPAA-97), pages 231–238. ACM Press, 1997.

Key Idea: Biased Tadpole

Idea of Biased Tadpole: Use a biased function f' such that $x^{f'(x)}$ is *usually* in a small subset of the full range. Then the biased tadpole looks as below, and the length of a tadpole walk along a biased tadpole is much shorter. **Biased Tadpole:**



(In the paper, we achieve biasing for J_4 (acting on $GF(2)$ over dimension 112) of one in 100. This makes tadpole walks $\sqrt{100} = 10$ times shorter.)

Biased Tadpole for Centralizer

- One step in the tadpole walk consists of taking a matrix M and conjugating by an element $g \in J_4$: $g^{-1}Mg$.
- We choose a generating set S of size 100 (much more than the two original Generators).
- Choose $g_1 = f(M)$
- Multiply by an element g_2, one of the 100 generatorS, to compute $g_2^{-1}g_1^{-1}Mg_1g_2$ ($g = g_1g_2$)
- Choose g_2 such that $g_2^{-1}g_1^{-1}Mg_1g_2$ is lexically least among all possible choices of g_2. (Decide lexically least by looking at image of a fixed basis vector v. Hence $vg_2^{-1}g_1^{-1}Mg_1g_2$ is minimized.)
- Space of matrices $M' = g_2^{-1}g_1^{-1}Mg_1g_2$ is reduced by a factor of approximately 100. This implies a reduction in the length of the tadpole walk of approximately $\sqrt{100} = 10$. Hence, a 10-times speedup.

Centralizer: Experimental Results

Table 1: Results of computing the centralizer of an involution in twelve different matrix groups (* indicates that GAP exceeded its maximum of 4 GB of RAM without completing).

Group	Involution Word	Centralizer Size	Our Centralizer Time (s)	Our Order Time (s)	GAP Centralizer and Order Time (s)
J_3	a	1920	17.78	0.90	*
McL	a	40320	4.64	0.69	18.12
He	a	161280	7.11	1.17	1010.06
A_{14}	$(ba)^6$	46080	13.39	1.14	1.15
$2^{1+8}.O_8^+(2)$	a	49152	15.11	2.26	*
Ru	a	116480	7.78	1.16	*
Co_3	bb	2903040	8.05	1.32	11.86
Co_2	a	743178240	6.91	1.66	1011.21
Fi_{22}	a	18393661440	15.74	5.69	*
$F_4(2)$	a	754974720	20.49	66.84	*
Co_1	a	2012774400	86.56	43.20	*
$E_6(2)$	a	135291469824	1643.73	11.87	*